Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Институт естественных и точных наук
Факультет математики, механики и компьютерных технологий
Кафедра математического и компьютерного моделирования
Направление подготовки Математика и компьютерные науки

РАБОТА ПРОВЕРЕНА
Рецензент, доцент кафедры УМФ,
к.ф.-м.н., доцент
_____/Д.Е. Шафранов
«____»_____ 2019 г.

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой,
д.ф.-м.н., доцент
_____/С.А. Загребина
«____»_____ 2019 г.

Восстановление трехмерных каркасных моделей по ортографическим проекциям

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ–02.03.01.2019.28. ВКР

Руководитель работы, доцент
кафедры, к.ф.-м.н.
_____/А.А. Акимова
«____»_____ 2019 г.

Автор работы,
студент группы ЕТ-411
_____/Д.С. Вишнев
«____»_____ 2019 г.

Нормоконтролер,
доцент кафедры, к.ф.-м.н.
_____/А.А. Акимова
«____»_____ 2019 г.

Челябинск 2019

АННОТАЦИЯ

Вишнев Д.С. Востановление трехмерных каркасных моделей по ортографическим проекциям. – Челябинск: ЮУрГУ, ЕТ-411, 35 с., 8 ил., библиогр. список – 34 наим., 1 прил.

Выпускная квалификационная работа выполнена с целью разработки программы по восстановлению трехмерных каркасных моделей по трем ортографическим проекциям.

В ходе работы был проведен анализ методов восстанавливающих трехмерные объекты, таких как метод отображения границ (B-rep oriented methods) и конструктивная сплошная геометрия (CSG).В работе рассматриваются ортографические проекции, которые являются частным случаем параллельных плоских геометрических проекций.

В результате работы была разработана программа позволяющая при внесении данных в три файла формата TXT,представляющих собой матрицы ортографических проекций, и получить восстановленную псевдокаркасную модель. Также пользователь может взаимодействовать с полученной фигурой,т.е. выполнять вращение.Листнинг программы приведен в приложении.

ОГЛАВЛЕНИЕ

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. В последние четыре десятилетия написан ряд работ о так называемой геометрической задаче восстановления, или просто восстановления (geometrical reconstruction or simply reconstruction problem):[1], [5], [12], [16].

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. В зависимости от основного подхода к решению задачи восстановления трехмерных объектов по проекциям, все алгоритмы можно разделить на две группы: метод отображения границ (B-rep oriented methods) [7], [2], [17], [27], [29] и конструктивная сплошная геометрия (CSG) [22], [24], [28], [18], [8]. Метод отображения границ можно рассматривать как метод движения «снизу вверх»: сначала восстанавливаются вершины, затем – ребра и грани, и только на последнем шаге – трехмерные составляющие (solids). В свою очередь, метод конструктивной сплошной геометрии можно рассматривать как метод движения «сверху вниз»: метод основан на работе с геометрическими примитивами, такими как куб или шар, к которым последовательно применяются операции булевой алгебра.

Наибольший прорыв в области восстановления 3D моделей совершили Wesley и Markowsky в своих работах [20] и [21], предложив использовать каркасную модель как промежуточный шаг процесса восстановления. На сегодняшний день эти авторы являются самыми известными исследователями в этой области, а их подход используется при разработке практически любого нового алгоритма восстановления 3D моделей. В работах [12], [13]

описаны схемы восстановления каркасных моделей как часть общей проблемы восстановления тел из нескольких плоских проекций. Предложенные схемы, однако, требуют значительного участия пользователя. Конечно, возникает желание составить алгоритм, не требующий участия пользователя в решении. Но такой алгоритм потребует больше входных данных и, при неполноте первоначальной информации, выдаст несколько решений.

В настоящей работе основное внимание уделено восстановлению псевдокаркасных моделей по ортографическим проекциям, выполненному на основе адаптированных алгоритмов восстановления 3D объектов по ортографическим проекциям [6] и [12]. Под псевдокаркасной моделью будем понимать совокупность всех каркасных трехмерных моделей, которые могут быть построены по данному набору трех ортографических проекций. Заметим, что, если набор трех ортографических проекций неоднозначно определяет псевдокаркасную модель, то результатом будет набор псевдокаркасных моделей. В статье Masanori Idesawa [12] были предложены способы, являющиеся своего рода классическими, но так как данная статья была выпущена в 1973 году, то пользоваться ей стоит только для понимания основных аспектов 3D моделирования. Статья Rocco Furferi [6], напротив, достаточно современна и выпущена она была в 2011 году, данный алгоритм прямолинеен и работать с ним проще.

Целью данной работы является разработка программы для восстановления трехмерных каркасных моделей по их ортографическим проекциям.

Для достижения данной цели необходимо решить следующие задачи:

1) выполнить сбор и анализ уже имеющихся методов;
2) разработать и реализовать алгоритм программы;
3) выполнить проверку на работоспособность программы;
4) продемонстрировать работоспособность программы на конкретных примерах.

В настоящее время рассматриваемая задача является актуальной. В современной жизни все чаще поднимается проблема преобразования бумажных архивов чертежно-конструкторской документации в электронные модели изделия (ЭМИ), как советскими и российскими учеными, так и зарубежными.

Большое число технических чертежей, выполненных в бумажном или электронном варианте, хранится в архивах крупных компаний. Кроме того, разработка большинства объектов начинается с построения технических чертежей, а не трехмерных моделей. Двумерные чертежи тяжелы для понимания и визуального восприятия.

Системы класса PDM, PLM, CAD, CAM, CAE изначально не предназначены для преобразования бумажных архивов чертежно-конструкторской документации. Они хранят, пусть даже связанные между собой, отдельные файлы, в которых находятся либо двумерные векторные чертежи, либо трехмерные электронные модели, либо отсканированные с бумаги документы. Все эти виды документов и хранимых моделей не предполагают двусторонних преобразований из одного вида в другой, в лучшем случае это могут быть проекционные изображения трехмерного объекта, полученные в результате его проецирования на картинную плоскость или поверхность.

В архивах предприятий, занимающихся разработкой различных технических изделий, накоплено большое количество, доходящее до десятков миллионов листов формата А0 технических чертежей, причем не все из них хранятся в электронном виде. На этих чертежах зачастую представлены детали и различные элементы, до сих пор используемые в процессе производства. Если следовать концепции информационной поддержки жизненного цикла продукции (CALS), обязывает конструкторов либо заново проектировать этот элемент в современных системах автоматизированного проектирования (САПР), либо же ссылаться на бумажную версию (что противоречит концепции CALS). Таким образом, задача восстановления трехмерных моделей по ортографическим проекциям остается актуальной и в настоящее время.

1. НЕКОТОРЫЕ СВЕДЕНИЯ О ПОДХОДАХ К ВОССТАНОВЛЕНИЮ КАРКАСНЫХ МОДЕЛЕЙ ПО ПРОЕКЦИЯМ

1.1. Ортографические проекции

Под проецированием на плоскость будем понимать отображение трехмерного пространства на двумерную плоскость проекции (картинную плоскость), построенное по следующему правилу. Из одной фиксированной точки (центра проекции), не лежащей в плоскости проекции, проводятся прямые (проекторы или проецирующие лучи) через каждую точку объекта. Тогда каждая точка трехмерного объекта отображается в некоторую двумерную точку проекции, образованную пересечением проектора и плоскости проекции. Результат такого отображения всех точек объекта называется плоской проекцией или просто проекцией.

Если расстояние от центра проекции до картинной плоскости конечно, то получается перспективная (центральная) проекция, а если бесконечно – параллельная проекция (все проекторы параллельны), см. рисунок 1.1. В технических чертежах применяются параллельные проекции. Художники и архитекторы используют перспективные проекции для изображения общих планов. Черчение с применением перспективных проекций сложных объектов довольно трудоемко, но использование вычислительной техники позволяет упростить эту процедуру. Перспективные проекции порождают эффект, называемый перспективным укорачиванием, который состоит в том, что размер перспективной проекции объекта обратно пропорционален расстоянию от центра проекции до объекта. Перспективная проекция выглядит более реалистичной, но непригодна для передачи точной формы и размера объекта.

В случае, если проекторы являются прямыми, проекция называется плоской геометрической.

В настоящей работе рассматриваются только ортографические проекции, которые являются частным случаем параллельных плоских геометрических проекций.

Под ортографической проекцией будем понимать результат работы следующего алгоритма.

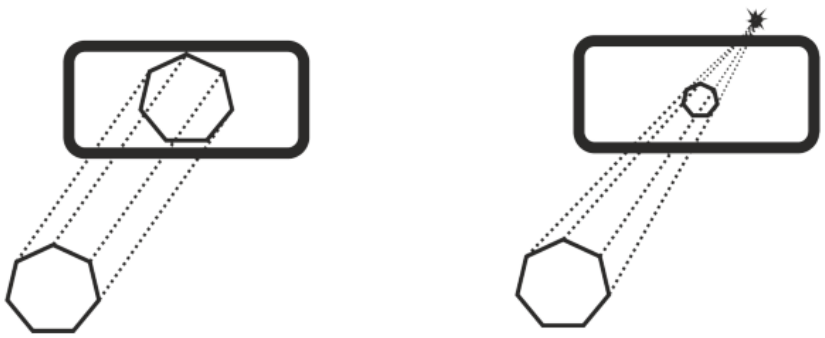

Рисунок 1.1 – Положение проекторов при построении параллельной (слева) и перспективной (справа) проекции

1. Выбрать картинную плоскость, параллельную или совпадающую с одной из координатных плоскостей.
2. Расположить проекторы параллельно координатной оси, ортогональной картинной плоскости.
3. Спроецировать объект на картинную плоскость.

В этом случае точно изображаются настоящие форма и размеры той плоской грани объекта, которая параллельна картинной плоскости, если такая грань есть.

Недостатком ортографических проекций является неясное (иногда – неоднозначное) представление формы объекта в целом. Тем не менее, ортографические проекции были и остаются востребованы, в частности, в инженерном проектировании как основа технических чертежей.

Главным фасадом будем называть вид спереди или сзади, боковым фасадом – вид слева или справа, планом – вид сверху или снизу. Такие определения оправданы для наиболее распространенной ситуации: объект располагается в первом октанте, а координатные оси относительно наблюдателя – так, как показано на рисунке 1.2.

При построении проекций рассматриваемыми в настоящей работе методами невидимые линии не отличаются от видимых, поэтому каждая пара (вид спереди и сзади, слева и справа, сверху и снизу) определяет одну и ту же проекцию.

Тройка ортографических проекций, включающая главный фасад, боковой фасад и план, используется наиболее часто, поскольку достаточно полно передает форму объекта, а для построения всех трех этих проекций

не требуется дополнительных преобразований. В самом деле, для любой из этих трех проекций центр проекции расположен в бесконечно удаленной точке координатной оси, ортогональной картинной плоскости, которая параллельна или совпадает с координатной плоскостью,пример показан на рисунке 1.2. Таким образом, для того, чтобы из матрицы тождественного преобразования построить матрицу главного фасада, бокового фасада и плана (ортографического проецирования на координатную плоскость OYZ, OXZ и OXY, соответственно), достаточно сделать нулевым столбец, соответствующий координате, информация о которой теряется при проецировании. Так, в матрице преобразования проецирования вдоль оси OX на плоскость OYZ нулевым является первый столбец, вдоль оси OY на OXZ – второй, а вдоль оси OZ на OXY – третий.

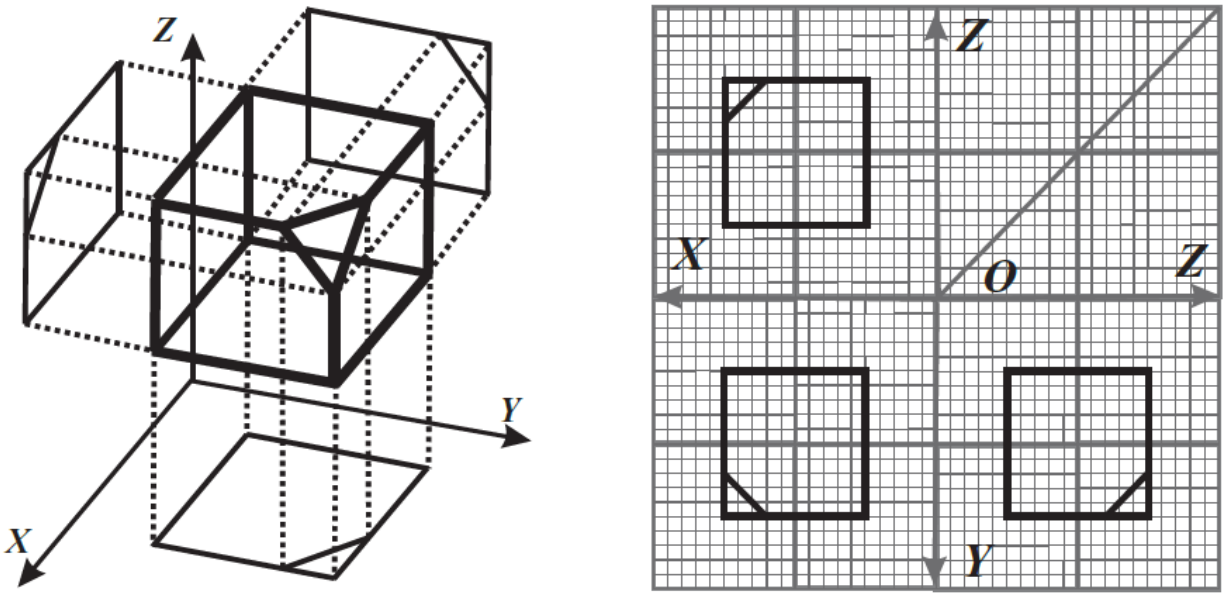

Рисунок 1.2 – Ортографические проекции: главный и боковой фасады, план

1.2. Выбор объекта

Под объектом мы будем понимать трехмерную геометрическую модель объекта реального мира, построенную из точек и отрезков в трехмерной системе координат. Таким образом, ограничимся рассмотрением объектов, представляющих собой каркасные (проволочные) модели, т.е. наборы опорных точек, соединенных в нужном порядке отрезками и однозначно определяющих положение объекта в пространстве.

Под псевдокаркасной моделью будем понимать совокупность всех каркасных трехмерных моделей, которые могут быть построены по данному набору трех ортографических проекций. Псевдокаркасная модель состоит из набора вершин и ребер, соединяющих эти вершины, и может содержать лишнюю, но в то же время более полную информацию об описываемом объекте. В отличие от каркасной модели, псевдокаркасная модель может включать в себя «лишние» ребра, которые принадлежат описывающим ее проекциям (рисунок 1.3). Одна псевдокаркасная модель может описывать большое количество 3D объектов (рисунок 1.4).

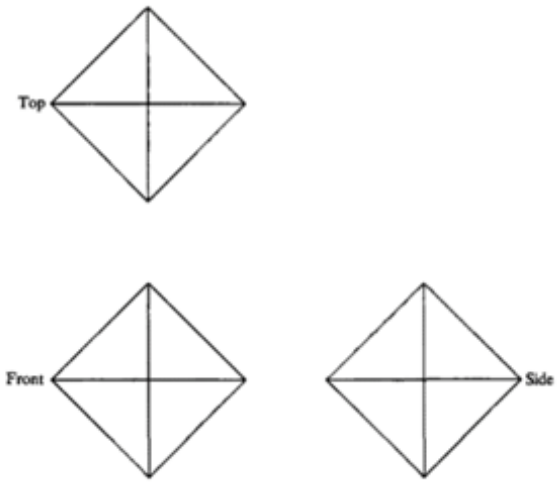

Рисунок 1.3 – Три проекции октаэдра [14]

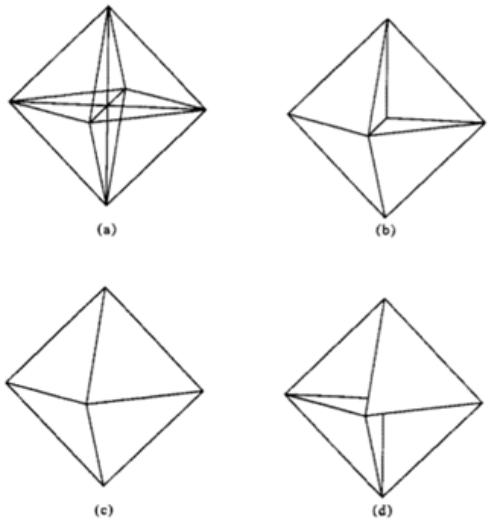

Рисунок 1.4 – (a) – псевдокаркасная модель октаэдра, (b, c, d) – возможные solid объекты, этой модели [14]

Объекты допускают матричное представление на компьютере в виде массивов. Преимуществом матричного представления является возможность компактного описания сложных преобразований в виде композиции (т.е., последовательного выполнения) простых, а также – быстрого программного и аппаратного выполнения преобразований как матричных операций с помощью имеющихся технических средств. Геометрические преобразования применяются для моделирования динамических процессов, включающих перемещение и модификацию объектов, а также для построения проекций трехмерных объектов.

В настоящей работе рассматриваются только полиэдральные объекты, т.е. реализуемый алгоритм ограничивается случаем узкого класса объектов, ребра которых – прямые линии, а все грани являются плоскими.

1.3. Выбор метода восстановления

Wesley, M.A. и Markowsky, G. предложили алгоритм восстановления многогранных (полиэдральных) тел по трем ортографическим проекциям. В их работе [20] предложен топологический алгоритм автоматического перехода от каркасной модели к объемному описанию объекта в терминах сплошных и пустых тел, топологии поверхностей и ребер. Поскольку алгоритм топологический, его можно обобщить на случаи более сложных объектов (не являющихся полиэдральными). Работы [20] и [21] наиболее популярны среди исследователей, интересующихся вопросом восстановления трехмерных объектов по проекциям, и до сих пор лежат в основе большинства алгоритмов, действующих по методу отображения границ. Классический метод состоит в последовательном нахождении трехмерных вершин, ребер и граней. К классическому методу мы относим наиболее разработанные разными авторами подходы, укладывающиеся в общую схему, намеченную полвека назад Idesawa [12], Markowsky и Wesley [20]. За полвека предлагались некие предложения по конкретной реализации (алгоритмизация, матричное представление, расширение класса объектов), но математический костяк, заложенный классиками, существенно не менялся. Преимуществом классического метода является относительная разработанность и строгое математическое доказательство корректности, проведенное основа-

телями метода.

В настоящей работе основное внимание уделено восстановлению псевдокаркасных моделей по ортографическим проекциям, выполненному на основе адаптированных алгоритмов восстановления 3D объектов по ортографическим проекциям [6] и [12]. В статье Masanori Idesawa [12] были предложены способы, являющиеся своего рода классическими, но так как данная статья была выпущена в 1973 году, то пользоваться ей стоит только для понимания основных аспектов 3D моделирования. Статья Rocco Furferi [6], напротив, достаточно современна и выпущена она была в 2011 году, данный алгоритм прямолинеен и работать с ним проще.

1.4. Выводы по первой главе

Под объектом понимается трехмерная геометрическая модель объекта реального мира, построенную из точек и отрезков в трехмерной системе координат.

Псевдокаркасная модель – совокупность всех каркасных трехмерных моделей, которые могут быть построены по данному набору трех ортографических проекций.

Работы Markowshy и Westly [20, 22] наиболее популярны среди исследователей, интересующихся вопросом восстановления трехмерных объектов по проекциям, и до сих пор лежат в основе большинства алгоритмов, действующих по методу отображения границ.

Преимуществом классического метода является относительная разработанность и строгое математическое доказательство корректности, проведенное основателями метода.

2. МАТЕМАТИЧЕСКОЕ ОПИСАНИЕ ИСПОЛЬЗУЕМОГО АЛГОРИТМА

2.1. Восстановление по трем ортографическим проекциям

В качестве трехмерного объекта возьмем самый простой случай: трехмерную каркасную модель, т.е. набор вершин (опорных точек), соединенных ребрами (отрезками). Рассмотрим решение задачи восстановления первоначальных координат каждой точки трехмерной каркасной модели по трем ортографическим проекциям (главный и боковой фасады, план). Решение состоит из следующих этапов, изложенных в пунктах 2.1.1 – 2.1.3.

1. Представить набор имеющихся трех ортографических проекций в виде матрицы.
2. Привести матричное представление набора ортографических проекций к стандартному.
3. Составить матрицы вершин и ребер трехмерной псевдокаркасной модели.

Под псевдокаркасной моделью будем понимать совокупность всех каркасных трехмерных моделей, которые могут быть построены по данному набору трех ортографических проекций. Заметим, что, если набор трех ортографических проекций неоднозначно определяет псевдокаркасную модель, то результатом будет набор псевдокаркасных моделей.

2.1.1. Матричное представление набора ортографических проекций

Каждую из ортографических проекций можно рассматривать как граф, вложенный в соответствующую координатную плоскость. Графом называется упорядоченная пара $(V; E)$, где V – непустое множество вершин, а E – множество пар вершин, называемых ребрами. В разделе 2.1 под ребром ортографической проекции мы также будем понимать отрезок, соединяющий пару вершин, называемых концами ребра.

Два ребра называются смежными, если они имеют общую вершину. Если ребро e образовано парой вершин v_1 и v_2 (обозначается $e = v_1v_2$), то вершина v_j и ребро e называются *инцидентными*, $j = \overline{1, 2}$.

Обозначим главный фасад, боковой фасад и план, соответственно, $\mathbf{\Pi^i}$,

$i = \overline{1,3}$.Для каждой ортографической проекции $\mathbf{\Pi^i}$, $i = \overline{1,3}$, определим по два набора элементов следующим образом.

1. $\mathbf{V^i}$= $\{v_1^i,\ v_2^i, ..., v_{n_v^i}^i\}$ – упорядоченный набор вершин (опорных точек) ортографической проекции $\mathbf{\Pi^i}$, где v_j^i – j-я вершина ортографической проекции $\mathbf{\Pi^i}$, n_v^i – число вершин ортографической проекции $\mathbf{\Pi^i}$, $j = \overline{1, n_v^i}$, $i = \overline{1,3}$. Заметим, что, по построению, каждая ортографическая проекция лежит в одной из координатных плоскостей. Следовательно, в любой ортографической проекции $\mathbf{\Pi^i}$ одна из трех пространственных координат каждой вершины

$$v_j^i = (x_j^i; y_j^i; z_j^i),\ j = \overline{1, n_v^i},\ i = \overline{1,3},$$

всегда нулевая. А именно,

$$v_j^1 = (x_j^1; 0; z_j^1),\ j = \overline{1, n_v^1},$$

$$v_j^2 = (0; y_j^2; z_j^2),\ j = \overline{1, n_v^2},$$

$$v_j^3 = (x_j^3; y_j^3; 0),\ j = \overline{1, n_v^3}.$$

2. $\mathbf{E^i}$= $\{e_1^i,\ e_2^i, ..., e_{n_e^i}^i\}$ – упорядоченный набор ребер ортографической проекции $\mathbf{\Pi^i}$, где e_k^i – k-е ребро ортографической проекции $\mathbf{\Pi^i}$, n_e^i – число ребер ортографической проекции $\mathbf{\Pi^i}$, $k = \overline{1, n_e^i}$, $i = \overline{1,3}$. Если ребро e_k^i соединяет вершины $v_{j_1}^i = (x_{j_1}^i; y_{j_1}^i; z_{j_1}^i)$ и $v_{j_2}^i = (x_{j_2}^i; y_{j_2}^i; z_{j_2}^i)$, где $j_1 < j_2$ и $j_1, j_2 \in \{1, 2, ..., n_v^i\}$, $k \in \{1, 2, ..., n_e^i\}$, $i \in \{1, 2, 3\}$, то ребро e_k^i может быть задано матрицей размера 2×3, имеющей вид

$$e_k^i = \begin{pmatrix} v_{j_1}^i \\ v_{j_2}^i \end{pmatrix} = \begin{pmatrix} x_{j_1}^i & y_{j_1}^i & z_{j_1}^i \\ x_{j_2}^i & y_{j_2}^i & z_{j_2}^i \end{pmatrix}.$$

Поскольку любая вершина ортографической проекции инцидентна по крайней мере одному ребру, каждая ортографическая проекция $\mathbf{\Pi^i}$, $i = \overline{1,3}$, полностью определяется набором своих ребер $\mathbf{E^i}$, т.е. может быть задана матрицей размера $2 \cdot n_e^i \times 3$ (называемой *матричным представлением ортографической проекции* $\mathbf{\Pi^i}$, $i = \overline{1,3}$), имеющей вид

$$\Pi^i = \begin{pmatrix} e^i_1 \\ e^i_2 \\ \dots \\ e^i_{n^i_e} \end{pmatrix},$$

а набор всех трех ортографических проекций – матрицей размера $2 \cdot (n^1_e + n^2_e + n^3_e) \times 3$ (называемой матричным представлением набора ортографических проекций $\mathbf{\Pi^i}$, $i = \overline{1,3}$), имеющей вид

$$OBJ = \begin{pmatrix} \Pi^1 \\ \Pi^2 \\ \Pi^3 \end{pmatrix}.$$

2.1.2. Приведение матричного представления набора ортографической проекции к стандартному

Заметим, что одна и та же ортографическая проекция может иметь несколько представлений различными наборами ребер. Например, на рисунке 2.1 показаны все четыре способа представить отрезок, проходящий через четыре вершины $v^3_j, j = \overline{1,4}$:

(A) ребро $v^3_1 v^3_4$,

(B) совокупность ребер $v^3_1 v^3_2$ и $v^3_2 v^3_4$,

(C) совокупность ребер $v^3_1 v^3_3$ и $v^3_3 v^3_4$,

(D) совокупность ребер $v^3_1 v^3_2$, $v^3_2 v^3_3$ и $v^3_3 v^3_4$.

Последующие шаги рассматриваемого в Разделе 2.1 метода разработаны только для одного из возможных матричных представлений набора ортографических проекций, называемого стандартным. Определим понятие стандартного матричного представления набора ортографических проекций и рассмотрим алгоритм сведения любого матричного представления набора ортографических проекций к стандартному.

Построение стандартного матричного представления набора ортографических проекций

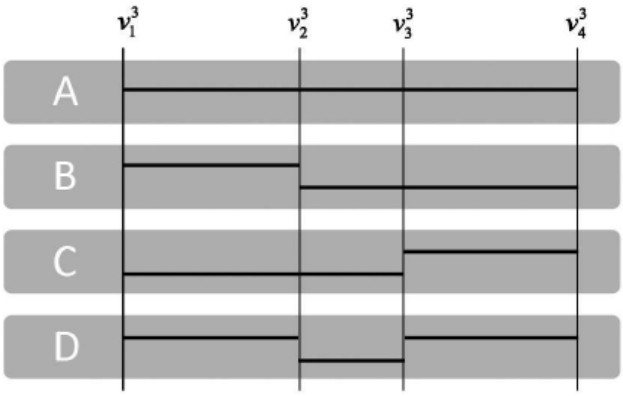

Рисунок 2.1 – Все четыре способа представить отрезок, проходящий через четыре вершины v_j^3, $j = \overline{1,4}$: (A) ребро $v_1^3v_4^3$, (B) совокупность ребер $v_1^3v_2^3$ и $v_2^3v_4^3$, (C) совокупность ребер $v_1^3v_3^3$ и $v_3^3v_4^3$, (D) совокупность ребер $v_1^3v_2^3$, $v_2^3v_3^3$ и $v_3^3v_4^3$.[6]

Матричное представление ортографической проекции

$$\Pi^i = \begin{pmatrix} e_1^i \\ e_2^i \\ \dots \\ e_{n_e^i}^i \end{pmatrix}, \text{где } i \in \{1,2,3\},$$

называется стандартным, при выполнении следующего условия.

Если ребро $e_k^i \in \mathbf{E^i}$ содержит внутренние вершины, то представление ортографической проекции Π^i содержит все ребра, из которых составлено ребро e_k^i, $k \in \{1,2,...,n_e^i\}$, $i \in \{1,2,3\}$. Вершина $v_j^i \in \mathbf{V^i}$ называется внутренней вершиной ребра $e_k^i \in \mathbf{E^i}$, если v_j^i лежит на ребре e_k^i, но не совпадает ни с одним из концов ребра e_k^i, где $i \in \{1,2,3\}$.

Таким образом, стандартное матричное представление ортографической проекции Π^i содержит все ребра любого другого матричного представления этой ортографической проекции Π^i, $i \in \{1,2,3\}$.

Матричное представление набора ортографических проекций $\mathbf{\Pi^i}$, $i = \overline{1,3}$,

$$OBJ = \begin{pmatrix} \Pi^1 \\ \Pi^2 \\ \Pi^3 \end{pmatrix},$$

называется стандартным , если матричное представление каждой ортографической проекции $\mathbf{\Pi^i}$, $i = \overline{1,3}$, стандартно. Таким образом, для получения

стандартного матричного представления OBJ достаточно матричное представление каждой ортографической проекции $\mathbf{\Pi^i}$, $i = \overline{1,3}$, привести к стандартному.

Алгоритм приведения матричного представления ортографической проекции к стандартному

Алгоритм заключается в последовательном дополнении текущего матричного представления каждой ортографической проекции недостающими ребрами из которых могут быть составлены имеющиеся ребра.

Разбиение ребер внутренними вершинами. В матричном представлении каждой ортографической проекции $\mathbf{\Pi^i}$ нужно проверить каждое ребро e^i_k, $k = \overline{1, n^i_e}$, $i = \overline{1,3}$, на наличие внутренних вершин. Если у ребра $e^i_k \in \mathbf{E^i}$, $i \in \{1,2,3\}$, такая вершина есть, нужно добавить (если отсутствует) в матричное представление ортографической проекции $\mathbf{\Pi^i}$ пару ребер, на которую ребро e^i_k разбивается этой внутренней вершиной. Заметим, что каждое из добавленных ребер нужно также проверить на наличие внутренних вершин.

На рисунке 2.1 показан пример: в результате проверки ребра $v^3_2 v^3_4$, см. (B), обнаружена промежуточная вершина v^3_3, поэтому ребро разбито на пару ребер $v^3_2 v^3_3$ и $v^3_3 v^3_4$, см. (D).

Следующий алгоритм реализует проверку на наличие внутренних вершин и разбиение на пару ребер каждой внутренней вершиной k-го ребра ортографической проекции $\mathbf{\Pi^3}$, соединяющего вершины $v^3_{j_1} = (x^3_{j_1}; y^3_{j_1}; 0)$ и $v^3_{j_2} = (x^3_{j_2}; y^3_{j_2}; 0)$, т.е. ребра, заданного матрицей

$$e^3_k = \begin{pmatrix} v^3_{j_1} \\ v^3_{j_2} \end{pmatrix} = \begin{pmatrix} x^3_{j_1} & y^3_{j_1} & 0 \\ x^3_{j_2} & y^3_{j_2} & 0 \end{pmatrix},$$

где $j_1, j_2 \in \{1, 2, ..., n^3_v\}$, $j_1 < j_2$, $k = \overline{1, n^3_e}$.

Таким образом, алгоритм составлен для обработки ребер ортографической проекции $\mathbf{\Pi^3}$. Для обработки ребер ортографической проекции $\mathbf{\Pi^1}$ или $\mathbf{\Pi^2}$ нужно, соответственно, рассмотреть координату z вместо координаты y или x, а также – сменить верхний индекс элементов с 3 на 1 или 2.

Алгоритм разбиения ребра e^3_k внутренними вершинами на пары ребер

1. Определить уравнение прямой $f^3_k(x, y) = 0$, содержащей ребро e^3_k.

В нашем случае, ребро e_k^3 – отрезок, соединяющий вершины $v_{j_1}^3 = (x_{j_1}^3; y_{j_1}^3; 0)$ и $v_{j_2}^3 = (x_{j_2}^3; y_{j_2}^3; 0)$, поэтому искомое уравнение – уравнение прямой, проходящей через две заданные точки:

$$\frac{x - x_{j_1}^3}{x_{j_2}^3 - x_{j_1}^3} - \frac{y - y_{j_1}^3}{y_{j_2}^3 - y_{j_1}^3} = 0.$$

2. Составить граничную матрицу ребра e_k^3

$$B_k^3 = \begin{pmatrix} B_{k_{11}}^3 \\ B_{k_{21}}^3 \end{pmatrix},$$

где

$$B_{k_{11}}^3 = max\{x_{j_1}^3; x_{j_2}^3\},\ B_{k_{21}}^3 = min\{x_{j_1}^3; x_{j_2}^3\}.$$

3. Для каждой вершины $v_{j_3}^3 = (x_{j_3}^3; y_{j_3}^3; 0) \in \mathbf{V^3}$, где $j_3 \neq j_1$ и $j_3 \neq j_2$, сделать следующее.

   3.1. Проверить, является ли $v_{j_3}^3$ внутренней вершиной ребра e_k^3, следующим образом. Определить значения параметров $l_p \in \{0, 1\}$, где $p = \overline{1, 3}$, по правилам, приведенным в Таблице 2.1.

Таблица 2.1 – Правила определения параметров $l_p \in \{0, 1\}$, где $p = \overline{1, 2}$

Проверяемое условие	l_p	l_p=1, если	l_p=0, если
Точка $(x_{j_3}^3; y_{j_3}^3; 0)$ лежит на прямой, содержащей ребро e_k^3	l_1	$f_k^3(x_{j_3}^3, y_{j_3}^3) = 0$	$f_k^3(x_{j_3}^3, y_{j_3}^3) \neq 0$
Вдоль оси OX точка $(x_{j_3}^3; y_{j_3}^3; 0)$ лежит между вершинами, инцидентными ребру e_k^3	l_2	$x_{j_3}^3 \in [B_{k_{11}}^3; B_{k_{21}}^3]$	$x_{j_3}^3 \notin [B_{k_{11}}^3; B_{k_{21}}^3]$

   Если $l_1 = l_2 = 1$, то вершина $v_{j_3}^3$ является внутренней вершиной ребра e_k^3, переходим к п. 3.2, в противном случае – проверяем следующую вершину.

   3.2. Если вершина $v_{j_3}^3$ является внутренней вершиной ребра e_k^3, то стандартное матричное представление ортографической проекции $\mathbf{\Pi^3}$ должно содержать пару ребер (обозначим их $e_{k_1}^3$ и $e_{k_2}^3$), на которые внутренняя вершина $v_{j_3}^3$ разбивает ребро e_k^3.

$$\begin{cases} e^3_{k_1} = \begin{pmatrix} v^3_{j_1} \\ v^3_{j_3} \end{pmatrix} \text{ и } e^3_{k_2} = \begin{pmatrix} v^3_{j_2} \\ v^3_{j_3} \end{pmatrix}, & \text{если } j_1 < j_2 < j_3, \\ e^3_{k_1} = \begin{pmatrix} v^3_{j_1} \\ v^3_{j_3} \end{pmatrix} \text{ и } e^3_{k_2} = \begin{pmatrix} v^3_{j_3} \\ v^3_{j_2} \end{pmatrix}, & \text{если } j_1 < j_3 < j_2, \\ e^3_{k_1} = \begin{pmatrix} v^3_{j_3} \\ v^3_{j_1} \end{pmatrix} \text{ и } e^3_{k_2} = \begin{pmatrix} v^3_{j_3} \\ v^3_{j_2} \end{pmatrix}, & \text{если } j_3 < j_1 < j_2. \end{cases}$$

Если какое-то из ребер $e^3_{k_1}$ и $e^3_{k_2}$ отсутствует в матричном представлении ортографической проекции $\mathbf{\Pi^3}$, то его нужно добавить.

2.1.3. Построение псевдокаркасной модели

Построение матричного представления псевдокаркасной модели проведем в два этапа:

1) построение матрицы трехмерных вершин,

2) построение матрицы трехмерных ребер.

Матрицы трехмерных ребер достаточно, чтобы построить трехмерную псевдокаркасную модель.

Построение матрицы трехмерных вершин псевдокаркасной модели

Пусть

n_v^{3D} – число трехмерных вершин псевдокаркасной модели,

V^{3D} – матрица трехмерных вершин псевдокаркасной модели, имеющая размер $n_v^{3D} \times 6$ и строки вида

$$(x_i^1 \ y_j^2 \ z_i^1 \ i \ j \ k),$$

где i,j и k -номера тех вершин ортографических проекций Π^1, Π^2 и Π^3, соответственно, которые являются проекциями трехмерной вершины v^{3D} с координатами $(x_i^1; y_j^2; z_i^1)$,

$\vec{j}$ и $\vec{k}$ - два вспомогательных вектора.

Зададим начальные значения:$V^{3D} = \vec{j} = \vec{k} = (\varnothing)$ и n_v^{3D}=0. Для каждой вершины $v_i^1 = (x_i^1; 0; z_i^1), i = \overline{1, n_v^1}$, ортографической проекции Π^1 восстановить все трехмерные вершины псевдокаркасной модели, для которых

вершина v_i^1 является ортографической проекцией на плоскость OXZ, следующим образом.

1. Записать в вектор $\vec{j}$ номера всех тех вершин ортографической проекции Π^2, имеющих такую же координату z, т.е.тех вершин $v_j^2 = (0; y_j^2; z_j^2), j \in 1, 2, ..., n_v^2$, для которых $z_j^2 = z_i^1$.
2. Записать в вектор $\vec{k}$ номера вершин ортографической проекции Π^3, имеющих такую же координату x, т.е. тех вершин $v_k^3 = (x_k^3; y_k^3; 0), k \in 1, 2, ..., n_v^3$, для которых $x_k^3 = x_i^1$.
3. Для каждого номера j, записанного в вектор $\vec{j}$, и каждого номера k записанного в вектор $\vec{k}$, сравнить координаты y вершины $v_j^2 = (0; y_j^2; z_j^2)$ ортографической проекции ^{2} и вершины $v_k^3 = (x_k^3; y_k^3; 0)$ ортографической проекции ^{3}.Если для некоторый пары номеров j и k окажется , что $y_j^2 = y_k^3$,то в матрицу положения трехмерных вершин псевдокаркасной модели V^{3D} записать строку вида $(x_i^1 y_j^2 z_i^1 ijk)$, а число n_v^{3D} трехмерных вершин псевдокаркасной модели увеличить на 1.
4. Положить $\vec{j} = \vec{k} = (\varnothing)$

Построение матрицы трехмерных ребер псевдокаркасной модели Пусть

n_e^{3D} – число трехмерных ребер псевдокаркасной модели,

E^{3D} – матрица трехмерных ребер псевдокаркасной модели, имеющая размер $n_e^{3D} \times 2$ и строки вида

$$(sp),$$

где $s < p$ и $s, p \in 1, 2, ..., n_v^{3D}$ - номера трехмерных вершин v_s^{3D} и $v^3 D_p$ псевдокаркасной модели, инцидентных ребру, задаваемому этой строкой (заметим, что мы используем одно и то же обозначение вида v_s^{3D} и для указания на s-ю трехмерную псевдокаркасной модели, где содержится информация об этой вершине),

D – вспомогательная матрица, имеющая размер $n_d^{3D} \times 7$ и строки вида

$$(sv_s^{3D}) = (sx_{is}^1 y_{js}^2 z_{is}^1 i_s j_s k_s),$$

где s-номер строки $v_s^{3D} = (x_{is}^1 y_{js}^2 z_{is}^1 i_s j_s k_s)$ матрицы V^{3D} трехмерных вершин псевдокаркасной модели, $s, n_d^{3D} \in 1, 2, ..., n_v^{3D}$,

$\vec{T}$-вспомогательный вектор.

Зададим начальные значения:$E^{3D} = D = \vec{T} = (\varnothing)$ и $n_v^{3D} = 0$.

Построение матрицы трехмерных ребер псевдокаркасной модели проведем в два этапа.

1. Построение матрицы трехмерных ребер, не ортогональных ни одной из ортографических проекций (такие ребра на каждой ортографической проекции задаются ребрами).
2. Построение матрицы трехмерных ребер, ортогональных отографической проекции
   2.1. $\mathbf{\Pi^1}$,
   2.2. $\mathbf{\Pi^2}$,
   2.3. $\mathbf{\Pi^3}$,

т.е. ребер, которые задаются ребрами только на тех двух ортографических проекциях, которым они параллельны, а на третьей, которой они ортогональны, — сжимаются в точку.

ЭТАП 1. Для каждой строки $v_s^{3D} = (x^1_{i\,s}\ y^2_{j\,s}\ z^1_{i\,s}\ i_s\ j_s\ k_s)$, где $s = \overline{1, n_v^{3D}}$, матрицы V^{3D} трехмерных вершин псевдокаркасной модели добавить в матрицу E^{3D} трехмерных ребер псевдокаркасной модели все ребра, инцидентные вершине v_s^{3D} и не ортогональные ни одной из ортографических проекций, следующим образом.

1. Для каждого i-го ребра ортографической проекции $\mathbf{\Pi^1}$, инцидентного вершине, номер которой записан в 4-м элементе строки $v_s^{3D} = (x^1_{i\,s}\ y^2_{j\,s}\ z^1_{i\,s}\ i_s\ j_s\ k_s)$, т.е. ребра, имеющего вид

$$e_i^1 = \begin{pmatrix} v^1_{i_s} \\ v^1_{t_i} \end{pmatrix} \text{ или } e_i^1 = \begin{pmatrix} v^1_{t_i} \\ v^1_{i_s} \end{pmatrix},$$

   записать в вектор $\overrightarrow{T}$ номер t_i второй вершины, инцидентной ребру e_i^1.
2. Для каждого номера t_i, записанного в вектор $\overrightarrow{T}$, сделать следующее.
   2.1. Найти все строки $v_p^{3D} = (x^1_{i\,p}\ y^2_{j\,p}\ z^1_{i\,p}\ i_p\ j_p\ k_p)$, $p \in \{1, 2, ..., n_v^{3D}\}$ матрицы V^{3D} трехмерных вершин псевдокаркасной модели, для которых $t_i = i_p$, и записать каждую такую строку v_p^{3D} в вспомогательную матрицу D в виде строки $(p\ v_p^{3D}) = (p\ x^1_{i\,p}\ y^2_{j\,p}\ z^1_{i\,p}\ i_p\ j_p\ k_p)$.
   2.2. Для каждой строки $(p_d\ x^1_{i\,p_d}\ y^2_{j\,p_d}\ z^1_{i\,p_d}\ i_{p_d}\ j_{p_d}\ k_{p_d})$ с номером $d =$

$\overline{1, n_d^{3D}}$ вспомогательной матрицы D проверить одновременное наличие двух других ортографических проекций того же трехмерного ребра, для которого была найдена ортографическая проекция e_i^1:

$$\begin{cases} e_j^2 = \begin{pmatrix} v_{j_s}^2 \\ v_{j_{p_d}}^2 \end{pmatrix} \text{ и } e_k^3 = \begin{pmatrix} v_{k_s}^3 \\ v_{k_{p_d}}^3 \end{pmatrix}, \text{ если } s < p_d, \\ e_j^2 = \begin{pmatrix} v_{j_{p_d}}^2 \\ v_{j_s}^2 \end{pmatrix} \text{ и } e_k^3 = \begin{pmatrix} v_{k_{p_d}}^3 \\ v_{k_s}^3 \end{pmatrix}, \text{ если } p_d < s \end{cases}$$

т.е. одновременного наличия и ребра e_j^2 ортографической проекции $\mathbf{\Pi^2}$, инцидентного вершинам $v_{j_{p_d}}^2$ и $v_{j_s}^2$ ортографической проекции $\mathbf{\Pi^2}$, и ребра e_k^3 ортографической проекции $\mathbf{\Pi^3}$, инцидентного вершинам $v_{k_{p_d}}^3$ и $v_{k_s}^3$ ортографической проекции $\mathbf{\Pi^3}$.
Если это условие выполнено, то добавить в матрицу E^{3D} трехмерных ребер псевдокаркасной модели ребро

$$e^{3D} = \begin{cases} \begin{pmatrix} v_s^{3D} \\ v_{p_d}^{3D} \end{pmatrix}, \text{ если } s < p_d, \\ \begin{pmatrix} v_{p_d}^{3D} \\ v_s^{3D} \end{pmatrix}, \text{ если } p_d < s \end{cases}$$

в виде строки номеров трехмерных вершин псевдокаркасной модели, инцидентных ребру e^{3D}:

$$\begin{cases} (s\ p_d), \text{ если } s < p_d, \\ (p_d\ s), \text{ если } p_d < s, \end{cases}$$

а число n_e^{3D} трехмерных ребер псевдокаркасной модели увеличить на 1.

3. Положить D=$\overrightarrow{T}$=$(\varnothing)$.

ЭТАП 2.1. Для каждой строки $v_s^{3D} = (x_{i\,s}^1\ y_{j\,s}^2\ z_{i\,s}^1\ i_s\ j_s\ k_s)$, $s = \overline{1, n_v^{3D}}$, матрицы V^{3D} трехмерных вершин псевдокаркасной модели, добавить в матрицу E^{3D} трехмерных ребер псевдокаркасной модели все ребра, инцидентные вершине v_s^{3D} и ортогональные ортографической проекции $\mathbf{\Pi^1}$, следующим образом.

1. Найти все строки $v_t^{3D} = (x_{i\,t}^1\ y_{j_t}^2\ z_{i\,t}^1\ i_t\ j_t\ k_t)$, $t \in \{1, 2, ..., n_v^{3D}\}$ (в том числе, $t = s$), матрицы V^{3D} трехмерных вершин псевдокаркасной модели, для которых $i_t = i_s$, и записать каждую такую строку v_t^{3D} в вспомогательную матрицу D в виде строки $(t\ v_t^{3D}) = (t\ x_{i\,t}^1\ y_{j_t}^2\ z_{i\,t}^1\ i_t\ j_t\ k_t)$.
2. Записать в вектор $\overrightarrow{T}$ весь первый столбец вспомогательной матрицы D.
3. Для каждой пары $f \neq g$ элементов вектора $\overrightarrow{T}$ такой, что $j_f < j_g$ и $k_f < k_g$, проверить одновременное наличие двух ребер

$$e_j^2 = \begin{pmatrix} v_{j_f}^2 \\ v_{j_g}^2 \end{pmatrix} \text{ и } e_k^3 = \begin{pmatrix} v_{k_f}^3 \\ v_{k_g}^3 \end{pmatrix},$$

т.е. одновременного наличия и ребра e_j^2 ортографической проекции $\mathbf{\Pi^2}$, инцидентного вершинам $v_{j_f}^2$ и $v_{j_g}^2$ ортографической проекции $\mathbf{\Pi^2}$, и ребра e_k^3 ортографической проекции $\mathbf{\Pi^3}$, инцидентного вершинам $v_{k_f}^3$ и $v_{k_g}^3$ ортографической проекции $\mathbf{\Pi^3}$. Если это условие выполнено, то добавить в матрицу E^{3D} трехмерных ребер псевдокаркасной модели ребро $e^{3D} = \begin{pmatrix} v_f^{3D} \\ v_g^{3D} \end{pmatrix}$ в виде строки номеров трехмерных вершин псевдокаркасной модели, инцидентных ребру e^{3D}: $(f\ g)$.
4. Положить $\overrightarrow{T}$=$(\varnothing)$.

ЭТАП 2.2. Для каждой строки $v_s^{3D} = (x_{i\,s}^1\ y_{j\,s}^2\ z_{i\,s}^1\ i_s\ j_s\ k_s)$, $s = \overline{1, n_v^{3D}}$, матрицы V^{3D} трехмерных вершин псевдокаркасной модели, добавить в матрицу E^{3D} трехмерных ребер псевдокаркасной модели все ребра, инцидентные вершине v_s^{3D} и ортогональные ортографической проекции $\mathbf{\Pi^2}$, следующим образом.

1. Найти все строки $v_t^{3D} = (x_{i\,t}^1\ y_{j_t}^2\ z_{i\,t}^1\ i_t\ j_t\ k_t)$, $t \in \{1, 2, ..., n_v^{3D}\}$ (в том числе, $t = s$), матрицы V^{3D} трехмерных вершин псевдокаркасной модели, для которых $j_t = j_s$, и записать каждую такую строку v_t^{3D} в вспомогательную матрицу D в виде строки $(t\ v_t^{3D}) = (t\ x_{i\,t}^1\ y_{j_t}^2\ z_{i\,t}^1\ i_t\ j_t\ k_t)$.
2. Записать в вектор $\overrightarrow{T}$ весь первый столбец вспомогательной матрицы D.
3. Для каждой пары $f \neq g$ элементов вектора $\overrightarrow{T}$ такой, что $i_f < i_g$ и

$k_f < k_g$, проверить одновременное наличие двух ребер

$$e_i^1 = \begin{pmatrix} v_{i_f}^2 \\ v_{i_g}^2 \end{pmatrix} \text{ и } e_k^3 = \begin{pmatrix} v_{k_f}^3 \\ v_{k_g}^3 \end{pmatrix},$$

т.е. одновременного наличия и ребра e_i^1 ортографической проекции $\mathbf{\Pi^1}$, инцидентного вершинам $v_{i_f}^1$ и $v_{i_g}^1$ ортографической проекции $\mathbf{\Pi^1}$, и ребра e_k^3 ортографической проекции $\mathbf{\Pi^3}$, инцидентного вершинам $v_{k_f}^3$ и $v_{k_g}^3$ ортографической проекции $\mathbf{\Pi^3}$. Если это условие выполнено, то добавить в матрицу E^{3D} трехмерных ребер псевдокаркасной модели ребро $e^{3D} = \begin{pmatrix} v_f^{3D} \\ v_g^{3D} \end{pmatrix}$ в виде строки номеров трехмерных вершин псевдокаркасной модели, инцидентных ребру e^{3D}: $(f\ g)$.

4. Положить $\overrightarrow{T}$=$(\varnothing)$.

ЭТАП 2.3. Для каждой строки $v_s^{3D} = (x_{i\,s}^1\ y_{j\,s}^2\ z_{i\,s}^1\ i_s\ j_s\ k_s)$, $s = \overline{1, n_v^{3D}}$, матрицы V^{3D} трехмерных вершин псевдокаркасной модели, добавить в матрицу E^{3D} трехмерных ребер псевдокаркасной модели все ребра, инцидентные вершине v_s^{3D} и ортогональные ортографической проекции $\mathbf{\Pi^3}$, следующим образом.

1. Найти все строки $v_t^{3D} = (x_{i\,t}^1\ y_{j_t}^2\ z_{i\,t}^1\ i_t\ j_t\ k_t)$, $t \in \{1, 2, ..., n_v^{3D}\}$ (в том числе, $t = s$), матрицы V^{3D} трехмерных вершин псевдокаркасной модели, для которых $k_t = k_s$, и записать каждую такую строку v_t^{3D} в вспомогательную матрицу D в виде строки $(t\ v_t^{3D}) = (t\ x_{i\,t}^1\ y_{j_t}^2\ z_{i\,t}^1\ i_t\ j_t\ k_t)$.
2. Записать в вектор $\overrightarrow{T}$ весь первый столбец вспомогательной матрицы D.
3. Для каждой пары $f \neq g$ элементов вектора $\overrightarrow{T}$ такой, что $i_f < i_g$ и $j_f < j_g$, проверить одновременное наличие двух ребер

$$e_i^1 = \begin{pmatrix} v_{i_f}^3 \\ v_{i_g}^3 \end{pmatrix} \text{ и } e_j^2 = \begin{pmatrix} v_{j_f}^2 \\ v_{j_g}^2 \end{pmatrix},$$

т.е. одновременного наличия и ребра e_i^1 ортографической проекции $\mathbf{\Pi^3}$, инцидентного вершинам $v_{i_f}^1$ и $v_{i_g}^1$ ортографической проекции $\mathbf{\Pi^1}$, и ребра e_j^2 ортографической проекции $\mathbf{\Pi^2}$, инцидентного вершинам $v_{j_f}^2$ и $v_{j_g}^2$ ортографической проекции $\mathbf{\Pi^2}$. Если это условие выполнено, то добавить в матрицу E^{3D} трехмерных ребер псевдокаркасной модели

ребро $e^{3D} = \begin{pmatrix} v_f^{3D} \\ v_g^{3D} \end{pmatrix}$ в виде строки номеров трехмерных вершин псевдокаркасной модели, инцидентных ребру e^{3D}: $(f\ g)$.

4. Положить $\overrightarrow{T}$=$(\varnothing)$.

2.2. Выводы по второй главе

В работе основное внимание уделено восстановлению псевдокаркасных моделей по ортографическим проекциям, выполненному на основе адаптированных алгоритмов восстановления 3D объектов по ортографическим проекциям [6] и [12].

Псевдокаркасной моделью являться совокупность всех каркасных трехмерных моделей, которые могут быть построены по данному набору трех ортографических проекций.

Одна и та же ортографическая проекция может иметь несколько представлений различными наборами ребер.

Стандартное матричное представление ортографической проекции Π^i содержит все ребра любого другого матричного представления этой ортографической проекции Π^i, $i \in \{1, 2, 3\}$.

Матрицы трехмерных ребер достаточно, чтобы построить трехмерную псевдокаркасную модель.

3. РАЗРАБОТКА ПРОГРАММЫ

3.1. Ввод данных

3.1.1. Выбор формата файлов входных данных

В виду простоты в освоении и легкости в использовании, для ввода данных был выбран формат текстовых файлов TXT. Файл TXT – стандартный текстовый документ Windows, который представляет текст без форматирования и может быть открыт любой программой для работы с текстом. Самая простая и доступная из этих программ – Блокнот Windows. Текстовые файлы по умолчанию ассоциированы именно с этой программой.

Текстовые файлы также открывают и большинство других программ, например, любые браузеры: Internet Explorer, Chrome или Firefox. Не является проблемой открытие текстовых файлов и на мобильных устройствах, в других операционных системах.

3.1.2. Выбор способа представления входных данных

Перед началом работы программы необходимо заполнить три текстовых документа, каждый из которых представляет расположение вершин в ортографической проекции на соответствующей координатной плоскости, следующим образом.

В первой строке файла указывается количество вершин в ортографической проекции. Далее приводится матрица, каждая строка которой имеет вид

$$x\ y\ z$$

и представляет вершину ортографической проекции с координатами $(x,\ y,\ z)$.

Примеры заполнения показаны на рисунке 3.1

В результате работы программы пользователю будет представлена трехмерная каркасная модель, описанная тремя ортографическими проекциями, которые были введены перед началом работы программы. Пример результата работы программы представлен на рисунке 3.2.

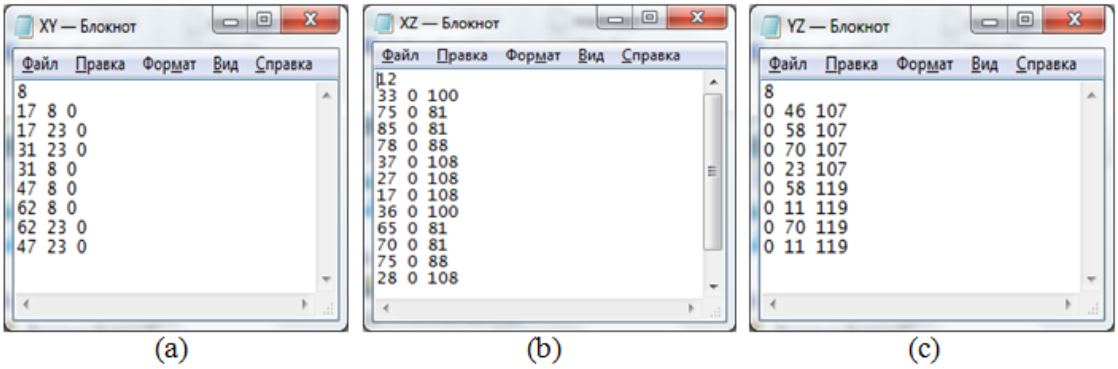

Рисунок 3.1 – Матричное представление ортографической проекции на плоскости OXY(a), OXZ(b), OYZ(c)

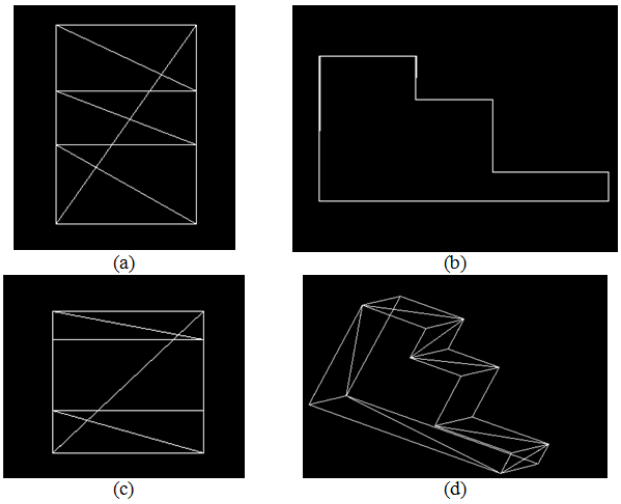

Рисунок 3.2 – Результат работы программы:(a) вид сверху,(b) вид сбоку,(c) вид спереди,(d) псевдокаркасная модель

3.2. Вывод данных

3.2.1. Вывод псевдокаркасной модели на экран

При описании трехмерного объекта используется матрица размера $n \times 4$

$$\begin{pmatrix} x_1 & y_1 & z_1 & 1 \\ x_2 & y_2 & z_2 & 1 \\ \dots & & & \\ x_n & y_n & z_n & 1 \end{pmatrix};$$

где x_i, y_i, z_i–координаты i-ой вершины, n–количество вершин, и матрица E^{3D} трехмерных ребер псевдокаркасной модели, имеющая размер $n_e^{3D} \times 2$ и строки вида

$$(sp),$$

где $s < p$ и $s, p \in 1, 2, ..., n_v^{3D}$ - номера трехмерных вершин v_s^{3D} и v_p^{3D} псевдокаркасной модели, инцидентных ребру, задаваемому этой строкой.

Для прорисовки ребер трехмерного объекта используется функция

```
void rib :: draw()
int X1, Y1, X2, Y2;
line (X1, Y1, X2, Y2);
```

3.2.2. Преобразования каркасной модели

Для точного понимания формы и конструкции пространственного объекта является качественная визуализация. В большинстве случаев, которые возникают при проектировании, наглядное представление о трехмерном объекте можно получить только при рассмотрении с различных сторон на определенных расстояниях (выполнении преобразований поворота и переноса) и построении проекций.

Матрица трехмерного поворота на некоторый угол в одной из координатных плоскостей, показанного на рисунке 3.3, строится в предположении, что

1) центр поворота располагается в начале координат $O(0;\ 0;\ 0)$;
2) осью поворота является координатная ось, перпендикулярная плоскости поворота;
3) при движении вдоль оси поворота в сторону уменьшения координаты поворот виден против часовой стрелки.

В программе реализован поворот относительно любой оси на угол α, выполняется данная операция с помощью матриц преобразования представленных ниже.

$$OX = \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos(\alpha) & sin(\alpha) \\ 0 & -sin(\alpha) & cos(\alpha) \end{pmatrix}.$$

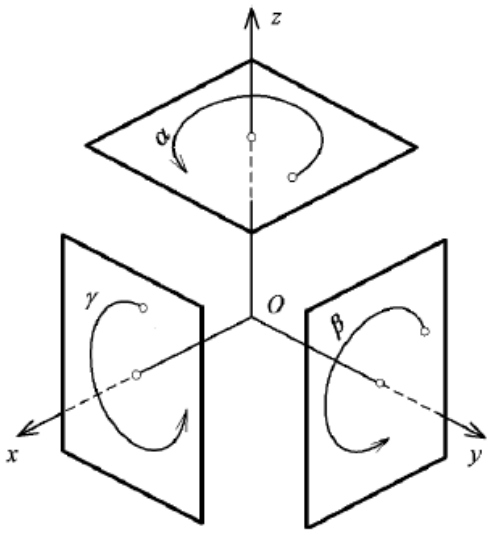

Рисунок 3.3 – Положительные трехмерные повороты в координатных плоскостях.

$$OY = \begin{pmatrix} cos(\alpha) & 0 & -sin(\alpha) \\ 0 & 1 & 0 \\ sin(\alpha) & 0 & cos(\alpha) \end{pmatrix}.$$

$$OZ = \begin{pmatrix} cos(\alpha) & sin(\alpha) & 0 \\ -sin(\alpha) & cos(\alpha) & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Здесь

OX– матрица поворота вокруг оси X,

OY– матрица поворота вокруг оси Y,

OZ– матрица поворота вокруг оси Z.

3.3. Выводы по третьей главе

Для работы с программой был выбран формат TXT, из-за большой распространенности в средах ОС, а также простоты использования.

До начала работы необходимо выполнить подготовительную работу, заключающаяся в заполнении трех текстовых документов, представляющих расположение вершин в ортографической проекции на координатной плоскости.

В программе реализованы методы поворота фигуры с помощью матриц преобразования.

ЗАКЛЮЧЕНИЕ

Целью данной выпускной квалификационной работы являлось создание программы по восстановлению трехмерной каркасной модели по трем ортографическим проекциям. Для достижения данной цели были поставлены следующие задачи:

1) выполнить сбор и анализ уже имеющихся методов;
2) разработать и реализовать алгоритм программы;
3) выполнить проверку на работоспособность программы;
4) продемонстрировать работоспособность программы на конкретных примерах.

При работе над поставленными задачами была изучена литература о восстановлении 3D каркасных моделей по их 2D проекциям, был рассмотрен метод Qing-Wen Yan [30] который описывает восстановление трехмерных ребер на основе таблицы возможных конфигураций.Также произведено ознакомление со статьей Idesawa [12], которая является одной из первых работ, предполагающих алгоритм восстановления трехмерных моделей.

Во второй главе рассмотрена адаптация алгоритма, предложенного Rocco Furferi [6], позволяющая выполнять восстановление 3D каркасных моделей по их ортографическим проекциям.В третьей главе представлена программная реализация этого алгоритма,а также функции, которые программа может выполнять.Была выполнена проверка работоспособности программы на разных примерах.

Таким образом, все поставленные задачи решены.

ОБОЗНАЧЕНИЯ

Обозначение	Понятие
$\mathbf{\Pi^i}$	ортографическая проекция (главный фасад, боковой фасад и план, соответственно), $i \in \{1, 2, 3\}$
$\mathbf{V^i}=$ $\{v_1^i,\ v_2^i, ..., v_{n_v^i}^i\}$	упорядоченный набор вершин (опорных точек) ортографической проекции $\mathbf{\Pi^i}$, $i \in \{1, 2, 3\}$
n_v^i	число вершин ортографической проекции $\mathbf{\Pi^i}$, $i \in \{1, 2, 3\}$
$v_j^i = (x_j^i; y_j^i; z_j^i)$	j-я вершина ортографической проекции $\mathbf{\Pi^i}$, $j \in \{1, 2, ..., n_v^i\}$, $i \in \{1, 2, 3\}$
$\mathbf{E^i}= \{e_1^i,\ e_2^i, ..., e_{n_e^i}^i\}$	упорядоченный набор ребер ортографической проекции $\mathbf{\Pi^i}$, $i \in \{1, 2, 3\}$
n_e^i	число ребер ортографической проекции $\mathbf{\Pi^i}$, $i \in \{1, 2, 3\}$
$e_k^i = \begin{pmatrix} v_{j_1}^i \\ v_{j_2}^i \end{pmatrix} = \begin{pmatrix} x_{j_1}^i & y_{j_1}^i & z_{j_1}^i \\ x_{j_2}^i & y_{j_2}^i & z_{j_2}^i \end{pmatrix}$	матрица k-го ребра ортографической проекции $\mathbf{\Pi^i}$, соединяющего вершины $v_{j_1}^i = (x_{j_1}^i; y_{j_1}^i; z_{j_1}^i)$ и $v_{j_2}^i = (x_{j_2}^i; y_{j_2}^i; z_{j_2}^i)$, где $j_1, j_2 \in \{1, 2, ..., n_v^i\}$, $k \in \{1, 2, ..., n_e^i\}$, $i \in \{1, 2, 3\}$
$\Pi^i = \begin{pmatrix} e_1^i \\ e_2^i \\ ... \\ e_{n_e^i}^i \end{pmatrix}$	матрица размера $2 \cdot n_e^i \times 3$, представление ортографической проекции $\mathbf{\Pi^i}$, $i = \overline{1, 3}$
$OBJ = \begin{pmatrix} \Pi^1 \\ \Pi^2 \\ \Pi^3 \end{pmatrix}$	матрица размера $2 \cdot (n_e^1 + n_e^2 + n_e^3) \times 3$, представление набора всех трех ортографических проекций
$f_k^i(x, y, z) = 0$	уравнение линии, содержащей ребро e_k^i, $k =\in \{1, n_e^i\}$, $i \in \{1, 2, 3\}$

СПИСОК ЛИТЕРАТУРЫ

1. Chen Z.,Perng D.,Chen C.,Wu C. Fast reconstruction of 3D mechanical parts from 2D orthographic views with rules. Journal of Computer Integrated Manufacturing, 1992, vol.5, no. 1,pp. 2–9.

2. Cicek A., Gulesin M. Reconstruction of 3D models from 2D orthographic views using solid extrusion and revolution. Journal of Materials Processing Technology, 2004, vol. 152, no. 3, pp. 291–298.

3. Diestel R. Graph theory. New York, 2000.

4. Dutta D., Srinivas Y.L. Reconstruction of curved solids from two polygonal ortographic views. 1992, vol. 24, no. 3, pp. 149–159

5. Fahiem M.A.,Haq S.A.,Saleemi F. A Review of 3D Reconstruction Techniques from 2D Orthographic Line Drawings. Geometric Modeling and Imaging, 2007, pp.60–66.

6. Furferi R., Governi L., Palai M., Volpe Y. The reconstruction problem: integrating different approaches into a systematic procedure for pseudo wireframe retrieval journal for geometry and graphics. 2011, vol. 15, no. 1, pp. 79—91.

7. Geng W., Wang J., Zhang Y. Embedding visual cognition in 3D reconstruction from multi-view engineering drawings. Computer-Aided Design, 2002, vol. 34, no. 4, pp. 321–336.

8. Golovin S.I., Veselov N.A. Automatic reconstruction of curved solids from three orthographic projections. 2007, vol. 2, pp. 53–58.

9. Gong J., Zhang H., Zhang G.,Sun J. Solid reconstruction using recognition of quadric surfaces from orthographic views. Elsevier, 2006.

10. Gong J., Zhang H., Zhang G.,Sun J. Converting Hybrid Wire-frames to B-rep Models. 2007.

11. Gorgani H.H., Pak A.J. A genetic algorithm based optimization method in 3D solid reconstruction from 2D multi-view engineering drawings. 2018, vol. 49, no. 1, pp. 161–170.

12. Idesawa M. A system to generate a solid figure from a three view. 1973, vol. 16, no. 92, pp. 216–225.

13. Idesawa M. Automatic input of line drawing and generation of solid figure from three-view data. Proceedings of the International Joint Computer Symposium, 1975, pp. 304 – 311.

14. Ismail N.B.,Bakar N.B.A. Boundary representation-based feature recognition. Jurnal Teknologi, 1997, pp. 65–74.

15. Kuo M.H. Reconstruction of quadric surface solids from three-view engineering drawings. Computer-Aided Design, 1998, vol. 30, no. 7, pp. 517—527.

16. Lafue G. Recognition of Three-Dimensional Objects from Orthographic Views. 3rd Annual Conference on Computer Graphics, Interactive Techniques, and Image Processing, 1976, pp. 103–108.

17. Lee H., Han S. Reconstruction of 3D interacting solids of revolution from 2D orthographic views, Computer-Aided Design, 2005, vol. 37, no. 13, pp. 1388–1398.

18. Liu S., Hu S., Chen Y., Sun J. Reconstruction of curved solids from engineering drawings. Computer-Aided Design, 2001 vol. 33 no. 14, pp. 1059–1072.

19. Marston R.E.,Kuo M.H. Reconstruction of 3D objects from three ortographic projections using a desision-chaining method.Workshop on Machine Vision Applications, 1994, pp. 423–426.

20. Markowsky, G., Wesley, M. Fleshing out wire frames: reconstruction of objects, part 1. IBM Journal of Research and Development, 1980, vol. 24, no. 5,pp. 582–597.

21. Markowsky, G., Wesley, M. Fleshing out wire frames: reconstruction of objects, part 2. IBM Journal of Research and Development, 1981, vol. 25, no. 6,pp. 934–954.

22. Masuda H., Numao M. A cell-based approach for generating solid objects from orthographic projections. Computer-Aided Design, 1997, vol. 29, no. 3, pp. 177–187.

23. Sakurai H.,Gossard D. Solid model input through orthographic views. Computer Graphics,1983, vol. 17, no. 3, pp.243–252.

24. Shin B. S., Shin Y. G. Fast 3D solid model reconstruction from orthographic views. Computer-Aided Design, 1998, vol. 30, no. 1. pp. 63–76.

25. Shixia L.,Shimin H.,Jiaguang S. Two accelerating techniques for 3D reconstruction. Journal of Computer Science and Technology, 2002, vol. 17, no. 3, pp. 362–368.

26. Tanakaa M.,Anthonyb L.,Kaneedaa T.,Hirookac J. A single solution method for converting 2D assembly drawings to 3D part drawings. Computer-Aided Design, 2004, vol. 36, pp. 723–734.

27. Wang Z., Latif M. Reconstruction of 3D Solid Models Using Fuzzy Logic Recognition. Proceedings of the World Congres on Engineering, 2007, vol. 1, pp 37–42.

28. Watanabe T. Revision of Inconsistent Orthographic Views. Journal for Geometry and Graphics, 1998, vol. 2, no. 1, pp 45–53.

29. Xie B., Chen L., Chen L. A new reconstruction method based on the volume-based approach. Proceedings of the 2009 International Symposium on Information Processing, 2009, pp. 132–135.

30. Yan Q.,Chen C.,Tang Z. Efficient algorithm for the reconstruction of 3D objects from orthographic projections. Computer-Aided Design, 1994, vol. 26, no.9,pp. 699–717.

31. You C.F., Yang S.S. Automatic Feature Recognition from Engineering Drawings, The International Journal of Advanced Manufacturing Technology, 2008, vol. 14, no. 7, pp. 495–507.

32. You C.F., Yang S.S. Reconstruction of curvilinear manifold objects from orthographic views. Computers & Graphics, 2009, vol. 20, no. 2, pp. 275–293.

33. Zakharov A., Zhiznyakov A., Three-dimensional reconstruction from projections based on incidence matrices of patterns. AASRI Procedia, 2014, vol. 9,pp. 72–77.

34. Zhang A.,Xue Y.,Sun X.,Hu Y.,Luo Y.,Wang Y.,Zhong S.,Wang J.,Tang J.,Cai G. Reconstruction of 3D curvilinear wireframe model from 2D orthographic views. Lecture Notes in Computer Science, 2004, pp. 404–412.

ПРИЛОЖЕНИЕ

figure.cpp

```
#include <cstdio>
#include <math.h>
#include "geometry.h"
//================================================
//rib :: rib()
//================================================
rib :: rib(){
   x1 = x2 = 0;
   y1 = y2 = 0;
   z1 = z2 = 0;
}
//================================================
//rib :: rib(double X1, double Y1, double Z1, double X2, double Y2,
 double Z2)
//================================================
rib :: rib(double X1, double Y1, double Z1, double X2, double Y2,
 double Z2){
   x1 = X1;
   x2 = X2;
   y1 = Y1;
   y2 = Y2;
   z1 = Z1;
   z2 = Z2;
}
//================================================
//void rib :: draw()
//================================================
void rib :: draw(){
   int X1, Y1, X2, Y2;
   X1 = width * ((x1 - x_min) / (x_max - x_min));
```

```
   Y1 = height * (1 - (y1 - y_min) / (y_max - y_min));
   X2 = width * ((x2 - x_min) / (x_max - x_min));
   Y2 = height * (1 - (y2 - y_min) / (y_max - y_min));

   line(X1, Y1, X2, Y2); //Соединяем точки
}
//================================================
//void rib :: multi(double *a)
//================================================
void rib :: multi(double *a){
   double X1, X2, Y1, Y2, Z1, Z2;
   X1 = x1 * a[0] + y1 * a[3] + z1 * a[6];
   X2 = x2 * a[0] + y2 * a[3] + z2 * a[6];
   Y1 = x1 * a[1] + y1 * a[4] + z1 * a[7];
   Y2 = x2 * a[1] + y2 * a[4] + z2 * a[7];
   Z1 = x1 * a[2] + y1 * a[5] + z1 * a[8];
   Z2 = x2 * a[2] + y2 * a[5] + z2 * a[8];
   x1 = X1;
   x2 = X2;
   y1 = Y1;
   y2 = Y2;
   z1 = Z1;
   z2 = Z2;
}
//================================================
//figure :: figure(const char *file)
//================================================
figure :: figure(const char *file){
   FILE *f1 = fopen (file, "r+");
   int k; //Кол-во точек в файле
   double x1, x2, y1, y2, z1, z2;
   fscanf(f1, "%d", &k);
   n = k;
   ribs.reserve(n);
```

```
   for(int i = 0; i < n; i++){
      fscanf(f1, "%lf%lf%lf%lf%lf%lf", &x1, &y1, &z1, &x2, &y2, &z2);
      rib buf(x1, y1, z1, x2, y2, z2);
      ribs.push_back(buf);
   }
   fclose(f1);
}
//================================================
//figure :: figure(int n)
//================================================
figure :: figure(int n){
   ribs.reserve(n);
   this->n = n;
}
//================================================
//void figure :: draw_figure()
//================================================
void figure :: draw_figure(){
   for(int i = 0; i < ribs.size(); i++){ //Отрисовать все ребра
      ribs[i].draw();
   }
}
//================================================
//void figure :: multi_figure(double *a)
//================================================
void figure :: multi_figure(double *a){
   for(int i = 0; i < ribs.size(); i++){
      ribs[i].multi(a);
   }
}
//================================================
//void figure :: po_Z(figure f1, figure f2)
//================================================
```

```
void figure :: po_Z(figure f1, figure f2){
   double x1, y1, z1, x2, y2, z2;
   for(int i = 0; i < f1.ribs.size(); i++){
      for(int j = 0; j < f2.ribs.size(); j++){
         if(f1.ribs[i].z1 == f2.ribs[j].z1){
            x1 = f1.ribs[i].x1;
            y1 = f2.ribs[j].y1;
            z1 = f1.ribs[i].z1;
            if(f1.ribs[i].z2 == f2.ribs[j].z1){
               x2 = f1.ribs[i].x2;
               y2 = f2.ribs[j].y1;
               z2 = f1.ribs[i].z2;
               rib buf(x1, y1, z1, x2, y2, z2);
               ribs.push_back(buf);
            }
            if(f1.ribs[i].z2 == f2.ribs[j].z2){
               x2 = f1.ribs[i].x2;
               y2 = f2.ribs[j].y2;
               z2 = f1.ribs[i].z2;
               rib buf(x1, y1, z1, x2, y2, z2);
               ribs.push_back(buf);
            }
            if(f1.ribs[i].z1 == f2.ribs[j].z2){
               x2 = f1.ribs[i].x1;
               y2 = f2.ribs[j].y2;
               z2 = f1.ribs[i].z1;
               rib buf(x1, y1, z1, x2, y2, z2);
               ribs.push_back(buf);
            }
         }
         if (f1.ribs[i].z2 == f2.ribs[j].z1){
            x1 = f1.ribs[i].x2;
            y1 = f2.ribs[j].y1;
```

```
            z1 = f1.ribs[i].z2;
            if (f1.ribs[i].z2 == f2.ribs[j].z2){
               x2 = f1.ribs[i].x2;
               y2 = f2.ribs[j].y2;
               z2 = f1.ribs[i].z2;
               rib buf(x1, y1, z1, x2, y2, z2);
               ribs.push_back(buf);
            }
         }
      }
   }
}
//================================================
//void figure :: po_Y(figure f1, figure f2)
//================================================
void figure :: po_Y(figure f1, figure f2){
   double x1, y1, z1, x2, y2, z2;
   for (int i = 0; i < f1.ribs.size(); i++){
      for (int j = 0; j < f2.ribs.size(); j++){
         //double x1, y1, z1, x2, y2, z2;
         if (f1.ribs[i].y1 == f2.ribs[j].y1){
            x1 = f1.ribs[i].x1;
            y1 = f1.ribs[i].y1;
            z1 = f2.ribs[j].z1;
           /* if (f1.ribs[i].y2 == f2.ribs[j].y1){
               x2 = f1.ribs[i].x2;
               y2 = f1.ribs[i].y2;
               z2 = f2.ribs[i].z1;
               rib buf(x1, y1, z1, x2, y2, z2);
               ribs.push_back(buf);
            }*/
            if (f1.ribs[i].y2 == f2.ribs[j].y2){
               x2 = f1.ribs[i].x2;
```

```
                y2 = f1.ribs[i].y2;
                z2 = f2.ribs[j].z2;
                rib buf(x1, y1, z1, x2, y2, z2);
                ribs.push_back(buf);
             }
             if (f1.ribs[i].y1 == f2.ribs[j].y2){
                x2 = f1.ribs[i].x1;
                y2 = f1.ribs[i].y1;
                z2 = f2.ribs[j].z2;
                rib buf(x1, y1, z1, x2, y2, z2);
                ribs.push_back(buf);
             }
          }
          if (f1.ribs[i].y2 == f2.ribs[j].y1){
             x1 = f1.ribs[i].x2;
             y1 = f1.ribs[i].y2;
             z1 = f2.ribs[j].z1;
             if (f1.ribs[i].y2 == f2.ribs[j].y2){
                x2 = f1.ribs[i].x2;
                y2 = f1.ribs[i].y2;
                z2 = f2.ribs[j].z2;
                rib buf(x1, y1, z1, x2, y2, z2);
                ribs.push_back(buf);
             }
          }
       }
    }
}
//================================================
//void figure :: control(char c)
//================================================
void figure :: control(char c){
   switch(c){
```

```
      case '1':
         rotate_x(5);
         break;
      case '2':
         rotate_y(5);
         break;
      case '3':
         rotate_z(5);
         break;
   }
}
//================================================
//void figure :: izom ()
//================================================
void figure :: izom (){
   double alpha = 45 * Pi;
   double beta = 35.26 * Pi;
   double array[9] = {
      cos(alpha), sin(alpha) * sin(beta), 0,
      0, cos(beta), 0,
      sin(alpha), -cos(alpha) * sin(beta), 0
   };
   multi_figure(array);
}
//================================================
//void figure :: rotate_x(double alpha)
//================================================
void figure :: rotate_x(double alpha){ //Поворот относительно оси
                                          OX на угол alpha
   alpha *= Pi;
   double array[9] = {
      1, 0, 0,
      0, cos(alpha), sin(alpha),
```

```
      0, -sin(alpha), cos(alpha),
   };
   multi_figure(array);
}
//================================================
//void figure :: rotate_y(double alpha)
//================================================
void figure :: rotate_y(double alpha){//Поворот относительно оси
                                                  OY на угол alpha
   alpha *= Pi;
   double array[9] = {
      cos(alpha), 0, -sin(alpha),
      0, 1, 0,
      sin(alpha), 0, cos(alpha),
   };
   multi_figure(array);
}
//================================================
//void figure :: rotate_z(double alpha)
//================================================
void figure :: rotate_z(double alpha){ //Поворот относительно оси
                                                  OZ на угол alpha
   alpha *= Pi;
   double array[9] = {
      cos(alpha), sin(alpha), 0,
      -sin(alpha), cos(alpha), 0,
      0, 0, 1,
   };
   multi_figure(array);
}
```

geometry.h

```
#ifndef _GEOMETRY_H
#define _GEOMETRY_H
#include <iostream>
#include <vector>
#include "graphics.h"
using namespace std;
//================================================
//Глобальные переменные / Константы
//================================================
const double Pi = 3.141592 / 180;
const int width = 800;
const int height = 600;
const double x_min = -20;
const double x_max = 20;
const double y_min = -20;
const double y_max = 20;
//================================================
//class rib
//================================================
class rib{
   public:
      double x1, y1, z1;
      double x2, y2, z2;
      rib();
      rib(double X1, double Y1, double Z1, double X2, double Y2, doubl
      void draw(); //Отрисовать ребро
      void multi(double *a);
};
//================================================
//class figure
//================================================
class figure{
```

```
   public:
      vector <rib> ribs;
      int n;
      figure(const char *file);
      figure(int n);
      ~figure(){}
      void po_Z(figure f1, figure f2); //Склейка точек по проекции XZ
      void po_Y(figure f1, figure f2); //Склейка точек по проекции XY
      void draw_figure(); //Отрисовать фигуру
      void multi_figure(double *a);
      void izom(); //Изометрическая проекция
      void rotate_x(double alpha);
      void rotate_y(double alpha);
      void rotate_z(double alpha);
      void control(char c);
};
#endif
```

main.cpp

```
#include <math.h>
#include "geometry.h"
//Управление на 1, 2, 3
//================================================
// int main()
//================================================
int main(){
   initwindow(width, height); //Создание окна
   figure YX("YX.txt"); //Проекция на плоскость OXY
   figure YZ("YZ.txt"); //Проекция на плоскость OYZ
   figure XZ("XZ.txt"); //Проекция на плоскость OXZ
   figure b(40);
   figure b1(40);
   figure b2(40);
```

```
    b.po_Z(XZ, YZ);
    b1.po_Y(YX, YZ);
    char c;
    // b.izom();
    //b1.izom();
    int p = 0;
    do{
      // YX.draw_figure();
       //YZ.draw_figure();
       //XZ.draw_figure();
       p = 1 - p;
       setactivepage(p); // активная страница == 1-видимая
       cleardevice();
       if(kbhit()){
          c = getch();
       }else{
          c = 0;
       }
       b.control(c);
       b1.control(c);
       b.draw_figure();
       b1.draw_figure();
       setvisualpage(p);
    }while(1);
    getch();
    return 0;
}
```