Глава 6

ИНСТРУМЕНТАРИЙ РАЗРАБОТКИ ПРОГРАММИРУЕМОЙ АНИМАЦИИ ДЛЯ КУРСА «НАЧЕРТАТЕЛЬНАЯ ГЕОМЕТРИЯ»

6.1. Методические основания применения анимации при создании электронных учебников по графическим дисциплинам

Информатизация учебного процесса требует создания электронных пособий, которые могут не только использоваться для самостоятельной работы студента, но и служить презентационным материалом преподавателю.

Начертательная геометрия, предлагающая графические методы решения и исследования пространственных задач на плоскости, до сих пор является самым мощным инструментом развития пространственного мышления. Построение эпюр Монжа – фундаментальное умение для этого курса, которое необходимо для успешного освоения инженерной графики при подготовке инженера любой специальности. Задачи начертательной геометрии сложны и трудоемки, построения требуют хорошего абстрактного мышления, внимания и временных затрат, а приобретение настоящих умений дается только решением множества задач различными методами.

Создание электронных методических пособий для некоторых предметов затруднено из-за отсутствия, в первую очередь, четких алгоритмов, позволяющих с необходимой достоверностью формализовать процесс решения тех или иных задач, во вторую – инструментов, простых для освоения человеком, далеким от программирования.

Практически любую задачу начертательной геометрии можно интерпретировать как задачу аналитической геометрии, которую можно решить методами аналитической геометрии, выделив ее математическое описание. В этом случае, однако, графические методы решения и исследования пространственных задач на плоскости оказываются ненужными. Такой подход не позволяет развивать навыки владения языком графических образов, совершенно необходимые инженеру для последующего создания проектной доку-

ментации. Решение с помощью компьютеров некоторой задачи начертательной геометрии состоит не только в получении конечного числового результата, но и в максимально подробном отображении всех операций в ходе решения, а также в предоставлении инструментов, позволяющих рассматривать задачу как в пространстве, так и на плоскости (эпюре).

Несмотря на большое количество способов задания параметров геометрических объектов, в преподавательской практике, в зависимости от характера задачи, ограничиваются несколькими, самыми нужными. Эти ограничения используются для построения условия задачи и создания интерфейса пользователя.

Возможность декомпозиции процесса решения задач на некоторое число элементарных, однотипных операций позволяет получить итерационные способы их решения, которые могут быть автоматизированы с помощью вычислительной техники.

Процесс решения практически любой задачи курса начертательной геометрии, начиная с формирования условия, можно представить некоторой совокупностью шагов по преобразованию размера, формы, положения объекта. Изменение этих свойств (и цвета) объекта во времени называется анимацией. Традиционные технологии обучения – объяснение лектора у доски, проработка учебников и методических пособий, просмотр учебных видео- и слайд-фильмов – в большой степени позволяют отразить ключевые моменты методики решения задач, но обладают рядом недостатков. Ограничения лекции по времени и бумажного носителя информации по объему уменьшают степень подробности изложения. Скорость смены кадров видеофильма, количество и содержание слайдов определяются их создателями и не всегда соответствуют скорости восприятия студента. Зачастую проблему представляет чтение готовых эпюр, так как они ввиду ограниченности печатного пространства демонстрируют не всю последовательность построения, а лишь некоторые промежуточные этапы и конечный результат. Кроме того, вернуться к некоторому шагу объяснения позволяют только книга и слайды [11].

Компьютерная графика и мультимедиа меняют технологию и идеологию проектирования, в полной мере дают возможность реализовать любые методические приемы и устранить отмеченные недостатки традиционных технологий обучения графическим дисциплинам.

В настоящее время существует несколько электронных курсов по начертательной геометрии и инженерной графике [13; 55; 73; 100], но все они имеют существенный недостаток – отображают про-

цесс решения на определенных разработчиком условиях. Среди подобных курсов следует выделить работу доцента кафедры «Инженерная графика» Новосибирского государственного технологического университета К. А. Вольхина [13] и разработку, созданную в Санкт-Петербургском национальном исследовательском университете ИТМО под руководством заведующего кафедрой инженерной и компьютерной графики В. Т. Тозика [100].

Важными достоинствами названных работ являются: обширный теоретический материал, развитые системы тестирования, прекрасные статические иллюстрации и подробно изложенный алгоритм решения множества задач в анимированных роликах различных популярных форматов.

Электронный курс [13] включает html-страницы, содержащие JavaScript, реализующий слайд-фильмы, основанные на показе некоторого числа изображений, а также предоставляет возможность работыс чертежамив системахавтоматизированногопроектирования AutoCAD, bCAD и КОМПАС. Недостатком является то, что работающему с этим курсом студенту необходимо устанавливать указанные программы. Кроме того, изучающий предмет студент-первокурсник не обязательно имеет навыки устойчивой работы в перечисленных CAD-пакетах.

Электронный курс [100] в качестве иллюстраций содержит набор анимированных swf-роликов, которые визуализируют решения большинства типичных задач курса «Начертательная геометрия». Однако в этой разработке новое условие задачи, как и в работе [13], требует создания нового ролика и не дает возможности возврата к некоторому шагу алгоритма.

Компьютерная анимация по определению соответствует методологии решения задач начертательной геометрии и предоставляет все инструментальные средства ее реализации.

6.2. Виды анимации и выбор инструментальных средств разработки

Различают следующие основные виды анимированных изображений: анимированный gif, move-клип, демонстрации в Power-Point, демо-ролики, созданные в специализированных пакетах программ, Flash-анимация.

Важнейшими критериями выбора вида анимации и инструментов ее создания являются:

- достижимая степень интерактивности;

- распространенность средств воспроизведения;
- размер файла, что особенно важно при создании интернет-приложений для дистанционного образования.

К достоинствам анимированного gif-файла относятся простота создания, небольшой размер файла, поддержка большинством приложений, воспроизводящих анимацию и всеми интернет-браузерами. Недостатками формата являются невозможность организации диалога и значительный (для сети Интернет) размер файла для сложной анимации с множеством слоев (фреймов).

Все прочие из перечисленных форматов обладают хотя бы одним из следующих недостатков: неразвитостью или отсутствием диалоговых средств, значительными размерами, необходимостью инсталлировать инструменты просмотра.

Наиболее популярны среди систем трехмерной графики и анимации 3D Studio Max, Maya и Houdini. Их основными областями применения являются реклама, мультипликация, создание компьютерных игр и оформление телевизионных передач. Развитый интерфейс, значительный выбор реализованных кривых и поверхностей, способов формирования сцен, множество библиотек с атрибутами объектов и готовыми инструментами их наполнения – неоспоримые достоинства всех этих приложений. Houdini, обладающий, возможно, самыми совершенными инструментами создания анимации, требует от разработчиков знания не только трехмерного моделирования, но и объектно ориентированного программирования. Однако требовательность к аппаратному обеспечению, значительные размеры выходных файлов, не простой в освоении интерфейс ограничивают возможности их использования для обучения, не позволяют размещать их в Internet для дистанционной работы и требуют существенных временных затрат при разработке. Решения, которые можно было бы получить с помощью встроенного в 3D Studio Max языка Max Script, также не решают обозначенных выше проблем из-за недостаточной развитости его программного инструментария для создания анимации.

Наилучшим образом названным критериям выбора вида анимации и инструментов ее реализации удовлетворяет Flash-технология, реализующая покадровую анимацию, морфинг, анимацию с построением промежуточных изображений и программную анимацию.

Кадр содержит одно статическое изображение. При последовательном просмотре таких изображений создается иллюзия движения. Ключевые кадры, которые служат для внесения изменений в анимацию, создаются разработчиком, а промежуточные кадры могут быть созданы автоматически. Покадровая анимация (frame

by frame) представляет собой последовательность ключевых кадров. Типичными ключевыми кадрами при создании обучающих роликов по начертательной геометрии являются главные события сценария: возникновение осей, точек, текста и других объектов, начало отрисовки линий связи и их завершение и пр.

Анимация с автоматическим построением промежуточных кадров при перемещении объекта или изменении его характеристик (motion tweening) содержит ключевые кадры в начале и конце временной шкалы. Особым ее типом является преобразование одного объекта в другой (shape tweening). Промежуточные кадры автоматически строятся между ключевыми кадрами – опорными точками анимации. Такая техника значительно ускоряет процесс создания анимации и существенно уменьшает размер ролика в сравнении с покадровой анимацией, поскольку файл хранит только данные о ключевых кадрах и числовые значения, касающиеся способа преобразования объекта.

Морфингом называется преобразование объекта из начальной формы в конечную за некоторое количество шагов. Цвет при этом также подвергается изменению.

Анимация типа motion tweening применяется для создания эффекта движения объекта, позволяет менять размер, цвет и ориентацию на сцене. Построением промежуточных кадров можно управлять с помощью изменения параметров типа анимации (shape tweening и path tweening), задания вращения, указания сложной траектории движения. Можно изменять скорость воспроизведения промежуточных кадров и некоторые другие установки. Важнейшей особенностью при создании роликов по начертательной геометрии является возможность встраивать действия (action) в ключевые кадры и таким образом управлять демонстрацией.

С помощью встроенного языка программирования Action Script реализуется программная анимация, создающая файлы минимального размера по сравнению со всеми известными форматами и методами анимации и обеспечивающая интерактивность. Перечислим его основные особенности:

- управление временной диаграммой (таймлайном): воспроизведением, остановкой, проигрыванием отрезка фильма, перемещением на конкретный кадр, зацикливанием анимации и синхронизацией анимационного содержания;
- реализация диалогового режима: интерактивного обмена данными между пользователем и приложением Flash с помощью объектно-событийной модели;

- управление визуальным и звуковым содержимым фильма;
- программная генерация и редактирование визуального и звукового содержимого фильма, взаимодействие с другими приложениями и технологиями;
- поддержка клиент-серверной работы.

Macromedia Flash – комплексное многозадачное приложение, эффективное мультимедийное инструментальное средство, способное интегрировать широкий набор языков программирования, стандартов, технологий, мультимедийных форматов при одновременной поддержке программирования в различных средах. Приложения Flash могут быть доступны практически на любых платформах: от портативных устройств до настольных компьютеров и телеаппаратуры. Для карманных персональных компьютеров и других мобильных устройств выпущена специальная версия платформы Flash Lite, чья функциональность ограничена в расчете на возможности мобильных операционных систем и их аппаратных ресурсов. Flash-технология получила широкое распространение в глобальной сети. По данным компании, Adobe Flash Player установлен на 95 % компьютеров, имеющих доступ к сети. Из них у 82–87 % пользователей (в зависимости от региона) установлена последняя версия плейера [1].

Основную функциональность среды обеспечивают следующие возможности:

- использование векторной графики в качестве основного формата для визуализации, а следовательно, возможности создания графических web-приложений малого размера и обеспечение прекрасной масштабируемости приложений;
- поддержка растровой графики;
- наложение покадровой анимации на объекты, возможность программного управления анимацией;
- работа со слоями;
- обработка аудио- и видеоданных;
- использование объектно ориентированного скриптового языка программирования Action Script.

Пакет Macromedia Flash не имеет специализированных инструментов трехмерного инженерного моделирования. Тем не менее это можно сделать следующими тремя способами:

- программно реализовать моделирование поверхности;
- импортировать модель в формате фильмов *.swf, сделанных специализированным инженерным пакетом;
- обработать импортированный массив данных о процессе моделирования из специализированного инженерного пакета.

Впрочем, надежным способом является только программная реализация. С чтением наборов данных и экспортированных форматов часто возникают проблемы различного характера.

6.3. Цели, задачи и особенности разработанного пакета прикладных программ

Главной целью разработки является создание электронного обучающего курса по начертательной геометрии, содержащего графический редактор, визуализирующий графические методы решения позиционных и метрических задач курса «Начертательная геометрия». Исходные данные вводятся пользователем в интерактивном режиме. Отрисовка условия и решения происходит пошагово, по команде пользователя. Реализована операция отката на произвольное количество шагов, что позволяет отменить и повторить построения с того шага, который вызвал наибольшие трудности.

Обучающий курс может использоваться лектором, для самостоятельной работы студентов и создания презентационной графики. В силу того, что рутинные операции по отрисовке автоматизированы, существенно повышается производительность аудиторной и самостоятельной работы.

Благодаря технологиям, примененным при создании редактора, он, кроме автономного применения, органично встраивается в обучающий веб-ресурс, содержащий также набор неинтерактивных роликов и теоретические материалы по начертательной геометрии.

Создание графического редактора и обучающей веб-оболочки для решения позиционных и метрических задач включает следующие этапы:

1) описание методик решения задач методами начертательной геометрии;

2) создание сценариев роликов [11];

3) создание математических моделей построения условия и каждого шага решения, подлежащего визуализации;

4) разработка иерархии объектов, интерфейсов классов и методов библиотеки формирования эпюр и создания роликов;

5) разработка и наполнение оболочки обучающего веб-ресурса.

Для программной реализации проекта были выбраны позиционные и метрические задачи начертательной геометрии.

6.4. Решение позиционных и метрических задач методами начертательной геометрии

Решение многих задач способами начертательной геометрии сводится к определению позиционных и метрических характеристик геометрических образов. Все многообразие задач может быть условно отнесено к двум группам – позиционным и метрическим задачам.

Позиционные задачи начертательной геометрии

Позиционными называются задачи, в результате решения которых можно получить ответ на вопрос о взаимном расположении заданных геометрических образов.

Решение позиционных задач в конечном счете сводится к установлению принадлежности точки к линии. Эта группа задач решается с помощью третьего инварианта параллельного проецирования

$$(\forall Al)(A \in l) \Rightarrow A^{\alpha} \in l^{\alpha}, \qquad (6)$$

то есть если в пространстве точка инцидентна линии, то и проекция этой точки принадлежит проекции линии.

Все многообразие позиционных задач может быть отнесено к трем группам:

1) задачи на принадлежность точки поверхности;
2) задачи на определение точек пересечения линии с поверхностью;
3) задачи на построения линий пересечения двух поверхностей.

Метрические задачи

К метрическим относятся задачи, связанные с определением истинных (натуральных) величин расстояний, углов и плоских фигур на комплексном чертеже. Можно выделить три группы метрических задач.

1. Группа задач, включающих в себя определение натуральной величины отрезка прямой общего положения, определение расстояний от точки до прямой; от точки до плоскости, от точки до поверхности; от прямой до другой прямой; от прямой до плоскости; от плоскости до плоскости.

2. Группа задач, включающая определение углов между пересекающимися или скрещивающимися прямыми, между прямой и плоскостью, между плоскостями (имеется в виду определение величины двухгранного угла).

3. Группа задач, связанная с определением истинной величины плоской фигуры и поверхности (развертки) [73].

Задачи на определение расстояний между геометрическими образами

Искомое расстояние во всех задачах этой группы измеряется длиной отрезка, заключенного между заданными геометрическими образами и перпендикулярного к одной из них или одновременно к обеим. Этот отрезок проецируется в конгруэнтный ему отрезок на плоскость проекций, которая будет перпендикулярна одной или обеим геометрическим фигурам, между которыми определяется расстояние. Алгоритм решения задач этой группы будет следующим:

1) одним из способов преобразования комплексного чертежа привести заданные геометрические образы (или один из них) в положение, перпендикулярное какой-либо плоскости проекций;

2) построить проекцию искомого отрезка на эту плоскость.

Выбирая способ преобразования комплексного чертежа при составлении алгоритма, следует учитывать требования к компактности чертежа, четкость и возможную простоту графических операций [104].

Задачи на определение действительных величин плоских геометрических фигур и углов между ними

Общей схемой решения задач этой группы является приведение заданной плоской фигуры или плоскости угла в положение, параллельное одной из плоскостей проекций.

При выборе способа преобразования комплексного чертежа следует стремиться к простоте графических операций, их четкости и наименьшему количеству. Наиболее часто при решении задач применяются способы замены плоскостей проекций и вращения вокруг линии уровня плоскости. Способ вращения вокруг линии уровня плоскости является наиболее целесообразным для решения большинства задач данной группы, так как дает решение путем одного преобразования комплексного чертежа. К задачам данной группы можно отнести:

1) определение действительной величины плоской фигуры;

2) определение угла, образованного двумя пересекающимися или скрещивающимися прямыми;

3) определение величины угла, образованного прямой и плоскостью, двумя пересекающимися плоскостями.

Для программной реализации проекта изначально были выбраны следующие задачи:

- построение эпюры точки, прямой, плоскости, определение взаимного положения прямых и видимости конкурирующих точек;
- определение взаимного положения прямой и плоскости;
- построение линии пересечения двух поверхностей общего и частного положения;
- методы плоскопараллельного перемещения и замены плоскостей проекций;
- определение натуральной величины отрезка по его ортогональным проекциям;
- определение расстояния от точки до плоскости;
- определение действительной величины плоского угла по его ортогональным проекциям.

Опишем методами начертательной геометрии решение перечисленных ниже задач:

1) нахождение натуральной величины отрезка прямой общего положения методом замены плоскостей проекций;

2) нахождение натуральной величины отрезка прямой общего положения методом вращения;

3) определение взаимного расположения прямой и плоскости;

4) определение взаимного расположение двух плоскостей.

Нахождение натуральной величины отрезка прямой общего положения методом замены плоскостей проекций

Постановка задачи. Дан отрезок прямой, проходящей через точки a(x, y, z) и b(x, y, z). Определить натуральную величину отрезка прямой ab общего положения, пользуясь методом замены плоскостей проекций (рис. 16).

Натуральную величину отрезка прямой *ab* можно построить, если эта прямая станет параллельной какой-либо плоскости проекций. Выбираем новую плоскость проекций V1, расположив ее параллельно прямой *ab*. Для этого проводим новую ось *x*1 системы плоскостей проекций H / V1 параллельно горизонтальной проекции *A'B'* прямой *ab*.

Из точек *A'* и *B'* проводим линии связи перпендикулярно новой оси проекций *x*1 (системы H / V1). От точки пересечения линий связи с новой осью *x*1, откладываем расстояние, равное расстоянию от снимаемой оси *x* (системы H / V) до снимаемой проекции точки *A''*(расстояние = *zA*). На чертеже показано двумя засечками. Так построим точки *A*1'' и *B*1'' и получим прямую *A*1''*B*1''. Это и есть натуральная величина отрезка *ab*.

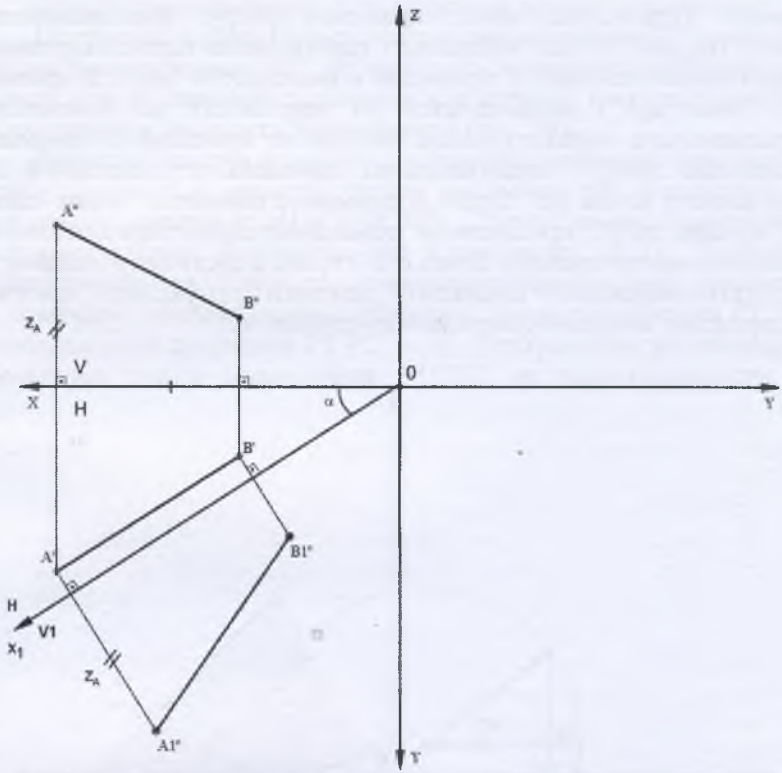

Рис. 16. Определение натуральной величины отрезка методом замены плоскостей проекций

Нахождение натуральной величины отрезка прямой общего положения методом вращения вокруг осей, перпендикулярных плоскости проекций

Постановка задачи. Дан отрезок прямой, проходящей через точки $a(x, y, z)$ и $b(x, y, z)$. Определить натуральную величину отрезка ab методом вращения (рис. 17) вокруг оси, перпендикулярной плоскости проекций.

Решение. Методы вращения предполагают неизменяемое положение плоскостей проекций и изменение положения объекта проецирования. В методах вращения точка вращается по окружности перпендикулярно оси вращения, то есть одна проекция траектории перемещения точки есть окружность, а другая –

прямая. Используем метод вращения вокруг фиксированной оси *i*. На рис. 17 ось вращения *i* расположена перпендикулярно фронтальной плоскости проекций и инцидентна точке *B* прямой *ab*. Точка a(*A*'') перемещается по окружности до положения, параллельного горизонтальной плоскости проекций и занимает положение *A*0''*B*'', горизонтальная проекция перемещенной по окружности точки *A*0'' будет построена с помощью линии связи на прямой *A*0'*A*', проведенной перпендикулярно горизонтальной проекции оси вращения *i*. Точка *B*(*B*') принадлежит оси вращения *i*'. Поэтому, соединив *A*0' с точкой *B*', получим прямую *A*0'*B*', которая и определит натуральную величину отрезка *ab*.

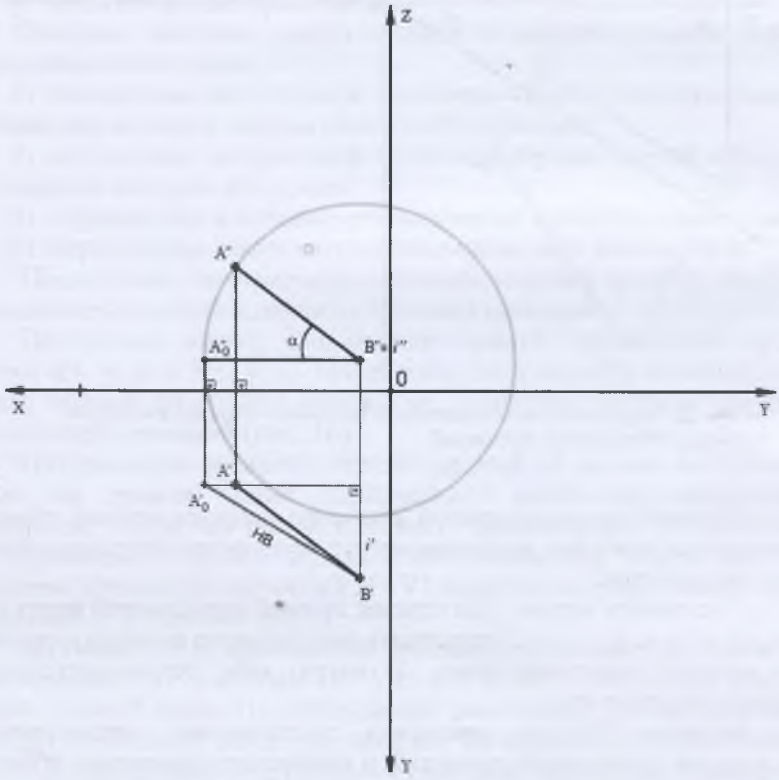

Рис. 17. Определение натуральной величины отрезка методом вращения вокруг оси, перпендикулярной фронтальной плоскости проекций

Определение взаимного расположения прямой и плоскости

Постановка задачи. Прямая *a* задана двумя точками *KM*, плоскость задана тремя точками *BCD*. Не пользуясь методами преобразования, определить их взаимное положение и построить точку пересечения, если прямая и плоскость пересекаются (рис. 18).

Решение. Вводим вспомогательную горизонтально-проецирующую плоскость γ, которую задаем ее следом γH. В плоскость γ заключаем прямую *a*: γH ≡ *a*'(*K*'*M*'). Строим линию пересечения вспомогательной плоскости γH и плоскости *BCD*. Учитывая собирательное свойство горизонтально-проецирующей плоскости γ, отмечаем горизонтальную проекцию линии пересечения *P*1'*P*2', совпадающую со следом *P*1'*P*2' ≡ γH. Достраиваем фронтальную проекцию линии пересечения *P*1''*P*2'' по принадлежности ее

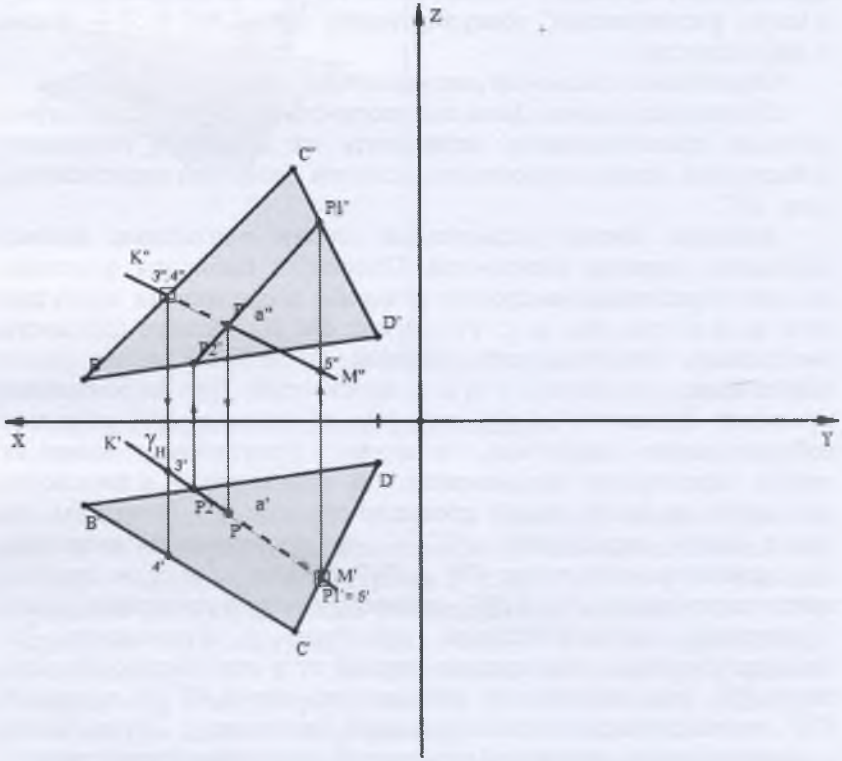

Рис. 18. Определение взаимного расположения прямой и плоскости

плоскости *BCD*. На фронтальной проекции отмечаем точку пересечения прямой *a*'' с прямой *P*1''*P*2''. Получаем фронтальную проекцию точки пересечения *P*'' = *a*''∩*P*1''*P*2''. По признаку принадлежности точки прямой *a* по линии связи достраиваем ее горизонтальную проекцию *P*'.

В позиционных задачах обычно определяют относительную видимость пересекающихся прямой и плоскости с помощью конкурирующих точек.

Конкурирующие точки 3'', 4'' имеют различные координаты *y*. Так как *y*4 > *y*3, то ближе к наблюдателю расположена прямая *BC*, которой принадлежит точка 4, отсюда – фронтальная проекция линии *km* до точки *P*'' – невидимая. Аналогичные рассуждения для пары конкурирующих точек *P*1', 5'. Ввиду того, что на фронтальной проекции видно, что *zP*1 > *z*5, то прямая *BC*(*B'C'*) в месте расположения конкурирующих точек *P*1' и 5' – ближе к наблюдателю.

Определение взаимного расположения двух плоскостей

Постановка задачи. Даны две плоскости *abc* и *fge*. Не пользуясь методом преобразования, определить их взаимное положение и построить линию пересечения, если эти плоскости пересекаются (рис. 19).

Решение. Линию пересечения строим с помощью вспомогательных секущих плоскостей. Плоскости выбираем фронтально- или горизонтально-проецирующими, проходящими через прямую *ac* и *cb* так, что *ac* ⊂ γV || *cb* ⊂ δV. На чертеже плоскости изображены фронтальными следами γV и δV. Строим линии пересечения плоскости γ и δ с плоскостью *fge*: вырожденная проекция плоскости γ (на чертеже обозначена γV) обладает собирательным свойством, и потому фронтальная проекция линии пересечения вспомогательной плоскости γ и плоскости *fge* лежит на фронтальной проекции плоскости γ (отмечаем две точки линии пересечения *P3''P4''* на фронтальной проекции). По принадлежности точек *P*3'' и *P*4'' прямым *ef* и *eg* по линиям связи достраиваем *P*3' и *P*4' – горизонтальную проекцию линии пересечения вспомогательной плоскости γ и плоскости *fge*. Определяем точку пересечения прямой *ac* с плоскостью *fge*. Это точка *PP*', она находится на пересечении линии *P*3'*P*4' и прямой *A'C'*, принадлежащих вспомогательной плоскости γ. Точно так же, с помощью вспомогательной плоскости δ, проходящей через прямую *cb*, строим точку *P* (*P*', *P*'') пересечения прямой *cb* с плоскостью *fge*. Соединив одноименные проекции точек *PP* и *P*, построим линию

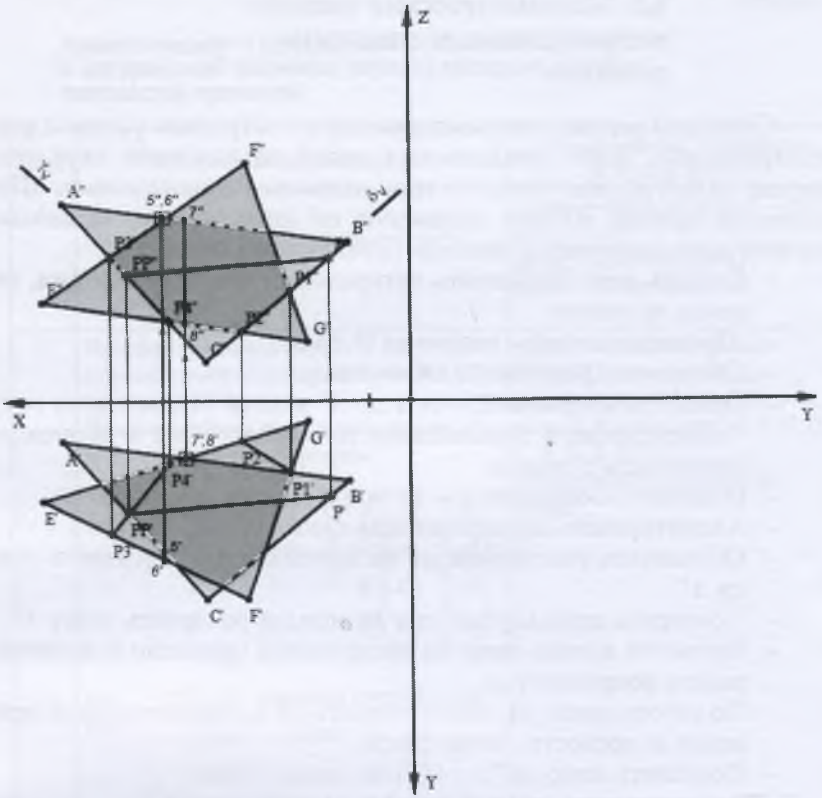

Рис. 19. Определение взаимного расположения двух плоскостей

пересечения плоскостей *abc* и *fge* . С помощью конкурирующих точек 5" и 6" определяем относительную видимость плоскостей *abc* и *fge*. По линиям связи строим 5' ⊂ *A'C'* и 6' ⊂ *E'F'*. Так как координата *y*6 > *y*5, то ближе к наблюдателю находится *fe* (*F"E"*). С помощью пары конкурирующих точек 7' и 8', принадлежащих, соответственно, *A'B'* и *E'F'* по линиям связи, построим фронтальные проекции этих точек. Определяем, что *z*7 > *z*8, и потому видим *ab* (*A'B'*), которая ближе к наблюдателю.

Для рассмотренных задач приведем математические модели построения их условия и решения.

6.5. Математические модели визуализации и сценарии роликов

Сценарий ролика, визуализирующего построение условий рассматриваемых задач, представляет собой объединение двух сценариев «Построение точки по трем заданным координатам», «Построение прямой общего положения по двум точкам, заданным своими координатами» и выглядит следующим образом.

- Создать оси. Изобразить натуральный масштаб чертежа, засечки на шкале.
- Провести линию – вектор от О до *xA*.
- Обозначить расстояние *xA* на оси *x*.
- Провести линию связи от *x* до *A'* и до *A''* (точек пока нет) – акцентировать параллельность с осями *y* и *z* и перпендикулярность с осью *x*.
- Отметить координату *y* – по оси *y* (так же, как по *x*).
- Акцентировать параллельность линии связи и оси *y*.
- Обозначить расстояние *yA* на линии связи и поставить точку *A'*.
- Повторить предыдущий шаг по оси *z* и поставить точку *A''*.
- Провести линию связи на профильной проекции и акцентировать координату *zA* .
- По *yw* отложить *yA*, акцентировать *yA* с горизонтальной проекции и провести линию связи.
- Поставить точку *A'''*.
- Акцентировать все три точки и их координаты.

Для задачи построения прямой добавить точку *B*. Получить прямые, соединив одноименные проекции точек.

В качестве акцентирования выбрано однократное «мелькание».

Определение натуральной величины отрезка через замену плоскостей проекций

Шаги методики решения этой задачи (рис. 16) описываются математическими зависимостями [64] и соответствующим программным кодом на языке ActionScript 3(AS3) в табл. 4. Для экономии места и времени читателей в таблице приведены только существенные с точки зрения авторов фрагменты кода на AS3. Следует обратить внимание на то, что для описания математической модели используется транспонированная матрица преобразования, так как в языке AS3 класс Matrix3D представляется именно транспонированной матрицей.

Таблица 4

Аналитическое и программное решение задачи о натуральной величине отрезка методом замены плоскостей проекций

Шаг	Математическая модель	Программная реализация в Action Script 3
1	Находим угол α. $\alpha = \operatorname{arctg}\left(\frac{y_b - y_a}{x_b - x_a}\right)$.	alpha = -Math.atan((b.y – a.y)/(b.x – a.x)) sina = Math.sin(alpha) cosa = Math.cos(alpha)

2	Поворачиваем плоскость V на угол α относительно оси, параллельной OZ, проходящей через A.
	Находим новые координаты точек $A1$ и $B1$ с помощью формулы:
	$TX = \begin{bmatrix} \cos\alpha & \sin\alpha & 0 & 0 \\ -\sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x1 \\ y1 \\ z1 \\ 1 \end{bmatrix}$,
	где $T = M^T$, M – матрица преобразования: $M = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 & 0 \\ \sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
		point - экземпляр класса Point3D
		matrix = new Matrix3D(cosa, -sina, 0, 0, sina, cosa, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1);
		point.transform.matrix3D = matrix;

3	Находим искомую величину: $l = \sqrt{(x_b - x_a)^2 + (y_b - y_a)^2 + (z_b - z_a)^2}$, где $A_1(x_a, y_a, z_a)$ $B_1(x_b, y_b, z_b)$	l = Math.sqrt((b.x – a.x)*(b.x – a.x) + (b.y – a.y)*(b.y – a.y) + (b.z – a.z)*(b.z – a.z));

Сценарий ролика, визуализирующего это решение, имеет следующий вид.

- Провести новую ось проекций $x1$ параллельно одной из проекций отрезка. Для этого из начала координат провести «резиновую линию» мышью.

- Достижение параллельности отметить изменением цвета резиновой линии с черного на красный.
- Кликом левой клавиши мыши зафиксировать достигнутое положение. Завершить построение новой оси, нажав на клавишу OK.
- При некорректном указании положения новой оси вывести об этом сообщение и обеспечить возможность повторить ввод.
- Опустить линии связи перпендикулярно новой оси проекций $x1$ из точек A и B.
- Акцентировать перпендикуляры. От точки пересечения линии связи с новой осью отложить расстояние от снимаемой оси до снимаемой проекции точки. Снимаемая ось – линия пересечения V и H.
- Соединив точки, получим искомую натуральную величину отрезка.

Определение натуральной величины отрезка методом вращения

Аналитическое и программное решение задачи о натуральной величине отрезка методом вращения (рис. 17) описано в табл. 5.

Таблица 5

Аналитическое и программное решение задачи о натуральной величине отрезка методом вращения

Шаг	Математическая модель	Программная реализация в Action Script 3
В случае преобразований на горизонтальной плоскости эпюры		
1	Находим угол α. $\alpha = -\text{arctg}\left(\frac{y_b - y_a}{x_b - x_a}\right)$	alpha = -Math.atan((b.y – a.y)/(b.x – a.x)) sina = Math.sin(alpha); cosa = Math.cos(alpha)

2	Находим новые координаты точек A и B с помощью матрицы преобразования
	$TX = \begin{bmatrix} \cos\alpha & 0 & -\sin\alpha & x_0 \\ \sin\alpha & 0 & \cos\alpha & y_0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x - x_0 \\ y - y_0 \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{bmatrix}$
		point - экземпляр класса Point3D
		matrix = new Matrix3D(cosa, 0, sina, x0, sina, 0 , cosa, y0, 0, 0, 1, 0, 0, 0, 0, 1);
		point.transform.matrix3D = matrix;
		matrix = new Matrix3D(cosa, sina, 0, a,-sina, cosa, 0, b, 0, 0, 1, c, 0, 0, 0, 1);

Продолжение табл. 5

Шаг	Математическая модель	Программная реализация в Action Script 3

	Здесь $T = M^T$, M – матрица преобразования: $M = \begin{bmatrix} \cos\alpha & \sin\alpha & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -\sin\alpha & \cos\alpha & 1 & 0 \\ x_0 & y_0 & 0 & 1 \end{bmatrix}$	point.transform.matrix3D = matrix;
		либо
		matrix = point.transform.matrix3D;
		matrix.appendTranslation(-x0, -y0, 0);
		matrix.appendRotation(-alpha, Vector3D.Z_AXIS);
		matrix.appendTranslation(x0, y0, 0);
		point.transform.matrix3D = matrix;

3	Находим искомую величину: $l = \sqrt{(x_b - x_a)^2 + (y_b - y_a)^2 + (z_b - z_a)^2}$, где $A_1(x_a, y_a, z_a)$ $B_1(x_b, y_b, z_b)$	l = Math.sqrt((b.x – a.x)*(b.x – a.x) + (b.y – a.y)*(b.y – a.y) + (b.z – a.z)*(b.z – a.z));
В случае преобразований на фронтальной плоскости эпюры		
1	Находим угол α. $\alpha = -\text{arctg}\left(\frac{z_b - z_a}{x_b - x_a}\right)$	alpha = -Math.atan((b.z – a.z)/(b.x – a.x)) sina = Math.sin(alpha) cosa = Math.cos(alpha)

2	Находим новые координаты точек A и B с помощью матрицы преобразования $TM = \begin{bmatrix} \cos\alpha & 0 & -\sin\alpha & x_0 \\ 0 & 1 & 0 & 0 \\ \sin\alpha & 0 & \cos\alpha & z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x - x_0 \\ y \\ z - z_0 \\ 1 \end{bmatrix} = \begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{bmatrix}$, где $T = M^T$, M – матрица преобразования и $M = \begin{bmatrix} \cos\alpha & 0 & \sin\alpha & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\alpha & 0 & \cos\alpha & 0 \\ x_0 & 0 & z_0 & 1 \end{bmatrix}$	point - экземпляр класса Point3D
		matrix = new Matrix3D(1, 0, 0,-x0, 0, 1, 0, 0, 0, 0, 1, -z0, 0, 0, 0, 1);
		point.transform.matrix3D = matrix;
		matrix = new Matrix3D(cosa, 0, -sina, x0, 0, 1, 0, 0, sina, 0, cosa, z0, 0, 0, 0, 1);
		point.transform.matrix3D = matrix;
		либо
		matrix = point.transform.matrix3D;
		matrix.appendTranslation(-x0, 0, -z0); matrix.appendRotation(-a, Vector3D.Y_AXIS);
		matrix.appendTranslation(x0, 0, z0); point.transform.matrix3D = matrix;

Окончание табл. 5

Шаг	Математическая модель	Программная реализация в Action Script 3
3	Находим искомую величину: $l = \sqrt{(x_b - x_a)^2 + (y_b - y_a)^2 + (z_b - z_a)^2}$, где $A_1(x_a, y_a, z_a)$ $B_1(x_b, y_b, z_b)$	l = Math.sqrt((b.x – a.x)*(b.x – a.x) + (b.y – a.y)*(b.y – a.y) + (b.z – a.z)*(b.z – a.z));

Сценарий ролика, визуализирующего это решение, имеет следующий вид.

- Акцентировать точки *A* и *B*.
- Вывести на экран сообщение: «Выберите точку на окружности». Вторая точка автоматически станет центром окружности.
- Отрисовать окружность на фронтальной и горизонтальной проекциях.
- Вывести на экран сообщение: «Перемещайте указатель мыши по окружности до тех пор, пока ее радиус к указателю не станет красным».
- Обеспечить возможность перемещать указатель мыши по окружности до тех пор, пока не будет достигнуто положение радиуса, соответствующее натуральной величине отрезка.
- Левой клавишей мыши зафиксировать найденное положение. Завершить указание точки нажатием на клавишу ОК.
- При некорректном указании искомого положения вывести сообщение об ошибке и предложение повторить указание точки и натуральной величины.

Определение взаимного расположения прямой и плоскости

Аналитическое и программное решение задачи о взаимном положении прямой и плоскости (рис. 18) приведено в табл. 6. При этом $KM \equiv (a_1, b_1)$; $BCD \equiv (a, b, c)$.

Сценарий ролика, визуализирующего это решение, имеет следующий вид.

- Вывести сообщение: «Выберите тип вспомогательной плоскости: горизонтально-проецирующую или фронтально-проецирующую».
- Пользователь в интерактивном режиме выбирает нужный переключатель (OptionButton) и завершает процедуру выбора плоскости нажатием клавиши ОК.
- Вывести рассчитанные координаты точки пересечения в окне редактора.

- На соответствующей плоскости проекций появятся проекции точек пересечения вспомогательной плоскости с заданной.
- Акцентировать полученные точки. Провести линии связи из полученных точек до соответствующих отрезков на другой проекции.
- Акцентировать точки пересечений линий связи отрезков, задающих плоскости.
- Соединить полученные точки отрезком.
- Акцентировать искомую точку на пересечении этого отрезка и проекции исходной прямой.
- Получить проекцию точки пересечения на другой плоскости проекций, проведя линию связи.
- Акцентировать искомую точку пересечения.
- Обозначить наблюдателя для определения конкурирующих точек.
- Акцентировать точку, находящуюся ближе к наблюдателю.

Таблица 6

Аналитическое и программное решение задачи о взаимном положении прямой и плоскости

Шаг	Математическая модель	Программная реализация в Action Script 3
1	Получаем уравнения плоскости в параметрической форме $p(t) = a + (b-a)t + (c-b)\tau = a + Vt + W\tau$; и в нормальной форме $(p - p_0) \circ N = 0$, где $p_0 = a, N = V \times W$, уравнение прямой в параметрическом виде: $p(t) = a_1 + (b_1 - a_1)t = p_{10} + V_1 t$.	Vi = bi.subtract(ai); Wi = ci.subtract(bi); Ni = Point3D.calcVector(Vi,Wi); pi0 = ai;

2	Проверяем прямую на параллельность плоскости: $V_1 \circ (V \times W) = O_3$; на ортогональность: $(p_0 - p_{10})(V \times W) = 0$; совпадение: $\begin{cases} V_1 \circ (V \times W) = O_3 \\ (p_0 - p_{10})(V \times W) = 0 \end{cases}$;	V1=b1.subtract(a1);
		p10=a1;
		p0 = a;
		V = b.subtract(a);
		W=c.subtract(b);
		N=Point3D.calcVector(V, W);
		isParralel = Point3D.isNull(Point3D.calcScal (V1, N));

Окончание табл. 6

Шаг	Математическая модель	Программная реализация в Action Script 3
	и пересечение, аналогично пересечению прямых: $p_{10} + V_1 t_1 = p_{20} + V_2 t_2 \Rightarrow [t_1\ t_2] = (p_{20} - p_{10}) \begin{bmatrix} V_1 \\ -V_2 \end{bmatrix}^{-1}$ $\Rightarrow q = p_{10} + V_1 t_1$ для которого условия существования решений имеют вид $\left\| N_1^T\ N_2^T \right\| \neq 0$ или $\begin{bmatrix} V_1 \\ V_2 \end{bmatrix} \neq 0$, то есть равносильны непараллельности прямых.	isOrtog=Point3D.isNull (Point3D.calcScal(p0.subtract (p10), N)); if (isParralel && isOrtog) //совпадают if (isParralel && !isOrtog) //параллельны if (!isParallel && isOrtog) //ортогональны if (!isParallel && !isOrtag) //пересекаются
3	Точка пересечения q плоскости и прямой ищется по формуле: $q = p_{10} + V_1 t = p_0 + V\tau + W\theta$, где $[t \quad \tau \quad \theta] = (p_0 - p_{10}) \begin{bmatrix} V_1 \\ -V \\ -W \end{bmatrix}^{-1}$ Условие существования решения: $V_1 \circ N \neq 0$	matrix = new Matrix3D(V1.x,V1.y,V1.z,1, -V.x,-V.y,-V.z,1, -W.x,-W.y, -W.z,1, 0,0,0,1); matrix.invert(); t = Point3D.calcScal(p0.subtract (p10),new Point3D(matrix[0], matrix[4], matrix[8])); q = a.add(V1.multiply(t));

Определение взаимного расположения двух плоскостей

В задачах начертательной геометрии плоскость принято задавать следующими способами: тремя точками, двумя параллельными либо пересекающимися прямыми, прямой и точкой. Если выбран способ задания плоскости тремя точками, следует проверить соблюдение зависимости: $(b - a) \times (c - b) \neq O_3$, то есть не принадлежности точек одной прямой. Параметрическая форма уравнения плоскости проходящей через три точки a, b и c имеет вид

$$p(t) = a + (b - a)t + (c - d)\tau = a + Vt + W\tau. \quad (7)$$

Нормальная форма

$$(p - p_0)N = 0, \text{ где } p_0 = a, N = V \cdot W \quad (8)$$

Аналитическое и программное решение задачи о взаимном положении двух плоскостей (рис. 19) описано в табл. 7. Плоскость ABC обозначена как (a_1, b_1, c_1), а FGE как (a_2, b_2, c_2).

Сценарий ролика, визуализирующего это решение, имеет следующий вид.

- Провести вспомогательную горизонтально/фронтально проецирующую секущую плоскость γ1 (для решения задачи необходимо провести две вспомогательные плоскости γ1 и γ2).
- Построить линии пересечения вспомогательной и двух заданных плоскостей (эти две линии принадлежат различным плоскостям и общей для них секущей плоскости).
- Обозначить точку пересечения двух этих прямых – K.
- Найти и обозначить другую проекцию найденной точки. Эта точка (две ее проекции) принадлежит искомой линии пересечения двух плоскостей.
- Повторить для γ2 пункты 2–4. Найти T.
- Обозначить искомую линию KT.

Таблица 7

Аналитическое и программное решение задачи о взаимном положении двух плоскостей

Шаг	Математическая модель	Программная реализация в Action Script 3
1	Получаем уравнения плоскостей в параметрической $p_i(t) = a_i + (b_i - a_i)t + (c_i - b_i)\tau = a_i + V_i t + W_i \tau$ и в нормальной форме $(p - p_{i0}) \circ N_i = 0$, где $p_{i0} = a_i, N_i = V_i \times W_i$, где $i = \{1, 2\}$	Vi = bi.subtract(ai); Wi = ci.subtract(bi); Ni = Point3D.calcVector(Vi,Wi); pi0 = ai;

2	Проверяем плоскости на параллельность: $(V_1 \times W_1) \times (V_2 \times W_2) = O_3$
	на ортогональность: $(p_{20} - p_{10})(V_2 \times W_2) = 0$
	Совпадение: $\begin{cases}(V_1 \times W_1) \times (V_2 \times W_2) = O_3 \\ (p_{20} - p_{10})(V_2 \times W_2) = 0\end{cases}$
	и пересечение
		isParralel = Point3D.isNull(Point3D.calcVector(N1, N2));
		isOrtog=Point3D.isNull(Point3D.calcScal(p20.subtract(p10), N2));
		if (isParralel && isOrtog) //совпадают
		if (isParralel && !isOrtog) //параллельны
		if (!isParallel && isOrtog) //ортогональны
		if (!isParallel && !isOrtag) //пересекаются

3	Уравнение линии пересечения плоскостей $p(t) = p_0 + Vt$ ищется по формуле $p_0 = -\begin{bmatrix} -p_{10} \circ N_1 & -p_{20} \circ N_2 \end{bmatrix} \begin{bmatrix} N_1 \circ N_1 & N_1 \circ N_2 \\ N_1 \circ N_2 & N_2 \circ N_2 \end{bmatrix}^{-1} \begin{bmatrix} N_1 \\ N_2 \end{bmatrix}$ $V = N_1 \times N_2$	V = Point3D.calcVector(N1, N2);

6.6. Разработка иерархии объектов, интерфейсов классов и методов библиотеки создания интерактивных роликов

В силу специфики поставленной задачи и описаний, полученных от специалиста по начертательной геометрии, в работе не реализована классическая иерархия объектов «точка – прямая – плоскость – поверхность». Этот подход обоснован тем, что:

1) работа ведется с тремя плоскостями проекций;
2) необходимо разделение двумерного пространства и трехмерного.

Библиотека создания интерактивных роликов состоит из двух больших пакетов классов:

- com.kr.epyur.core – ядро библиотеки. В нем собраны классы, отвечающие за хранение и преобразование информации о геометрических примитивах, а также визуализацию этой информации, и классы, обеспечивающие обмен информацией между ядром и любыми внешними классами;
- com.kr.epyur.problems – классы, каждый из которых представляет собой решение некоторой задачи из курса «Начертательная геометрия».

Для визуализации объектов реализовано наследование. Пакет com.kr.epyur.core включает в себя три пакета классов:

- com.kr.epyur.core.math – классы-объекты и класс-утилита, для хранения и обработки информации о стандартных геометрических примитивах;
- com.kr.epyur.core.draw – классы для визуализации объектов на плоскостях проекций эпюры, а также вспомогательные классы для хранения объектов;
- com.kr.epyur.core.controller – классы для диспетчеризации событий между ядром и внешними классами.

Введем понятие «мира чертежа» (представлен классом EpyurWorld). Определим его как среду, в которой существует решение некоторой задачи начертательной геометрии (рис. 20).

Эта среда обладает следующими параметрами:

- холст, на который добавляются визуальные объекты (canvas), необходимые для визуализации решения;
- ширина и высота «холста» (width, height);
- точка начала координат (origin);

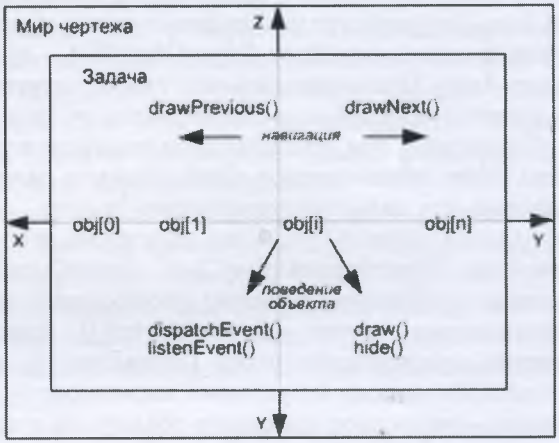

Рис. 20. Существование задачи в «мире чертежа»

- масштаб чертежа (scale);
- координатные оси (axes).

В этой среде существует некоторая абстрактная задача (EPSolution), для которой характерны:

- начальные и дополнительные данные о геометрических объектах;
- методы преобразования данных в массив объектов (obj), подлежащих отрисовке (экземпляры EpyurObjectClass);
- методы навигации по массиву объектов (drawNext(), hidePrevious()).

Иерархия объектов, подлежащих отрисовке, представлена в табл. 8.

Таблица 8

Иерархия объектов для отрисовки

EpyurClassesDraw	Любой визуальный объект, который может быть добавлен на холст	
	EpyurAxesDraw	Класс для отрисовки осей
	EpyurObjectClass	Любой визуальный объект, который может быть как добавлен на холст «мира», так и быть частью другого объекта – экземпляра класса EpyurClassesDraw

EPSolution содержит массив экземпляров подклассов класса EpyurObjectClass. Для удобства навигации по этому массиву создан вспомогательный класс-контейнер EpyurObjectSet – обертка стандартного класса Array. При активации некоторого элемента массива возможны варианты поведения в зависимости от типа этого элемента. Это обусловлено тем, что методики решения позиционных и метрических задач предполагают необходимость запроса дополнительных данных для дальнейшего решения задачи.

Подклассы класса EpyurObjectClass делятся на два типа:

- *событийные* (EpyurInputActionClass, EpyurQuestionClass) – отправляют сообщение (событие) о необходимости введения дополнительных данных (dispatchEvent()), прослушивают сообщение с результатом ввода (listenEvent()), дополняют либо изменяют массив визуальных элементов.
- *рисующие* – добавляют проекцию объекта на холст (draw()) либо удаляют ее (hide()).

Примером активации событийного класса EpyurInputActionClass служит задача определения натуральной величины треугольника, для которой окно редактора выглядит, как показано на рис. 21.

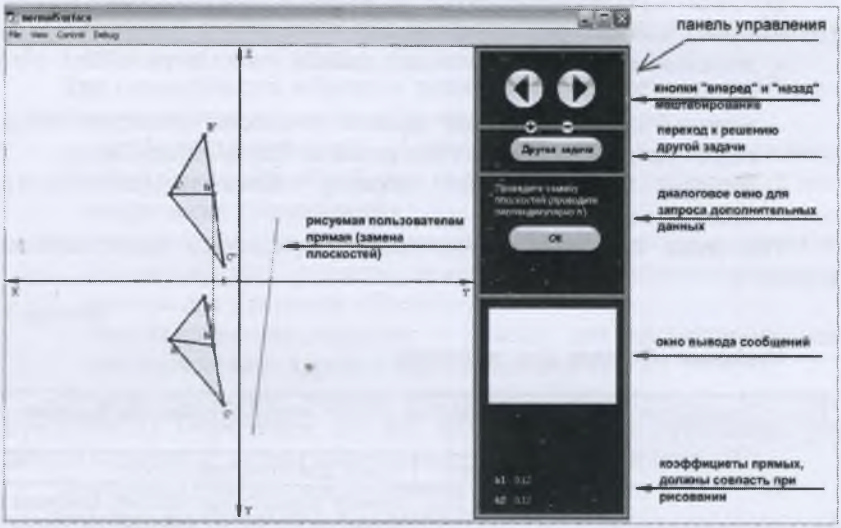

Рис. 21. Задача определения натуральной величины треугольника. Запрос ввода дополнительных данных

В окне приложения справа на панели появится соответствующее диалоговое окно с запросом. Пользователь самостоятельно отрисовывает вспомогательную плоскость, которая вырождается в прямую. Координаты отрисованного отрезка прямой считываются с экрана. При этом работать приходится с двумерным представлением объекта. Трехмерный объект формируется только для расчетов [55], по мере необходимости.

Подклассы класса EpyurObjectClass – классы визуализации объектов плоскостей проекций эпюры – представляют собой контейнеры, содержащие команды Flash. Так, для отрисовки программно полученного многоугольника метод draw(), содержит, кроме прочего, вызов Flash-функций сплошной векторной заливки замкнутой области child.graphics.beginFill(surfaceColor, 0.2) и child.graphics.endFill().

Параметры контура рассчитываются программно и хранятся в виде экземпляра класса Plane2D.

Структура ядра библиотеки создания интерактивных роликов по курсу «Начертательная геометрия» представлена в табл. 9.

Таблица 9

Структура ядра библиотеки создания интерактивных роликов по курсу «Начертательная геометрия»

<table>
<tr><td colspan="3">com.kr.epyur.core.controller</td></tr>
<tr><td>EpyurEvent</td><td colspan="2" rowspan="4">Классы для диспетчеризации событий между ядром и внешними классами</td></tr>
<tr><td>EpyurEventController</td></tr>
<tr><td>EpyurEvent Dispatcher</td></tr>
<tr><td>EpyurEventListener</td></tr>
<tr><td colspan="3">com.kr.epyur.core.draw</td></tr>
<tr><td rowspan="3">EpyurClassesDraw</td><td colspan="2">Любой визуальный объект, который может быть добавлен на холст</td></tr>
<tr><td>Epyur Axes Class</td><td>Класс для отрисовки осей эпюры</td></tr>
<tr><td>Epyur Object Class</td><td>Любой визуальный объект, который может быть добавлен как на холст «мира», так и быть частью другого объекта – экземпляра класса EpyurClassesDraw</td></tr>
</table>

Продолжение табл. 9

Визуализаторы		
	Corner	Отображение прямого угла
	Looker	отображение направления взгляда
	EpyurPointClass	Проекция точки
	EpyurLineClass	проекция отрезка прямой
	EpyurPlaneClass	Проекция плоскости
	Epyur SphereClass	Проекция сферы либо окружности
Визуализаторы		
	EpyurPrism SurfaceClass	Проекция призмы
	EpyurPiram idSurfaceClass	Проекция пирамиды
Событийные		
	EpyurInput ActionClass	Класс, отправляющий событие на добавление либо изменение данных с помощью выбора элементов на эпюре либо с помощью рисования объектов (например, проведение замены плоскостей)
	Epyur Quection Class	Класс, отправляющий событие на добавление либо изменение данных с помощью выбора из списка одного из предложенных вариантов
Общие		
ExportClass	Класс, экспортирующий символы из флеш-файла ресурсов.	
EpyurWorld	Класс, представляющий «мир чертежа»	
EpyurCanvas	Класс, представляющий глобальный холст (обертка, он же паттерн Decorator [16], flash.display). Представляет собой Sprite для реализации паттерна Singleton.	
Вспомогательные		
EpyurObjectSet	Рабочий класс – массив визуальных объектов (экземпляров подклассов EpyurObjectClass)	

Окончание табл. 9

com.kr.epyur.core.math	
PointSet	Рабочий класс – массив точек
LineSet	Рабочий класс – массив прямых
Point3D	Класс для работы с 3-х мерными координатами
Point2D	Класс для работы с двумерными координатами (координаты эпюры)
PointViewType	Класс является перечислением значений констант, с помощью которых задается тип точки
Line3D	Класс для работы с линией в трехмерных координатах
Line2D	Класс для работы с линией в двухмерных координатах (координаты эпюры)
LineViewType	Класс является перечислением значений констант, с помощью которых задается тип прямой
Plane3D	Класс для работы с плоскостью в пространстве
Plane2D	Класс для работы с проекцией плоскости (координаты эпюры)
Circle3D	Класс для работы со сферой либо окружностью в пространстве
Circle2D	Класс для работы с окружностью на плоскости (координаты эпюры)
PiramidSurface	Класс для работы с 3х гранной пирамидой
PrismSurface	Класс для работы с 3х гранной призмой

В процессе написания данной работы было создано более двенадцати роликов, визуализирующих решение задач начертательной геометрии. Классы, формирующие программируемую анимацию и решение этих задач, описаны в табл. 10.

Таблица 10

Классы визуализаций и решения задач начертательной геометрии

com.kr.epyur.problems		
EPSolution	*Класс*-родитель	
	EPLine	Задача о построении эпюры прямой
	EPLineInProjection	Задача о переводе прямой общего положения в проецирующее

Окончание табл. 10

com.kr.epyur.problems		
	EPLinesCrossing	Задача определения взаимного положения двух прямых и определения видимости конкурирующих точек
	EPNormalLine	Задача о нахождении натуральной величины отрезка методом замены плоскостей
	EPNormalPoint	Метод замены плоскостей на примере точки
	EPNormalSurface	Задача о нахождении натуральной величины треугольника методом замены плоскостей
	EPPiramidPlaneCrossing	Задача о нахождении пересечения пирамиды и плоскости
		Задача о нахождении пересечения пирамиды и сферы
	EPPlaneInProjection	Задача о переводе плоскости общего положения в проецирующее
	EPPlaneLineCrossing	Задача о нахождении точки пересечения прямой и плоскости
	EPPlanesCrossing	Задача об определении относительного положения двух плоскостей и нахождении их линии пересечения
	EPPoint	Задача о построении эпюры точки
	EPPrismPlaneCrossing	Задача о нахождении пересечения призмы и плоскости
	EPRotationLine	Задача нахождения натуральной величины отрезка с использованием метода вращения вокруг оси

Блок-схема (рис. 22) отражает порядок следования диалоговых и предопределенных процедур типичного интерактивного ролика.

Пояснения к блок-схеме

Ввод данных – окно ввода координат объектов.

Корректность – например, в случае задания плоскости тремя точками точки проверяются на компланарность, при задании пересекающимися прямыми – на пересечение.

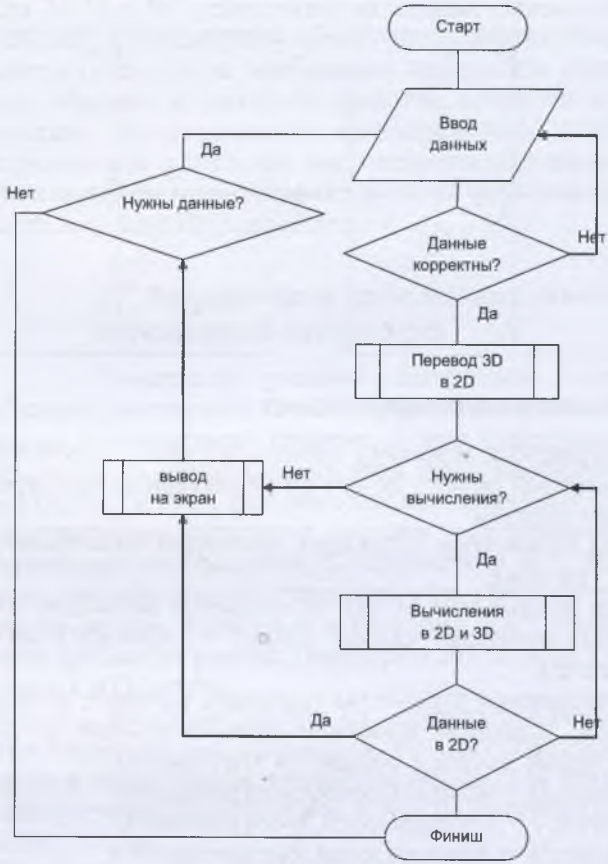

Рис. 22. Порядок следования диалоговых и предопределенных процедур типичного интерактивного ролика

Перевод в 2D – перевод 3-мерных координат, задающих объекты, в представление, подходящее для отрисовки, проводится по принципу, изображенному на рис. 23. Слева представлена эпюра, справа – трактовка эпюры в виде ПДСК.

Вычисления – применение известных методов и алгоритмов для получения результата.

Необходимость в дополнительных данных зачастую возникает при решении метрических задач. Пользователю предоставляется возможность провести горизонталь или фронталь, заменить плоскости либо выбрать дальнейший ход решения.

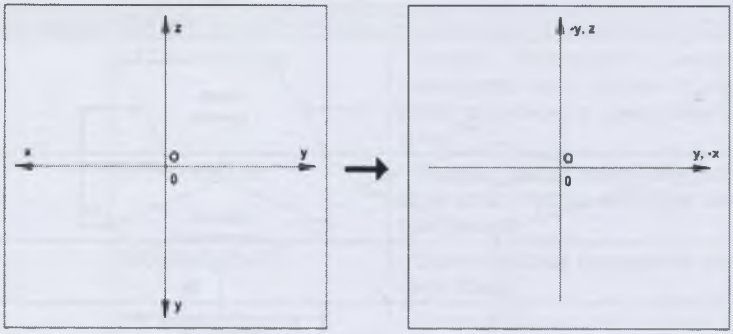

Рис. 23. Принцип перевода координат

В абстрактных понятиях процесс от ввода параметров, задающих некоторый объект, до его визуализации можно представить следующим образом:

- ввод параметров, задающих некоторый геометрический объект (params);
- если данные корректны, то создается экземпляр (object) соответствующего класса: object = AbstractGeom3DObject (params);
- вычисляются параметры (paramsH, paramsV, paramsW) объектов, задающих проекции данного объекта на плоскости проекций эпюры и создаются эти объекты:
  - object.H = AbstractGeomObject(paramsH);
  - object.V = AbstractGeomObject(paramsV);
  - object.W = AbstractGeomObject(paramsW);
- для нужных проекций объекта, например, горизонтальной, создается экземпляр (drawObject) класса EpyurAbstractClass: drawObject = new EpyurAbstractClass(object.H);
- проекция добавляется на сцену flash-ролика вызовом метода draw() у экземпляра drawObject: drawObject.draw();
- при необходимости удаления (актуально для реализации функции отката) вызывается drawObject.delite().

Рассмотрим использование библиотечных символов на примере визуализации точки.

Создаем точку: var point3D = new Point3D(10, 20, 30).

Создаем экземпляр класса отрисовки:

var drawPoint = new EpyurPointClass(point3D.H);

Вызываем drawPoint.draw() для добавления на сцену.

Для V, H и W существуют отдельные экземпляры двухмерных классов, и трехмерный объект не создается. Исключение составляют случаи, когда необходимо определить взаимное расположение объектов и другие их свойства, в том числе полученные в результате интерактивного взаимодействия с пользователем. Для определения отношения «пересечения, скрещивание, включение» реализована, например, статическая функция класса Line3D. crossing(Line1:Line3D, Line2:Line3D).

6.7. Разработка и наполнение оболочки обучающего веб-ресурса

Размещение учебных материалов в сети Интернет способствует улучшению качества образования по начертательной геометрии и инженерной графике. В web существует немало полезных ресурсов, посвященных этой теме, например, уже упомянутые [13; 55; 73; 100]. Все они содержат обширные теоретические и иллюстративные материалы, системы тестирования и наглядные анимированные изображения. К недостаткам всех перечисленных ресурсов следует отнести невозможность самостоятельного сформирования задачи, возврата к некоторому шагу решения, задания скорости просмотра ролика. Некоторые обучающие курсы требуют установки CAD-систем.

Работа с роликами требует от студента предварительной проработки теоретических положений, поэтому обязательным этапом обучения и структурным элементом обучающей программы является набор стандартных роликов и тексты описания методик решения задач.

Представляемый веб-ресурс реализован в виде Flex-оболочки, которая содержит следующие инструменты:

- теоретические материалы;
- типовые примеры в виде неинтерактивных роликов;
- графический редактор для решения задач;
- руководство пользователя.

Для наполнения этой оболочки используются xml-файлы, содержащие ссылки на html- и swf-файлы [62]. Flash является традиционным способом реализации анимации в веб-приложениях, поскольку размер выходного файла минимальный из всех возможных инструментов и технологий (Microsoft PowerPoint, Camtasia, Wink, Pinnacle, iSpring, Flash, gif анимация). Для реализации этой оболочки выбран Flex, потому как он является расширением функ-

циональности Flash-плейера и включает инструменты для быстрого создания Rich Internet Applications (RIA). RIA-приложения могут передавать веб-клиенту необходимую часть пользовательского интерфейса, оставляя большую часть данных (ресурсы программы, данные и пр.) на сервере, запускаться в браузере и, не требуя дополнительной установки программного обеспечения, запускаться локально в среде безопасности, называемой «песочница» (sandbox). Созданное приложение размещено в сети Интернет и доступно для самостоятельного использования студентами в качестве средства дистанционного обучения.

На главном меню расположены закладки (рис. 24):

- Теоретические материалы;
- Типовые примеры – предоставляет доступ к роликам, отражающим ход решения задач на предопределенных данных (неинтерактивные ролики);
- Задачи – визуализация решения определенной задачи на пользовательских данных. Здесь реализован доступ к графическому редактору, содержащему интерактивные ролики;
- Справка – содержит инструкции по использованию приложения, форму для отправки вопросов разработчикам, ссылки на ресурсы по начертательной геометрии.

К сожалению, из-за того, что Flex несовместим с ActionScript 2.0, использовать ролики и библиотеки, написанные с его использованием в оболочке, не представляется возможным.

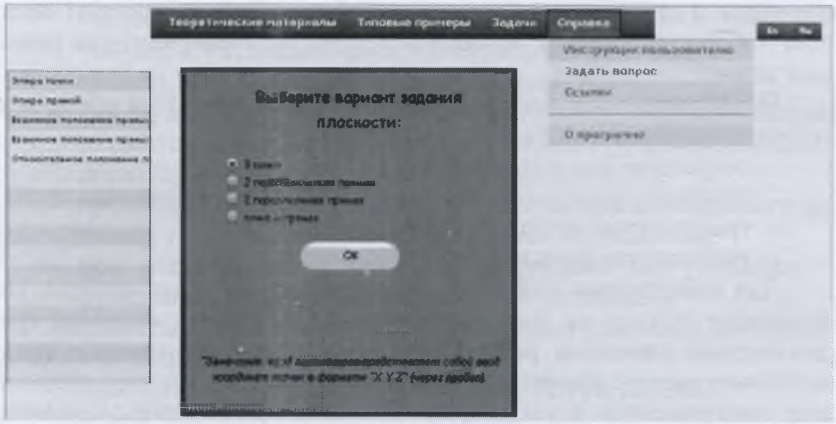

Рис. 24. Окно приложения с редактором формирования ролика. Задача о взаимном положении прямой и плоскости

Приложение поддерживает переключение английского и русского языков отображения меню, справки и теоретических материалов.

Инструментами создания и ведения электронного учебного интернет-ресурса по курсу «Начертательная геометрия» являются микроархитектурный фреймворк Cairngorm, графический редактор Flash, XML, стилевые таблицы XSL и XSLT для форматирования и отображения на различных аппаратных ресурсах.

Микроархитектурный фреймворк Cairngorm предназначен для средних и крупных Flex RIA приложений. Cairngorm представляет собой адаптацию для Flex известных в Java архитектурных решений. Он использует следующие паттерны: Singlenton, Command, Delegate, ValueObject, Observer, Service to Worker. Использование Cairngorm позволяет на практике применить подход «Модель–Представление–Контроллер» (MVC) к проектированию приложений.

Модель отражает состояние приложения, хранит данные с сервера либо локальные данные (реализуется cairngorm ModelLocator).

Представление – это пользовательский интерфейс, представленный компонентами mxml и контролами. Данные из модели подставляются в представление через binding (связывание данных или автоматическая синхронизация).

Контроллер отвечает за взаимодействие между уровнями приложения, связывает команды и события. События в Cairngorm отвечают за передачу данных в приложении. Дерево классов, использованных для написания оболочки, представлено на рис. 25.

Приложение работает следующим образом. В Controller регистрируется некоторое событие (связывается с некоторой командой). В ответ на действия пользователя вызывается (вещается) событие. Controller вызывает метод execute() нужной команды. Команда либо обращается за данными к серверу с помощью BuisenessDelegate, и результаты обрабатываются в методе result() команды, либо в execute() производятся вычисления. Изменяется/обновляется модель через ModelLocator. Изменяется представление.

Наполнение приложения-оболочки контентом осуществляется без вмешательства в программный код. Ресурсными являются файлы типа html, css, и swf. Ссылки на них содержатся в базе данных, которую составляют файлы swfMenu.xml и htmlMenu.xml.

Для редактирования оболочки созданные html- или swf-файлы указываются в соответствующих тегах xml-файлов по образу уже реализованных в системе (рис. 26). Этот файл отвечает за наполне-

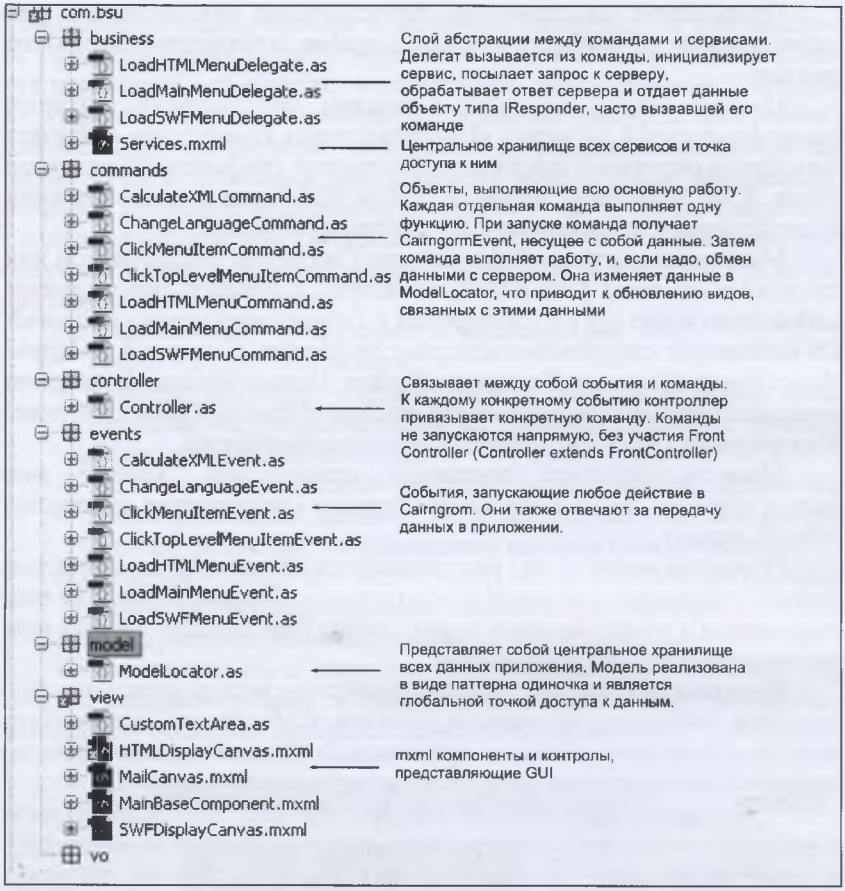

Рис. 25. Дерево классов оболочки электронного учебного ресурса

ние оболочки теоретическими материалами и справкой по использованию системы. Для добавления необходимо дописать строку с тегом <item label="[some_label]" link="[some_link]"/>, где [some_label] – название страницы, которое будет отображаться в списке, [some_link] – относительный путь к html-странице относительно swf-файла приложения. Например, < item label=»Комплексные чертежи точки» link=»assets/html/EpyurPoint.html»/>. Эта строка размещается в теле элемента <en></en> либо <ru></ru>. Элемент <guide/> содержит ссылки на файлы справки, элемент <theory/> – на теорию. Теория разделена на позиционные и метрические задачи.

```
<?xml version="1.0" encoding="utf-8"?>
<htmlmenu>
<theory>
<positional>
<ru>
<!-- < item label= "Комплексные чертежи точки" link="assets/html/EpyurPoint.html"/>. -->
<item/>
</ru>
<en>
<item/>
</en>
</positional>
<metrical>
<ru>
<!-- < item label = "Натуральная величина треугольника" link="assets/html/ Tri-
angleNorm.html"/>. -->
<item/>
</ru>
<en>
<item/>
</en>
</metrical>
</theory>
<guide>
<ru>
<!-- < item label = "Руководство пользователя" link="assets/html/ HelpFirst.html"/> -->
<item/>
</ru>
<en>
<item/>
</en>
</guide>
<htmlmenu>
```

Рис. 26. Структура htmlMenu.xml

Файл swfMenu.xml хранит ссылки на flash-ролики. Добавление новых роликов аналогично способу, описанному выше. Элемент <problems/> представляет динамические ролики, эле-

мент <routine/> – статические, отражающие решение типовых примеров.

На рис. 27–32 представлены скриншоты описанного веб-ресурса. Рис. 27 демонстрирует главную страницу электронного курса по начертательной геометрии http:// kreasl.org/geometry/. Отображение теоретического материала из раздела «Ортогональные построения» представлено на рис. 28. На рис. 29 в окне графического редактора веб-ресурса отображается задача на тему «Пересечение плоскостей», решенная на пользовательских данных. Результат работы неинтерактивного ролика, демонстрирующего построение линии пересечения поверхностей с помощью метода секущих сфер представлен на рис. 30. На рис. 31 отражен процесс ввода исходных данных задачи о взаимном положении прямых. На рис. 32 приведен результат решения этой задачи в окне редактора и элементы его пользовательского интерфейса.

Графический редактор, представляющий главный интерес в описываемой работе, обладает рядом недостатков и требует доработки. К ним следует отнести нереализованность задач на пересечение поверхностей, невозможность переносить надписи (поэтому они отчасти бывают перекрыты изображением), в отдельных случаях обработка исключений и процедуры отката не до конца доработаны. Исправление недоработок и активное внедрение пакета программ в учебный процесс позволят получить электронный учебник и веб-ресурс по курсу «Начертательная геометрия», который будет особенно полезен для самостоятельной работы студента, для преподавателя при проведении лекций и практических занятий и создании презентационных и методических материалов.

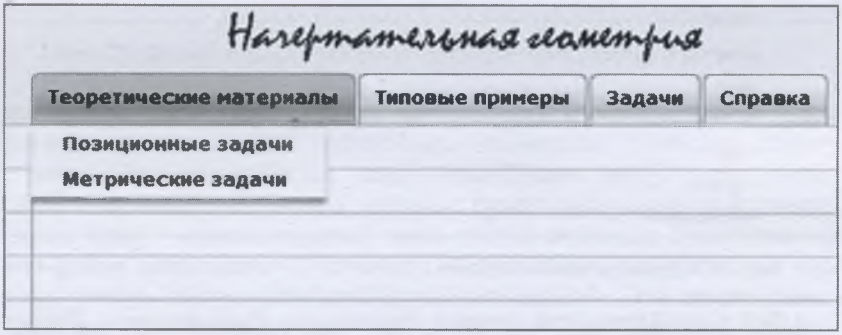

Рис. 27. Главная страница электронного курса по начертательной геометрии http://kreasl.org/geometry/

§ 10. ПРОЕКЦИИ ОТРЕЗКА ПРЯМОЙ ЛИНИИ

Положим, что даны фронтальные и горизонтальные проекции точек *A* и *B* (рис. 45). Проведя через одноименные проекции этих точек прямые линии, мы получаем проекции отрезка *AB* - фронтальную (*A"B"*) и горизонтальную (*A'B'*).

Можно ли утверждать, что такой чертеж (рис. 45) выражает именно отрезок прямой линии? Да; если представить себе (рис. 46), что через *A'B'* и через *A"B"* проведены проецирующие плоскости (т.е. перпендикулярные соответственно к π_1 и к π_2), то в пересечении этих плоскостей получается прямая и ее отрезок *AB*. При этом точка, заданная своими проекциями на *A'B'* и на *A"B"*, принадлежит отрезку *AB*.

На рис. 47 дан чертеж отрезка *AB* в системе π_1, π_2, π_3. Проекции *A'''* и *B'''* построены так, как это было показано на рис. 18 для одной точки *A*.

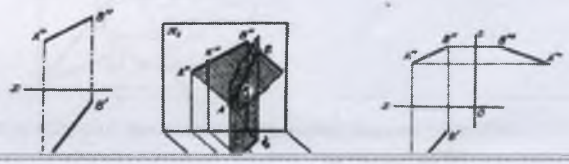

Рис. 28. Отображение теоретического материала из раздела «Ортогональные построения»

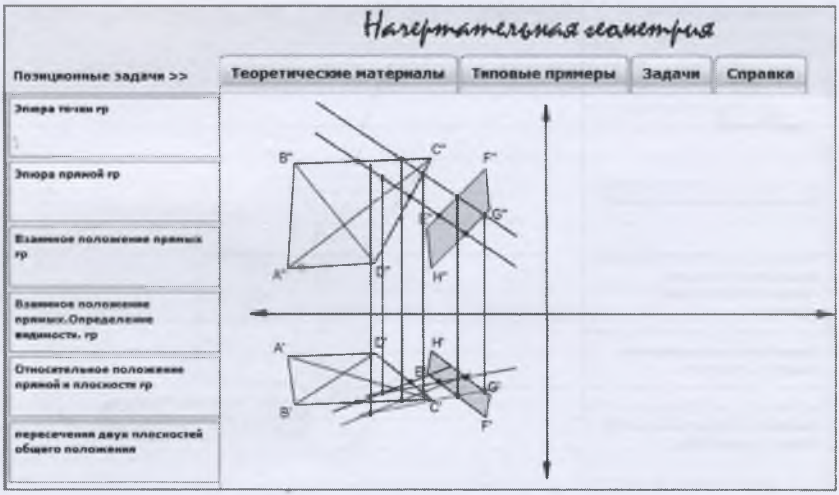

Рис. 29. Интерактивный ролик «Пересечение плоскостей»

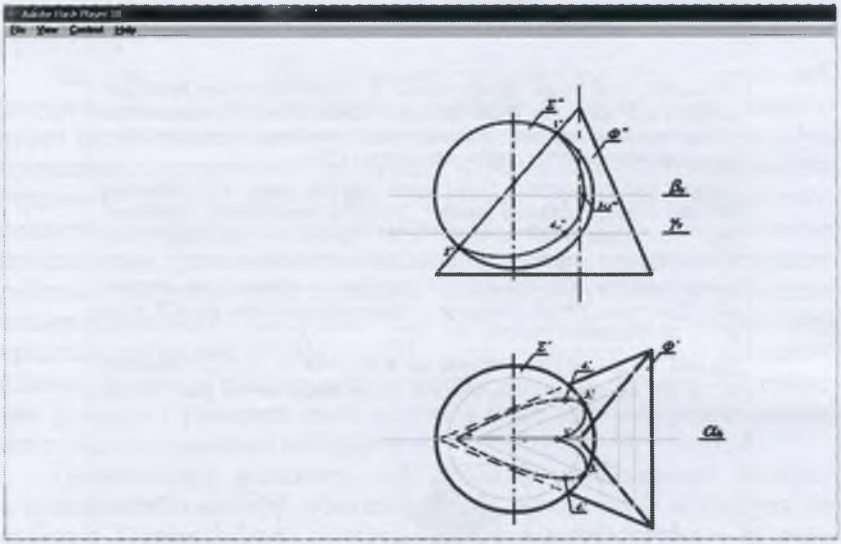

Рис. 30. Неинтерактивный ролик «Пересечение поверхностей – метод секущих плоскостей»

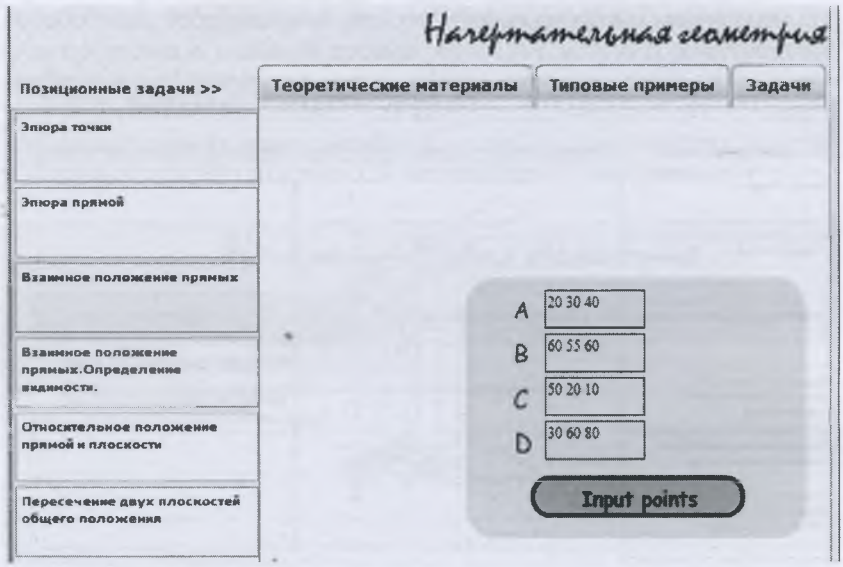

Рис. 31. Исходные данные задачи о взаимном положении прямых