***НОВ БЪЛГАРСКИ УНИВЕРСИТЕТ***

**Матрични трансформации – интерактивен самоучител**

***Дипломна работа за получаване на образователната степен бакалавър по информатика***

Йордан Милков Костадинов, F76814

учебна година 2020/2021

Съдържание:

[Увод 5](#_Toc73063286)

[Математически дефиниции 8](#_Toc73063287)

[Вектор 8](#_Toc73063288)

[Координатни системи. 8](#_Toc73063289)

[Декартова 9](#_Toc73063290)

[Полярна координатна система (в 2D) 9](#_Toc73063291)

[Цилиндрична координатна система 10](#_Toc73063292)

[Сферична координатна система 11](#_Toc73063293)

[Матрици 11](#_Toc73063294)

[Дефиниции 11](#_Toc73063295)

[Умножение на матрици 13](#_Toc73063296)

[Свойства на умножението на матрици 14](#_Toc73063297)

[Обратни матрици 14](#_Toc73063298)

[Трансформации 16](#_Toc73063299)

[Матриците като трансформации 16](#_Toc73063300)

[Афинни трансформации 16](#_Toc73063301)

[Скалиране 16](#_Toc73063302)

[Ротация 17](#_Toc73063303)

[Ойлерови ъгли от ротационна трансформация: 18](#_Toc73063304)

[Транслация 19](#_Toc73063305)

[Скалиране и ротация с хомогенна координата система: 19](#_Toc73063306)

[Наклоняване (shearing) 20](#_Toc73063307)

[Отражение 21](#_Toc73063308)

[Неафинни трансформации 21](#_Toc73063309)

[Заостряне (tapering) 21](#_Toc73063310)

[Композиция от трансформации 24](#_Toc73063311)

[Техническа информация 25](#_Toc73063312)

[Система за контрол на версиите 25](#_Toc73063313)

[Използвани технологии 25](#_Toc73063314)

[Архитектура 25](#_Toc73063315)

[Vectorworks като приложение 25](#_Toc73063316)

[VCOM обекти и комуникация между тях 25](#_Toc73063317)

[Основни VCOM обекти 26](#_Toc73063318)

[Организация на Vectorworks документа 27](#_Toc73063319)

[Трансформации при рисуване на геометрични обекти на екрана 29](#_Toc73063320)

[Основни библиотечни класове във Vectorworks SDK 29](#_Toc73063321)

[Организация на модален диалог 29](#_Toc73063322)

[Имплементация 31](#_Toc73063323)

[Входна функция plugin\_module\_main 31](#_Toc73063324)

[Разширение *Menu Transform Matrix* 31](#_Toc73063325)

[Диалог *Transform Matrix* 32](#_Toc73063326)

[Стартиране на диалога 32](#_Toc73063327)

[Динамично свързани данни 32](#_Toc73063328)

[Обработка на събития 33](#_Toc73063329)

[Екран с интерактивен изглед 34](#_Toc73063330)

[Клас за операции с матрици и композиране на трансформации 35](#_Toc73063331)

[Прилагане на резултата върху обектите в документа 36](#_Toc73063332)

[Разширение *Pick Object Tool* 37](#_Toc73063333)

[Стариране на тула 37](#_Toc73063334)

[Дефиниране на разширението Pick Object Tool 37](#_Toc73063335)

[Обработка на събитията на Pick Object Tool 38](#_Toc73063336)

[Компилиране 38](#_Toc73063337)

[Инсталиране: 38](#_Toc73063338)

[Потреибителски интерфейс 40](#_Toc73063339)

[Generate Transformation 42](#_Toc73063340)

[Transform/Result група 43](#_Toc73063341)

[Rotate 43](#_Toc73063342)

[Translate 43](#_Toc73063343)

[Scale 44](#_Toc73063344)

[Euler Angle Rotation 45](#_Toc73063345)

[Object Matrix 45](#_Toc73063346)

[Result 46](#_Toc73063347)

[Invert 47](#_Toc73063348)

[Matrix View 47](#_Toc73063349)

[Preview group 47](#_Toc73063350)

[Прилагане на трансформацията на обектите в модела 50](#_Toc73063351)

[Заключение: 51](#_Toc73063352)

[Използвана литература: 52](#_Toc73063353)

# Увод

Векторите и матричните трансформации са най-широко използваният метод за моделиране на геометрия в областта на компютърната графика. Заедно с координатните системи, които те дефинират, са в основата на всяко приложение за проектиране и 3D моделиране, както и на видео игрите. Чрез тях фигури, дефинирани с точки или математически функции, могат да се комбинират, преоразмеряват и огъват. Матрици отговарят и за промяната на позицията на камерата в играта, проекцията на пейзажа, а така също и за движението на играча и обектите.

За да бъде осмислена и приложена теорията на матричните трансформации, може да бъде създадена приложна програма, която да използва такива трансформации. Курсовете по линейна алгебра, аналитична геометрия и графично програмиране предоставят пълния набор от знания, необходими за създаване на програма за 3D визуализиране и трансформиране на обект. За съжаление, създаването на приложение от нулата е времеемко, а резултатът обикновено е с ограничена функционалност. Съществуват и различни проекти в GitHub, които могат да бъдат използвани за създаване, композиране и прилагане на трансформации. Но това може да стане само чрез написване на програмен код от потребителя, което отмества фокуса от заниманията му с матрични трансформации, принуждавайки го да се запознава с програмни езици и технологии, както и да се бори с изчистването на технически грешки.

Различни сайтове представят теорията на матричните трансформации чрез премери и изображения, но или използват 2D трансформации, или само предварително дефинирани обекти. Например [GeoGebra Visualizing Matrix Transformations](https://www.geogebra.org/m/y3eAJDXq) дава възможност за интерактивно задаване на точки и трансформирането им, но е ограничено в 2D пространството. Друг пример за 2D трансформации е [Matrix visualizer](https://web.ma.utexas.edu/users/ysulyma/matrix/), където може да качват различни изображения, които се трансформират чрез числово зададена матрица. Трансформация с 3D матрица може да бъде видяна в [Matrix Visualization - Kevin Bergamin](https://quazikb.github.io/matrixvis/index.html), но и тук матрицата се задава само чрез числови стойности, а трансформираният обект е фиксиран. При всяко от тези решения имаме възможност да трансформираме обекта, но липсва контрол върху вида на трансформацията и начина на задаването ѝ, както и възможност за моделиране на обекта, който се трансформира. Липсват и възможности за създаване на композиция от няколко матрици.

Тази дипломна работа има за цел да въведе потребителя в основите на матричните трансформации в 3D, предоставяйки му интерактивни възможности за създаването и композирането на различни видове трансформации, без това да изисква писане на програмен код. Обединяването на целия процес, от моделирането на матрични трансформации до прилагането им върху избрани от потребителя геометрични обекти, поставя няколко проблема.

За да бъдат решени тези проблеми, първо, има нужда от платформа, която да позволява създаването на разнообразни 2D и 3D геометрични обекти, както и възможности за трансформирането им. Следващата задача е изготвяне на лесен за употреба и гъвкав потребителски интерфейс, през който да се управляват трансформациите. Този интерфейс трябва да предоставя възможност за създаване и редактиране на различни типове трансформации. Трябва да предоставя възможност и за създаване на композиции от трансформации. От особена важност е наличието на метод за пренареждане на трансформациите, за да се онагледи факта, че прилагането на матрични трансформации не е комутативно и редът на прилагането им определя крайния резултат. За да покаже как линейната алгебра се използва за дефиниране на трансформация, потребителският интерфейс трябва да я показва като матрица от реални числа или математически формули. Последната задача е да се създаде интерактивен начин, който да показва как трансформацията се прилага върху избрани геометрични обекти.

В тази дипломна работа се представя основния математическия апарат, на които се базират матричните трансформации. Следват математически дефиниции на някои от най-използваните видове трансформации, някои техни свойства и композирането им.

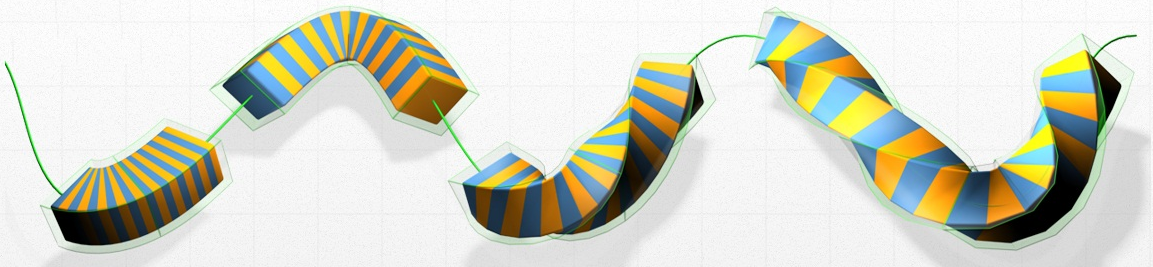
Представят се архитектурата и възможностите на избраната платформа – приложението представлява допълнителен модул към програмата за 3D моделиране и дизайн Vectorworks 2020. В самата програма потребителят може да моделира различни геометрични форми, върху които да се прилагат матрични трансформации.

Взаимодействието на потребителя с матричните трансформации се постига чрез диалог, от който потребителите имат възможност да избират от различни видове трансформации, като задават необходимите параметри за генериране на съответната матрица. Всяка промяна се визуализира в интерактивна компонента, която предоставя възможност за анимирано проследяване на трансформацията. След композирането на желаната трансформация, тя може да бъде приложена върху входния обект, променяйки го в модела.

При постигане на заложените цели, това приложение може да бъде използвано за обучение на програмисти, които искат да създават програми за 3D моделиране или игри. Може да помогне за осмислянето и откриването на практическата страна на курсове по линейна алгебра. Средата, в която е разработено, позволява използването му от професионалисти в областта на графиката и моделирането, с цел постигане на различни форми, или запознаване с теорията на 3D моделирането.



1. Създаване на сложни (многокомпонентни) обекти чрез позициониране на прости компоненти.



1. Форми, постигнати чрез усукване и превиване на паралелепипед.

# Математически дефиниции

## Вектор

За разбирането на матриците и координатните системи е важно понятието *вектор*.

В математиката и физиката вектори се наричат елементите на линейните алгебрични пространства. Най-често те се отъждествяват с размерностните си представяния като наредени n-орки от съответното числово поле – обикновено тройки или четворки реални числа. Векторът, понякога наричан геометричен или пространствен вектор, е геометричен обект, който има величина (или дължина) и посока, и може да бъде добавен към други вектори, съгласно векторната алгебра. В евклидовата геометрия векторът често се представя като линия с определена посока [[2]](#_Използвана_литература:).

Чрез вектори се описват и координатните системи: ако с (, , ) означим единичните вектори (спрямо метриката на координатната система) съответстващи на осите (Ox, Oy, Oz), то всеки вектор **P(a, b, c) = a\*i + b\*j + c\*k**., , са линейно независими вектори (наричат се декартов базис).



=a\*+b\*+c\*

1. Декартов базис и матрично представяне.

## Координатни системи.

Има различни координатни системи, чрез които могат да се дефинират представянията на моделираните обекти. Чрез координатни системи се определят и трансформациите. [[3]](#_Използвана_литература:)

### Декартова

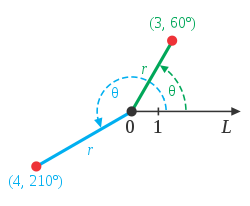
Тримерна декартова координатна система се определя от три взаимно перпендикулярни прави, които чрез избиране на положителна посока върху тях се превръщат в оси на системата, пресичащи се в точка О – начало на системата. Хоризонталната ос Ох се нарича абсцисна ос (от лат. abscissa – "отрязък'), а вертикалната ос Oy -– ординатна ос (от лат. ordinatus - "подреден"). Координатата Oz, която определя височината в пространството, се нарича апликата (от лат. applicata - "приложен", "добавен"), а оста – апликатна ос. Върху всяка ос се избира мерна единица за измерване дължините на отсечките. Положението на всяка точка се определя с координати х, y и z, които са проекциите ѝ върху трите оси – тройката (a, b, c).



1. Декартова координатна система.

### Полярна координатна система (в 2D)

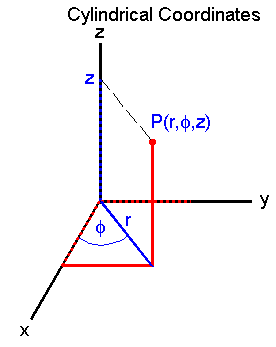
Полярна координатна система е двумерна координатна система, в която всяка точка в равнината се определя с две числа – *полярен ъгъл* и *полярен радиус*. Полярната координатна система е особено полезна в случаите, когато връзката между точките е по-лесна за изразяване чрез радиуси и ъгли, докато в декартова координатна система тази връзка може да се изрази математически само чрез прилагането на тригонометрични равенства. Лъчът в полярната система с полярен ъгъл 0 градуса, се нарича *полярна ос*. Точката, от която излиза полярната ос, се нарича *начало* или *полюс*. Всяка точка в равнината се определя от двете полярни координати: радиална и ъглова. Радиалната координата съответства на разстоянието от точката до началото. Ъгловата координата, която се нарича полярен ъгъл или *азимут*, означава се с θ и е равна на ъгъла, на който полярната ос трябва да се завърти обратно на часовниковата стрелка, за да достигне тази точка. Определената по този начин радиална координата може да приема значения от нула до безкрайност, а ъгловата координата се изменя в границите от 0° до 360°. Понякога за удобство областта от възможни значения на ъгловата координата може да бъде разширена извън границите на пълния ъгъл или да ѝ се присвоят отрицателни стойности, което съответства на завъртане на полярната ос в посока обратна на часовниковата стрелка.



1. Полярна координатна система.

### Цилиндрична координатна система

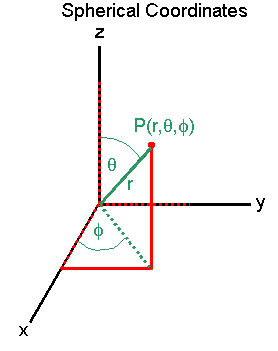
Цилиндричната координатна система е разширение на полярната 2D координатна система с добавяне на ос Oz. И всяка 3D точка се определя от тройката (r, φ, z). Аналогично на декартовата, трета координата задава височината на точката в пространството, проектирана върху местоположението определено в полярни координати.



1. Цилиндрична координатна система.

### Сферична координатна система

Сферичната координатна система е разширение на полярната 2D координатна система с добавяне на ъгъл. Положението на точката се представя с тройка сферични координати *( r , θ , φ )*, където *r* е полярният радиус на точката, *φ* е нейният азимут, а *θ* е полярният ѝ ъгъл. За дефиниция на сферична координатна система се избира полярната ос *Z* и перпендикулярна на *Z* равнина. Върху равнината се избира лъч с начало *O*, който определя референтния вектор за измерване на азимута (обикновено оста *X*).

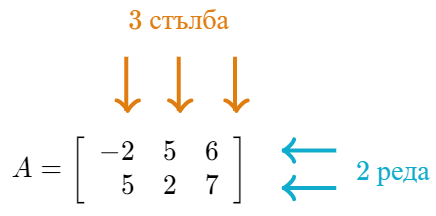


1. Цилиндрична координатна система.

## Матрици

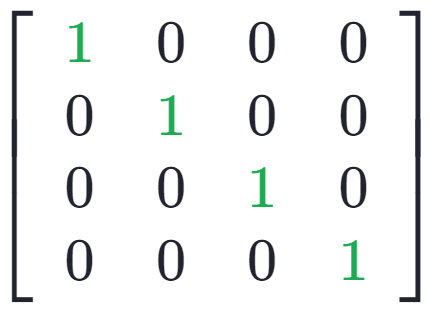
### Дефиниции

В математиката матрица А представлява правоъгълна таблица от елементи, най-често числа (числова матрица). Елементите на матрицата могат да бъдат от произволно алгебрично поле, обикновено реални числа. Матрица от тип n × m над поле F се нарича матрица, елементите на която са от полето F и има n реда и m стълба. За целите на компютърната графика се ползват матрици 3×3 или 4×4, в зависимост от трансформацията. [[4]](#_Използвана_литература:)



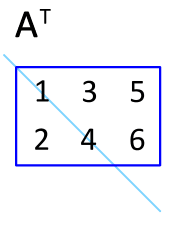
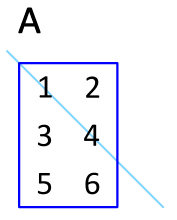
1. Представяне на матрица А с 2 реда и 3 стълба от цели числа

Позициите (1,1), (2,2),... (р,р), където р = min{n, m}, образуват главния диагонал на матрицата. Единична матрица е тази, на която всичките ѝ диагонални елементи са равни на 1. Чрез единична матрица може да бъде представена декартовата координатна система – редовете дефинират вектора на всяка от осите. [[5]](#_Използвана_литература:)



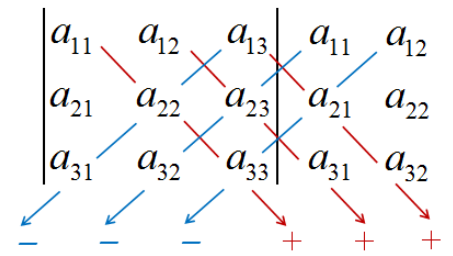
1. Единична матрица с размер 4×4

*Транспонирана матрица* е записването на редовете на матрицата A като стълбове (колони) на AT. Това може да бъде представено и като обръщането на A по главния ѝ диагонал. [[5]](#_Използвана_литература:)



1. Матрица A с размер 3×2 и AT с размер 2×3

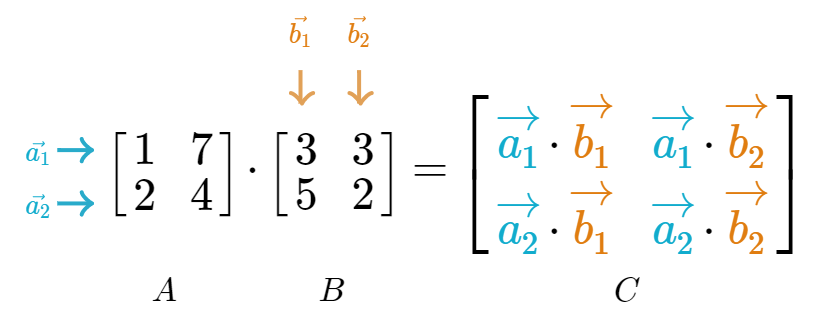
Детерминанта е функция, съпоставяща на квадратна матрица над комутативен пръстен с единица K, елемент от пръстена – многочлен, в който всеки едночлен е произведение от по един множител от всеки ред и стълб на матрицата с определен знак в зависимост от четността на пермутацията от елементи. [[6]](#_Използвана_литература:) За матрици от трети ред, каквито се използват в 3D моделирането, удобен начин за пресмятане е по правилото на Сарус. При него първите две колони се преписват още веднъж, след което се събират стойностите по получените диагонали. Сбора на всички вдясно от главният диагонал се събират, а тези вдясно от вторничния диагонал се изваждат. [[7]](#_Използвана_литература:)



1. Пресмятане на детерминанта по правило на Сарус.

### Умножение на матрици

*Скаларно произведение* на две наредени n-торки е сума от произведенията на съответните им елементи. Скаларното произведение на две наредени n-торки е винаги едно реално число. Когато умножаваме матрици разглеждаме всеки ред и колона от матрицата като наредена n-торка. *Произведение* на две матрици е матрица, в която елементът в i-тия ред и j-тия стълб е *скаларното произведение* на i-тия ред на първата матрица с j-тия стълб на втората. [[5]](#_Използвана_литература:)



1. Правило за умножение на матрици.

### Свойства на умножението на матрици

За да бъде произведението на две матрици добре дефинирано, броят на стълбовете n (колоните) в първата матрица трябва да бъде равен на броя на редовете m във втората матрица, защото скаларното произведение се намира като сбор от произведенията на съответстващите елементи. Ако произведението е дефинирано, получената матрица ще има същия брой редове като първата матрица и същия брой колони като втората матрица. [[5]](#_Използвана_литература:)



1. Условие за дефиниране на умножение на матрици.

Една от най-големите разлики между умножението на реални числа и умножението на матрици е, че умножението на матрици не е разместително (комутативно). При умножението на матрици редът, по който се умножават, има значение. [[5]](#_Използвана_литература:)



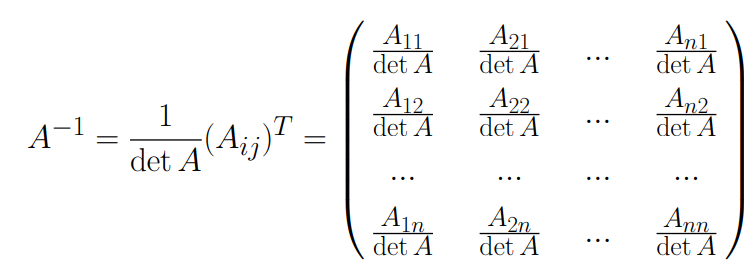
1. Резултат от умножението на матрица A и B в различна последователност.

Умножението на матрици е съдружително (асоциативно) и разпределително (дистрибутивно), както при реалните числа.

Произведението на всяка матрица с подходящата единична матрица е винаги първоначалната матрица, без значение от реда, в който е извършено умножението. Ролята, която единичната матрица играе при умножението на матрици, е сходна на ролята, която числото 1 играе при умножението на реалните числа. [[5]](#_Използвана_литература:)

### Обратни матрици

*Обратната матрица* на дадена е аналог на реципрочното число на дадено реално число. Както при умножението на число *x* с неговото реципрочно *1/x* се получа едно, така при умножението на матрицата A с нейната обратна A-1 резултатът е единичната матрица. Обратната матрица може да се изчисли по метода на адюнгираните количества. Този метод умножава транспонираната форма на матрицата AT по реципрочното на детерминантата det A. В случай, че детерминанта е 0, то нямаме дефинирана обратна матрица. В този случай при матрични трансформации ортогонални, афинни трансформации, за обрана матрица се използва транспонираната. [[2]](#_Използвана_литература:)



1. Формула за намиране на обратна матрица.

# Трансформации

## **Матриците като трансформации**

Трансформациите в геометрията са вид функции. Трансформацията приема като "вход" (аргумент) някакъв обект и дава като "изход" (стойност на функцията) някакъв друг обект. Трансформациите се използват, за да променим по някакъв начин обекта-аргумент – да го преместим, да го разтегнем или свием и т.н. В тримерното моделиране, обикновено, обектите са представени като точки в пространството (координатната система). Нека приемем всяка точка за вектор – нередна n-торка, или по-точно – матрица с n реда и една колона, а всяка n х *n* матрица дефинира трансформация. Тогава, за да трансформираме входния вектор е достатъчно да го умножим по трансформационната матрица. Произведението също ще е матрица с n реда и една колона – резултатът от трансформацията на вектора. [[1, 8]](#_Използвана_литература:)

Разглеждаме два основни вида трансформации – *афинни* и *неафинни*. За целите на тази дипломна работа, трансформациите ще са представени в декартова координатна система.

## Афинни трансформации

За тях е характерно, че запазват паралелността на линиите и съхраняват пропорциите върху паралелни линии. Не е задължително да се запазват ъглите между линиите, както и не е задължително да се съхраняват пропорциите върху непаралелни линии. Ако една афинна трансформация запазва дължините – т.е. запазва декартовият базис, ще я наричаме *ортогонална*. [[1, 8]](#_Използвана_литература:)

### Скалиране

Матрицата на скалиране задава коефициентите (sx, sy, sz) за умножение на всяка координата. Това променя координатите на обектите описани в нея – казваме че те се скалират. [[1, 8]](#_Използвана_литература:)

Скалиране с коефициенти по всички оси:



**x’ = sx x**

**y’ = sy y**

**z’ = sz z**

1. Матрична форма на скалиране с коефициент S.

*x*

*z*

*y*

1. Пример за трансформацията скалиране.

### Ротация

Задава се чрез ъгъл  и ос на завъртане. Този тип трансформация завърта останалите оси около избраната на съответния ъгъл. Приемаме, че положителна ротация е тази обратна на часовниковата стрелка. [[1, 8]](#_Използвана_литература:)

Ротация около оста Oz:



**x’ = x cos() – y sin()**

**y’ = x sin() + y cos()**

**z’ = z**

1. Матрична форма ротация около оста Oz.

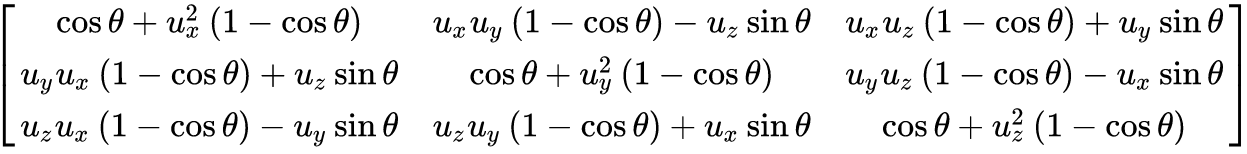
*x*

*z*

*y*

1. Пример за трансформацията ротация.

Това е опростен запис на формулата, чрез която се създава ротационна матрица за въртене около зададена ос. За да е една матрица ротационна и ортогонална, то оста на ротация трябва да е с дължина 1. За завъртане на θ около оста **u** = ( ux, uy, uz ), с размер 1, е изпълнено **ux2 + uy2 + uz2 = 1**. По този метод можем да конструираме ротация по всяка от другите оси. [[9]](#_Използвана_литература:)



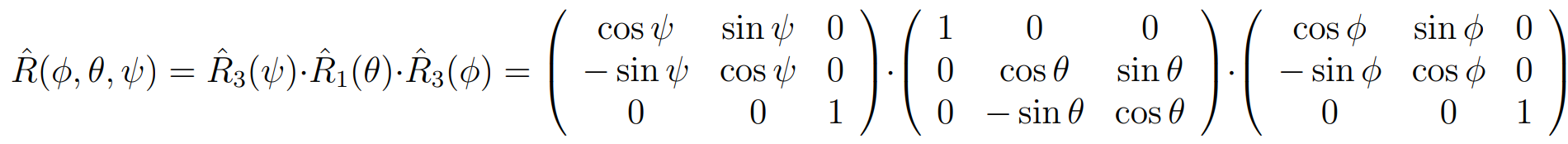
1. Матричен запис на формулата за трансформиралия с ротация θ около оста **u**.

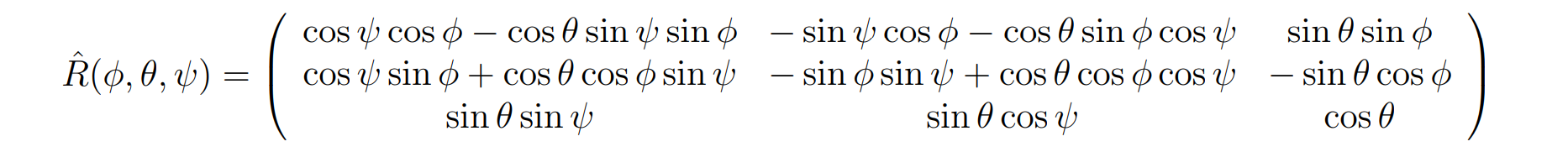
### Ойлерови ъгли от ротационна трансформация:

Използваме термина "ъгъл на Ойлер" за всяко представяне на ротационна трансформация, където разлагаме завъртането на 3 отделни ъгли. Това е репрезентация, която е близка до естественият начин на мислене и е подходяща за използване в потребителски интерфейс. Проблемът е, че последователността на прилагане на трансформациите е от значение, което прави пермутация от три елемента за подредба: xyz, xzy, yxz, yzx, zxy и zyx. [[8]](#_Използвана_литература:)

За да изведем формула за всяка от подредбите, се умножават матричните форми на трансформациите по всяка ос. От получената матрица, чрез тригонометрични уравнения, могат да се пресметнат съответните ъгли.

Изчислението на Ойлеровите ъгли за x(φ) y(θ) z(ψ) подредбата би изглеждала така: [[11]](#_Използвана_литература:)





1. Матричен запис на матрица композирана от Rx, Ry и Rz.

От този запис може да изведем:

**φ = arctangent( а13 / - а23 ) =** **arctangent( ( sin θ sin φ ) / ( - sin θ cos φ ) )**

**θ = arccosine( а33 )**

**ψ = arctangent( а31 / а32 ) =** **arctangent( ( sin θ sin ψ ) / ( sin θ cos ψ ) )**

### Транслация

Транслацията е преместване на модела на някакво разстояние, задава се чрез вектор на отместване (tx, ty, tz): [[1, 8]](#_Използвана_литература:)



**x’ = x + tx**

**y’ = y + ty**

**z’ = z + tz**

1. Матрична форма на транслация.

За да се стигне до унифицирано матрично представяне се добавя четвърта координата h:

или , където .

h се нарича също скалиращ коефициент. Това е формула на представяне чрез хомогенни координати. Ако h = 1, то транслацията в матрична форма ще е:



1. Хомогенна матрична форма на транслация.

### Скалиране и ротация с хомогенна координата система:



1. Скалиране с хомогенна координата система.



1. Ротация около оста Oz с хомогенна координата система.

### Наклоняване (shearing)

Наклоняването представлява отместване на точките на модела спрямо тяхната позиция. Задава се като добавяне на множество от един ред или колона към друга. [[1, 8]](#_Използвана_литература:)

Наклоняване по оста Ox:



**x’ = x + hy**

**y’ = y**

**z’ = z**

1. Матрична форма на наклоняване.

*x*

*z*

*y*

1. Пример за трансформацията наклоняване.

Обобщено за всички видове наклонявания, това може да се представи като: 

**x’ = x + h1\*y + h1\*z**

**y’ = y + k1\*x + k2\*z**

**z’ = z + l1\*x + l2\*y**

1. Обобщена матрична форма на наклоняване с хомогенна координата система.

### Отражение

Отражението е трансформация, при която се получава огледален образ на модела. [[1, 8]](#_Използвана_литература:)

Отражение по оста Ox:



**x’ = (-1) x**

**y’ = y**

**z’ = z**

1. Матрична форма на отражение.

1. Пример за трансформацията отражение.

## Неафинни трансформации

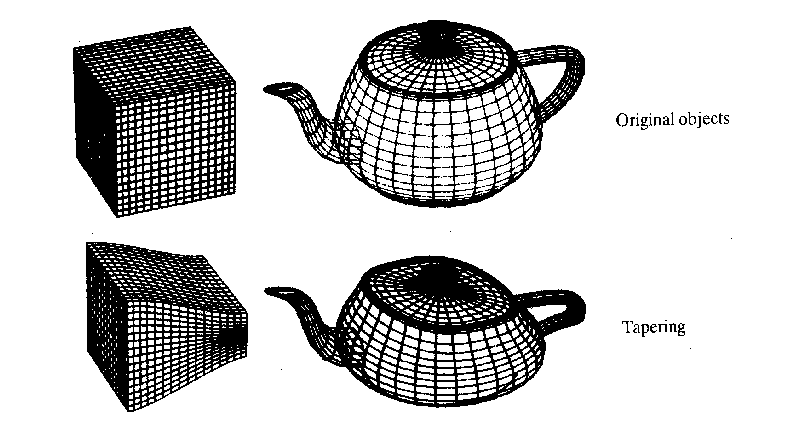
При тях за паралелни линии не е необходимо запазване на паралелността им, както и пропорциите им. [[1, 8]](#_Използвана_литература:)

### Заостряне (tapering)

Това е трансформация, при която точките се трансформират използвайки математическа функция върху някоя от осите. Обикновено колкото по-голяма става координата на точката по съответната ос, толкова по-голямо е отместването или обратното. [[1, 8]](#_Използвана_литература:)



1. Матрична форма на трансформацията заостряне.



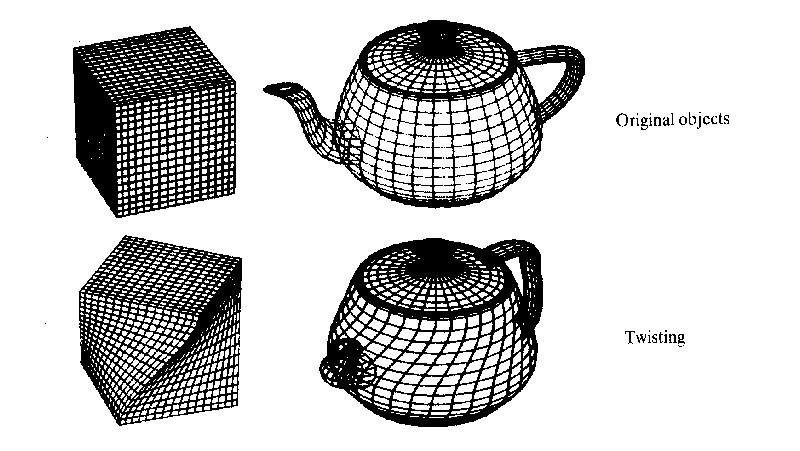
1. Пример за трансформацията заостряне.

Усукване (twisting)

Матрица на усукване се задава чрез ъгъл и ос. При усукването с движението по избраната ос се добавя ротация към модела. [[1, 8]](#_Използвана_литература:)



1. Матрична форма на трансформацията усукване.



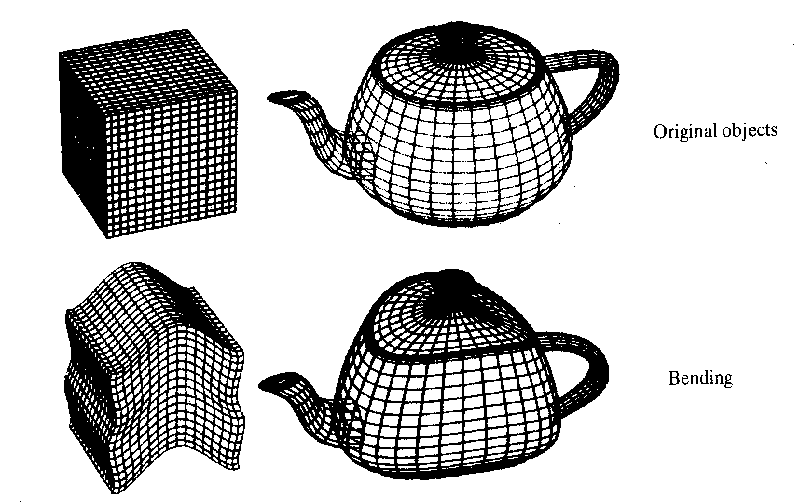
1. Пример за трансформацията усукване.

Превиване (bending)

Превиването се задава чрез една или няколко функции по някоя зададена ос. Чрез използването на различни функции могат да се създават много видове трансформации. [[1, 8]](#_Използвана_литература:)



1. Матрична форма на трансформацията превиване.



1. Пример за трансформацията превиване.

## Композиция от трансформации

Ако искаме да трансформираме по няколко начина едновременно един и същ модел се използва композиция от трансформации. Първо се дефинира всяка една от матриците, а след това те се умножават. Важно е да се отбележи, че редът на прилагане е от значение, тъй като това определя последователността на трансформиране. [[1, 8]](#_Използвана_литература:)

# Техническа информация

## Система за контрол на версиите

За контрол на версиите е използван GitHub.com. Това е онлайн система, базирана на Git и е широко използвана. Системата е безплатна за употреба за проекти с отворен код. Адресът на този проект е: <https://github.com/yordan-milkov/TransformMatrix>

## Използвани технологии

Приложението е разработено като разширение на програмата Vectorworks 2020 и използва VWSDK на съответната версия. SDK библиотеките са изтеглени от сайта на програмата, като включват инструкции и изисквания. Библиотеките са създадени на Visual Studio 2017 и използва Platform Toolset v141. Те включват и основните хедър файлове (.h) които се използват за компилиране. Към библиотеките има и приложени примерни проекти, използвани за основа на текущата разработка. [[13]](#_Използвана_литература:)

## Архитектура

### Vectorworks като приложение

Информация за техническите характеристики и възможности за създаване на нови модули за Vectorworks е налична в сайта за разработчици на адрес <https://developer.vectorworks.net> [[12]](#_Използвана_литература:). Vectorworks 2020 e 64-битово приложение, създадено за операционните системи Mac OS и Windows. То е модул-базирано и голяма част от функционалността на продукта е реализирана използвайки Vectorworks SDK. Vectorworks SDK е C++ библиотека, която дава достъп до базовите функционалности на продукта и методи за създаване на нови такива. Всеки модул (plug-in) представлява динамично свързана библиотека (Dynamic linked library) с разширение .LVB, както и специфичен файл с ресурси, който позволява промяна на езика на потребителския интерфейс, без промяна на изходния код. Също така, там се съдържат файловете с изображения, както и файлове с оформлението на диалозите.

### VCOM обекти и комуникация между тях

Модела на комуникацията между Plug-In обектите и Vectorworks се нарича VCOM (Vectorworks Component Object Model). Това е концепция, която много прилича на Microsoft COM. Тя дава възможност за създаване на обектно ориентирани, платформено независими, разпределени системи, използвайки двоични файлове. VCOM стандарта задава програмите изисквания за моделирането на обектите. [[14]](#_Използвана_литература:) Един VCOM обект съдържа съвкупност от данни и функции, чрез които се извършва комуникацията между ядрото на Vectorworks и различните модули. VCOM обектите могат да комуникират помежду си чрез указатели от абстрактен клас, използвайки виртуални методи.

За да предоставя функционалност, VCOM обект се състои от две основни части – дефиниция от абстрактен клас (интерфейс), и идентификатор - VCOM уникален стандартизиран ключ. Всеки програмен модул може да съдържа един или повече обекта, които могат да си взаимодействат. Имплементацията на всеки обект се нарича доставчик (provider) на функционалността, а този който я използва – клиент. Основните обекти се имплементират от ядрото на Vectorworks, а други се добавят от различните модули.

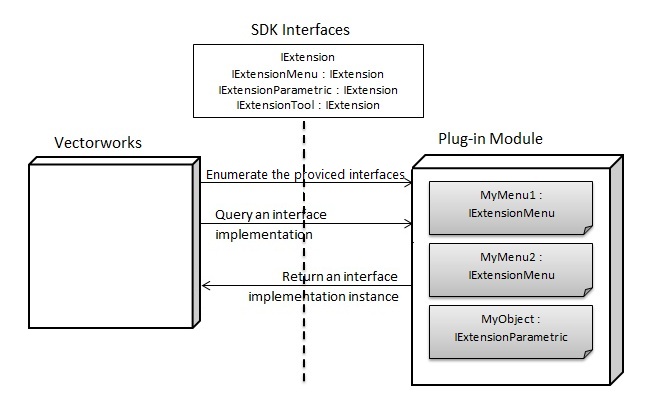
Всеки клиент може да извика доставчик ако има неговата дефиниция и идентификатор. От тях може да бъде взета инстанция към обекта използвайки **VCOMPtr**. Този теймплейтен клас, част основната SDK библиотека, предоставя достъп до методи, използвайки указател от абстрактен клас. Той се грижи за откриването, създаването и освобождаването на инстанцията на всеки VCOM обект. От своя страна, за да се определи един клас като VCOM доставчик, то той трябва да наследява **VCOMImpl**, подавайки му VCOM дефиницията като темплейтен параметър. Също така наследникът трябва да имплементира всички чисто виртуални методи.

При стартиране на Vectorworks, VCOM системата събира информация за възможните VCOM обекти от модули като извиква extern функцията plugin\_main. На нея като параметри се подава всичко необходимо за регистриране, инстанциране и освобождаване на VCOM обект. Това става чрез извикване на съответната темплейтна функция REGISTER от дефиницията на обекта, неговият ключ като параметър, както и останалите входни параметри.

### Основни VCOM обекти

Основният VCOM обект ISDK дава достъп до голяма част от ядрото на Vectorworks – създаване на обекти, промяна на документа, достъп до настройки, и много други. В SDK библиотеката на Vectorworks се съдържа други видове VCOM обекти, най-характерните които са разширенията (extensions). Това са дефиниции на доставчици за няколко типа основни функционалности, като умни обекти в документа, менюта и тулове в потребителския интерфейс на Vectorwoks, и други. Разширението меню е команда, която се избира през стандартния потребителски интерфейс. Разширението тул се стартира от палета и Vectorworks му предоставя интерактивен контрол върху документа – реагира на бутона на мишката, местене на курсора, статус на завършеност на операцията. Разширението умен обект се изобразява чрез 2D и/или 3D геометрични примитиви в документа, контролирани от набор от параметри. Разширението доставя геометрия, реагира на преместване на обекта в модела или друга промяна от потребителя.

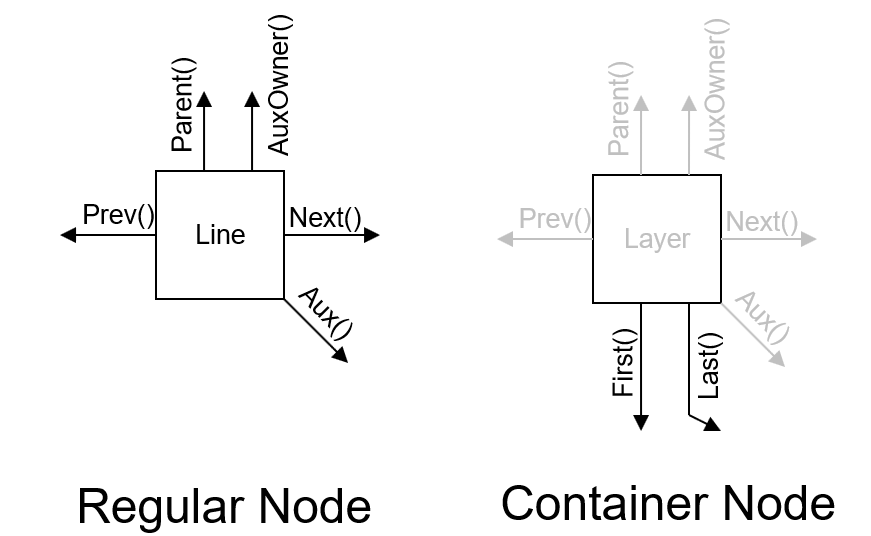
Разширение е композиция от имплементация на два основи VCOM обекта – **IExtension** и **IEventSink**. **IExtension** се използва за дефиниране на разширението пред Vectorworks – името на функционалността, както и необходимите данни, като категория за тулове и менюта, и параметри на умните обекти. За да осигури интерактивност за разширенията, те дефинират и **IEventSink** класа, който доставя и обработва данни свързани със събитията, които потребителят изпратил при своята работа във Vectorwroks. Ядрото на Vectorwroks се грижи за създаване на инстанции на различните разширения. Няколко примера за събития са: стартиране на меню командата от потребителския интерфейс; натискане бутона на мишката единично или двойно в документа се изпраща като събитие на активния тул.



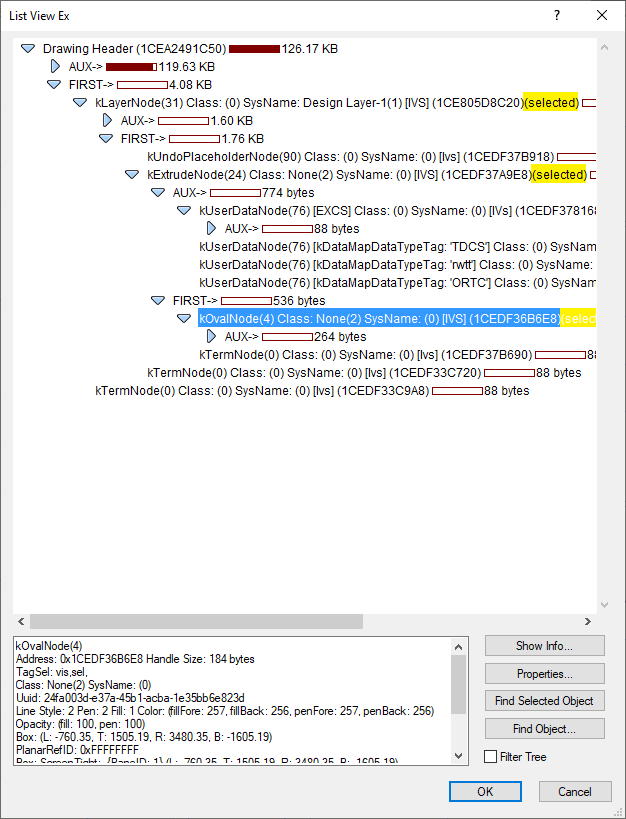
1. Схема на комуникацията между ядрото на Vectorworks и различните разширения.

### Организация на Vectorworks документа

Във Vectorworks за моделиране и визуализация се използва документа. Чрез различни менюта и тулове в него се създават геометрични примитиви и умни обекти – контейнери на геометрия, програмно моделирана чрез параметри. Организацията на документа представлява дървовидна структура, като всеки елемент (node) съдържа един или два двойносвързани списъка, виж Фиг. 38:. Първият списък e задължителен и съдържа допълнителна информация (auxiliary). Вторият списък е специфичен само за елементите, които са контейнери на геометрия. Информацията за всеки документ се държи корена на дървото (Document Header). В списъка с допълнителна информация на документа се откриват различните ресурси, които обикновено не дефинират геометрия, но са неразделна част от логиката – класове, текстури, дефиниции на символи и различни видове потребителски данни. В списъка за геометрия се откриват всички слоеве (layers) на модела – основните контейнери за обекти. Всеки създаден в модела геометричен обект принадлежи на определен слой.



1. Елементи от дървото на Vectorworks документа



1. Изглед на дървото на документа през диалога за разработчици Debug List View.

### Трансформации при рисуване на геометрични обекти на екрана

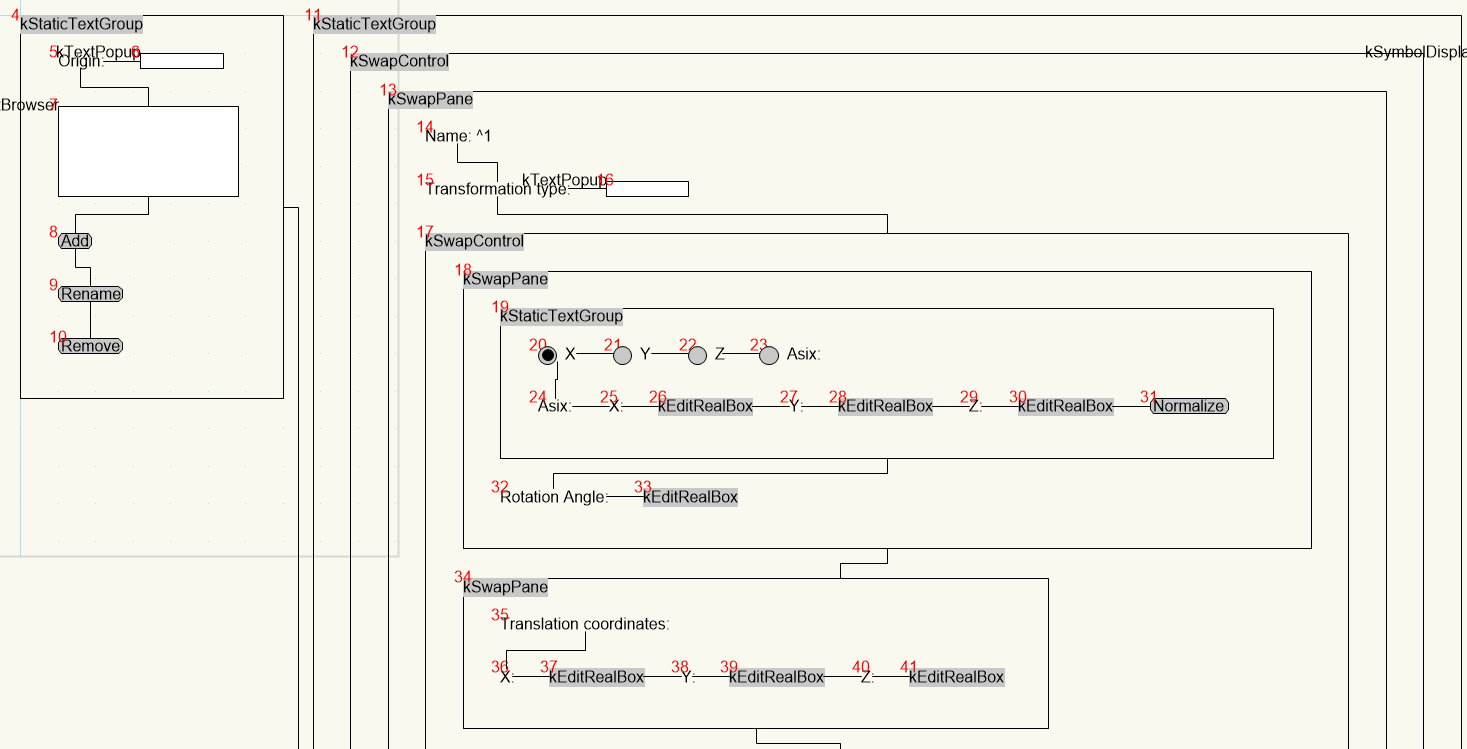
За да бъдат изобразени на екрана, модула за рисуване прочита съдържанието на документа започвайки от корена и последователно наслагва видимите обекти. Умните обекти представляват контейнери на геометрия, създадена в локална, независима от модела, координатна система. Те имат матрица на трансформация, чрез която се дефинира ориентацията им в модела. Друг обект с матрица е символ обекта. Неговата геометрия е дефинирана като ресурс – символ дефиниция, а в модела може да има неограничен брой инстанции, всяка със собствена трансформация. Модула за рисуване трансформира всяка от съдържаните в тези контейнери геометрични примитиви през матрицата на обекта. След това всички геометрични обекти се трансформират през матрицата на изгледа за да се появят правилно на екрана.

### Основни библиотечни класове във Vectorworks SDK

За улеснение на промяна на документа и моделиране на геометрия, Vectorworks SDK предоставя и набор от базисни класове VWFC. Тези класове могат да представляват различни типове обекти в документа, модификатори и други инструменти.

### Организация на модален диалог

За визуалното оформление на диалога се използва инструмент, вграден във Vectorworks – Dialog Builder. Той позволява всеки елемент на диалога – контрола, да бъде създадена като обект в чертежа и след това контролите да бъдат свързани в диалог. Типовете контроли, техните етикети и подравняване се редактира като при всеки друг обект в чертежа. Чрез бутон за визуализиране бързо може да се проверява състоянието на диалога до постигане на желания изглед. Готовото оформление се експортира в два ресурсни файла – един за изгледа и един за текстовете, който подлежи на превеждане на други езици. Последващи редакции по оформлението са възможни чрез зареждане на създадения ресурс на диалога и повторното му експортиране.

****

1. **Част от оформлението на диалог в документа използвайки** Dialog Builder.

**VWFC::VWDialog** е базов клас в Vectorworks SDK, който предоставя необходимата структура за имплементиране на модален диалог. Диалога се стартира от виртуален публичен метод, който се имплементира от базовия клас. Управлението на диалога се осъществява чрез реагиране на различни събитията, които настъпват от създаването до затварянето на диалога. За някои събития се регистрират специфични функции, а други събития се обработват чрез наследяване на различни виртуални функции. За допълнително удобство, към различните контроли могат да бъдат прикачени променливи от съответния тип – Dynamic Data Exchange. По време на обработване на събитията на диалога, всяка от свързаните променливи съдържа текущата стойност от съответната контрола. Промяна на тези данни може да бъде прехвърлена към потребителския интерфейс.

# Имплементация

## Входна функция plugin\_module\_main

Модула **Create Transform Matrix** дефинира две разширения – меню команда и тул.

extern "C" Sint32 GS\_EXTERNAL\_ENTRY plugin\_module\_main( Sint32 action

, void\* moduleInfo, const VWIID& iid, IVWUnknown\*& inOutInterface, CallBackPtr cbp )

{

// initialize VCOM mechanism

::GS\_InitializeVCOM( cbp );

Sint32 reply = 0L;

using namespace VWFC::PluginSupport;

REGISTER\_Extension<CreateTransformMatrix::CExtMenuTransformMatrix>

( GROUPID\_ExtensionMenu, action, moduleInfo, iid, inOutInterface, cbp, reply );

REGISTER\_Extension<CreateTransformMatrix::CExtToolPickObject>

( GROUPID\_ExtensionTool, action, moduleInfo, iid, inOutInterface, cbp, reply );

return reply;

}

1. Входна функция на модула **Create Transfrom Matrix**.

## Разширение Menu Transform Matrix

Дефиницията на меню командата CExtMenuTransformMatrix наследява VWExtensionMenu – SDK имплементация на **IExtension** за меню команда. Този клас дава възможност да се създаде бързо нова меню команда, предоставяйки готова обработка на абстрактните методи. Събитията се обработват от CMenuTransformMatrix\_EventSink, наследник на VWMenu\_EventSink - стандартна SDK имплементация на класа **IEventSink** за меню команда. Този клас улеснява обработката на събитията, добавяйки виртуални методи за всяко от тях. Единственият метод, който е предефиниран, е абстрактният DoInterface, който се извиква при стартиране на меню командата.

namespace CreateTransformMatrix

{

using namespace VWFC::PluginSupport;

class CMenuTransformMatrix\_EventSink : public VWMenu\_EventSink

{

public:

CMenuTransformMatrix\_EventSink( IVWUnknown\* parent );

virtual ~CMenuTransformMatrix\_EventSink();

virtual void DoInterface();

private:

CDlgTransformMatrix fDialog;

};

class CExtMenuTransformMatrix : public VWExtensionMenu

{

DEFINE\_VWMenuExtension;

public:

CExtMenuTransformMatrix( CallBackPtr cbp );

virtual ~CExtMenuTransformMatrix();

};

}

1. Дефиниция на разширението Menu Transform Matrix.

## Диалог Transform Matrix

### Стартиране на диалога

При извикване на меню командата се показва диалог – CDlgTransformMatrix който наследява **VWFC::VWDialog** класа. За стартирането на този диалог се използва функцията RunDialogLayout. Неговият изглед се зарежда от файла DlgTransformMatrix.vs – създаден чрез Dialog Builder, в имплементацията на събитието CreateDialogLayout. Първоначално съдържанието на диалога се попълва на събитието OnInitializeContent – запълва се списъка със създадените трансформации, съдържанието на контролите с падащ списък, както и се изчислява матрицата – резултат.

### Динамично свързани данни

Конфигурациите на различните матрици се пазят във вектор от структурата SDDXData. Тази структура съдържа информацията за конструиране на всяка от трансформациите – име, тип трансформация и специфичните ѝ параметри. В обработката на събитието OnDDXInitialize, както и при избиране на друга матрица от списъка, се извиква функцията RebuildDDX. В нея параметрите на избираната матрица се прехвърлят в потребителският интерфейс, използвайки Dynamic Data Exchange технологията. Промени от страна на потребителя се отразяват автоматично в съответната SDDXData структура. Диалога дефинира и допълнителни DDX променливи, които са свързани с цялостната логика, като тези за интерактивното трансформиране и началното място на трансформация.

// DDX

enum class ETransformType : Sint32

{

Rotation = 0,

Translation = 1,

Scaling = 2,

ObjectMat = 3,

Result = 4

};

struct SDDXData

{

bool fUse;

TXString fName;

Sint32 fTransformType;

bool fXRotRadio;

bool fYRotRadio;

bool fZRotRadio;

bool fAsixRotRadio;

VWPoint3D fAsixRotEdit;

double fRotAngleEdit;

double fXTranslateEdit;

double fYTranslateEdit;

double fZTranslateEdit;

double fXScaleEdit;

double fYScaleEdit;

double fZScaleEdit;

bool fSymetricScale;

InternalIndex fObjectIndex;

bool fObjectRotation;

bool fObjectTrans;

bool fInvert;

};

std::vector< SDDXData > fDDXControls;

SDDXData fResultDDX;

size\_t fOulerIndexPopup;

bool fFormulaView;

Sint32 fViewMarker;

Sint32 fRenderMarker;

bool fDetailedPreview;

short fOriginIndex;

Sint32 fSliderValue;

1. Дефиниция на динамично свързаните данни.

### Обработка на събития

При взаимодействие на потребителя с диалога се прихващат различни събития, чрез имплементирани функции за всяка от желаните контроли. Те са регистрирани използвайки макротата EVENT\_DISPATCH\_MAP\_BEGIN/END и ADD\_DISPATCH\_EVENT за задаване на връзка между константа на контрола и функцията, която да се извика. Такива функции са регистрирани за списъка с трансформации, бутоните за добавяне, преименуване и изтриване на трансформация. Бутоните за задаване и смяна на име са имплементирани чрез стандартизирания диалог за въвеждане на текст CStandardEditTextBoxDlg, а бутона за изтриване – чрез диалога за отказ или потвърждение VWDialog::AlertQuestion. Други контроли с функция за обработка са някои специфични за различните трансформации, като те се използват за валидиране на данни и осигуряване на логичното поведение на диалога.

Друго основно събитие, което се имплементира чрез виртуален метод, е OnUpdateUI. Този метод се извиква всеки път след действие на потребителя и задава кои контроли да са позволени за редактиране и кои не. Там се попълва и информацията за стойността на всяка клетка от матрицата. Това е и мястото на което се обновява екрана с интерактивна трансформация.

namespace CreateTransformMatrix

{

class CDlgTransformMatrix : public VWDialog

{

public:

CDlgTransformMatrix();

virtual ~CDlgTransformMatrix();

TransformMatrixAdvanced GetTransform( bool useSlider );

void TransformObject(VWObject handle=nullptr, bool useSlider=false);

// virtuals

protected:

virtual bool CreateDialogLayout();

virtual void OnInitializeContent();

virtual void OnDDXInitialize();

virtual void OnUpdateUI();

virtual void OnSetDownEvent();

// dispatch map

protected:

DEFINE\_EVENT\_DISPATH\_MAP;

void OnListBrowser( Sint32 controlID, VWDialogEventArgs& eventArgs);

void OnAddButton( Sint32 controlID, VWDialogEventArgs& eventArgs);

void OnRenameButton( Sint32 controlID, VWDialogEventArgs& eventArgs);

void OnRemoveButton( Sint32 controlID, VWDialogEventArgs& eventArgs);

void OnTypePopup( Sint32 controlID, VWDialogEventArgs& eventArgs);

void OnRotationRadio( Sint32 controlID, VWDialogEventArgs& eventArgs);

void OnNormalizeAsix( Sint32 controlID, VWDialogEventArgs& eventArgs);

void OnScaleEdit( Sint32 controlID, VWDialogEventArgs& eventArgs);

void OnSymetricCheck( Sint32 controlID, VWDialogEventArgs& eventArgs);

void OnPickObjectButton( Sint32 controlID, VWDialogEventArgs& eventArgs );

void OnResultPaneChage(Sint32 controlID,VWDialogEventArgs& eventArgs);

void OnRenderChnage( Sint32 controlID, VWDialogEventArgs& eventArgs );

void OnDetailedPreviewChnage( Sint32 controlID, VWDialogEventArgs& eventArgs );

...

1. Дефиниция на публична и виртуална част диалога Transform Matrix.

### Екран с интерактивен изглед

Технологията, използвана за визуализиране на геометрия в диалога, изисква създаване на символ дефиниция. В нея се поставя началната и трансформираната геометрия. За това се грижи имплементацията на класа CPreviewGeometry. Той събира избраните в модела обекти и ги трансформира с композираната матрица на всяка промяна от страна на потребителя. Този клас създава и управлява 3 основни обекта:

* Кеш: група-контейнер на копие на селектираната от модела геометрия
* Геометрия на опростен изглед: паралелепипед със страни успоредни на осите на глобалната координатна система и размер достатъчен да обхване всички точки на кеш групата.
* Мрежа от точки (mesh): Поради системни лимитации на Vectorworks, геометричните примитиви могат да бъдат трансформирани само с ортогонални, неафинни трансформации. При създаване на такава трансформация, копие на кеш обекта се превръща в мрежа от точки. Този обект позволява всяка точка да бъде трансформирана успешно, но и води до загуба на детайлност, защото всяка повърхнина е представена от триъгълници, а не чрез математически модели.

class CPreviewGeometry

{

public:

CPreviewGeometry();

~CPreviewGeometry();

public:

MCObjectHandle AddCacheObjectCopy(MCObjectHandle inContainer);

MCObjectHandle AddSimplePreviewCopy(MCObjectHandle inContainer);

MCObjectHandle AddMeshCopy(MCObjectHandle inContainer);

WorldCube GetCacheCube();

void Clear();

protected:

MCObjectHandle GetOrGenerateCache();

private:

MCObjectHandle fCacheObject;

MCObjectHandle fPreviewObject;

MCObjectHandle fMeshObject;

};

1. Дефиниция на класа за интерактивен изглед.

### Клас за операции с матрици и композиране на трансформации

Трансформационните матрици и операциите с тях се осигуряват от класа TransformMatrixAdvanced, наследник на VWFC класа **VWTransformMatrix**. Този базов клас предоставя имплементирани методи за създаване на различни трансформации по входни данни, проверка за ортогоналност, както и умножение на матрици. TransformMatrixAdvanced разширява тези функционалности с няколко метода – проверка за афинност на трансформацията, извличане на ойлерови ъгли в различен ред, както и създаване на матрица от ойлерови ъгли. [[15]](#_Използвана_литература:)

namespace CreateTransformMatrix

{

class TransformMatrixAdvanced : public VWTransformMatrix

{

public:

TransformMatrixAdvanced();

~TransformMatrixAdvanced();

public:

bool IsAffine();

enum class EOulerAnglesOrder

{

XYZ,

XZY,

YXZ,

YZX,

ZXY,

ZYX

};

VWPoint3D GetOulerAngles

( EOulerAnglesOrder order = EOulerAnglesOrder::XYZ );

void SetOulerAnglesRotation

(const VWPoint3D& angles

, EOulerAnglesOrder notation = EOulerAnglesOrder::XYZ );

};

}

1. Дефиниция на класа за операции с матрици.

Всяка една от създадени в диалога трансформации, както и композицията им, може да се представи чрез този клас. Функцията GetMatrixForDDX на CDlgTransformMatrix е отговорна за създаване на инстанция от TransformMatrixAdvanced за всяка от добавените трансформации. За задаване на ротационна матрица се използва метода SetRotation, подавайки ъгъл и ос на ротация. Транслация се постига чрез метода SetTranslation. Матрица за скалиране се създава чрез метода ScaleAfter. Ако имаме матрица от обект, то тя директно се извлича чрез VWObject::GetObjectModelMatrix. За целите на интерактивния изглед се взема предвид и степента на трансформация, която да се приложи. Това става като входните параметри се променят процентно, като 0% винаги трябва да създава единичната матрица. За композирането на резултатната трансформация отговаря GetTransform от CDlgTransformMatrix, спазвайки реда, посочен от потребителя.

### Прилагане на резултата върху обектите в документа

При избиране на ОК на диалога, селектираните в документа обекти биват трансформирани използвайки матрицата на резултата. За тази цел всеки обект в документа се конструира инстанция от VWObject и се използва метода TransformObject. Ако матрицата не е афинна или ортогонална, то обектите първо се превръщат в мрежа от точки. След това се обхожда всяка точка, трансформирайки я чрез метода TransformMatrixAdvanced::TransformPoint.

## Разширение Pick Object Tool

### Стариране на тула

Един от възможните типове трансформация не се дефинира чрез математическа функция, а позволява извличане на матрица от обект в модела - Object Matrix. За да бъде избран обект е добавена функция към бутона Pick Object. При нейното изпълнение диалога подава връзка към променлива за резултата от тула, стартира го във режим за временна операция. Самият диалог се затваря, което приключва и изпълнението на меню командата.

### Дефиниране на разширението Pick Object Tool

Тулът CExtToolPickObject е имплементиран като наследник на **VWExtensionTool** – SDK имплементация на **IExtension** за тул. Към този клас, аналогично на меню командата, е създаден и CExtToolPickObject\_EventSink, отговорен за обработване на събитията. Чрез наследяване и имплементиране на няколко виртуални метода този тул позволява интерактивно избиране на обект, чиято матрица да бъде прочетена.

namespace CreateTransformMatrix

{

using namespace VWFC::PluginSupport;

class CExtToolPickObject\_EventSink : public VWTool\_EventSink

{

public:

CExtToolPickObject\_EventSink(IVWUnknown\* parent);

virtual ~CExtToolPickObject\_EventSink();

public:

virtual void DoSetDown

(bool bRestore, const IToolModeBarInitProvider\* pModeBarInitProvider);

virtual void PointAdded();

virtual void MouseMove();

virtual void HandleComplete();

public:

virtual void CallbackObjectIndex( InternalIndex& index );

protected:

void Clear();

protected:

InternalIndex\* fpResultIndex;

MCObjectHandle fPickObject;

};

// -----------------------------------------------------------------------

class CExtToolPickObject : public VWExtensionTool

{

DEFINE\_VWToolExtension;

public:

CExtToolPickObject(CallBackPtr cbp);

virtual ~CExtToolPickObject();

};

}

1. Дефиниция на разширението Pick Object Tool.

### Обработка на събитията на Pick Object Tool

Vectorworks извиква събитието MouseMove когато курсора на мишката се движи върху чертожната повърхност. В него се проверява дали на тази позиция може да се избере обект, и ако да, той се маркира в червено. Всички предишно маркирани обекти се изчистват на всяко MouseMove събитие. Когато потребителя кликне с мишката в документа се извиква имплементацията на метода PointAdded. В нея се проверява дали е избран валиден обект, и ако не е, то този клик бива изтрит от масива с избрани от тула точки. По подразбиране този тул очаква само една избрана точка от потребителя. При избран валиден обект се получава събитието за приключване работата на тула - HandleComplete. В този метод избраният обект се проверява за валидно име, необходимо за използване в диалога. Ако няма, то потребителя задава име използвайки стандартизирания диалог за въвеждане на текст CStandardEditTextBoxDlg. Информацията за обекта се зарежда в подадената преди стартиране променлива fpResultIndex, тула приключва своята работа и стартира отново меню командата. При приключване на тула се получава събитието DoSetDown, в него се изчистват данните, за да осигури безпроблемно последващо използване.

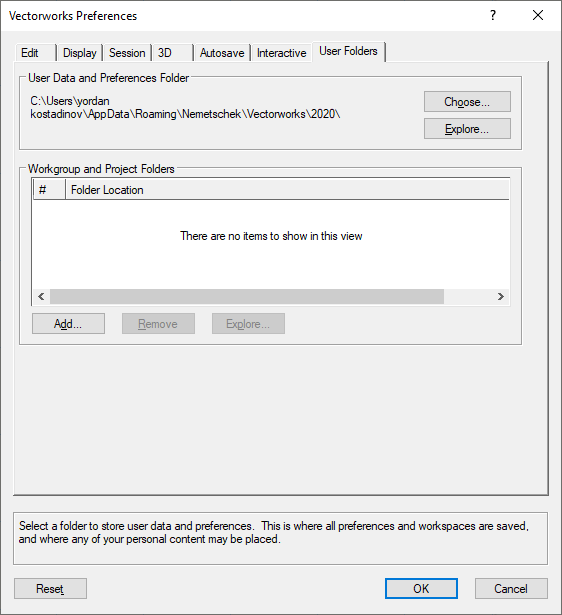
## Компилиране

На проекта във Visual Studio се задават Vectorworks SDK библиотеките, които могат да се използват. Изходният код представлява динамично свързвана библиотека, компилирана на 64 битова архитектура. Към изходния код се добавя и специален файл, който съдържа ресурсите, използвани от приложението.

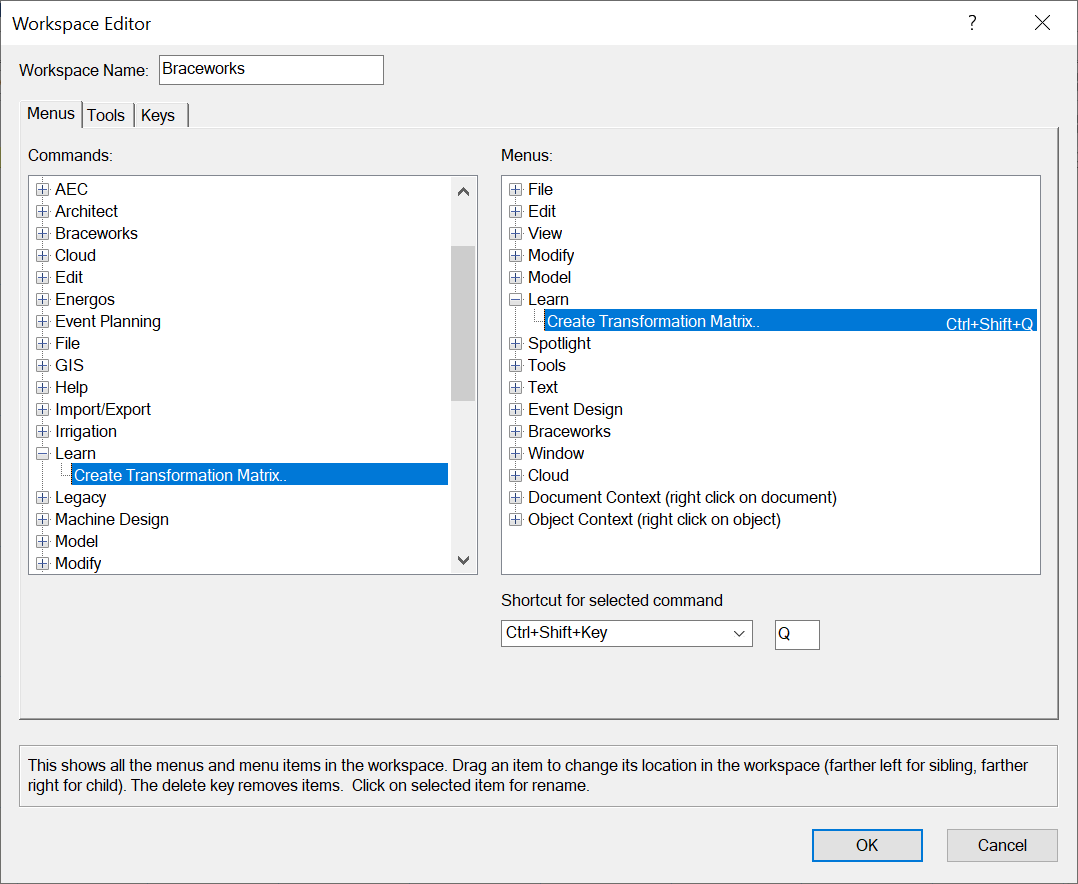
## Инсталиране:

Изходните файлове – динамичната библиотека и ресурсите, се поставят в потребителската папка на приложението, поддиректория Plug-ins. Потребителската папка на приложението може да бъде открито през интерфейса на Vectorworks 2020 – от меню Tools изберете Options, Vectorworks Preferences. В показалия се диалог отворете таба User Folder, където ще откриете пътят до нея във файловата система, както и бутона Explore. който ще я отвори в Windows Explorer. Уверете се, че сте изключили Vectorworks 2020 преди да преминте към копирането на файловете на разширението Transform Matrix в поддиректория Plug-ins.

Стартирайте отново Vectorworks 2020 от меню Tools и наивиграйте до Workspaces, Edit Current Workspace. От отворения диалог, таба Menus, от Commands, категория Learn, добавете менюто Create Transform Matrix на желаното място в Menus дървото. Затворете диалога с ОК. Меню командата трябва да е достъпна в лентата с менютата.



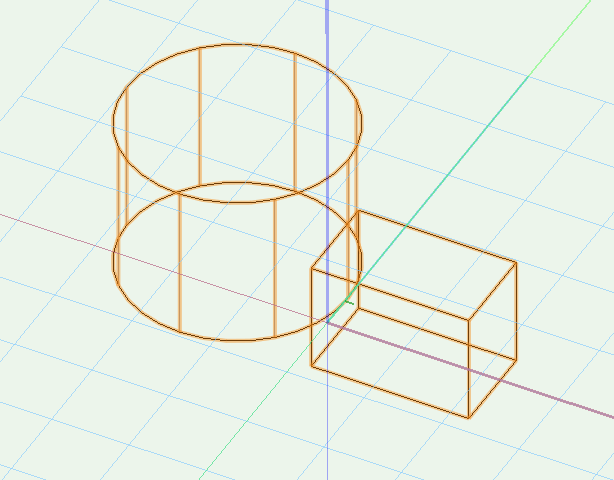
1. Vectorworks Preferences, User Folder.



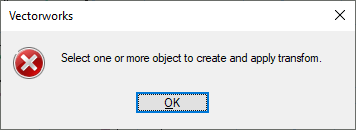
1. Редактиране на Workspace

# Потреибителски интерфейс

За да започнете работа с приложението, първо трябва да бъде създаден и маркиран поне един обект в модела. След това се стартира менюто **Create Transform Matrix...** , като избраните обекти ще бъдат трансформирани чрез създадената матрица. Ако нито един обект не е маркиран, стартиране на менюто показва съобщение за грешка.

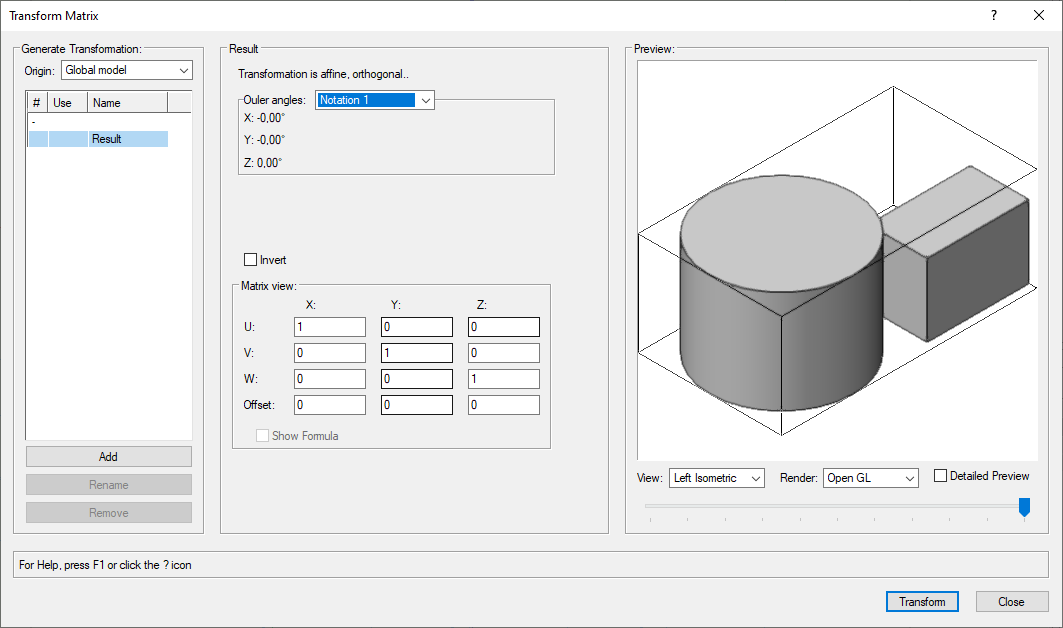


1. Обекти в модела.



1. Съобщение за грешка.

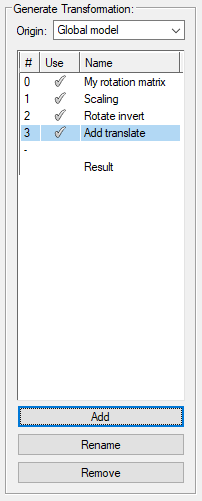
Отваря се основния контролен диалог. Той се състои от 3 основни групи. В ляво се намира групата за генериране на трансформация (**Generate Transformation**). В централната група **Transform** се визуализира дефиницията за конкретната трансформация, други настройки и стойностите на клетките на матрицата. Вдясно в групата **Preview** се намира интерактивния екран, където може да се проследява трансформацията върху обектите в различни настройки за визуализация.



1. Диалога Transform Matrix.

## **Generate Transformation**

От тази група може да се добавят, премахват и редактират трансформационни матрици. Първо е дадена възможност за определяне на центъра на координатната система, в която е трансформацията. Потребителят може да избира между глобалния център на модела, както и точките на центъра или на ъглите на обграждащия куб на селектирания обект.  
Следва таблицата с всички създадени до момента трансформационни матрици, както последният елемент **Result** съдържа данните за резултата от умножението на матриците. За всеки ред в списъка се съдържа информация за последователността на трансформацията, дали да се прилага при пресмятане на резултата, както и името ѝ. Списъкът винаги е сортиран по колоната **#** и чрез влачене с мишката на някой от редовете, последователността на трансформациите се променя. Знакът **✅** в колоната **Use** показва дали конкретната трансформация да е част от композицията или да не се пресмята. Последната колона **Name** съдържа уникалното име на матрицата.  
Следва бутон за добавяне на нова матрица – **Add**. При натискането му, потребителят въвежда името на следващата трансформация в диалог. Ако такова име вече съществува, диалога се показва наново до въвеждане на валидно име или отказ от операцията. След него е бутона **Rename**, чрез който може да се редактира името на избраната в трансформационна матрица. Това става през диалог, аналогичен на този за добавяне. Бутона **Remove** изтрива селектираната матрица от списъка с трансформации.



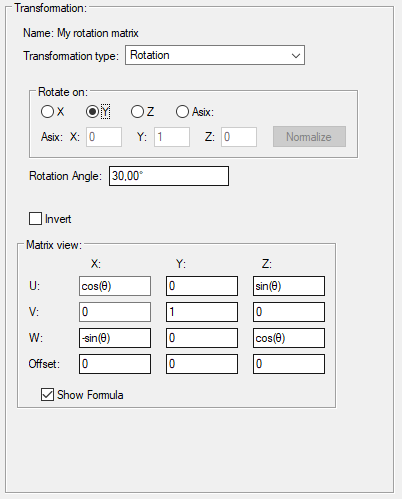
1. Групата Generate Transformation.

## Transform/Result група

Спрямо избрания в таблицата ред, в **Transform/Result** групата се визуализират данните на трансформационната матрица. Ако е избран ред с трансформация от композицията, то групата се нарича **Transform** и дава възможност за редактиране матрицата. Първо е изписано името на трансформацията. Следва контролата **Transformation type**, която определя типа на трансформацията. Различните типове трансформации определят изгледа на останалата част от **Transform** групата.

### Rotate

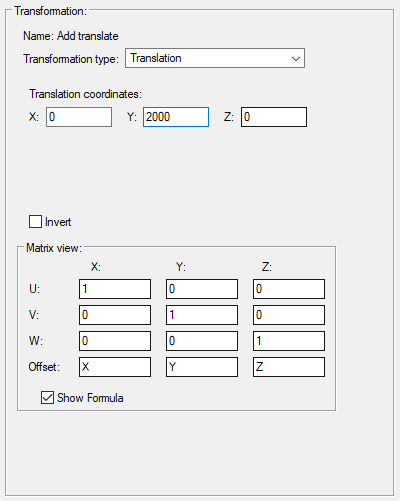
Първата възможна трансформация е ротация. Този тип трансформация върти геометричната фигура по зададен вектор и ъгъл на ротация. От показаната група с радио бутони **Rotate on** се избира ос на ротация. Тази група съдържа четири възможности – **X**, **Y**, **Z** и **Axis**. При избиране на **Axis**, потребителят може да конфигурира вектор, който да служи за ос. За тази цел контролите за въвеждане на цяло число на следващия ред, с етикети **X**, **Y** и **Z**, стават активни. Следва бутон **Normalize**, който променя дължината на вектора до 1 – нормала. Използване на ненормализиран вектор води до създаване на неортогонална матрица – скалиране по оста на ротация. Следва поле за въвеждане на градуса на ротация.



1. Контроли за трансформацията Rotate.

### Translate

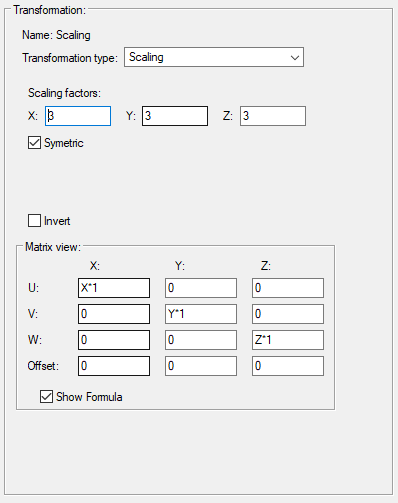
Следващата трансформация позволява транслация на обекта. Транслацията премества геометричният обект в пространството по зададен вектор. При избор на тази трансформация, потребителят може да въведе реални числа, които да формират вектора в полетата означени с **X**, **Y** и **Z**.



1. Контроли за трансформацията Translate.

### Scale

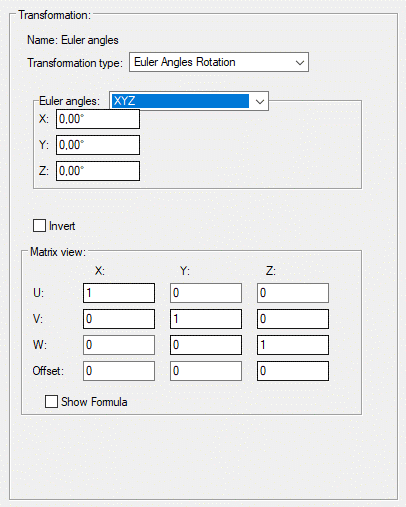
Друга възможна трансформация е скалиране. Скалирането увеличава или намалява обектите като променя дължината на **X**, **Y** или **Z** векторите, които дефинират матрицата. За всеки един от векторите потребителят може да въведе реално число – скалар, с което да бъде умножен. Следва контрола, в която може да се отбележи дали скалирането да е симетрично или асиметрично. Ако се избере симетрично скалиране, след всяка промяна на някоя от стойностите, тя автоматично ще се прилага и на останалите стойности на вектора.



1. Контроли за трансформацията Scale.

### Euler Angle Rotation

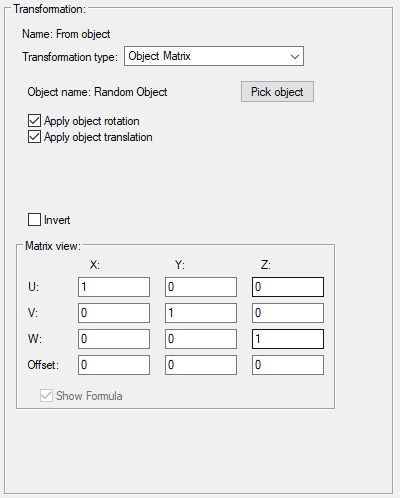
Ротационна матрица от Ойлерови ъгли позволява тя да бъде дефинирана по три ъгъла на ротация по всяка ос. Първо от падащо меню се избира последователността, в която да бъдат завъртани всяка от осите. Последователността е важна, защото всяко завъртане променя векторите, които дефинират останалите оси. Следват полета за ъгъл по **X**, **Y** и **Z**.



1. Контроли за трансформацията Euler Angle Rotation.

### Object Matrix

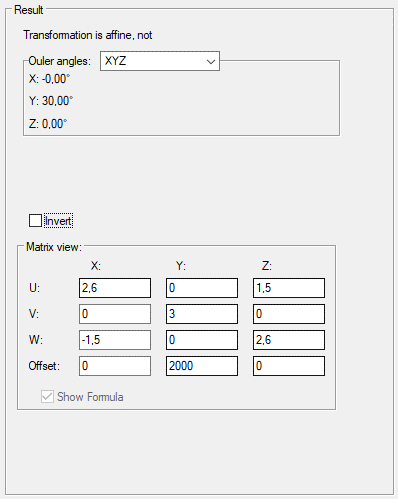
Голяма част от обектите във Vectorworks използват матрична трансформация, за да дефинират ориентацията и позицията си в координатната система на модела. Чрез тази матрица геометрията, създадена около своя собствена координатна система, се трансформира до позицията си в модела. Това е полезно при конструиране на трансформация свързана със съществуващ обект. Приложението позволява тази матрица да бъде прочетена и да се използва чрез опцията **Object Transform**. При нейното избиране се визуализира текст с името на избрания обект и бутон **Pick Object**. Ако няма избран валиден обект, вместо име е изписано **<No object picked>**, и е представена от единичната матрица в композицията. При натискане на бутона **Pick Object**, диалога се затваря и чрез интерактивен тул потребителят избира обект, чиято матрица да бъде използвана. При избиране на обект в документа, потребителят трябва да въведе уникално име на обекта, ако такова няма. При приключване работата на тула, диалога отново се отваря. Ако успешно е избран обект, неговото име ще бъде изписано, а матрицата ще бъде визуализирана. След това потребителят може да избере дали да използва ротацията и/или транслацията от извлечената матрица за пресмятане на резултата.



1. Контроли за трансформация от Object Matrix.

### Result

Ако от таблицата с трансформации е избран **Result**, то групата **Transform** ще се нарича **Result** и ще съдържа информация за матрицата резултат от композицията. Първо е изписана информация дали трансформацията е афинна и дали е ортогонална. Следва групата **Euler Angles**, от която се избира нотация, в която да бъдат пресмятани Ойлеровите ъгли.



1. Контроли за визуализация на Result трансформацията.

### Invert

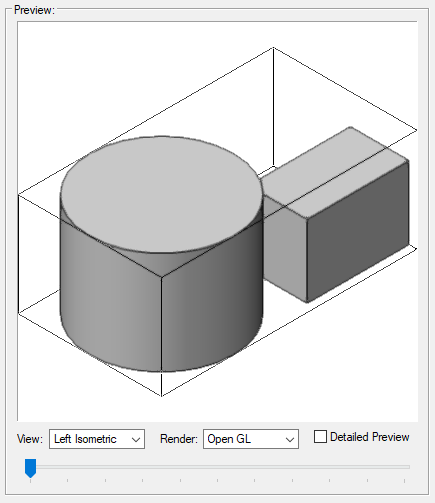
Следващите два елемента се отнасят до всяка една матрица, която може да се покаже в **Transform** групата. Първо е чекбокса **Invert**, който заменя избраната матрица с обратна ѝ.

### Matrix View

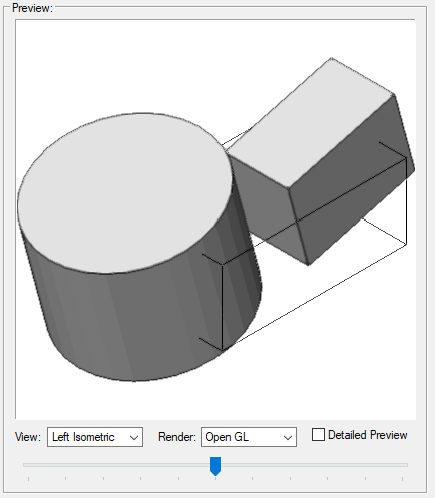
Следва подгрупата **Matrix View**, където в таблица с размер 3 колони на 4 реда са показани стойностите на матрицата. Редовете са обозначени като **U**, **V**, **W** и **Offset** векторите, а колоните с координатите **X**, **Y** и **Z**. Всяко поле в таблицата е попълнено със съответния елемент от матрицата. Ако от последващата възможност **Show Formula** е избрана, то вместо стойността на клетката, в полето е изписана формулата, по която е получена. Тази опция няма ефект при показване на матрица на обект, както и за матрицата на резултата.

## Preview group

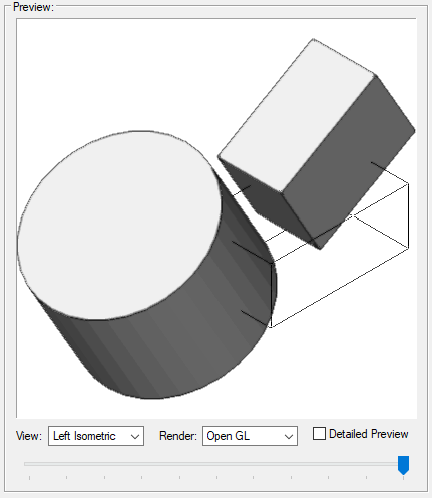
Следва групата за интерактивно проследяване на трансформациите. Там в прозорец с визуализация на модела, е показан входния обект и резултата след като е бил трансформиран през композираната матрица. Под този прозорец са настройките, които могат да бъдат прилагани към изгледа. Първата е **View**, която контролира ъгъла на камерата на визуализацията. Тази контрола съдържа списък със стандартните изгледи. Следва **Render**, която контролира как се създава визуализацията – 3D рендер в **OpenGL**, само линии в **Wireframe**, или само видимите от изгледа линии – **Hidden Line**. За визуализацията на входния обект има два варианта – представен детайлно от самата входна геометрия, или опростената до обграждащия ѝ куб. Това се избира от чекбокса Detailed Preview, който по подразбиране е включен. Потребителят може да проследи как трансформацията се изпълнява прогресивно, използвайки сайдера най-долу. Когато е в най-дясно положение, това е пълната, завършена трансформация, а когато е най-вляво – няма приложена трансформация. Проследяването на трансформацията показва нагледно как реда на умножение на матриците се отразява на модела. Това е близко до човешката представа за местене, въртене и разпъване в пространството около нас, което го прави по-лесно за разбиране.



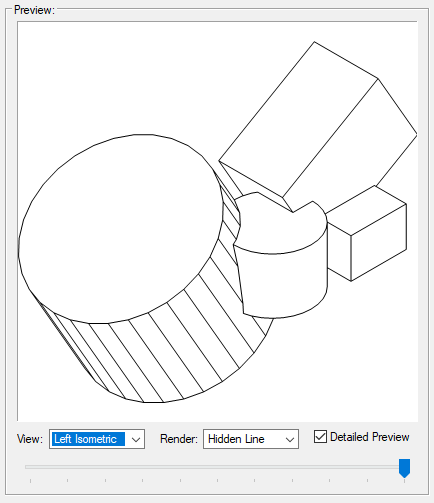
1. Интерактивно представяне без да е приложена трансформация.



1. Интерактивно представяне с приложена трансформация наполовина.



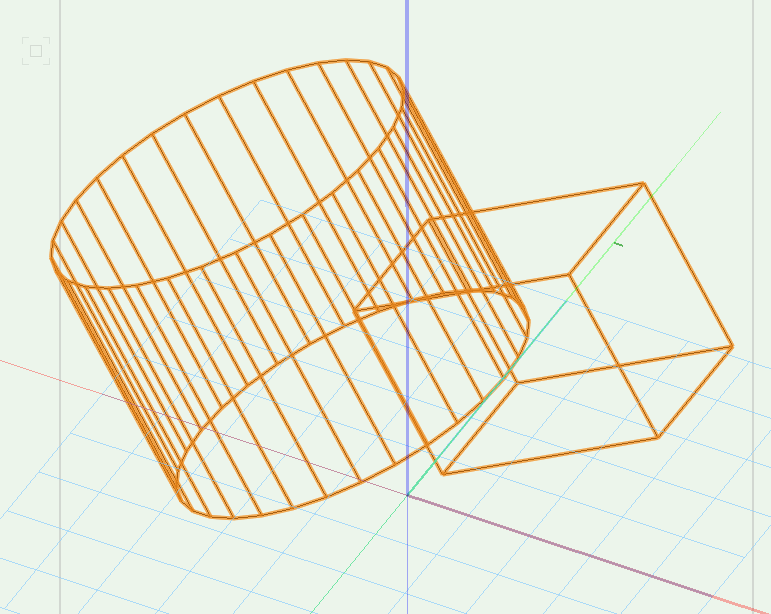
1. Интерактивно представяне с пълна приложена трансформация.



1. Интерактивно представяне в Hidden Line и с Detailed Preview.

## Прилагане на трансформацията на обектите в модела

Винаги при затваряне на диалога, създадената до момента трансформация ще бъде запазена за следващата сесия. При затваряне със системен бутон X или Close, нищо в модела няма да се промени. Но ако бъде избран бутона Transform, то на избраните в модела обекти ще бъде приложена резултатната матрица, така както е създадена. Ако резултатната матрица не е ортогонална, поради системни ограничения, геометрията на всички трансформирани обекти ще бъде превърната в Mesh обекти.



1. Трансформираните обекти в модела.

# Заключение:

Приложението Матрични трансформации – интерактивен самоучител предоставя възможности за интерактивно дефиниране и работа матрици. Основното му полза е, че матричните трансформации се случват на момента, без нужда от създаване на програмен код и поддържайки неограничен набор от геометрични обекти. Също така, самите матрици са видими за потребителя, което позволява по-лесно осмисляне на математическата абстракция в контекста на практическото приложение.

Основните задачи, които са изпълнени, бяха избиране на среда, в която да бъде реализирано – приложението Vectorworks, достъпно в залите на НБУ и предоставящ безплатен лиценз за студенти; Беше създаден програмен модул, през който различни трансформации могат да се конфигурират и композират, показвайки текущото състояние на матриците, както визуален резултат от прилагането ѝ. Този подход позволява лесна работа с приложението както от страна на студенти в специалности свързани с информационните технологии, така и на потребители, които професионално ползват софтуер за триизмерно моделиране.

По време на създаването на приложението усвоих нови знания относно триизмерното моделиране. Използвах и самото приложение за да моделирам сложни композиции от трансформации, които приложих в други задачи. Готовото приложение и документацията към него са основа на обучение за работа с матрици.

Някои трудности пред изпълнението на задачите бяха създаването на потребителски интерфейс, който да е удобен, функционален и лесен за употреба. Също така, откриването на информация за представяне на трансформация чрез ойлерови ъгли, както и доказателства за изчисляването им, също беше сложна задача. Това се дължи на факта, че това представяне на трансформации не е удобно за употреба в програмен код, а се използва предимно за визуализация пред потребители. От програмна гледна точка, най-голямото предизвикателство беше оптимизирането на диалога по начин, който позволява бързо обновяване на визуалната компонента. Необходимостта от динамична промяна на последователността на прилагане на трансформациите наложи детайлно разучаване на избраната контрола за таблично представяне на данни.

Възможности за бъдещо развитие на приложението са добавяне на още видове трансформации, като наклоняване, превиване и други.

# Използвана литература:

1. Материали за трансформации, разработени съвместно с колеги и представяни на курс в друг университет.
2. Елементи на линейната алгебра: вектори и матрици - проф. Михаил Константинов <https://www.matematika.bg/visha-matematika/lineina-algebra-matrici>
3. <https://en.wikipedia.org/wiki/Coordinate_system>
4. <https://bg.wikipedia.org/wiki/Матрици_(Математика)>
5. <https://bg.khanacademy.org/math/algebra-home/alg-matrices/alg-intro-to-matrices>
6. <https://bg.wikipedia.org/wiki/Детерминананта_на_Матрица>
7. <https://en.wikipedia.org/wiki/Rule_of_Sarrus>
8. <https://www.euclideanspace.com/maths/geometry>
9. https://en.wikipedia.org/wiki/Rotation\_matrix
10. <http://mathworld.wolfram.com/Matrix.html>
11. <https://www.phas.ubc.ca/~berciu/TEACHING/PHYS206/LECTURES/FILES/euler.pdf>
12. <https://developer.vectorworks.net/index.php/SDK:Tutorial>
13. <https://www.vectorworks.net/support/custom/sdk>
14. <https://docs.microsoft.com/en-us/windows/win32/com/component-object-model--com--portal>
15. <https://www.geometrictools.com/Documentation/EulerAngles.pdf>
16. Video Demo от Vectorwroks 2020.
17. Изображения - Свалени с учебни цели от различни сайтове за трансформации.