Федеральное агентство по образованию РФ
Нижегородский Государственный Университет
им. Н. И. Лобачевского

Факультет вычислительной математики и кибернетики

Кафедра математического обеспечения ЭВМ

Описание проекта:

“Mechanics Studio .NET”

Система моделирования пространственных механизмов

Нижний Новгород
2005г.

Содержание

ВВЕДЕНИЕ 3

РАЗРАБОТЧИКИ ПРОЕКТА 4

1. ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ: ТЕОРИЯ МЕХАНИЗМОВ И МАШИН 5

1.1 Общее понятие механизма 5
1.2 Звенья механизма 5
1.3 Кинематические пары 7
1.4 Кинематические цепи 10
1.5 Подвижность механизма 11
1.6 Закон образования механизмов по Ассуру 13

2. ПРОЕКТ ПРОГРАММНОЙ СИСТЕМЫ 17

2.1 Описание проблемы 17
2.2 Требования к системе 17
2.3 Средства разработки 20
2.4 Модули системы 23
2.5 Диаграммы классов 25
2.6 Структура данных механизма 28
2.7 Аналог структурной схемы механизма 32
2.8 Программирование 3D-графики на Managed DirectX 34
2.8.1 Типы данных для выполнения матричных преобразований 34
2.8.2 Пространственные и проекционные системы координат 35
2.8.3 Материалы, текстуры и загрузка полигональных моделей из X-файла 37

3. ОПИСАНИЕ ГРАФИЧЕСКОГО ИНТЕРФЕЙСА (GUI) ПРОГРАММЫ “MECHANICS EDITOR .NET” 39

3.1 Установка и запуск программы 39
3.2 Панели главного окна программы 39
3.3 Главное меню 40
3.4 Настройка интерфейса 41
3.5 Инструментарий редактора 42
3.6 Управление видами 43
3.7 Конструктор механизмов 44
3.7.1 Создание новых звеньев механизма 44
3.7.2 Создание новых кинематических пар механизма 44
3.8 Структура механизма 45
3.9 Редактор объектов 46

ЗАКЛЮЧЕНИЕ И РЕЗУЛЬТАТЫ 47

ЛИТЕРАТУРА 49

ССЫЛКИ 50

Введение

Пространственные рычажные механизмы являются важной составляющей современной техники и производственных технологий. Механические системы часто выполняют весьма серьёзные задачи, требующие высокой надёжности и точности. Конструктору, обычно требуется знать, как будет действовать разработанный им механизм ещё до его окончательного изготовления. В связи с этими требованиями к процессу создания механизмов, крайне важными являются возможности выполнения полного проектирования механизмов и моделирования их работы.

Известными примерами пространственных механизмов (ПРМ) являются шасси, механизация крыла и механизмы управления самолётов, роботы-манипуляторы последовательной и параллельной структуры (в том числе платформы Стюарта) и др. Проектирование названных технических систем осуществляется сегодня с использованием CAD-систем, таких как “AutoCad” или “Компас”. Эти системы позволяют абсолютно точно спроектировать конструкцию механизма и его деталей до мельчайших подробностей, но обычно не позволяют выполнять структурный анализ и какие-либо расчеты работы этого механизма.

Моделирование кинематики ПРМ отличается высокой геометрической сложностью и определяет качество конечного продукта, стоимость его изготовления и эксплуатации. Выполнение кинематических расчетов ПРМ ставит на первый план совершенно другие задачи перед программной системой, в отличие от систем проектирования. Первостепенную важность приобретают проблемы представления структуры механизмов, методы параметрического моделирования анализируемых объектов, а так же возможности выполнения кинематических расчетов по построенной модели.

На сегодняшний день существует несколько систем динамического и кинематического анализа пространственных механизмов, в числе которых программы ADAMS и Dynamic Designer (компании MacNeal-Schwendler Corporation), используемые совместно с продуктами Autodesk Mechanical Desktop, SolidWorks и SolidEdge. [5] Несмотря на широкие возможности, предоставляемые этими системами, они используются лишь узким кругом специалистов работающих в крупных производственных компаниях, что связано, в первую очередь, с ценой на эти продукты. Таким образом, следует констатировать отсутствие достойного массового средства для автоматизации проектирования кинематики ПРМ.

Основной задачей этого проекта, является разработка новой программной системы, позволяющей проектировать кинематику пространственных механизмов. Программа должна иметь удобный для пользователя графический интерфейс, обладающий возможностью 3-х мерного моделирования аналога структурных схем механизмов. Кроме того, важно обеспечить средства представления структуры механизма и средства параметрического моделирования его объектов.

В разделе "теория механизмов" делается краткое введение в предметную область: теорию механизмов и машин. В следующем разделе "проект системы" описывается проблема разработки и вырабатываются требования к программной системе и средствам программирования для её создания, выполняется проектирование программы: разрабатываются типы данных для представления ПРМ и решения задач пользователя. Раздел "описание GUI" содержит подробное описание написанной на данный момент части программы для пользователя.

Разработчики проекта

- ## Турлапов Вадим Евгеньевич

Профессор кафедры МО ЭВМ факультета ВМК ННГУ, доктор технических наук;
Области научных интересов:

- Автоматизация геометрического моделирования и проектирования кинематики пространственных рычажных механизмов (шасси и другие механизмы летательных аппаратов, роботы-манипуляторы и тренажеры на основе платформ Стюарта и т.д.);
- Прикладное проектирование и развитие геоинформационных систем и технологий;
- Инженерная геометрия и компьютерная графика (прикладные системы и их развитие);
- Интернет-приложения;

Роль участника проекта: научный руководитель, постановщик задач;
Электронная почта: *vadim.turlapov@cs.vmk.unn.ru*

- ## Городецкий Евгений Станиславович

студент 1-го курса магистратуры кафедры МО ЭВМ, факультета ВМК ННГУ.
Области интересов:

- Современные технологии программирования;
- Прикладное компьютерное моделирование;
- Компьютерная графика, 3D моделирование и дизайн;
- Интернет-приложения;

Роль участника проекта: разработчик программы, разработчик сайта;
Электронная почта: *evsgor@gmail.com*

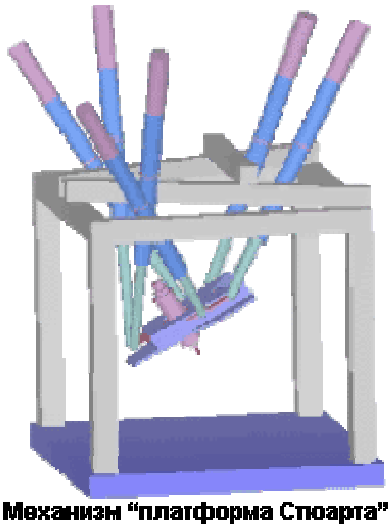
1. Описание предметной области: теория механизмов и машин

1.1 Общее понятие механизма

Перед тем, как начать подробное описание структуры механизма, дадим самое общее понятие механизма, не базирующееся на других понятиях:

> **Механизм** - это совокупность взаимосвязанных твёрдых тел, предназначенная для преобразования входов на одном или нескольких твёрдых телах в выходы на других твёрдых телах. [4]

Данное определение объясняет назначение любого механизма, заключающееся в преобразовании движения, но не даёт точного понимания структуры этого механизма. В следующих разделах, будет дано подробное описание структурных элементов механизма и способов их взаимодействия. После этого в разделе "кинематические цепи" будет дано более точное определение механизма, описывающее его структуру и базирующееся на введённых понятиях.



Механизм “платформа Стюарта”

На расположенном справа рисунке приведён пример механизма, называемого *платформой Стюарта*, который состоит в общей сложности из 8 твёрдых тел: конструкции крепления, 6 поршней и прикреплённой к ним платформы. Этот механизм служит для точного позиционирования платформы вместе с прикреплённым к ней устройством.

1.2 Звенья механизма

> **Звенья** – это твёрдые тела, из которых образуется механизм. При этом имеются в виду как абсолютно твёрдые, так и деформируемые и гибкие тела (в дальнейшем мы ограничимся рассмотрением только абсолютно твёрдых тел). Звеном может быть либо одна деталь, либо несколько деталей, соединённых в одну неизменяемую систему. [1]

Звенья различают по конструктивным признакам (*коленчатый вал, шатун, поршень, зубчатое колесо и т.д.*) и по характеру движения - например:

- *кривошип* – звено, вращающееся на полный оборот, вокруг неподвижной оси;
- *коромысло* – звено, вращающееся на не полный оборот;
- *ползун* – звено, совершающее поступательное прямолинейное движение;
- *стойка* – неподвижное звено механизма;

Понятие неподвижности стойки для механизмов транспортных машин, в частности летательных аппаратов, - условное, поскольку в этом случае сама стойка движется.

Если рассматривать звено свободным, т. е. не связанным с другими звеньями, то оно имеет шесть степеней подвижности.

> **Степенью подвижности** твердого тела называется возможность тела совершать движение, определяемое одним независимым параметром (понятие из теоретической механики).

Сложное движение свободного звена можно разложить, как это делается в теоретической механике, на два составляющих движения, а именно: на поступательное движение вместе с произвольно выбранной точкой, называемой *полюсом*, и вращательное движение вокруг этого полюса. В свою очередь, поступательное движение звена определяется законом изменения координат x, y, z полюса. Для определения вращательного движения достаточно знать закон изменения, например, трех углов Эйлера - *угла прецессии ψ, угла нутации θ и угла собственного вращения φ*. Таким образом, шесть величин x, y, z, ψ, θ, φ и являются шестью независимыми параметрами, определяющими движение свободного звена. Каждому такому параметру соответствует одно *простое движение*. В связи с этим *сложное движение* звена можно разложить на шесть простых составляющих движений.

Характер относительного движения подвижно-соединённых звеньев зависит только от мест их соединения.

> **Геометрическим элементом** называется место соединения одного звена с другим звеном. Геометрическим элементом может являться совокупность поверхностей, линий и точек звена, входящих в соприкосновение (контакт) с другим звеном.

Для того, чтобы элементы пары находились в постоянном соприкосновении, пара должна быть *замкнута* геометрическим (за счёт конструктивной формы звеньев) или силовым (силой тяжести, пружиной, силой давления жидкости или газа) способом.

Одно звено может иметь несколько геометрических элементов. Таким образом, для звена, подлежащего изучению в теории механизмов, характерным и *главным являются форма геометрических элементов и их взаимное расположение на звене*. Все другие факторы, как-то: материал, форма, конструкция звеньев, не отражают их относительного движения и поэтому не подлежат изучению в теории механизмов.

Кроме того, в теории механизмов пренебрегают также малыми деформациями звеньев, т. е. рассматривают механизмы с абсолютно твердыми звеньями, в отличие от механизмов реальных, в которых происходят и деформации звеньев и изменения их размеров вследствие неточности их изготовления.

Различаю входные и выходные звенья механизма:

- **Выходным** называю звено, совершающее движение, для которого предназначен механизм;
- **Входным** называю звено, которому сообщается движение, преобразуемое механизмом в требуемое движение выходного звена.

Число входных звеньев обычно равно числу степеней свободы механизма, т. е. числу его обобщённых координат, но возможно и не совпадение их.

1.3 Кинематические пары

Кинематической парой (сокращённо - парой) называют подвижное соединение двух соприкасающихся звеньев. [1]

Всякая кинематическая пара ограничивает движение соединяемых звеньев.

Ограничение, наложенное на движение твёрдого тела, называется **условием связи**.

Таким образом, *кинематическая пара накладывает условия связи на относительное движение двух соединяемых звеньев*. Очевидно, что наибольшее число условий связи наложенное кинематической парой, равно пяти.

Различное число условий связи, накладываемых на относительное движение звеньев кинематическими парами, позволяет разделить последние на 5 *классов*, так что пара k-го класса накладывает k условий связи, где k из {1,2,3,4,5}. Отсюда следует, что кинематическая пара k-го класса допускает в относительном движении звеньев 6-k степеней подвижности.

Следует заметить, что в механизмах применяются кинематические пары только пятого, четвертого и третьего классов. Кинематические же пары первого и второго классов не нашли применения в существующих механизмах.

Так как звенья соприкасаются геометрическими элементами, то, очевидно, кинематическая пара представляет собою совокупность таких элементов соединяемых звеньев. Отсюда следует, что *характер относительного движения соединяемых звеньев зависит от формы геометрических элементов*. Это относительное движение одного звена по отношению к другому может быть получено, если одно из двух соединяемых звеньев сделать неподвижным, а другому сообщить движение, допускаемое связями, накладываемыми кинематической парой.

Любая точка подвижного звена описывает в относительном движении траекторию, которую для краткости будем называть *траекторией относительного движения*. Если траектории относительного движения таких точек являются плоскими кривыми и располагаются в параллельных плоскостях, то пара называется *плоской*. В случае *пространственных* кинематических пар указанные траектории относительного движения представляют собою пространственные кривые.

Кроме разделения по классам, кинематические пары так же делят в зависимости от типа геометрического элемента пары:

- **высшие пары** – это пары, в которых при соединении двух звеньев контакт осуществляется лишь на кривых или точках;
- **низшие пары** – это пары, в которых при соединении двух звеньев контакт осуществляется по поверхностям.

Высшие кинематические пары применяются для уменьшения трения в элементах этих пар и часто реализуются в качестве роликов или подшипников. Но особенности внутреннего строения таких элементов, в общем случае, не влияют на относительное движение соединяемых парой звеньев. Существуют так же определённые приёмы, позволяющие

заменять механизмы с высшими кинематическими парами их аналогами с низшими парами (что позволяет упростить исследование кинематики механизма в дальнейшем). Поэтому далее мы будем рассматривать только механизмы с низшими парами.

Низшие кинематические пары наиболее часто применяются на практике и имеют более простое внутреннее строение, по сравнению с высшими парами. Элемент низшей кинематической пары представляет собой две скользящие друг по другу поверхности, что, с одной стороны распределяет нагрузку в этом элементе, а с другой стороны увеличивает трение при относительном движении звеньев. В связи с этим, использование низших кинематических пар позволяет передавать значительную нагрузку от одного звена на другое, благодаря именно тому, что в этих парах звенья соприкасаются по поверхности.

Таблица 1: Классификация кинематических пар по числу степеней свободы и числу связей [2]

Число связей (класс пары)	Число степеней свободы	Название пары	Рисунок	Условное обозначение
1	5	***Вращательная***		
1	5	***Поступательная***		
1	5	***Винтовая***		
2	4	***Цилиндрическая***		
2	4	***Сферическая с пальцем***		
3	3	***Сферическая***		
3	3	***Плоская***		
4	2	***Цилиндр-плоскость***		
5	1	***Шар-плоскость***		

В **таблице 1** приведены примеры кинематических пар всевозможных классов. В таблице для каждой из пар указано её название, рисунок и условное обозначение на схеме. Далее будут приведены дополнительные пояснения для наиболее часто используемых пар из этой таблицы.

Вращательная пара – одноподвижная (условное обозначение ”**1 В**”), допускает лишь относительное вращательное движение звеньев вокруг оси. Звенья пары соприкасаются по цилиндрической поверхности, следовательно, это низшая пара, замкнутая геометрически. Роль такой кинематической пары выполняет и более сложная конструкция – шарикоподшипник.

Поступательная пара – одноподвижная (условное обозначение ”**1 П**”), с геометрическим замыканием, низшая, допускает лишь прямолинейное поступательное относительное движение звеньев.

Цилиндрическая пара – двухподвижная (”**2 Ц**”), с геометрическим замыканием, низшая, допускает независимое вращательное и поступательное относительное движение звеньев.

Сферическая пара – трёхподвижная (”**3 С**”), с геометрическим замыканием, низшая, допускает три независимых относительных вращения звеньев вокруг осей x, y, z.

Сферическая пара с пальцем – двухподвижная (”**2 С**”), с геометрическим замыканием, низшая, допускает два независимых относительных вращения звеньев вокруг осей, определяемых прорезью и пальцем (добавленным к сферической паре).

Винтовая пара – одноподвижная, с геометрическим замыканием, низшая, допускает относительное винтовое движение звеньев с постоянным шагом. Угловые и линейные перемещения звеньев винтовой пары имею однозначное соответствие, в результате чего остаётся только одна степень подвижности.

Плоскостная пара, ***цилиндр-плоскость*** и ***шар-плоскость*** пары используют силовое замыкание, причём первая из них низшая, а две другие высшие. Эти пары практически не применяются в реальных механизмах и описаны в данном обзоре для полноты представления классификации кинематических пар.

Пары так же разделяют по типу относительного движения соединяемых звеньев на обратимые и необратимые:

- **Обратимые кинематические пары** допускают один и тот же вид относительного движения независимо от того, какое из звеньев условно выбрано неподвижным. Если кинематическая пара одноподвижная, то она однозначно обратимая;
- **Необратимые кинематические пары** допускают различные виды относительного движения соединяемых звеньев в зависимости от того, какое из них условно принято неподвижным.

В качестве необратимых пар в существующих механизмах применяются кулачковые пары, пары зацепления (различного вида зубчатые пары), фрикционные пары с гибкими звеньями. Изучение относительных движений звеньев, соединения которых представляют собой указанные пары, можно осуществить лишь при рассмотрении механизмов в целом, так как с помощью необратимых пар нельзя образовать двухзвенные механизмы, как это можно сделать с помощью обратимых пар. Изучение необратимых пар выходит за рамки данной работы и будет нами пропущено.

1.4 Кинематические цепи

Кинематической цепью называют систему звеньев, образующих между собой кинематические пары. [1] Кинематические цепи бывают простыми и сложными:

- **Простой кинематической цепью** называется такая цепь, у которой каждое звено входит не более чем в две кинематические пары;
- **Сложной кинематической цепью** называется цепь, в которой имеется хотя бы одно звено, входящее более чем в две кинематические пары.

Простые и сложные кинематические цепи, в свою очередь, делятся на замкнутые и разомкнутые:

- **Простой замкнутой кинематической цепью** называется простая кинематическая цепь, каждое звено которой входит в две кинематические пары;
- **Простой разомкнутой кинематической цепью** называется простая кинематическая цепь, в которой есть звенья, входящие только в одну кинематическую пару;
- **Сложной замкнутой кинематической цепью** называется такая сложная кинематическая цепь, каждое звено которой входит, по крайне мере, в две кинематические пары;
- **Сложной разомкнутой кинематической цепью** называется такая сложная кинематическая цепь, в которой имеются звенья, входящие в одну кинематическую пару.

Теперь, когда введены основные понятия теории механизмов, можно дать новое определение механизма, более точно описывающее его структуру:

Механизм – это кинематическая цепь, в состав которой входит неподвижное звено (стойка) и число степеней свободы которой равно числу обобщённых координат, характеризующих положение цепи относительно стойки. [1]

Движение звеньев механизма рассматривается по отношению к неподвижному звену – *стойке*. Те звенья, которые соединяются со стойкой, образуя пары пятого класса, передают на неё усилия и носят названия основных звеньев. Из числа последних выделяют входные звенья, закон движения которых является заданным.

Механизмы классифицируют по различным признакам, и в первую очередь их делят на механизмы с низшими и высшими парами; те и другие могут быть плоскими и пространственными.

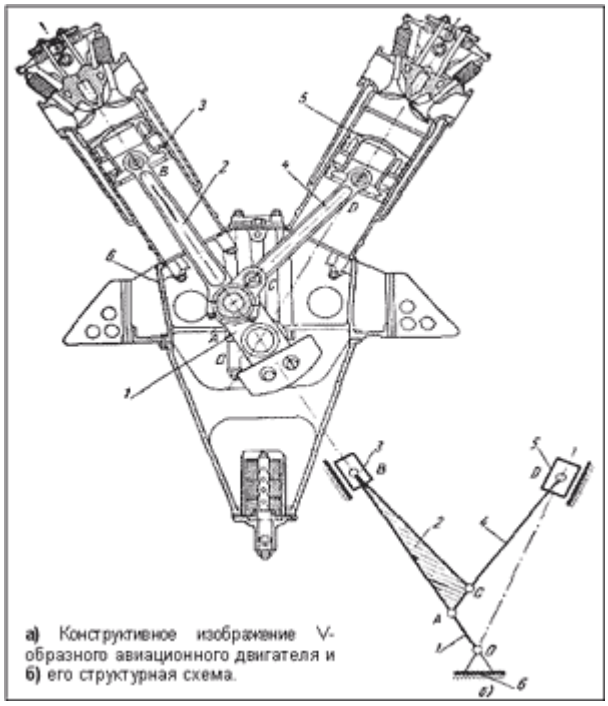
Плоским называется механизм, все подвижные точки которого движутся в параллельных плоскостях. [1]

Пространственным называют механизм, подвижные точки которого описывают неплоские траектории или траектории, лежащие в пересекающихся плоскостях. [1]

Наиболее распространённые механизмы с низшими парами – *рычажные, клиновые и винтовые*; с высшими парами – *кулачковые, зубчатые, фрикционные, мальтийские и храповые*.

При изображении механизма на чертеже различают его ***структурную схему*** с применением условных обозначений звеньев и пар (без указания размеров звеньев) и ***кинематическую схему*** с размерами необходимыми для кинематического расчёта. На схемах звенья обозначают цифрами, а пары и различные точки звеньев – буквами (с указанием числа степеней подвижности). Очевидно, кинематическая схема механизма не изображает его действительного устройства, а может быть использована лишь для кинематического или динамического исследования.

В качестве примера, на **рис. а)** приведено конструктивное изображение *V-образного механизма авиационного двигателя*, а на **рис. б)** – его структурная схема. В некоторых случаях применяют полу схематическое изображение механизмов, представляющее собою нечто среднее между конструктивным чертежом и структурной схемой.



а) Конструктивное изображение V-образного авиационного двигателя и **б)** его структурная схема.

1.5 Подвижность механизма

Существуют общие закономерности в структуре (строении) самых различных механизмов, связывающие число степеней свободы W механизма с числом звеньев и числом и видом его кинематических пар. Эти закономерности носят название *структурных формул механизмов*.

Под **<u>числом степеней подвижности механизма</u>** (обозначение **W**) будем понимать число независимых параметров, определяющих положение всех подвижных звеньев механизма.

Такими параметрами являются независимые координаты, определяющие положения входных звеньев. Очевидно, число степеней подвижности будет равно числу всех входных звеньев.

Для пространственных механизмов в настоящее время наиболее распространена *формула Малышева*, вывод которой производится следующим образом.

Введём обозначения следующих числовых величин:

- **m** – число звеньев (включая стойку);
- **p1, p2, p3, p4, p5** – число одно-, двух-, трёх-, четырёх- и пятиподвижных пар;
- **n=m-1** – число подвижных звеньев механизма.

Если бы все подвижные звенья были свободными телами, общее число степеней свободы было бы равно ***6n***. Однако каждая одноподвижная пара V-го класса, накладывает на относительное движение звеньев, образующих пару 5 связей, каждая двухподвижная пара IV-го класса – 4 связи и т. д. Следовательно, общее число степеней свободы, равное шести, будет уменьшено на величину:

$$\sum_{k=1}^{5}(6-k)\cdot p_k = 5\cdot p_1 + 4\cdot p_2 + 3\cdot p_3 + 2\cdot p_4 + p_5 \qquad (2.1)$$

В общее число наложенных связей может войти некоторое число q *избыточных (повторных) связей*, которые дублируют другие связи, не уменьшая подвижности механизма, а только обращая его в статически неопределимую систему [1]. Поэтому число степеней свободы пространственного механизма, равное числу степеней свободы его подвижной кинематической цепи относительно стойки, определяется по следующей *формуле Малышева:*

$$W = 6n - \left[\sum_{k=1}^{5}(6-k)\cdot p_k - q\right] = 6n - \left[5\cdot p_1 + 4\cdot p_2 + 3\cdot p_3 + 2\cdot p_4 + p_5 - q\right] \qquad (2.2)$$

При **q=0** механизм – статически определимая система, при **q>0** – статически неопределимая система. В частном случае, если число степеней подвижности механизма W найдено из геометрических соображений, то из формулы (2.2) можно найти число избыточных связей q и решить вопрос о статической определимости механизма. Или же, зная, что механизм статически определимый можно найти W.

Если избыточных связей нет (**q=0**), сборка механизма происходит без деформирования звеньев, последние как бы самоустанавливаются; поэтому такие механизмы называют самоустанавливающимися [1]. Если избыточные связи есть, то сборка механизма и движение его звеньев становится возможным только при деформировании последних.

Для плоских механизмов без избыточных связей структурная формула носит имя П. Л. Чебышева, впервые предложившего её в 1869 году для рычажных механизмов с вращательными парами и одной степенью свободы. В настоящее время формула Чебышева распространена на любые плоские механизмы и выводится с учётом избыточных связей следующим образом.

Пусть в плоском механизме, имеющем **m** звеньев (включая стойку), **n=m-1** – число подвижных звеньев, $\mathbf{p_н}$ – число низших пар и $\mathbf{p_в}$ – число высших пар. Если бы все подвижные звенья были свободными телами, совершающими плоское движение, общее число степеней свободы было бы равно **3n**. Однако каждая низшая пара накладывает на относительное движение звеньев, образующих пару, две связи, а каждая высшая пара накладывает одну связь, оставляя 2 степени свободы.

В число наложенных связей может войти некоторое число $\mathbf{q_н}$ *избыточных (повторных) связей*, устранение которых не увеличивает подвижности механизма. Следовательно, число

степеней свободы плоского механизма, т.е. число степеней свободы его подвижной кинематической цепи, относительно стойки, определяется по следующей *формуле Чебышева*:

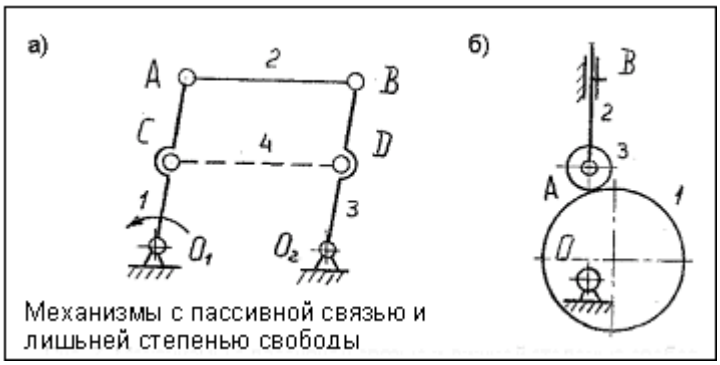
$$W_{\Pi} = 3n - (2 \cdot p_{H} + p_{B} - q_{\Pi}) \qquad (2.3)$$

Индекс “П” напоминает о том, что речь идёт об идеально плоском механизме, или, точнее, о его плоской схеме, поскольку за счёт неточностей изготовления плоский механизм в какой-то мере является пространственным.

По формулам (2.2), (2.3) проводят структурный анализ имеющихся механизмов и синтез структурных схем новых механизмов.

Пассивной связью можно назвать дополнительное звено, которое не накладывает ограничений на движение механизма. Обычно она вводится для увеличения жёсткости механизма или перевода механизма через особые положения, в которых наблюдается неопределенность движения звеньев.

В так называемом “*механизме параллельных кривошипов*” (рис. снизу а) звено 3 может изменить направление вращения при неизменном направлении вращения ведущего звена 1, когда механизм приходит в горизонтальное положение. Для того, чтобы этого избежать, в состав механизма включают дополнительное звено 4.



Механизмы с пассивной связью и лишней степенью свободы

При структурном анализе механизма пассивные связи не учитывают. Если по формуле степень подвижности механизма будет равна нулю, но заранее известно, что рассматриваемая цепь подвижна, то следует искать пассивные связи.

Наоборот, когда в механизм вводятся дополнительные звенья, имеющие собственную свободу движения, тогда говорят о **лишней степени свободы**. Так, степень подвижности *кулачкового механизма* (рис. сверху, б) W=2 (n=3, p1=3, p2=1). Построив заменяющий механизм, нетрудно убедиться, что он имеет степень подвижности W=1. Лишнюю степень подвижности для кулачкового механизма вносит свободный поворот ролика вокруг оси вращения без влияния на характер движения механизма в целом.

1.6 Закон образования механизмов по Ассуру

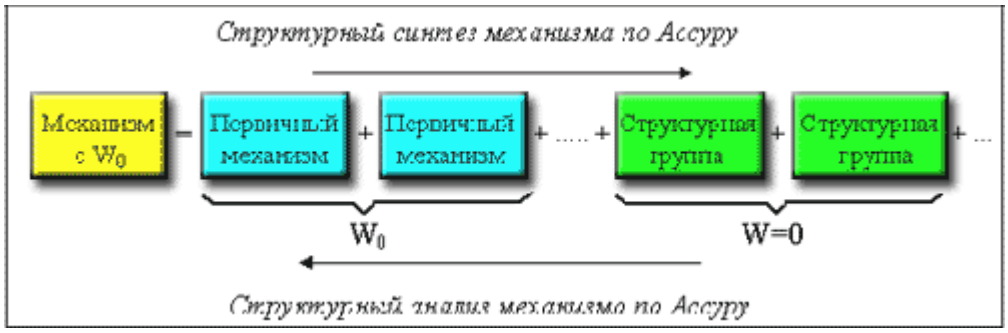
Впервые закон образования механизмов был сформулирован в 1914 г. русским ученым Леонидом Владимировичем Ассуром применительно к плоским шарнирным механизмам и

затем позднее распространен на другие механизмы И. И. Артоболевским. Закон образования механизмов можно сформулировать следующим образом:

> **Закон Ассура**: Всякий механизм представляет собою совокупность одного или нескольких, двухзвенных (первичных) механизмов и одной или нескольких групп нулевой подвижности.

Группы нулевой подвижности не изменяют числа степеней подвижности двухзвенных механизмов. В связи с этим сформулированный закон позволяет без помощи структурной формулы определить число степеней подвижности любого механизма.

Закон Асура является первостепенно важным для решения задач моделирования пространственных механизмов. Этот закон даёт простой алгоритм решения задачи синтеза произвольного механизма и методику анализа имеющегося механизма (см. рис. снизу).



> **Двухзвенный (первичный) механизм** состоит из подвижного звена и стойки, образующих обратимую пару.

Двухзвенные простейшие механизмы (т. е. механизмы, имеющие пару пятого класса) широко применяются в технике, например, в турбинах, электродвигателях, генераторах, воздуходувках и т. п.

Как следует из сформулированного закона Ассура, двухзвенный механизм представляет собою основу многозвенного механизма. Если двухзвенный механизм является простейшим, то в этом случае подвижное звено является входным звеном образуемого механизма (см. рис.1.4).

Очевидно, одна и та же стойка может одновременно входить в состав нескольких таких двухзвенных, не зависящих друг от друга механизмов. Подвижные звенья этих механизмов представляют собою не что иное, как начальные звенья, причем каждое из них всегда будет иметь одну степень подвижности.

> **Структурной группой Ассура (или группой нулевой подвижности)** называется кинематическая цепь, образованная только подвижными звеньями механизма, подвижность которой (на плоскости и в пространстве) равна нулю (Wгр = 0).
>
> Конечные звенья групп Ассура, входящие в две кинематические пары, из которых одна имеет свободный элемент звена, называются **поводками**.

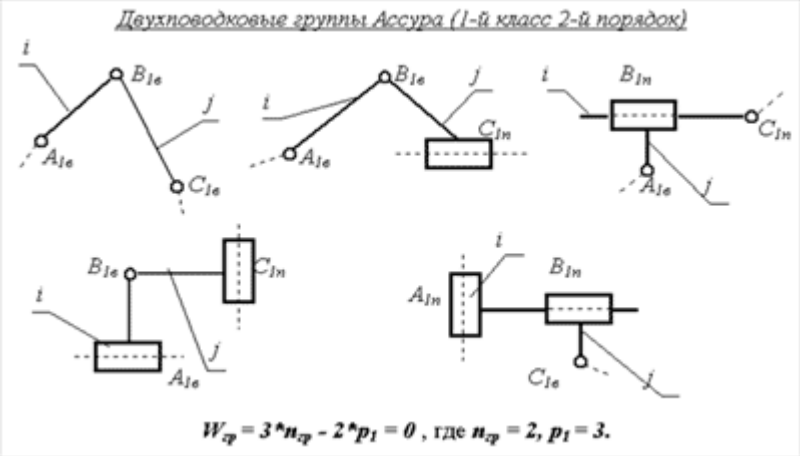
Группы могут быть различной степени сложности. В зависимости от сложности, структурные группы Ассура делятся на классы, а классы в свою очередь делятся на порядки.

Классом структурной группы Ассура называется число кинематических пар, входящих в замкнутый контур, образованный внутренними кинематическими парами группы (И.И. Артоболевского).

Порядок группы определяется числом внешних элементов кинематических пар, которыми группа присоединяется к имеющемуся механизму: первая группа присоединяется к первичному механизму, каждая последующая – к полученному механизму (при этом нельзя присоединять группы к одному звену).

Механизмы классифицируются по степени сложности групп входящих в их состав. *Класс и порядок механизма* определяется классом и порядком наиболее сложной из входящих в него групп.

Особенность структурных групп Ассура - их статическая определимость. Если группу Ассура свободными элементами звеньев присоединить к стойке, то образуется статически определимая *ферма*. Используя группы Ассура удобно проводить структурный, кинематический и силовой анализ механизмов. Наиболее широко применяются простые рычажные механизмы, состоящие из групп Ассура 1-го класса 2-го порядка. Число разновидностей таких групп для плоских механизмов с низшими парами невелико - их всего пять (см. рис. снизу).



При *структурном синтезе механизма* по Ассуру (рис. 1.3) к выбранным первичным механизмам с заданной подвижностью W0 последовательно присоединяются структурные группы с нулевой подвижностью. Полученный таким образом механизм обладает рациональной структурой, т.е. не содержит избыточных связей и подвижностей.

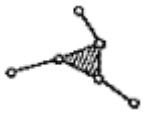
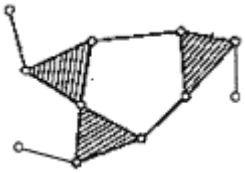
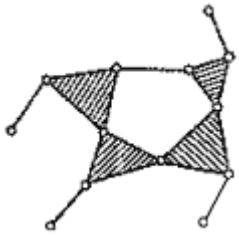
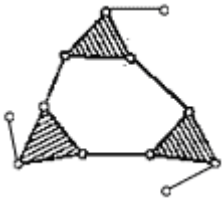
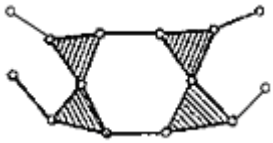
Структурному анализу по Ассуру можно подвергать только механизмы, не содержащие избыточных связей и подвижностей. Поэтому перед проведением структурного анализа необходимо устранить избыточные связи и выявить местные подвижности. Затем необходимо выбрать первичные механизмы и, начиная со звеньев наиболее удаленных от первичных, выделять из состава механизма структурные группы нулевой подвижности (рис. 1.3). При этом необходимо следить, чтобы звенья, остающиеся в механизме, не теряли связи с первичными механизмами.

Ассур разработал структурную классификацию для плоских рычажных шарнирных механизмов (т.е. для механизмов только с вращательными парами). В дальнейшем Артоболевский И.И. усовершенствовал и дополнил эту классификацию, распространив ее на плоские механизмы и с поступательными парами. При этом были изменены и принципы

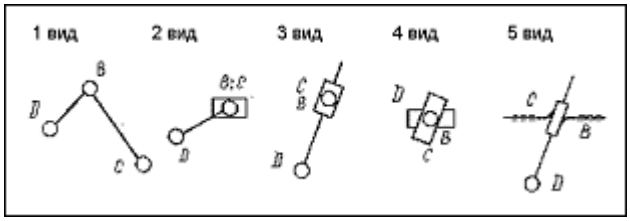
классификации. В плоских механизмах группами являются кинематические цепи с низшими парами, которые удовлетворяют следующему условию:

Решения этого уравнения в целых числах определяют параметры групп Ассура.

Таблица 2: Классификация структурных групп Ассура II-VI классов

Группа	Порядок			
	2-й	**3**-й	**4**-й	**5**-й
II класса		—	—	—
III класса	—			
IV класса			—	—
V класса	—			—
VI класса	—			—

Структурные группы 2-го класса (обычно называемые *двухповодковыми группами* Ассура) дополнительно классифицируются по видам, различающихся сочетанием вращательных и поступательных пар (рис. ниже):



Структурный анализ механизма проводится путем расчленения его на структурные группы и первичные механизмы в порядке, обратном образованию механизма.

2. Проект программной системы

2.1 Описание проблемы

Основной задачей данного проекта является разработка программной системы, предоставляющей пользователю удобные средства визуального геометрического проектирования пространственных механизмов, а так же возможностей решения проблем анализа структуры механизма, и выполнения кинематических расчётов.

При создании такой программной системы, первостепенной проблемой является проектирование инструментов и приёмов конструирования и редактирования пространственных механизмов в виртуальной среде программы. Другой, не менее важной проблемой, является разработка структур данных, наиболее оптимально отражающих структуру механизмов, и обеспечивающих возможность выполнения кинематических расчетов.

В связи с развитием основной задачи по созданию системы моделирования пространственных механизмов возникает ред второстепенных задач по разработки вспомогательных приложений:

- Создание базы данных для хранения механизмов;
- Создание сервиса, выполняющего ряд запросов и обеспечивающего доступ клиентских приложений к базе данных;
- Создание сайта, позволяющего быстро и наглядно получить информацию о любом имеющемся в базе данных механизме, а так же выполнить задачи по администрированию БД;

Следует отметить, что основная задача по созданию системы моделирования на данный момент является частично решённой, в то время как остальные (второстепенные) задачи лишь планируются.

Основная проблема и связанные с ней требования к программной системе будут более подробно рассмотрены в следующем разделе.

2.2 Требования к системе

Как уже говорилось в предыдущем пункте, основной проблемой при создании системы моделирования механизмов является проектирование программного инструментария для конструирования и редактирования механизмов. Поэтому, в первую очередь, определим требования, связанные с основной функциональностью программы – редактированием механизмов. В связи с этим, программа должна выполнять следующие функции:

- **Визуальное отображение 3-х мерного аналога структурной схемы механизма.** Это требование подразумевает наличие средств рисования пространственной схемы механизма в нескольких проекциях с использованием возможностей 3D-графики. Изображение схемы механизма должно в реальном масштабе времени отображать изменение всех параметров механизма и его структуры;
- **Визуальное представление структуры пространственного механизма в виде дерева.** Корневым узлом дерева является сам механизм, его дочерними узлами -

группы кинематических пар. В свою очередь каждая пара включает, в качестве подузлов, те звенья, которые к ней присоединены. Таким образом, пользователь сможет наглядно представить полную структуру механизма;

- **Редактирование параметров любых объектов механизма и их настройка.** В этом требовании имеется в виду необходимость наличия инструмента выбора любого объекта механизма, отображения его параметров и изменения любого доступного параметра. Редактирование параметров механизма должно сопровождаться адекватным изменением его визуального представления;
- **Последовательная сборка механизма из предоставляемых типов звеньев и элементов кинематических пар.** Здесь имеется в виду, что программа должна предоставлять пользователю средства интерактивной сборки механизма, посредством выбора структурных объектов механизма и присоединения их к имеющемуся механизму. Присоединение структурных объектов к механизму должно допускать только корректную сборку механизма, без нарушения правил его внутреннего строения;
- **Возможности сохранения и загрузки модели построенного механизма.** Программа должна позволять сохранять созданные механизмы в специальном файле, а так же загружать их из файлов совместимого формата.

Это требования к основным функциональным возможностям программы. Для того чтобы более конкретно разобраться с требованиями, рассмотрим каждое из них более подробно в отдельности.

Для визуального отображения 3-х мерного аналога структурной схемы механизма необходимо решить следующие задачи:

- **Разработка геометрических (полигональных) моделей структурных объектов механизма**, которые будут являться заменой соответствующим плоским обозначениям объектов структуры механизма (см. таблица 1);
- **Проектирование структур данных геометрических объектов сцены**, обеспечивающих выполнение рендеринга объектов, основных операций над ними и настройки визуального представления;
- **Построение подсистемы видов**, отображающих проекции пространственного механизма, с использованием средств 3D-графики, выполняющих операции пользователя над объектами и обеспечивающих индивидуальную настройку каждого вида;
- **Обеспечение возможности настройки визуальных параметров** отображаемых геометрических объектов, таких как цвета и некоторые геометрические размеры.

Кроме визуального геометрического представления механизма, программа должна отображать его структуру в виде дерева (как минимум с пятью уровнями вложенности), содержащего различные типы узлов:

1. **узел механизма** – корневой узел;
2. **узлы группы кинематических пар** – дочерние узлы механизма;
3. **узлы кинематических пар** – являются дочерними по отношению к узлам групп;
4. **узлы звеньев** – дочерние по отношению к узлам присоединённых пар;
5. **узлы подзвеньев** – дочерние по отношению к звеньям, которые они составляют;
6. **узлы геометрических точек** – дочерние по отношению к содержащим их подзвеньям;

Для дерева структуры механизма должны существовать средства его редактирования, выполняющие следующие основные операции:

- **Выбор узла дерева** с автоматическим выделением этого объекта на видах структурной схемы и отображением параметров этого объекта;
- **Полный обход дерева** – перемещение “вниз” или “вверх” по узлам дерева;
- **Удаление произвольного узла дерева** с сохранением корректности структуры механизма;
- **Создание групп** кинематических пар механизма;
- **Выбор текущей для редактирования группы** механизма (той группы, в которую будут добавляться по умолчанию все создаваемые пары);
- **Перемещение кинематических пар** из одной группы в другую;

Программа должна иметь список всех объектов с редактируемыми параметрами и компонент выбора активного в редакторе параметров объекта. Функциональный компонент программы для отображения и редактирования свойств объектов должен иметь возможности отображения и изменения параметров следующих типов:

- **Строковые**;
- **Целочисленные**;
- **Вещественные числовые**;
- **Логические** (True/False);
- **Перечисления**;
- **Цветовые**;
- **Поля объектов** внутренних классов системы;

Для сборки модели механизма, пользователю должны предоставляться следующие структурные объекты:

- **Звенья** следующих типов:
  - **Крепление** – представляет закреплённое звено стойки;
  - **Линейное звено (ломаная)** – звено, состоящее из линейных цилиндров, соединяющих узловые точки звена;
  - **Контурное звено** – частный случай линейного звена, но с соединением начальной и конечной точек;
  - **Ползунок** – звено ползунка, входящее в поступательную или цилиндрическую кинематическую пару;
- **Элементы** следующих кинематических пар:
  - **Вращательная**;
  - **Сферическая**;
  - **Поступательная**;
  - **Цилиндрическая**;

Перед созданием звена, должна обязательно осуществляться привязка к уже существующей кинематической паре. В свою очередь элемент какой либо пары должен присоединяться к одной из узловых точек заранее созданного звена. Для того, чтобы начать создание нового механизма необходимо разрешить создание звена крепления без присоединения к паре (поскольку в механизме может не существовать ни одной пары). Это можно реализовать, как создание звена крепления вместе со сферической или вращательной парой.

Приведённый здесь список требований к программе нельзя считать полным или законченным, поскольку разработка этой программы предусматривает добавление к ней в дальнейшем возможностей расчета и клиент-серверных расширений.

2.3 Средства разработки

Язык программирования, на котором будет реализована система, заслуживает большого внимания. Исследования показали, что выбор языка программирования несколькими способами влияет на производительность труда программистов и качество создаваемого ими кода [12]. Выбор делался из следующих уже знакомых языков:

- **Visual C++**;
- **Visual C#**;
- **Object Pascal**;

Object Pascal является объектно-ориентированным языком, который, до недавнего времени, использовался только для разработки Win32 приложений. Для разработки приложений на этом языке используется среда визуального программирования **Borland Delphi** и библиотека визуальных компонент **VCL**. С выходом версии **Delphi 2005** стала возможным разработка приложений для платформы Microsoft **.NET Framework** непосредственно на Object Pascal с использованием VCL.

Концепция **Visual C#** разработана для одновременного использования объектно-ориентированного языка общего назначения, а так же среды программирования и исполнения приложений “**Microsoft .NET**”. Этот язык поддерживает все современные методики ООП, но, тем не менее, используется исключительно для разработки приложений под **.NET**.

Язык **Visual C++** на данный момент является самым широко используемым языком программирования как для разработки Win32 приложений, так и для создания программ для среды .NET Framework. Компания Microsoft разработала расширения этого языка - **Managed Extensions for Visual C++**, которые позволяют писать полноценные .NET.

С выходом новой версии платформы **.NET 2.0** синтаксис языка Visual C++ был значительно обновлён. Это обновление языка, называемое **C++/CLI** (*Common Language Infrastracture*) было сделано для обеспечения возможности объединения управляемого (*managed*) и неуправляемого (*native*) исходного кода в единые модули, что позволило создавать так называемые **mixed mode приложения**. Таким образом возможности языка C++ .NET даже превосходят возможности C#, сохраняя всё старое и добавляя новые возможности среды .NET.

.NET Framework – это платформа для построения и исполнения приложений. Её основные компоненты – общеязыковая исполняющая среда CLR (*Common Language Runtime*) и библиотека классов FCL (*Framework Classes Library*). [6]

Среда CLR абстрагирует сервисы ОС и служит механизмом для исполнения управляемых приложений (*managed applications*), любое действие которых должно получить одобрение со стороны CLR. Программы, предназначенные для работы с CLR, написаны на промежуточном языке **CIL** (*Common Intermediate Language*), представляющем объектно-ориентированный аналог ассемблера.

Перед исполнением, среда CLR выполняет, так называемую, компиляцию по требованию **JIT** (*Just-In-Time*) отдельных участков кода CIL и переводит их в машинный код. В связи с таким подходом к исполнению приложений можно говорить о виртуальной машине .NET. Несмотря на накладные расходы, связанные с JIT-компиляцией, приложения .NET работают

достаточно быстро (примерно 60% от производительности неуправляемого аналога), благодаря высокой оптимизации этого процесса.

Библиотека FCL представляет набор объектно-ориентированных API, к которым обращаются управляемые приложения. Инструментарий, предлагаемый этой библиотекой, настолько широк, что позволяет полностью отказаться от использования вспомогательных библиотек, таких как Windows API, MFC, ATL, COM. Средства этой библиотеки позволяют разрабатывать следующие типы клиентских и серверных приложений [7]:

- **Консольные приложения Windows**;
- **Windows Forms** – оконные приложения для Windows;
- **Службы Windows** – приложения управляемые диспетчером SCM;
- **Библиотеки компонент** – удобные в использовании с другими приложениями автономные модули, содержащие реализацию классов пользователя;
- **Web Forms** – веб сайты, описанные на языке HTML с северными сценариями;
- **Web Services XML** – серверные программы, предоставляющие некоторые методы, к которым можно легко обратиться через Интернет;

Microsoft .NET Framework предоставляет разработчикам массу преимуществ, перечислю лишь некоторые из них [7]:

- Единая программная модель;
- Отсутствие проблем с версиями;
- Упрощение процесса разработки и модели программирования;
- Возможность работы на нескольких платформах;
- Интеграция языков программирования и возможность их совместного использования;
- Автоматическое управление памятью (сборка мусора);
- Проверка безопасности типов;
- Развитая поддержка отладки;
- Управление безопасностью приложений;

Из всего перечисленного, необходимо отметить, наличие встроенного сборщика мусора, выполняющего автоматическое управление памятью. Среда CLR автоматически отслеживает использования ресурсов, гарантируя, что не произойдёт их утечки. По сути она исключает возможность явного освобождения памяти. При разработке нашего приложения, будет создана сложная система классов, для управления объектами которой, очевидно потребуется сборщик мусора. Наличие же встроенного сборщика мусора в .NET Framework является несомненным плюсом для использования этой среды при разработке приложения.

Важным требованием к языку и среде разработки программы редактирования механизмов является возможность использования современных библиотек и средств программирования 3-х мерной компьютерной графики, что необходимо для визуализации структурных схем механизмов. В данный момент самыми широко используемыми библиотеками для программирования 3D графики являются: **OpenGL** и **DirectX**.

С выходом новой версии **Microsoft DirectX 9.0** стало возможным использование этой библиотеки в среде .NET. Microsoft разработал специальное расширение существующей библиотеки DirectX “слоем” управляемых API, называемых **Managed DirectX**. Эти библиотеки открывают доступ к функциям библиотеки DirectX из управляемого кода.

Использование управляемого DirectX даёт разработчикам следующие преимущества:

- Поддержка всех .NET-совместимых языков программирования:
- Возможности использования всей мощи библиотеки классов FCL;
- Удобства программирования, связанные со строгим именованием типов;
- Сокращение исходного кода на 15 - 20% по сравнению с неуправляемым аналогом [16];
- Поддержка новейших аппаратных средств;

Минусом же является только снижение производительности рендеринга, из-за исполнения CIL-кода. Тем не менее, современные аппаратные ускорители 3D-графики компенсируют этот недостаток. Производительность рендеринга 3D-графики в управляемом коде можно повысить до 98% от производительности неуправляемого аналога [16].

Исходя из всего выше сказанного, я принял решение об использовании языка программирования Visual C++ .NET для создания приложения под платформу .NET Framework. Если некоторые компоненты программы удобнее писать на Visual C#, то их можно создавать в виде отдельных модулей. Для программирования 3D графики будет использоваться библиотека Managed DirectX 9.0.

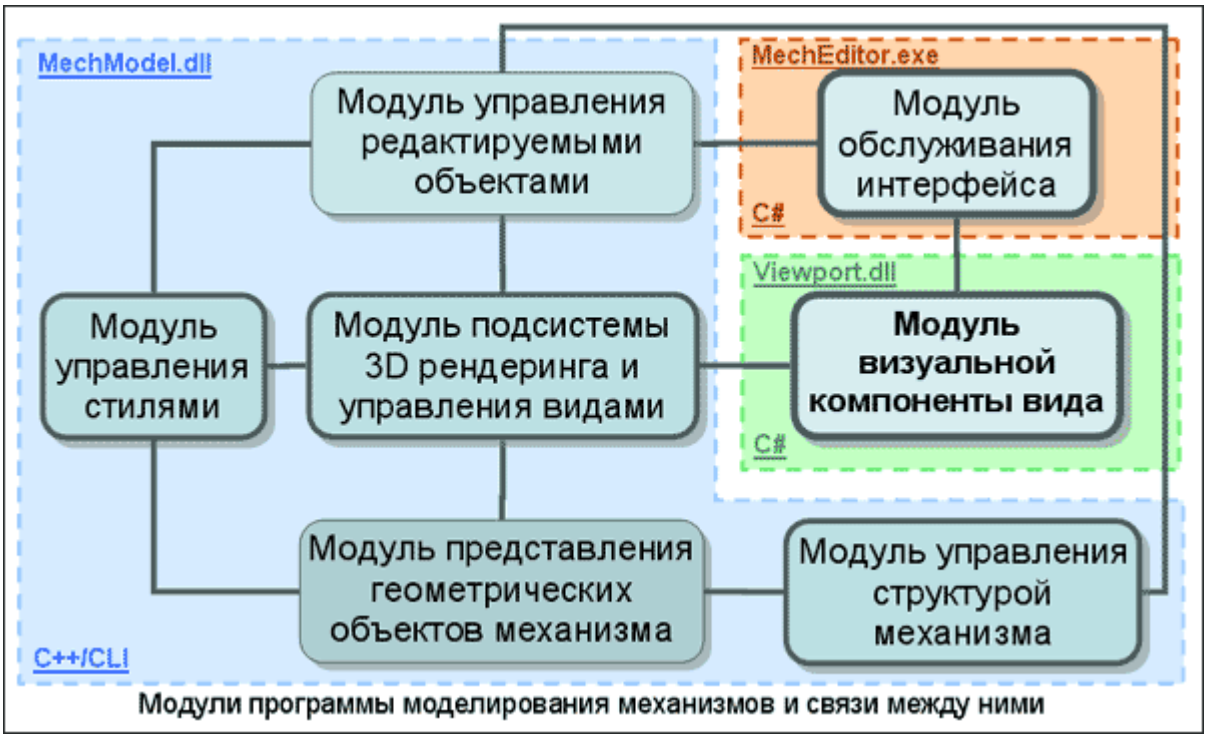
Кроме того, для визуализации 3-х мерных аналогов элементов структурной схемы механизма, необходимо спроектировать их полигональные модели, а затем сохранить их в формате X-файла (для загрузки с помощью функций DirectX). Для этого необходимо использовать среду 3-х мерного моделирования, которая имеет возможности сохранения построенных моделей в X-файлах. Для этого я предлагаю использовать систему компании **Paradox – Maxon Cinema 4D**. Эта система имеет удобный инструментарий для создания 3-х мерных моделей и имеет возможности их сохранения и загрузки в X-файлах.

В качестве итога ко всему сказанному, ниже приводится список всех использыемых программных средств разработки для данного проекта:

- Выбранная платформа: Microsoft .NET Framework 2.0;
- Языки программирования:
  - **Microsoft Visual C++/CLI** - язык для программирования ядра системы;
  - **Microsoft Visual C#** - язык для программирования интерфейса системы;
- Используемые библиотеки классов:
  - **FCL** - основная библиотека классов;
  - **Managed DirectX 9** - библиотека программирования 3D-графики;
- Среда разработки: **Microsoft Visual Studio 2005**;
- Система 3-х мерного иоделирования: Paradox Maxon Cinema 4D.

2.4 Модули системы

Процесс разработки программы моделирования механизмов (**Mechanics Editor**) начнём с разбиения её на несколько логических модулей. Диаграмма взаимодействия этих модулей представлена на рис. снизу.



Модули программы моделирования механизмов и связи между ними

На приведённой диаграмме сплошными линиями обозначено отношение ассоциации между модулями в смысле взаимодействия этих модулей между собой. Кроме того, на диаграмме выделены множества логических модулей составляющие отдельные физические модули: dll или exe. Наименование и язык исходного кода физических модулей указаны в левом верхнем и левом нижнем углу групп соответственно.

Далее я привожу развёрнутое описание функций каждого из модулей:

- **Модуль обслуживания интерфейса** – представляет собой все классы форм программы, выполняющих обработку событий действий пользователя;
- **Модуль управления редактируемыми объектами** – объединяет базовый класс редактируемых объектов программы и производный класс управления этими объектами. Этот класс содержит список всех редактируемых объектов и содержит методы доступа и управления ими;
- **Модуль управления стилями** – содержит класс настроек цветового оформления и типовых параметров различных объектов. Изменение, какого либо стиля отражается на всех соответствующих объектах за счёт обработки происходящего события;
- **Модуль подсистемы 3D-рендеринга и управления видами** – представляет класс, реализующий работу вида, отображающего проекцию структурной схемы механизма. Этот класс также реализует операции работы с геометрическими объектами сцены, такие, как выделение мышью, перемещение, вращение, масштабирование;
- **Модуль представления геометрических объектов механизма** – объединяет все классы, представляющие каждый из геометрических объектов сцены и реализующие операции над ними;

- **Модуль управления структурой механизма** – объединяет все классы, реализующие представление типовых объектов структуры механизма. Именно эти классы и позволяют построить модель механизма.

Как видно из построенной диаграммы связей между логическими модулями, модуль управления структурой механизма связан всего лишь с двумя другими модулями, хотя и включает в себя основную часть классов программы. Это позволяет выполнить разработку программы в два этапа:

1. Разработать и отладить работу программы без модуля управления структурой механизма – решающую задачи визуализации и редактирования геом. объектов;
2. Добавить модуль управления структурой механизма и выполнить отладку.

2.5 Диаграммы классов

На рисунке ниже приведена диаграмма классов, обслуживающих визуализацию данных. Обозначения на диаграмме соответствуют нотации языка **UML**, хотя и имеют нестандартное графическое оформление. Стрелками обозначается *отношение обобщения*, а квадратиками - *отношение агрегации* и *композиции* [14]. Под именем каждого класса приведено его краткое описание.

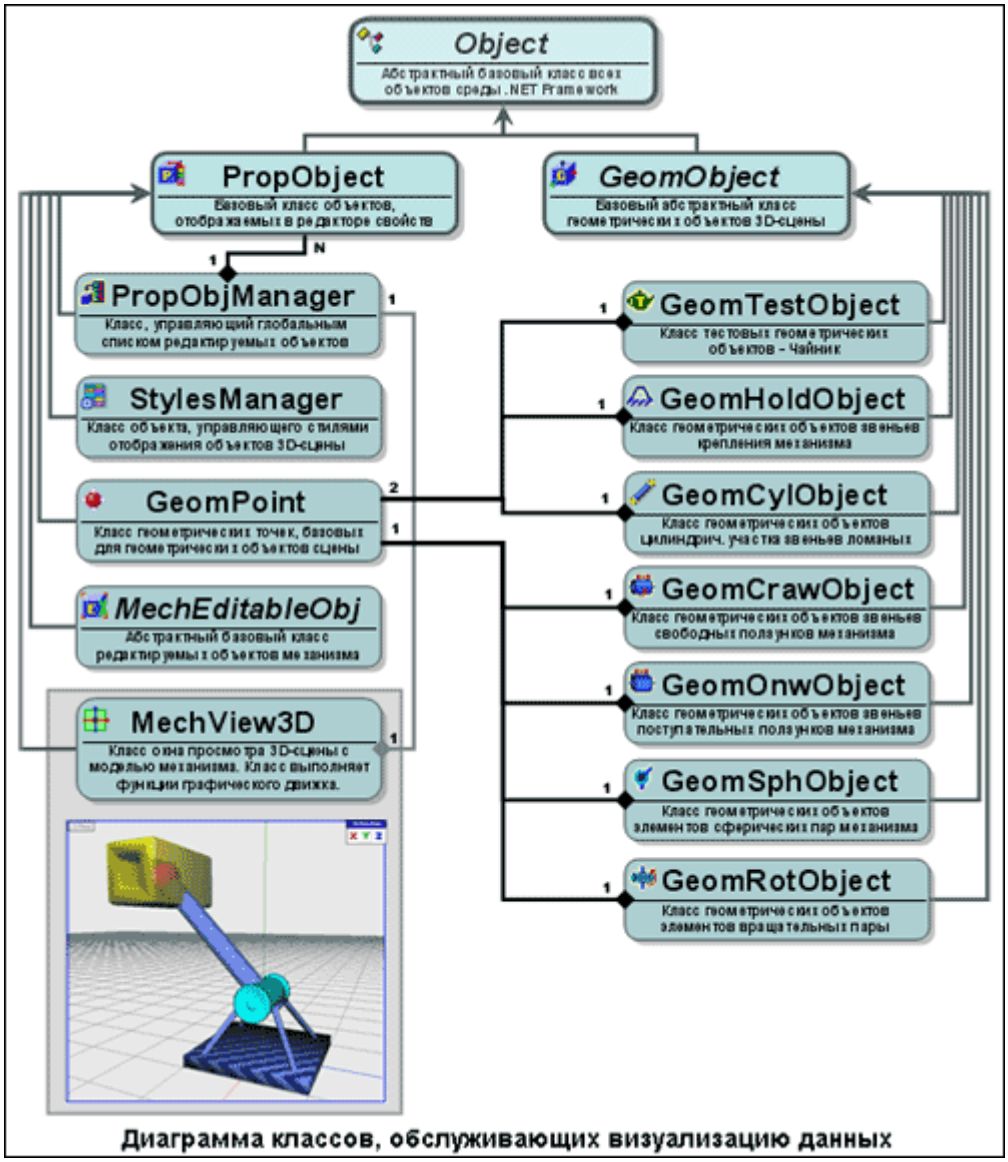
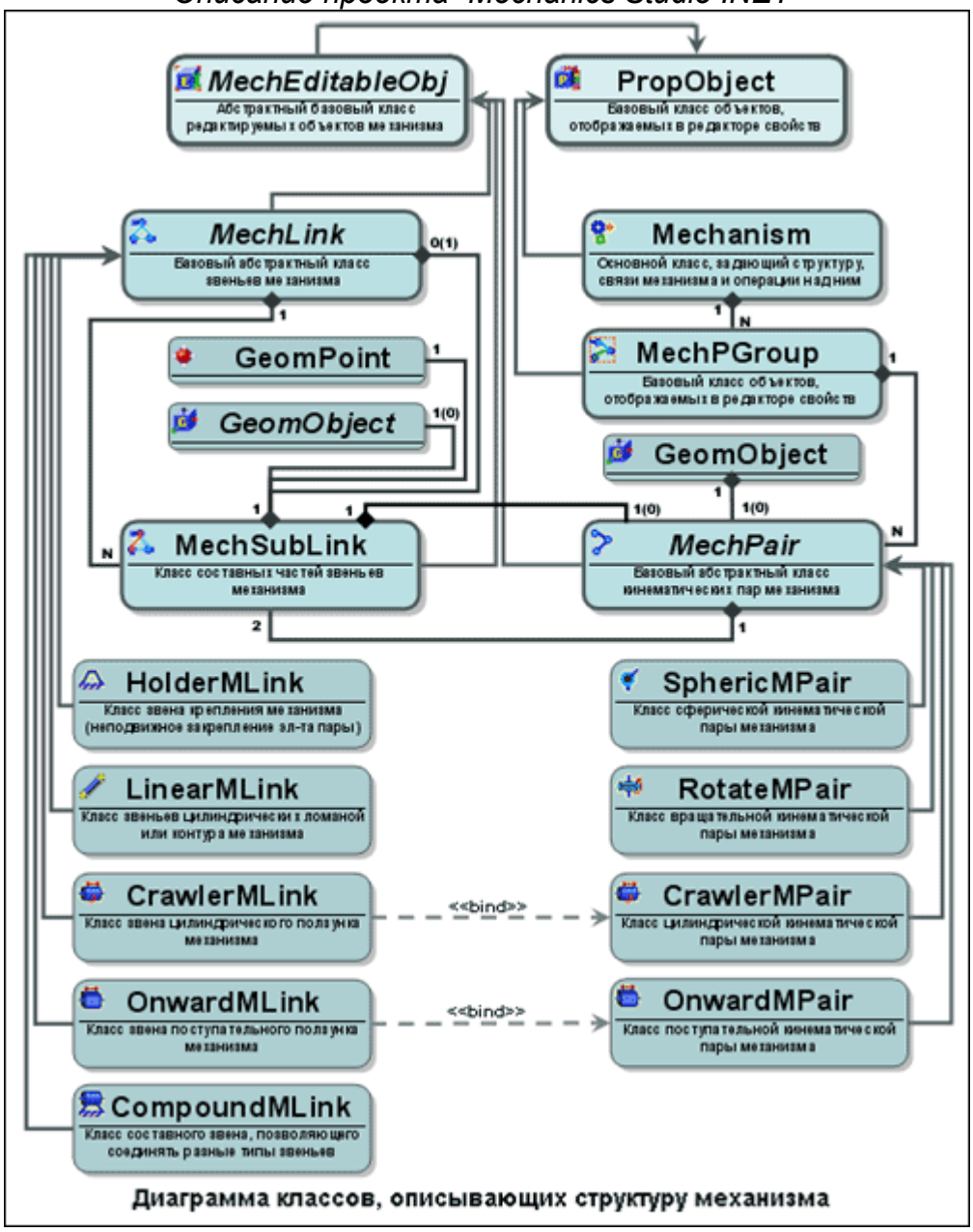


Диаграмма классов, обслуживающих визуализацию данных

На следующем рисунке приведена диаграмма классов, описывающих структуру механизма. Необходимо отметить, что на диаграмме отражены только ключевые отношения агрегации, отражающие структуру механизма. Кроме того, на диаграмме присутствует *отношение зависимости* между классами (обозначаемое пунктирной стрелкой от (зависимого) *класса-клиента* к (независимому) *классу-источнику*) [14].



Ниже приведена таблица дающая краткое описание каждого класса, представленного на диаграмме и разбивающая классы на логические модули:

Таблица 3: Краткое описание основных функциональных классов программы

№	Название класса	Краткое описание
	Модуль управления редактируемыми объектами:	
1	**PropObject**	Данный класс является базовым классом всех объектов, которые могут отображаться в редакторе свойств. Класс реализует общие для таких объектов свойства и методы, связанные с идентификацией в списке редактируемых объектов;
2	**PropObjManager**	Класс объекта, управляющего списком всех редактируемых объектов и выполняющего основные операции над этими объектами. Кроме того, класс реализует некоторые операции по редактированию структуры механизма;

Модуль управления стилями отображения объектов:		
3	**StylesManager**	Класс объекта, управляющего стилями отображения видов и структурных схем механизмов в видах. Класс содержит описание событий, вызываемых обработчиками изменения свойств стилей;
Модуль подсистемы 3D-рендеринга и управления видами:		
4	**MechView3D**	Класс вида – выполняет настройку вида и его рендеринг на указанную панель. Класс работает со списком геометрических объектов и реализует операции выполнение операций редактирования этих объектов – такие как выделение мышью, перемещение, вращение, масштабирование;
Модуль представления геометрических объектов механизма:		
5	**GeomPoint**	Класс точек, определяющих положение геометрических объектов. Класс точек имеет методы рендеринга в проекциях механизма. Класс геометрической точки так же имеет методы, позволяющие выполнять её выделение и перемещение в пространстве;
6	**GeomObject**	Этот абстрактный класс является базовым для всех геометрических объектов сцены. Класс содержит свойства настройки таких объектов, методы рендеринга и методы их редактирования, такие как перемещение, вращение, масштабирование;
7	**GeomTestObject**	Класс тестового геометрического объекта (чайник).
8	**GeomHoldObject**	Класс геометрических объектов звеньев крепления;
9	**GeomCylObject**	Класс геометрических объектов цилиндрических участков звеньев ломаных;
10	**GeomCrawObject**	Класс геометрических объектов звеньев цилиндрических ползунков;
11	**GeomOnwObject**	Класс геометрических объектов звеньев поступательных ползунков;
12	**GeomSphObject**	Класс геометрических объектов элементов сферических пар;
13	**GeomRotObject**	Класс геометрических объектов элементов вращательных пар;
Модуль управления структурой механизма:		
14	**Mechanism**	Класс механизма является высшим звеном иерархической инкапсуляции классов, описывающих структуру механизма. Класс реализует итераторы кК по группам механизма, так и по всем парам и звеньям. Кроме того, в классе реализованы основные методы, выполняющие конструирование механизма и выполняющие задачи пользователя;
15	**MechPGroup**	Класс группы кинематических пар механизма. Класс группы реализует итератор по парам этой группы, а так же методы добавления, изменения и удаления пар из группы;
16	**MechEditableObj**	Абстрактный класс, базовый всех объектов механизма, позволяющих выполнять над ними операции геометрического редактирования;
17	**MechSubLink**	Класс подзвена, являющегося составной частью звена. Содержит информацию о структурной точке звена, геометрическом объекте части звена или других звеньях, являющихся частью составного звена. Подзвено также содержит указатель на присоединённую пару;
18	**MechLink**	Абстрактный класс звена, базовый для различных типов звеньев. Содержит общие для всех типов звеньев методы и свойства,

		унифицирующие работу со звеньями;
19	**HolderMLink**	Класс звена крепления механизма (для неподвижного закрепления элемента кинематической пары);
20	**LinearMLink**	Класс линейных звеньев - ломаной или контура
21	**CrawlerMLink**	Класс звена цилиндрического ползунка механизма
22	**OnwardMLink**	Класс звена поступательного ползунка механизма
23	**CompoundMLink**	Класс составного звена, позволяющего соединять разные типы звеньев;
24	**MechPair**	Абстрактный класс, являющийся базовым для различных типов кинематических пар. Класс реализует методы и свойства, унифицирующие работу с кинематическими парами;
25	**SphericMPair**	Класс сферической кинематической пары механизма;
26	**RotateMPair**	Класс вращательной кинематической пары механизма;
27	**CrawlerMPair**	Класс цилиндрической кинематической пары механизма;
28	**OnwardMPair**	Класс поступательной кинематической пары механизма;

2.6 Структура данных механизма

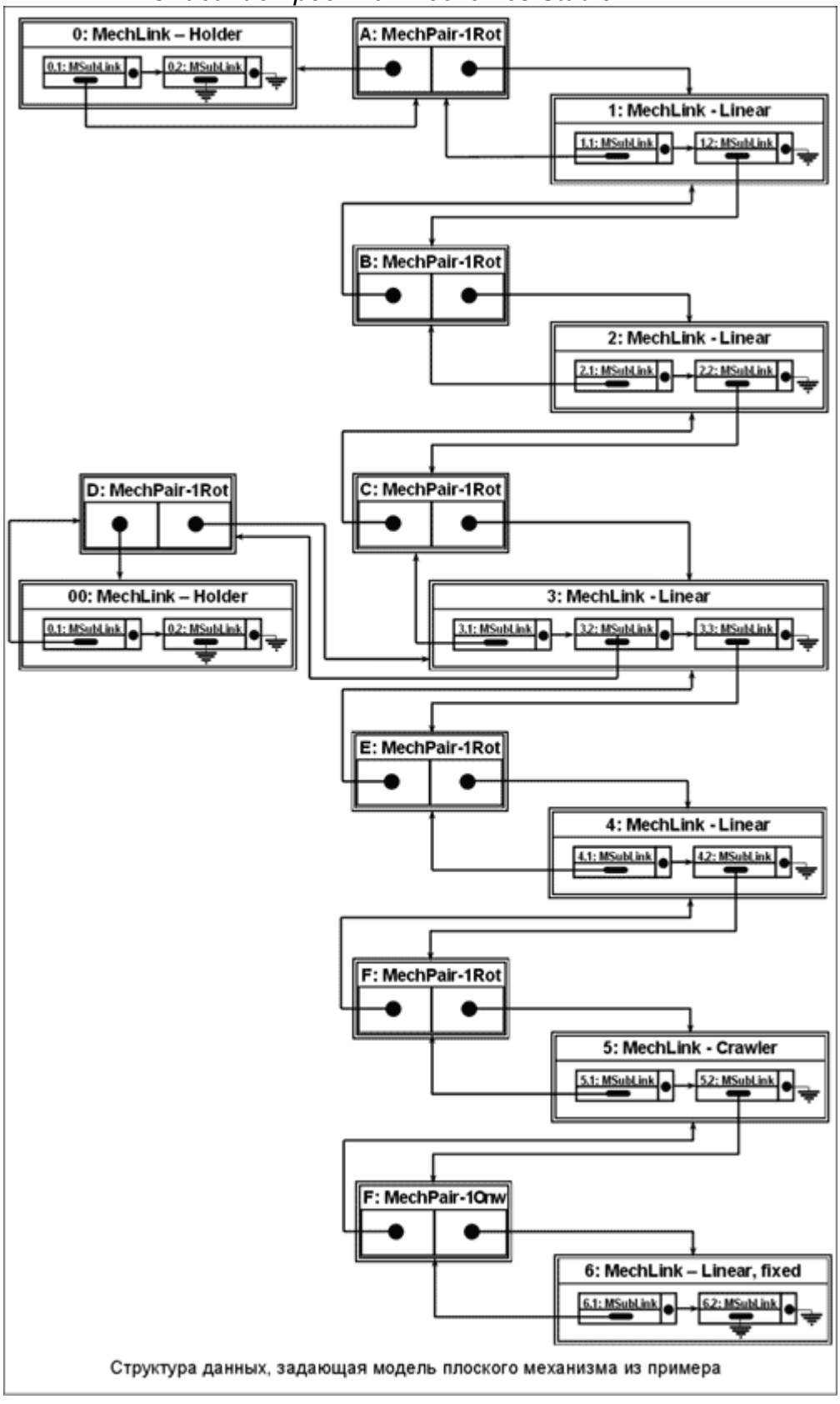
Рассмотрим схему построения структуры данных, которая задаёт модель пространственного механизма, на конкретно взятом примере – рис. справа.

Пример плоского механизма

Далее будем обозначать элементы структуры данных следующими идентификаторами:

- Элемент СД, соответствующий звену механизма - **MechLink**;
- Элемент СД, соответствующий отрезку или составной части звена - **MSubLink**;
- Элемент СД, соответствующий кинематической паре механизма - **MechPair**.

Ниже приводится схема связей между основными элементами структуры данных, задающей модель механизма (представленного на рис. выше):



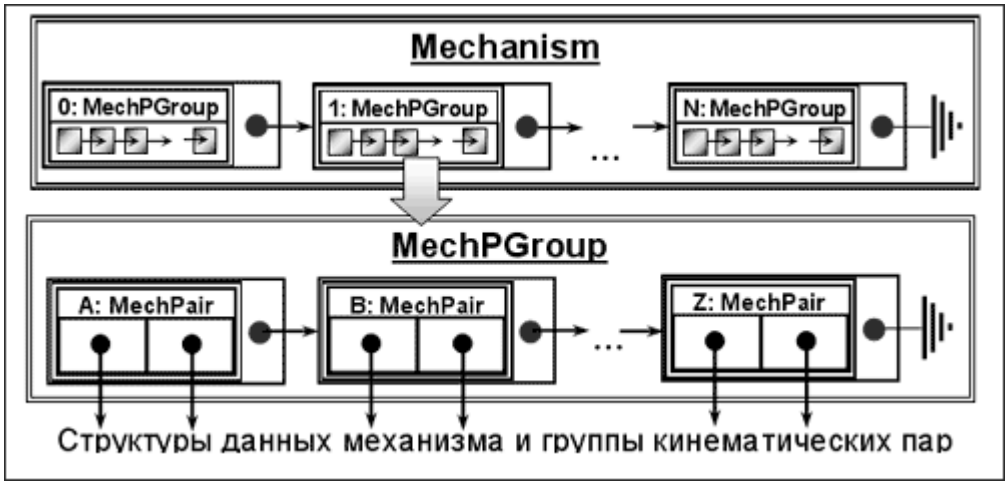
Структура данных, задающая модель плоского механизма из примера

Из представленной схемы видно, что каждый из объектов звена **MechLink** содержит список указателей на его составные элементы – подзвенья **MSubLink**. Каждое подзвено соответствует некоторой структурной точке его звена, и может быть либо доступным для соединения с парой, либо недоступным. Если подзвено соединено с парой, то оно содержит указатель на присоединённую пару.

Каждый объект пары **MechPair** содержит два указателя на присоединённые звенья. В действительности же, для присоединения звена к паре, удобно передавать указатель на объект его подзвена (к которому производится присоединение), а в паре реализовать свойство, дающее указатель на присоединённое звено.

На рис. механизма и рис. структурной схемы каждое звено идентифицируется его номером **0..6**, а элементы кинематических пар - буквой **A..F** той точки, в которой находится этот элемент и условным обозначением типа кинематической пары. Эти обозначения позволяют установить соответствие объектов структуры данных и элементов рисунка структурной схемы.

Структура данных механизма **Mechanism** (представленная на рис. ниже), по своей сути, представляет собой список указателей на объекты групп кинематических пар **MechPGroup**. Каждая группа **MechPGroup**, в свою очередь содержит список указателей подмножество кинематических пар механизма **MechPair**. Основные связи пар с другими объектами структуры данных механизма уже представлены на предыдущей структурной схеме.

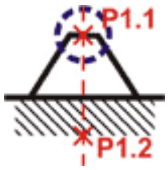
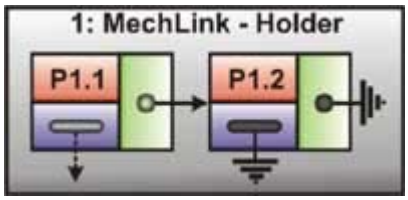
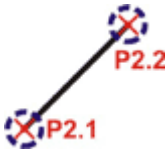
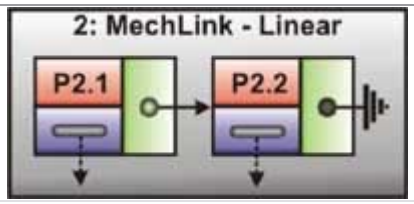
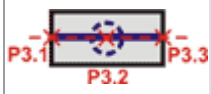
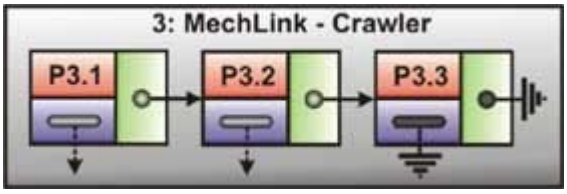
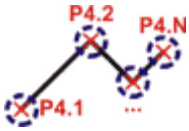
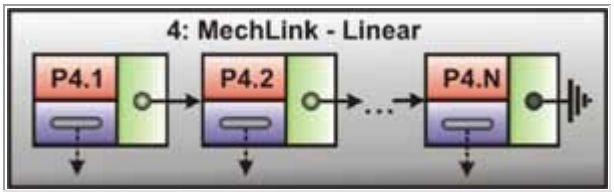
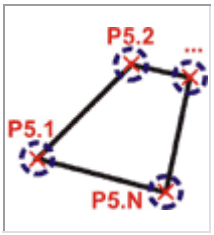
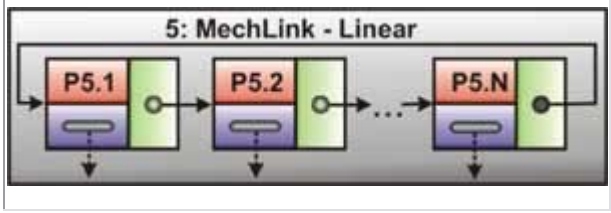
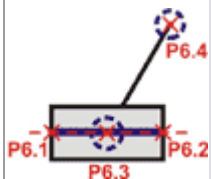
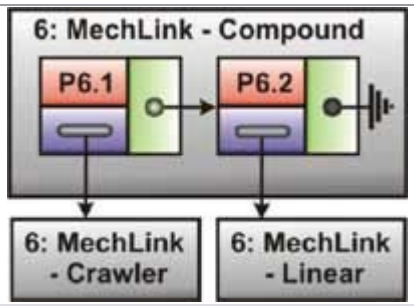


Звено механизма в данном представлении описывается списковой структурой - **MechLink**, каждый элемент списка которой, **SubLink** содержит следующие основные поля данных:

- Указатель на объект структурной точки **GeomPoint**, содержащий координаты этой точки в 3-х мерном пространстве;
- Указатель на объект некоторой кинематической пары **MechPair**, к которой присоединено данное звено механизма. Причём элемент этой пары располагается в точке данного подзвена;
- Указатель на геометрический объект **GeomObject**, определяющий визуальное представление данной части звена в структурной схеме.

Использование списков для представления звеньев механизма позволяет описывать структуру и положение основных типов звеньев, которые часто используются в пространственных механизмах (см. таблица 4).

Таблица 4: Представление основных типов звеньев механизмов с помощью списковых структур данных

№	Название	Рисунок звена	Структура данных	Использование объекта
1	Звено - **крепление**			Для неподвижного закрепления элементов кинематических пар или других звеньев, за счёт создания составного звена;
2	Звено - **линейный отрезок**			Для линейного соединения 2-х элементов кинематических пар;
3	Звено - **ползунок**			В качестве подвижной части, перемещающейся по некоторому другому линейному звену. Всегда соединено с элем-м поступательной кинематической пары;
4	Звено - **ломаная**			Для линейного соединения 3-х и более элементов кинематич. пар;
5	Звено - **контур**			Для линейного соединения в контур 3-х и более элементов кинематических пар;
6	**Составное звено**			Для создания звеньев более сложной структуры, на основе имеющихся звеньев 1-5, путём их объединения;

На рисунках звеньев красными крестиками отмечены точки, определяющие структуру и положение звена. Каждой такой точке соответствует один элемент списка, названный *подзвеном*. Синие пунктирные кружки, вокруг некоторых точек, обозначают места возможной *привязки элементов кинематических пар*. В элементах списка привязке элемента кинематической пары соответствует указатель на соответствующий объект пары.

В последнем случае, для составного звена, указатели элементов списка указывают не на объекты элементов пар, а на другие звенья, объединяемые составным звеном в одно целое.

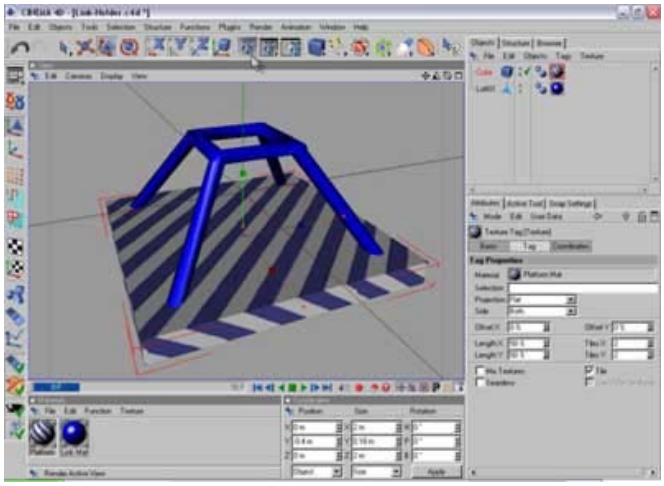
2.7 Аналог структурной схемы механизма

В разделе, описывающем предметную область проекта - теорию механизмов и машин говорилось, что для изображения механизма на чертеже используют так называемые структурные схемы, в которых применяются общепринятые условные обозначения частей механизма. Система моделирования пространственных механизмов, несомненно, должна уметь отображать механизм с помощью структурной схемы.

Поскольку для визуализации пространственной структурной схемы механизма было решено использовать средства 3D-графики, то важно разработать модели составных объектов, из которых будет собираться эта схема так, чтобы они соответствовали общепринятым плоским обозначениям (см. таблица 1).

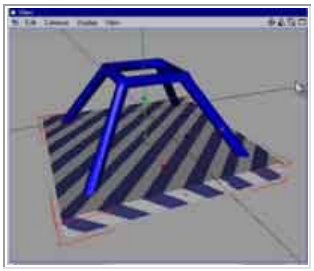
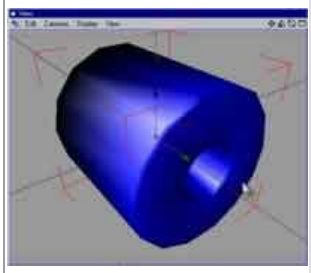
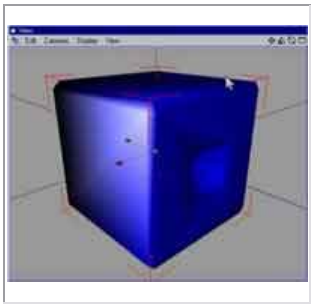
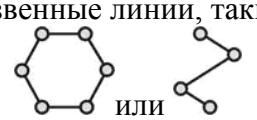
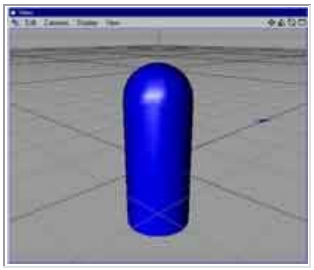
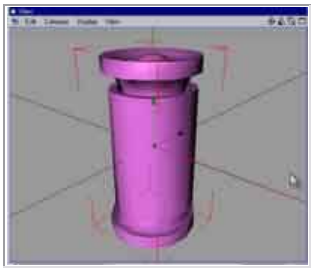
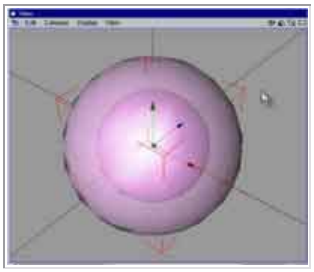
Как уже говорилось в разделе "Средства разработки", важно использовать средства 3-х мерного моделирования, позволяющие сохранять построенные модели непосредственно в формате X-файлов. Это связано с тем, что мы решили пользоваться графической библиотекой DirectX, а она работает именно с X-файлами.

Поэтому для создания 3-х мерных моделей было решено использовать систему фирмы **Paradox – Maxon Cinema 4D**. Эта система имеет удобные средства геометрического моделирования (позволяет точно задавать многие параметры модели) и имеет возможности сохранения моделей в X-файле.



Все модели создавались с точно определёнными размерами, так чтобы при дальнейшем масштабировании этих моделей средствами функций DirectX, можно было точно рассчитать их новые размеры. Далее приведена таблица созданных моделей, с указанием всей необходимой для разработчика информацией о них:

Таблица 5: 3-х мерные полигональные модели составных объектов структурной схемы механизма

№	X-файл модели	Назначение модели	Изображение	Размеры
1	**Link-Holder.x** **Holder-Hatch.jpg**	**Модель звена крепления**: Используется для обозначения стоек механизма, является аналогом стандартного обозначения: 		Size X = 2, Size Y = 1, Size Z = 2
2	**Link-Crawler.x**	**Модель цилиндрического ползунка**: используется для обозначения звена ползунка, входящего в цилиндрическую пару, является аналогом стандартного обозначения: 		Outer Diam = 1, Inner Diam = 0.45, Size Y = 1
3	**Link-Onward.x**	**Модель поступательного ползунка**: используется для обозначения звена ползунка, входящего в поступательную пару, является аналогом стандартного обозначения: 		Outer Size X,Z = 1, Inner Size X,Z = 0.45, Size Y = 1
4	Модель строится методами DirectX: **Mesh.Cylinder** **Mesh.Sphere**	**Модель части линейного звена**: используется для сборки линейных звеньев, образующих контуры или многозвенные линии, такие как: 		Radius = 1, Height = 1,
5	**Elem-Rotator.x**	**Модель элемента вращательной пары**: используется в качестве аналога принятого плоского обозначения: 		Radius = 1, Height = 2,
6	**Elem-Sphere.x**	**Модель элемента сферической пары**: используется в качестве аналога принятого плоского обозначения: 		Outer Diam = 1, Inner Diam = 0.65

2.8 Программирование 3D-графики на Managed DirectX

2.8.1 Типы данных для выполнения матричных преобразований

В библиотеке **Managed DirectX 9** имеются уже реализованные и удобные в использовании типы данных для работы с матрицами и векторами. Эти типы имеют удобные и быстрые методы выполнения векторных и матричных операций. Поэтому, в данном случае, нет необходимости создавать какие-либо свои типы данных, а нужно использовать уже имеющиеся. Далее я приведу краткое описание этих типов.

Структуры векторов:

- **Двумерные вектора** – структура Vector2:
  `Vector2 pV2 = new Vector2(float X, float Y);` – конструктор создания 2D вектора;
- **Трёхмерные вектора** – структура Vector3:
  `Vector3 pv3 = new Vector3(float X, float Y, float Z);` – создание 3D вектора;
- **Трёхмерные вектора в однородных координатах** – структура Vector4:
  `Vector4 pv4 = new Vector4(float X, float Y, float Z, float W);` – создание 4D вектора;

Вектора имеют поля, через которые можно обращаться к компонентам вектора: **X, Y, Z, W**. Вне зависимости от того, какого типа вектор, он имеет общие для всех векторов методы. Многие методы имеют как обычную, так и статическую реализацию. Я приведу лишь названия основных статических методов, без указания типов передаваемых в качестве параметров векторов:

- `Vector Vector::Add(pVa, pVb);` – сложение двух векторов: pVa + pVb;
- `Vector Vector::Subtract(pVa, pVb);` – вычитание двух векторов: pVa - pVb;
- `Vector Vector::Scale(pV, float K);` – умножение вектора на число: pV·K;
- `Vector4 Vector::Transform(pV, Matrix M);` – умножение вектора на матрицу: pV·M;
- `Vector Vector::Normalize( pV );` – нормализация вектора: pV/|pV|;
- `Vector Vector::Cross(pVa, pVb);` – векторное произведение векторов: [pVa x pVb];
- `float Vector::Dot(pVa, pVb);` – скалярное произведение векторов: (pVa·pVb);
- `float Vector::Length( pV );` v вычисление длины (нормы) вектора: |pV|;

Структура матрицы:

`Matrix pM = new Matrix( );` – конструктор создания 0-й матрицы размерности 4x4;

Доступ к компонентам матрицы осуществляется через поля именуемые как **Mij**, где **i=1..4, j=1..4**. В структуре матрицы так же определены статические методы, позволяющие создавать матрицы, выполняющие стандартные пространственные аффинные преобразования:

- `Matrix pM = Matrix::Translation(float Tx, float Ty, float Tz);` – создание матрицы, выполняющей преобразование смещения на вектор (Tx, Ty, Tz);
- `Matrix pM = Matrix::RotationX[Y,Z]( float Angle );` – создание матрицы, выполняющей преобразование поворота вокруг оси X (Y или Z) на угол Angle (в рад.);
- `Matrix pM = Matrix::RotationAxis( Vector pVAxis, float Angle);` – создание матрицы, выполняющей преобразование поворота вокруг оси pVAxis на угол Angle (в рад.);

- **Matrix** pM = **Matrix::Scaling**(**float** Sx, **float** Sy, **float** Sz); – создание матрицы, выполняющей преобразование масштабирования на скалярные велисины (Sx, Sy, Sz);

Как и для вектора, в структуре матрицы реализованы статические и нестатические методы, выполняющие основные операции над матрицами. Приведу лишь некоторые из статических реализаций этих методов:

- **Matrix Matrix::Add**(pMl, pMr); – сложение двух матриц: pMl+pMr;
- **Matrix Matrix::Multiply**(pMl, pMr); – умножение двух матриц: pMl·pMr;
- **Matrix Matrix::Invert**(pM); – вычисление обратной матрицы: pM^{-1};
- **Matrix Matrix::TransposeMatrix**(pM); – транспонирование матрицы: pM^{T};

2.8.2 Пространственные и проекционные системы координат

Системы координат в DirectX описываются матрицами **Matrix**. В зависимости от использования, системы координат принято называть по разному.

Исходную систему координат, по отношению к которой выполняются все преобразования объектов сцены, принято называть **мировой системой координат**.

Систему координат, связанную с некоторым объектом, и преобразуемую вместе с объектом, называют **объектовой системой координат** (говорят, что объект “вморожен” в систему координат).

Для изменения мировой системы координат в DirectX, необходимо изменить определяющую её матрицу. *Матрица мировой системы* координат хранится в поле объекта **Device** – устройства, выполняющего все графические операции Direct3D:

```
Device→Transform→World;
```

Если мировая система координат (X, Y, Z) преобразована к (Xo, Yo, Zo), то все объекты будут отображаться в новой объектовой системе координат (см. рис.).

Кроме пространственных систем координат, в DirectX есть так же *проекционные системы координат*, которые используются в процессе рендеринга сцены – т.е. при проецировании 3-х мерных объектов на плоскость камеры. Положение и направление камеры определяются *матрицей вида*:

```
device→Transform→View;
```

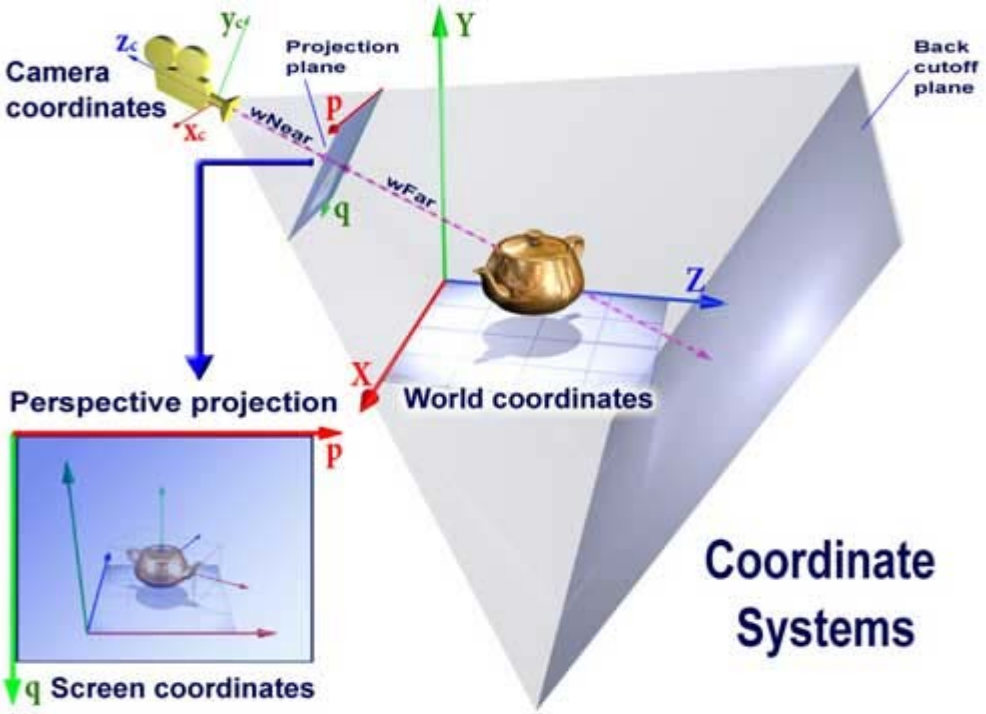
Таким образом, матрица вида определяет положение и ориентацию **системы координат камеры** (Xc, Yc, Zc) (см. рис. ниже - Camera coordinates).

Матрица вида может быть автоматически сформирована путём вызова статического метода *LookAtLH* по передаваемым векторам положения камеры (*CamPosition*), точки фокуса (*camTarget*) и верха камеры (*CamUpVector*):

```
device→Transform→View = Matrix::LookAtLH( Vector3 camPosition, Vector3
                    camTarget, Vector3 camUpVector );
```

Кроме того, для полного определения процесса проецирования, нужно задать параметры используемой нами камеры. Эти параметры хранятся в *матрице проецирования*:

```
device→Transform→Projection;
```



Прежде всего, необходимо определить какой тип проекции будет использоваться:

- **Перспективная проекция** – используется для описания сцены, представленной в виде усечённой пирамиды (*frustrum*), где внутренняя часть пирамиды является просматриваемой областью нашей сцены (см. рис. 2.13). В перспективной проекции все лучи зрения пересекаются в точке расположения камеры. Матрица перспективной проекции создаётся методом *PerspectiveFovLH*:

```
   device→Transform→Projection = Matrix::PerspectiveFovLH( float
viewAngle, float aspectRation, float znearPlane, float zfarPlane );
```

  Два параметра в этой функции – ближняя и дальняя плоскости (*znearPlane, zfarPlane*) описывают пределы этой пирамиды, причём дальняя плоскость является основанием, а ближняя плоскость проходит по месту отсечения вершины пирамиды. Поле зрения камеры определяется углом раствора камеры (*viewAngle*) и форматным соотношением (*aspectRatio*) сторон плоскостей усечения пирамиды видимого объёма.

- **Ортогональная проекция** – используется для описания сцены, представленной в виде параллелепипеда, внутренняя часть которого является просматриваемой областью сцены. В ортогональной проекции все лучи параллельны оси зрения камеры. Матрица ортогональной проекции задаётся методом *OrthoLH*:

```
device→Transform→Projection = Matrix::OrthoLH( float width, float
          height, float znearPlane, float zfarPlane );
```

  В функцию передаются два параметра, определяющие расстояние от камеры до ближней и дальней плоскостей отсечения (*znearPlane, zfarPlane*). Другие два параметры определяют поле зрения камеры – высоту и ширину сторон параллелепипеда (*width, height*).

Когда сцена спроецирована на проекционную плоскость, то координаты точек этой плоскости измеряются в **экранных координатах** (p, q) (Screen Coordinates).

2.8.3 Материалы, текстуры и загрузка полигональных моделей из X-файла

Для моделирования реалистичных геометрических сцен, в качестве составных объектов используются полигональные модели или Mesh-объекты.

Полигональной моделью (*Mesh*) называется объект, содержащий вершинный и индексный буферы, определяющих полигоны некоторого геометрического объекта.

Основным преимуществом использования Mesh-объектов является возможность удобного хранения и формирования сложных геометрических объектов, а также их эффективный рендеринг.

Для реалистичного отображения не достаточно только информации о его вершинах. К примеру, Mesh-объекты не содержат информации о цвете вершин, а так же способе наложения на объект текстуры, более подробно описывающеё его поверхность.

Материал – это описание, того каким образом покрываемая им поверхность будет отражать свет. Материал включает в себя отражаемый диффузный (*Diffuse*) и общий цвет (*Ambient*), цвет блика (*Speccular*), параметры отражения света и уровень прозрачности материала (*Alpha*).

Текстура – это изображение, которое накладывается на поверхность геометрического объекта определённым образом. Способ наложения текстуры определяют текстурные координаты.

Информация о текстурных координатах обычно добавляется в буфер вершин геометрического объекта, на который будет наложена текстура. Обычно задание текстурных координат (и буфера вершин в целом) достаточно трудоёмкий процесс для сложных объектов. Но, к счастью, вся информация о полигональной модели может быть загружена из файла специального формата – X-файла.

X-файл – это файл в формате XML, хранящий следующие данные полигональной модели:

- Подобъекты полигональной модели;
- Буферы вершин (с координатами текстуры) каждого из подобъектов;
- Индексные буферы каждого из подобъектов;
- Параметры материалов каждого из подобъектов;
- Информацию о файле текстуры каждого из подобъектов.

Данные из X-файла могут быть загружены с помощью следующей функции:

```
// Код функции приведён на языке C++/CLI:
Mesh^            MyMesh;      // Внешнее поле для меш-объекта
Array<Material>^ MyMaterials; // Внешний массив материалов модели
Array<Texture^>^ MyTextures;  // Внешний массив текстур модели

void LoadMesh( String^ file )
{
  Array<ExtendedMaterial>^ mtrl;
  // Загружаем модель из X-файла:
  MyMesh = Mesh::FromFile(file, MeshFlags::Managed, device, mtrl);

  // Если модель содержит материалы, сохраняем их:
  if ((mtrl != nullptr) && (mtrl->Length > 0))
  {
    MyMaterials = new Array<Material>( mtrl->Length );
    MyTextures  = new Array<Texture^>( mtrl->Length );
    // Сохраняем каждый материал и текстуру:
    for (int i=0; i < mtrl->Length; i++)
    {
      MyMaterials[i] = mtrl[i].Material3D;
      if ( (mtrl[i].TextureFilename != nullptr) &&
           (mtrl[i].TextureFilename != String::Empty) )
      {// Если есть информация о текстуре - пробуем загрузить её:
        MyTextures[i] = TextureLoader::FromFile(device,
                        mtrl[i].TextureFilename);
      }
    }
  }
}
```

3. Описание графического интерфейса (GUI) программы “Mechanics Editor .NET”

3.1 Установка и запуск программы

Для работы программы необходим компьютер с минимальными требованиями:

- Процессор с тактовой частотой не ниже 1 GHz;
- Видеоадаптер совместимый с DirectX 9.0c;
- Операционная система Microsoft Windows NT SP6 / 2000 SP4 / XP / 2003 / Vista;

Перед запуском программы, на компьютер необходимо установить следующие вспомогательные приложения (все продукты в свободном доступе):

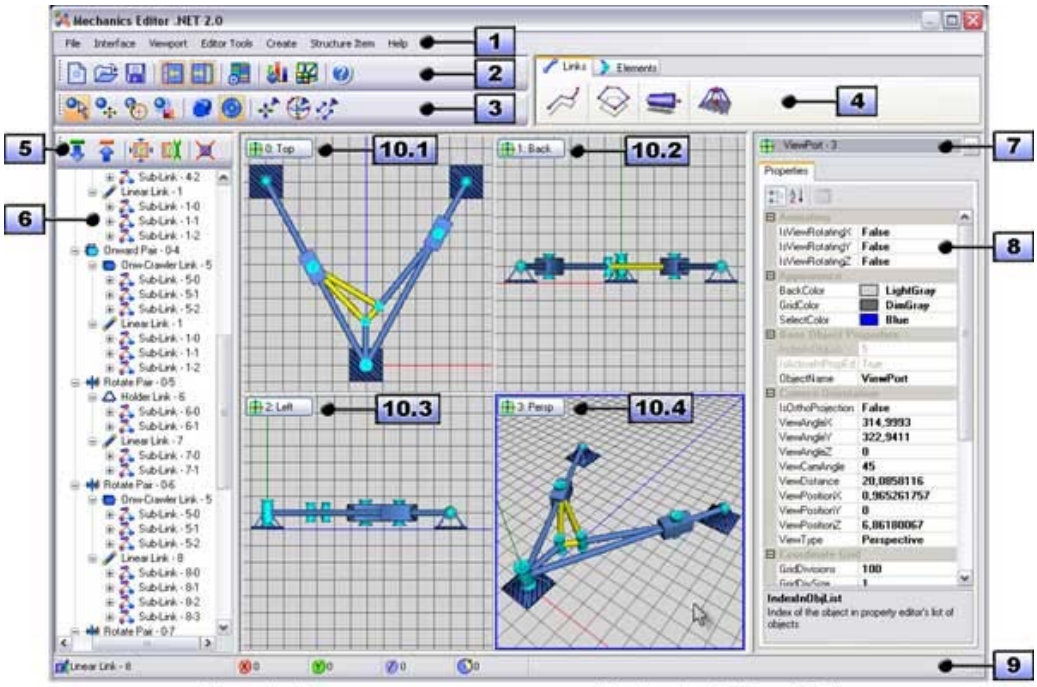
- Microsoft DirectX 9.0c;
- Microsoft .NET FX 2.0 (Beta2);
- Microsoft Managed DirectX (входит в дистрибутив DirectX);

Чтобы установить программу, просто загрузите с сайта и распакуйте архив файлов программы в каталог на локальном диске компьютера. Для нормальной работы в каталоге должны находиться следующие файлы:

1. **MechEditor.exe** (264 Kb);
2. **MechModel.dll** (152 Kb);
3. **MDXUtility.dll** (236 Kb);
4. **Viewport.dll** (80 Kb);
5. **TabSplitPan.dll** (20 Kb);

Для запуска программы нужно выполнить файл **MechEditor.exe**;

3.2 Панели главного окна программы

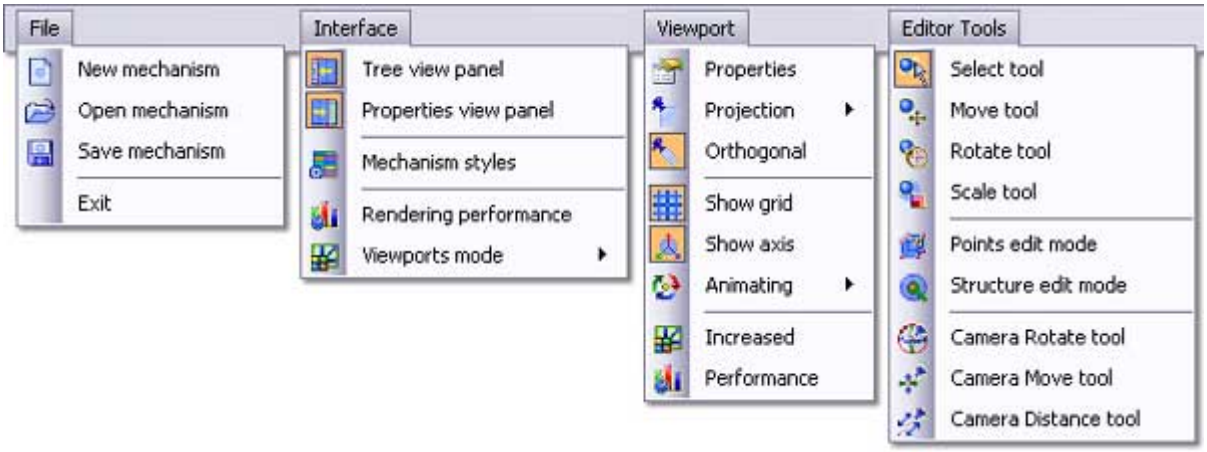


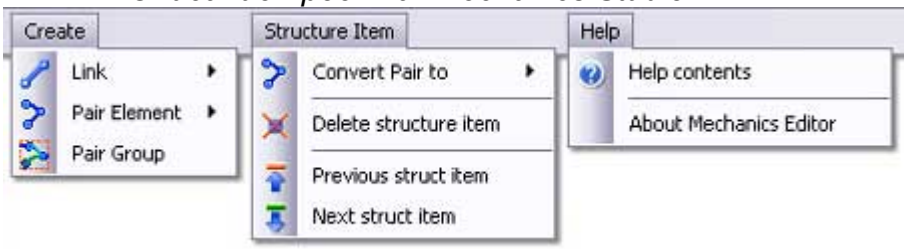
Интерфейс главного окна программы "Mechanics Editor .NET"

Назначение панелей главного окна приводится ниже:

1. **Главное меню программы** – содержит меню по всем основных операциям системы;
2. **Панель управления интерфейсом программы** – содержит такие кнопки, как “создание нового механизма”, “сохранение”, “загрузка”, кнопки управления панелями главного окна и открытия других окон, а так же кнопки выполнения настроек;
3. **Панель инструментов редактирования** – содержит инструменты редактирования геометрических объектов (“выделение”, “перемещение”, “вращение”, “масштабирование”), изменения режимов редактирования и кнопки управления камерами видов;
4. **Панель выбора объектов для конструктруирования механизмов** – содержит кнопки выбора структурных объектов механизма, позволяющих выполнять конструирование;
5. **Панель управления структурой механизма** – эта панель содержит кнопки навигации по структуре механизма, управления группами и удаления объектов;
6. **Дерево структуры механизма** – на этой панели отображается структура механизма в виде дерева (описанная в пункте " Требования к системе");
7. **Выпадающий список выбора объекта, отображаемого в редакторе свойств**;
8. **Редактор свойств выбранного из списка объекта** – позволяет редактировать выбранный объект из списка: просматривать и изменять его параметры;
9. **Информационная панель программы** – отображает вспомогательную информацию: название выделенного объекта, координаты курсора мыши в видах и др.;
10. **Панели 4-х видов** – каждая из панелей независимо отображает установленную в ней проекцию пространственной структурной схемы механизма и позволяет выполнять управление камерой вида. По умолчанию, виды загружаются в следующей конфигурации:
    1. **Вид № 0**: по умолчанию - вид сверху;
    2. **Вид № 1**: по умолчанию - вид спереди;
    3. **Вид № 2**: по умолчанию - вид слева;
    4. **Вид № 3**: по умолчанию - перспектива.

3.3 Главное меню





Главное меню программы показано на рис. выше в развёрнутом виде. Практически все команды главного меню дублируются кнопками панелей инструментов или другими контекстными меню программы. Поскольку об инструментарии программы далее речь пойдёт более подробно, то я опишу только корневые пункты главного меню, не останавливаясь на подпунктах:

- **Меню File** – содержит команды создания, сохранения и загрузки механизмов, а также выхода из программы;
- **Меню Interface** – содержит команды управления интерфейсом программы, такие как настройка видимости панелей, открытие других окон и настройка стилей отображения структурной схемы;
- **Меню Viewport** – содержит команды настройки активного вида (обозначаемого жирной рамкой), дублируется контекстным меню вида;
- **Меню Editor Tools** – содержит команды переключения активных инструментов редактора объектов и режимов редактирования;
- **Меню Create** – содержит команды создания новых звеньев, элементов кинематических пар или групп;
- **Меню Structure Item** – содержит команды редактирования структуры механизма, дублируется контекстным меню панели просмотра структуры механизма;
- **Меню Help** – содержит команды вызова помощи по программе и краткой информации о ней.

Далее при описании функций программы, будут указываться соответствующие им пункты главного меню.

3.4 Настройка интерфейса

Кнопки приведённой на рисунке панели выполняют следующие функции:

-  **(File→New Mechanism)** – выполняет создание нового механизма;
-  **(File→Open Mechanism)** – выполняет загрузку сохранённого механизма;
-  **(File→Save Mechanism)** – выполняет сохранение созданного механизма;
-  **(Interface→Tree view panel)** – отображает или скрывает панель структуры механизма;
-  **(Interface→Properties view panel)** – отображает или скрывает панель редактирования свойств;

- **(Interface→Mechanism Styles)** – отображает настройки стилей отображения механизма;
- **(Interface→Rendering Performance)** – открывает окно производительности рендеринга;
- / **(Interface→Viewports Mode)** – переключает режим отображения 4-х видов или 1-го увеличенного вида.

3.5 Инструментарий редактора

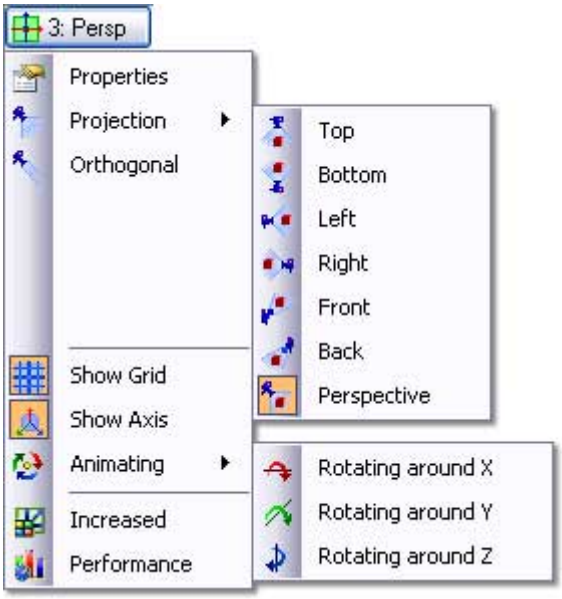
Кнопки приведённой на рисунке панели выполняют следующие функции:

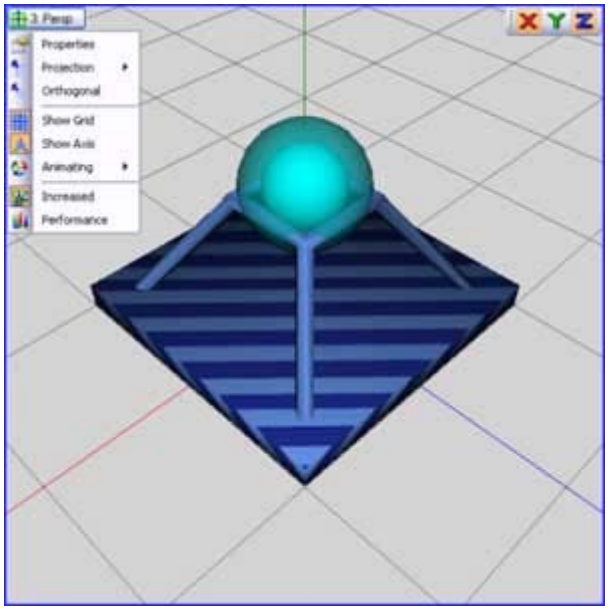
- **(Editor Tools→Select Tool)** – Инструмент выделения объектов механизма: При наведении мыши на объект в виде, курсор принимает вид , обозначая возможность выделения объекта;
- **(Editor Tools→Rotate Tool)** – Инструмент вращения объектов механизма: При наведении мыши на объект в виде, курсор принимает вид , обозначая возможность выделения объекта и его вращения вокруг активной оси вида;
- **(Editor Tools→Move Tool)** – Инструмент перемещения объектов механизма: При наведении мыши на объект в виде, курсор принимает вид , обозначая возможность выделения объекта и его перемещения вдоль активных осей вида;
- **(Editor Tools→Scale Tool)** – Инструмент масштабирования объектов механизма: При наведении мыши на объект в виде, курсор принимает вид , обозначая возможность выделения объекта и его масштабирования;
- / **(Editor Tools→Points Edit Mode)** – переключает режим отображения точек структуры механизма. Режим редактирования структурных точек механизма позволяет изменять положение точек, определяющих геометрическую структуру механизма;
- / **(Editor Tools→Structure Edit Mode)** – переключает режим редактирования структуры объектов механизма. Если режим включён, то разрешается выделять и изменять положение объектов подзвеньев;
- **(Editor Tools→Camera Rotate Tool)** – Инструмент вращения камеры активного вида. Курсор мыши, соответствующий этому инструменту имеет вид ;
- **(Editor Tools→Camera Rotate Tool)** – Инструмент перемещения камеры активного вида. Курсор мыши, соответствующий этому инструменту имеет вид ;

- **(Editor Tools→Camera Rotate Tool)** – Инструмент изменения фокусного расстояния камеры активного вида. Курсор мыши, соответствующий этому инструменту имеет вид ;

Инструменты перемещения и вращения объектов предусматривают выбор активных осей, вдоль которых будет производиться перемещение или вращение объектов. Выбор активных осей выполняется посредством панели **Axis Control**. Активными являются те оси, кнопки которых утоплены.

3.6 Управление видами





Вид перспективной проекции представлен на рис. справа. В левом верхнем угле каждого вида расположена кнопка, на которой написан индекс данного вида и тип проекции. При щелчке левой кнопкой мыши по этой кнопке, отображается контекстное меню вида, представленное на рис. слева:

- **Properties** – отображение свойств объекта данного вида в редакторе свойств;
- **Projection** – выбор типа проекции вида из следующего списка: Top – сверху, Bottom – снизу, Left – слева, Right – справа, Front – спереди, Back – сзади, Perspective – перспектива;
- **Orthogonal** – определяет тип проецирования: ортогональная или перспективная проекция;
- **Show Grid** – определяет видимость координатной сетки данного вида;
- **Show Axis** – определяет видимость осей координат данного вида;
- **Animating** – позволяет включать анимацию вращения камеры вокруг нескольких координатных осей X, Y или Z;
- **Increased** – переключает режим отображения видов – 4 вида одновременно или 1 увеличенный вид;
- **Performance** – открывает окно построения графиков, отражающих производительность рендеринга.

Описанное выше контекстное меню дублируется главным меню программы - Viewport.

Для работы с видом используются 3 кнопки мыши, выполняющие различные действия:

- Левая кнопка мыши – выполняет операции выбранного инструмента редактирования;
- Средняя кнопка мыши – включает инструмент перемещения камеры вида;
- Правая кнопка мыши – включает инструмент изменения фокусного расстояния камеры;

Оси координат видов программы обозначаются различными цветами – в соответствии:

- Ось X – обозначается красным цветом;
- Ось Y – обозначается зелёным цветом;
- Ось Z – обозначается синим цветом.

Точка на координатной плоскости вида, соответствующая положению курсора мыши отображается в виде 3-х мерного крестика, а её координаты выводятся в информационной панели, в нижней части окна.

3.7 Конструктор механизмов

3.7.1 Создание новых звеньев механизма

Для создания нового звена механизма выберите это звено на панели:

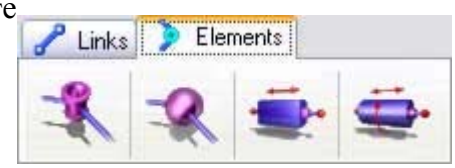
- Линейное звено – **Poly-Line Link**;
- Контурное звено – **Contour-Line Link**;
- Звено ползунка – **Crawler Link**;
- Звено крепления – **Holder Link**.

Когда нажата кнопка, соответствующая выбранному звену, на структурной схеме в видах отображаются точки элементов кинематических пар, к которым возможно присоединение звена. Для того, чтобы начать создание звена, нужно щёлкнуть левой кнопкой мыши по одному из элементов, к которому возможно присоединение. Далее щёлкая левой кнопкой мыши по видам, добавьте необходимое количество точек в структуру звена. Программа либо автоматически завершит процесс создания звена, либо это нужно будет сделать самостоятельно, нажав правую кнопку мыши.

Новое созданное звено будет автоматически добавлено в кинематическую пару, указанного вами элемента и соединено с его структурной точкой. Это можно видеть на дереве иерархической структуры механизма.

3.7.2 Создание новых кинематических пар механизма

Для создания новой кинематической пары, выберите элемент этой пары на панели (рис. 3.10):



- Элемент вращательной пары – **Rotator Element**;
- Элемент сферической пары – **Sphere Element**;
- Элемент поступательной пары – **Onward-Crawler Element**;
- Элемент цилиндрической пары – **Cyl-Crawler Element**.

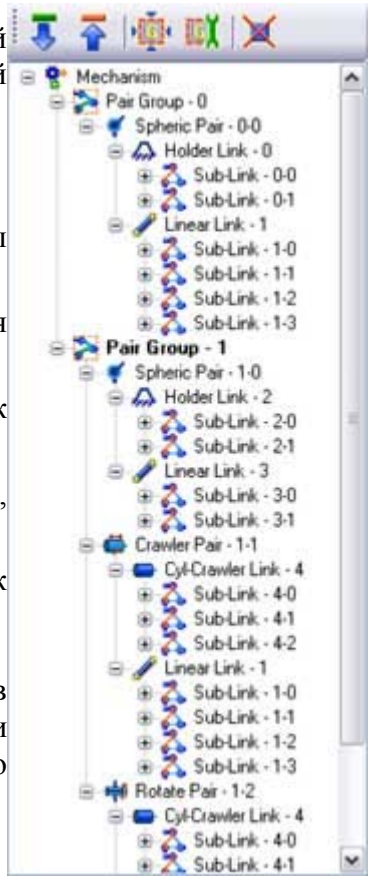
Когда нажата кнопка, соответствующая выбранному элементу, отображаются точки звеньев, к которым возможно присоединение данного элемента. Дальнейшие действия зависят от типа выбранного элемента:

- *Если выбран вращательный или сферический элемент*, то нужно щёлкнуть левой кнопкой мыши на одной из точек звена, к которому будет прикреплён этот элемент;
- *Если выбран поступательный или цилиндрический элемент*, то нужно щёлкнуть левой кнопкой мыши на одном из отрезков линейного или контурного звена, на который будет установлен данный элемент и соответствующее звено ползунка;

В новую созданную пару автоматически включается то звено, к которому был присоединён элемент этой пары. Так же необходимо отметить, что элементы поступательной и цилиндрической пар создаются и существуют всегда вместе с соответствующим звеном ползунка.

3.8 Структура механизма

Структура механизма отображается в виде иерархической древовидной структуры (см. рис. 3.11) имеющей 5 уровней вложенности.



1. **узел механизма** – корневой узел;
2. **узлы группы кинематических пар** – дочерние узлы механизма;
3. / / / **узлы кинематических пар** – являются дочерними по отношению к узлам групп;
4. / / / **узлы звеньев** – дочерние по отношению к узлам присоединённых пар;
5. **узлы подзвеньев** – дочерние по отношению к звеньям, которые они составляют;
6. **узлы геометрических точек** – дочерние по отношению к содержащим их подзвеньям;

При выборе, какого либо узла дерева, в редакторе свойств отображаются свойства объекта, соответствующего этому узлу и производится выделение соответствующего геометрического объекта на структурной схеме.

Управление структурой механизма осуществляется посредством панели, расположенной сверху от дерева структуры (см. рис. выше). Далее приводится описание кнопок этой панели инструментов:

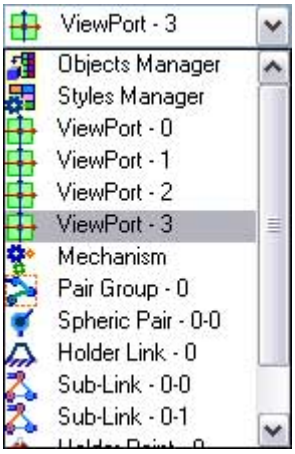
- **(Structure Item→Next Struct Item)** – выполняет переход на расположенный ниже объект структуры механизма, выделяет его и отображает в редакторе свойств;
- **(Structure Item→Previous Struct Item)** – выполняет переход на расположенный выше объект структуры механизма, выделяет его и отображает в редакторе свойств;

- **(Create→Pairs Group)** – создаёт новую группу кинематических пар;
- **(Structure Item→Editing Group)** – выполняет установку выбранной группы в качестве текущей для редактирования – т. е. группы, в которую будут добавляться по умолчанию все новые создаваемые пары;
- **(Structure Item→Delete struct item)** – удаляет выбранный объект механизма, если это не нарушает структуры механизма;

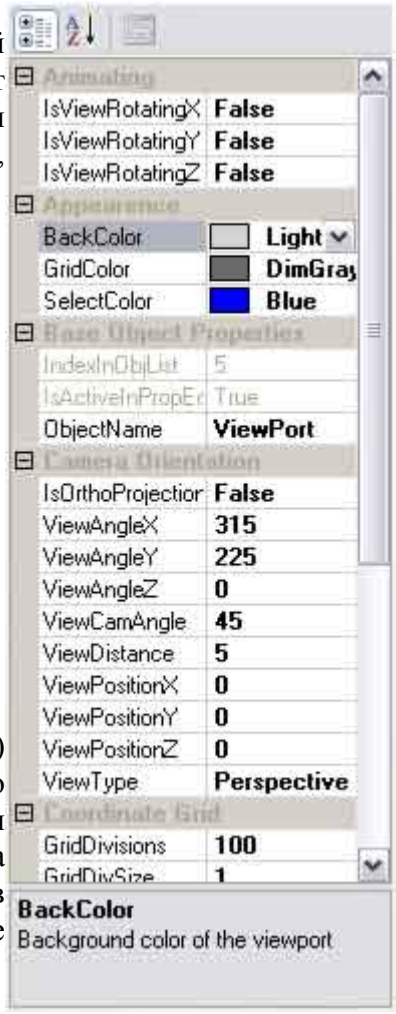
Нажатием правой кнопки мыши на выделенном узле дерева вызывается контекстное меню, содержащее список возможных операций над объектом данного узла. Кроме перечисленных выше операций, в контекстном меню содержится операция конвертирования объектов кинематических пар в совместимый тип.

3.9 Редактор объектов

Программа позволяет выбирать пользователю любой доступный объект и изменять его свойства. Выбор объект производится с помощью выпадающего списка, в котором указываются индексированные имена объектов и иконки, соответствующие их типам (рис. ниже):



После выбора объекта, в редакторе свойств (см. рис. справа) отображаются свойства этого объекта разделённые на группы, по смысловому значению. Названия недоступных для редактирования свойств обозначены серым шрифтом, а изменяемых – чёрным. При выборе произвольного свойства, в нижней части панели редактора, отображается краткое описание этого свойства.



Если пользователь вводит неверное или недопустимое значение свойства, то программа выдаст окно с подсказкой о том, как нужно правильно задавать значения этого свойства.

Заключение и результаты

На данный момент, результатом данного проекта является работоспособная программная система геометрического и структурного моделирования широкого класса пространственных рычажных механизмов. Данная версия программы названа “**Mechanics Editor v. 1.0**”.

В работе проведёно исследование предметной области – теории механизмов и машин, на основе чего были **спроектированы и реализованы структуры данных**, необходимые для решения задач **конструирования механизмов** из имеющихся звеньев и элементов кинематических пар. Структуры данных разрабатывались с целью обеспечения возможности их дальнейшего расширения для решения расчетных задач над построенным механизмом.

Программа была реализована для работы на платформе **.NET Framework 1.1** и затем модернизирована под **.NET 2.0**. Для программирования была выбрана именно технология .NET, поскольку, на сегодняшний день, она позволяет свести временные затраты кодирования до минимума. Кроме того, выбор технологии .NET обеспечивает в дальнейшем использовать все самые современные технологии Microsoft только за счёт классов библиотеки **FCL**.

В процессе конструирования программной системы потребовалось **использование средств 3D-графики**, для отображения 3-х мерных структурных схем моделей механизмов. В качестве библиотеки для программирования 3D-графики было использовано новое на данный момент расширение библиотеки Microsoft DirectX 9.0, называемое **Managed DirectX**. Данная библиотека позволила программировать 3-х мерную графику на любом из .NET-совместимых языков.

В дальнейшем планируется продолжить работу над данной программой и решить следующие важные для пользователя программы задачи:

- Проектирование таблиц реляционной базы данных, позволяющей хранить всю необходимую информацию о механизмах и настройках программы, её реализация на SQL Server и средства работы с ней на ADO.NET;
- Сохранение и загрузка моделей механизмов, а так же профилей пользователей в файлах и других хранилищах в XML-совместимом формате;
- Реализация алгоритма автоматического распознавания структуры механизма и формирования групп Ассура по имеющимся группам механизма;
- Разработка и реализация расчетных алгоритмов решающих прямую (и, возможно, обратную) задачи расчёта кинематики звеньев механизма;

Таким образом, данную программу и разработанные структуры данных можно рассматривать, как основу для построения более сложной системы, решающей часть широкого круга задач, возникающих при проектировании и исследовании пространственных механизмов.

В следующей таблице перечислены постановки задач и статус их выполнения.

Таблица 6: Результаты работы над задачами проекта

№	Постановка задачи	Статус выполнения	Начало работы	Конец работы
1	Проектирование структур данных для представления моделей механизмов	Завершено	1 сентября 2004г.	1 октября 2004г.
2	Изучение библиотеки прогаммирования 3D графики - Managed DirectX	Завершено	1 октября 2004г.	1 февраля 2005г.

3	Конструирование и программирование системы моделирования механизмов "Mechanics Editor .NET"	Завершено	20 января 2005г.	10 мая 2005г.
4	Подготовка документации по написанной программе	Завершено	10 мая 2005г.	31 мая 2005г.
5	Рефакторинг кода написанного приложения с целью реструктуризации и перехода от .NET 1.1 к .NET 2.0	Завершено	1 августа 2005г.	27 августа 2005г.
6	Создание данного сайта описывающего проект "Mechanics Studio .NET"	Завершено	28 августа 2005г.	4 сентября 2005г.
7	Продолжение работы над программой "Mechanics Editor .NET": разработка БД механизмов и функций сохранения и загрузки механизмов.	Запланировано	Сентябрь 2005г.	
8	Разработка и реализация алгоритма разбиения пар механизма на группы 0-й подвижности	Не начато		
9	Разработка и реализация алгоритмов кинематического рассчёта движения механизмов (решение прямой задачи)	Не начато		

Литература

1. К.В. Фролов, С.А. Попов, А.К. Мусатов и др. “**Теория механизмов и машин**”, учебник для вузов, под редакцией К.В. Фролова – М.: Высш. шк., 1987. – 496с.: ил.;
2. Левитский Н.И. “**Теория механизмов и машин**”: учебное пособие для вузов. – 2-е изд., перераб. и доп. – М.: Наука. Гл. ред. физ.-мат. лит., 1990. – 592 с.;
3. Артоболевский И.И. “**Теория механизмов и машин**”: учебник для вузов. – 4-е изд., перераб. и доп. – М.: Наука. Гл. ред. физ.-мат. лит., 1988. – 640с.;
4. Э. Е. Пейсах “**О терминологии по теории механизмов и машин**”, статья из журнала “Теория механизмов и машин”, 2004, №2, том 2, стр. 80-94;
5. Турлапов В.Е., Лукин Д.В. “**Моделирование кинематики пространственных механизмов в CAD-среде, на примере AutoCad**” / В сб. докладов конф. “Graphicon 2000”;
6. Просиз Дж. “**Программирование для Microsoft .NET**” / Пер. с англ. – М.: Издательско-торговый дом “Русская Редакция”, 2003. – 704 стр.: ил.;
7. Рихтер Дж. “**Программирование на платформе Microsoft .NET Framework. Мастер-класс.**”, пер. с англ. – 3-е изд. – М.: Издательско-торговый дом “Русская Редакция”; Спб.: Питер, 2005. – 512 стр.: ил.;
8. Троелсен Э. “**C# и платформа .NET. Библиотека программиста**” – СПб.: Питер, 2003. – 800 стр.: ил.;
9. Том Миллер “**DirectX 9 с управляемым кодом. Программирование графики и игр**” KickStart, перевод с английского Созинова С. Б. – М. “КомБук”, 2005 – 400 стр.: ил.;
10. Станислав Горнаков “**DirectX 9: Уроки программирования на C++**” – СПб.: БХВ-Петребург, 2004 – 400 стр.: ил.;
11. Френсис Хилл “**OpenGL: Программирование компьютерной графики. Для профессионалов**” – СПб.: Питер, 2002. – 1088 стр.: ил.;
12. Макконнелл С. “**Совершенный код. Мастер класс**” / Пер. с англ. – М.: Издательско-торговый дом “Русская Редакция”; Спб.: Питер, 2005. – 896 стр.: ил.;
13. Stewart Baird, “**Teach Yourself Extreme Programming in 24 Hours**”, Sams publishing, 2002, 480 pages;
14. Леоненков А.В. "**Самоучитель UML**" - 2-е изд., перераб. и доп. - СПб.: БХВ-Петербург, 2004. - 432 с.: ил.;

Ссылки

15. www.gotdotnet.com/team/directx/ - материалы, статьи и форум по DirectX;
16. www.intel.com/cd/ids/developer/asmo-na/eng/57590.htm – Статья инженера ПО компании Vital Images, с введением в Managed DirectX (от Intel);
17. www.aspfree.com/c/a/ASP.NET/Managed-DirectX-First-Steps-Direct-3D-Basics-and-Direct-X-vs-GDI/ - большая статья по созданию приложений на Managed DirectX;
18. www.gotdotnet.ru - форумы по Windows Forms, ASP.NET, ADO.NET и др.;
19. www.rsdn.ru - Russian Software Developer Network – статьи и примеры программ по различным темам;
20. ru.thespoke.net - сайт российского сообщества .NET-разработчиков, называемый "Спицы";
21. www.intelcup.ru - сайт конкурса "Intel Cup" в котором учавствует данный проект;