ВСЕРОССИЙСКАЯ КОНФЕРЕНЦИЯ, 2014 ГОД
Исследования и практика - путь к новым знаниям

Никаноров Александр Андреевич, студент бакалавр

Национальный исследовательский университет «Высшая школа экономики»

Научный руководитель:

Ахметсафина Римма Закиевна

доцент кафедры УРПО отделения Программной инженерии

Национального исследовательского университета «Высшая школа экономики»

Москва

ПРОГРАММА ВОССТАНОВЛЕНИЯ КАРКАСНЫХ 3D ОБЪЕКТОВ ПО 2D ПРОЕКЦИЯМ

Объектом разработки является программа восстановления каркасных 3D объектов по 2D проекциям.

Цель работы – разработка приложения, которое получает на вход три ортогональные 2D проекции объекта и создает трехмерную каркасную модель на выходе.

В ходе работы проводились анализы существующих подходов к восстановлению 3D объектов: метода отображения границ и конструктивной сплошной геометрии. Также были проанализированы некоторые алгоритмы восстановления трехмерных моделей.

В результате работы была разработана программа, позволяющая загрузить три файла ортогональных проекций в одном из основных форматов хранения изображений САПР DXF и восстанавливающая трехмерную псевдокаркасную модель объекта. Пользователь может взаимодействовать с полученной моделью (вращать, масштабировать). Кроме того, пользователь

может интерактивно удалять ребра для того чтобы получить необходимую ему каркасную модель.

От своих аналогов программа отличается простотой в использовании и поддержкой операционной системой OS X.

Программа может применяться при изучении инженерной и компьютерной графики.

В дальнейшем планируется добавить в программу возможность восстановления solid объектов из 2D проекций.

Ключевые слова и фразы: каркасная модель, псевдокаркасная модель, восстановление псевдокаркасной модели, восстановление каркасной модели, восстановление каркасных 3d объектов по 2d проекциям.

Определения, обозначения и сокращения

Псевдо-каркасная модель (pseudo-wireframe) – 3D-модель, представляющая собой совокупность вершин и ребер. Может включать в себя дополнительные ребра, которые принадлежат проекции, но могут не принадлежать итоговому объекту.

Каркасная модель (wireframe) – 3D-модель, которая однозначно определяет форму отображаемого многогранного объекта.

CSG – constructive solid geometry, конструктивная сплошная геометрия – один из подходов к восстановлению 3D объектов по проекциям.

Введение

Восстановление 3D моделей по их 2D проекциям является важным объектом исследования уже долгое время. В основном свое применение эта область исследования нашла в инженерном деле. До того, как построить некоторую деталь, создаются ее различные чертежи и проекции. Для того чтобы получить более широкое представление о том, что получится в итоге, создается 3D модель, даже если этот процесс занимает значительное количество времени.

Наибольший прорыв в области восстановления 3D моделей совершили Wesley и Markowsky в своих работах [9, 12], предложив использовать каркасную модель как промежуточный шаг процесса восстановления. На сегодняшний день они являются одними из самых известных исследователей в этой области, а их подход используется при разработке практически любого нового алгоритма восстановления 3D моделей.

Процесс восстановления, который включает в себя взаимодействие пользователя с системой, чаще всего занимает значительное количество времени, вследствие чего разрабатываются различные алгоритмы для автоматического восстановления 3D моделей. Эти алгоритмы могут быть разделены на два типа в зависимости от существующих подходов. На данных момент существует два основных похода к восстановление 3D моделей: метод отображения границ и метод конструктивной сплошной геометрии. Метод отображения границ состоит в работе с линиями (ребрами) и вершинами, в свою очередь метод конструктивной сплошной геометрии основан на работе с геометрическими примитивами, такими как куб или шар.

Можно выделить две основные стадии восстановления 3D объектов по проекциям: создание псевдокаркасной модели объекта (pseudo-wireframe) и создание объекта (solid). Псевдокаркасная модель состоит из набора вершин и ребер, соединяющих эти вершины. Она может содержать лишнюю, но в то же время более полную информацию об описываемом объекте. В отличие от

каркасной модели, она может включать в себя «лишние» ребра, которые принадлежат описывающим ее проекциям (рис. 1) Одна псевдокаркасная модель может описывать большое количество 3D объектов (рис .2).

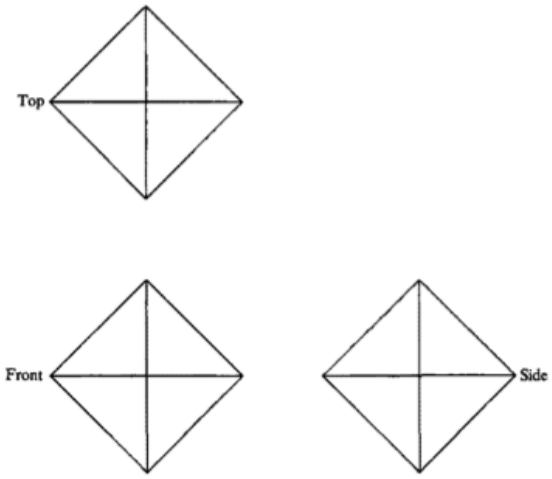

Рисунок 1. Три проекции октаэдра

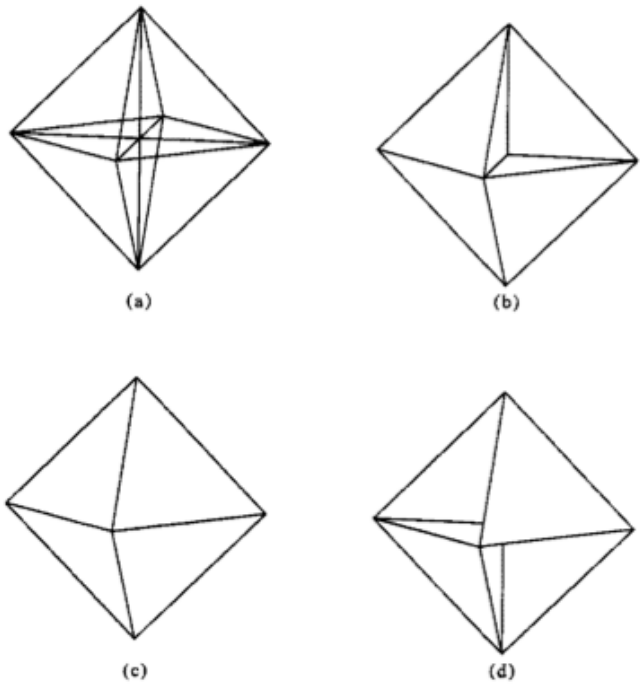

Рисунок 2. (a) – псевдокаркасная модель октаэдра, (b, c, d) – возможные solid объекты, этой модели

При анализе существующих приложений, предоставляющих возможность работы с 3D моделями, было выделено несколько самых популярных решений: Autodesk 3DS Max, Autodesk Autocad, Kompas-3D, Matlab, Blender (табл. 1).

Таблица 1.

Сравнительная таблица существующих решений

Приложения	Windows	OS X	Необходимость специальных знаний	Восстановление каркасной модели	Автоматическое выполнение функции	Лицензия
3DS Max	+	-	+	+	+	+
Autocad	+	+	+	+	-	+
Kompas-3D	+	-	+	-	-	+
Blender	+	-	+	-	-	-
Matlab	+	+	+	+	-	+

Основным недостатком всех перечисленных систем за исключением последней является их платное распространение. Кроме этого Kompas-3D и Blender не имеют функционала восстановления 3D моделей из 2D проекций.

Autocad широко используется в сфере инженерного проектирования, однако реализованная в этой программе система восстановления 3D моделей из проекций основана на взаимодействии пользователя с системой.

3DS Max является практически самой распространенной и широко используемой системой для работы с трехмерной графикой. Программа предоставляет возможность восстанавливать 3D модель по проекциям, однако имеет довольно сложный интерфейс, а также работает только в операционной системе Windows.

Matlab - кроссплатформенный пакет программ, который, как и Autocad широко используется в сфере инженерного проектирования. Кроме этого Matlab является высокоуровневым интерпретируемым языком программирования, поэтому для работы необходимы соответствующие знания языка.

Таким образом, очевидна актуальность разработки свободно распространяемой программы восстановления каркасных 3D объектов по трем 2D проекциям для операционной системы OS X, которая была бы доступна для работы пользователю без специальной подготовки. Выбор операционной

системы обуславливается тем, что из перечисленных приложений ее поддерживают только Autocad, Blender и Matlab. А эти приложения либо не предоставляют необходимой функциональности, либо требуют дополнительных знаний для работы с ними, либо не распространяются свободно.

Целью данной работы является разработка программы восстановления каркасных 3D объектов, состоящих из многогранников, по 2D ортогональным проекциям для операционной системы OS X.

Для достижения цели работы необходимо решить следующие задачи:

1. Изучить существующие форматы представления инженерных чертежей;
2. Рассмотреть основные подходы в восстановлении 3D объектов по 2D проекциям;
3. Изучить существующие алгоритмы восстановления трехмерной модели по трем двухмерным ортогональным проекциям;
4. Разработать программу, принимающую на вход три 2D проекции (вид сверху, вид спереди, вид сбоку) в одном из базовых CAD форматов (DXF) и восстанавливающую корректную каркасную 3D модель заданного объекта на выходе.
5. Протестировать работу приложения при помощи известных примеров.

1. Обзор существующих подходов и программных решений

1.1. Существующие подходы

1.1.1. Метод отображения границ

Одни из первых исследований в области восстановления 3D моделей по проекциям были проведены профессором Masanori Idesawa в 1973 году и были представлены в его работе “A system to generate a solid figure from three views” [7,8]. Idesawa предложил способы восстановления трехмерных вершин, ребер и лицевых поверхностей для многогранных объектов. Кроме того, так как некоторый набор вершин и ребер (каркасная модель) может описывать несколько объектов, он предложи способ, как удалять “призрачные” ребра для того чтобы в итоге получать уникальный объект.

Это был один из первых подходов к восстановлению 3D объектов. Впоследствии алгоритмы, в которых происходит работа с ребрами, получили общее название Boundary representation (Метод отображения границ). Модели в методе отображения границ создаются при помощи топологии и геометрических объектов. Основными топологическими единицами являются: faces (лицевые грани, или поверхности), edges (ребра) и vertices(вершины). Face – это ограниченная часть поверхности, относящаяся в некоторому ребру, ребро – сторона грани. В 1980 году George Markowky, профессор университета Мэна, и Michael A. Wesley из исследовательского института IBM Томаса Уотсона в своей работе “Fleshing out Wireframes” [9,12] предложили использовать псевдокаркасную модель как промежуточную стадию процесса восстановления 3D объекта. Эта работа до сих пор является основополагающей для разработки новых алгоритмов в рамках метода отображения границ.

1.1.2. Конструктивная сплошная геометрия

Первые исследования в рамках второго подхода к восстановлению 3D объектов по 2D проекциям были произведены Aldefel в 1983 году [1]. Его алгоритм работает только для объектов равномерной толщины. Подход носит название Конструктивная сплошная геометрия (Constructive solid geometry | CSG), и его основной концепцией является возможность математического описания любых сложных объектов при помощи более простых. Простейшими телами в CSG являются примитивы – тела простой формы, такие как куб, сфера, цилиндр, призма. Более сложные объекты создаются при помощи применения булевых операций (объединение, пересечение, разность) к некоторому набору примитивов.

Как конструктивная сплошная геометрия, так и метод отображения границ используются для создания итоговых 3D объектов. Как правило, восстановление псевдокаркасной модели является этапом работы алгоритмов обоих подходов. Однако задачей работы является восстановления каркасных моделей из 2D проекций. Основной вклад в разработку алгоритма восстановления каркасной модели внесли Markowsky и Wesley. При разработке последующих алгоритмов именно их подход всегда используется при восстановлении каркасных моделей объектов. Поэтому именно этот алгоритм был выбран для разработки приложения.

2. Описание используемых алгоритмов

2.1. Извлечение 2D ортографической информации

Каркасная модель W(O) объекта O строится по фронтальной проекции (FV – front view), боковой проекции – виду справа (SV – side view) и верхней проекции – вид сверху (TV – top view), которые подаются на вход программе в формате систем автоматизированного проектирования (САПР) DXF. Файл каждой проекции необходимо проанализировать и загрузить информацию о хранящихся в нем геометрических компонентах – вершинах и ребрах – в матрицы $\mathbf{FI}(O), \mathbf{TI}(O)$ и $\mathbf{SI}(O)$:

$$\mathbf{FI}(O) = \begin{bmatrix} x_{1,1} & y_{1,1} & z_{1,1} \\ x_{1,2} & y_{1,2} & z_{1,2} \\ \vdots & \vdots & \vdots \\ x_{n,1} & y_{n,1} & z_{n,1} \\ x_{n,2} & y_{n,2} & z_{n,2} \end{bmatrix},$$

где $x_{i,1}, y_{i,1}, z_{i,1}$ - координаты вершины - начала ребра,

$x_{i,2}, y_{i,2}, z_{i,2}$ - координаты вершины - конца ребра,

где $i = 1,2 \dots n$,

n – количество ребер в проекции.

Каждое ребро задается координатами двух вершин, координаты одной вершины могут встречаться в описании несколько раз по количеству сходящих в ней ребер.

2.2. Разметка 2D граней и вершин

Для того чтобы уменьшить количество параметров, определяющих ребра, каждая вершина отмечается порядковым номером. Номера вершин будут описывать ребра вместо координат этих вершин. Кроме того, сами ребра также отмечаются порядковыми номерами. Таким образом для каждой проекции создаются матрицы $\mathbf{FV}(O), \mathbf{TV}(O), \mathbf{SV}(O)$ и $\mathbf{FE}(O), \mathbf{TE}(O), \mathbf{SE}(O)$ для хранения информации о вершинах и ребрах соответственно:

$$\mathbf{FV}(0) = \begin{bmatrix} nv_1 & x_1 & y_1 & z_1 \\ \vdots & \vdots & \vdots & \vdots \\ nv_k & x_k & y_k & z_k \end{bmatrix},$$

где nv_i – порядковый номер i-ой вершины,

x_i, y_i, z_i - координаты i-ой вершины, i =1,2,…,k

k – количество вершин проекции.

$$\mathbf{FE}(0) = \begin{bmatrix} oe_1 \\ \vdots \\ oe_n \end{bmatrix},$$

где oe_i - объект типа «ребро», i = 1,2,…,n

n – количество ребер проекции.

Каждый объект типа «ребро» содержит4 следующую информацию:

$$oe_i = [ne_i \quad nv_{i,1} \quad nv_{i,2} \quad et_i],$$

где ne_i - порядковый номер ребра,

$nv_{i,1}$ – порядковый номер вершины, определяющей начало ребра,

$nv_{i,2}$ - порядковый номер вершины, определяющей конец ребра,

et_i - тип ребра,

i = 1,2,…,n,

n – количество ребер проекции.

Таким образом матрица вершин содержит порядковые номера вершин проекции и их координаты, а матрица ребер – порядковые номера ребер и порядковые номера вершин, определяющих эти ребра. Тип ребер предлагается запоминать для обработки пунктирных линий[1] на проекции, которые обычно используются для обозначения ребер, находящихся не на лицевых гранях проекции.

Координаты вершин проекций могут значительно отличаться, поэтому для упрощения работы с ними предлагается сдвинуть каждую проекцию таким

[1] В системе Autocad линии, которые в своем названии имеют подстроку “DASHED”.

образом, чтобы она находилась в первом квадранте и хотя бы одна точка лежала на оси X и хотя бы одна точка лежала на оси Y. Для этого необходимо найти минимальную координату x_{min} и минимальную координату y_{min} вершин в каждой проекции. После этого необходимо выполнить следующую операцию для матриц каждой проекции:

$$\forall \text{fv}[i,2] \in \mathbf{FV}(0) |\ \text{fv}[i,2] = \text{fv}[i,2] - x_{min}\ ,$$

$$\forall \text{fv}[i,3] \in \mathbf{FV}(0) |\ \text{fv}[i,3] = \text{fv}[i,3] - y_{min}\ ,$$

где fv[i,2] – элемент i-ой строки и второго столбца матрицы $\mathbf{FV}(0)$,

fv[i,3] – элемент i-ой строки и третьего столбца матрицы $\mathbf{FV}(0)$.

Впоследствии ребра проекции будут сравниваться с ребрами каркасной модели. Так как одно ребро проекции может описывать большое количество ребер в каркасной модели, ребра проекции должны быть максимальной длины, а не множеством коротких отрезков, лежащих на одной прямой.

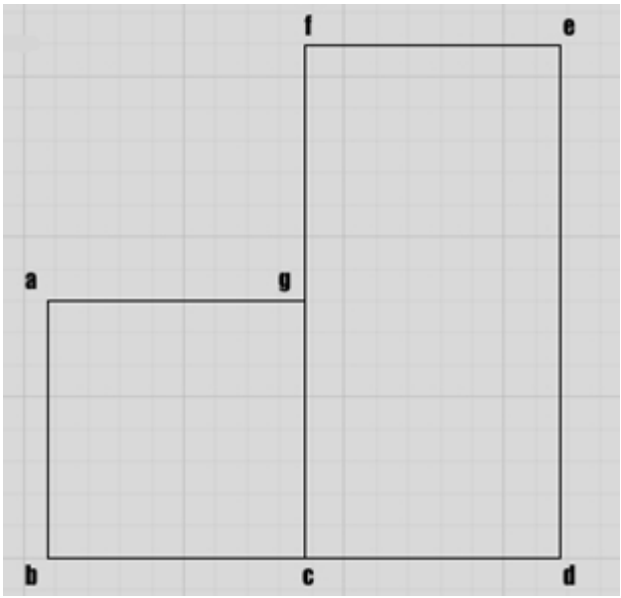

Рисунок 3. Вид сбоку

Например, на рисунке 3 представлена проекция некоторого объекта. В конечной матрице проекции не должно существовать несколько ребер, лежащих на одной прямой и пересекающихся в вершинах, таких как bc, cd и fg, gc. Если они были обнаружены, то оба ребра, лежащие на одной прямой, удаляются из $\mathbf{SV}(0)$, и добавляются ребра bd и fc соответственно.

2.3. Объединение граней и вершин

Для последующей работы с данными о проекциях необходимо создать две новых матрицы $\mathbf{RTV}(0) = \mathbf{RV}(0)$ и $\mathbf{RSV}(0) = \mathbf{SV}(0)$ для верхней и боковой проекции соответственно. После этого нужно выполнить следующие операции над новыми матрицами:

$$\forall \mathrm{rtv}[i,3] \in \mathbf{RTV}(0), \mathrm{rtv}[i,4] \in \mathbf{RTV}(0) | \mathrm{rtv}[i,4] = \mathrm{rtv}[i,3], \mathrm{rtv}[i,3] = 0,$$

$$\forall \mathrm{rsv[i,2]} \in \mathbf{RSV}(0), \mathrm{rsv}[i,4] \in \mathbf{RSV}(0) | \mathrm{rsv}[i,4] = \mathrm{rsv}[i,2], \mathrm{rsv}[i,2] = 0,$$

где rtv[i,3] – элемент i-ой строки и третьего столбца матрицы $\mathbf{RTV}(0)$ (координата y),

rtv[i,4] – элемент i-ой строки и четвертого столбца матрицы $\mathbf{RTV}(0)$ (координата z),

rsv[i,2] – элемент i-ой строки и второго столбца матрицы $\mathbf{RSV}(0)$ (координата x),

rsv[i,4] – элемент i-ой 4строки и четвертого столбца матрицы $\mathbf{RTV}(0)$ (координата z).

Предполагается, что проекции, полученные в начале работы программы из файлов, созданы в едином масштабе. Для упрощения работы с каркасной моделью необходимо перенести данные о проекциях из матриц вершин $\mathbf{FV}(0), \mathbf{RTV}(0), \mathbf{RSV}(0)$ в общую матрицу вершин $\mathbf{V}(0)$ и из матриц ребер $\mathbf{FE}(0), \mathbf{TE}(0), \mathbf{SE}(0)$ в матрицу ребер $\mathbf{E}(0)$.

Данные о проекциях расположены в $\mathbf{V}(0)$ и $\mathbf{E}(0)$ последовательно, поэтому теперь легко можно провести перпендикуляры из всех вершин: по z для фронтальной проекции, по y вверх для верхней проекции и по x вправо для боковой проекции, - и добавить новые вершины в $\mathbf{V}(0)$, а новые ребра в $\mathbf{E}(0)$.

На этом этапе работы алгоритма часть ребер никак не связана между собой, а также существуют повторяющиеся ребра, поэтому нужно удалить повторяющиеся вершины из $\mathbf{V}(0)$ и заменить порядковые номера этих вершин в $\mathbf{E}(0)$ на соответствующие не удаленные. Кроме того, нужно удалить повторяющиеся ребра из $\mathbf{E}(0)$.

2.4. Нахождение точек пересечения ребер

Перпендикуляры, проведенные на предыдущем шаге все еще никак между собой не связны, поэтому необходимо найти точки их пересечения. Заранее не известно, какие именно ребра пересекаются; для каждого ребра проверяются все остальные, еще не проверенные ребра, на предмет их пересечения. В этой проверке можно выделить четыре случая: ребра параллельны / находятся на одной прямой / скрещиваются (лежат в параллельных плоскостях) / пересекаются. Проверку нужно провести только на нахождение ребер на одной прямой и на пересечение.

2.4.1. Определение нахождения ребер на одной прямой

Уравнение прямой, проходящей через две точки, в трехмерном пространстве выглядит следующим образом:

$$\frac{x - x_1}{x_2 - x_1} = \frac{y - y_1}{y_2 - y_1} = \frac{z - z_1}{z_2 - z_1},$$

где x_1, y_1, z_1 - координаты первой точки,

x_2, y_2, z_2 - координаты второй точки.

Так как проверяемые ребра (восстановленные перпендикуляры) параллельны хотя бы одной оси координат, в этой формуле будет происходить деление на ноль, во избежание этого нужно воспользоваться правилом пропорции:

$$(x - x_1) * (y_2 - y_1) * (z_2 - z_1) = (y - y_1) * (x_2 - x_1) * (z_2 - z_1) = (z - z_1) * (y_2 - y_1) * (x_2 - x_1) \quad (2).$$

Допустим, сравнивается ребро AB и ребро CD, если подставить в (2) вместо x_1, y_1, z_1 координаты вершины A, вместо x_2, y_2, z_2 координаты вершины B, а вместо x, y, z поочередно координаты вершин C и D, и оба выражения будут выполняться, то ребра лежат на одной прямой.

В этом случае ребра могут не пересекаются. Если же ребра ab и cd пересекаются могут возникнуть пять ситуаций:

- ребра пересекаются в одной из существующих вершин (рис. 4);

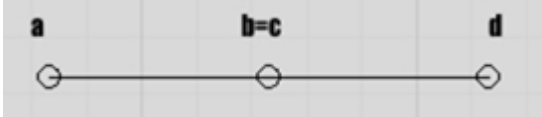

Рисунок 4.

- ребра совпадают, в этом случае второе ребро удаляется из **E**(O);
- одна из вершин одного ребра находится на другом ребре, к примеру вершины ребер ab и cd расположены в следующем порядке: a c b d, в этом случае ребра ab и cd удаляются из **E**(O), а новые ребра ac, cb и bd добавляются в матрицу (рис. 5);

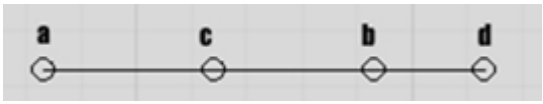

Рисунок 5.

- одно ребро находится внутри другого, но вершины не совпадают, в этом случае меньшее ребро удаляется из **E**(O). Например на рис. 6 нужно удалить ребро ab;

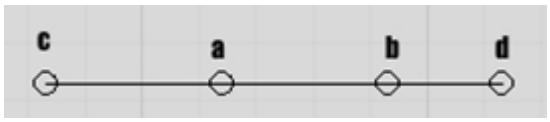

Рисунок 6

- одно ребро находится внутри другого, и одна из вершин ребра ab совпадает с одной из вершин ребра cd, к примеру если a = c и вершины расположены в порядке: a=c, d, b (рис. 7), ребро ab удаляется из **E**(O), а новое ребро db добавляется в матрицу.

Рисунок 7.

2.4.2. Определение пересечения ребер

Две прямые AB и CD:

$$\frac{x-x_1}{x_2-x_1}=\frac{y-y_1}{y_2-y_1}=\frac{z-z_1}{z_2-z_1} \text{ и } \frac{x-x_3}{x_4-x_3}=\frac{y-y_3}{y_4-y_3}=\frac{z-z_3}{z_4-z_3},$$

проходящие через точки x_1, y_1, z_1 , x_2, y_2, z_2 и x_3, y_3, z_3 , x_4, y_4, z_4соответственно, пересекаются, если выполняется условие:

$$\begin{vmatrix} x_3-x_1 & y_3-y_1 & z_3-z_1 \\ x_2-x_1 & y_2-y_1 & z_2-z_1 \\ x_4-x_3 & y_4-y_3 & z_4-z_3 \end{vmatrix} = 0,$$

и если ранг этой матрицы равен двум. Иными словами, для прямых AB и CD нужно найти такие коэффициенты u и v, что система из трех уравнений (3) имеет решение:

$$\begin{aligned} x_1+u*(x_2-x_1)&=x_3+v*(x_4-x_3)\\ y_1+u*(y_2-y_1)&=y_3+v*(y_4-y_3)\\ z_1+u*(z_2-z_1)&=z_3+v*(z_4-z_3) \end{aligned} \quad (3)$$

Определитель матрицы 3x3 считается по формуле (4):

$$\Delta=\begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} = a_{11}*\begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix} - a_{12}*\begin{vmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{vmatrix} + a_{13}*\begin{vmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{vmatrix} =$$

$$= a_{11}*a_{22}*a_{33} - a_{11}*a_{23}*a_{32} - a_{12}*a_{21}*a_{32} + a_{12}*a_{23}*a_{31} + a_{13}*a_{21}*a_{32} - a_{13}*a_{22}*a_{31} \quad (4).$$

Ранг матрицы равен количеству ненулевых строк в матрице ступенчатого вида. Для того чтобы привести матрицу к ступенчатому виду можно воспользоваться методом Гаусса, для этого введем элементарные преобразования матрицы:

I. Перестановка двух столбцов или строк матрицы;

II. Умножение элементов одной строки или столбца на одно и то же число, отличное от нуля;

III. Прибавление к элементами одной строки или столбца соответствующих элементов другой строки или столбца, умноженных на одно и то же число.

Для того чтобы привести матрицу к ступенчатому виду методом Гаусса необходимо выполнить следующие шаги:

1. Выбрать в первом столбце элемент, отличный от нуля (ведущий элемент). Если строка с ведущим элементов (ведущая строка) не первая, переставить ее на место первой строки (преобразование I типа). Если в первом столбце нет ведущего элемента, то продолжить поиск в оставшейся части матрицы. Нужно закончить преобразования если исключены все столбцы.

2. Разделить все элементы ведущей строки на ведущий элемент. Если строка последняя, то преобразования заканчиваются.

3. К каждой строке ниже ведущей строки прибавить ведущую строку, умноженное на такое число, что все элементы ниже ведущего элемента становятся равны нулю.

4. Исключив ведущую строку и ведущий столбец, перейти к пункту 1 для оставшейся части матрицы.

Если определитель равен нулю и ранг матрицы равен двум, систему уравнений (2) можно решить при помощи формул Крамера [1]. Для решения системы из трех уравнений с двумя неизвестными достаточно двух уравнений, поэтому решение, использующее первое и второе уравнение будет иметь вид:

$$u = \begin{vmatrix} x_3 - x_1 & x_3 - x_4 \\ y_3 - y_1 & y_3 - y_4 \end{vmatrix} / \mathrm{D}$$

$$v = \begin{vmatrix} x_2 - x_1 & x_3 - x_1 \\ y_2 - y_1 & y_3 - y_1 \end{vmatrix} / \mathrm{D}$$

$$\mathrm{D} = \begin{vmatrix} x_2 - x_1 & x_3 - x_4 \\ y_2 - y_1 & y_3 - y_4 \end{vmatrix}.$$

Если плоскость, в которой находятся пересекающиеся ребра, перпендикулярна плоскости XY, то D будет равен нулю. Аналогично плоскость пересечения может быть перпендикулярна плоскостям XZ и YZ. Поэтому для нахождения точки пересечения выбираем пару уравнений, где D не равен нулю. Таким образом, решение для первого и третьего уравнений:

$$u = \begin{vmatrix} x_3 - x_1 & x_3 - x_4 \\ z_3 - z_1 & z_3 - z_4 \end{vmatrix} /\mathrm{D}$$

$$v = \begin{vmatrix} x_2 - x_1 & x_3 - x_1 \\ z_2 - z_1 & z_3 - z_1 \end{vmatrix} /\mathrm{D}$$

$$\mathrm{D} = \begin{vmatrix} x_2 - x_1 & x_3 - x_4 \\ z_2 - z_1 & z_3 - z_4 \end{vmatrix}.$$

Решение для второго и третьего уравнений:

$$u = \begin{vmatrix} y_3 - y_1 & y_3 - y_4 \\ z_3 - z_1 & z_3 - z_4 \end{vmatrix} /\mathrm{D}$$

$$v = \begin{vmatrix} y_2 - y_1 & y_3 - y_1 \\ z_2 - z_1 & z_3 - z_1 \end{vmatrix} /\mathrm{D}$$

$$\mathrm{D} = \begin{vmatrix} y_2 - y_1 & y_3 - y_4 \\ z_2 - z_1 & z_3 - z_4 \end{vmatrix}.$$

В случае пересечения ребер возможны четыре случая:

- Прямые, определенные вершинами ребер пересекаются, но сами ребра не пересекаются;
- Ребра пересекаются в одной из их вершин. Если ребро ab пересекает cd в точке a (рис. 8), то cd удаляется из **E**(O), а новые ребра ca и ad добавляются в список;

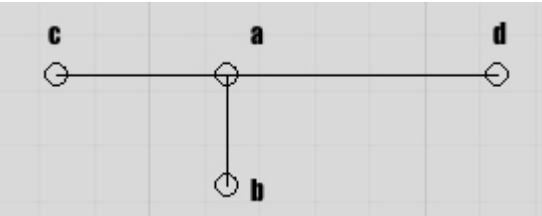

Рисунок 8.

- Ребра пересекаются в новой вершине. Если ребро ab пересекает cd в некоторой точке m (рис. 9), ab и cd удаляются из **E**(O), а новые ребра am, mb, cm, md добавляются в список;

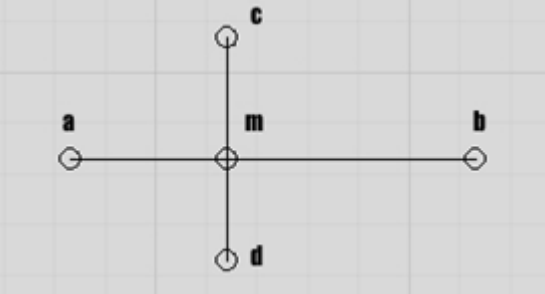

Рисунок 9.

- Ребра пересекаются двумя вершинами (рис. 10).

Рисунок 10.

Таким образом в каркасную модель добавлены все ребра, ортогональные ко всем трем проекциям. Следующим шагом является добавление ребер, не ортогональных ни к одной проекции.

2.5. Создание не ортогональных ребер

Для того чтобы создать все возможные ребра не ортогональные ни к одной плоскости проекции, необходимо спроецировать вершины **V**(O) на каждую из плоскостей XY, XZ и YZ. Под проецированием понимается приравнивание к нулю одной из координат трехмерной вершины, Таким образом при проецировании на плоскость XY координата z приравнивается к нулю, при проецировании на XZ координата y приравнивается к нулю, при проецировании на YZ координата x приравнивается к нулю.

После этого для каждой вершины каркасной модели перебираются вершины соответствующей проекции (**FV**(O) для XY, **TV**(O) для XZ, **SV**(O) для

YZ). Для каждой *i*-ой вершины проекции выделяются координаты вершин, соответствующие плоскости, на которую модель спроецирована, которые соединяются ребрами с *i*-ой вершиной (соседних вершин). В каркасной модели создаются ребра, соединяющие *i*-ую вершину и все вершины с выделенными двухмерными координатами.

На рисунке 11 приведен пример проекций некоторого объекта (a,b,c) и результат, который был достигнут на момент выполнения шага 2.4.2.

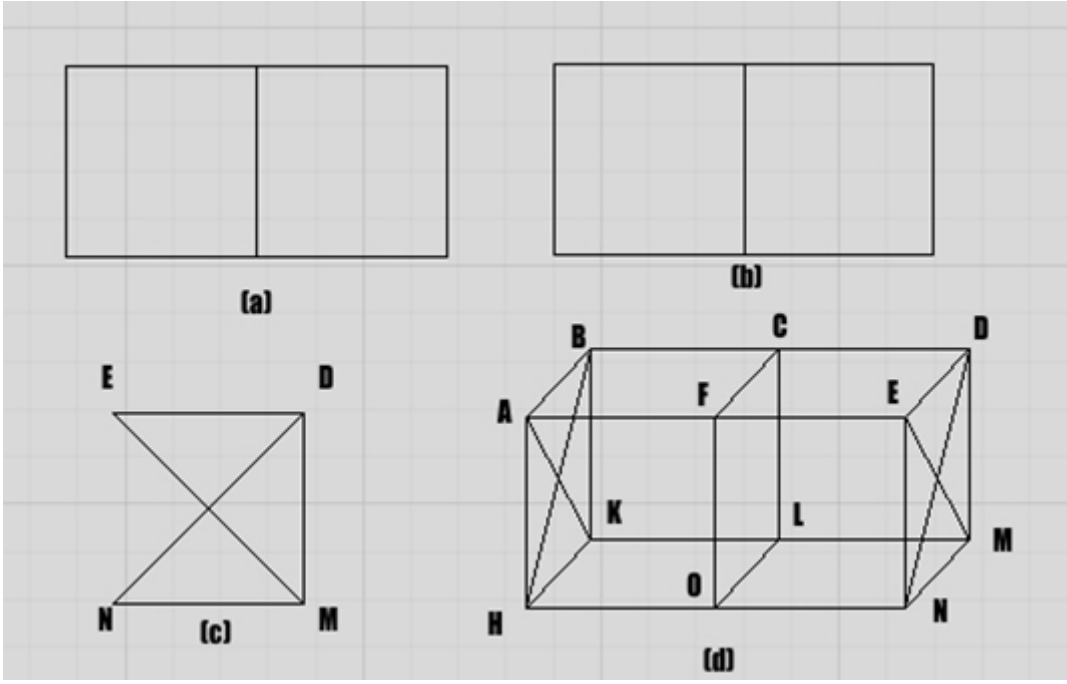

Рисунок 11. (a) – фронтальная проекция, (b) – вид сверху, (c) – вид сбоку, (d) – текущая каркасная модель.

На текущем этапе алгоритма необходимо восстановить ребра FL и CO. Из модели видно, что для этого необходимо спроецировать модель на плоскость YZ (боковая проекция). В этом случае координаты y и z каждой вершины сравниваются с соответствующими координатами матрицы **RSV**(0). К примеру вершина C по этим координатам равна вершине D проекции. Для вершины D находятся все вершины, которые соединяются ребрами с D. На проекции это вершины E, N, M. После этого в каркасной модели находятся все вершины, чьи координаты y и z (в данном случае) равны координатам y и z вершин E, N или M. Из проверяемой вершины C проводятся все возможные

ребра к найденным вершинам, если таких ребер еще не существует. Таким образом в каркасную модель добавятся ребра CA, CE, CK, CM, CH, CO, CN (рис. 12).

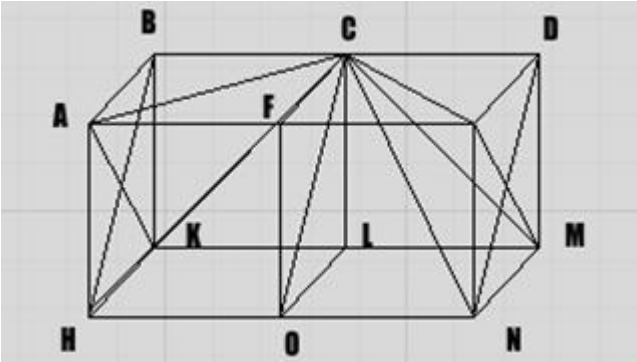

Рисунок 12.

Таким образом, необходимое ребро CO, которое не ортогональной ни одной проекции, было добавлено в список ребер. Все остальные ребра, скорее всего, не принадлежат всем проекциям и будут удалены на следующем шаге алгоритма. Такие операции производятся для каждой вершины модели, а также для каждой проекции.

2.6. Удаление лишних ребер

На данном этапе выполнения алгоритма в псевдокаркасной модели существует большое количество ребер, не принадлежащих проекциям. Для того чтобы их удалить, необходимо спроецировать вершины $\mathbf{V}(O)$ на каждую из плоскостей XY, XZ и YZ и сравнить ребра $\mathbf{E}(O)$ с $\mathbf{FE}(O)$, $\mathbf{TE}(O)$ и $\mathbf{SE}(O)$ соответственно. Если спроецированное ребро модели совпадает с одним из ребер соответствующей проекции, но не выходит за его пределы, оно остается в списке, в противном случае – удаляется.

В дополнение к алгоритму построения псевдокаркасной модели объекта по трем 2D проекциям, предложенному Markowsky и Wesley [12], в работе предлагается алгоритм обработки пунктирных линий, который можно использовать для несимметричных относительно центра проекций.

Изначально предполагается что объект, чья псевдокаркасная модель создается в ходе алгоритма, симметричен относительно центра, поскольку

известны только три его проекции. На предыдущем шаге могли быть созданы ребра, которые принадлежат проекциям, но в итоговую каркасную модель не входят.

Рассмотрим возможность восстановления каркасных моделей несимметричных объектов. Предположим, что фигура симметрична только относительно двух плоскостей XY и YZ. Для каждого пунктирного ребра проекции создается список ребер из $\mathbf{E}(O)$, которые совпадают с пунктирным ребром. Если удалять ребра на ближайшей и дальней гранях каждой плоскости, то может возникнуть ситуация, при которой на дальней грани удалятся нужные ребра. Поэтому предлагается для фронтальной проекции из получившихся списков удалять ребро с наименьшей координатой z. Для боковой проекции удалять ребро с наибольшей координатой x. Для верхней проекции удалять ребро с наибольшей координатой y.

Таким образом, из трех двумерных проекций в результате работы алгоритма получается псевдокаркасная модель с допущением, что на дальних гранях объекта могут сформироваться ребра, принадлежащие проекциям, но, возможно, не принадлежащие объекту. Пользователю предоставляется возможность интерактивного удаления лишних ребер.

3. Разработка программы

3.1. Формат DXF

DXF™ – открытый формат файлов для обмена данными между приложениями систем автоматизированного проектирования (САПР). Был создан для Autocad фирмой Autodesk и поддерживается практически всеми системами САПР на платформе PC. Наряду с форматом DWG является одним из самых распространенных форматов файлов для обмена графической информацией, однако DWG является закрытым форматом, тогда как для DXF Autodesk предоставляется бесплатную спецификацию [4]. DXF формат содержит все информацию об изображении в виде маркированных данных. Это означает что различные данные обозначаются целыми числами.

Формат DXF состоит из семи следующих разделов:

- HEADER – содержит основную информацию о чертеже. К примеру, здесь может храниться название программы, в которой был создал файл, версия программы, максимальные и минимальные координаты в чертеже.
- CLASSES – содержит информацию об объявленных в приложении экземплярах классов, которые используются в разделах BLOCKS, ENTITIES и OBJECTS.
- TABLES – хранит массивы данных, такие как таблица слоев, таблица стилей, таблица типов линий.
- BLOCKS – хранит данные о примитивах[2], объединенных в блоки. Блок имеет уникальный идентификатор и название и может использоваться в разделе ENTITIES.
- ENTITIES – хранит данные о примитивах.
- OBJECTS – содержит данные о не графических объектах, которые используются в приложениях, написанных на AutoLISP и ObjectARX.
- THUMBNAILIMAGE – хранит изображение того, что находится на чертеже.

[2] Геометрческие фигуры такие как точка, линия, окружность, дуга и т.д.

3.2. Формат входных данных

Входные данные представлены в виде файлов трех проекций: вид спереди, вид сверху, вид сбоку, - в формате DXF. Программе необходимо знать, какой проекцией является каждый из файлов, поэтому проекции должны иметь название “FV.dxf” или “fv.dxf”, “TV.dxf” или “tv.dxf”, “SV.dxf” или “sv.dxf” соответственно. Проекции должны быть описаны линиями. Важно отметить, что линии, находящиеся за пределами грани проекции должны быть обязательно описаны при помощи линии типа пунктир (рис. 13).

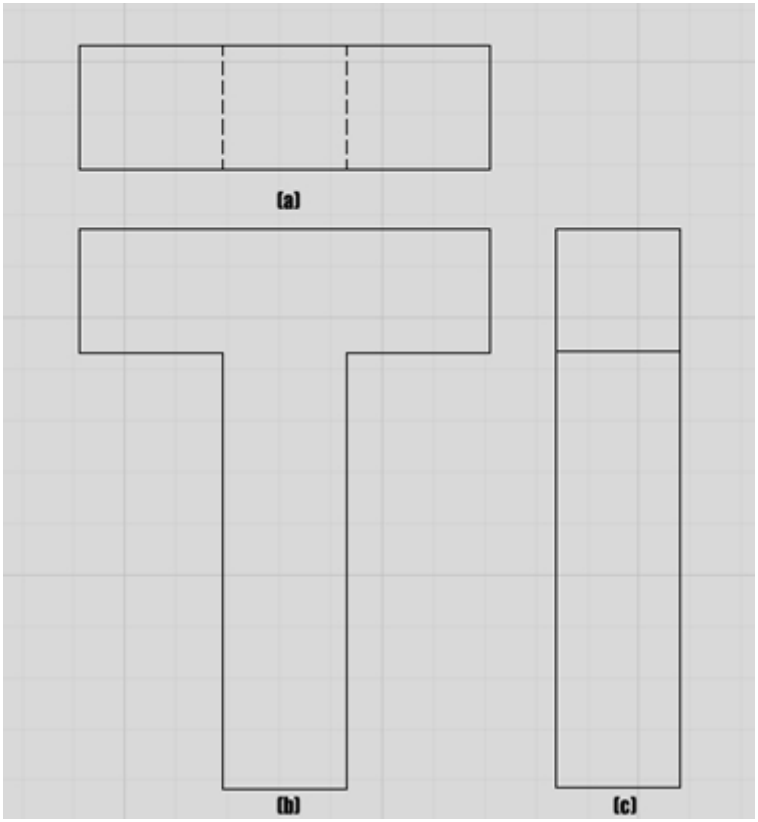

Рисунок 13. Проекции буквы «T». (a) –вид сверху, (b) –вид спереди, (c) –вид сбоку.

Программа распознает пунктирные линии, которые имеют подстроку «DASHED» в своем названии. Все остальные типы линий распознаются как обычные.

3.3. Выходные данные

В результате работы программы формируется каркасная 3D модель объекта, описанного тремя проекциями, полученными на входе в программу (рис. 14).

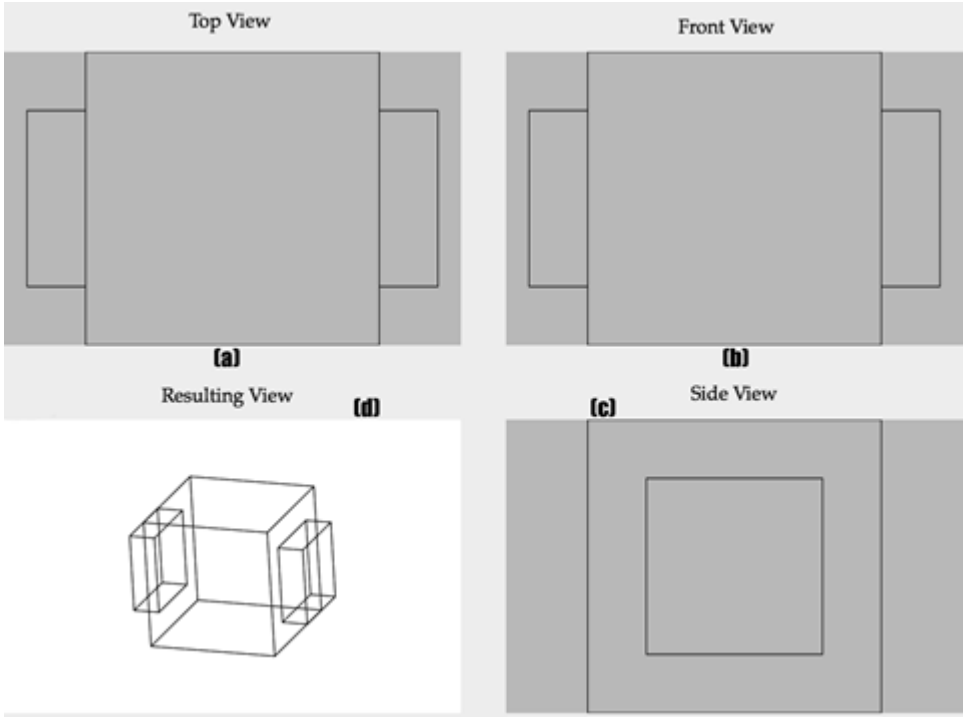

Рисунок 14. Результат работы программы. (a) – вид сверху, (b) – вид спереди, (c) – вид сбоку, (d) –получившаяся псевдокаркасная модель.

Так как итогом работы описанного алгоритма является псевдокаркасная модель, пользователь может сам выбрать и удалить лишние по его мнению ребра. В связи с этим реализованы функции “Отменить” и “Повторить” при помощи класса “NSUndoManager”.

3.4. Выбор средств разработки

В работе предлагается разработать приложение, работающее под операционной системой OS X. Ввиду этого в качестве языка разработки выбран объектно-ориентированный язык Objective C и среда разработки Xcode. Для работы с файлами формата DXF выбран фреймворк DXFReader [11]. При

анализе файла он возвращает массив примитивов, содержащихся в файле, и их данных. Для вывода проекций и каркасной модели на экран используются методы фреймворка “QuartzCore”.

3.5. Диаграмма классов

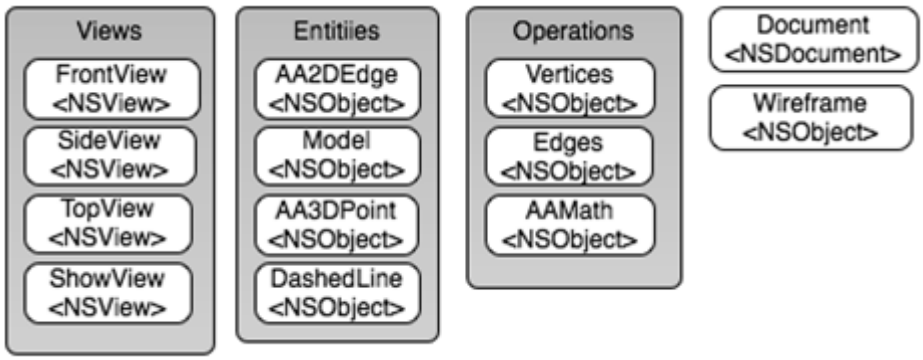

Рисунок 15. Диаграмма классов разрабатываемого приложения восстановления каркасных 3D объектов по 2D проекциям

Основным классом программы является “Document”, в нем создается объект класса “Wireframe” и вызывается метод “wire”, в котором происходит создание каркасной модели. Кроме того, в “Document” происходит обработка работы пользовательского интерфейса. Все данные о проекциях и каркасной модели хранятся в экземпляре класса “Model”. Классы группы “Views” являются классами, обрабатывающими вывод на экран проекций и самой каркасной модели. “View1” выводит фронтальную проекцию, “View2” выводит верхнюю проекцию, “View3” выводит боковую проекцию, “ShowView” обрабатывает выводит на экран каркасной модели при помощи методов, которые будут описаны в следующем разделе. Класса группы “Operations” содержат классовые методы для работы с вершинами (класс “Vertices”) и ребрами (класс “Edges”), а также некоторые математические операции (класс “AAMath”).

3.6. Вывод на экран каркасной модели

При описании фигуры для вывода ее на экран используется матрица вершин **A** размером [n, 4]**.**

$$\mathbf{A} = \begin{bmatrix} x_1 & y_1 & z_1 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ x_n & y_n & z_n & 1 \end{bmatrix},$$

где x_i, y_i, z_i – координаты x, y, z i-ой вершины,

n – количество вершин.

Четвертый элемент строки является однородной координатой k, и для однозначности принимается k=1. В ходе преобразований этот элемент изменяется и корректируется. Таким образом в однородных координатах каждая вершина x_i, y_i, z_i представляется в виде

$$(k_{xi}, k_{yi}, k_{zi}, k_i),$$

где $k_{xi} = x_i * k_i,$

$k_{yi} = y_i * k_i,$

$k_{zi} = z_i * k_i.$

Для того чтобы в ходе преобразований получить реальные координаты, необходимо делить каждую строку матрицы **A** на четвертый элемент этой строки.

В программе реализованы такие преобразования трехмерной машинной графики как

- Повороты вокруг осей координат;
- Масштабирование по осям;
- Сдвиг вдоль осей координат.

Для выполнения соответствующего преобразования необходимо умножить матрицу **A** матрицу соответствующего преобразования.

Поворот вокруг оси Ox на угол υ представлен матрицей $\mathbf{R}_x$:

$$\mathbf{R}_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\upsilon) & \sin(\upsilon) & 0 \\ 0 & -\sin(\upsilon) & \cos(\upsilon) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Поворот вокруг оси Oy на угол υ представлен матрицей $\mathbf{R}_y$:

$$\mathbf{R}_y = \begin{bmatrix} \cos(\upsilon) & 0 & -\sin(\upsilon) & 0 \\ 0 & 1 & 0 & 0 \\ \sin(\upsilon) & 0 & \cos(\upsilon) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Масштабирование по осям представлено матрицей $\mathbf{R}_s$:

$$\mathbf{R}_s = \begin{bmatrix} \mathrm{d} & 0 & 0 & 0 \\ 0 & \mathrm{d} & 0 & 0 \\ 0 & 0 & \mathrm{d} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

где d – коэффициент масштабирования.

Сдвиг вдоль осей координат представлен матрицей $\mathbf{R}_m$:

$$\mathbf{R}_m = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \mathrm{dx} & \mathrm{dy} & \mathrm{dz} & 1 \end{bmatrix},$$

где dx – коэффициент сдвига по оси Ox,

dy – коэффициент сдвига по оси Oy,

dz – коэффициент сдвига по оси Oz.

Стоит отметить, что после процесса восстановления фигура находится полностью в первом квадранте плоскости XZ (рис. 16).

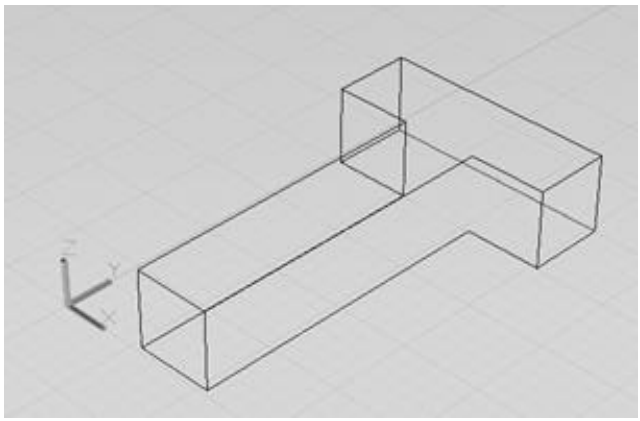

Рисунок 16.

Пример результата работы программы до вывода на экран

Для удобства работы с фигуры необходимо сдвинуть ее центр в начало координат, для этого нужно умножить матрицу **A** на матрицу $\mathbf{R}_{\boldsymbol{m}}$,

где dx =$(x_{max_{\mathbf{FV(O)}}} - x_{min_{\mathbf{FV(O)}}})/2$,

dy =$(y_{max_{\mathbf{FV(O)}}} - y_{min_{\mathbf{FV(O)}}})/2$,

dz =$(x_{max_{\mathbf{SV(O)}}} - x_{min_{\mathbf{SV(O)}}})/2$.

На экран выводят только координаты x и y матрицы **A**. Так как в NSView центр координат находится в левой нижней точке окна, при выводе на экран для нормального отображения фигуры к каждой координате x прибавляется переменнннная drawX,, а к каждой координате y прибавляется переменная drawY. Изначально drawX равен половине ширины окна “ShowView”, a drawY равен половине высоты окна “ShowView”. Эти значения изменяются при изменении положения центра координат.

Заключение

В работе были решены следующие задачи:

- Изучены существующие форматы представления инженерных чертежей;
- Изучены существующие алгоритмы восстановления трехмерной каркасной модели по трем двухмерным ортогональным проекциям;
- Предложен алгоритм удаления лишних ребер на описанных проекциями лицевых поверхностях, обозначенных на проекциях пунктирными линиями;
- Разработано приложение, принимающее на вход три 2D проекции (вид сверху, вид спереди, вид сбоку) формата DXF и восстанавливающее корректную каркасную 3D модель заданного объекта на выходе.

Результатом работы является программа восстановления каркасных 3D объектов из 2D проекций для операционной системы OS X. В программе реализовано автоматическое создание псевдокаркасной модели, необходимую каркасную модель пользователь может получить при помощи удаления ненужных по его мнению ребер в полученной псевдокаркасной модели.

В дальнейшем планируется изучить алгоритмы восстановления solid объектов из 2D проекций и добавить этот функционал в программу. Также возможно изучение процесса восстановления трехмерных объектов по набору фотографий.

Используемые источники

1. Мальцев А. И. — Основы линейной алгебры. — Изд. 3-е, перераб.// М.: «Наука». — 1970. — 400 с.

2. Aldefeld, B. — On automatic recognition of 3D structures from 2D representations // Computer Aided Design. — 1983. — № 15(2). — 59-72.

3. Ç ̈içek, A. & Gülesin, M. — Reconstruction of 3D models from 2D orthographic views using solid extrusion and revolution // Journal of Materials Processing technology. — 2004. — № 152, — 291-298.

4. DXF Format [DXF - DXF Reference]: [Электронный ресурс]. Режим доступа: http://www.autodesk.com/techpubs/autocad/acad2000/dxf/dxf_format.htm (Дата обращения 10.02.2014)

5. Furferi, R., Governi, L., Palai, M., & Volpe, Y. — 3D Model Retrieval from mechanical drawings analysis // International Journal of mechanics. — 2011. — № 5(2). — 91-99.

6. Governi, L. & Furferi, R. & Palai, M. & Volpe, Y. — 3D geometry reconstruction from orthographic views: A method based on 3D image processing and data flitting // Computers in industry. — 2013. — № 64. — 1290-1300.

7. Idesawa, M. — A system to generate a solid figure from three views // Bull JSME. — 1973. — № 16. — 216-225.

8. Lee, H. & Han, S. — Reconstruction of 3D interacting solids of revolution from 2D orthographic views // Computer-Aided Design. — 2005. — № 37. — 1388-1398.

9. Markowsky, G. & Wesley, M. Fleshing out wireframes // IBM Journal of Research and Development. — 1980. — № 24(5). — 582–597.

10. Shum, S. S. P., Lau, W. S., Yuen, M. M. F., & Yu, K. M. — Solid reconstruction from orthographic opaque views using incremental extrusion // Computer Graphics. — 1997. — № 26(6). — 787-800.

11. SourceForge.net: DXFReader - Project Web Hosting - Open Source Software: [Электронный ресурс]. Режим доступа: http://dxfreader.sourceforge.net (Дата обращения 10.01.2014)

12. Wesley, M.A. & Markowsky, G. (1981). — Fleshing out projections // IBM Journal of Research and Development. — 1981. — № 25(6). — 934–954.