***НОВ БЪЛГАРСКИ УНИВЕРСИТЕТ***

**Матрични трансформации – интерактивен самоучител**

***Дипломна работа за образователна степен бакалавър по информатика на***

Йордан Милков Костадинов, F76814

учебна година 2019/2020

Съдържание:

[Въведение 4](#_Toc37543054)

[Матрични трансформации 6](#_Toc37543055)

[Определение на Трансформация: 6](#_Toc37543056)

[Вектор 6](#_Toc37543057)

[Координатни системи. 7](#_Toc37543058)

[Декартова 7](#_Toc37543059)

[Полярна координатна система (в 2D) 8](#_Toc37543060)

[Цилиндрична координатна система 9](#_Toc37543061)

[Сферична координатна система: 10](#_Toc37543062)

[Видове трансформации: 11](#_Toc37543063)

[Афинни трансформации: 11](#_Toc37543064)

[Скалиране 11](#_Toc37543065)

[Ротация 12](#_Toc37543066)

[Транслация 12](#_Toc37543067)

[Скалиране и ротация с хомогенна координата система: 13](#_Toc37543068)

[Наклоняване (shearing) 14](#_Toc37543069)

[Отражение: 15](#_Toc37543070)

[Неафинни трансформации: 16](#_Toc37543071)

[Заостряне (tapering): 16](#_Toc37543072)

[Композиция от трансформации: 18](#_Toc37543073)

[Техническа информация: 19](#_Toc37543074)

[Система за контрол на версиите: 19](#_Toc37543075)

[Използвани технологии: 19](#_Toc37543076)

[Архитектура: 19](#_Toc37543077)

[Имплементация 22](#_Toc37543078)

[Компилиране 24](#_Toc37543079)

[Инсталиране: 24](#_Toc37543080)

[Потреибителски интерфейс: 26](#_Toc37543081)

[Generate Transformation 26](#_Toc37543082)

[Transform/Result група 26](#_Toc37543083)

[Rotate 27](#_Toc37543084)

[Translate 27](#_Toc37543085)

[Scale 27](#_Toc37543086)

[Object Matrix 27](#_Toc37543087)

[Result 28](#_Toc37543088)

[Invert 28](#_Toc37543089)

[Matrix View 28](#_Toc37543090)

[Preview group 28](#_Toc37543091)

[Прилагане на трансформацията на обектите в модела 28](#_Toc37543092)

[Използвана литература: 30](#_Toc37543093)

# Въведение

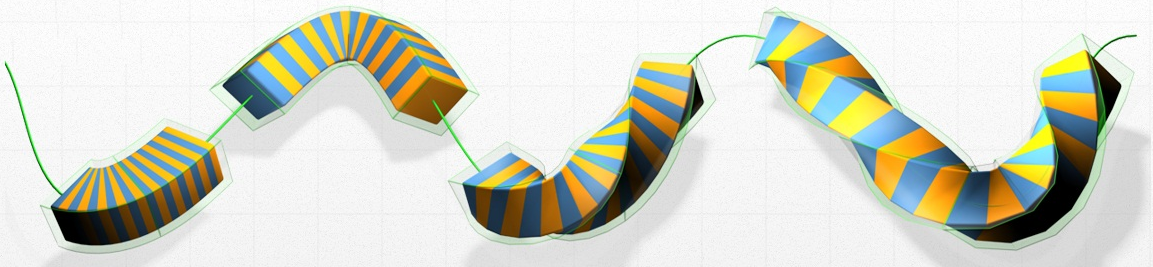
Векторите и матричните трансформации са най-широко използваният метод за моделиране на геометрия в областта на компютърната графика. Заедно с координатните системи, които дефинират, те са в основата на всяко приложение за проектиране и 3D моделиране, както и на видео игрите. Чрез тях моделите, дефинирани 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, също и за движението играча и обектите.

Това приложение-самоучител има за цел да покаже основите на матричните трансформации, като предоставя интерактивни възможности за създаване и композиране на матрични трансформации. Потребителите имат възможност да избират от различни видове трансформации, като задавайки необходимите параметри, се генерира съответната матрица. За да покаже, че умножението на матрици не е комутативна операция, приложението предоставя възможност за наслагване на различни трансформации и лесна промяна на реда им. Всяка промяна се визуализира в интерактивна компонента. Интерактивната компонента има и възможност за постъпково проследяване на трансформацията.

Приложението е разработено като допълнителен модул към програмата за 3Д моделиране и дизайн Vectorworks версия 2019. В самата програма потребителят може да моделира различни геометрични форми, върху които да се прилагат матрични трансформации. След композирането на жалната трансформация, тя може да бъде приложена върху входния обект, променяйки го в модела.

Фиг.: създаване на сложни (многокомпонентни) обекти чрез позициониране на прости компоненти

  
Фиг.: Форми, постигнати чрез усукване и превиване на паралелепипед.



# Матрични трансформации

## **Определение на Трансформация:**

функция, която задава съответствие между две множества/две пространства/две координатни системи.

В математиката, матрица 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 m × n над поле F се нарича матрица, елементите на която са от полето F и има m реда и n стълба. За целите на компютърната графика се ползват матрици 3 × 3 или 4 × 4, в зависимост от трансформацията.

## Вектор

За разберем матриците и координатните системи, важно е понятието Вектор.

В математиката и физиката вектори се наричат елементите на линейните алгебрични пространства. Най-често те се отъждествяват с размерностите 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 с векторната алгебра. В евклидовата геометрия векторът често се представя от част от линия с определена посока.

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.



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

, , са линейно независими вектори (наричат се декартов базис).

## Координатни системи.

Има различни координатни системи, чрез които могат да се дефинират представянията на моделите, същите които определят и трансформациите.

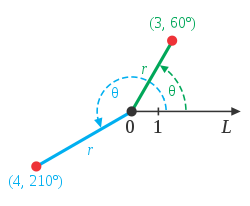
### Декартова

Тримерна декартова координатна система се състои 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).



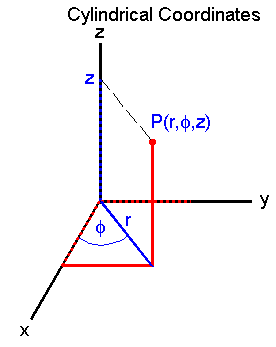
### Полярна координатна система (в 2D)

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 уравнения. Полярната система съдържа лъч при 0 градуса, който 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 и се означава с θ , е равна на ъгъла, на който полярната ос трябва да се завърти срещу 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 полярната 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.



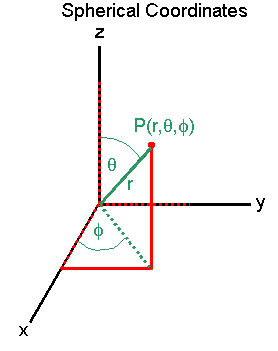
### Цилиндрична координатна система

Разширение на полярната 2D координатна система с добавяне на ос Oz. 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.



### Сферична координатна система:

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 полярния радиус на точката, φ е нейния азимут, а θ е полярния 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).



# Видове трансформации:

Разглеждаме два основни вида трансформации – афинни и неафинни.

## 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 съхраняват пропорциите върху непаралелни линии.

### Скалиране

Матрица на скалиране задава коефициенти за умножение на всяка координата (sx, sy, sz). Това променя размерността на координатната система и обектите описани в нея се скалират.

Скалиране с коефициент S по всички оси:

x’ = sx x

y’ = sy y

z’ = sz z

V матрична форма това изглежда така:



*x*

*z*

*y*

### Ротация

Задава се чрез ъгъл θ и ос на завъртане. Този тип трансформация завърта останалите оси около избраната на съответния ъгъл.

ротация около оста Oz:

x’ = x cos(θ) – y sin(θ)

y’ = x sin(θ) + y cos(θ)

z’ = z

в матрична форма това изглежда така:



*x*

*z*

*y*

### Транслация

Транслацията е преместване на модела на някакво разстояние, задава се чрез вектор на отместване (tx, ty, tz):

x’ = x + tx

y’ = y + ty

z’ = z + tz

В матрична форма може да се запише така:



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

или където 

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



### Скалиране и ротация с хомогенна координата система:

Скалиране:



Ротация около оста Oz:



### 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.

Наклоняване по оста Ox:

x’ = x + hy

y’ = y

z’ = z

В матрична форма това изглежда така:



*x*

*z*

*y*

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

В матрична форма това изглежда така:



### Отражение:

Отражението е трансформация, при която се получава огледален образ на модела.

Отражение по оста Ox:

x’ = (-1) x

y’ = y

z’ = z

В матрична форма това изглежда така:

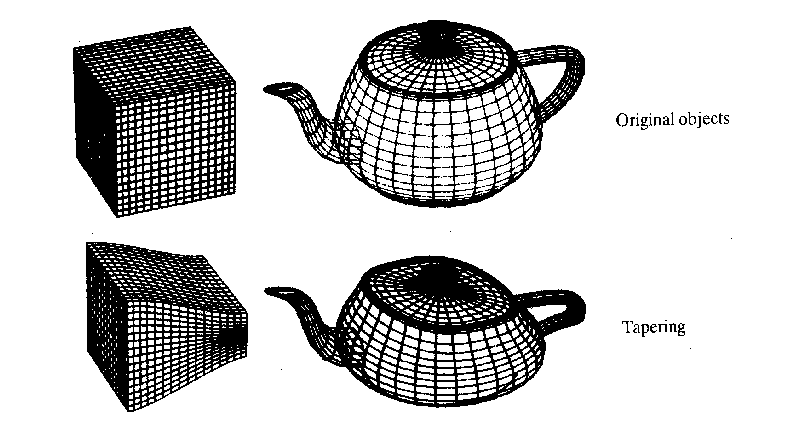


## Неафинни трансформации:

При тях за паралелни линии не е необходимо запазване на паралелността им, както и пропорциите им.

### 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.

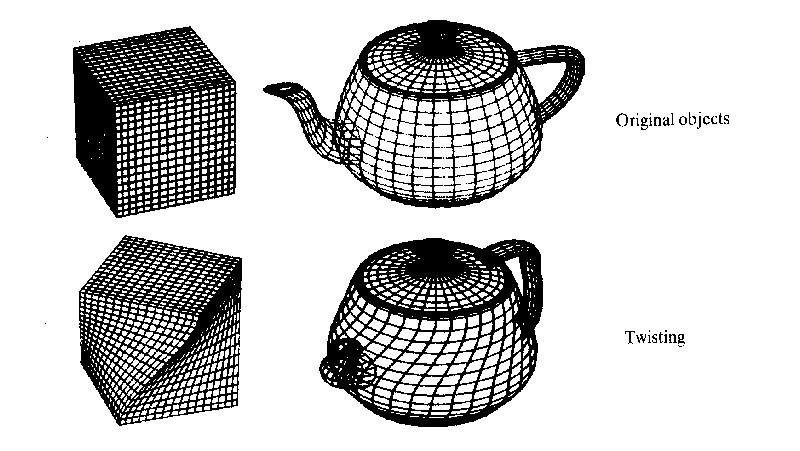


В матрична форма това изглежда така:



Усукване (twisting)

Матрица на усукване се задава чрез ъгъл и ос. При усукването с движението по избраната ос се добавя ротация към модела.

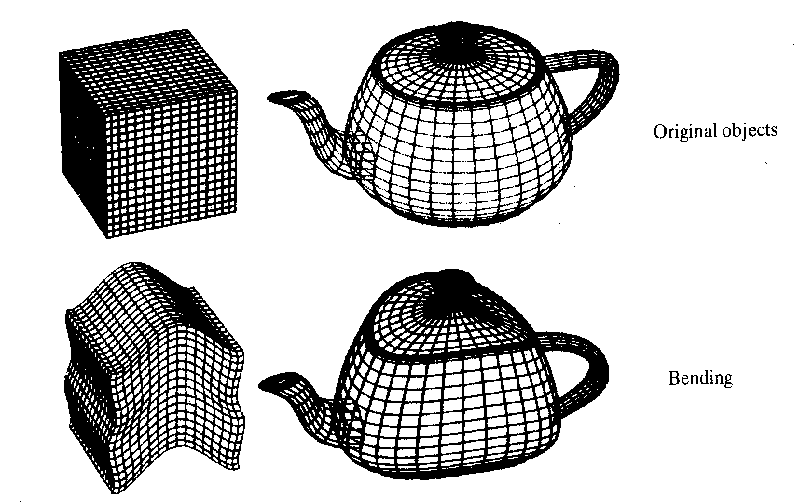


в матрична форма това изглежда така:



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.



в матрична форма това изглежда така:



## 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.

# 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 билиотеките са изтеглени от сайта на програмата, като включват инструкции и изисквания. Библиотеките са създадени на Visual Studio 2019 и използва tool-set версия ХХХ . 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.

## Архитектура:

Vectorworks 2020 e 64-битово приложение, достъпно за операционните системи Mac OS и Windows. То е модул-базиран и голяма част от функционалността на продукта е реализирана използвайки Vectorworks SDK. Vectorworks SDK е C++ базирана 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 сменяна на езика на приложението, без промяна на изходния код. Също така там се съдържат файловете с изображения, както и файлове с оформлението на диалозите.

Архитектурата е 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. Един VCOM обект съдържа съвкупност от данни и функции, чрез които се извършва комуникацията между ядрото на Vectorworks и различните модули. VCOM обектите могат за 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 (интерфейси), 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, представлява 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 дефиницията, използвайки я като темплейт параметър на класа **VCOMImpl**, и да имплементира всички чисто виртуални методи.

На 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 изтриване на VCOM обекти. Това става чрез извикване на съответната темплейтна функция REGISTER от дефиницията на обекта, неговият ключ като параметър, както и останалите входни параметри.

Основният 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 Vectowroks, и други. Разширение меню е команда, която се избират през стандартния потребителски интерфейс. Разширението тул се стартира от палета и Vectorworks му предоставя итнерактиревн контрол върху документа – реагира на цъкания с мишката, местене на курсора, статус на завършеност на операцията. Разширението умен обект се изобразява чрез 2D и 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 свойства 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; кликане единично или двойно в документа се изпраща като събитие на активния тул.

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 списъцa. 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. Всеки създаден в модела геометричен обект принадлежи на определен слой.

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

За да бъдат изобразени на екрана, модула за рисуване прочита съдържанието на документа започвайки от корена и последователно наслага 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 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 геометричните примитиви през матрицата на обекта. След това всички геометрични обекти се трансформират през матрицата на изгледа за да се появят правилно на екрана.

**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.

За визуалното оформление на диалога се използва инструмент, вграден във Vectorworks – Dialog Builder. Той позволява всеки елемент от 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, които подлежат на превеждане на други езици. Последващи редакции по оформлението са възможни чрез зареждане на създадения ресурс на диалога и повторното му експортиране.

# Имплементация

Модула Transform Matrix дефинира две разширения – меню команда и тул. 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 метода. 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 нови виртуални методи за всяко от тях. Единственият метод, който е предефиниран, е абстрактният DoInterface, който се извиква при стартиране на меню командата.

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 TransformMatrixDlg.vs – създаден чрез Dialog Builder, в имплементацията на събитието CreateDialogLayout. Първоначално съдържанието на диалога се попълва на събитието OnInitializeContent – запълва се списъка за създадените трансформации, съдържанието на контролите с падащ списък, както и се изчислява матрицата – резултат.

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.

При взаимодействие на потребителя с диалога се прихващат различни събития чрез имплементирани функции за всяка от желанието контроли. Те са регистрирани чрез макрото EVENT\_DISPATCH\_MAP\_BEGIN/END, използвайки 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 на име и преименуване 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 във всяка клетка от матрицата. Това е и мястото на което се преизчертава екрана с интерактивна трансформация.

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 всяко действие 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, нефини 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 модел.

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 различни нотации, както и създаване на матрица от ойлерови ъгли.

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. При необходимост За композирането на резултатната трансформация отговаря GetTransform от CDlgTransformMatrix, спазвайки реда, посочен от потребителя.

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.

Един от възможните типове трансформация не се дефинира чрез математически модел, 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.

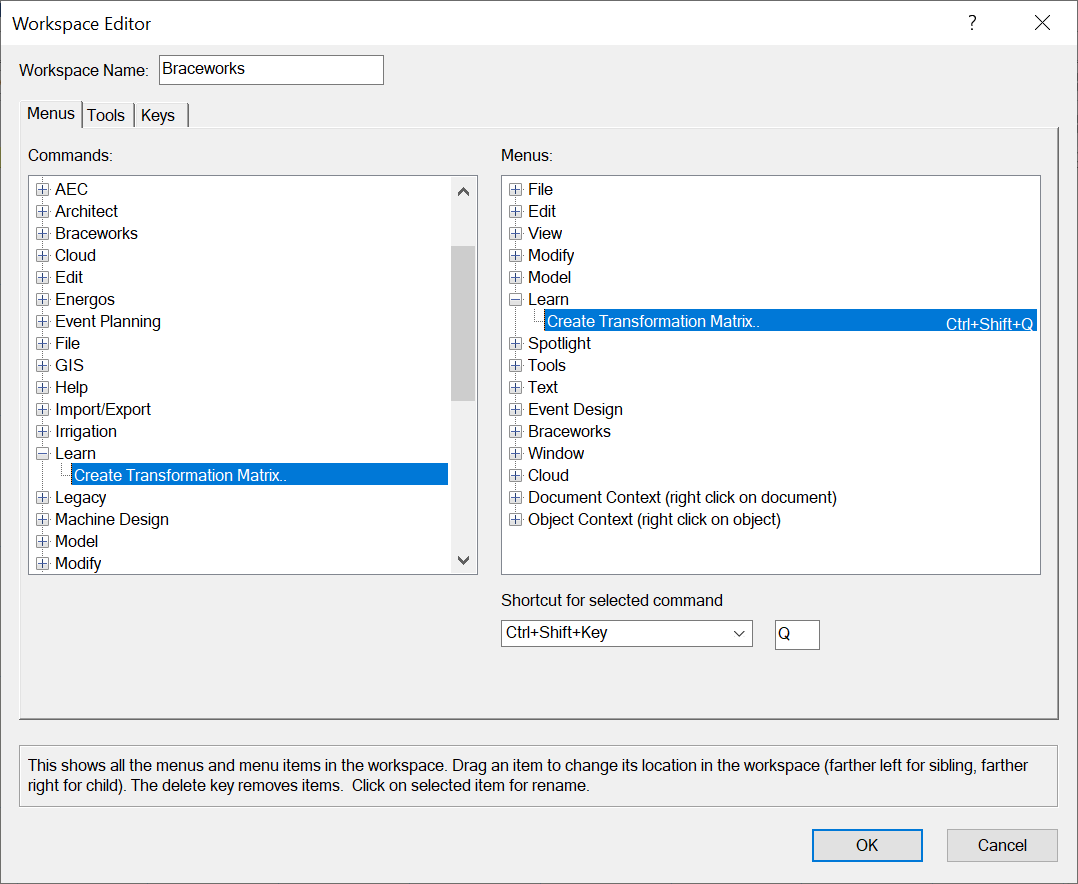
Тулът CExtToolPickObject е имплементиран като наследник на **VWExtensionTool** – SDK имплементация на **IExtension** за тул. Към този клас, аналогично на меню командата, е създаден и CExtToolPickObject\_EventSink, отговорен за обработване на събитията. Чрез наследяване и имплементиране на няколко виртуални метода този тул позволява интерактивно избиране на обект, чиято матрица да бъде прочетена. 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 събитие. Когато потребителя кликне с мишката в документа се извиква имплементацията на метода PointAdded. В него 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, 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 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. Потребитеската папка на приложенито 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 наиграйте 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.



# 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.

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 гупата Preview се намира интерактивния екран, където може да се проследява трансформацията върху обектите в различни настройки за визуализация.

## **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 и на ъглите на обграждащия куб.  
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. Тикчето 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.

## 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.

### 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.

### 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.

### 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.

### 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.

### 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 е винаги изключена 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 изключен.  
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.

## 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.

# Използвана литература:

Материали за трансформации, разработени съвместно с колеги и представяни на курс в друг университет.

<https://developer.vectorworks.net/index.php/SDK:Tutorial>

<https://docs.microsoft.com/en-us/windows/win32/com/component-object-model--com--portal>

Допълнителна информация - bg.wikipedia.org/ - Страници за тях от българската версия на Wikipedia.

Video Demo от Vectorwroks 2018.

Изображения - Свалени с учебни цели от различни сайтове за трансформации.

http://mathworld.wolfram.com/Matrix.html