Ордена Ленина
ИНСТИТУТ ПРИКЛАДНОЙ МАТЕМАТИКИ имени М.В.Келдыша
Российской академии наук

Т.А.Игнатюк, В.Е.Павловский, В.А.Прошкин

КОМПЬЮТЕРНЫЙ ПРАКТИКУМ ПО НЕБЕСНОЙ МЕХАНИКЕ. КОНЦЕПЦИЯ И СТРУКТУРА.

Препринт N ..

Москва, 2002 г.

УДК 531.1

Т.А.Игнатюк, В.Е.Павловский, В.А.Прошкин

КОМПЬЮТЕРНЫЙ ПРАКТИКУМ ПО НЕБЕСНОЙ МЕХАНИКЕ. КОНЦЕПЦИЯ И СТРУКТУРА.

АННОТАЦИЯ.

В работе изложены концепция и основные принципы организации структуры компьютерного практикума, основанного на схеме управляемой базы задач. Обсуждаются особенности программной реализации такого интегрированного приложения. Приводится описание практикума по небесной механике, разработанного на основе предложенной структуры.

Ключевые слова и выражения: компьютерный практикум, концепция и структура практикума, небесная механика, пользовательский интерфейс, визуализация движения.

T.A.Ignatuk, V.A.Proshkin, V.E.Pavlovsky

CELESTIAL MECHANICS COMPUTER TRAINING PROGRAM. CONCEPT AND STRUCTURE.

ABSTRACT.

In the paper conception and basic principles of the forming of computer training program structure based on the scheme of controlled base of tasks are stated. Features of program implementation of such integrated application are discussed. Description of the computer training program on celestial mechanics developed on the basis of the structure proposed is represented.

Key words and phrases: computer training program, computer training program conception and structure, celestial mechanics, user interface, visualization of motion.

СОДЕРЖАНИЕ

Введение.

Динамичное развитие и совершенствование компьютерных технологий за последнее десятилетие привело к тому, что мультимедийные средства стали действительно общедоступным инструментом, который в значительной мере повышает эффективность усвоения информации в процессе обучения. Современные компьютерные программы учебного назначения являются неотъемлемой частью образовательного процесса. В этом ряду особое место занимают компьютерные программы, посвященные изучению таких областей знания, где моделирование физических явлений в лабораторных условиях частично затруднено или полностью невозможно, например, в теоретической механике, механике космического полета или небесной механике.

Настоящая работа посвящена разработке структуры и концепции компьютерного практикума на основе такой информационной модели, как управляемая база (библиотека) задач.

Разработанная структура обсуждается на примере мультимедийного практикума, предназначаемого в качестве сопроводительного материала для учебного курса по небесной механике. Этот же практикум может служить как дополнительный материал по основному университетскому курсу теоретической механики. Практикум предназначен для демонстрации движения небесных тел, моделирования межпланетных перелетов, а также визуализации решений различных задач, возникающих в небесной механике. Предполагается как применение программы в компьютерном классе непосредственно на занятии, так и индивидуально, при закреплении изученного материала или при самостоятельной подготовке к сдаче экзамена.

В данной работе разработка структуры компьютерного практикума по небесной механике велась на основе ряда специальных принципов, реализующих информационную модель приложения и составляющих общую концепцию практикума. Эти принципы, среди которых можно особенным образом выделить принцип *наращиваемости*, излагаются в следующем разделе работы.

1. КОНЦЕПЦИЯ ПРАКТИКУМА.

Компьютерный мультимедийный практикум – это программа, предназначенная для обучения посредством активного вовлечения пользователя в проведение экспериментов и решение задач на основе компьютерных моделей сложных систем и явлений с использованием современных средств передачи оцифрованной видео и звуковой информации.

Компьютерный практикум по небесной механике – это пакет программ, позволяющий продемонстрировать законы и теоретические основы небесной

механики, проводить модельные эксперименты, а также предоставляющий в распоряжение пользователя инструмент для решения определенных классов задач космического полета.

Рассмотрим подробнее задачи практикума:

- демонстрация теории, базовых понятий и объектов,
- проведение экспериментов,
- решение задач.

1. Демонстрация законов может происходить двумя способами: на основе анимации или путем динамического моделирования. Анимационный клип представляет собой набор слайдов, последовательно сменяющих друг друга. Клип может сопровождаться звуковой информацией.

С одной стороны, применение анимации несет в себе целый ряд положительных моментов, например, анимационный клип может содержать видеоинформацию любого рода, практически без каких-либо ограничений, начиная, например, от схематической демонстрации законов Кеплера, в которой планета и Солнце условно представлены в виде точек, и заканчивая визуализацией реального облета космическим кораблем Земли во время исполнения гравитационного маневра. Стандартные средства просмотра клипов предоставляют удобный инструментарий, с помощью которого можно осуществлять навигацию по клипу, включая ускоренный просмотр и выбор отдельного слайда.

С другой стороны, основным недостатком анимации является то, что клип содержит неизменную информацию, которая была заложена на момент создания клипа. Таким образом, отсутствует очень важный элемент обучения – интерактивная модифицируемость программы.

Этого минуса лишены динамические модели, позволяющие вести диалог с пользователем. Например, благодаря моделированию, при демонстрации различных явлений можно варьировать такие параметры, как масштаб изображения, координаты наблюдателя и точки наблюдения, масштаб времени и другие.

2. Экспериментирование происходит на основе использования численных динамических моделей. Пользователь имеет возможность активно воздействовать на модель: в любой момент приостановить работу модели, изменить параметры, возвратиться к начальному состоянию. Необходимо предусмотреть возможность многократного повторения вычислительного эксперимента. Предлагается следующий сценарий работы с программой. Сначала формулируются предположения, в рамках которых строится модель системы. Далее объясняется какие параметры изучаемой системы можно изменять. Затем пользователь задает значения параметров, например, начальные данные или краевые условия, и проводит эксперимент. На основе предоставляемой графической и численной информации пользователь, производя, в случае необходимости, вычисления и используя ранее

накопленные теоретические знания, сверяет полученный результат с ожидаемым.

3. Практикум является удобным инструментом решения различных классов задач, таких, как краевая задача, задача достижимости, попадание в движущуюся планету, оптимальный перелет между двумя точками, расчет траекторий в приближении мгновенного гравитационного манёвра и т.д. Решение задачи происходит в рамках выбранной модели процесса путем численного экспериментирования.

Сценарий здесь таков. В качестве задания пользователю предлагается найти параметры некоторого доступного для визуализации многообразия в фазовом пространстве, пространстве координат, скоростей или каких-то других величин, каждая точка которого будет давать одно из решений некоторой задачи. Например, как известно, многообразие концов векторов скоростей, необходимых для перелета между двумя заданными точками в задаче Кеплера, представляет собой гиперболу в пространстве скоростей. Выбирая точку на гиперболе, пользователь, тем самым, задает начальную скорость, что полностью задает перелет (примеры приведены в части 3 работы).

Привлечение современных технологий преподавания, основанных на применении компьютера, позволяет решать многие методические проблемы благодаря возможности:

- обрабатывать большие объемы информации
- варьировать темп и стиль подачи информации
- моделировать сложные механические системы
- контролировать процесс усвоения знаний
- вовлечь (заинтересовать) обучаемого непосредственно в процесс обучения
- повысить эффективность и качество усвоенной информации
- сбалансировать временные затраты на исследование и обучение.

Для того чтобы практикум обеспечивал реализацию перечисленных потенциальных возможностей электронных средств обучения, программа обязана обладать, как минимум, следующими важными характеристиками:

- удобный пользовательский интерфейс,
- быстрое обучение работе с программой и легкое использование предоставляемых возможностей,
- максимальное использование вычислительной мощи компьютера – математические вычисления, графика, мультимедиа и интерактивность,
- развитая система подсказок и контекстной помощи,

- простая установка,
- гибкая система настройки оболочки и актуализации содержательной части практикума.

Условно можно выделить три основных этапа создания практикума: разработка кода, формирование состава библиотеки реализованных задач, использование созданных программных модулей. К каждому из этих этапов предъявляются определенные требования, основанные на принципах, совокупность которых составляет общую концепцию практикума.

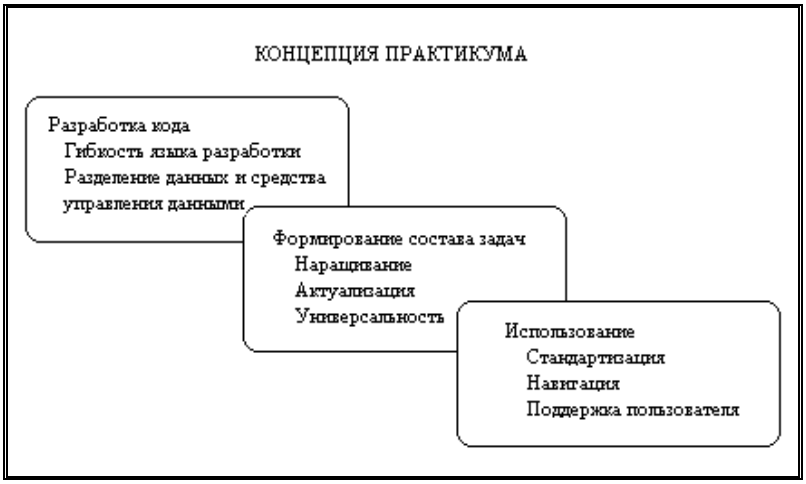


Рис.1. Концепция практикума

- Разработка кода

Возможность выбора языка программирования

Очевидно, каждый язык программирования ориентирован на решение определенного класса задач, поэтому разработчик должен иметь возможность выбора средства создания кода, оптимального для данной задачи.

Реализация задач в практикуме в виде стандартных модулей DLL, которые не зависят от выбранного средства создания программного кода, является эффективным решением поставленной задачи.

Разделение данных и средств управления данными

В структуре практикума можно обозначить две основные части – это данные и средства управления данными. В качестве данных выступают каждая задача в отдельности, база данных, файлы помощи, библиотеки функций и другие разделяемые ресурсы. Другая часть программы отвечает за манипуляцию данными пользователем. Разделение программы на две независимые части позволяет разбить сложную задачу создания и обновления практикума на более простые этапы разработки управляющей оболочки и отдельных модулей, что повышает эффективность написания программного кода.

- Формирование состава задач

Наращиваемость

На этапе создания практикума невозможно охватить весь спектр задач, которые потребуется моделировать в рамках конкретного курса. Поэтому преподавателю обязательно должна быть предоставлена возможность расширения (*наращивания*) библиотеки по собственному усмотрению, что может быть обеспечено благодаря применению автономных DLL библиотек.

Актуализация

В ходе эксплуатирования практикума могут обнаружиться неточности или ошибки при реализации задачи. Со временем задача с точки зрения преподавателя может потерять свою актуальность. Тем самым, возникает необходимость *гибкого* формирования наполнения библиотеки задач путем удаления или замены модуля без ущерба для остальных задач.

Универсальность

Реализация модулей в виде библиотек, структура которых не зависит напрямую от содержательной части, позволяет использовать в практикуме модели и задачи любых предметных областей.

- Использование практикума

Стандартизация

Максимальное использование вычислительных и мультимедийных возможностей компьютера при минимальных требованиях к ресурсам, необходимым для работы, может быть достигнуто за счет использования стандартного программного обеспечения и библиотек. Поэтому в качестве средства визуализации в практикуме взята библиотека *OpenGL*, которая, по сути, является современным графическим стандартом. Библиотека входит в состав *стандартной* поставки оболочки Windows и, следовательно, для работы практикума не требуется установки дополнительной программы, в отличие от случая, когда используется, к примеру, пакет DirectX.

Навигация

Важным аспектом использования средств обучения, основанных на применении компьютера, является возможность определения самим обучающимся скорости и последовательности изучения материала. Пользователь должен иметь возможность вернуться к ранее исследованной задаче или перейти к определенной задаче интересующего раздела, минуя остальные модули. Данная возможность предоставляется специальной *Оболочкой*, отвечающей за удобную навигацию между модулями.

Поддержка пользователя

В процессе использования программы действия пользователя должны происходить при постоянной поддержке и контроле со стороны программы.

Всякий модуль должен ставить перед пользователем определенную цель и направлять действия обучающегося соответствующим образом на основе применения развитой системы помощи и контекстных подсказок.

С учетом сформулированных условий предлагается следующая архитектура приложения. Практикум состоит из библиотеки задач, реализованных в виде автономных модулей – динамически подключаемых библиотек DLL, и управляющей оболочки – средства управления данными. Каждый модуль представляет собой файл, моделирующий какую-нибудь задачу изучаемой предметной области. Сам практикум представляет собой базу таких обучающих модулей.

Работа модулей не зависит от наличия других реализованных задач библиотеки. Подключение и удаление модулей осуществляется *Мастером установки и удаления задач*, входящим в состав управляющей оболочки. *Мастер* также обеспечивает корректную установку файлов помощи (HELP-файлов), связанных с задачами.

2. СТРУКТУРА ПРАКТИКУМА ПО НЕБЕСНОЙ МЕХАНИКЕ.

В состав практикума (см. таблицу 1) прежде всего, входят *основная управляющая оболочка* в виде исполняемого файла, *Мастер установки и удаления задач*, и *обучающие модели* в виде автономных, независимо подключаемых динамически компонуемых библиотек.

Практикум включает в себя также общие ресурсы (библиотеки дополнительных функций *OpenGL* и небесной механики (НМ)), окно «О программе…», локальную базу данных основных параметров объектов Солнечной системы, файлы справочной системы и конфигурационный файл.

Таблица 1.

компонент	файл	формат
Основной файл	Управляющая оболочка Мастер установки/удаления модулей задач	EXE
Библиотеки	*подключаемые задачи*	DLL
	библиотека функций по НМ и классов (объектов)	DLL
	библиотека OpenGL	DLL
	расширение OpenGL	DLL
	окно «О программе»	DLL
Помощь	файлы помощи	HLP
Данные	база данных	DBF
Настройка системы	конфигурационный файл	CFG

Рассмотрим подробнее каждую составляющую в отдельности.

2.1. Управляющая оболочка.

Управляющая оболочка реализуется в виде автономного исполняемого файла. Оболочка предназначена в первую очередь для запуска задач библиотеки. Помимо этого программа позволяет изменить набор доступных задач, содержит средства просмотра базы данных, а также обеспечивает доступ к справочной информации, знакомящей пользователя с программой. Работа с практикумом начинается с запуска оболочки. Один из рабочих моментов оболочки представлен на рис.2.

Главное окно программы разбито на две части. Слева предлагается список разделов курса небесной механики, справа – список задач, составляющих текущий раздел.

Выбор раздела происходит с помощью манипулятора «мышь». Задачи, непосредственно соответствующие выбранной теме, отображаются в правой части экрана в виде списка. Двойной щелчок по элементу списка приводит к запуску задачи, хранящейся в библиотеке в виде DLL файла. Выбранное приложение начинает работу в отдельном окне, основное окно программы при этом минимизируется. По завершении работы окно оболочки восстанавливает прежние размеры и положение.

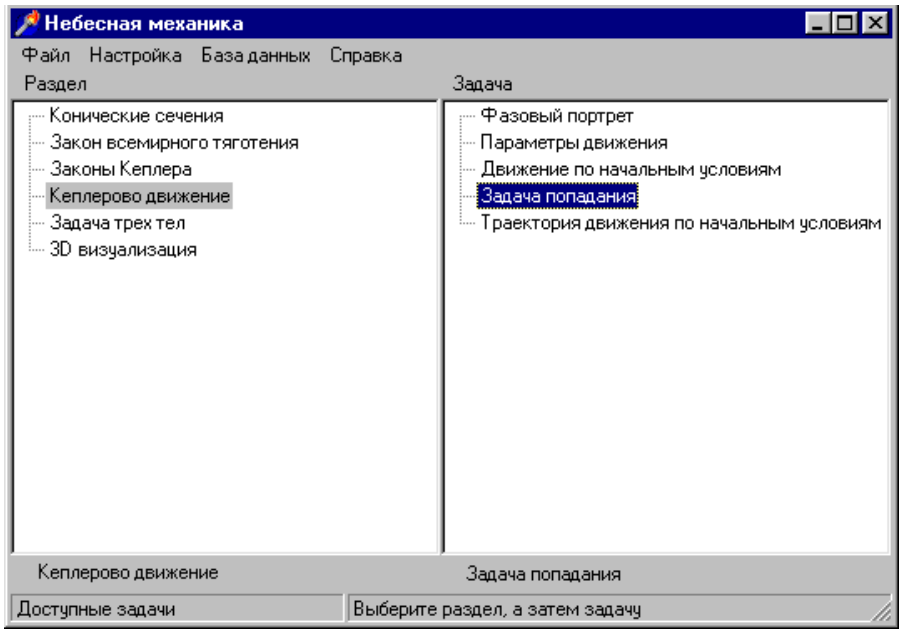


Рис.2. Главное окно приложения.

Основное окно содержит стандартное меню, позволяющее пользователю запустить Мастер установки и удаления задач, просмотреть базу данных, вызвать окно «О программе...», а также обратиться к справочной системе.

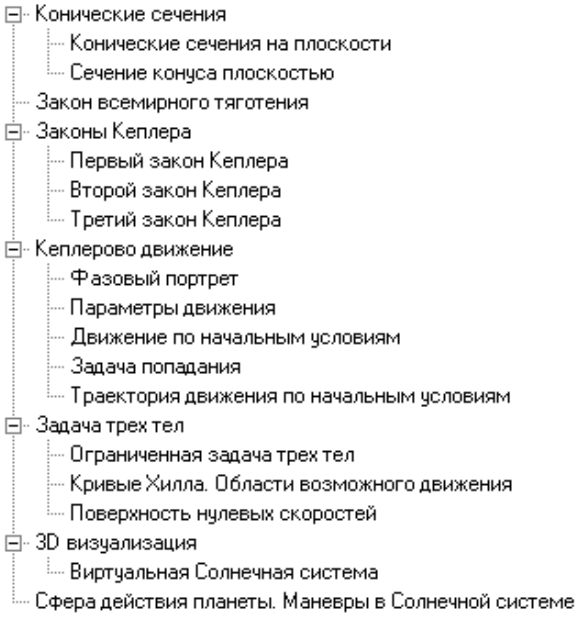


Рис.3. Двухуровневая иерархическая структура библиотеки задач.

2.2. Мастер установки и удаления задач.

Мастер установки и удаления задач позволяет изменить состав доступных пользователю задач. Мастер запускается из основной оболочки при выборе пункта меню *Настройка/Установка и удаление задач*. Программа последовательно проводит пользователя через ряд этапов, необходимых для корректного изменения состава базы задач.

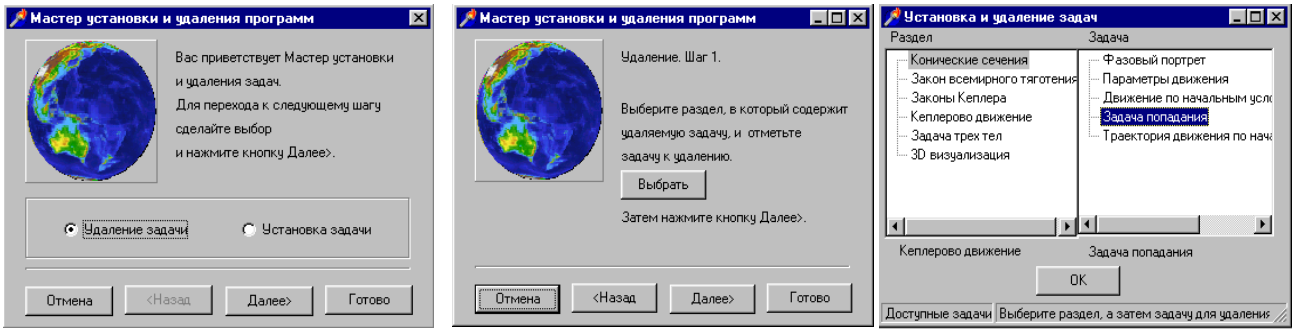
В верхней части окна каждого этапа коротко комментируется, *что* происходит на данном шаге и какие действия необходимо предпринять для перехода к следующему шагу. На любом этапе имеется возможность отмены выполняемого действия или возврата к предыдущему шагу.

На начальном этапе предлагается выбрать, какого рода действие будет производиться – удаление или установка. В зависимости от выбора пользователя Мастер действует по двум сценариям.

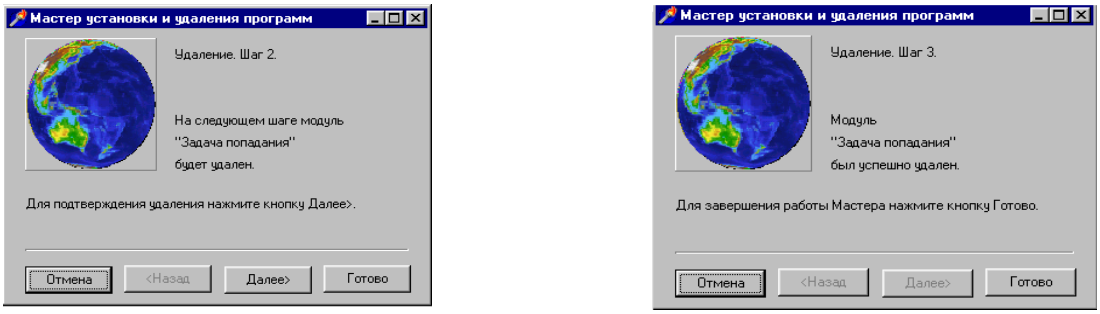
Удаление. Шаг Первый. Для выбора удаляемой задачи пользователю предлагается окно, аналогичное основному окну оболочки программы, также разбитое на две части с соответствующим наполнением. Выбор задачи к удалению происходит простым щелчком мыши.

Удаление. Шаг Второй. Программа запрашивает у пользователя подтверждение на удаление и предоставляет возможность отказаться от изменения. В случае согласия пользователя осуществляется переход к третьему шагу.

Удаление. Шаг Третий. При успешном удалении сообщается имя удаленной задачи и работа Мастера завершается. В противном случае пользователь информируется о возникших ошибках и возможных путях их устранения.



а) Начало работы Мастера **б) Шаг 1** **в) Шаг 1. Выбор модуля**



г) Шаг 2 **д) Шаг 3**

Рис.4. Этапы работы Мастера при удалении модуля.

При установке сценарий работы следующий.

Установка. Шаг Первый. Пользователю необходимо выбрать путь к директории, в которой находится устанавливаемая программа.

Установка. Шаг Второй. В случае если выбранная библиотека имеет допустимый формат и отвечает требованиям, необходимым для корректного обмена данными между оболочкой и задачей, анализируется наличие сопроводительного файла помощи и предлагается раздел небесной механики, в который будет установлена программа. По желанию пользователь может согласиться с предлагаемым по умолчанию разделом, выбрать другой или даже создать новый раздел, просто выбрав соответствующий пункт контекстного меню. Меню вызывается нажатием правой клавиши мыши в области экрана, содержащей доступные разделы.

Установка. Шаг Третий. Пользователь информируется о результате работы Мастера, в том числе о возможных ошибках, возникших в ходе установки.

В результате работы Мастера происходят соответствующие изменения в конфигурационном файле, отвечающем за настройку программы. Все действия, производимые Мастером, также могут быть осуществлены путем прямого редактирования конфигурационного файла.

2.3. Конфигурационный файл.

Конфигурационный файл предназначен для настройки программы и обеспечивает обмен данными между основной программой, подключаемыми задачами и разделяемыми ресурсами. Рассмотрим подробнее его структуру.

Файл логически разбит на несколько секций. Каждая секция начинается с зарезервированного слова, заключенного в квадратные скобки. Например, путь к директории, содержащей основной исполняемый файл, находится в разделе, начинающемся со слова *PATH*.

[PATH]
C:\CelestialMechanics

Далее расположен сектор с информацией об установленных базах данных. Каждая строка начинается с имени установленной базы, через слеш «/» указывается настоящее имя файла, содержащего базу. Ниже представлен возможный вариант наполнения секции *[DATABASES]:*

[DATABASES]
Сферы действия планет/sphere.dbf
Астродинамические постоянные планет/astro_constants.dbf
Средние элементы/elements.dbf

Информация об установленных задачах хранится в секторе *[TASKS]*. Каждая строка состоит из названия раздела небесной механики, имени задачи, представленного в правой части основного окна, и имени файла, в котором содержится задача. Например:

[TASKS]
Задача трех тел/Кривые нулевой скорости/ZeroVelocityCurves.dll
Задача трех тел/Интеграл Якоби/JacobyIntegral.dll
Законы Кеплера/Первый закон Кеплера/FirstKeplersLaw.dll
Законы Кеплера/Второй закон Кеплера/SecondKeplersLaw.dll

Конфигурационный файл может быть изменен в любом текстовом редакторе. Удаление строки, содержащей информацию о некоторой задаче, приведет к тому, что она станет недоступной в соответствующем окне задач, однако это не приведет к физическому удалению файла с диска, поэтому об уничтожении файла необходимо позаботиться отдельно. В отличие от указанного способа удаления программ, Мастер установки и удаления самостоятельно проследит за уничтожением как самого файла, так и связанных с ним ресурсов. Простая структура конфигурационного файла делает настройку системы легкой и доступной, в том числе, и для пользователей, обладающих минимальными навыками работы с компьютером.

2.4. Библиотека задач. Формирование состава, принцип пополняемости.

Ключевая роль в практикуме принадлежит задачам, хранящимся в библиотеке задач. Библиотека задач представляет собой совокупность автономных модулей, реализованных в виде динамически компонуемых

библиотек. Каждый модуль посвящен некоторой задаче небесной механики. Запуск модуля на исполнение и контроль над ходом работы происходит из *Основной оболочки*. Каждое приложение скомпоновано таким образом, что его исполнение не зависит от наличия или отсутствия других задач в библиотеке. Благодаря автономности модулей обеспечивается возможность формирования произвольного состава библиотеки, например, отвечающего требованиям, предъявляемым преподавателем к учебному курсу. На рис.5 представлены основные этапы формирования библиотеки задач практикума.

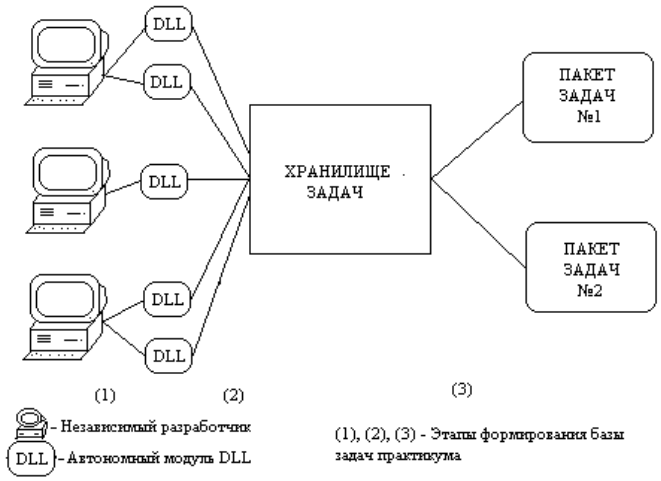


Рис.5. Формирование пакета задач практикума.

(1) На первом этапе независимые разработчики создают автономные модули, которые моделируют определенные задачи небесной механики и знакомят учащихся с наиболее важными понятиями изучаемого предмета. Разработка модулей ведется на произвольном языке программирования, поддерживающем работу с DLL файлами, в соответствии с определенными требованиями, предъявляемыми в целях итогового объединения отдельных задач в единый программный комплекс.

Приведем специальные требования к структуре выполняемых файлов. Загрузка библиотек DLL может производиться двумя способами – *явным* и *неявным* образом. В случае *неявной загрузки* система Windows при инициализации приложения автоматически загружает библиотеку, содержащую внешнюю функцию. По завершении работы приложения библиотека выгружается. При *явной загрузке* библиотека DLL загружается только по конкретному требованию, исходящему от приложения. По окончании работы с модулем, библиотека явным образом выгружается из памяти, освобождая занимаемые кодом ресурсы.

Для явной загрузки модуля *Основной оболочке* требуется имя загружаемого модуля и имя главной функции модуля, которая инициализирует его работу. Информацию об имени библиотеки *Основная оболочка* считывает из конфигурационного файла. По определению принимается, что запуск

модуля осуществляется функцией *Launch*. Таким образом, к автономному модулю предъявляется всего лишь одно существенное техническое требование – каждая библиотека обязана содержать функцию *Launch*, обеспечивающую инициализацию данных и запуск программы.
(2) На втором этапе модули, поступающие от независимых разработчиков, формируют общее хранилище задач. Далее модули тестируются на соответствие предъявленным требованиям и на корректность обмена данными с разделяемыми ресурсами практикума.
(3) Третий этап состоит в следующем. На основе приложений, отобранных на предыдущем этапе, преподавателем формируется определенный пакет программ, который наиболее полно отвечает задачам, возникающим в процессе изложения той или иной темы небесной механики. В связи с тем, что правильная работа приложения не зависит от наличия остальных модулей DLL, преподаватель может сформировать оптимальный набор задач, не вдаваясь в технические особенности реализации программ. Ресурсы хранилища задач открыты к пополнению, что обеспечивает потенциальную возможность в конечном итоге сформировать пакет задач, решающий любую учебную задачу.

Хотя модули непосредственно между собой не связаны, работа практически каждого из них построена на основе совместного использования разделяемых ресурсов. Например, ресурсы *Базы данных, Библиотеки функций по небесной механике, Библиотеки функций OpenGL* являются общими для всех приложений и могут использоваться несколькими приложениями одновременно. Корректный обмен данными между подключаемыми задачами и общими ресурсами обеспечивается главным приложением на основе конфигурационного файла. Совместное использование данных, возможное благодаря использованию динамически компонуемых библиотек, приводит к существенной экономии ресурсов компьютера.

Динамически компонуемые библиотеки DLL избавляют пользователя от необходимости полностью переустанавливать библиотеку задач в случае, если вносятся изменения или исправления только в один из модулей библиотеки. Вместо этого достаточно просто установить тот модуль, которого непосредственно коснулись изменения, без замены остальных модулей.

Состав Библиотеки задач формируется по усмотрению преподавателя в зависимости от целей, преследуемых учебным курсом. По характеру подключаемые задачи условно можно разбить на две группы:

- Анимационные
- Моделирующие

Предпочтение отдается моделирующим задачам, поскольку именно они требуют активного участия учащихся в процессе обучения, тем самым, развивая их исследовательский интерес, что, в свою очередь, является залогом эффективного усвоения изучаемого материала.

Возможность неограниченного пополнения библиотеки задач – *принцип пополняемости* – избавляет практикум от такого существенного недостатка, как ограниченность числа исследуемых задач. Положенный в основу формирования библиотеки принцип потенциальной возможности неограниченного расширения круга доступных для изучения моделей превращает практикум в эффективное средство поддержки учебных задач, возникающих в процессе преподавания небесной механики.

2.5. База данных.

База данных представляет собой набор файлов в формате Dbase. Приложение обеспечивает пользователя удобным инструментарием для извлечения информации из таблиц, редактирования данных, а также расширения базы.

Исходная версия Базы данных содержит только основные астрономические постоянные, средние элементы орбит планет Солнечной системы, параметры сфер действия и сфер притяжения планет и другие астродинамические постоянные. Однако база открыта для пополнения произвольного рода данными, и её объем ограничен исключительно ресурсами компьютера, на который установлен практикум. На рис.6 представлен фрагмент работы программы с Базой данных *Параметры движения планет.*

Данные из базы доступны для любого приложения, наделенного средствами работы с форматом Dbase. Например, подключаемый модуль *Сфера действия планеты. Маневры в Солнечной системе* считывает из базы информацию о радиусе сферы действия планеты, а модуль *Планеты Солнечной системы* извлекает из таблиц необходимые для моделирования данные о радиусе планеты и средних значениях элементов орбиты.

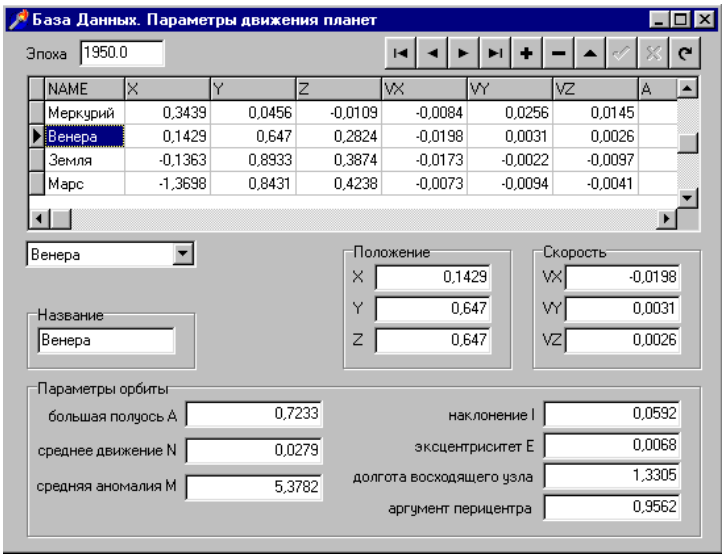


Рис. 6. Окно программы во время работы с Базой данных «Параметры движения планет».

Просмотр и редактирование данных происходит в отдельном окне. Из этого же окна можно произвести установку нового файла в базу данных. Дополнить базу можно также, сохранив таблицу формата Dbase в директории, содержащей данные, и произведя соответствующие изменения в секции *[DATABASES]* конфигурационного файла.

2.6. Библиотека функций и классов (объектов).

Библиотека функций по небесной механике предоставляет в распоряжение разработчика набор функций, предназначенных для решения специальных задач, возникающих в процессе моделирования движения объектов в поле действия центральной силы. На основе указанных процедур и функций, следуя парадигме объектно-ориентированного программирования, создаются классы (именуемые также объектами). Функции и объекты хранятся в динамически компонуемой библиотеке – CM_Functions.dll. Библиотека была создана в системе визуального объектно-ориентированного программирования Delphi, в основу которой положен язык Object Pascal. Одним из важных свойств DLL библиотек является возможность использовать функции, содержащиеся в библиотеке, созданной в любой среде разработки, поддерживающей работу с файлами DLL. Таким образом, функции библиотеки CM_Functions.dll могут быть импортированы как приложением Delphi, так и C++ или Visual Basic.

Для оптимизации работы с векторными переменными создается объект типа *Tvector*, объединяющий в себе три действительные координаты и модуль вектора. Функции, реализующие стандартные операции над векторами, такие как скалярное и векторное умножение, сложение и вычитание, умножение на число, вычисление угла между двумя векторами, вычисление расстояния между двумя точками и т.д., объединены в специально разработанной оригинальной математической библиотеке math.dll.

Рассмотрим основные функции библиотеки CM_Functions.dll.

- *function Arealconst(r,v:Tvector):Tvector;*

По заданному положению и скорости рассчитывает постоянные площадей.

- *function Laplaceconst(r,v:Tvector):Tvector;*

По заданному положению и скорости рассчитывает постоянный вектор Лапласа.

- *function p_c(c:Tvector):real;*

Исходя из постоянной площадей определяет фокальный параметр *p* конического сечения – траектории объекта

- *function e_f(f:Tvector):real;*

Исходя из постоянного вектора Лапласа, определяет эксцентриситет *e*.

- *function a_fc(f,c:Tvector):real;*

Исходя из постоянных площадей и вектора Лапласа, определяет большую полуось *a*.

- *function velocity(f,c:Tvector;teta:real):Tvector;*

Определяет скорость на основе констант движения и текущей истинной аномалии.

- *function abs_velocity(r0,v0:Tvector;fi:real):Tvector;*

Определяет скорость по начальному положению, начальной скорости и текущей угловой дальности.

- *function abs_r(r0,v0:Tvector; fi:real):Tvector;*

Определяет положение по начальному положению и скорости и текущей угловой дальности.

- *function newposition_dt(f, c, r0 :Tvector; dt:real):Tvector;*

Определяет новое положение на основе начальных данных и времени движения.

- *function time(r0,fi0,r1,fi1,alfa:real):real;*

Возвращает время, требуемое для перелета между двумя заданными положениями и углом между вектором начальной скорости и радиус-вектором начального положения. Для расчета используются конечные формулы, полученные в результате интегрирования.

- *function TIME_razl(r0,fi0,r1,fi1,alfa:real):real;*

Аналогична предыдущей функции, но вычисление интеграла ведется по формуле Симпсона.

- *function timedfi(r0,v0,r1:Tvector):real;*

Определяет время перелета при заданных начальном и конечном положениях и скорости в момент вылета.

- *function doletim(rs,rf:real;w:Tvector):boolean;*

Определяет, произойдет ли пересечение траекторией окружности заданного радиуса, если в начальный момент известна скорость вылета.

- *function find_meet_angle(rs,v:Tvector; rf:real):real;*

Вычисляет угловую дальность на траектории с заданными начальными данными при перелете в точку, принадлежащую окружности фиксированного радиуса.

- *function CircularV(r:Tvector):Tvector;*

Вычисляет вектор местной круговой скорости.

- *function get_E(teta,e:extended):extended;*

Возвращает эксцентрическую аномалию при заданном эксцентриситете и истинной аномалии.

- *function get_H(teta,e:real):real;*

Вычисляет аналог эксцентрической аномалии в эллиптическом движении для гиперболического движения.

- *function InitialE(M,e:real):real;*

Вычисляет начальное значение эксцентрической аномалии для решения уравнения Кеплера.

- *function new_E(M,e:extended):extended;*

Разрешает уравнение Кеплера относительно эксцентрической аномалии методом простых итераций.

- *function NewtonE(M,e, E0: Real): Real;*

Аналогично предыдущей функции, однако в качестве метода решения трансцендентного уравнения взят метод Ньютона.

- *function NewtonH(M,e, H0: Real): Real;*

Разрешает аналог уравнения Кеплера для гиперболического движения относительно эксцентрической аномалии, используя метод Ньютона.

На основе вышеперечисленных функций в стиле объектно-ориентированного программирования создается, наряду с другими объектами, специальный класс (набор переменных и функций, обрабатывающих эти переменные) типа *Ttransfer* -“перелет”. Данный класс позволяет значительно упростить работу в случае, когда одновременно моделируется движение нескольких независимых объектов, например, в приложениях *Виртуальная Солнечная система* или *Сфера действия планеты. Маневры в Солнечной системе*. Класс хранит все необходимые параметры перелета, такие как начальные положение и скорость, время старта, константы движения, текущее значение средней и эксцентрической аномалии, среднее движение, время перелета. Для инициализации перелета задается начальное положение и скорость, остальные параметры рассчитываются автоматически. Текущее положение определяется сдвигом по времени или путем задания угловой дальности перелета. Для корректной работы методы класса определяются в разделе *private*, таким образом обеспечивается безопасное хранение значений параметров перелета.

Класс Ttransfer.

- *constructor Create;*

Создание объекта, выделение памяти.

- *procedure SetRV(r0,v0: Tvector);*

Инициализация параметров перелета на основе начальных данных

- *procedure NewPosition(dt: real);*

Расчет положения и всех текущих параметров при перелете за время *dt*

- *procedure NewPositionDfi(dfi : real);*

Аналогичный расчет в случае, когда известна угловая дальность перелета

- *function GetV :Tvector;*

Возвращает текущую скорость.

Библиотека функций реализуется в виде динамически компонуемой библиотеки, то есть представляет собой программный модуль, содержащий код и ресурсы, которые могут совместно использоваться несколькими приложениями. Подключаемый модуль представляет собой динамически

компонуемую библиотеку DLL. Библиотека DLL позволяет приложению, в нашем случае *Основной оболочке*, загружать код, который отрабатывается во время выполнения, вместо того, чтобы компоновать его в само приложение в процессе компиляции.

Существует два метода, с помощью которых компилятор разрешает вызовы процедур и функций в выполняемом коде – это методы *статической* и *динамической компоновки*. При статической компоновке функции и процедуры модуля становятся частью конечного выполняемого файла. Расположение функций заранее известно относительно места, занимаемого в памяти программой. При *динамической компоновке* связь между вызовом функции и ее выполняемым кодом устанавливается во время выполнения приложения посредством создания внешней ссылки на конкретную функцию библиотеки DLL.

Благодаря динамической компоновке в процессе работы приложения с помощью специального механизма *отображения файла в память (memory-mapped files)* в памяти размещается только одна копия библиотеки. Это приводит к значительной экономии ресурсов, занимаемых приложениями, которые экспортируют функции из разделяемой библиотеки. Так как формат библиотеки DLL не зависит от используемого языка программирования, то функции библиотеки могут вызываться из любых приложений, написанных на любом языке, который поддерживает работу с файлами DLL. При этом следует учитывать некоторые особенности вызова функций библиотеки, связанные со спецификой помещения параметров вызываемых функций в стек. Корректный обмен данными осуществляется с помощью соответствующих директив компилятора.

Библиотека функций по небесной механике может быть пополнена по мере расширения круга рассматриваемых задач, при этом перекомпилировать необходимо только модуль, которого непосредственно коснулись изменения, без компоновки всего приложения.

Обращение к функциям и объектам библиотеки облегчает процесс разработки и сопровождения мультимедийных приложений, моделирующих движение в задачах небесной механики и помогает сконцентрировать внимание разработчика на сути изучаемого явления.

3. ПРИМЕРЫ ПОДКЛЮЧАЕМЫХ ЗАДАЧ.

С учетом сформулированных требований был разработан ряд прикладных программ-модулей, которые отвечают основным разделам классической небесной механики и касаются связанных с ними наиболее важных понятий – Кеплерово движение (законы Кеплера, конические сечения), Задача трех тел (точки либрации, области возможного движения), межпланетные перелеты (сфера действия планеты). Предлагаемые приложения лежат в основе *Базы задач*, они приводятся здесь в качестве примера, который

служит иллюстрацией разработанной методики и не претендует на полноту освещения той или иной темы. Наполнение *Базы задач* формируется по усмотрению преподавателя в соответствии с целями, преследуемыми курсом. Рассмотрим подробнее функциональные возможности программ. Приведем примеры некоторых реализованных задач.

3.1. Конические сечения.

Конические сечения играют ключевую роль в классической небесной механике. Для закрепления знаний, полученных в курсе аналитической геометрии, разработано два модуля - *Конические сечения на плоскости* и *Сечение конуса плоскостью*.

Конические сечения на плоскости.

Программа изображает коническое сечение – «траекторию космического тела» и соответствующие ему два фокуса, один из которых символизирует Солнце. Положение фокуса можно менять, просто «перетащив» его с помощью мыши. Тем самым изменяется большая полуось сечения, а малая полуось остается неизменной.

Диалоговое окно, вызываемое из контекстного меню, позволяет непосредственно изменять значения пары параметров большая-малая полуоси или эксцентриситет-фокальный параметр.

Сечение конуса плоскостью.

Модуль демонстрирует природу происхождения названия «конические сечения». На экране изображаются полупрозрачный круговой конус и квадратная площадка. Вершина конуса расположена в центре трехмерной системы координат, оси которой, для облегчения ориентации в пространстве, окрашены в соответствующие цвета: *Ox* в красный, *Oy* в зеленый, *Oz* в синий. Такая последовательность цветов будет применяться и в дальнейшем. Общие точки плоскости площадки и конуса образуют коническое сечение. Положение площадки может варьироваться за счет изменения значений параметров площадки – расстояния от плоскости площадки до центра и угла наклона к плоскости *Oxy*. Параметры задаются с помощью стандартных компонент Windows (и Delphi) – «бегунков».

Во время исполнения программы на экран выводятся текущие значения параметров сечения – фокальный параметр и эксцентриситет. Изменение точки зрения наблюдения позволяет пользователю взглянуть на объемную сцену из любой точки трехмерного пространства.

Коническое сечение строится на опорных точках, которые получаются в результате пересечения плоскости и прямых, образующих конус. При написании программы были разработаны функции, позволяющие находить пересечение прямой и плоскости, обрабатывать и выводить на экран массивы точек, непосредственно составляющих сечение.

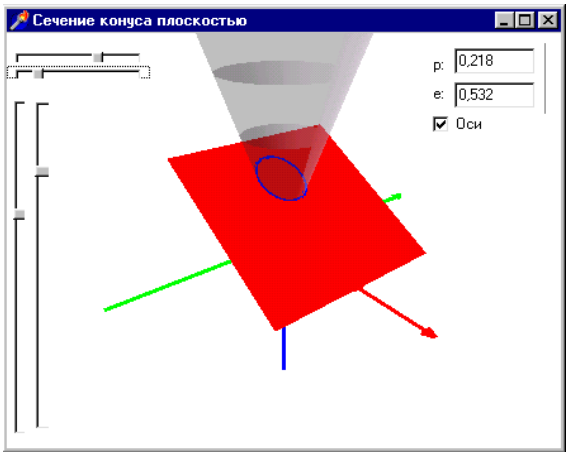


Рис.7. Приложение «Сечение конуса плоскостью».

3.2. Ограниченная задача трех тел.

Модули *Задача трех тел. Кривые Хилла. Области возможного движения* и *Поверхность нулевых скоростей* предназначены для построения линий уровня интеграла Якоби плоской ограниченной круговой задачи трех тел двумя способами: заданием константы интеграла и графическим построением сечений плоскостями соответствующей поверхности. Эти кривые, получающиеся в результате сечения плоскостей и поверхности, ограничивают области возможного движения при фиксированном значении константы интеграла Якоби, что даёт некоторое качественное представление о возможных траекториях.

Специальная подпрограмма модуля «*Задача трех тел. Кривые Хилла. Области возможного движения*» - *Libration(n,μ)* - позволяет рассчитать координаты точек относительного равновесия – точек либрации, где *n* – номер точки либрации. Решение возникающего в задаче уравнения пятой степени находится методом простых итераций. Положение точек либрации отмечается знаком ×.

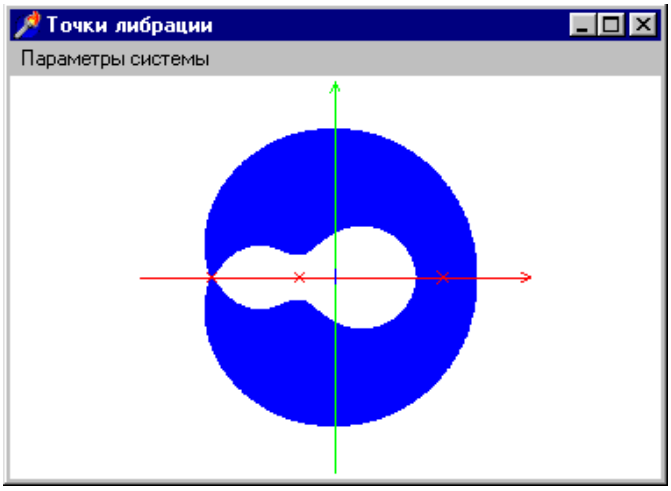


Рис.8. Приложение «Задача трех тел. Кривые Хилла. Области возможного движения».

Поверхность нулевых скоростей.

На основном экране строится исследуемая поверхность $z=\Omega(x,y)$ и секущая плоскость. Положение плоскости можно задать произвольным образом, что позволяет сформировать четкое представление о характерных особенностях поверхности – точках минимума и максимума, характере поведения функции в точках либрации. При формировании изображения предусмотрено отключение вывода на экран самой плоскости, а также координатных осей. Пользователь может изменять масштаб сцены, выбирать точку зрения и также задавать значения масс притягивающих тел. Установка параметров производится в отдельном окне.

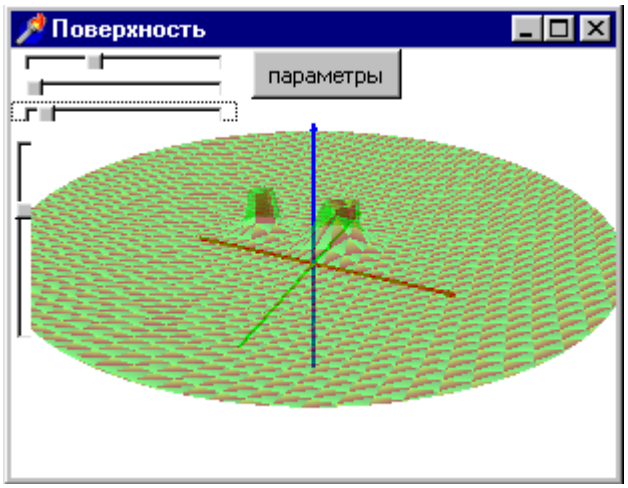


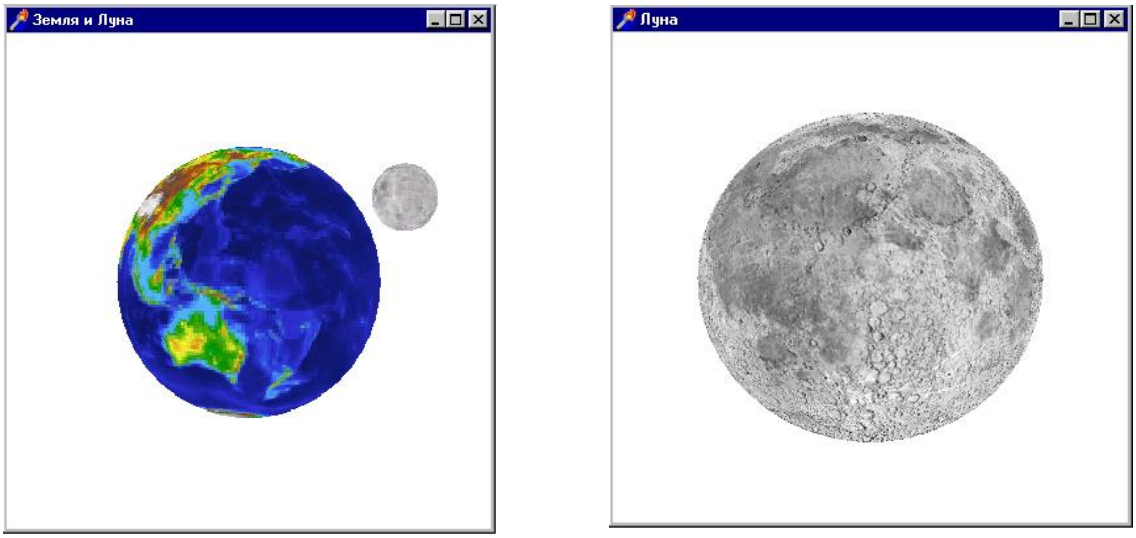
Рис.9. Приложение «Поверхность нулевых скоростей».

Вывод (визуализация) поверхности осуществляется средствами *OpenGL*. Графическая библиотека *OpenGL* не предоставляет явных инструментов для построения поверхностей, поэтому были разработаны функции, организующие вывод любой поверхности, заданной уравнением вида $z=f(x,y)$. Как и любой объект, поверхность аппроксимируется простейшими объектами графики - примитивами, в данном случае - треугольниками, вершины каждого из которых принадлежат поверхности. При инициализации приложения вершины треугольников заносятся в массив, который передается в качестве аргумента функции, осуществляющей вывод поверхности. Изменение величины треугольников позволяет строить поверхность максимально приближенную к действительной, однако следует учитывать, что слишком маленький шаг разбиения предъявляет высокие требования к машинным ресурсам.

3.3. Виртуальная Солнечная система.

Виртуальная Солнечная система – мультимедийное приложение, позволяющее провести «экскурсию» по всем планетам Солнечной системы и Луне. Установив начальные значения положения и скорости траектории облета, пользователю предоставляется возможность увидеть планету с борта

виртуального спутника, совершающего облет планеты по кеплеровской траектории задачи двух тел. Предусмотрено изменение масштаба, выбор положения наблюдателя и направления наблюдения.



а) Земля и Луна. ***б) Луна***

Рис.10. Приложение «Виртуальная Солнечная система».

Заметим, что данная программа по своим функциональным возможностям во многом схожа с приложением *«Музей планет»* интегрированной мультимедиа обучающей среды по небесной механике [1]. В отличие от *«Музея планет»*, где при создании анимационных трехмерных моделей использовались технологии машинной графики *3Dstudio* и *DirectX*, модели планет *Виртуальной солнечной системы* реализованы на основе технологии *OpenGL*.

3.4 Законы Кеплера.

Первый закон Кеплера.

Модуль предназначен для демонстрации первого закона Кеплера. В основном окне программы моделируется движение трех ближайших к Солнцу планет. На экране изображается текущее положение планет, траектория движения и постоянный вектор Лапласа, неизменно направленный на перигелий соответствующей траектории. В любой момент для выбранной траектории на экран можно вывести значения расстояний от планеты до двух фиксированных точек – фокусов, и тем самым удостовериться в том, что движение происходит согласно Первому закону Кеплера. В качестве задания также предлагается подобрать точку, для которой сумма расстояния от этой точки до планеты и расстояния от Солнца до планеты сохраняется во время движения, то есть найти второй фокус эллипса, по которому происходит движение.

Рис.11. Приложение «Первый закон Кеплера» и вспомогательное окно с формулировкой закона.

Каждая модель планеты представляет собой сферу с наложенным изображением поверхности – текстурой. Сфера является одним из так называемых квадрик-объектов, которые хранятся в библиотеке GLU.dll. Помимо самих квадрик-объектов библиотека предоставляет в распоряжение разработчика удобный инструментарий для работы с текстурами, позволяющий определять качество изображения в зависимости от ресурсов компьютера пользователя. Текстура накладывается с помощью специального механизма – списка изображений, благодаря которому достаточно спроецировать битовый образ на некоторую поверхность всего один раз при инициализации объекта. Применение списков избавляет от необходимости считывать образ при перерисовке каждого кадра и, тем самым, делает переход между кадрами более плавным. Данная технология применяется практически во всех задачах, в которых присутствуют объекты с текстурой.

Параметры орбиты каждой планеты хранятся в специально разработанной структуре типа *Tplanet*, содержащей полуоси, фокальный параметр, эксцентриситет и т.д. орбиты, а также текущее положение планеты. Воспользовавшись стандартным меню, пользователь может ознакомиться с формулировкой закона, обратиться к *Базе данных параметров Солнечной системы,* на основе которой рассчитывается движение планет, а также получить справку о текущем модуле.

Второй закон Кеплера.

Модуль предназначен для демонстрации второго закона Кеплера на примере движения Земли вокруг Солнца. На основном экране отображаются

планета, её траектория и радиус-вектор. Через равные промежутки времени площадь, заметенная радиус-вектором планеты за заданный интервал, окрашивается определенным цветом, контрастирующим с цветом предыдущего сектора. Временной интервал задается пользователем.

Рис.12. Приложение «Второй закон Кеплера» и вспомогательное окно.

Пользователь может проследить за зависимостью величины заметенной площади от времени на графике, доступном в отдельном окне. График строится с помощью стандартного компонента *Tchart*, обладающего встроенной возможностью изменения масштаба простым движением мыши. Заметим, что в данном случае график является просто прямой.

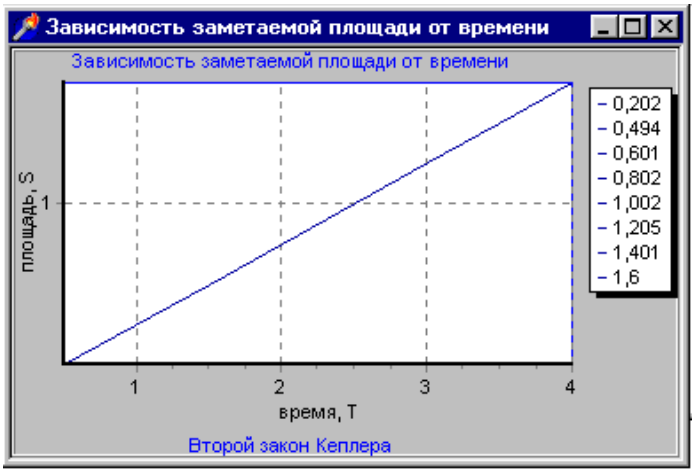


Рис.13. График зависимости заметаемой площади от времени.

3.5. Движение в центральном поле сил.

Движение по начальным условиям. Задача попадания в точку.

Программа моделирует движение под действием силы притяжения «Солнца», расположенного в центре системы координат. В начальный момент

движущийся объект находится на оси *Ox*. Значения проекций вектора скорости на оси координат вводятся пользователем с клавиатуры. Во время движения изображается траектория движения и текущее положение объекта. В любой момент на экран можно вывести основные параметры системы – компоненты вектора скорости, компоненты радиус - вектора, параметры конического сечения, константы площадей и энергии. Движение моделируется на основе объекта *Tplanet.*

Задача пользователя состоит в том, что ему необходимо подобрать начальную скорость таким образом, чтобы траектория прошла через конечную точку и при этом выполнялось какое-нибудь дополнительное условие. Концы векторов начальной скорости выбираются на гиперболе скоростей – однопараметрическом множестве скоростей, позволяющих осуществить перелет между двумя точками, благодаря чему решение задачи облегчается. Для однозначности решения можно потребовать, например, чтобы значение начальной скорости было минимальным.

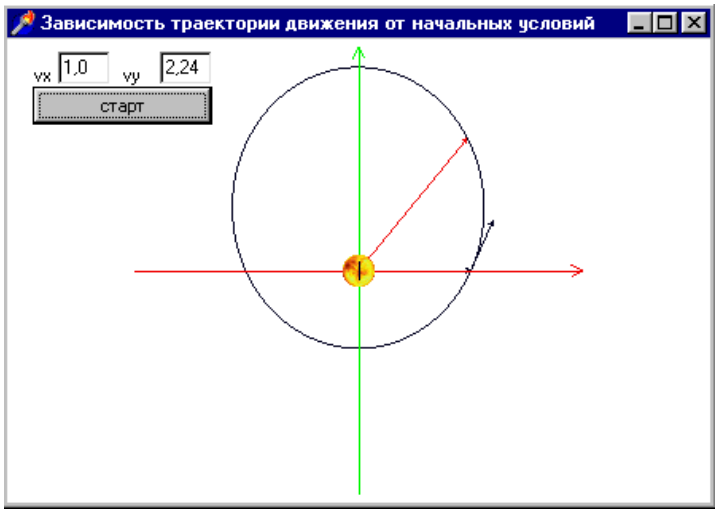


Рис.14. Приложение «Движение по начальным условиям».

Для задания точки старта достаточно просто «перетащить» основание вектора скорости в нужную точку. Потянув конечную точку вектора, можно изменить его модуль, сохранив направление, а также можно изменить и направление вектора, вращая его вокруг собственного начала. Такие возможности предоставляются благодаря использованию специально разработанного для этих целей компонента *Tarrow*. Форма курсора изменяется в зависимости от текущего режима, тем самым пользователь в любой момент может определить характер производимых изменений – растяжение/сжатие, поворот или перенос. Механизм переноса объектов мышью реализуется путем применения особого способа прорисовки объектов сцены в специальном буфере выбора с параллельным именованием самих объектов. К сожалению, на данный момент выбор объектов в среде *OpenGL* представляет значительные сложности при работе с двумя и более элементами сцены одновременно,

поэтому планируется разработка специальных пользовательских компонентов, ориентированных на применение механизма выбора объектов.

3.6. Сфера действия планеты. Маневры в Солнечной системе.

Программа предназначена для отыскания траекторий перелетов космических аппаратов в Солнечной системе с учетом влияния притяжения планет. Как известно, задача о движении материальной точки под действием притяжения нескольких подвижных тел приводит к неинтегрируемым дифференциальным уравнениям. В случае, когда массой точки можно пренебречь, для приближенной оценки истинной траектории пространство Солнечной системы можно разбить на несколько областей в каждой из которых учитывается влияние только одного тела притяжения. В данной работе используются так называемые сферы действия Лапласа [2]. Приближенная методика обычно дает достаточно хорошее представление о траектории и является эффективным средством при моделировании межпланетных траекторий на компьютере [2,3].

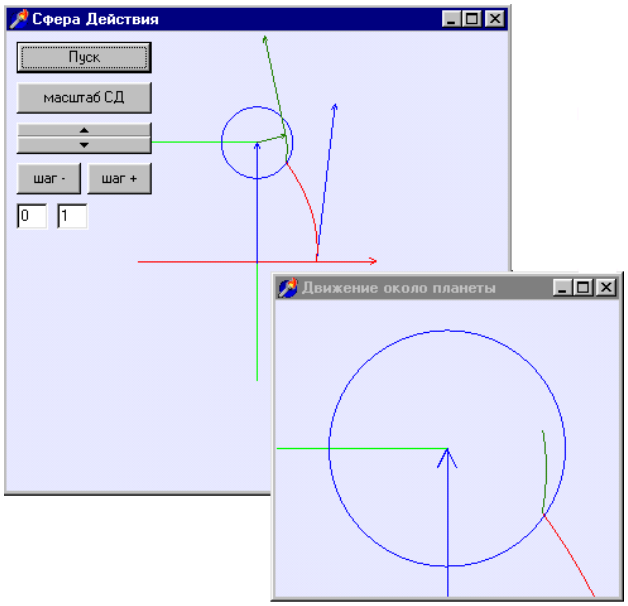


Рис.15. Приложение «Сфера действия планеты. Маневры в Солнечной системе».

В начале движения пользователь определяет исходное положение и скорость точки, положение тела, возмущающего траекторию, а также радиус его сферы действия. Данные, касающиеся параметров планет, в частности радиус сферы действия, содержатся в соответствующей *Базе Данных*. Установка векторных параметров, реализованных на основе компонента *Tarrow*, производится простым движением мыши. Положение возмущающего движение объекта можно задавать в отдельном окне в масштабе сферы действия. Данное окно позволяет исследовать движение около планеты.

В процессе моделирования на экране вычерчивается траектория точки, а также изображается вектор скорости относительно объекта, влияние которого учитывается в текущий момент. Помимо изменения масштаба изображения предусмотрен выбор шага по времени – интервала между двумя последовательными положениями движущейся точки. Наглядное представление о влиянии возмущающего тела на отклонение кусочно-конической траектории от траектории задачи двух тел можно получить, если воспользоваться возможностью одновременного моделирования вышеуказанных траекторий с одинаковыми начальными условиями.

С точки зрения визуализации особенность данной программы состоит в том, что результат моделирования представляется параллельно в двух отдельных окнах. Изменение изображения в одном окне приводит к синхронному изменению в соседнем окне. В процессе параллельного вывода изображения в нескольких окнах следует учитывать ряд особенностей, связанных со стратегией организации многонитевых приложений в *OpenGL*. Для корректной работы программы необходим постоянный контроль над тем, чтобы в каждый момент окно обладало собственной нитью, с которой связаны единственный контекст воспроизведения и контекст устройства. Создание многонитевых приложений облегчается благодаря использованию специально разработанных для такого рода приложений шаблонов модулей.

Заключение.

В работе представлены концепция и структура компьютерного практикума по небесной механике. Рассмотрены особенности программной реализации практикума. Приведены примеры подключаемых задач, разработанных с учетом предъявленных требований.

В ходе реализации задач был проведен ряд экспериментов по работе с задачами практикума. Эксперименты подтвердили эффективность принятой модели практикума.

Литература.

1. Ю.Ф.Голубев, В.Е.Павловский, Е.Ю.Голубева, А.Ю.Жаркова, Е.С.Нечаева, В.В.Павловский. *Интегрированная мультимедиа обучающая среда по небесной механике.* // Математическое моделирование, 2000, N 5, т.12, с. 74 - 80.
2. Справочное руководство по небесной механике и астродинамике. Под ред. Г.Н. Дубошина, «Наука», 1976.
3. Ф.И.Карманов. Компьютерное моделирование межпланетных перелетов в Солнечной системе. // Соросовский образовательный журнал, 2000, N 9, т.6, с.103-109.