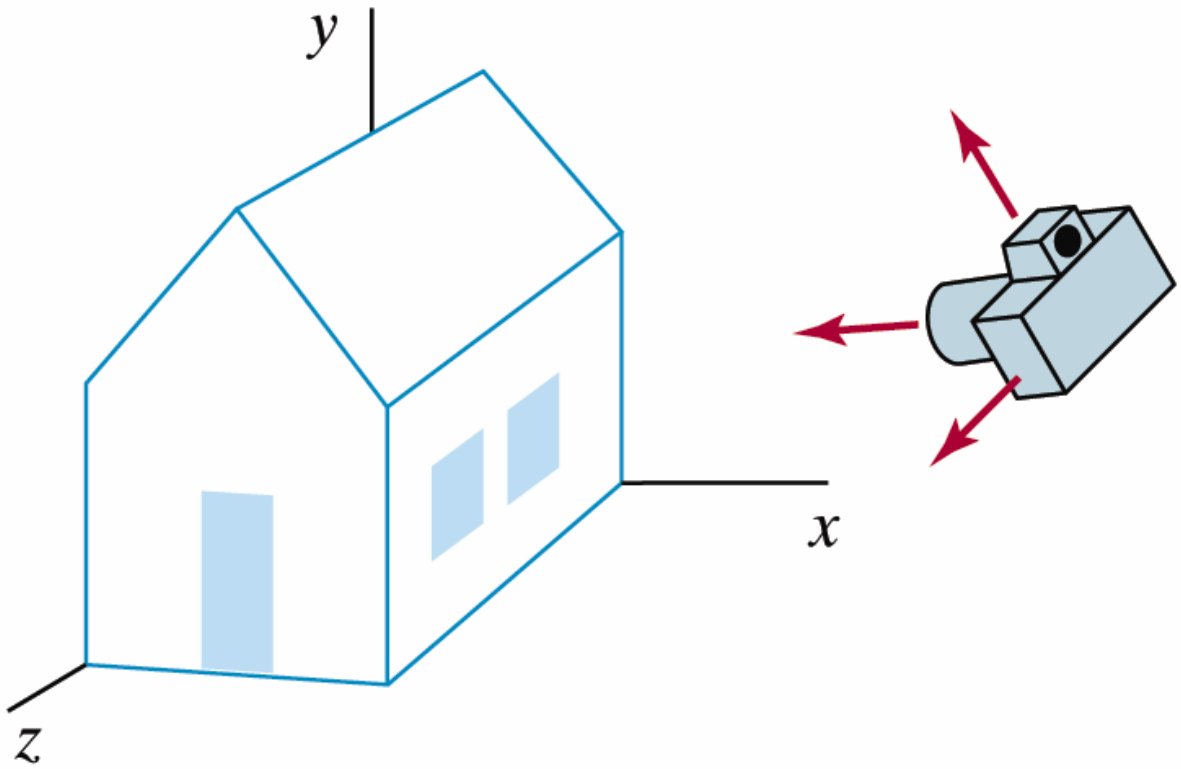
Компютърна графика

Двумерни геометрични трансформации

доц. Милена Лазарова, кат. КС, ФКСУ

Компютърна графика

- Създаване на компютърно генерирани образи



Hearn & Baker, Computer Graphics with OpenGL, 2004

Компютърна графика

- Графичен дисплей (видеодисплей)
  - област за изобразяване, съставена от масив от точки, които могат да се осветят по екрана
    - пиксели (pixels - picture elements)
  - атрибутите на отделните точки се задават и променят програмно

Компютърна графика

- Разделителна способност
  - максималния брой точки, които могат да се осветят по екрана, без да се препокриват
    - 480 реда всеки с по 640 пиксела – общо 307 000 пиксела
    - 1024 реда всеки с по 1280 пиксела – общо 1 310 720 пиксела
  - по-голям брой пиксели при дадени размери на екрана
    - по-висока разделителна способност
    - по-голям обем памет за представяне на сцената

- Коефициент на пропорционалност (Aspect Ratio)
  - отношение на броя на вертикалните към броя на хоризонталните точки, необходими за генериране на отсечки с еднаква дължина

Компютърна графика

- Всеки реален или въображаем ***обект*** се представя чрез модел
  - визуализира се като изображение
- ***Моделът*** е абстрактно описание на формата и атрибутите на обекта
  - чрез моделът се определят точките от обекта, които съответстват на пиксели в областта на изобразяване
  - при даден модел чрез приложна графична програма се генерира съответното изображение
- ***Изображението*** е двумерен масив от пиксели

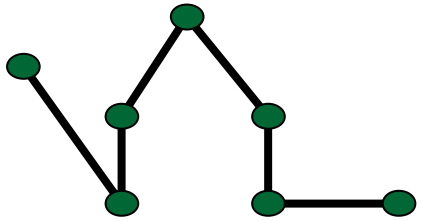
Компютърна графика

- Графична библиотека
  - набор от графични команди или функции
  - ориентирана към различни програмни езици
  - специфицира примитиви за 2D и 3D геометрични модели, които да се визуализират

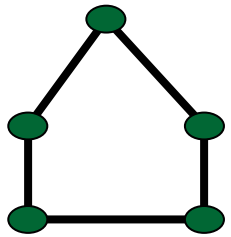
- Примитивен модел (примитив)
  - прости геометрични форми

Примитиви

- Полилинии
  - свързана последователност от прави линии
  - възли и линии (дъги, контури, вектори)

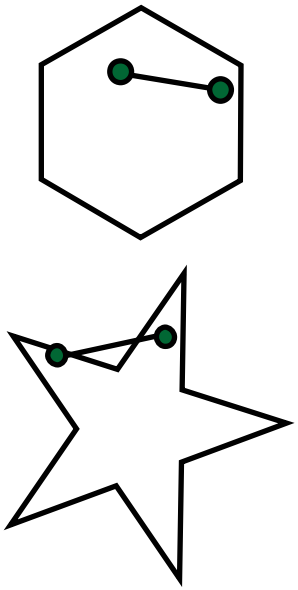


- Полигон
  - полилиния с начална и крайна точка свързани с линия
  - прост полигон
    - полигон, за който няма пресичащи се линии

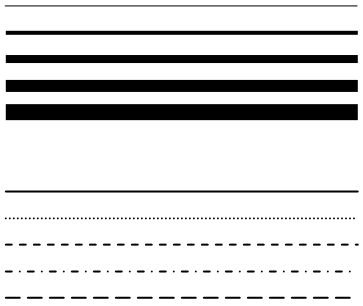


Примитиви

- Видове полигони
  - изпъкнал (convex)
    - отсечката между всеки две точки в полигона се съдържа изцяло в полигона
  - неизпъкнал (вдлъбнат) (concave)
    - съществуват поне две точки вътре в полигона, свързана с отсечка, която не е изцяло в полигона

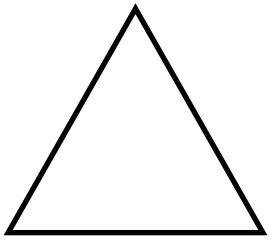


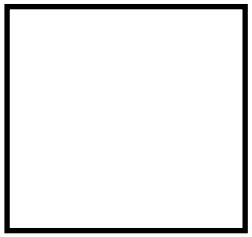
- Атрибути на полилинии
  - цвят (color)
  - дебелина на линията (thickness)
  - стил на линията (pattern)



Примитиви

- Специални полигони


триъгълник

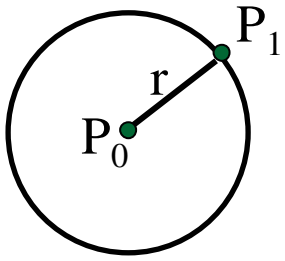

квадрат

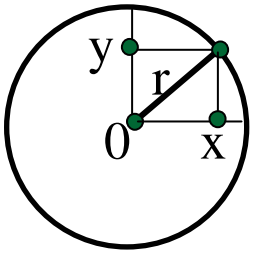

правоъгълник

- Окръжност
  - всички точки на определено разстояние от дадена точка (център)

$$r^2 = x^2 + y^2$$

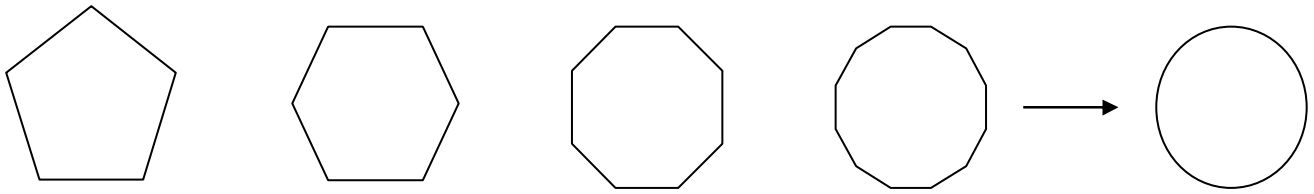
в Декартова координати система
и център на окръжността в началото
на координатната система



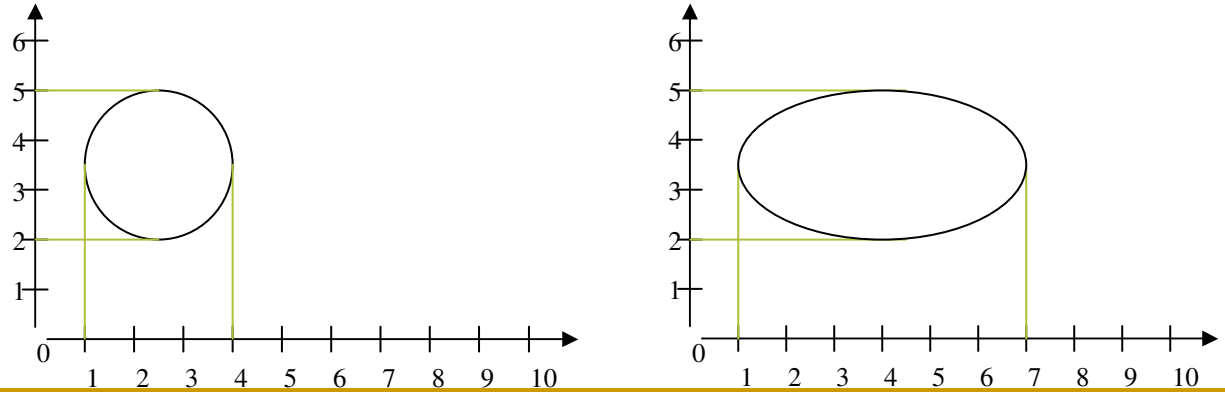


Примитиви

- Окръжност като полигон
  - окръжността може да се апроксимира с полигон с много страни (>15)

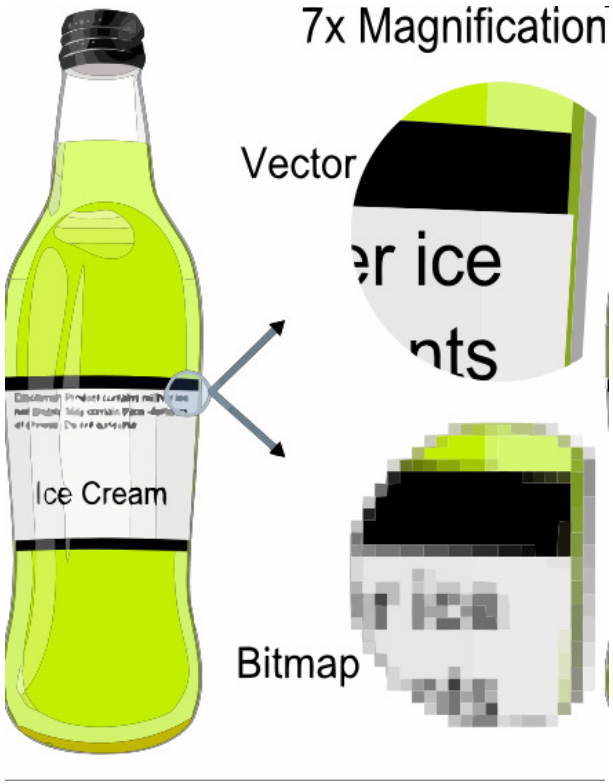


- Елипса
  - окръжност, мащабирана по x или y



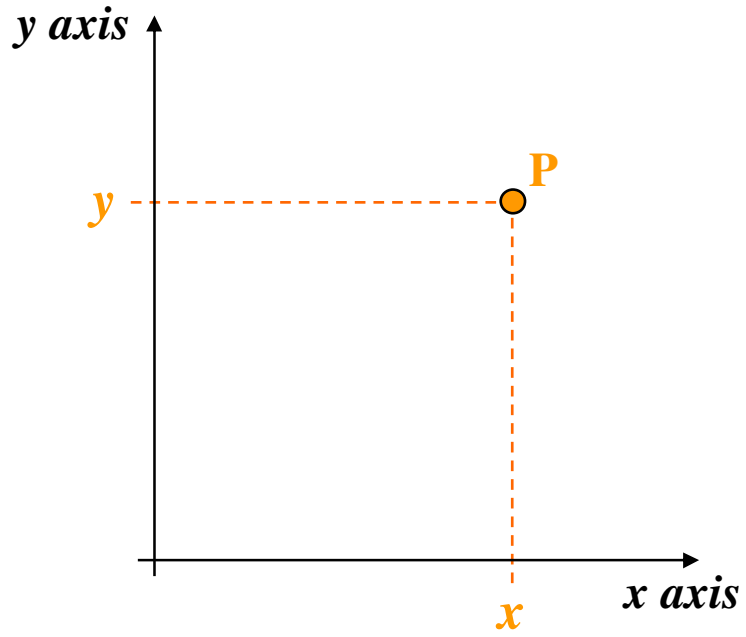
Примитиви

- Текст
  - растерен шрифт (bitmap font)
  - контурен шрифт (stroke font)
- Атрибути на текст
  - форма на символите (font)
  - цвят (color)
  - размер (size)
  - междуредие (spacing)
  - ориентация (orientation)

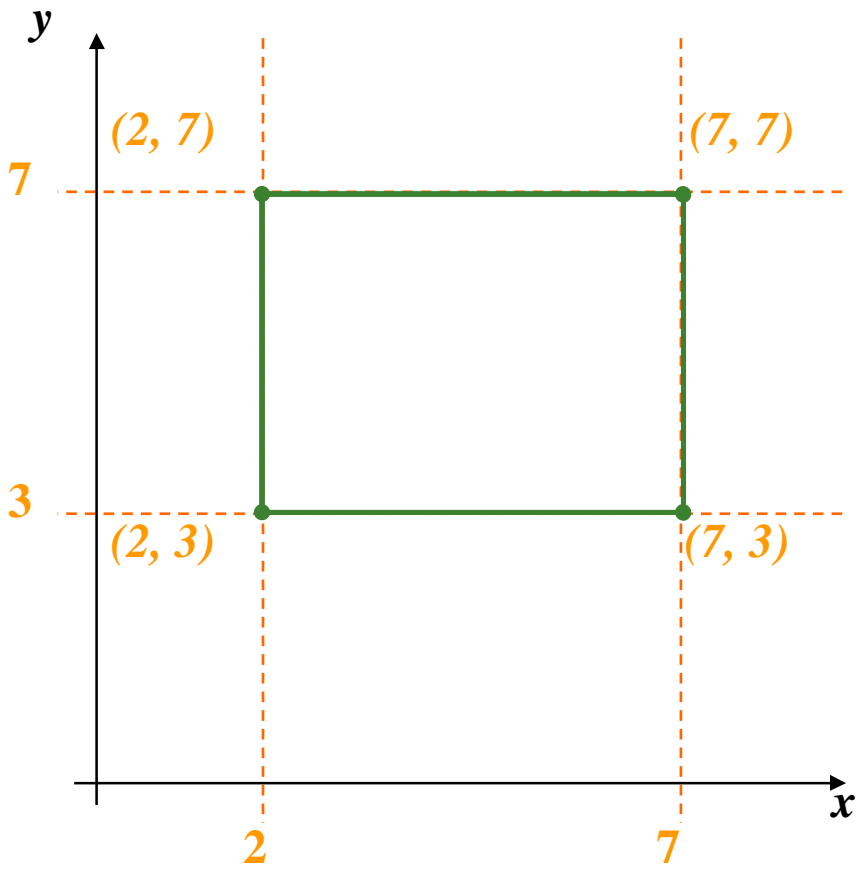


Координатна система – 2D

- Създаване на сцена – чрез прости геометрични примитиви
- В двумерното пространство (2D) се използва двумерна Декартова координатна система
- Всички обекти се задават чрез координатни двойки

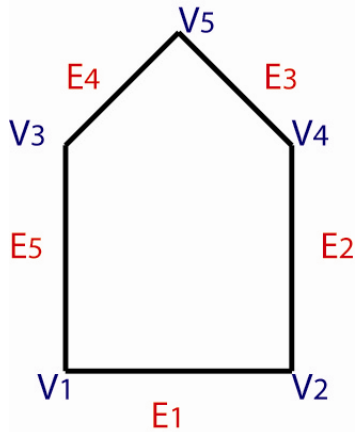


Координатна система – 2D



Координатна система – 2D

- Представяне на примитиви в 2D
  - таблица с възли и таблица с линии
    - всеки възел се среща веднъж
    - всяка линия е подредена двойка с индекси на възли



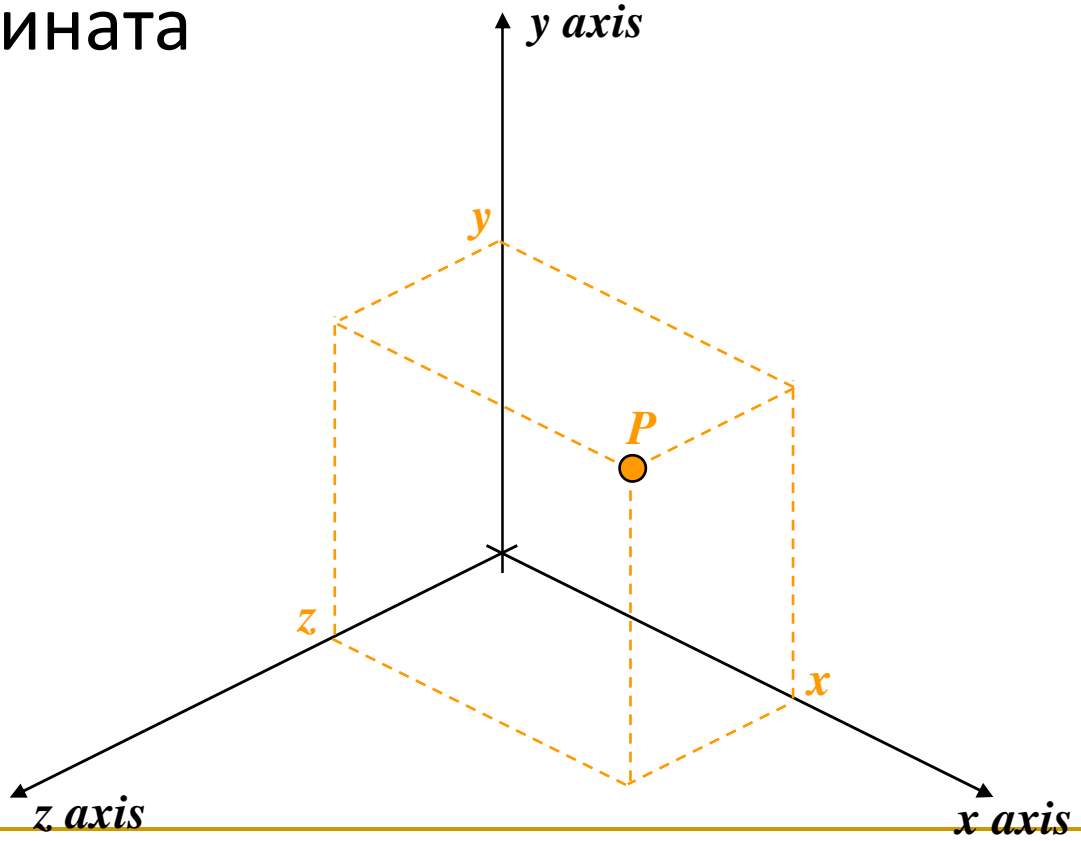
  - достатъчна информация за визуализиране на примитивите и изпълнение на прости операции с тях
  - реда на линиите в таблицата не е от значение
    - съществува конвенция за ред в посока обратна на часовниковата стрелка

Vertex	
1	(0,0)
2	(1,0)
3	(0,1)
4	(1,1)
5	(0.5,1.5)

Edge	
1	(1,2)
2	(2,4)
3	(4,5)
4	(5,3)
5	(3,1)

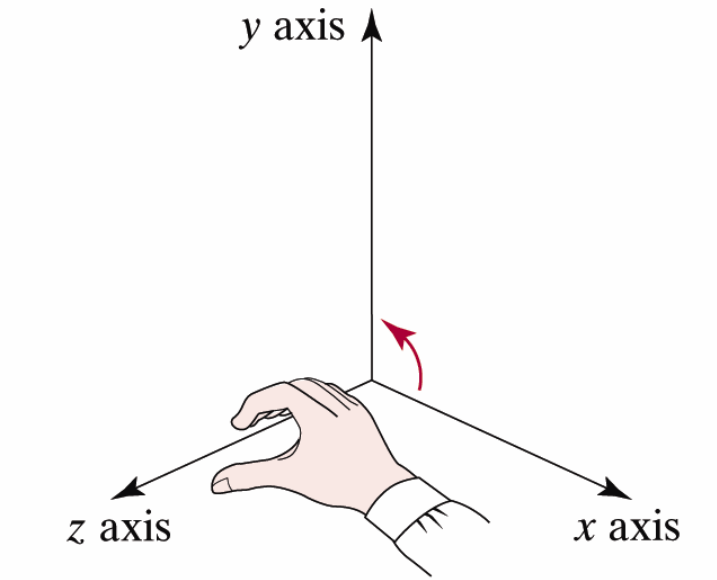
Координатна система – 3D

- За тримерни сцени – добавя се трета координата

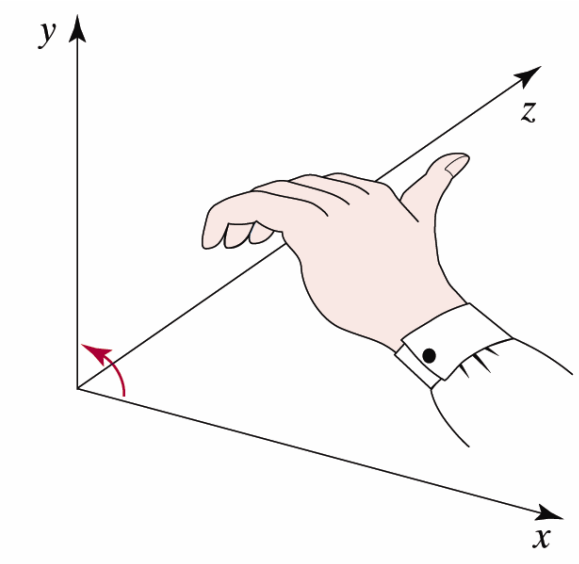


Координатна система – 3D

- Два варианта за 3D координатна система



дясно ориентирана КС



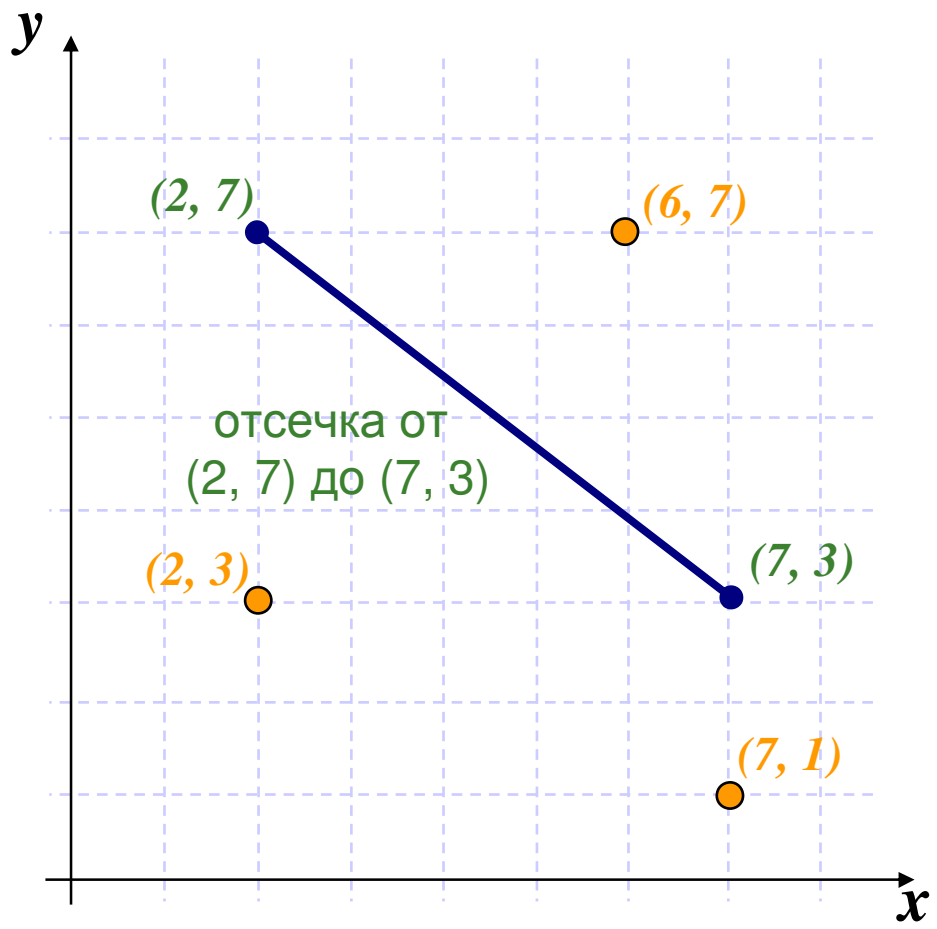
ляво ориентирана КС

Hearn & Baker, Computer Graphics with OpenGL, 2004

Точки и линии

- Точки
  - точка в двумерното пространство се задава като подредена двойка от координати (x, y)
  - точка в тримерното пространство се задава като подредена тройка от координати (x, y, z)
- Линии
  - отсечка се дефинира чрез начална и крайна точка
  - в 2D: от (x_{start}, y_{start}) до (x_{end}, y_{end})
  - в 3D: от $(x_{start}, y_{start}, z_{start})$ до $(x_{end}, y_{end}, z_{end})$

Точки и линии



Уравнение на линия

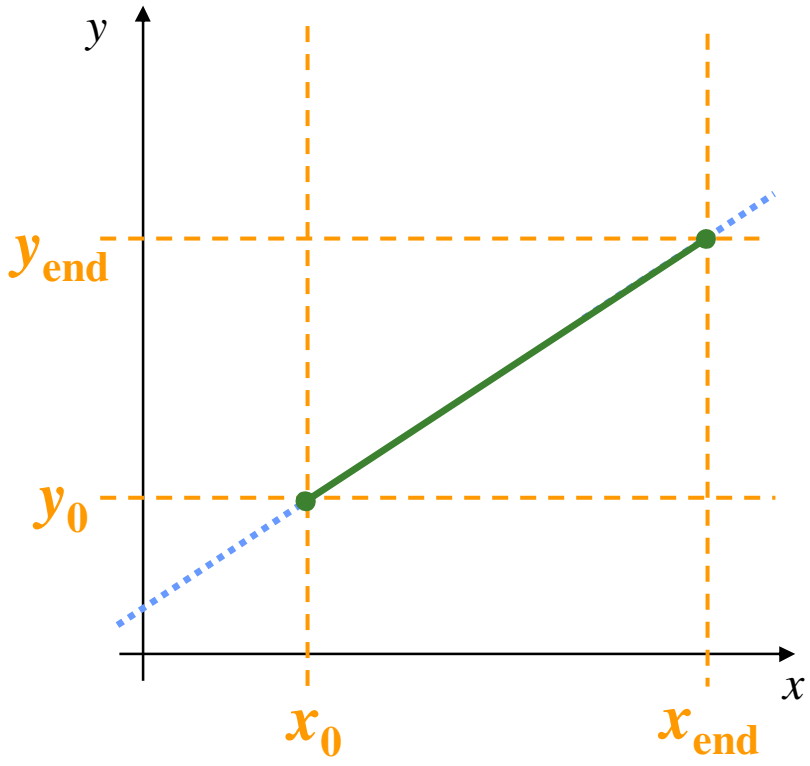
- Уравнение на линия

$$y = m \cdot x + b$$

- където

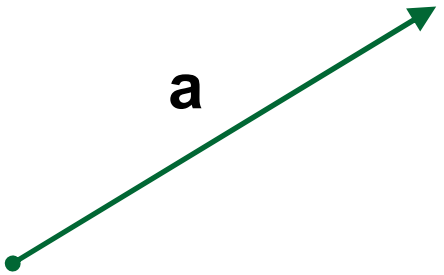
$$m = \frac{y_{end} - y_0}{x_{end} - x_0}$$

$$b = y_0 - m \cdot x_0$$



Вектор

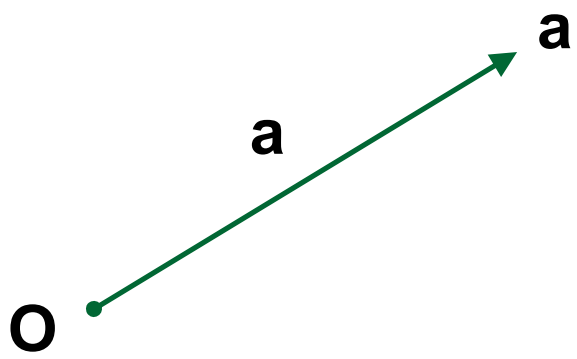
- Математическа величина, която има
  - големина (magnitude)
  - посока (direction)



- Математическо означение: **a**
- Графично изображение
  - стрелка с дължина равна на големината на вектора, сключваща ъгъл с положителната посока на оста x и определена от посоката на вектора

Вектор

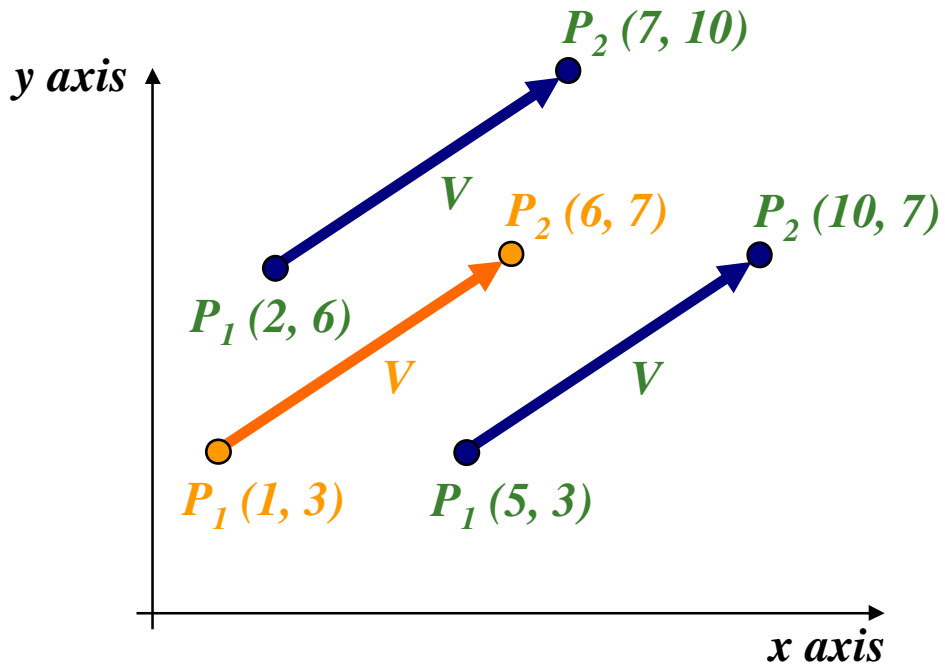
- Може да се интерпретира като преместване от една точка в друга



- Може да се интерпретира като точка, ако се определи начална точка

Вектор – 2D

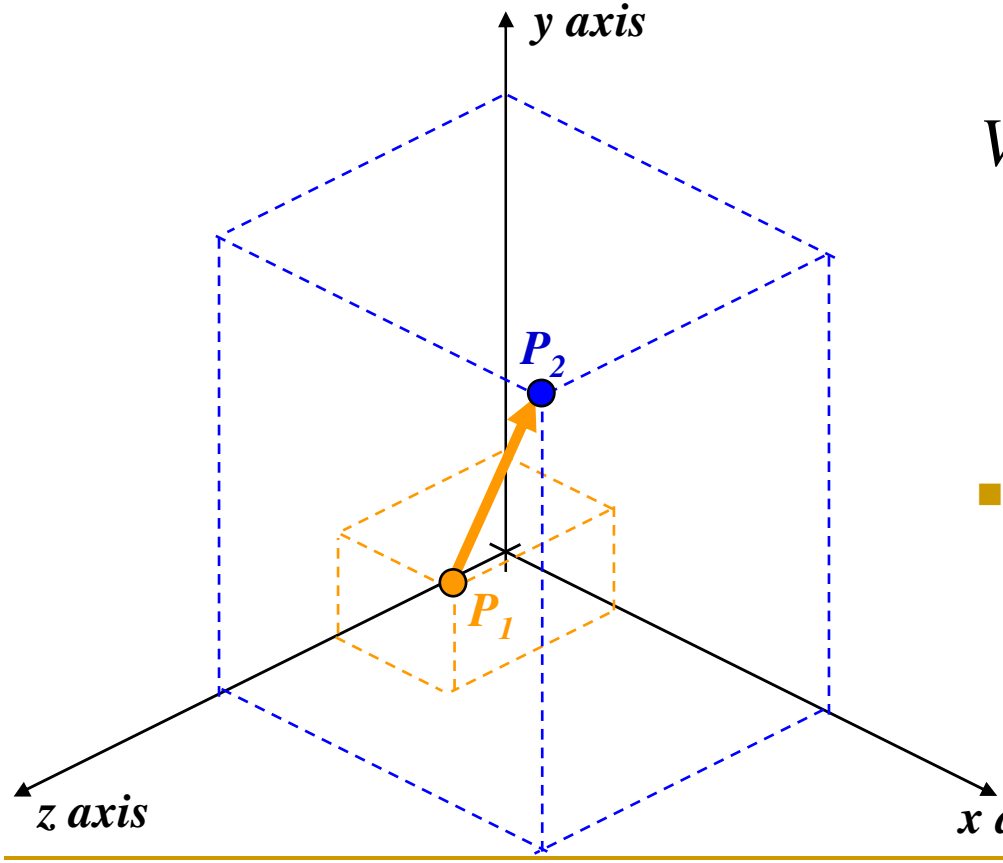
- Определяне на вектор между две точки
  - изваждане на координатите им



$$V = P_2 - P_1$$
$$= (x_2 - x_1,\ y_2 - y_1)$$
$$= (6 - 1,\ 7 - 3)$$
$$= (5,\ 4)$$

Вектор – 3D

- Определяне на вектор между две точки в 3D



$$V = P_2 - P_1$$

$$= (x_2 - x_1, y_2 - y_1, z_2 - z_1)$$

$$= (V_x, V_y, V_z)$$

- Вектор от (2, 1, 3) до (7, 10, 5) се определя

$$= (7 - 2, 10 - 1, 5 - 3)$$

$$= (5, 9, 2)$$

Големина на вектор

- Големина на вектор
  - скаларна величина **V** = | *V* |
  - в 2D: $|V| = \sqrt{V_x^2 + V_y^2}$
  - в 3D: $|V| = \sqrt{V_x^2 + V_y^2 + V_z^2}$

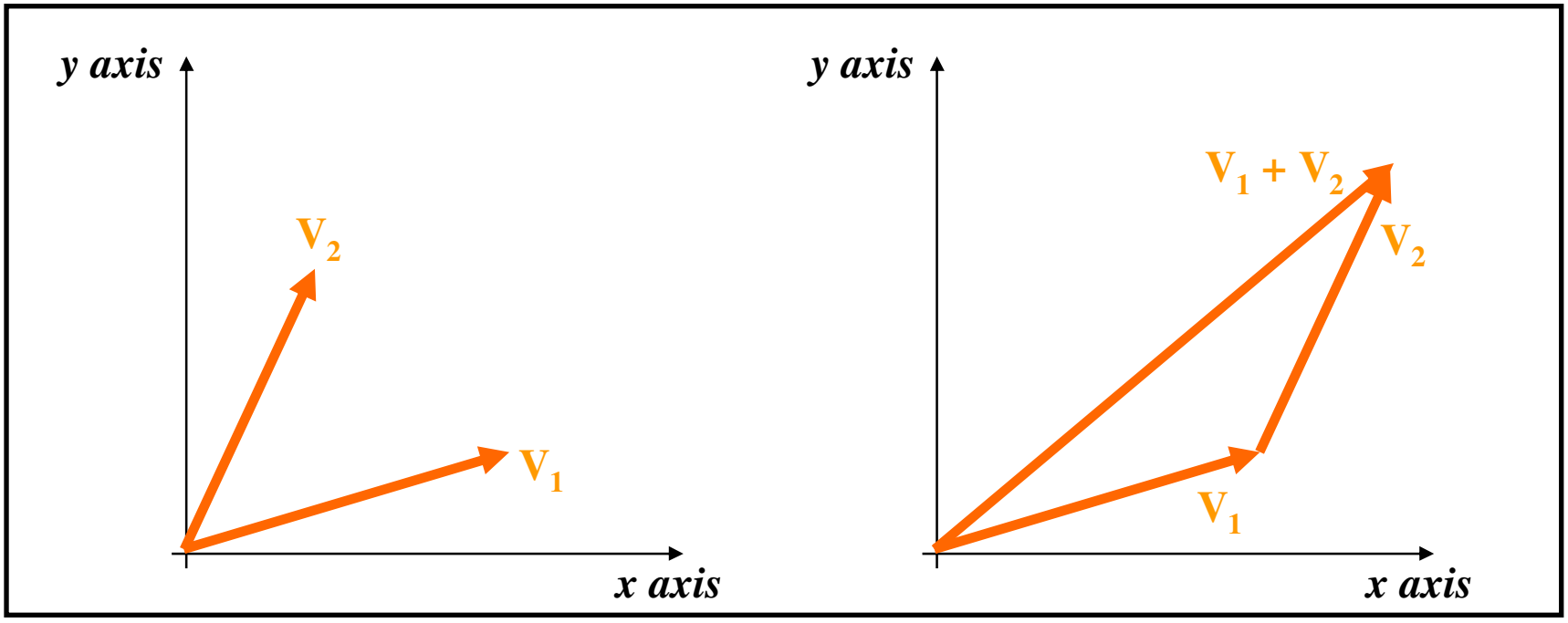
Големина на вектор

- Единичен вектор
  - всеки вектор с големина 1
- Нулев вектор
  - вектор с големина 0 и недефинирана посока
- Два вектора са еднакви тогава и само тогава когато имат еднаква големина и еднаква посока

Векторна аритметика

- Сума на два вектора

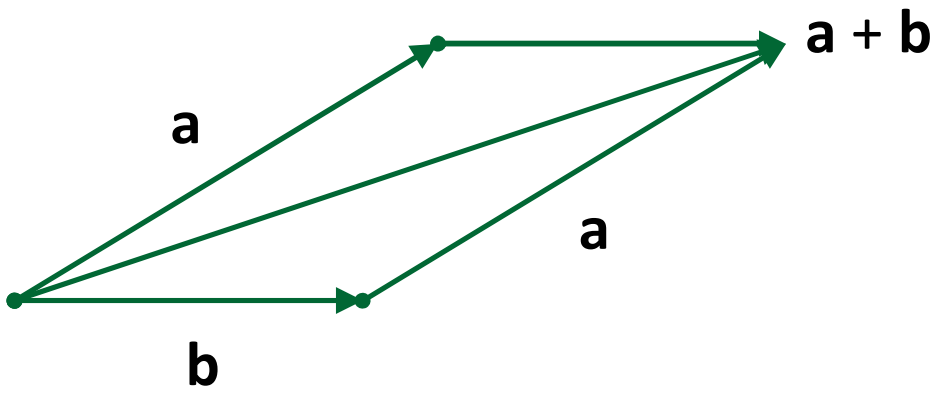
$$V_1 + V_2 = (V_{1x} + V_{2x}, V_{1y} + V_{2y})$$



Векторна аритметика

- Сума на два вектора
  - комутативна

$$\mathbf{a} + \mathbf{b} = \mathbf{b} + \mathbf{a}$$



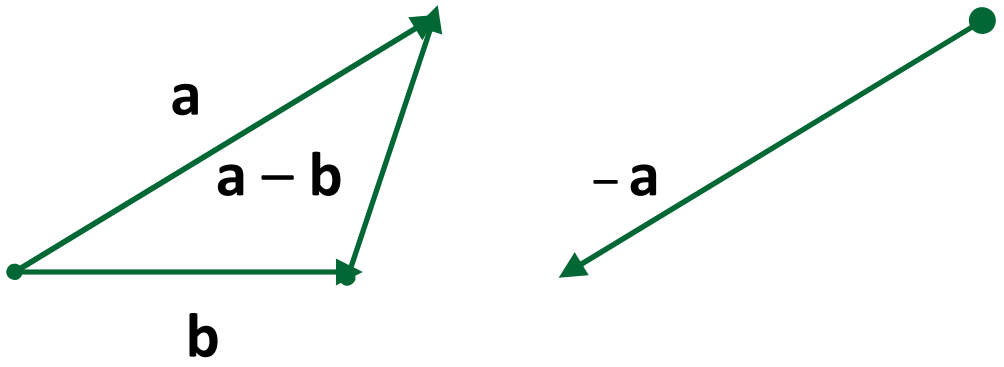
Векторна аритметика

- Разлика на два вектора
  - унарен минус

$$-\mathbf{a} + \mathbf{a} = 0$$

  - разлика на вектори може да се дефинира като

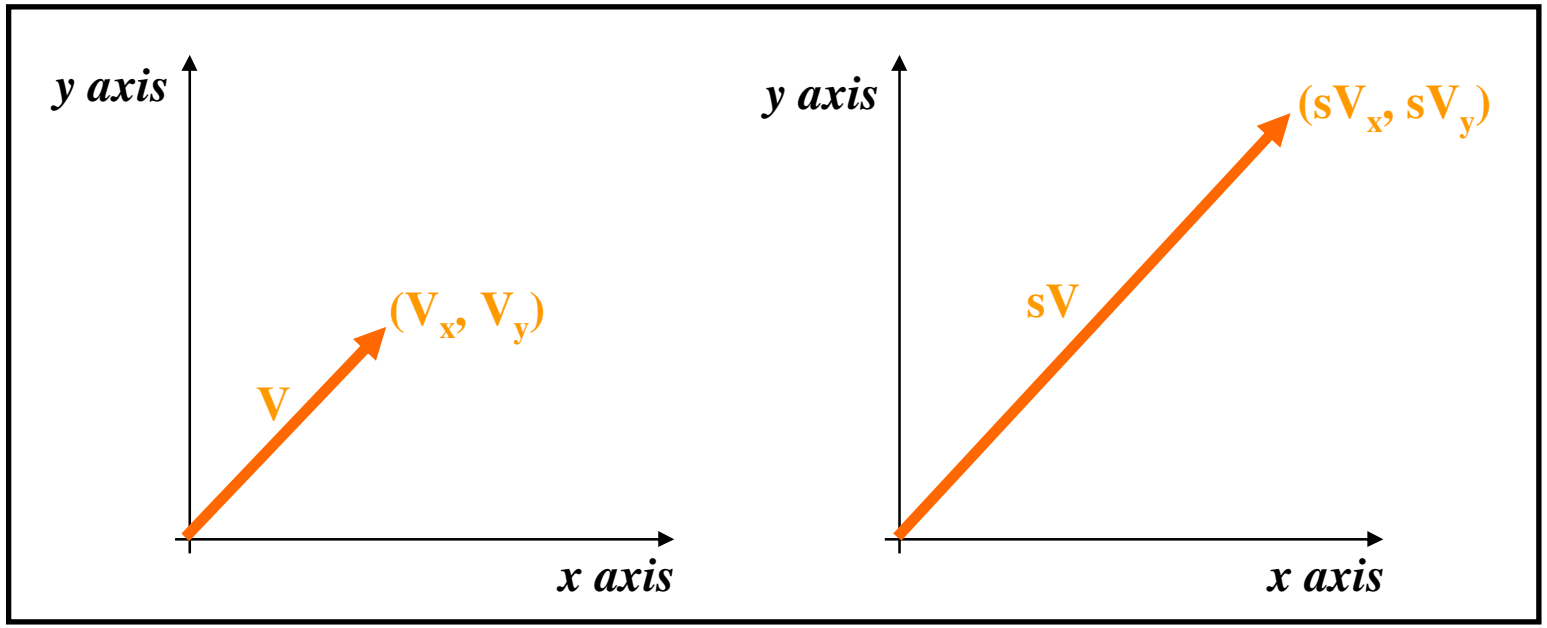
$$\mathbf{a} - \mathbf{b} \equiv -\mathbf{b} + \mathbf{a}$$



Векторна аритметика

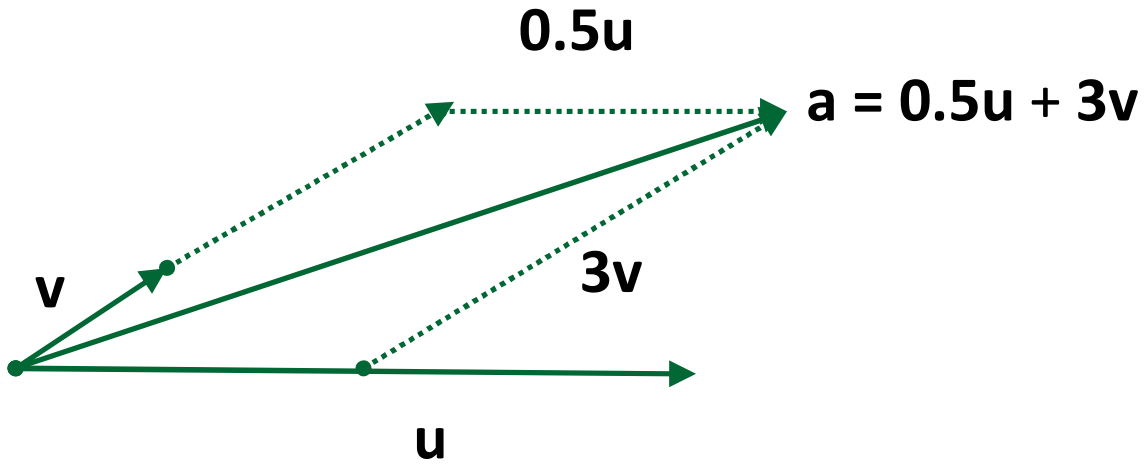
- Умножение на вектор със скалар

$$sV = (sV_x, sV_y)$$



Векторна аритметика

- Умножение на вектор със скалар
  - в 2D всеки вектор може да се представи като ***линейна комбинация*** (претеглена сума) от произволни ***два*** не-паралелни ***базови*** вектора



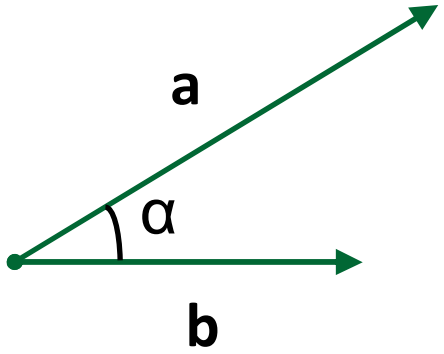
  - в 3D са необходими ***три*** не-паралелни, не-планарни базови вектора

Векторна аритметика

- Скаларно произведение
  - dot product

$$\mathbf{a} \bullet \mathbf{b} = |\mathbf{a}| . |\mathbf{b}| . \cos \alpha$$

където α е ъгъла между двата вектора



  - скаларното произведение на два вектора е ***скаларно число***

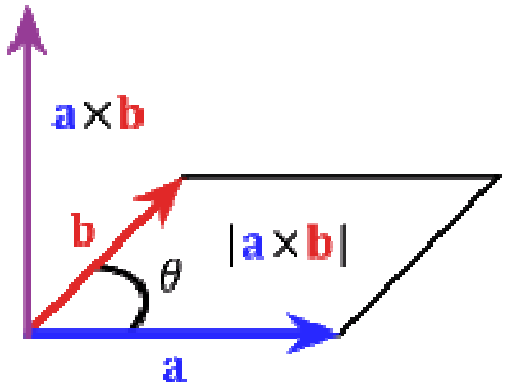
$$\mathbf{a} \bullet \mathbf{b} = a_x b_x + a_y b_y$$

Векторна аритметика

- Векторно произведение
  - cross product

$$\mathbf{a} \times \mathbf{b} = (a_y b_z - a_z b_y,\ a_z b_x - a_x b_z,\ a_x b_y - a_y b_x)$$

  където $a(a_x, a_y, a_z)$ и $b(b_x, b_y, b_z)$ са двата вектора



  - векторното произведение на два вектора е ***вектор***, перпендикулярен и на двата вектора

Матрични операции

- Умножение на матрица със скалар

$$s.\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} = \begin{bmatrix} s.a & s.b & s.c \\ s.d & s.e & s.f \\ s.g & s.h & s.i \end{bmatrix}$$

- Пример

$$3.\begin{bmatrix} 2 & 4 & 6 \\ 8 & 10 & 12 \\ 14 & 16 & 18 \end{bmatrix} = \begin{bmatrix} 6 & 12 & 18 \\ 24 & 30 & 36 \\ 42 & 48 & 54 \end{bmatrix}$$

Матрични операции

- Сума на матрици

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} + \begin{bmatrix} r & s & t \\ u & v & w \\ x & y & z \end{bmatrix} = \begin{bmatrix} a+r & b+s & c+t \\ d+u & e+v & f+w \\ g+x & h+y & i+z \end{bmatrix}$$

двете матрици трябва да са с еднакъв размер

- Пример

$$\begin{bmatrix} 2 & 4 & 6 \\ 8 & 10 & 12 \\ 14 & 16 & 18 \end{bmatrix} + \begin{bmatrix} 3 & 5 & 7 \\ 9 & 11 & 13 \\ 15 & 17 & 19 \end{bmatrix} = \begin{bmatrix} 5 & 9 & 13 \\ 17 & 21 & 25 \\ 29 & 33 & 37 \end{bmatrix}$$

Матрични операции

- Умножение на матрици
  - за да се умножат матриците A и B броя колони на A трябва да е равен на броя редове на B
- Произведението на матрица A ($m \times n$) с матрица B ($p \times q$) е матрица C ($m \times q$)

$$C = AB$$

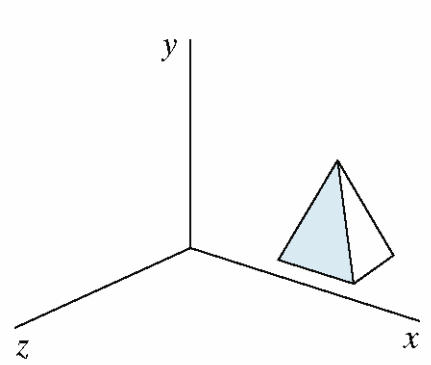
- Елементите на матрицата C се изчисляват като

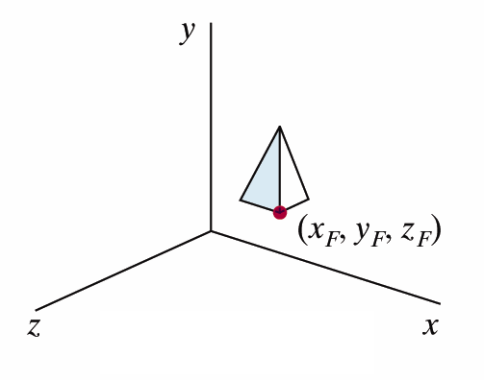
$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{ki}$$

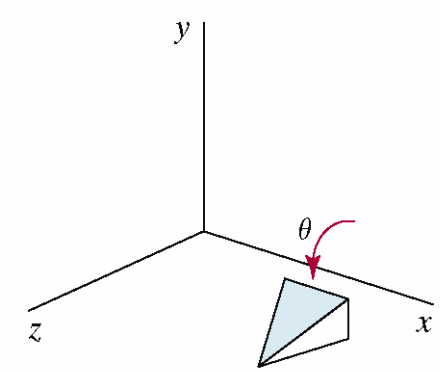
- умножението на матрици не е комутативно: $AB \neq BA$

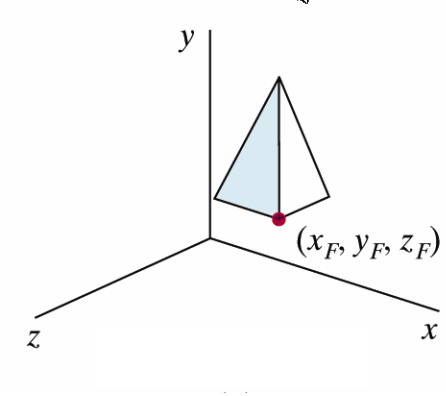
Трансформации

- Трансформациите в компютърната графика се използват за промяна на разположението и формата на обектите
  - преместване, ротиране, мащабиране









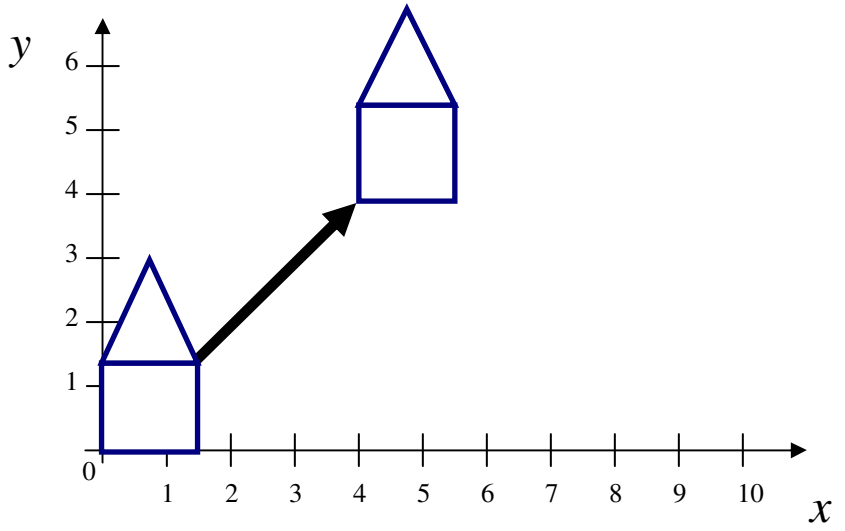
Hearn & Baker, Computer Graphics with OpenGL, 2004

Транслация

- Преместване на обект от една позиция в друга

$$x_{new} = x_{old} + dx$$
$$y_{new} = y_{old} + dy$$

където точка с координати (x_{old}, y_{old}) се премества в нова позиция с координати (x_{new}, y_{new}) с коефициенти на транслация dx , dy



Позицията на обекта се премества спрямо началото на КС

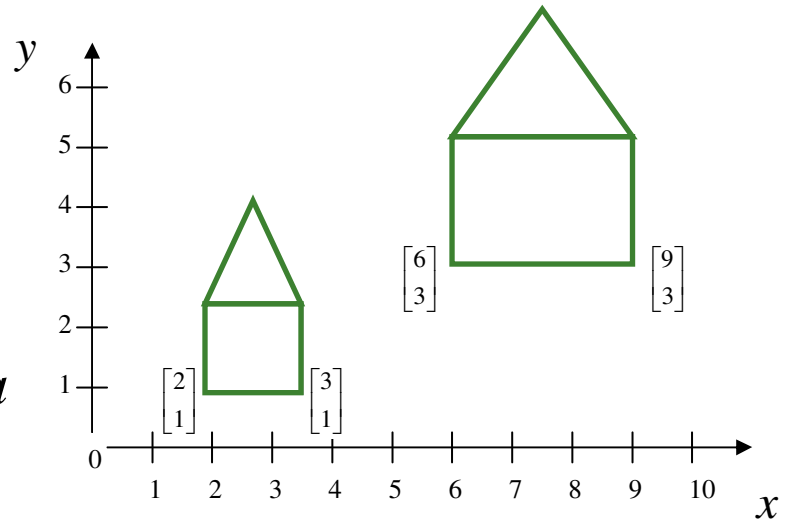
Мащабиране

- Промяна на размерите на обекта
  - всички координати се умножават със скаларна стойност

$$x_{new} = Sx \,.\, x_{old}$$

$$y_{new} = Sy \,.\, y_{old}$$

  - *променя се не само размера а и позицията на обекта*



Позицията на обекта се премества спрямо началото на КС

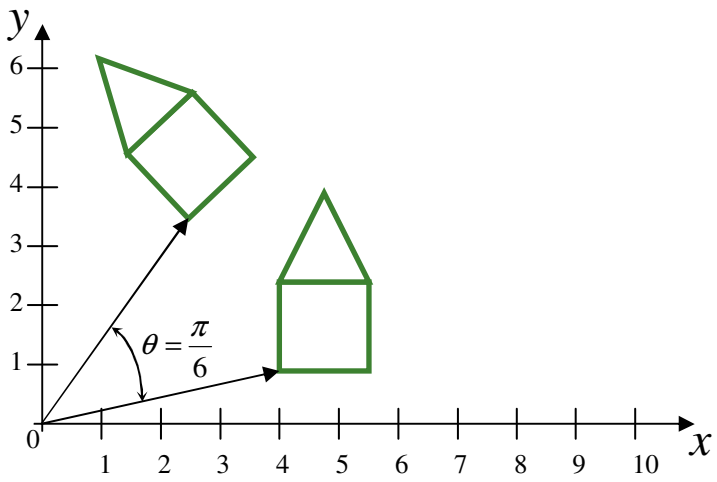
Ротация

- Завъртане на обекта
  - ротация на всички координати на определен ъгъл θ

$$x_{new} = x_{old} \,.\, \cos\theta - y_{old} \,.\, \sin\theta$$

$$y_{new} = x_{old} \,.\, \sin\theta + y_{old} \,.\, \cos\theta$$

  - *ротацията е спрямо началото на координатната система*



Трансформации

- Транслация
  - изометрична
    - запазва дължините
  - конформална
    - запазва ъглите
- Мащабиране
  - не запазва дължините
  - не запазва ъглите
    - освен когато е с еднакви коефициенти
- Ротация
  - запазва дължините на обектите
  - запазва ъглите между части от обектите

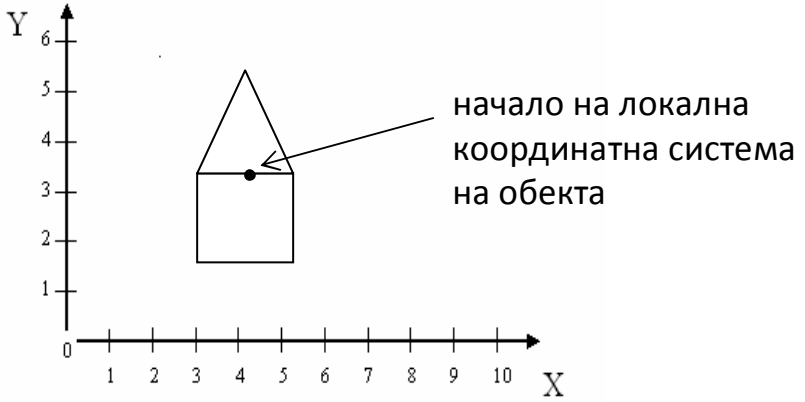
Матрично представяне

$$\mathsf{P} = \begin{bmatrix} x \\ y \end{bmatrix}, \mathsf{P}_{new} = \begin{bmatrix} x_{new} \\ y_{new} \end{bmatrix}$$

- транслация $\mathsf{T} = \begin{bmatrix} dx \\ dy \end{bmatrix}, \mathsf{P}_{new} = \mathsf{T} + \mathsf{P}$
- мащабиране $\mathsf{S} = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix}, \mathsf{P}_{new} = \mathsf{S}.\mathsf{P}$
- ротация $\mathsf{R} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}, \mathsf{P}_{new} = \mathsf{R} \cdot \mathsf{P}$

Трансформации

- Трансформациите са спрямо началото на координатната система
- За да се мащабира и/или ротира обект, който не се намира в началото на координатната система
  - транслира се в началото на КС
  - мащабира се и/или ротира
  - транслира се обратно



- Необходимост от комбиниране на последователни трансформации

Трансформации

- Комбиниране на последователни трансформации
  - транслацията не се представя като матрично умножение
  - комбинация от трансформации не може да се представи като матрично умножение

- **Хомогенни координати**
  - позволяват представянето на комбинации от трансформации в матрична форма чрез умножение с матрици

Хомогенни координати

- Дадена е т.P с координати $(x,\ y)$
- Хомогенни координати на точката P

  $$(x_h,\ y_h,\ h)$$

  където параметъра $h \neq 0$ е такъв, че

  $$x = \frac{x_h}{h} \qquad y = \frac{y_h}{h}$$

- Всяка точка може да се опише с $(hx,\ hy,\ h),\ h \neq 0$
- Ако $h=1$, то всяка точка се представя с $(x,\ y,\ 1)$

Хомогенни координати

- Използването на хомогенни координати в компютърна графика позволява чрез матрично представяне на трансформациите необходимите изчисления да се извършват с матрични умножения

- Хомогенни координати на 2D точка $\mathsf{P} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$

Хомогенни координати

- Транслация

$$\begin{bmatrix} 1 & 0 & dx \\ 0 & 1 & dy \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 1.x+0.y+dx.1 \\ 0.x+1.y+dy.1 \\ 0.x+0.y+1.1 \end{bmatrix} = \begin{bmatrix} x+dx \\ y+dy \\ 1 \end{bmatrix}$$

- Матрица на транслация

$$\begin{bmatrix} 1 & 0 & dx \\ 0 & 1 & dy \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathsf{P}_{new} = \mathsf{T}(dx, dy).\mathsf{P}$$

Хомогенни координати

- Мащабиране

$$\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} s_x . x \\ s_y . y \\ 1 \end{bmatrix}$$

- Матрица на мащабиране

$$\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{P}_{new} = \mathbf{S}(s_x, s_y).\mathbf{P}$$

Хомогенни координати

- Ротация

$$\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta.x - \sin\theta.y \\ \sin\theta.x + \cos\theta.y \\ 1 \end{bmatrix}$$

- Матрица на ротация

$$\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{P}_{new} = \mathbf{R}(\theta).\mathbf{P}$$

Хомогенни координати

- Обобщена матрица на 2D трансформации

$$T = \left[\begin{array}{cc|c} a & b & m \\ c & d & n \\ \hline p & q & s \end{array}\right]$$

  - a, b, c, d определят ротацията, мащабирането (както и огледален образ, еластични деформации)
  - m, n определят транслацията
  - p, q са 0, 0 до момента

Хомогенни координати

- Ако p, q не са 0, 0

$$\begin{bmatrix} X \\ Y \\ w \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ p & q & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ px + qy + 1 \end{bmatrix}$$

- то резултантната точка е върху равнина в 3D пространството, определена от

$$w = px + qy + 1$$

Обратни трансформации

- Транслация $\mathsf{T}^{-1} = \begin{bmatrix} 1 & 0 & -dx \\ 0 & 1 & -dy \\ 0 & 0 & 1 \end{bmatrix}$
- Мащабиране $\mathsf{S}^{-1} = \begin{bmatrix} 1/s_x & 0 & 0 \\ 0 & 1/s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$
- Ротация $\mathsf{R}^{-1} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Примери

- Транслация на [1,3] с [7,9]

$$\begin{bmatrix} 1 & 0 & 7 \\ 0 & 1 & 9 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 3 \\ 1 \end{bmatrix} = \begin{bmatrix} 8 \\ 12 \\ 1 \end{bmatrix}$$

- Мащабиране на [2,3] с коефициент 5 в посока X и с коефициент 10 в посока Y

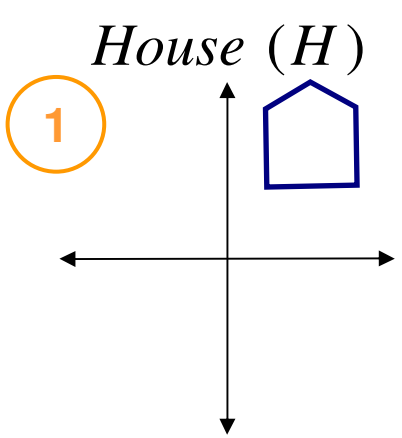
$$\begin{bmatrix} 5 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 2 \\ 3 \\ 1 \end{bmatrix} = \begin{bmatrix} 10 \\ 30 \\ 1 \end{bmatrix}$$

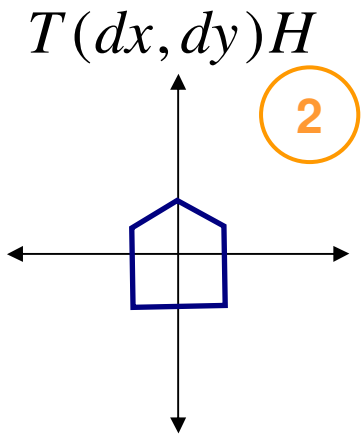
- Ротация на [2,2] на 90° (π/2)

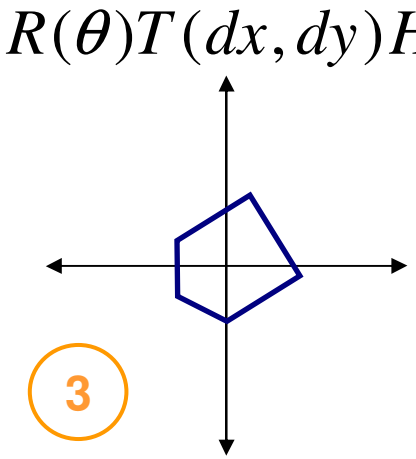
$$\begin{bmatrix} \cos(\pi/2) & -\sin(\pi/2) & 0 \\ \sin(\pi/2) & \cos(\pi/2) & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 2 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 2 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} -2 \\ 2 \\ 1 \end{bmatrix}$$

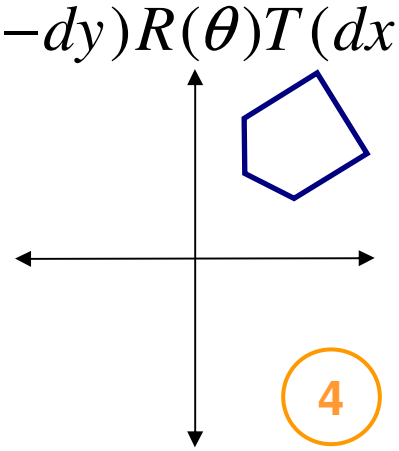
Комбинирани трансформации

- За да се ротира обект, който не се намира в началото на КС
  - транслира се в началото на КС (2)
  - ротира се (3)
  - транслира се обратно (4)









Комбинирани трансформации

- Трите матрици на отделните трансформации могат да се обединят в една

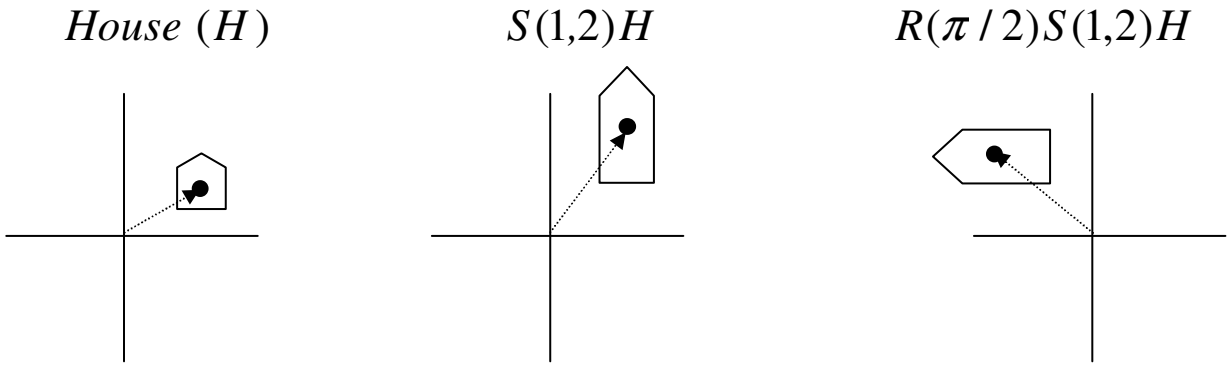
$$\begin{bmatrix} 1 & 0 & -dx \\ 0 & 1 & -dy \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & dx \\ 0 & 1 & dy \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$P_{new} = T(-dx,-dy)R(\theta)T(dx,dy)P$$

- умножението на матрици не е комутативно, затова редът има значение

Комбинирани трансформации

- Как ще изглежда матрицата на трансформация, с която обекта се мащабира по y с коефициент 2 и се ротира на 90°



- умножението на матрици не е комутативно, затова редът има значение

Комбинирани трансформации

- Две последователни транслации
  - P се транслира в P′ с T (T_{x1}, T_{y1})
    - P′ = T (T_{x1}, T_{y1}). P
  - P′ се транслира в P″ с T (T_{x2}, T_{y2})
    - P″ = T (T_{x2}, T_{y2}). P′
- Комбинирана матрица на трансформацията
  - P″ = T (T_{x2}, T_{y2}). P′= T (T_{x2}, T_{y2}).[(T_{x1}, T_{y1}). P]

    P″ = [T (T_{x2}, T_{y2}).(T_{x1}, T_{y1})]. P

- Две последователни транслации са ***адитивни*** операции

$$T(T_{x2}, T_{y2}).(T_{x1}, T_{y1}) = \begin{bmatrix} 1 & 0 & T_{x1} + T_{x2} \\ 0 & 1 & T_{y1} + T_{y2} \\ 0 & 0 & 1 \end{bmatrix}$$

Комбинирани трансформации

- Две последователни мащабирания

$$P'' = [S\,(S_{x2}, S_{y2}).(S_{x1}, S_{y1})].\,P$$

- Две последователни мащабирания са ***мултипликативни*** операции

$$S\,(S_{x2}, S_{y2}).(S_{x1}, S_{y1}) = \begin{bmatrix} S_{x1}.S_{x2} & 0 & 0 \\ 0 & S_{y1}.S_{y2} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Комбинирани трансформации

- Две последователни ротации

$$P'' = [R(\theta_2).\ R(\theta_1)].\ P$$

- Две последователни ротации са ***адитивни*** операции

$$R(\theta_2).R(\theta_1) = \begin{bmatrix} \cos(\theta_1+\theta_2) & -\sin(\theta_1+\theta_2) & 0 \\ \sin(\theta_1+\theta_2) & \cos(\theta_1+\theta_2) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Комбинирани трансформации

- Ротация около произволна точка

$$P' = T(x_1, y_1)R(\theta)T(-x_1, -y_1)P$$

$$\begin{bmatrix} \cos\theta & -\sin\theta & x_1(1-\cos\theta)+y_1\sin\theta \\ \sin\theta & \cos\theta & y_1(1-\cos\theta)-x_1\sin\theta \\ 0 & 0 & 1 \end{bmatrix}$$

Комбинирани трансформации

- Мащабиране спрямо произволна точка

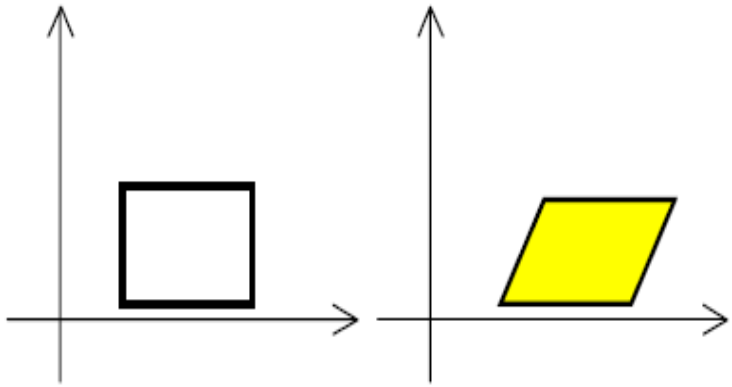
$$P' = T(x_1, y_1)S(S_x, S_y)T(-x_1, -y_1)P$$

$$\begin{bmatrix} S_x & 0 & x_1(1-S_x) \\ 0 & S_y & y_1(1-S_y) \\ 0 & 0 & 1 \end{bmatrix}$$

Други трансформации

- Еластични деформации (Shear)
  - по оста x с коефициент на деформация a

$$Sh_x = \begin{bmatrix} 1 & a & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

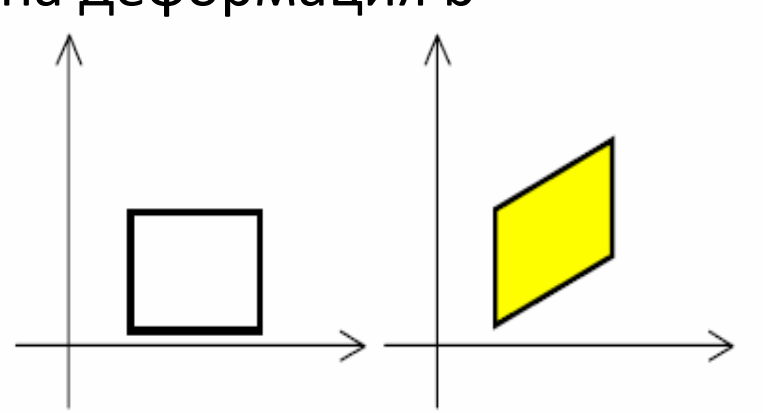


$$P' = Sh_x.P = \begin{bmatrix} x + ay \\ y \\ 1 \end{bmatrix}$$

Други трансформации

- Еластични деформации (Shear)
  - по оста y с коефициент на деформация b

$$Sh_y = \begin{bmatrix} 1 & 0 & 0 \\ b & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

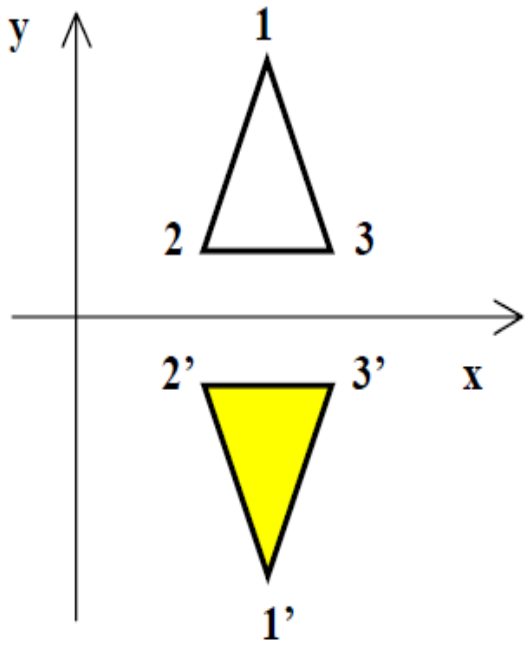


$$P' = Sh_y.P = \begin{bmatrix} x \\ y + bx \\ 1 \end{bmatrix}$$

Други трансформации

- Огледален образ (Reflection)
  - спрямо оста x

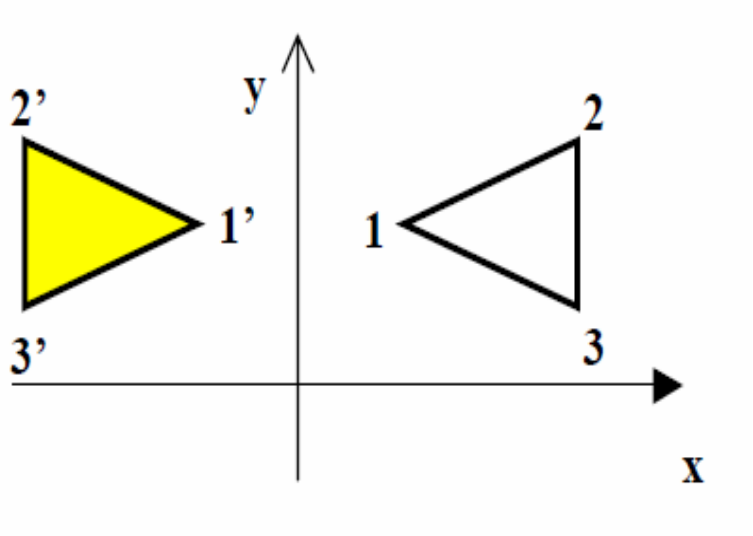
$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



Други трансформации

- Огледален образ (Reflection)
  - спрямо оста y

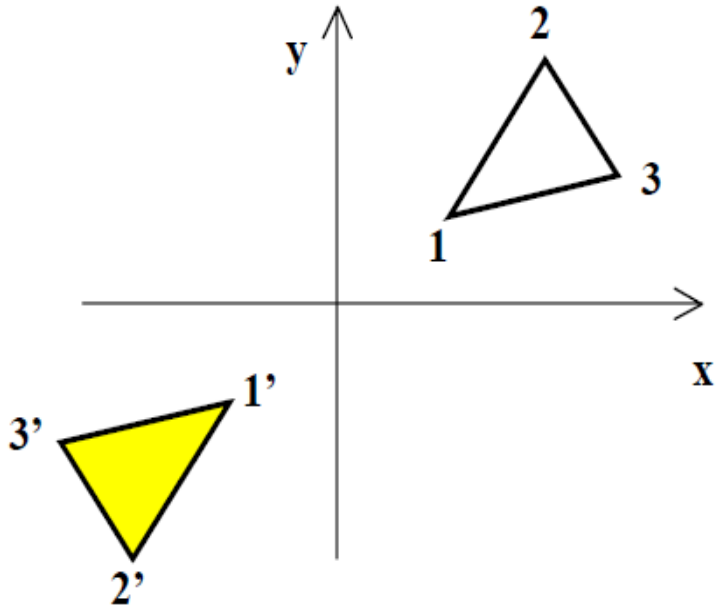
$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



Други трансформации

- Огледален образ (Reflection)
  - спрямо началото на координатната система
    - еквивалентно на ротация около началото на КС с ъгъл 180°

$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



Други трансформации

- Огледален образ (Reflection)
  - спрямо произволна точка P
    - еквивалентно на ротация около точката на 180°
  - спрямо произволна права:
    - правата се пренася така, че да минава през началото на КС
    - правата се ротира така, че да съвпадне с една от осите, например оста X
    - намира се огледалният образ спрямо избраната ос
    - изпълнява се обратна ротация
    - изпълнява се обратна транслация

OpenGL

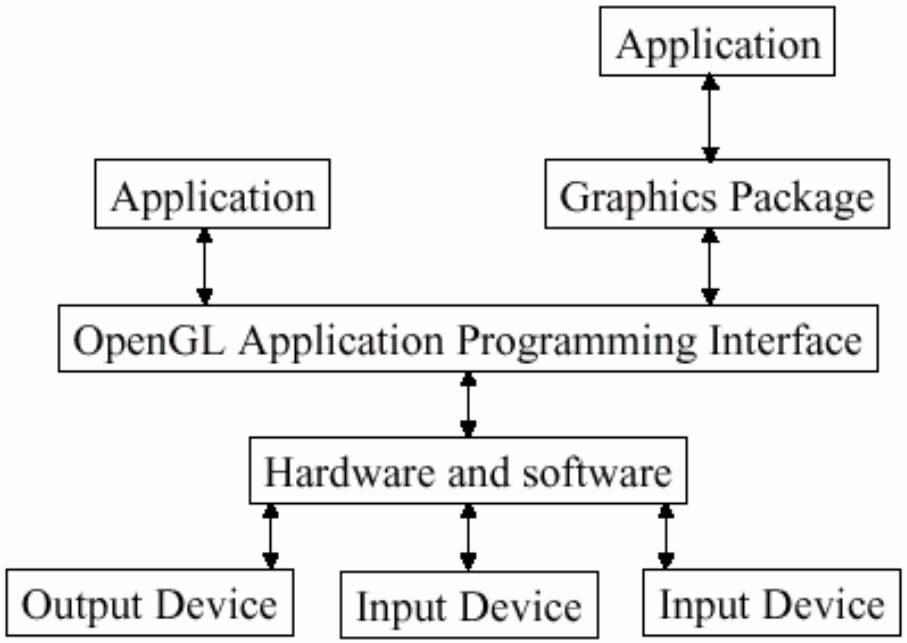
- Индустриален стандарт
- Стабилен
- Надежден
- Преносим
- Развиващ се
- Мащабируем
- Лесен
- Добре документиран API

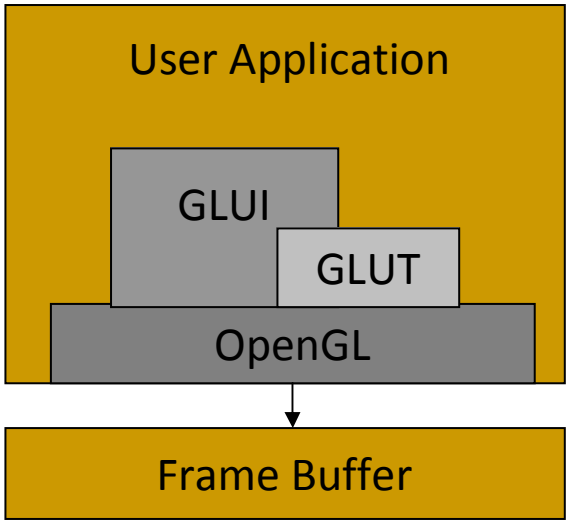
OpenGL

- отворен, много платформен, неутрален спрямо производители графичен стандарт
  - OpenGL Architecture Review Board (ARB)
    - създаден като независим консорциум през 1992
    - от 2006 е част от Khronos Group
    - участват компании като 3Dfx, 3Dlabs, ATI, Compaq, Evans & Sutherland, Hewlett-Packard, IBM, Intel, NVIDIA, Microsoft, SGI, Sun

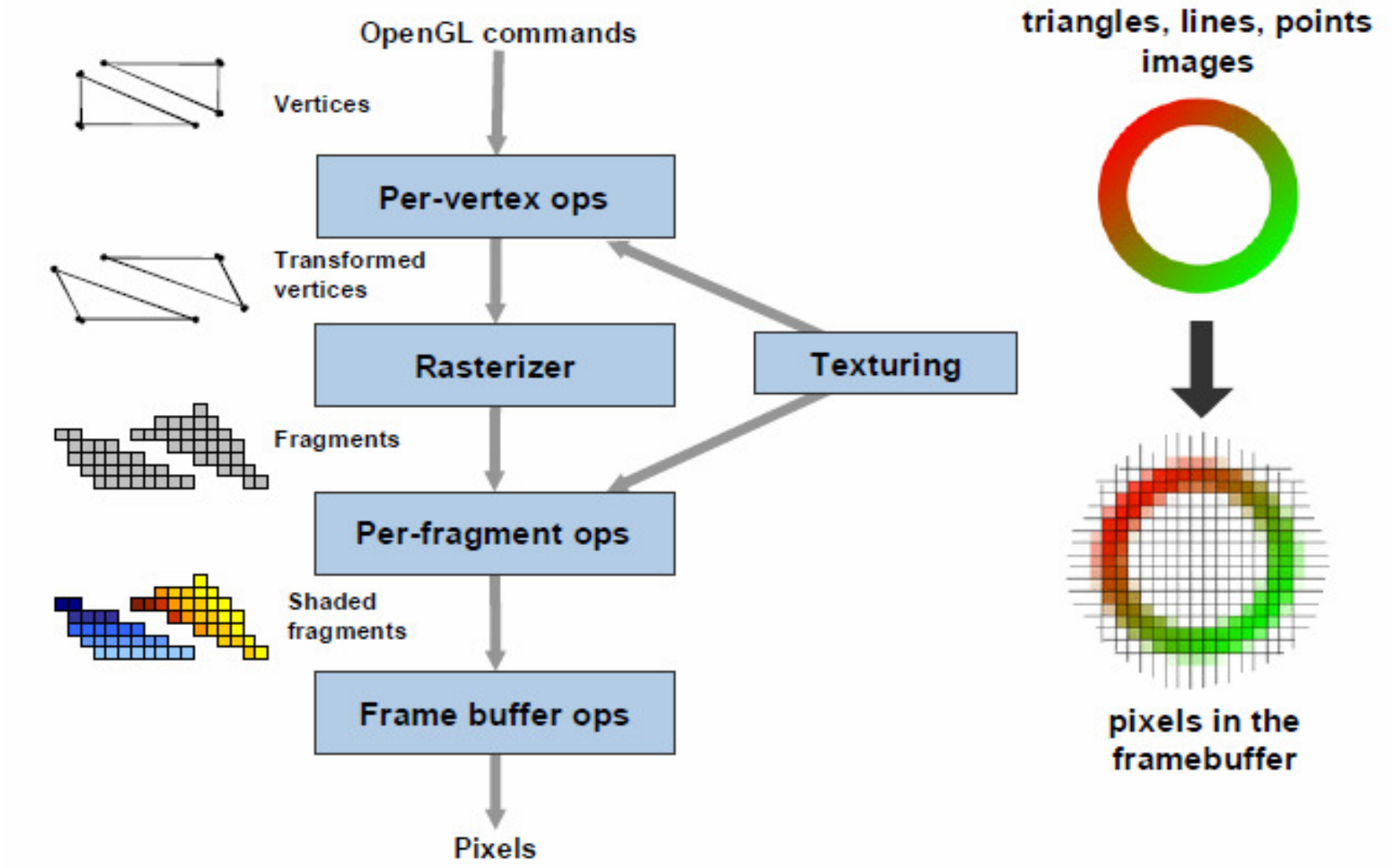
OpenGL

- интерфейс между потребителската програма и хардуерните устройства





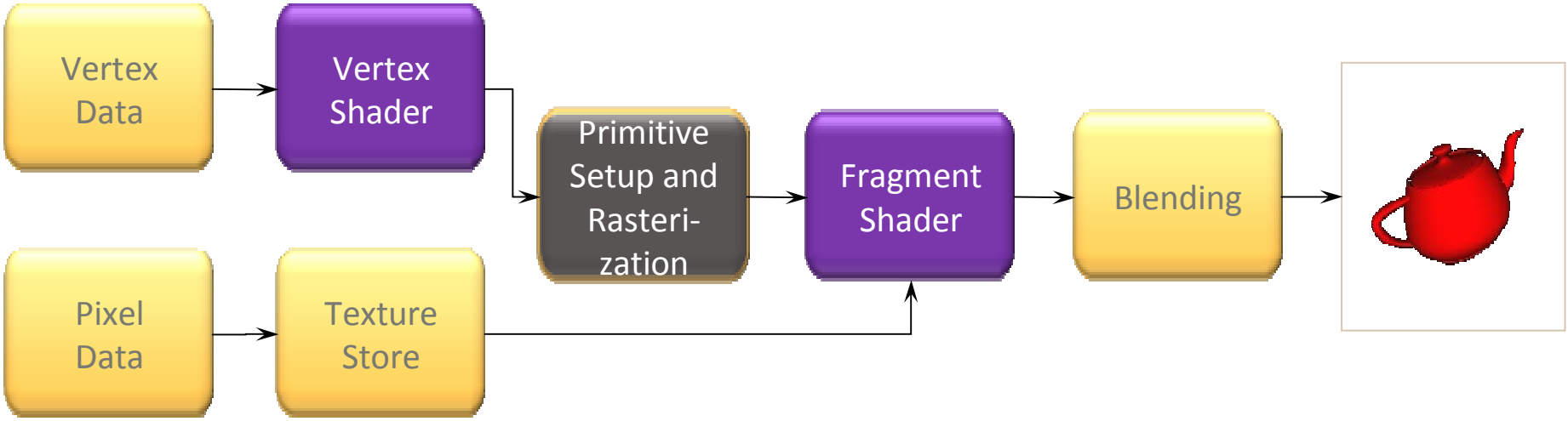
OpenGL Rendering Pipeline



OpenGL Rendering Pipeline

- “Frame Buffer”
  - съхранява генерираното изображение
  - честота на визуализиране може да е различна от честота на рендиране на изображение
- Single-buffer
  - директно генериране на изображение в буфера за визуализиране
- Double-buffer
  - визуализира се “front” буфер
  - генерира се в “back” буфер
  - разменят се двата буфера

OpenGL Rendering Pipeline



OpenGL

- Не съдържа операции за работа и управление на прозорци
  - отваряне на прозорец
  - управления на събития с прозорци
  - комуникация с потребител
  - комуникация с мениджър на прозорци
  - ...
- Независим
  - от дисплеи и устройства за визуализиране
  - от система за управление на прозорци
  - от операционна система

GLUT (OpenGL Utility Toolkit)

- използва се като интерфейс с различни системи за управление на прозорци
- Структура на приложение с OpenGL и GLUT
  - конфигуриране и отваряне на прозорец (GLUT)
  - инициализиране на OpenGL (OpenGL)
  - регистриране на callback функции (GLUT)
    - рендериране
    - промяна на размери
    - вход: клавиатура, мишка и др.
  - цикъл за обработване на събития (GLUT)

OpenGL

```
/* Example: draw randomly generated points */
#include <stdlib.h>
#include <GL/glut.h>

#define Height 400
#define Width 400

void display(void)
{
    int x, y;

    //a. generate a random point
    x = rand() % Width;
    y = rand() % Height;
    //b. specify a drawing color: red
    glColor3f(1, 0, 0);

    //c. specify to draw a point
    glBegin(GL_POINTS);
    glVertex2i (x,y);
    glEnd();

    //d. start drawing
    glFlush();
}
```

OpenGL

```
static void reshape(int w, int h)
{   //e. specify the window's coordinates
    glMatrixMode (GL_PROJECTION);
    glLoadIdentity ();
    glOrtho(0, Width, 0, Height, -1.0, 1.0);
}

int main(int argc, char **argv)
{   //f. initialize a drawing area
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE);
    glutInitWindowSize(Width, Height);
    glutCreateWindow("Example: random points");

    //g. specify event callback functions
    glutReshapeFunc(reshape);
    glutDisplayFunc(display);
    glutIdleFunc(display);
    glutMainLoop();
}
```

OpenGL

```
#include <GL/glut.h>
#include <GL/gl.h>

Void main(int argc, char** argv)
{
    int mode = GLUT_RGB|GLUT_SINGLE;
    glutInitDisplayMode(mode);
    glutInitWindowSize(500,500);
    glutCreateWindow("Example");
    init();
    glutDisplayFunc(display);
    glutReshapeFunc(resize);
    glutKeyboardFunc(key);
    glutMainLoop();
}
```

определя режима
на визуализиране

OpenGL

```
#include <GL/glut.h>
#include <GL/gl.h>

Void main(int argc, char** argv)
{
    int mode = GLUT_RGB|GLUT_SINGLE;
    glutInitDisplayMode(mode);
    glutInitWindowSize(500,500);
    glutCreateWindow("Example");
    init();
    glutDisplayFunc(display);
    glutReshapeFunc(resize);
    glutKeyboardFunc(key);
    glutMainLoop();
}
```


създава прозорец с име “Example” и размер 500x500

OpenGL

```
#include <GL/glut.h>
#include <GL/gl.h>

Void main(int argc, char** argv)
{
    int mode = GLUT_RGB|GLUT_SINGLE;
    glutInitDisplayMode(mode);
    glutInitWindowSize(500,500);
    glutCreateWindow("Example");
    init();
    glutDisplayFunc(display);
    glutReshapeFunc(resize);
    glutKeyboardFunc(key);
    glutMainLoop();
}
```

инициализиращ OpenGL код (опционален)

OpenGL

- Инициализация на параметри

например

```
void init(void)
{
   glClearColor(0,0,0,1);
   glEnable(GL_LIGHT0);
   glEnable(GL_LIGHTING);
   …
}
```

OpenGL

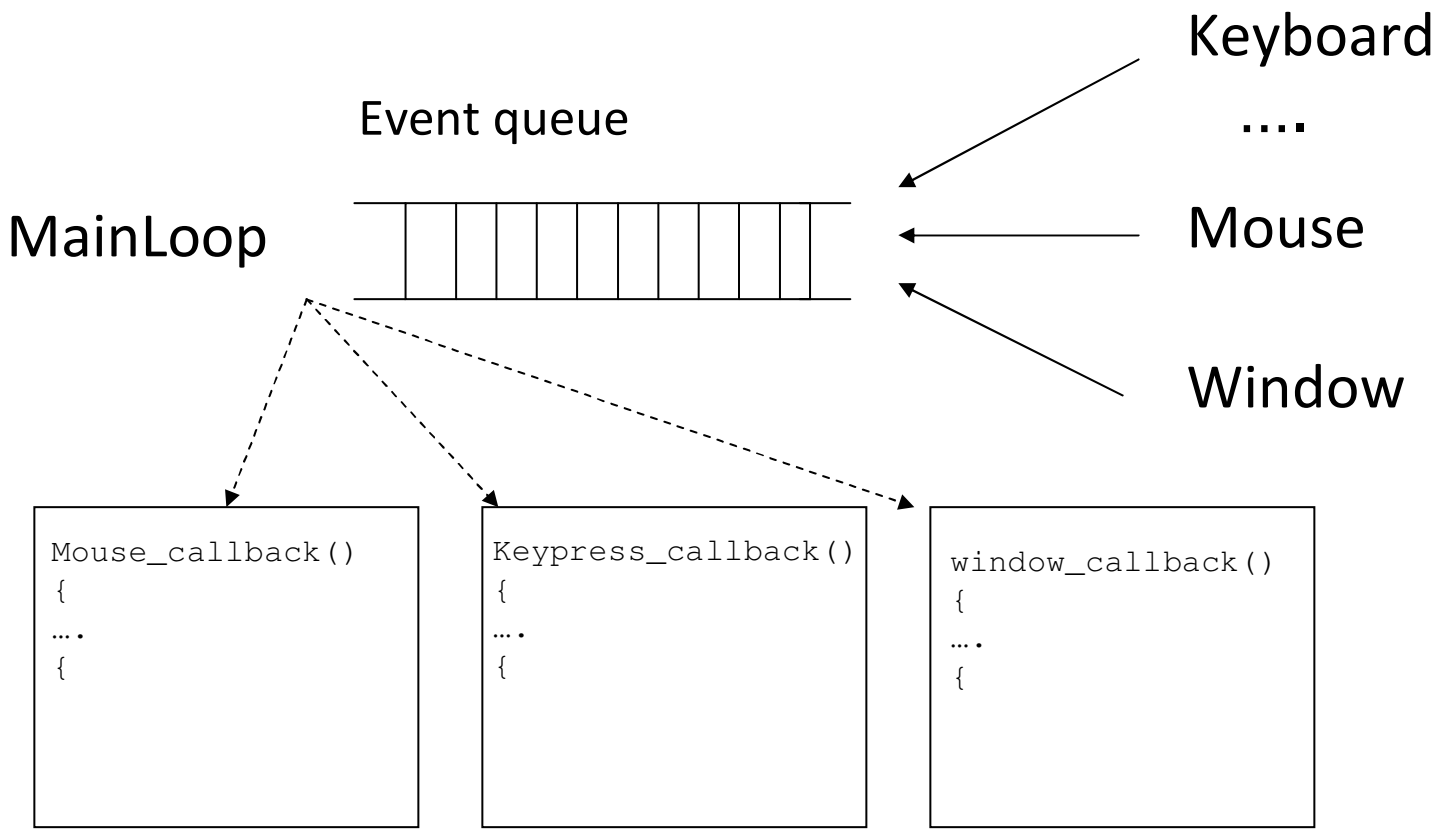
```
#include <GL/glut.h>
#include <GL/gl.h>

Void main(int argc, char** argv)
{
    int mode = GLUT_RGB|GLUT_SINGLE;
    glutInitDisplayMode(mode);
    glutInitWindowSize(500,500);
    glutCreateWindow("Example");
    init();
    glutDisplayFunc(display);
    glutReshapeFunc(resize);
    glutKeyboardFunc(key);
    glutMainLoop();
}
```

регистриране на callback функции

OpenGL

- Callback функции
  - обработват събития



OpenGL

- glutReshapeFunc(void (*func)(void))

```
void main(int argc, char** argv)
{
        …
    glutDisplayFunc(display);
    glutReshapeFunc(resize);
    glutKeyboardFunc(key);
    …
}
```

void resize () – потребителски дефинирана функция, извиква се при промяна на размера на прозореца

OpenGL

- glutKeyboardFunc()
  - регистрира функция, която се извиква при натискане на клавиш от клавиатурата
- glutMouseFunc()
  - регистрира функция, която се извиква при натискане на клавиш на мишката
- glutMotionFunc()
  - регистрира функция, която се извиква при натискане на клавиш на мишката и движение
- glutIdleFunc()
  - регистрира функция, която се извиква при липса на събитие

OpenGL

```
#include <GL/glut.h>
#include <GL/gl.h>

Void main(int argc, char** argv)
{
    int mode = GLUT_RGB|GLUT_SINGLE;
    glutInitDisplayMode(mode);
    glutInitWindowSize(500,500);
    glutCreateWindow("Example");
    init();
    glutDisplayFunc(display);
    glutReshapeFunc(resize);
    glutKeyboardFunc(key);
    glutMainLoop();
}
```

програмата влиза в безкраен цикъл в очакване на събитие

КРАЙ

Следваща тема:

Построяване на двумерен изглед