Компютърна графика

3D визуализация

доц. Милена Лазарова, кат. КС, ФКСУ

2 D Визуализиране

1) Изграждане на общата сцена чрез моделиращи трансформации над обектите
2) Задаване на прозорец и рамка за визуализиране
3) Преобразуване от световни координати на прозореца в екранни координати на рамката за визуализиране
4) Изрязване по прозореца
5) Растеризация на образите в координати на изходното устройство (екранни координати)

3D Визуализиране

1) Изграждане на общата сцена чрез моделиращи трансформации над обектите
   - модел на сцената в световни координати
2) ***Задаване на метод за проектиране и определяне на визуални обеми*** (viewing volumes)
   - визуалният обем определя какво ще се визуализира
   - viewport определя къде ще се визуализира
3) ***Нормализация и изрязване на обектите***
4) Преобразуване от световни координати на визуалния обем в екранни координати на рамката за визуализиране
5) ***Определяне на модел на осветеност и премахване на скрити обекти и повърхнини***
6) Растеризация на образите в координати на изходното устройство (екранни координати)

3D Визуализиране

- Изкуствена камера
  - програмен ***модел*** за специфициране на параметрите за 3D проектиране и визуализиране

- Параметри
  - позиция на камерата
  - ориентация
  - наблюдавана област: ъгъл, нормала, ...
    - field of view
  - дълбочина на наблюдавана област: близко и далечно разстояние
  - фокусно разстояние
    - “замъгляване”
  - наклон на камерата
    - ако “равнината на филма” не е нормално разположена на посоката на наблюдение се получава наклонена проекция
  - перспективна или паралелна проекция
    - разстояние на камерата до обектите: близко или безкрайно

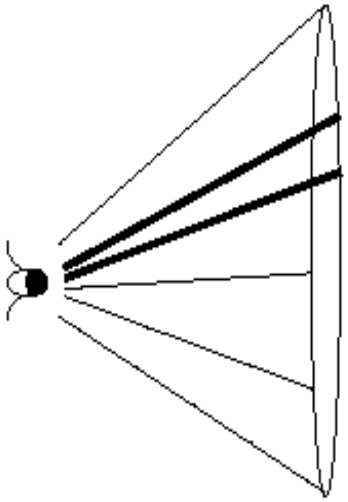
Визуални обеми

- Визуалният обем съдържа всичко, което може да се види от позицията на наблюдение или посоката на наблюдение
  - какво “вижда” камерата

- ***Коничен*** визуален обем
  - апроксимира наблюдаваната с човешко око област
  - сложна математика за изрязване на обектите спрямо повърхността на конус
- ***Правоъгълен*** визуален обем
  - апроксимация на конус
    - пресечена пирамида (frustum)
  - работи добре с правоъгълна рамка за визуализиране
  - линейни уравнения за лесно изчисляване на изрязването на обектите спрямо страните на визуалния обем

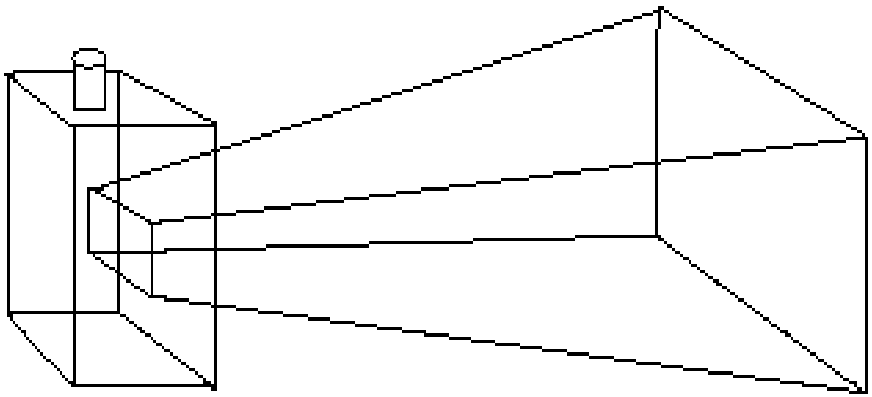
Визуални обеми

око



коничен перспективен визуален обем

камера на наблюдение



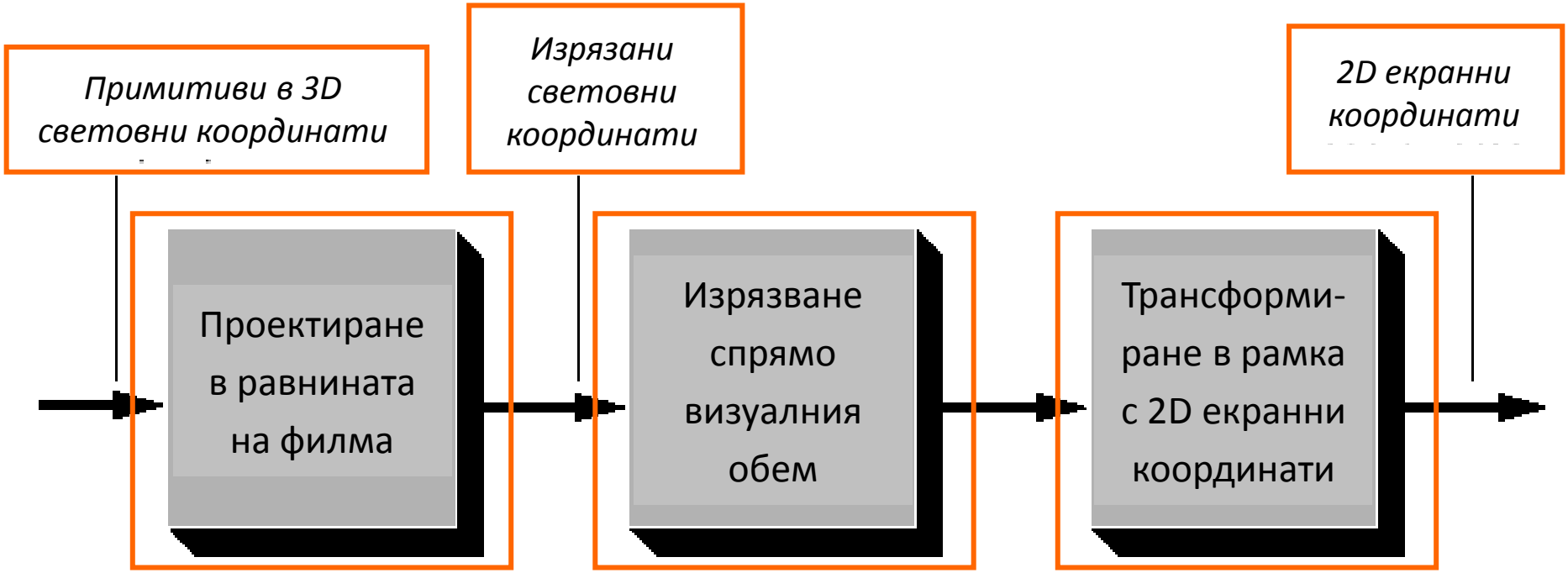
визуален обем с frustum апроксимация

3D Визуализиране

- Рамка за визуализиране (Viewport)
  - правоъгълна област от екрана, в която сцената се визуализира
    - ***window*** е 2D изрязващ правоъгълник за 2D световни координати
    - ***viewport*** е регион с 2D целочислени екранни координати, в който се визуализира съдържанието на изрязания прозорец

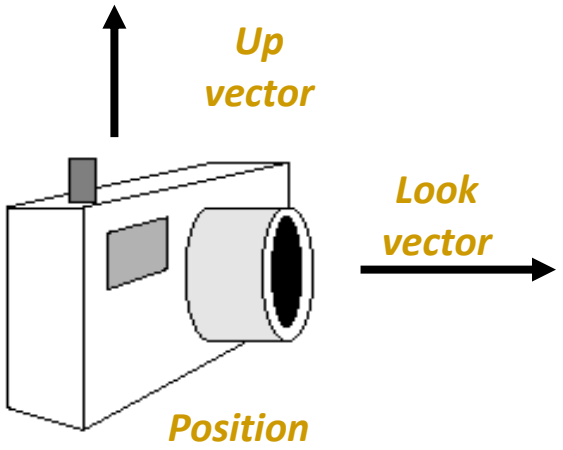
- Viewport и напречното 2D сечение на 3D визуалния обем могат да имат различни пропорции (aspect ratio)
  - изобразяването в рамката за визуализиране (viewport) преобразува координатите от прозореца в проекционната равнина (равнината на “филма” – film plane) в рамка за визуализиране с 2D екранни координати)
    - специфицира се как се преобразуват координатите при различни пропорции

3D Визуализиране



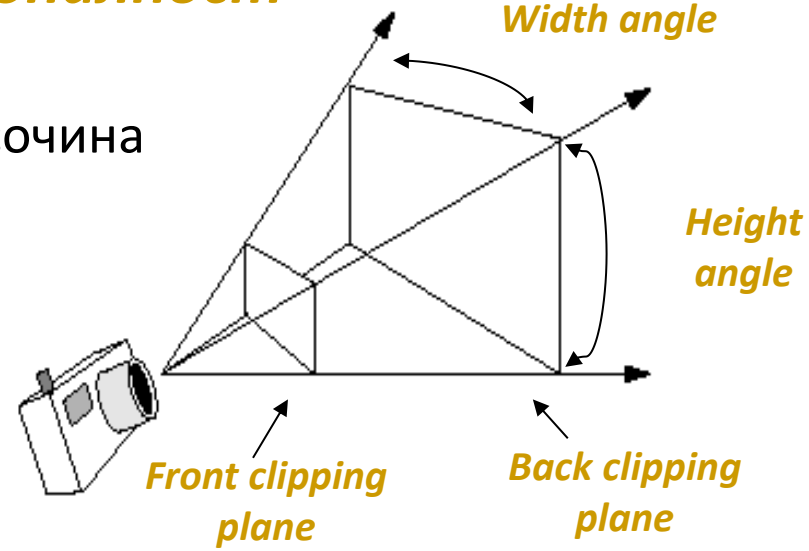
Визуален обем

- За да се определи какво се “вижда” с камера са нужни 6 параметъра
  (1) ***Позиция на камерата***
  (*position*)
  - от къде се наблюдава
  (2) ***Вектор на наблюдение***
  (*look vector*)
  - в каква посока е насочена камерата
  (3) ***Ориентация на камерата***
  (*up vector*)
  - определя се от ъгъла на завъртане на камерата спрямо вектора на наблюдение



Визуален обем

- За да се определи какво се “вижда” с камера са нужни 6 параметъра
  (4) ***Коефициент на пропорционалност*** (*aspect ratio*)
  - съотношение на ширина и височина на проекционната равнина
  (5) ***Ъгъл на височина*** (*height angle*)
  - каква част от сцената попада във визуалния обем
  - ***ъгъл на ширина*** (*width angle*)
    - определя се от ъгъл на височина и коефициент на пропорционалност
    - по-голям ъгъл – по-голямо перспективното изкривяване

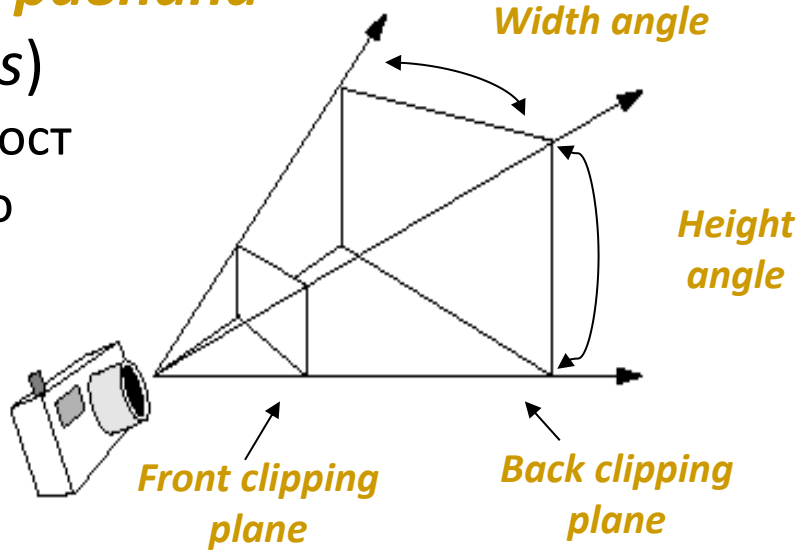


Визуален обем

- За да се определи какво се “вижда” с камера са нужни 6 параметъра

  (6) ***Предна и задна изрязващи равнини*** (*front and back clipping planes*)
  - ограничават обхвата на видимост на камерата до областта, която попада между тях

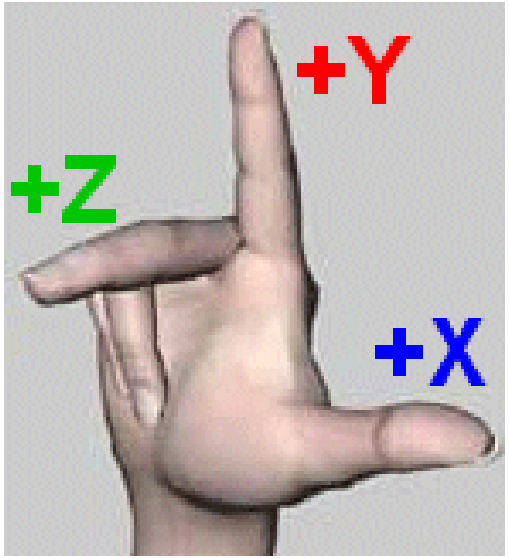
  Опционален параметър:
  - ***фокусно разстояние***
    - използва се за фотореалистично визуализиране
      - обектите на разстояние равно на фокусното разстояние се визуализират с ясен фокус
      - обектите на по-голямо или по-малко разстояние се “замъгляват” (размиват)



Позиция

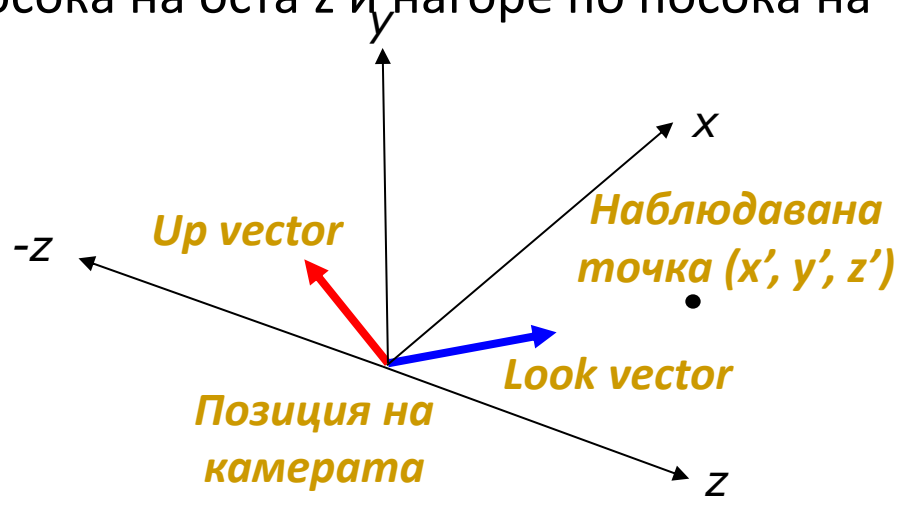
- Определянето на позицията на камерата е аналогично на определяне на подходяща позиция и гледна точка за заснемане на фотография

- Три степени на свобода
  - x, y и z координати в тримерно пространство
  - дясно ориентирана координатна система



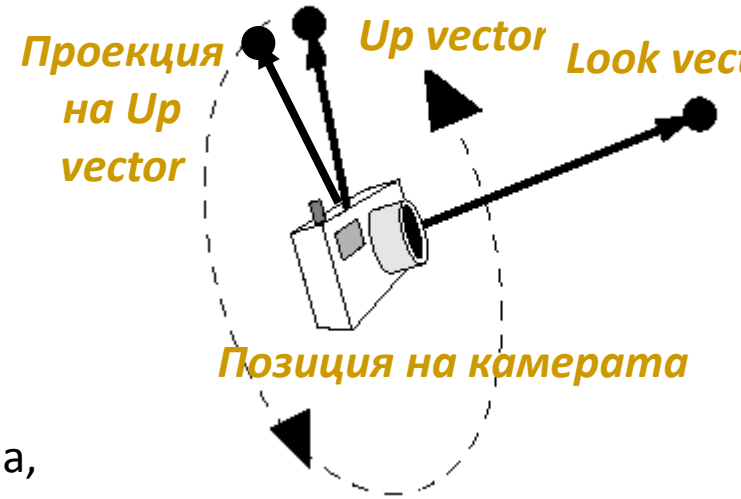
Ориентация

- Определя се с точка в 3D пространството, която се наблюдава (или посока на наблюдение) и ъгъл на ротация спрямо тази посока
- ***Подразбираща се (канонична) ориентация***
  - надолу към отрицателната посока на оста z и нагоре по посока на оста y
- В общия случай
  - камерата е разположена в началото на координатната система и е насочена към произволна точка в произволна посока



Ориентация

- ***Look vector***
  - определя посоката, в която е насочена камерата
  - три степени на свобода
    - може да бъде всеки вектор в тримерното пространство
- ***Up vector***
  - определя ориентацията на камерата
  - определя как е ротирана камерата спрямо Look vector
    - например дали камерата е вертикална, хоризонтална или с друго положение
- двата вектора Up vector и Look vector не трябва да са паралелни
  - векторът на ориентация Up vector може да се зададе под произволен ъгъл спрямо вектора на наблюдение Look vector



Не е задължително двата вектора Up vector и Look vector да са ортогонални, задължително е да не са успоредни

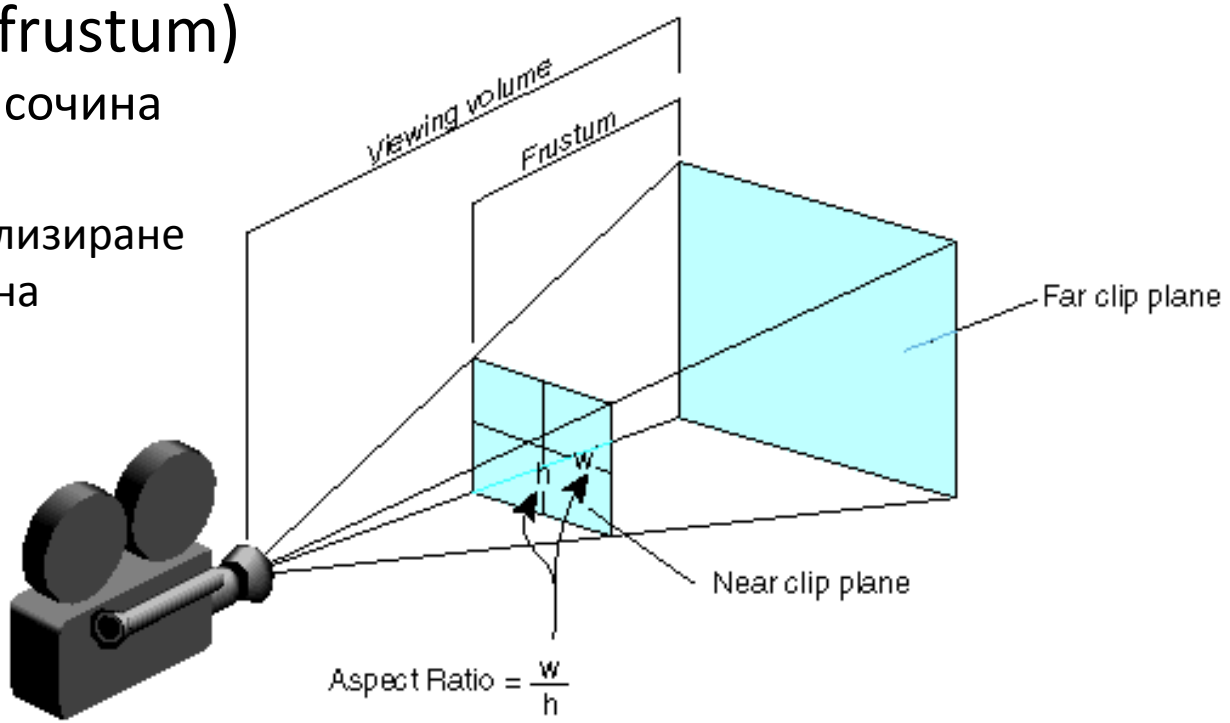
Коефициент на пропорционалност

- Аналогичен на пропорциите на размера на филм в камера
  - определя отношението на ширината и височината на изображение, визуализирано на екрана

- Квадратен прозорец на визуализиране
  - аспект 1:1
- Формат на филми в кино
  - 2:1
- NTSC телевизия
  - 4:3
- HDTV
  - 16:9 или 16:10

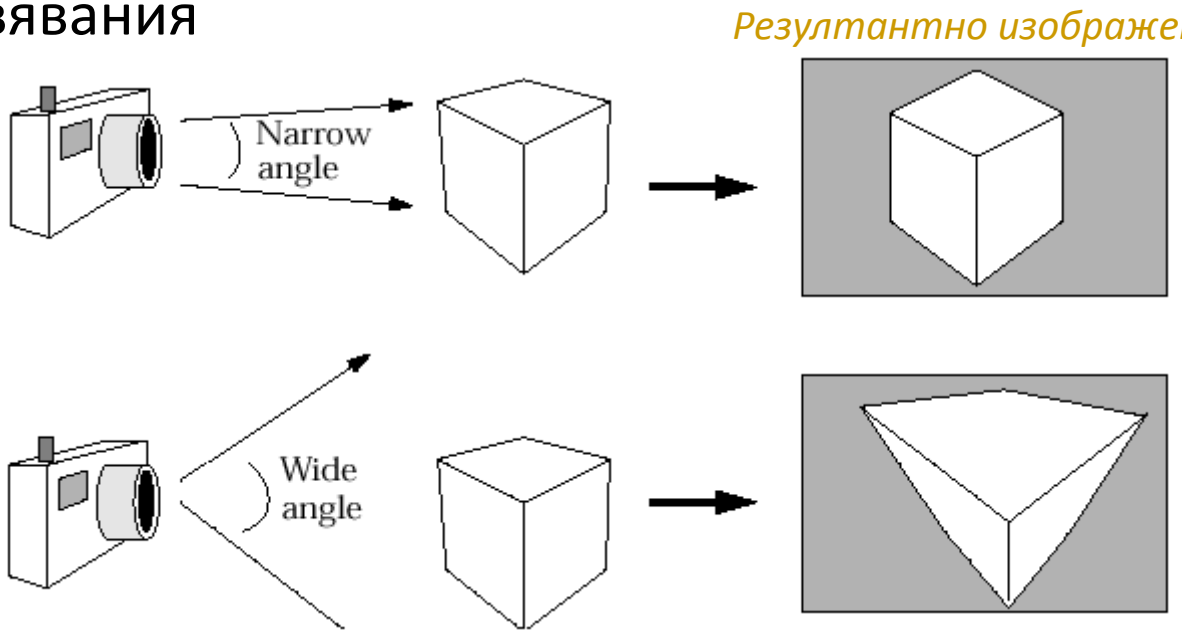
Ъгъл на визуализиране

- Определя степента на перспективно изкривяване в изображението
  - от никакво при паралелна проекция
  - до голямо при широкоъгълни лещи
- Два ъгъла на визуализиране в апроксимирания визуален обем (frustum)
  - на ширина и на височина
- Избор на ъгъл на визуализиране е аналогичен на избор на специални лещи във фотографията (напр. широкоъгълни)



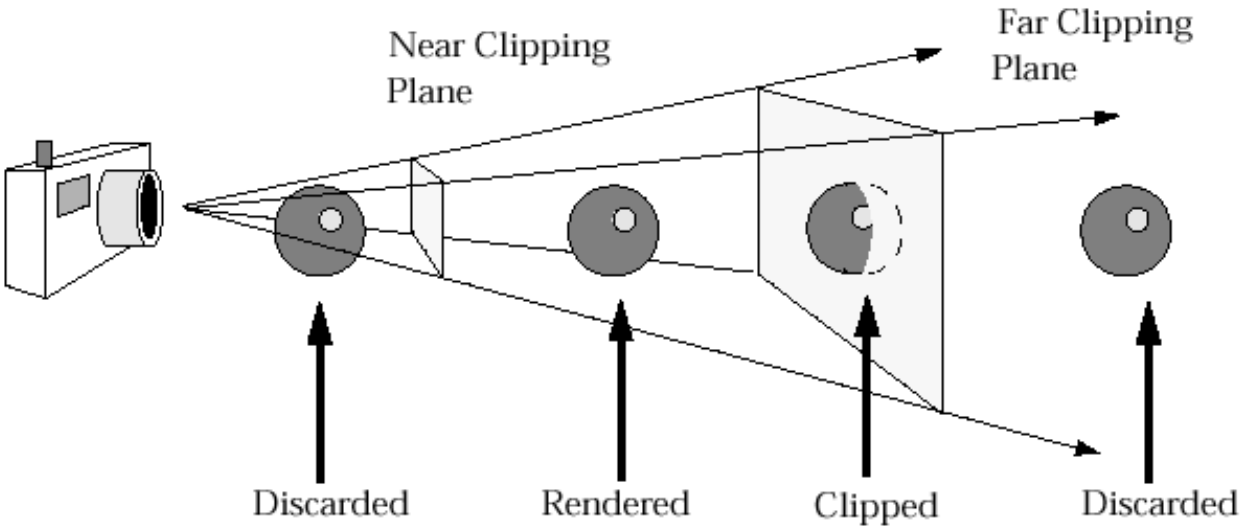
Ъгъл на визуализиране

- Лещите за снимки на отдалечени обекти обикновено имат почти паралелен ъгъл на визуализиране и внасят минимално перспективно изкривяване
  - за сметка на това скъсяват “дълбочината”
- Широкоъгълните лещи водят до големи перспективни изкривявания



Изрязващи равнини

- Обемът от пространството между предната и задната изрязващи равнини определя какво се вижда от камерата
- Позицията на равнините се дефинира чрез разстояние по посока на вектора на наблюдение Look vector
  - Обектите, разположени извън визуалния обем не се визуализират
  - Обектите, пресичащи визуалния обем се изрязват



Изрязващи равнини

- Предна (близка) изрязваща равнина
  - Не искаме да визуализираме обекти, които са разположени твърде близо до камерата
    - ще скриват и пречат на видимостта на останалата част от сцената
    - ще са твърде изкривени
  - Не искаме да визуализираме обекти, които са разположени зад камерата
    - не се очаква да се виждат подобни обекти
    - в случай на перспективна проекция, дори и да се визуализират такива обекти, ще бъдат обърнати отдолу нагоре и отвътре навън

Изрязващи равнини

- Задна (далечна) изрязваща равнина
  - Не искаме да визуализираме обекти, които са разположени твърде далече от камерата
    - отдалечените обекти може да се визуализират твърде малки за да са визуално значими, а същевременно визуализирането им изисква изчислителни ресурси и отнема време
      - чрез изрязването им се губят малко детайли, но се печели значително време за визуализиране на сцената
    - в друг случай, ако сцената съдържа много важни обекти, за да е по-ясно визуализирането им е по-удачно да се визуализират само тези, които са близо до камерата и да се изрежат останалите

Изрязващи равнини

- Внезапно появяващи се на заден план обекти
  - това са обекти, преминали пред задната изрязваща стена
    - например дърво в състезание

- Трик за предотвратяване на подобни внезапно изскачащи обекти
  - добавяне на мъгла на заден план
    - пример: Turok: Dinosaur Hunter
  - особено удачно за сцени на открито

- Благодарение на бързия хардуер и нивото на детайлност на алгоритмите, разстоянието до задната изрязваща равнина се увеличава, а мъглата се използва рядко

Изрязващи равнини

- Позициониране на близката изрязваща равнина колкото е възможно по-далече
  - удачно от гл.т. на Z точността
- Възможно е камерата да заеме позиция, такава че предната част на обект да се изреже и да се “види” вътрешността му

решение

- премахване на обектите в близост до предната изрязваща равнина преди да бъдат разрязани
  - пример: Okami

Фокусно разстояние

- Някои модели на изкуствени камери използват фокусно разстояние
  - фокусното разстояние е мярка за идеалния фокусен обхват
  - апроксимира поведението на реални лещи на камера
    - Обектите на разстояние равно на фокусното се визуализират с ясен фокус
    - Обектите на по-малко или по-голямо от фокусното разстояние се “размиват”

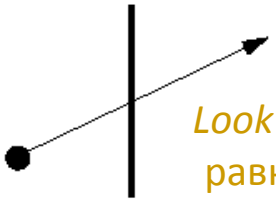
- Фокусното разстояние се използва заедно с изрязващите равнини
  - само обектите във визуалния обем се визуализират независимо дали добре фокусирани или размити

Визуални обеми

- С модел на изкуствена камера могат да се създадат следните визуални обеми
  - перспективен
    - положителен ъгъл на визуализиране
  - паралелен
    - нулев ъгъл на визуализиране
- С модел на изкуствена камера ***не*** могат да се създадат наклонени визуални обеми

Не-наклонен визуален обем

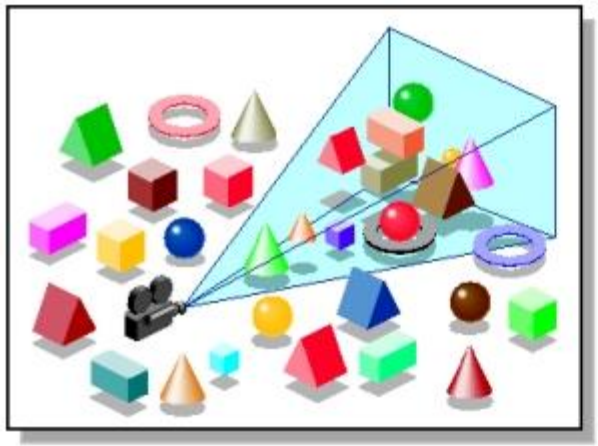
Наклонен визуален обем

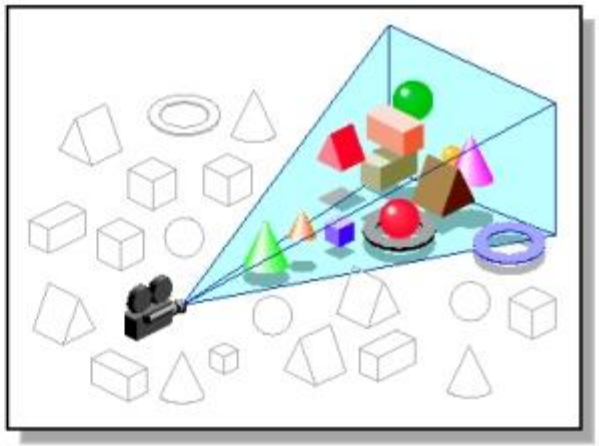


- Например за заснемане на високи сгради с използват камери с “мях”
  - равнината на филма е паралелна на фасадата, а камерата е насочена нагоре
  - така се постига наклонен визуален обем без изкривявания на фасадата

Визуални обеми

- С параметрите позиция, векторите look и up, коефициент на пропорционалност, ъгъл на височина, изрязващи равнини и (опционално) фокусно разстояние се специфицира пресечен визуален обем
  - truncated view volume
  - спецификация на ограниченото пространство, което камерата може да “види”



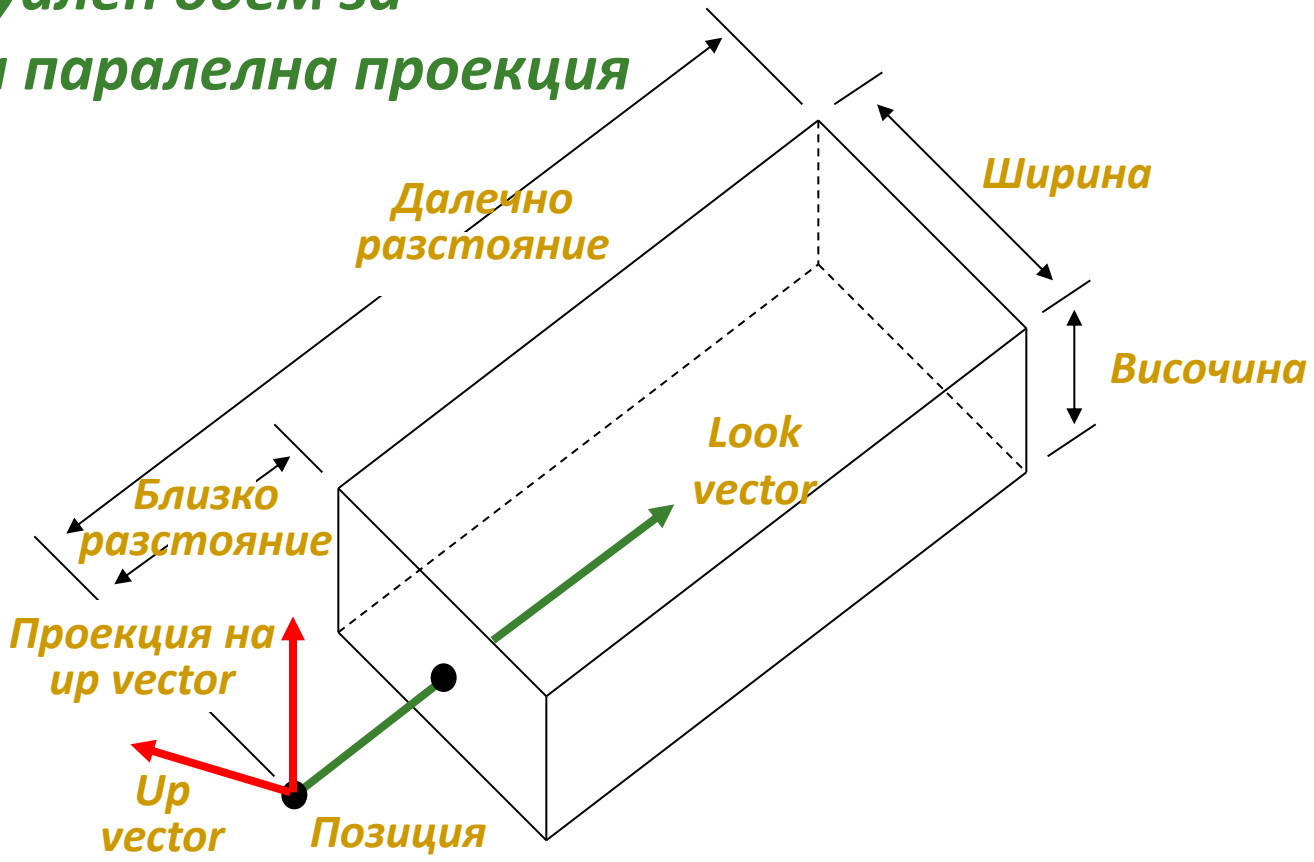


- 2D изглед на 3D сцена се изчислява чрез проектиране на пресечения визуален обем в равнината на “филма”
  - ***паралелен*** визуален обем
  - ***перспективен*** визуален обем

Паралелен визуален обем

- ***Пресечен визуален обем за ортографска паралелна проекция***

- Ограничаването на визуалния обем елиминира странични обекти
- Ортографската паралелна проекция има нулеви ъгли на височина и ширина

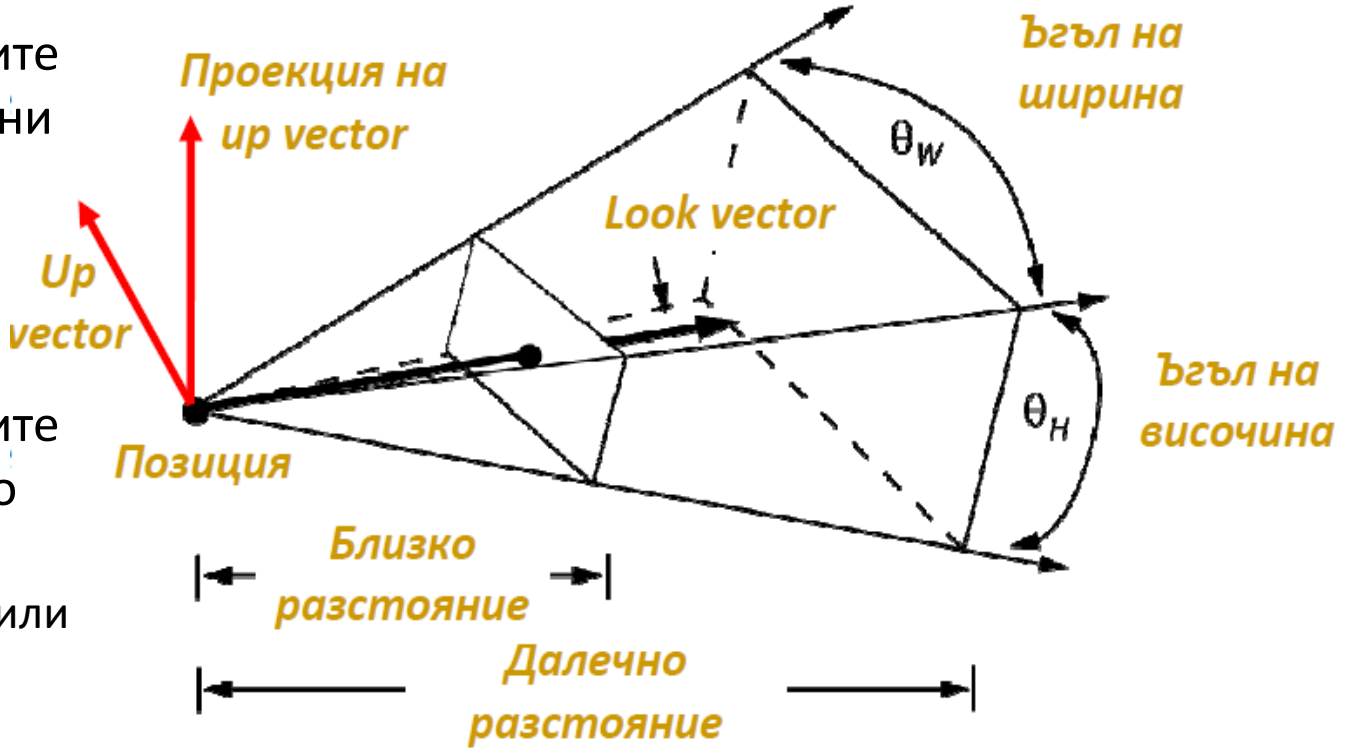


Паралелен визуален обем

- ***Пресечен визуален обем за перспективна проекция (Frustum)***

- Премахва обектите твърде отдалечени от позицията
  - биха се слели в петна

- Премахва обектите твърде близки до позицията
  - биха се изкривили значително



Равнина на филма

- Реалните камери имат филм за съхраняване на изображенията
- “Филмът” на изкуствената камера е правоъгълник в безкрайната равнина на филма, който съдържа изображението на сцената

- Защо параметрите на изкуствената камера не се отнасят до филма (освен аспекта)?
  - Как е разположена равнината на филма спрямо другите части на камерата?
  - Къде е разположена спрямо изрязващите равнини?

Равнина на филма

- Защото позицията на равнината на филма (равнината на проекция) не оказва влияние на формирания изглед
  - при паралелния визуален обем
    - паралелната проекция в равнината на филма ще бъде една и съща без значение колко далече се намира равнината на филма от сцената
  - при перспективния визуален обем
    - последната стъпка от изчисляване на перспективна проекция е трансформация, която разтяга перспективния обем в паралелен обем

- Обикновено се приема, че равнината на филма лежи в ***задната изрязваща равнина***

3D Визуализиране

- Генерирането на 2D изображение от параметри на 3D изглед в общия случай е трудна задача

- Използват се
  - ***каноничен визуален обем***
    - паралелепипед на 3D паралелна проекция
  - ***канонична позиция на наблюдение***
    - камера с позиция в началото на координатната система, насочена по отрицателната посока на оста z

3D Визуализиране

- ***Етапи на изграждане на 3D изглед***

(1) определяне на параметри, специфициращи визуалния обем

(2) трансформация от специфицирания визуален обем в каноничен визуален обем

(3) използване на канонична позиция на наблюдение за изрязване, проектиране и растеризиране на сцената за създаване на 2D изображение

Етап 1

- ***Специфициране на визуалния обем***
  - Редуциране на степените на свобода
  - 4 стъпки – задават се основните параметри

    (a) ***позиция на камерата***
    - съответно равнина на визуализиране/равнина на филма

    (b) ***ориентиране на камерата***
    - така че да бъде насочена в желаната посока

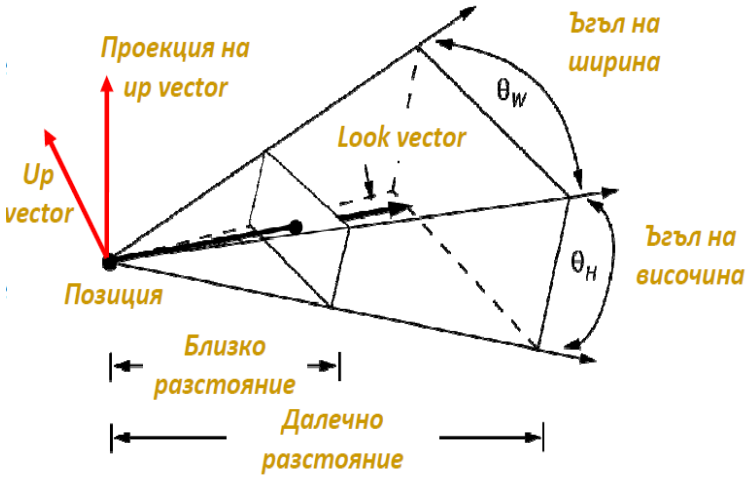
    (c) ***област на наблюдение***
    - перспективна проекция
      - коефициент на пропорционалност на филма и ъгъл на визуализиране
    - паралелна проекция
      - ширина и височина

    (d) избор на перспективна или паралелна ***проекция***

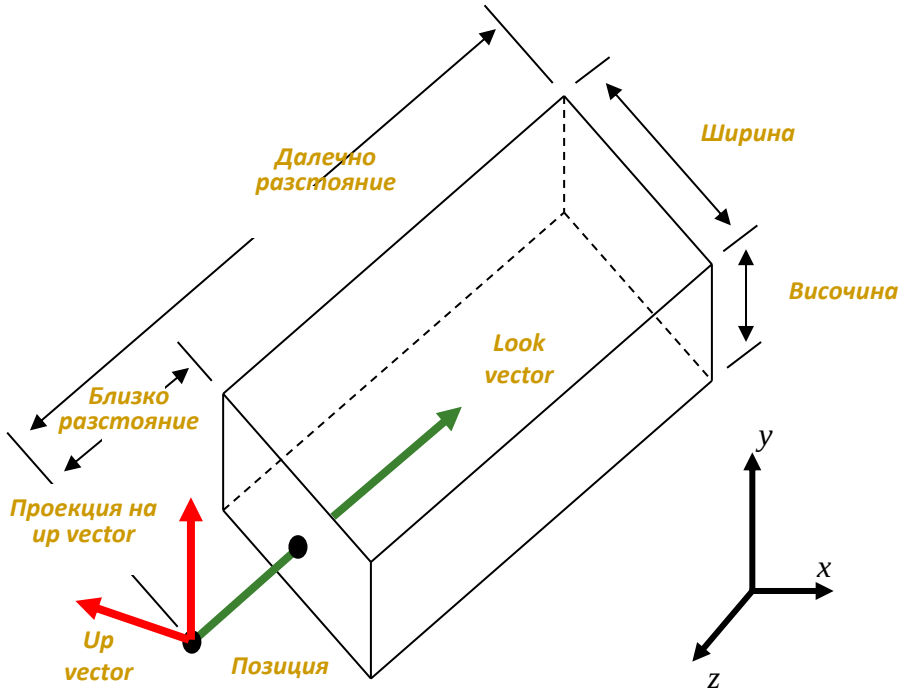
- Опционално се специфицират ***фокусно разстояние*** и ***време на експозиция***

Етап 1

- ***Перспективна проекция***
  - ***пресечена пирамида (Frustum)***
    - Look vector е насочен по височината на пирамидата



- ***Ортографска паралелна проекция***
  - ***паралелепипед***
    - няма параметър ъгъл на визуализиране

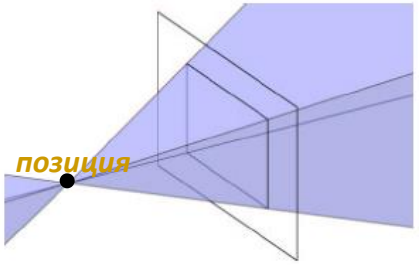


Етап 1

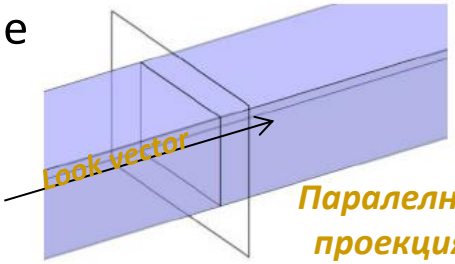
- ***Специфициране на произволен визуален обем***
  - ***Разположение*** на визуалния обем (видимата част от света)

    специфицира се позиция и ориентация на камерата
    - позиция (точка)
    - Look и Up вектори
  - ***Форма*** на визуалния обем

    специфицира се със задаване на
    - хоризонтален и вертикален ъгъл на визуализиране
    - предна и задна изрязваща равнина



Перспективна проекция



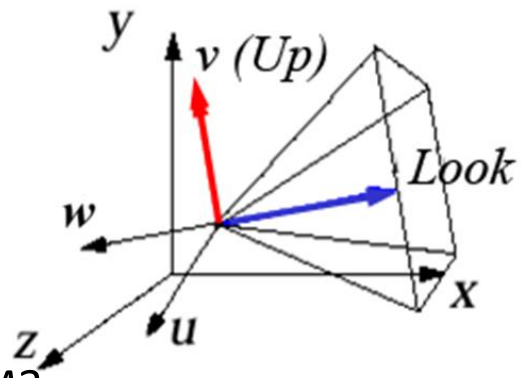
Паралелна проекция

- ***Перспективна проекция***
  - проекционните лъчи се пресичат в позицията на камерата
- ***Паралелна проекция***
  - проекционните лъчи са паралелни на Look vector и не се пресичат

Етап 1

- ***Специфициране на произволен визуален обем***
  - Координатни системи
    - ***световни координати***
      - стандартна дясно ориентирана 3D КС с оси x, y, z
    - ***координати на камерата***
      - дясно ориентирана координатна система в пространството на камерата
        - начало на КС в позицията на камерата
        - координатни оси u, v, w ротирани спрямо ориентацията на камерата

          (векторът v е проекцията на вектор Up)
      - използва се за трансформиране на произволен визуален обем в каноничен



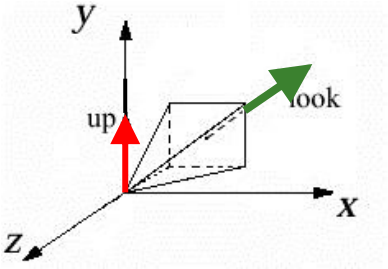
Произволен перспективен визуален обем

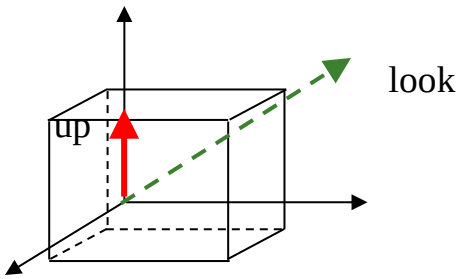
- координатната система на камерата е дефинирана в координатната система на сцената

Етап 1

- ***Специфициране на произволен визуален обем***
  - задачата за изобразяване на произволен визуален обем в 2D изображение на сцената е сложна по отношение на изрязване и проектиране
  - ***решение***: редуцира се до по-прост проблем, който може лесно да бъде решен
    - използва се канонична позиция на наблюдение,

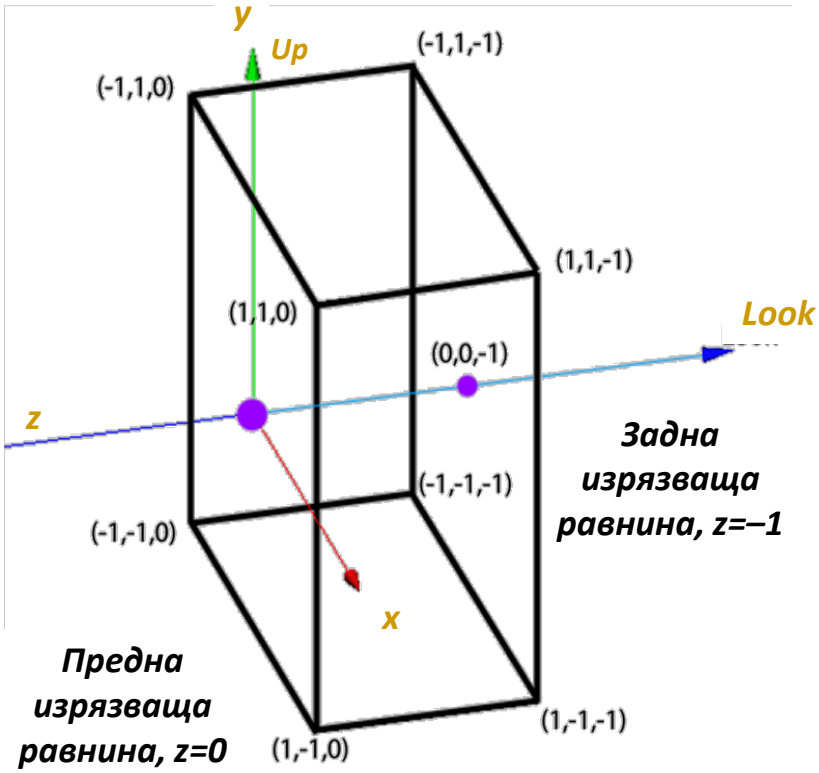
      за която лесно могат да се извършат изчисленията
      - позиция на камерата в началото на КС
      - насочена по отрицателната посока на оста z





Етап 1

- ***Специфициране на произволен визуален обем***



за паралелна проекция

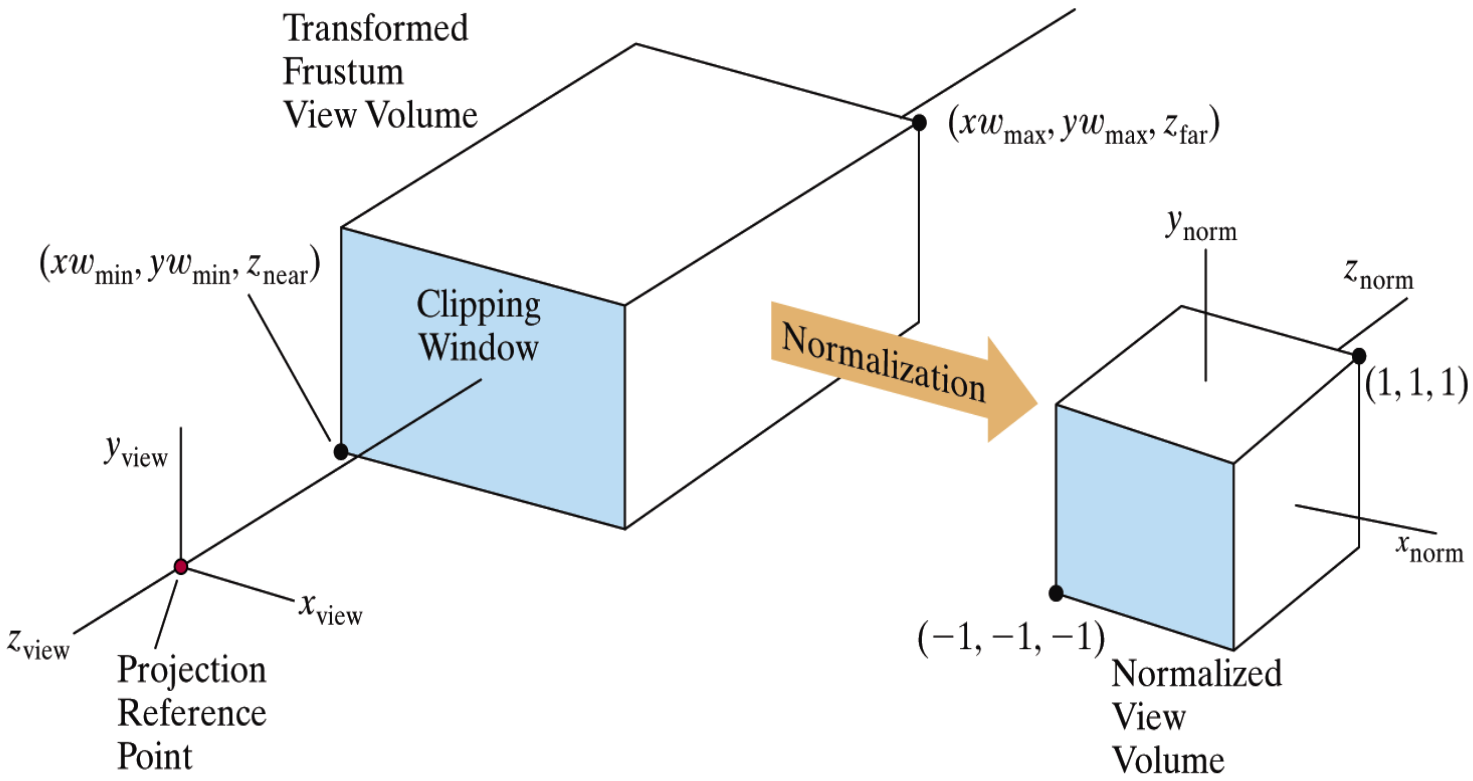
- позиция в началото на КС
  - позиция = (0, 0, 0)
- насочена по отрицателната посока на оста z
  - Look vector = (0, 0, –1)
- ориентирана вертикално
  - Up vector = (0, 1, 0)
- равнината на филма е разположена между –1 и 1 по осите x и y
  - изборът на Look vector по отрицателната, а не по положителната посока на оста z прави изчисленията по-прости

Етап 2

- ***Нормализиране до каноничен визуален обем***
  - цел
    - да се трансформира произволния визуален обем до каноничен обем
  - ***нормализираща трансформация***
    - ***афинна трансформация за паралелен визуален обем***
      - съставена от линейни трансформации (ротация и мащабиране) и транслация
    - ***трансфотрмацията не е афинна за перспективен визуален обем***
      - включва не-афинни перспективни трансформации, с които пресечената пирамида се преобразува в куб
  - афинните трансформация запазват успоредността, но не и дължината и ъглите
  - перспективната трансформация е проекционна, не-афинна трансформация, която не запазва успоредността

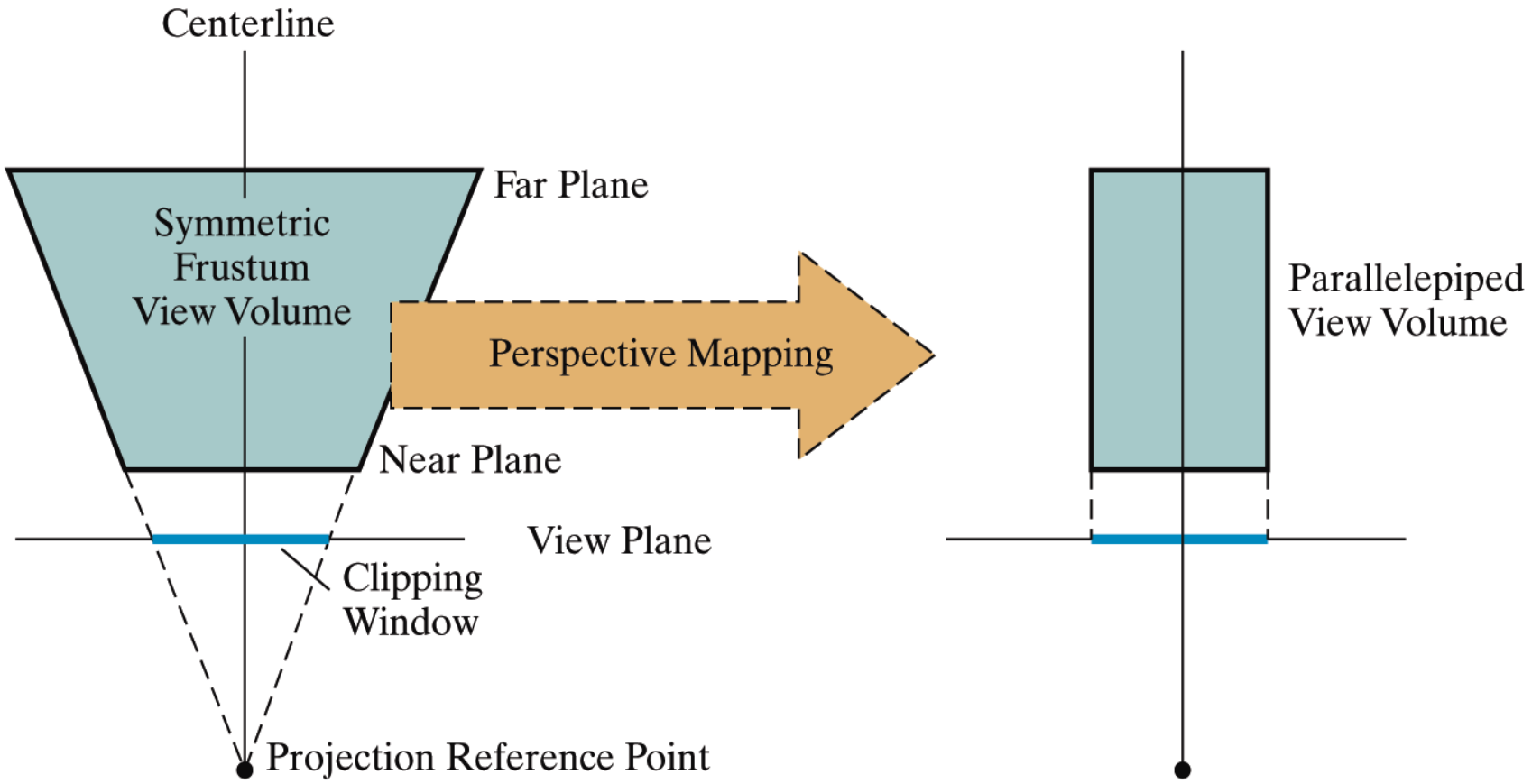
Етап 2

- ***Нормализиране до каноничен визуален обем***



Етап 2

- ***Нормализиране до каноничен визуален обем***



Етап 2

- ***Нормализиране до каноничен визуален обем***
  - ***нормализираща трансформация***
    - композитна трансформация
      - преобразува произволен визуален обем в каноничен
    - използва се хомогенна матрица 4x4
    - матрицата на трансформацията има и обратна матрица
  - изрязването е лесно при каноничен визуален обем
    - изрязващите равнини са успоредни на координатните оси
  - проектирането е лесно при каноничен визуален обем
    - просто се пропуска z координатата
    - за наклонена паралелна проекция част от композитната трансформация е еластична деформация, която да “възстанови” и изправи наклонения визуален обем

Етап 2

- ***Визуализираща трансформация ↔ Нормализираща трансформация***
  - Проблемът за генериране на 2D изглед от 3D модел се редуцира до проблем за определяне на коректна нормализираща трансформация
- Определянето на ротиращата компонента на нормализиращата трансформация е трудно
  - по-лесно е да се определи инверсната ротационна операция
- Задачата е да се определи обратната трансформация на нормализиращата трансформация
  - нарича се визуализираща трансформация
  - преобразува каноничния визуален обем в произволен визуален изглед
  - трансформация от (x, y, z) в (u, v, w)

Етап 2

- ***Определяне на визуализираща трансформация по специфициран визуален обем***
  - известни са **позицията** и двата вектора (**Look vector**, **Up vector**)
  - трябва да се определи афинна трансформация от тези параметри за ***транслиране*** и ***ротиране*** на каноничния изглед в произволен изглед
  - на следващ етап се извършват
    - ***мащабиране*** на “филма” (т.е. напречното сечение на визуалния обем) за да се формира квадратно сечение
    - ***изрязване*** на обектите извън визуалния обем

Етап 2

- ***Определяне на визуализираща трансформация по специфициран визуален обем***
  - ***Транслация***
    - лесно се определя транслационната матрица в хомогенни координати
    - началото на координатна система се транслира в точката, зададена като позиция на изгледа *Position*(Pos_x, Pos_y, Pos_z)

$$T(Position) = \begin{bmatrix} 1 & 0 & 0 & Pos_x \\ 0 & 1 & 0 & Pos_y \\ 0 & 0 & 1 & Pos_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

  - ***Ротация***
    - определянето на ротационна матрица за преобразуване на x, y, z в u, v, w с използване на параметрите, специфициращи визуалния обем е по-трудна задача

Етап 2

- ***Ротация***
  - Евклидова трансформация (“rigid-body”)
    - трите единични вектора, ориентирани по координатните оси x, y и z се ротират в нова ориентация
    - при това, тъй като трансформацията е Евклидова

      то резултантните вектори след ротацията са
      - *отново с единична дължина*
      - *отново перпендикулярни един на друг*
      - *удовлетворяват правилото на дясната ръка*

  - ***всяка трансформационна матрица, която има тези три свойства води до ротация около някаква ос на някакъв ъгъл***

Етап 2

- Означаваме трите единични вектора, ориентирани по координатните оси x, y и z с e_1, e_2, e_3

$$e_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \qquad e_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \qquad e_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

- Задачата е:
  - да се определи ротационна матрица, такава че

    $R_{rot}\, u = e_1$

    $R_{rot}\, v = e_2$

    $R_{rot}\, w = e_3$

Етап 2

- Означаваме ротационната матрица M, а колоните ѝ v_1, v_2, v_3

$$M = \begin{bmatrix} v_1 & v_2 & v_3 \end{bmatrix}$$

- Произведението на M с e_1 е първата колона в матрицата: v_1

$$Me_1 = \begin{bmatrix} v_1 & v_2 & v_3 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = v_1 \longrightarrow$$

Me_1 е първата колона в матрицата M

Етап 2

- Аналогично произведението на M с e_2 и e_3 е v_2, v_3

$$Me_1 = \begin{bmatrix} v_1 & v_2 & v_3 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = v_1 \longrightarrow$$ Me_1 е първата колона в матрицата M

$$Me_2 = \begin{bmatrix} v_1 & v_2 & v_3 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = v_2 \longrightarrow$$ Me_2 е втората колона в матрицата M

$$Me_3 = \begin{bmatrix} v_1 & v_2 & v_3 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = v_3 \longrightarrow$$ Me_3 е третата колона в матрицата M

- Следователно M = [u v w] ще ротира осите x, y, z в осите u, v, w
  - u, v и w са единични вектор-колони

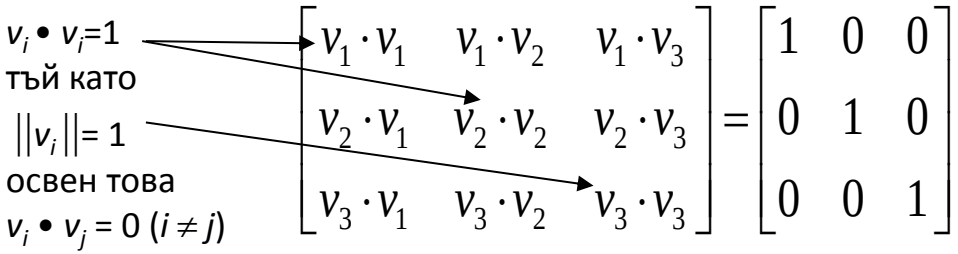
Етап 2

- Ако матрицата M = [u v w] ротира осите x, y, z в осите u, v, w
  - u, v и w са единични вектор-колони
- то матрицата M^{-1} ще ротира осите u, v, w в осите x, y, z
  - което е решение на поставената задача

- Следователно
  - най-напред по параметрите на специфицирания изглед се изчисляват u, v и w и се определя ротационната матрица M
  - след това се определя обратната матрица на M

Етап 2

- За ротационна матрица с колони v_i
  - колоните са единични вектори: $||v_i|| = 1$
  - колоните са перпендикулярни: $v_i \bullet v_j = 0 \quad (i \neq j)$



  - това скаларно произведение може да се представи като $M^T M = I$

    където M^T е матрица с редове v_1, v_2, v_3
  - за всяка неособена (обратима) матрица: $M^{-1} M = I$

- Следователно за ротационната трансформираща матрица M^{-1} е M^T
  - M^T се определя тривиално, докато определянето на M^{-1} изисква сложни изчисления

Етап 2

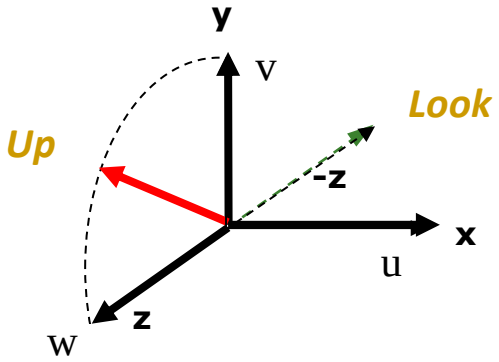
- ***Визуализираща трансформация: ротация***
  - ако M е ротационна матрица, то колоните и са две по две перпендикулярни и имат единична дължина
  - и обратното, ако колоните на матрица са две по две перпендикулярни и имат единична дължина, то матрицата е ротационна
- За такава матрица $$M^T M = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$
  - следователно $$M^T = M^{-1}$$
- Това позволява определянето на нормализиращата трансформация на базата на по-лесната за определяне визуализираща трансформация

Етап 2

- Ротационната матрица нормализира единичен вектор с начало в началото на КС в пространството на камерата с оси (u, v, w) в световни координати с оси (x, y, z)
  - трябва ротационната матрица да се изчисли по параметрите на изгледа
    - ***матрицата на визуализиращата трансформация***
      - ротационната матрица M преобразува (x, y, z) в (u, v, w)
      - M има колони (u, v, w)
    - ***матрицата на нормализиращата трансформация***
      - ротационната матрица $M^{-1} = M^T$ преобразува (u, v, w) в (x, y, z)
      - M^T има редове (u, v, w)
- Задачата за определяне на коректната ротационна матрица се свежда до задача за определяне на правилните перпендикулярни единични вектори u, v и w

Етап 2

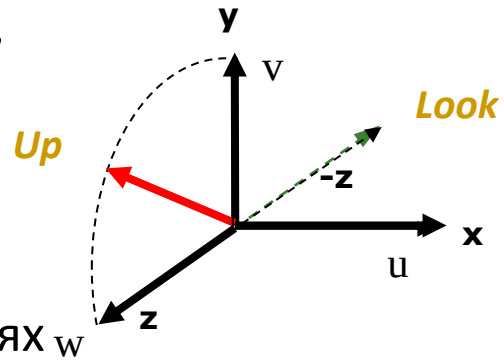
- ***Визуализираща трансформация: ротация***



  - на базата на параметрите на изгледа Position, Look vector, Up vector да се изчисли визуализираща ротационна матрица M с колони u, v, w
  - по обратната матрица M^{-1}, която е транспонираната матрица M^T, с редове векторите u, v, w да се определи нормализираща ротационна матрица

Етап 2

- Изчисляване на u, v, w по Position, Look vector, Up vector
  - изисквания за координатните оси (u, v, w)
    - произволния вектор Look да лежи върху отрицателната ос w
    - проекцията на вектора Up в равнината,
      за която оста w е нормала,
      да лежи върху оста v
    - оста u да е взаимно перпендикулярна
      на осите v и w и да образува заедно с тях
      дясно ориентирана КС



(1) определя се w от вектора Look

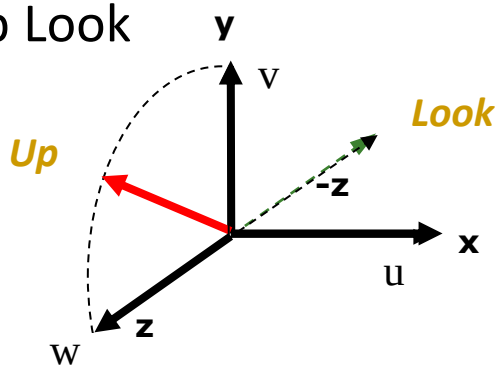
(2) определя се v от векторите Up и w

(3) определя се u като нормала на равнината определена от w и v

Етап 2

- ***За определяне на w***
  - в каноничния визуален обем вектор Look лежи върху оста –z
  - тъй като z се преобразува в w, то w е нормализиран вектор насочен в обратна посока на произволния вектор Look

$$w = \frac{-Look}{\|Look\|}$$



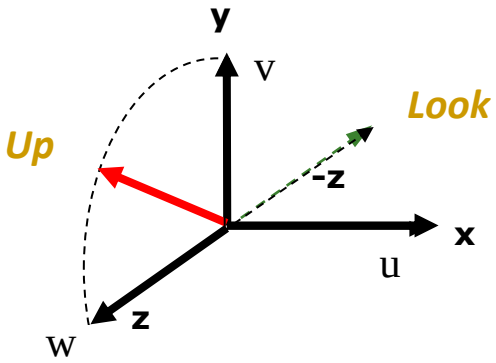
  - векторите Up и w определят равнина
  - u е нормала за тази равнина
  - v е нормала за равнината, определена от w и u

Етап 2

- ***За определяне на v***
  - *задача*: търси се вектор v, перпендикулярен на w
  - *решение*: проектира се вектор Up в w и се нормализира

$$v = Up - (w \bullet Up)w$$

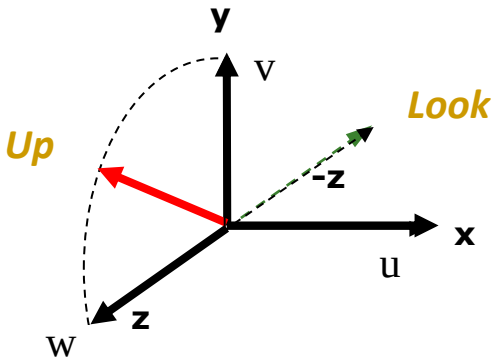
$$v = \frac{v}{\|v\|}$$



  - векторът w е с единична дължина, но векторът Up може да не е с единична дължина или перпендикулярен на w, така че се премахва компонента w и се нормализира
  - резултатът е вектор – компонент на Up в посока перпендикулярна на w

Етап 2

- ***За определяне на u***
  - използва се векторно произведение
    - $w \times v$ и $v \times w$ са перпендикулярни на равнината, но в различни посоки
  - тъй като резултатът трябва да е дясно ориентирана КС, то се използва $v \times w$



$$u = \frac{v \times w}{\|v \times w\|}$$

- Векторното произведение на два вектора е $a \times b = \begin{bmatrix} a_2 b_3 - a_3 b_2 \\ a_3 b_1 - a_1 b_3 \\ a_1 b_2 - a_2 b_1 \end{bmatrix}$

Етап 2

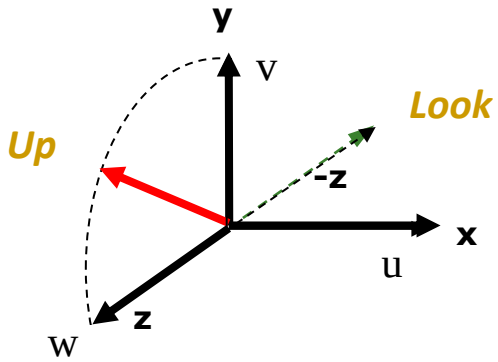
- ***Визуализираща трансформация: ротация***

$$w = \frac{-Look}{\|Look\|}$$

$$v = Up - (w \bullet Up)w$$

$$v = \frac{v}{\|v\|}$$

$$u = \frac{v \times w}{\|v \times w\|}$$

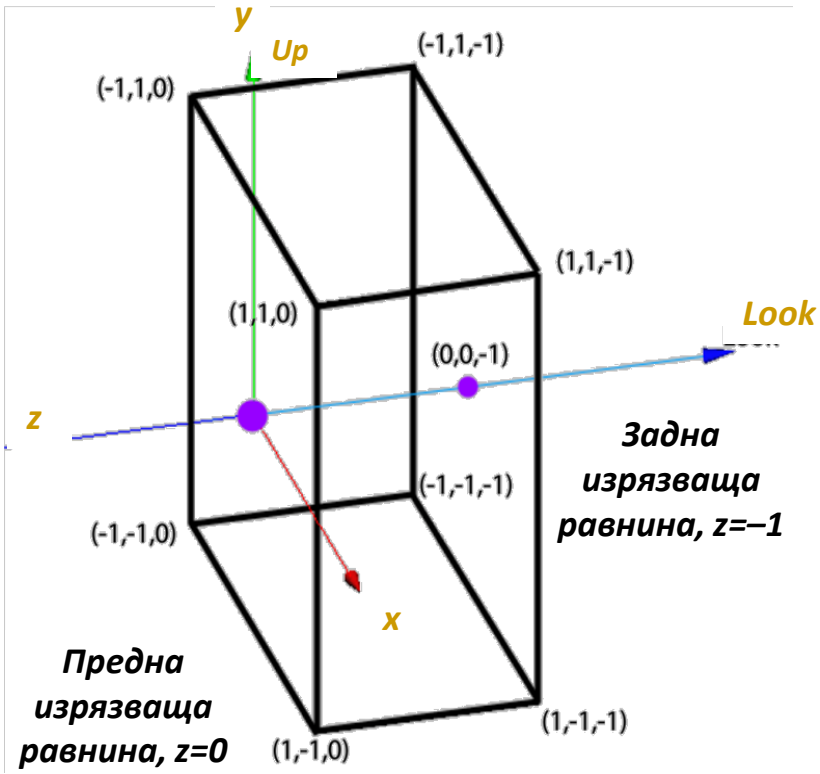


- изчислените параметри се използват за определяне на нормализиращата трансформация, която преобразува произволен вектор в каноничен изглед

Етап 2

- ***Нормализиране до каноничен визуален обем***
  - ***за паралелен визуален обем*** и множество възли на обекти

  - използва се нормализираща трансформация,
    т.е. обратна визуализираща трансформация
  - нормализира се визуалният обем
  - след това се изрязват и проектират възлите на обектите като се игнорира z координатата им



Етап 2

- ***Нормализиране до каноничен визуален обем***
  - ***за перспективен визуален обем***

  - нормализира се специфицираният перспективен визуален обем до единична пресечена пирамида в началото на КС насочена по оста –z
  - трансформира се перспективният визуален обем в паралелен (куб), с което се опростяват изрязването и проектирането

Етап 2

- ***Нормализиране до каноничен визуален обем***
  - ***за паралелен визуален обем***
    - необходими са няколко стъпки: за всяка стъпка се дефинира матрица на трансформация
    - произведението на матриците за отделните стъпки определя обобщената трансформационна матрица
    - стъпките са
      - *транслиране на камерата в началото на координатната система*
      - *преобразуване на (u, v, w) до (x, y, z)*
      - *мащабиране на визуалния обем в областта от стойности от –1 до 1 по осите x и y, така че задната изрязваща равнина е z = –1, а предната изрязваща равнина е z = 0*
  - ***за перспективен визуален обем***
    - същите стъпки, както при паралелния вариант + една допълнителна стъпка
      - *преобразуване на пирамидата в куб, така че перспективното изкривяване да съответства на предна изрязваща равнина z = 0*

Етап 2

- ***Нормализиране до каноничен визуален обем***
  - ***Стъпка 1***
    - ***Транслиране на камерата в началото на КС***
      - преобразува се точката Position (Pos_x, Pos_y, Pos_z) в (0, 0, 0)
      - използва се обратната матрица на визуализиращата транслационна трансформация

$$(t_x, t_y, t_z) = (-Pos_x, -Pos_y, -Pos_z)$$

    - Всички възли в сцената се преобразуват с транслационната матрица

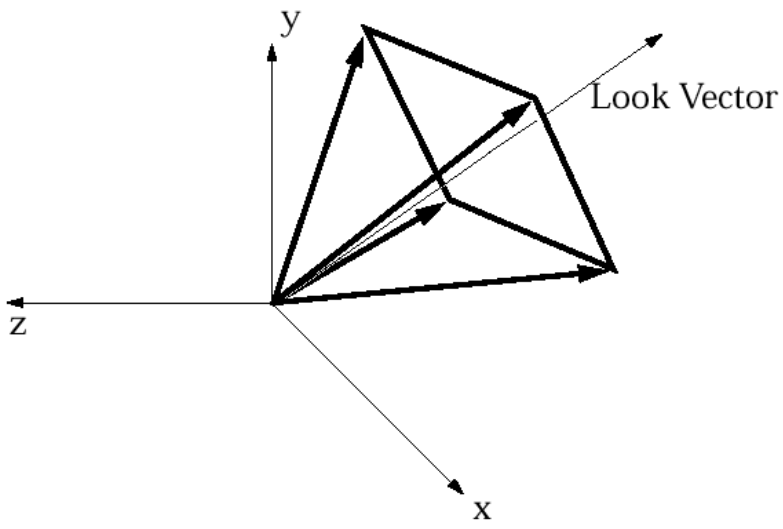
$$T_{trans} = \begin{bmatrix} 1 & 0 & 0 & -Pos_x \\ 0 & 1 & 0 & -Pos_y \\ 0 & 0 & 1 & -Pos_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad p' = T_{trans} p$$

Етап 2

- ***Нормализиране до каноничен визуален обем***
  - ***Стъпка 1***
    - ***Транслиране на камерата в началото на КС***

    - Камерата е позиционирана в началото на КС

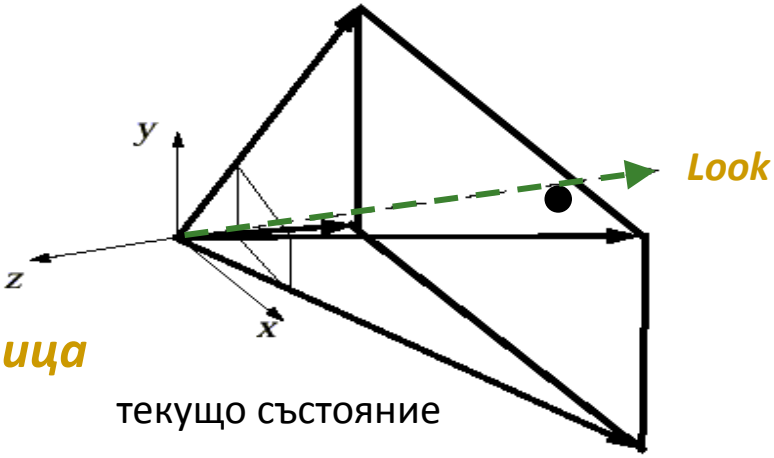
    - но координатните оси на КС на камерата не съвпадат с координатните оси на изгледа



Етап 2

- ***Нормализиране до каноничен визуален обем***
  - ***Стъпка 2***
    - ***Ротация на изгледа и изравняване със световната КС***
      - ротационна матрица M_{rot} с редове u, v, w ротира осите u, v, w в осите x, y, z

$$M_{rot} = \begin{bmatrix} u_x & u_y & u_z & 0 \\ v_x & v_y & v_z & 0 \\ w_x & w_y & w_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



текущо състояние

- Всички възли в сцената се преобразуват с ***обобщена матрица***

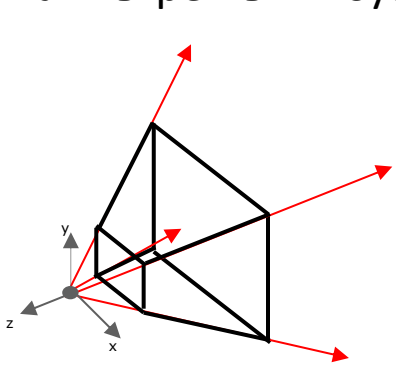
$$M_{rot} T_{trans}$$

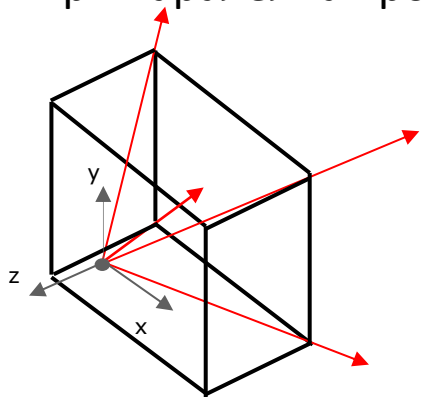
Етап 2

- ***Нормализиране до каноничен визуален обем***
  - ***Стъпка 3***
    - ***Мащабиране на пропорциите на визуалния обем***
      - нормализира се до квадратно напречно сечение с размери 2×2 единици
        - защо не квадрат с размери 1?
        - за да лежи ъгъла на далечната изрязваща равнина в (+1, +1, –1)
      - една и съща математическа операция и за паралелен, и за перспективен обем
        - мащабиране по x и y, така че векторите от началото на КС до ъглите на задната изрязваща стена да са под ъгъл 45° с осите x и y

Етап 2

- ***Нормализиране до каноничен визуален обем***
  - ***Стъпка 3***
    - ***Мащабиране на пропорциите на визуалния обем***
      - векторите от началото на КС до ъглите на задната изрязваща стена
        - ръбове на пресечената пирамида при перспективна проекция
        - лежат вътре във визуалния обем при паралелна проекция



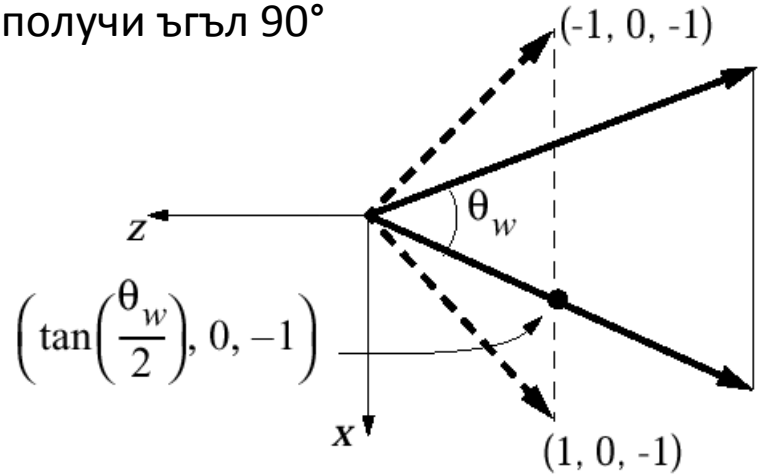


  - мащабиране по x и y, така че векторите от началото на КС до ъглите на задната изрязваща стена да са под ъгъл 45° с осите x и y

Етап 2

- ***Нормализиране до каноничен визуален обем***
  - ***Стъпка 3-1***
    - ***Мащабиране на изрязващите равнини***
      - **отделно по x и y**
        - мащабиране по оста x за да се получи ъгъл 90° с коефициент

$$\frac{1}{\left(tg\,\frac{\theta_w}{2}\right)} = cotg\left(\frac{\theta_w}{2}\right)$$



        - аналогично по оста y

Етап 2

- ***Нормализиране до каноничен визуален обем***
  - ***Стъпка 3-1***
    - ***Мащабиране на изрязващите равнини***
      - Мащабиране по x и y с мащабираща матрица

$$S_{xy} = \begin{bmatrix} cotg\left(\frac{\theta_w}{2}\right) & 0 & 0 & 0 \\ 0 & cotg\left(\frac{\theta_h}{2}\right) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

      - ***Обобщена преобразуваща матрица***

$$S_{xy} M_{rot} T_{trans}$$

Етап 2

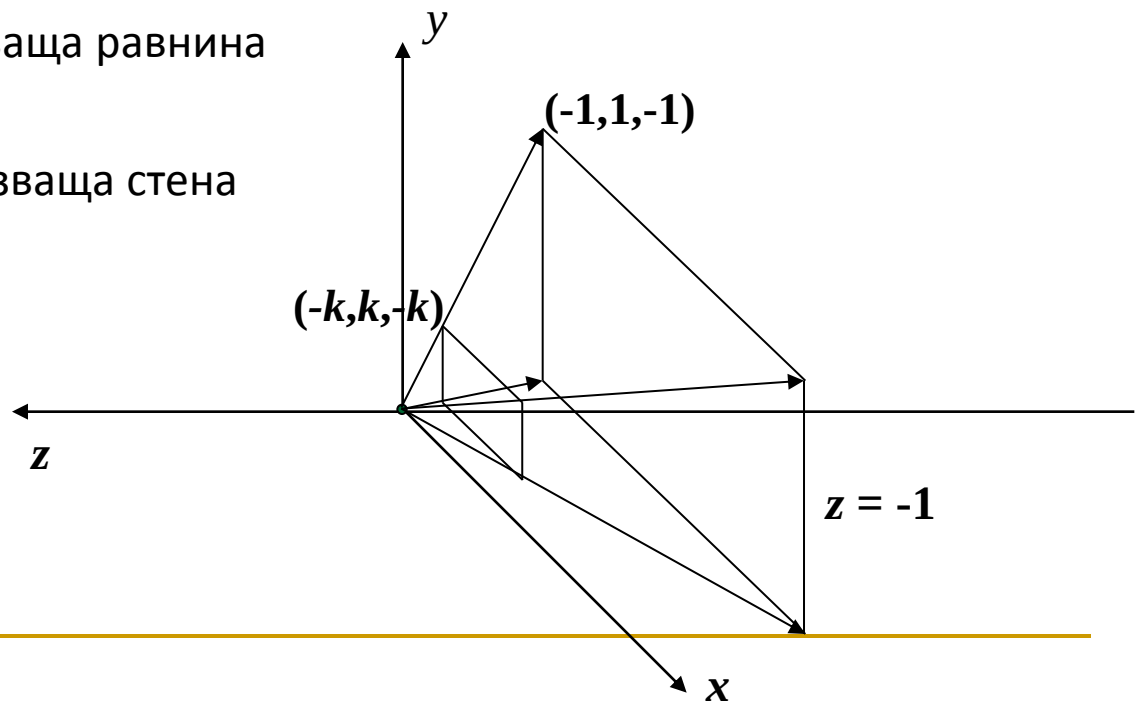
- ***Нормализиране до каноничен визуален обем***
  - ***Стъпка 3-2***
    - ***Мащабиране на задната изрязваща равнина***
      - За да имат всички точки вътре във визуалния обем координати $0 \le z \le -1$, трябва задната изрязваща стена да се смали, така че да е в равнината $z = -1$

$$S_{far} = \begin{bmatrix} \frac{1}{far} & 0 & 0 & 0 \\ 0 & \frac{1}{far} & 0 & 0 \\ 0 & 0 & \frac{1}{far} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- разстоянието от точката на наблюдение до тази точка не се е променило
- за да не се променят пропорциите на визуалния обем се мащабира не само по оста z, а пропорционално по всички оси

Етап 2

- ***Нормализиране до каноничен визуален обем***
  - ***Стъпка 3-2***
    - ***Мащабиране на задната изрязваща равнина***
      - текущо състояние
        - задна изрязваща равнина
          z = –1
        - предна изрязваща стена
          z = –k (k > 0)



Етап 2

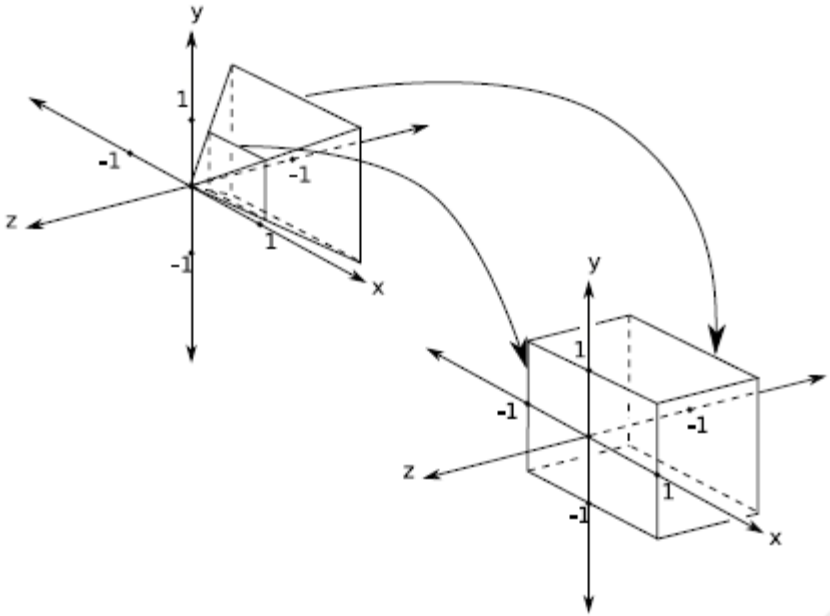
- ***Нормализиране до каноничен визуален обем***
  - ***Обобщена преобразуваща матрица***

$$S_{far} S_{xy} M_{rot} T_{trans}$$

  - T_{trans} транслира позицията на камерата в началото на световната координатна система
  - M_{rot} ориентира камерата по оста –z
  - S_{xy} мащабира изрязващите равнини, така че ъглите им да са в точките (±1, ±1)
  - S_{far} мащабира далечната изрязваща равнина така че да лежи в равнината z=–1

Етап 2

- ***Нормализиране до каноничен визуален обем***
  - ***перспективна трансформация***
    - перспективният визуален обем е преобразуван в канонична позиция, ориентация и размер
    - *последната стъпка е да се преобразува пирамидата в куб*



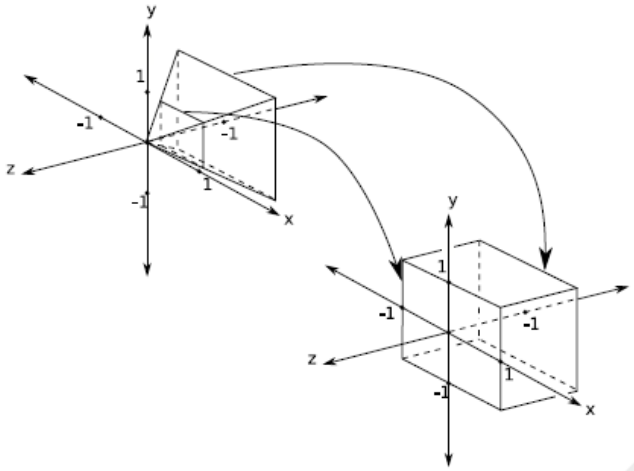
Етап 2

- ***Нормализиране до каноничен визуален обем***
  - ***перспективна трансформация***
    - Дадена точка *p* лежаща върху вектора Look в близката изрязваща равнина

$$p = Position + near \cdot Look$$

се преобразува в т.p′ $p' = S_{far} S_{xy} M_{rot} T_{trans} p$

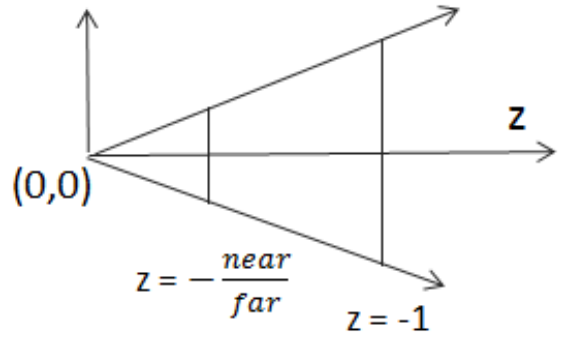
- Точка p′ е върху отрицателната посока на оста z: $p' = \begin{pmatrix} 0 & 0 & -k \end{pmatrix}$

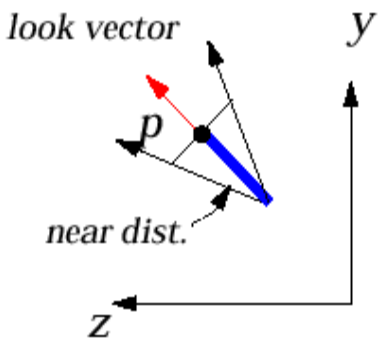


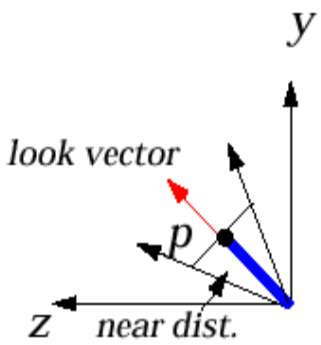
Етап 2

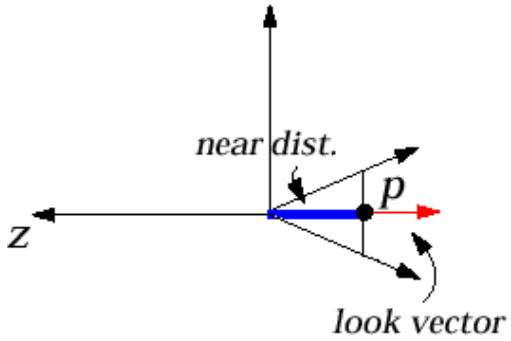
- Каква е стойността на k?
  - *Стъпка 1*: точка *p* се транслира към началото на КС , т.е. $near \cdot Look$
  - *Стъпка 2*: транслираната точка се ротира до (near)($-e_3$) (лежи в –z)
  - *Стъпка 3*: мащабирането по xy не води до промяна, мащабирането на задната изрязваща стена променя точката в $\left(-\frac{near}{far}\right)e_3$

- Следователно $k = \frac{near}{far}$



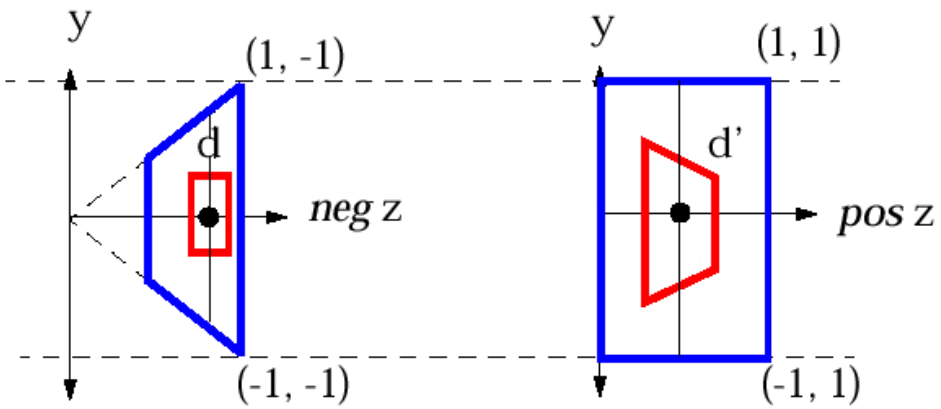






Етап 2

- Перспективната трансформация преобразува точките от стандартен перспективен визуален обем между –k и –1 до стандартен паралелен визуален обем



- алгоритъмът z-buffer за определяне на видимите повърхнини изисква стойностите на z координатите да са в интервала [0 1], а не [–1 0]
- перспективната трансформация преобразува сцената в положителния интервал $0 \leq z \leq 1$

Етап 2

- Трансформиращата матрица за перспективна трансформация е

$$D_{persp} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{k-1} & \frac{k}{k-1} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

0, а не 1!

Обръща z оста

  - стойността на к е

$$0 < k < 1$$

Етап 2

- Точка p с координати (x, y, z) преди перспективната трансформация може да се представи като

$$\begin{bmatrix} x \\ y \\ -k-d \\ 1 \end{bmatrix} \qquad 0 \le d \le 1-k$$

  - p се параметризира от разстоянието в пресечената пирамида
    - за d = 0 точката лежи в близката изрязваща равнина
    - за d = 1 – k точката лежи на далечната изрязваща равнина
    - в зависимост от стойностите на x, y, z точката може да попада вътре или извън перспективния визуален обем

Етап 2

- Перспективната трансформация преобразува т.p в т. p’

$$p' = D_{persp} \cdot \begin{bmatrix} x \\ y \\ -k-d \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ \frac{(-k-d)}{k-1} + \frac{k}{k-1} \\ -1/(-k-d) \end{bmatrix} = \begin{bmatrix} x \\ y \\ -d/(k-1) \\ k+d \end{bmatrix} \rightarrow \begin{bmatrix} x/(k+d) \\ y/(k+d) \\ -d/((k-1)(k+d)) \\ 1 \end{bmatrix}$$

  - тъй като $w = k+d \neq 1$
  - хомогенизираме координатите с разделяне на $k+d$
    - това води до промяна в перспективата по x и y
    - най-много се мащабират точките в близост до близката изрязваща равнина
    - z също се преобразува, но при проектирането в равнината на наблюдение z не се разглежда

  - Какво се получава при различни стойности на d: 0, 1–k, ½(1–k), –1, 1

Етап 2

- Перспективната трансформация преобразува т.p в т.p’

$$p' = \begin{bmatrix} x/(k+d) \\ y/(k+d) \\ -d/((k-1)(k+d)) \\ 1 \end{bmatrix}$$

  - При d→∞: x→0 и y→0
    - за стойности на d по-големи от (1-k) точката p е извън визуалния обем (такива точки се изрязват)
      - този резултат осигурява перспективното скъсяване
      - паралелните линии се пресичат в точката на сходимост

  - При d<0
    - точката p лежи пред близката изрязваща равнина,
    - възможно е да е пред точката на наблюдение
      - когато k+d<0 знаците на x и y координатите се сменят
      - тези точки не се “виждат”, защото се изрязват

Етап 2

- ***Нормализиране до каноничен визуален обем***
  - ***Краен резултат***
    - ***обобщена трансформационна матрица***

$$p' = D_{persp} S_{far} S_{xy} M_{rot} T_{trans} p$$

- при зададени параметри за визуализиране
  - Position, Up vector, Look vector, Height angle, Aspect ratio, Near, Far

матриците

$$D_{persp}, S_{far}, S_{xy}, M_{rot}, T_{trans}$$

могат да се изчислят и да се извърши умножението

- получава се единствена матрица 4x4
- прилага се за всички възли от всички обекти за преобразуването им от световни координати в каноничен паралелен визуален обем

Етап 3

- ***Изграждане на 3D изглед***

Етап 1: Специфициране на визуален обем

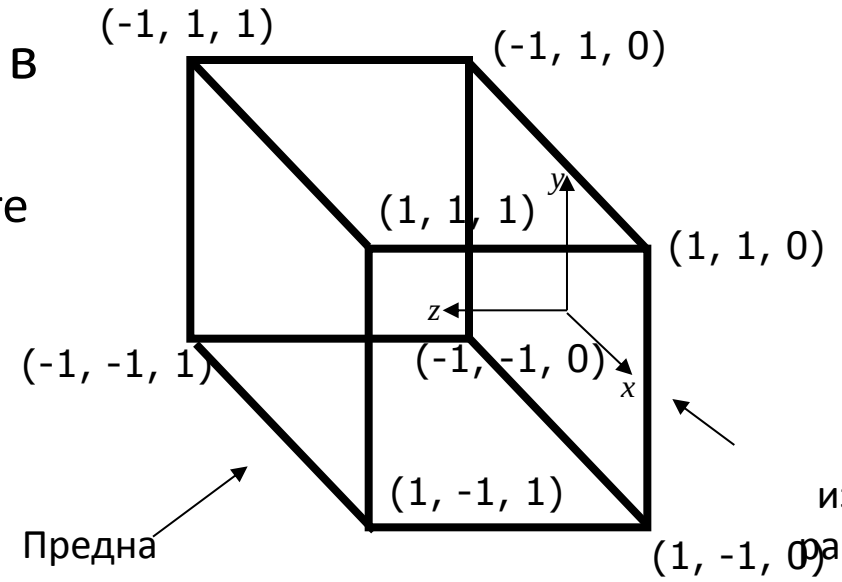
Етап 2: Трансформация от специфицирания визуален обем в каноничен визуален обем

Етап 3: Изрязване, проектиране и растеризиране на сцената за създаване на 2D изображение

Етап 3

- ***Изрязване (Clipping)***
  - сцената трябва да бъде изрязана спрямо страните на визуалния обем
  - визуалният обем е нормализиран в куб, ориентиран по координатните оси между –1 и 1 по x и y и между 0 и 1 по z

- изрязването се състои в
  - сравнение на x и y координатите на възлите от сцената с +/-1
  - сравнение на z координатата с 0 и 1



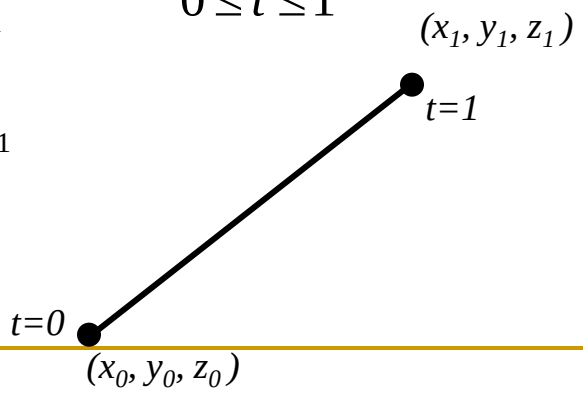
Етап 3

- ***Изрязване***
  - изрязване на точки
    - възлите с координати в допустимите интервали се запазват
    - останалите възли се изрязват
  - изрязване на отсечки
    - спрямо координатите на пресечните им точки със страните на куба
    - заменят се x, y или z с 1 в съответната параметрично линейно уравнение и се решава спрямо параметъра t

$$x = (1-t)\cdot x_0 + t\cdot x_1 \qquad 0 \le t \le 1$$

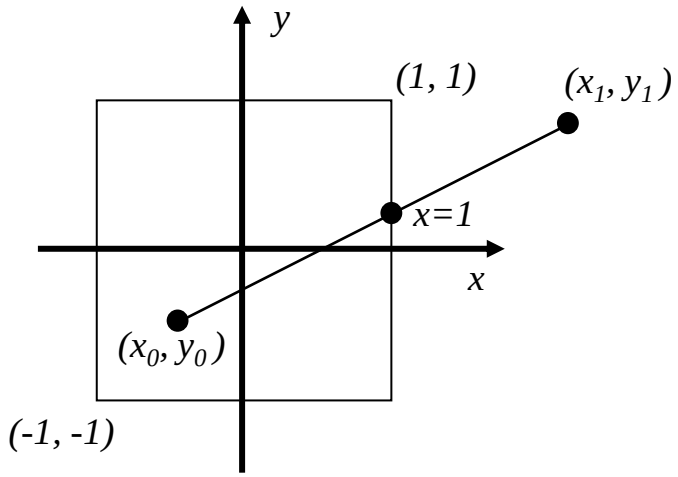
$$y = (1-t)\cdot y_0 + t\cdot y_1$$

$$z = (1-t)\cdot z_0 + t\cdot z_1$$



Етап 3

- ***Изрязване на отсечки***
  - в двумерния случай



$$1 = (1-t)x_0 + tx_1$$

$$1 - x_0 = -tx_0 + tx_1$$

$$1 - x_0 = t(x_1 - x_0)$$

$$t = \frac{1 - x_0}{x_1 - x_0}$$

Етап 3

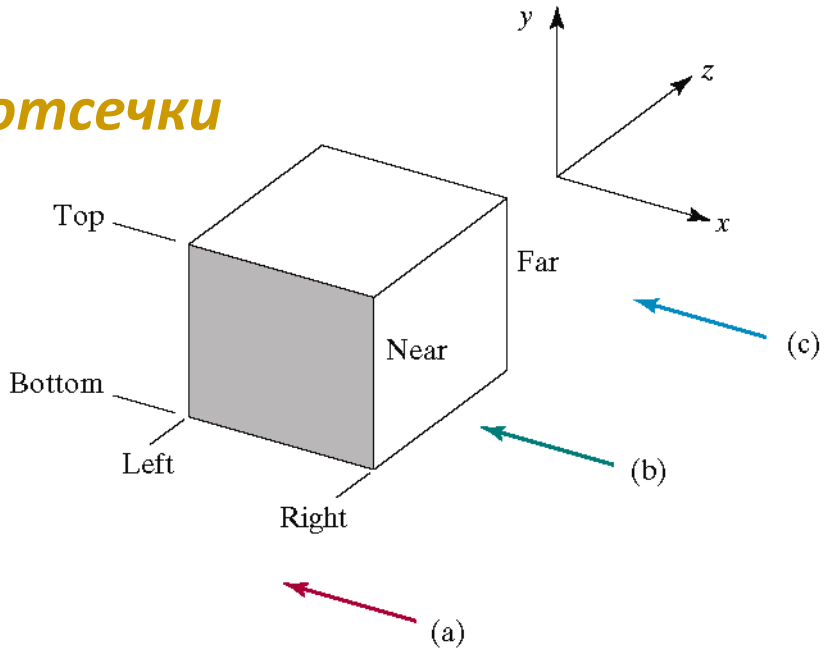
- ***Изрязване с алгоритъм на Коен-Съдърленд***
  - 6 битови кодове
  - 27 региона

bit 6	bit 5	bit 4	bit 3	bit 2	bit 1
Far	Near	Top	Bottom	Right	Left

- ***изрязване на точки***
  - тривиално сравнение
- ***изрязване на отсечки***
  - условие за тривиално приемане
    - крайни точки с кодове [0 0 0 0 0 0]
  - условие за тривиално отхвърляне
    - крайни точки с еднакъв бит в кодовете
  - по битовите кодове се определя коя от стените на куба се пресичат

Етап 3

- ***Изрязване на отсечки***



011001	011000	011010
010001	010000	010010
010101	010100	010110

Region Codes
In Front of Near Plane
(a)

001001	001000	001010
000001	000000	000010
000101	000100	000110

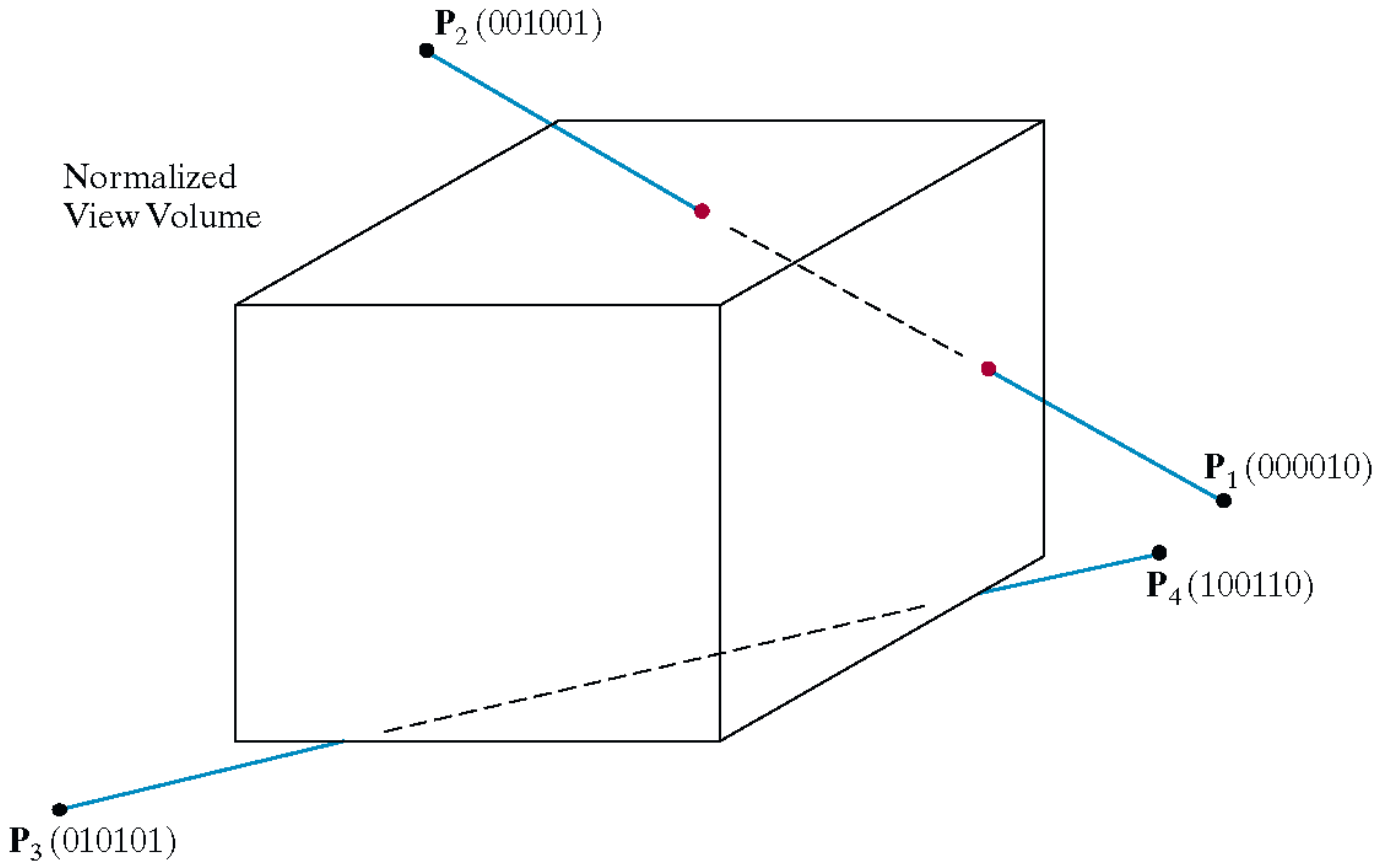
Region Codes
Between Near and Far Planes
(b)

101001	101000	101010
100001	100000	100010
100101	100100	100110

Region Codes
Behind Far Plane
(c)

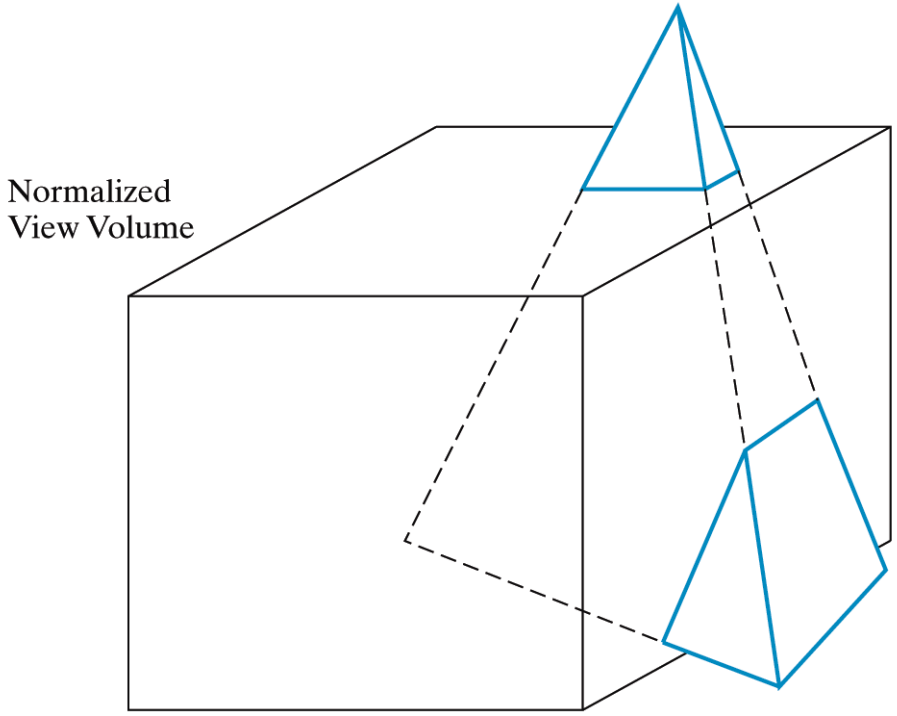
Етап 3

- ***Изрязване на отсечки***



Етап 3

- ***Изрязване на обекти, съставени от полигони***
  - условие за тривиално отхвърляне
    - ако може обекта се изрязва чрез ограждащ паралелепипед
  - в противен случай за всеки полигон се прилага алгоритъм на Съдърланд-Ходжман



Етап 3

- ***Проектиране***
  - 2D изображение на 3D сцената се създава като се игнорира z координатата на всяка точка за да се проектира в равнината xy
  - Точка с координати (x, y, z), $-1 \leq x, y \leq 1, 0 \leq z \leq 1$

    се преобразува в точка с координати (x′, y ’) в равнината на наблюдение
    - предполага се, че прозореца на наблюдение е целия екран
    - тъй като z координатата се игнорира паралелната проекция е еднаква независимо дали е върху предната, задната или друга успоредна на тях равнина
    - z координатата се запазва
      - използва се за подреждане на обектите при определяне на видимите повърхнини

3D Визуализиране

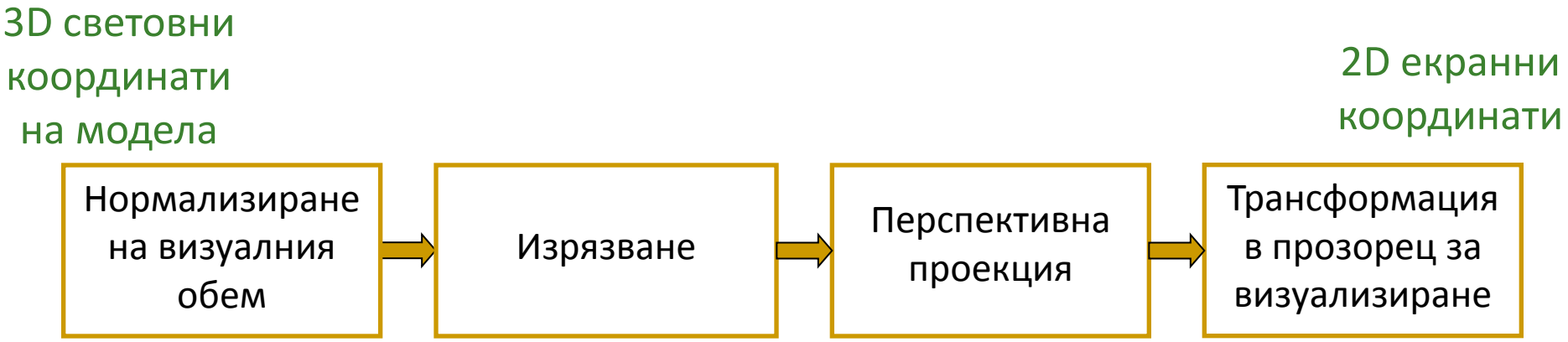
Построяване на тримерен изглед (3D viewing pipeline)

- Сцената е зададена с върховете на обектите
  - Мащабиране и ротация на обекта
  - Преобразуване от 3D до 2D изглед
    - Преобразуване до каноничен визуален обем
      - Транслация на обекта в общата сцена (световна координатна система)
      - Транслация на точката на наблюдение до началото на координатната система
      - Ротация на посоката на наблюдение до оста z
      - Мащабиране на визуалния обем до каноничен
    - Изрязване по стените на визуалния обем
    - Проектиране в равнината xy
  - Преобразуване от 2D изглед в изображение на екрана
    - Изрязване по двумерен прозорец
    - Преобразуване в екранни координати
    - Растеризация

Задачата се свежда до умножение на възлите с обобщена трансформационна матрица, изрязване и умножение с матрица за преобразуване в екранни координати

3D Визуализиране

- ***Построяване на тримерен изглед в OpenGL***

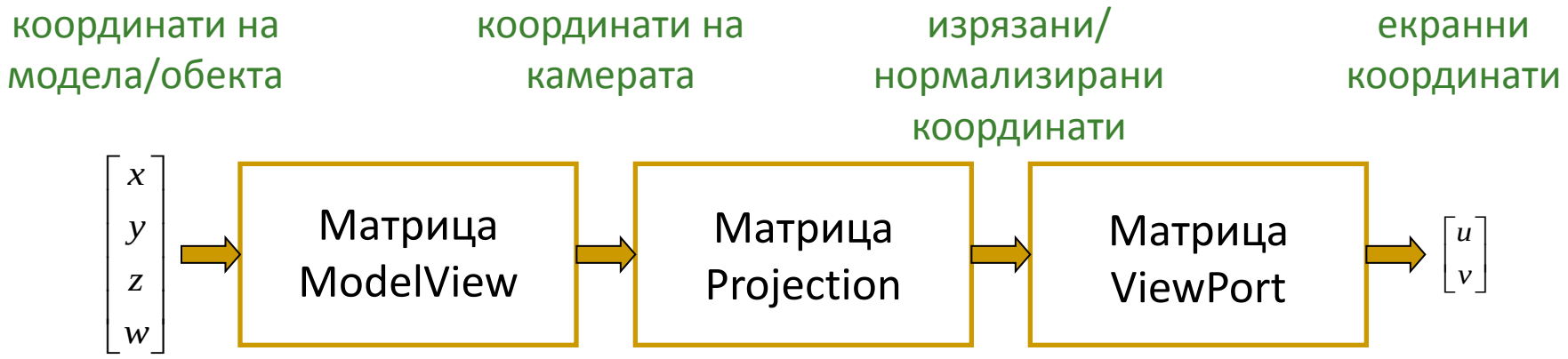



```
glOrtho();
glFrustum();
```

```
glViewport();
```

3D Визуализиране

- ***Построяване на тримерен изглед в OpenGL***



3D Визуализиране

- ***Построяване на тримерен изглед в OpenGL***
  - в OpenGL се поддържат няколко различни стека матрици
  - избор на конкретен матричен стек

    ```
    glMatrixMode(…);
    ```
    - `GL_MODELVIEW`
    - `GL_PROJECTION`
    - `GL_TEXTURE`
  - след избора промените на матриците с коя да е OpenGL команда се отнасят само за избрания стек

    ```
    glLoadIdentity();
    glScalef(x,y,z);
    glTranslate(x,y,z);
    glRotate(theta,Rx,Ry,Rz);
    ```

3D Визуализиране

- ***Построяване на тримерен изглед в OpenGL***
  - Матриците от стека *modelview* се променят чрез

```
glMatrixMode(GL_MODELVIEW);
glScalef(sx,sy,sz);
```

  - Матрицата за паралелна проекция се задава чрез

```
glMatrixMode(GL_PROJECTION);
gluOrtho2D(left,right,top,bottom);
```

  - Матрицата *viewport* дефинира трансформацията от прозорец в рамка за визуализиране

```
glViewport(x,y,width,height);
```

3D Визуализиране

- ***Построяване на тримерен изглед в OpenGL***
  - Генерирането на сцена в OpenGL може да се разглежда като заснемане с камера/фотоапарат
    - Формиране на сцената
      - ***modeling transformation***
    - Поставяне на камерата и насочване към сцената
      - ***viewing transformation***
    - Избор на камера и избор на обектив (zoom)
      - ***projection transformation***
    - Определяне на размера на фотографията
      - ***viewport transformation***
  - След дефиниране на тези трансформации, всички възли в сцената преминават през конвейера за определяне на нови координати на всеки възел

$$v' = Mv$$

3D Визуализиране

- ***Построяване на тримерен изглед в OpenGL***
- ***Трансформация на моделите на обектите в сцената***
  - ***Modeling transformation***

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glTranslatef(0.5, 0.5, 0.0);
glRotatef(rx, 1.0f, 0.0f, 0.0f);
glRotatef(ry, 0.0f, 1.0f, 0.0f);
glScalef (1.0f, 2.0f, 1.0f);
```

3D Визуализиране

- ***Построяване на тримерен изглед в OpenGL***
- ***Трансформация на моделите на обектите в сцената***
  - Функцията `glLoadIdentity()` се извиква в началото на display callback функцията за инициализиране на матрицата ModelView с единична матрица
  - Всяка от функциите `glTranslatef()` и `glRotatef()` създава нова матрица, която се умножава със съдържанието на най-горната матрица в стека ModelView (C = CM – Composite Modelling)
  - Резултатът от умножението е матрица, която се копира в матрицата най-отгоре в стека
  - Функцията `glScalef()` създава нова матрица, която се умножава със съдържанието на най-горната матрица в стека ModelView
  - Резултатът от умножението е матрица, която се копира в матрицата най-отгоре в стека

3D Визуализиране

- ***Построяване на тримерен изглед в OpenGL***
- ***Трансформация за визуализиране***
  - ***Viewing Transformation***
  - аналогична на позициониране и насочване на камера
  - по подразбиране камерата е позиционирана в началото на координатната система и е насочена по отрицателната посока на оста z
  - За определяне на позиция и посока на наблюдение в OpenGL

```
gluLookAt(ex, ey, ez, cx, cy, cz, ux, uy, uz)
```

    където

    - ex, ey, ez определят позицията на камерата
    - cx, cy, cz определят вектор Look
    - ux, uy, uz определят вектор Up

3D Визуализиране

- ***Построяване на тримерен изглед в OpenGL***

- Визуализиращата и моделиращата трансформации са обединени в OpenGL в единствена трансформация ModelView
  - с матрицата ModelView трансформацията се получава с умножение на матриците за визуализиращата и моделиращата трансформации
  - визуализиращата трансформация е преди моделиращата трансформация
    - матриците се задават в обратен ред
      - най—напред матрицата за визуализиращата трансформация
      - след това матрицата за моделиращата трансформация

3D Визуализиране

- ***Построяване на тримерен изглед в OpenGL***
- Поради реда на умножение на матрицата отгоре в стека и новата матрица се изчислява коректна матрица
  - текущо изпълняваната трансформация за всички възли

$$v' = CMv$$

  - последната зададена трансформация е първата изпълнена за възлите трансформация
  - задаването на матриците в обратен ред е правилният ред за изпълняване на трансформациите

- Най-напред се задава визуализиращата трансформация
- Първа се изпълнява моделиращата трансформация

3D Визуализиране

- ***Построяване на тримерен изглед в OpenGL***
- Трансформацията ModelView може да се разглежда като промяна на локалната координатна система на модела,

  така че всяка следваща трансформация се прилага спрямо тази координатна система
  - в горния пример координатната система е транслирана, след това ротирана спрямо транслираната координатна система
  - общата трансформация е

$$v' = S(sx, sy, sz) * R_y(ry) * R_x(rx) * T(0.5,0.5,0.5) * I * v$$

    - транслира се локалната координатна система
    - ротира се транслираната координатна система спрямо оста x
    - ротира се получената КС спрямо оста y
    - мащабира се резултантната система

3D Визуализиране

- ***Построяване на тримерен изглед в OpenGL***

- С функцията `glPushMatrix` матриците в стека се изместват с една надолу като се дублира текущата матрица
  - след изпълнение на `glPushMatrix` матрицата най-отгоре в стека е идентична с тази под нея

- С функцията `glPopMatrix` се премахва матрицата най-отгоре в стека и текуща става тази след нея

- При инициализация всички стекове с матрици съдържат само по една единична матрица

3D Визуализиране

- ***Построяване на тримерен изглед в OpenGL***

- Пример за преместване на всички обекти и преместване на камерата
  - Симулатор на полет с визуализиране на сцената, наблюдавана от пилота

```
void pilotView { GLdouble planex, GLdouble planey,
                 GLdouble planez, GLdouble roll,
                 GLdouble pitch, GLdouble heading) {
   glRotated(roll, 0.0, 1.0, 0.0);
   glRotated(pitch, 1.0, 0.0, 0.0);
   glRotated(heading, 0.0, 0.0, 1.0);
   glTranslated(-planex, -planey, -planez);
}
```

3D Визуализиране

- ***Построяване на тримерен изглед в OpenGL***

- Пример за преместване на всички обекти и преместване на камерата
  - Завъртане на камерата около обект, намиращ се в нейния център

```
void polarView { GLdouble distance, GLdouble twist,
                GLdouble elevation, GLdouble azimuth) {
   glTranslated(0.0, 0.0, -distance);
   glRotated(-twist, 0.0, 0.0, 1.0);
   glRotated(elevation, 1.0, 0.0, 0.0);
   glRotated(azimuth, 0.0, 0.0, 1.0);
}
```

3D Визуализиране

- ***Построяване на тримерен изглед в OpenGL***
- ***Паралелна проекция***
  - визуалният обем е правоъгълен паралелепипед
    - страничните стени са дефинирани чрез размерите на прозореца за визуализиране
    - близката и далечната равнини изрязват визуалния обем
    - точките вътре във визуалния обем се проектират върху равнина успоредна на прозореца за визуализиране и перпендикулярна на посоката на наблюдение (оста -z)
  - проектирането е просто игнориране на z координатата на точката
- за задаване на паралелна проекция се използват функциите

```
glOrtho(left,right,bottom,top,near,far)
gluOrtho2D(left,right,bottom,top)
```

3D Визуализиране

- ***Построяване на тримерен изглед в OpenGL***
- ***Паралелна проекция***
  - пример

```
void init() {
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   gluOrtho2D(-3.0f, 3.0, -3.0f, 3.0f);
   //glOrtho(-3.0f, 3.0, -3.0f, 3.0f, -1, 1);

   glMatrixMode(GL_MODELVIEW);
   glLoadIdentity();
}
```

3D Визуализиране

- ***Построяване на тримерен изглед в OpenGL***
- ***Перспективна проекция***
  - за задаване на перспективна проекция текущата матрица в стека се умножава с матрицата за перспективна трансформация, определена с функцията

```
glFrustum(left, right, bottom, top, nearVal, farVal);
```

  - резултатът заменя текущата матрица
  - произволен визуален обем се трансформира в нормализиран с използване на резултантната матрица

$$R = \begin{bmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & \frac{-(f+n)}{f-n} & \frac{-2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

3D Визуализиране

- ***Построяване на тримерен изглед в OpenGL***
- ***Перспективна проекция***
  - друг вариант за задаване на перспектива проекция

    `gluPerspective(fovy, aspect, zNear, zFar);`
  - задава визуалния обем в световна координатна система
    - fovy определя ъгъла на наблюдение, в градуси, в посока y
    - aspect определя аспекта на визуалния обем: аспекта е отношението на x (ширина) и y (височина)
  - генерираната матрица е

$$R = \begin{bmatrix} \frac{f}{aspect} & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & \frac{-(f+n)}{f-n} & \frac{2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

3D Визуализиране

- ***Построяване на тримерен изглед в OpenGL***
- ***Перспективна проекция***
  - пример

```
void reshape (int w, int h) {
    glViewport (0, 0, (GLsizei) w, (GLsizei) h);
    glMatrixMode (GL_PROJECTION);
    glLoadIdentity ();
    gluPerspective (40.0, (GLfloat) w /(GLfloat) h, 1.0, 20.0);
    glMatrixMode (GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt (5.0, 3.0, 5.0, 1.0, 0.5, 0.5, 0.0, 1.0, 0.0);
}
```

- Аспектът на рамката за визуализиране обикновено съвпада с аспекта на визуалния обем

3D Визуализиране

- ***Построяване на тримерен изглед в OpenGL***
- ***Перспективна проекция***
  - пример

```
void myPerspective(double fovy, double aspect,
                                double near, double far) {
   double left, right, bottom, top;
   fovy = fovy*Math.PI/180; // convert degree to arc
   top = near*Math.tan(fovy/2);
   bottom = -top;
   right = aspect*top;
   left = -right;
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   glFrustum(left, right, bottom, top, near, far);
}
```

КРАЙ

Следваща тема:

Премахване на скрити стени и обекти