Республиканская конференция учащихся
Малые чтения НОУ «Сигма»
«Первые шаги в науку»

Секция: Информатика

DarkBasic: моделирование солнечной системы

Работу выполнила: ученица 8 класса
МКОУ «СОШ №5 г.Баксана»
. Беканова Диана

Руководитель: Фотов Р.Б.

2015

Оглавление

Введение

В начале 8 класса я заинтересовалась программированием и попросила своего преподавателя информатики, Руслана Борисовича, научить меня составлять программы.

И после уроков я начала заниматься BASICом. Вначале все было интересно, увлекательно, но потом я поняла, что это не то чего я хотела. Я хотела красивых эффектов, трехмерной графики. Неужели этим могут заниматься только профессиональные программисты? Для того чтобы сделать что-то интересное и красочное необходимо закончить ВУЗ?

Поиск в интернете показал, что я ошибалась, и программ для работы с трехмерной графикой очень даже много. И в ходе поисков я поняла, что главной проблемой для меня будет не изучение языка программирования, а углубление своих знаний по алгебре и геометрии, изучение таких тем, которые не предусмотрены в школьном курсе.

Перед собой я поставила ***цель***: написать программу *моделирования солнечной системы*.

Определились с ***задачами***:

1. Краткий исторический обзор развития компьютерной графики;
2. Систематизировать существующие программы для работы с трехмерной графикой;
3. Определиться с языком программирования и написать программы моделирования солнечной системы.

История развития компьютерной графики

История развития компьютерной графики началось уже в 20 веке и продолжается сегодня. Не секрет то, что именно графика способствовала быстрому росту быстродействию компьютеров.

1940-1970гг. – время больших компьютеров (эра до персональных компьютеров). Графикой занимались только при выводе на принтер. В этот период заложены математические основы.

Особенности: пользователь не имел доступа к монитору, графика развивалась на математическом уровне и выводилась в виде текста, напоминающего на большом расстоянии изображение. Графопостроители появились в конце 60-х годов и практически были не известны.

1971-1985гг. – появились персональные компьютеры, т.е. появился доступ пользователя к дисплеям. Роль графики резко возросла, но наблюдалось очень низкое быстродействие компьютера. Программы писались на ассемблере.

Особенности: этот период характеризовался зарождением реальной графики.

1986-1990гг. – появление технологии Multimedia (Мультимедиа). К графике добавились обработка звука и видеоизображения, общение пользователя с компьютером расширилось.

Особенности: появление диалога пользователя с персональным компьютером; появление анимации и возможности выводить цветное изображение.

1991-2008гг. – появление графики нашего дня Virtual Reality. Появились датчики перемещения, благодаря которым компьютер меняет изображения при помощи сигналов посылаемых на него. Появление стереоочков (монитор на каждый глаз), благодаря высокому быстродействию которых, производится имитация реального мира.

Основные определения 3-мерной графики.

1) Простейший 3D-объект (Primitive) - треугольник, точка, линия, многоугольник. Все 3D-объекты состоят из простейших объектов.

2) Вершина (Vertex) - вершина 3-мерного объекта (вершина треугольника, например). Чтобы создать 3D-объект необходимо сконструировать его из простейших объектов. Например, любой объект можно сделать из треугольников. Треугольник определяется с помощью задания трех точек - его вершин, каждая из точек определяется тремя координатами (в трехмерном пространстве, конечно).

3) Рендеринг (Rendering) - процесс построения изображения 3-мерного объекта. Процесс рендеринга включает:

- Рендеринг (Rendering) - проектирование 3-мерного объекта из 3-мерной системы координат на экран;

- Фрагментирование;

- смешивание (Blending) и прочие операции над фрагментами (освещение, материал, текстурирование см. ниже);

4) Фрагмент (Fragment) - во время рендеринга многоугольник разбивается на фрагменты, каждый из которых соответствует одна точка экрана.

5) Цвет RGB - наш мозг воспринимает цвет как смесь красного, синего и зеленого. Но наши глаза воспринимают эти компоненты по отдельности. Например, яркий красный и яркий зеленый - это желтый, синий и красный - сиреневый. Поэтому, для задания цвета используют три числа RGB-триплет (R - красный, B - синий, G - зеленый).

6) Освещение (Lighting) - использование света для создания реалистичного и объемного изображения.

Свет имеет следующие характеристики:

- позиция - точка из которой исходит свет;

- угол освещения - свет может исходить во все стороны или в определенную область пространства (например, свет от прожектора);

- рассеяный свет (Diffuse) - цвет света, который отражается от объекта (рассеивается им)

- отраженный свет (Specular) - цвет света, который отразился объектом (как металлический блеск);

- разбросанный свет (Ambient) - цвет света, который падает на объект со всех сторон.

Сам свет невидим. Можно видеть только его действие на объект.

7) Материал (Material) - цветовые характеристики, которые определяют какой либо материал. Материал применяется к объекту. Материал включает следующие основные характеристики:

- рассеяный свет (Diffuse) - цвет света, который отражается от объекта (рассеивается им)

- отраженный свет (Specular) - цвет света, который отразился объектом (как металлический блеск);

- разбросанный свет (Ambient) - цвет света, который падает на объект со всех сторон;

- испущенный свет (Emission) - цвет свечения объекта.

Материал и свет определяют цвет обекта.

8) Координаты текстуры (Texture coords) - каждой вершине объекта назначается пара координат для "натягивания" текстуры на объект. Эти координаты отмеряются от какого-либо угла текстуры.

9) Текстура (Texture) - 2D картинка, которая натягивается на объект в соответствии с координатами текстуры. Подобно тому, как объект разбивается на фрагменты, текстура разделяется на части, называемые текселями (Texel - "texture element").

10) Нормаль (Normal) - вектор определающий как освещается ваш объект. Если Вы хотите, чтобы Ваш объект был правильно освещен светом, то Вы должны назначить каждой вершине вектор нормали. Если нормаль направлена точно против направления света, то освещение соответствующей

вершины максимально, если направление точно по направлению распространения света, то освещение мининмально.

OpenGL и DirectX

Для программирования графической составляющей компьютерных игр уже давно используются специальные интерфейсы, которые также находят применение не только в играх. Самые известные из них это — библиотеки DirectX и OpenGL.

OpenGL

Интерфейс OpenGL (Open Graphics Library) создан и запущен в 1992 году известными компаниями в сфере разработки программ. Он был представлен как кросс-платформенный интерфейс, не зависящий от аппаратного обеспечения. В базе лежала IRIS GL — библиотека, которую ранее разработали специалисты из Silicon Graphics Inc (SGI).

Развитие библиотеки OpenGL происходит за счет специализированной структуры — Architectural Review Board (ARB). Это «Комитет по пересмотру архитектуры». В него включены несколько компаний, которые имеют определенные интересы в процесс развития рассматриваемой библиотеки. Сюда входят такие гиганты как 3D Labs, NVIDIA, ATI, Intel, id Software, SGI, Apple, и Microsoft, который в своих реализациях Windows использует OpenGL. Версия же этого OpenGL сходна с более ранними разработками и не может работать с аппаратным ускорением, но эти возможности появляются благодаря драйверам используемых видеокарт.

OpenGL используется повсюду, начиная с 90-х голов 20 века. Это связано с тем, что данная библиотека была грамотно написана с удачной архитектурой. Эта библиотека ведет себя стабильно при использовании, а также поведение OpenGL легко предсказать. Одним из недостатков является медленное развитие. Это связано с медленной работой ARB, а именно в том, что при изменении определенных стандартов много времени уходит на согласование различных документов.

Особенно это заметно в последнее время, когда дешевые видеокарты стали достаточно профессиональными и их возможности меняются примерно раз в год. А развитие OpenGL отстает от развития технологий видеокарт,

поэтому существуют расширения, использующие отдельными разработчиками для возможности использования новых функций видеокарт.

DirectX

К выходу операционной системы Windows 95 все же большая часть видеоигр работала под MS-DOS. Ранние версии Windows не представляли возможностей для разработок игр.

Уровни абстрагирования, обуславливающие универсальность и совместимость, не позволяли быстрый доступ к видеокартам и такая модель не подходила для игр.

В связи с этим приняли решение о создании совершенно новой библиотеки, позволяющей напрямую работу с оборудованием. Это могло увеличить производительность игр, а также следовательно и продажи Windows 95.

Компания Microsoft решила не использовать разработки своего API и взяли за основу решения небольшой фирмы RenderMorphic. Известно, что первоначально данный API был студенческим заданием и не узнал успеха на экзамене. Но все же Microsoft встроили эту библиотеку в Game SDK, считая это отличной возможностью.

Позже эта разработка называлась DirectX 1.0. и не стала популярной. Из-за медленной работы, кучи ошибок и негибкой архитектуры, а также достаточно сложного кода.

Microsoft учитывали методы разработчиков игр и продолжали развивать DirectX. Версия 3.0 была более работоспособной. Потом были разработаны версии 5, 6 и 7. Седьмая версия тоже неплохо работала и ее использование было более удобным. Вскоре вышел DirectX 8, принеся с собой нововведения, а именно шейдеры — мини-программы, занимающиеся расчетом освещения и созданием специальных эффектов. Позже вышел 9-й DirectX со своими наработками. В данный момент существуют 10 и 11 версия. Хотя некоторые считают, что последние версии DirectX являются больше маркетинговым ходом, чем новыми разработками.

Сначала DirectX считался неудачной заменой OpenGL, но позже нововведения сделали DirectX мощной и работоспособной библиотекой. Сейчас большинство считает именно эту библиотеку образцом для программирования графики, также компания Microsoft тесно сотрудничает с разработчиками аппаратного обеспечения и не отстают от них.

Ранее говорилось о DirectX как о графической библиотеке, хотя также в ней есть интерфейсы работы со звуком, мультимедиа и прочими составляющими. Это также превосходит возможности OpenGL, потому что последний работает только с графикой.

Начиная с первых версий DirectX в своём составе ряд инструментов:

1. DirectDraw - инструмент для растровой графики,
2. Direct3D - инструмент для 3D графики,
3. DirectInput - инструмент для получения данных с мыши, клавиатуры, т.д.
4. DirectPlay - инструмент для обмена данными в сети,
5. DirectSound - инструмент для работы со звуком,
6. DirectMusic - инструмент воспроизведения музыки,
7. DirectShow - инструмент для ввода-вывода аудио и видео данных,
8. DirectSetup - инсталятор,
9. DirectX Media Objects - поддержка потоковых обработчиков (кодеры, декодеры).

Классификация программ для работы с 3-мерной графикой

Программы для работы с трехмерной графикой условно можно разделить на несколько групп:

- Системы автоматизированного проектирования (САПР) для 3D моделирования и построения чертежей по 3D моделям
- 3D-редакторы
- Игровые движки
- Языки программирования

Системы автоматизированного проектирования (САПР) для 3D моделирования и построения чертежей по 3D моделям

Система автоматизированного проектирования – автоматизированная система, реализующая информационную технологию выполнения функций проектирования, представляет собой организационно-техническую систему, предназначенную для автоматизации процесса проектирования

Основная цель создания САПР — повышение эффективности труда инженеров, включая:

- ✓ сокращения трудоёмкости проектирования и планирования;
- ✓ сокращения сроков проектирования;
- ✓ сокращения себестоимости проектирования и изготовления, уменьшение затрат на эксплуатацию;
- ✓ повышения качества и технико-экономического уровня результатов проектирования;
- ✓ сокращения затрат на натурное моделирование и испытания.

Естественно, увидев 3D модель двигателя вы поймете намного больше, чем по чертежу также как и то, что деталь выполненная станком с ЧПУ по 3D модели будет точнее, чем рабочим по 2D чертежу.

Программы:

- AutoCAD,
- Компас-График,
- Solid Works,

- Solid Edge,
- Компас-3D,
- CATIA,
- Pro/ENGINEER,
- NX

3D-редакторы

Для создания трехмерной модели требуются специальные программные и аппаратные средства. К программным принадлежат приложения 3D-визуализации, о которых пойдет речь ниже. К аппаратным относят то, с помощью чего создается и отображается модель (компьютер, 3D-мониторы, 3D-принтеры).

Процесс создания трехмерной модели включает три этапа:

1. Моделирование.
2. Визуализация.
3. Вывод модели (печать либо на монитор).

Моделирование - создание модели из ничего, проектирование с помощью программных средств, задание соответствующих размеров, текстур, освещения. Создается, так сказать, каркас объектов, описывается математическими формулами.

Следующим этапом является рендеринг (англ. render - визуализация) - преобразование сырого каркаса в приятную для глаза форму, закругление углов, отображение света, отображение текстур. Осуществляется с помощью программных средств.

Вывод на печать, либо на экран монитора полученной визуальной модели - последний этап.

Программные пакеты, позволяющие создавать трёхмерную графику очень разнообразны. Последние годы устойчивыми лидерами в этой области являются продукты, такие, как:

- ✓ Autodesk 3ds Max
- ✓ Autodesk Maya

- ✓ Autodesk Softimage
- ✓ Blender
- ✓ Cinema 4D
- ✓ Houdini
- ✓ Modo
- ✓ LightWave 3D
- ✓ Caligari Truespace

Игровые движки

Игровой движок – это центральный программный компонент компьютерных и видео игр или других интерактивных приложений с графикой, обрабатываемой в реальном времени. Он обеспечивает основные технологии, упрощает разработку и часто даёт игре возможность запускаться на нескольких платформах, таких как игровые консоли и настольные операционные системы, например, GNU/Linux, Mac OS X и Microsoft Windows.

Основную функциональность обычно обеспечивает игровой движок, включающий:

- ❖ движок рендеринга («визуализатор») для 2D или 3D графики,
- ❖ физический движок или обнаружение столкновений (и реакцию на столкновение),
- ❖ звук,
- ❖ искусственный интеллект,

Часто на процессе разработки можно сэкономить за счет повторного использования одного игрового движка для создания множества различных игр.

Термин «игровой движок» появился в середине 1990-х годов – в это время окончательно установилось доминирование IBM-совместимых компьютеров, а быстрые процессоры и «хитрое» программирование дали 30 и более кадров в секунду в трёхмерных играх. Игры Doom и Quake от id Software оказались настолько популярными, что другие разработчики вместо

того, чтобы работать с чистого листа, лицензировали основные части программного обеспечения и создавали свою собственную графику, персонажей, оружие и уровни – «игровой контент» или «игровые ресурсы». Движок Quake был использован в более чем десяти проектах.

Более поздние игры, такие как Unreal 1998 года (движок Unreal Engine) и Quake III Arena (на движке id Tech 3) 1999 года, были спроектированы с применением данного подхода, с отдельно разработанными движком и наполнением. Практика лицензирования такой технологии оказалась полезным вспомогательным доходом для некоторых разработчиков игр. Так, стоимость одной лицензии на коммерческий игровой движок класса high-end может варьироваться от 10 тыс. до 3,75 млн $ (в случае Warcraft III, а число лицензиатов может достигать несколько десятков компаний (как для Unreal Engine). По крайней мере, многократно используемые движки ускоряют и упрощают разработку игры, что является ценным преимуществом в конкурирующей индустрии компьютерных игр.

Дальнейшее усовершенствование игровых движков привело к сильному разделению между рендерингом, скриптингом, художественным дизайном и дизайном уровней. Сейчас для типичной команды разработчиков игр является вполне обычным иметь в составе столько же художников, сколько и программистов.

Самые популярные игровые движки:

- 3D Rad
- NeoAxis Game Engine SDK
- Unity 3D
- Unreal Development Kit (UDK)
- CryENGINE 3

Языки программирования

Язык программирования – это в первую очередь инструмент для решения определённого круга задач. Инструмент — это как топор или бензопила, молоток или кувалда, ножницы или кусачки. Круг задач, которые

необходимо решать, это как разработка операционной системы или написание антивируса, разработка драйвера или написание скрипта, разработка движка игры или разработка движка системы управления сайтом.

Топор или бензопила — что выбрать? Зависит от задачи. Если нам нужно срубить дерево и нарезать его равными частями, обрубив ветки с сучками, то для решения этой задачи нам подойдут оба инструмента. Разница в том, что топором будет медленнее, возможно качественнее, и потребуется много физических и временных затрат. Бензопилой получится быстрее (сокращение времени работы), но появится расход топлива. Есть такие типы задач, которые сложно решить одним инструментом или вообще невозможно. Например, если дерево очень толстое, то одной бензопилой дерево срезать не получится, нужен топор.

Если у нас задача наколоть дрова, то для данной задачи нелогично использовать бензопилу.

То же самое происходит и в программировании. Всё зависит от того, какую задачу вы хотите решить или какого типа задачи вы постоянно будете решать из одного направления.

C++ – компилируемый статически типизированный язык программирования общего назначения.

C++ широко используется для разработки программного обеспечения, являясь одним из самых популярных языков программирования. Область его применения включает создание операционных систем, разнообразных прикладных программ, драйверов устройств, приложений для встраиваемых систем, высокопроизводительных серверов, а также развлекательных приложений (игр).

Lua – интерпретируемый язык программирования, разработанный подразделением Tecgraf (Computer Graphics Technology Group) Католического университета Рио-де-Жанейро (Бразилия).

Язык широко используется для создания тиражируемого программного обеспечения (например, на нём написан графический интерфейс пакета Adobe Lightroom). Также получил известность как язык программирования уровней и расширений во многих играх (в том числе World of Warcraft) из-за удобства встраивания, скорости исполнения кода и лёгкости обучения

Python – высокоуровневый язык программирования общего назначения, ориентированный на повышение производительности разработчика и читаемости кода.

С Python поставляется библиотека tkinter для создания кроссплатформенных программ с графическим интерфейсом.

Существуют расширения, позволяющие использовать все основные библиотеки графических интерфейсов – wxPython. Некоторые из них также предоставляют широкие возможности по работе с базами данных, графикой и сетями, используя все возможности библиотеки, на которой основаны.

Для создания игр и приложений, требующих нестандартного интерфейса, можно использовать библиотеку Pygame. Она также предоставляет обширные средства работы с мультимедиа: с её помощью можно управлять звуком и изображениями, воспроизводить видео.

PureBasic – коммерческий язык программирования высокого уровня, основан на синтаксисе BASIC.

Предназначен для создания кроссплатформенных прикладных программ для AmigaOS, Linux, Microsoft Windows, Windows NT и Mac OS X.

В PureBasic встроены стандартные библиотеки для программирования консольного и графического интерфейса, библиотеки для создания 2D и 3D игр.

DarkBASIC – специализированный язык программирования, созданный компанией The Game Creators специально для создания трёхмерных и двумерных игр. Структура языка заимствована из BASIC.

Из BASIC в DarkBASIC перешли почти все операторы, и добавились специфичные команды, относящиеся к игровому движку, разработанному в The Game Creators для создания игр с использованием DirectX.

С помощью данного продукта практически любой пользователь, даже никогда ранее не знакомый с программированием, сумеет создать полноценное графическое приложение: игру с трехмерной графикой, видеозаставку, «хранитель экрана» и т. п. Причем ему не придется вникать в тонкости устройства ОС и способов отображения графики, не надо будет думать о механизмах работы графических ускорителей и учить мудреные слова наподобие «анизотропная фильтрация», – достаточно просто придумать алгоритм программы и реализовать его на довольно понятном и легком языке.

Простота создания подобных программ достигается тем, что этот язык практически полностью состоит из уже готовых команд для работы с графическими объектами. Значит, чтобы, например, отобразить на экране куб, достаточно одной строчки, в которой будут присутствовать название объекта и его параметры (длина ребра, поворот, координаты центра), а также команда отображения на экране. Чтобы проделать с предметом какие-либо действия вроде вращения, перемещения и т. д., достаточно двух-трех команд. Вместо куба можно брать и более трудные объекты (задав их в электронном виде) – команды сложнее не будут.

Практическая часть: Моделирование солнечной системы.

Выбор языка прогаммирования

Для создания модели солнечной системы решили воспользоваться языком программирования ***DarkBasic***. Из рассмотренных нами языков программирования он оказался самым простым в изучении начинающими программистами в области 3D.

DarkBasic позволяет загружать и работать с трехмерными объектами, созданными в различных редакторах. А также с изображениями в различных форматах: BMP, RLE, DIB, DDS, JPG, PCX, PNG, PSD, TGA, TIFF; видео в формате avi; звуки в формате mid, mp3, wav.

Создание модели солнечной системы

Так как я начала изучение языка DarkBasic поздно (конец января 2015), то создать серьезный проект мы не успели бы, то в этой работе решили познакомить с основами языка, а саму модель сделать упрощенным:

- ✓ Использовали только 8 планет.
- ✓ Не использовали карликовые планеты и астероиды.
- ✓ Колец у Сатурна нет.
- ✓ Планеты вращаются по круговым орбитам.
- ✓ Звезд на заднем фоне нет.

Из справочных данных использованы следующие данные

- ➢ Name- Название
- ➢ Diametr- Диаметр (км)
- ➢ Distans- Растояние от Солнца (млн. км)
- ➢ Day- Период вращения (часы)
- ➢ Year- Орбитальный период (дни)
- ➢ Speed- Орбитальная скорость (км/с)

Создаем пользовательский тип данных, состоящий из полей с этими параметрами и объявляем переменную

```
Type Planet
        Name As String
        Diametr As Integer
        Distans As integer
        Day As Integer
        Year As Integer
        Speed As Integer
        angle as float
endtype
Dim Planets(8) As Planet
```

И заполняем массив (**Приложение 1**)

Оператор *backdrop on* служит для включения заднего фона, а *color backdrop 0* устанавливает цвет заднего фона, в данном случае 0 – черный.

Создание Солнца и планет было очень простым. Создаем сферу нужного диаметра и «натянем» текстуру.

```
`Создаем солнце
make object sphere 100,DSol#/kDr#,64,64
load image "sun.jpg",100
texture object 100,100
```

Оператор *MAKE OBJECT SPHERE Номер_Объекта, Размер* создает сферу с нужным радиусом.

В DarkBasic’е все объекты и изображения имеют уникальные номера по которым идет обращение к ним. Причем нумерация объектов и изображений не связаны.

load image "sun.jpg",100 – загружает изображение а оперативную память, а *texture object 100,100* «натягивает» на объект 100 изображение под номером 100.

Текстуры солнца и планет были специально найдены сферические, чтобы не было швов при наложении.

Были использованы специальные переменные, которые задавали пропорции в размерах и расстояниях.

```
DSol#=140000.0          `Диаметр Солнца
kDr#=10000.0            `Коэффициент для диаметра планет
kDs#=1.0                `Коэффициент для расстояния от солнца
```

Планеты создаются точно также (**Приложение 2**)

```
`Создаем планеты
for i=1 to 8
    make object sphere i,Planets(i).Diametr/kDr#,64,64
    position object i,Planets(i).Distans*kDs#,0,0
    t$="m"+str$(i)+".jpg"
    load image t$,i
    texture object i,i
next i
```

Функция для вращения планет вокруг своей оси:

```
function PlanetRotation()
    for i=1 to 8
        angle#=1.0/Planets(i).Day
        yrotate object i, wrapvalue(object angle y(i)+angle#)
    next i
endfunction
```

В переменной *angle#* запоминается угол, на который должна повернуться планета в зависимости от периода вращения. Функция *object angle y(i)* возвращает угол поворота объекта i вокруг оси *Y* в градусах. Оператор *yrotate object i* поворачивает объект под номером i на *angle#* градусов больше чем было.

Функция вращения планет вокруг Солнца

```
function PlanetRun()
    for i=1 to 8
        k#=Planets(i).Angle+360.0/Planets(i).Year/kDr#*100.0
        Planets(i).Angle=k#
        x#=Planets(i).Distans*kDs#*cos(k#)
        y#=Planets(i).Distans*kDs#*sin(k#)
        position object i,x#,y#,0
    next i
endfunction
```

В переменной *k#* рассчитывается новый угол, на который должна повернуться планета вокруг Солнца с учетом орбитального периода. Вычисляются новые координаты *x* и *y* в пространстве и задается новая позиция: *position object i,x#,y#,0*. Так как вращение планет только в плоскости *x* и *y*, то координата *z*=0.

И, наконец, опрос клавиатуры на предмет нажатия цифровых кнопок от «1» до «8». При нажатии на кнопку крупным планом показывается выбранная планета и некоторая информация о планете.

```
a=asc(inkey$())-48
if a>0 and a<9 then PlanetSelect(a)
```

Функция просмотра планеты.

```
function PlanetSelect(a)
    position camera object position x(a),object position y(a),-Planets(a).Diametr/kDr#*2.0-0.3
    point camera object position x(a),object position y(a),0
    angle#=1.0/Planets(a).Day
    set cursor 100,700
    print Planets(a).Name
    set cursor 100,720
    print "Диаметр: ",Planets(a).Diametr/kDr#
    set cursor 100,740
```

*Print "Расстояние от Солнца: ",Planets(a).Distans*kDs#*

endfunction

При выборе планеты *point camera* устанавливает камеры по координатам *x* и *y* планеты но на некотором удалении по оси *z*, чтобы было видно планету.

Заключение

В последние годы сильно увеличился вычислительный потенциал компьютеров. Работа с трехмерной графикой перешло из разделов «для избранных» в раздел «доступно желающим». Создано очень много недорогих или бесплатных программ для обработки трехмерных объектов.

В руководстве пользователя продукта «DarkBasic. Программа для создания 3D-игр и не только» написано: «DarkBasic – уникальное средство, которое может дать вам возможность воплотить свои мечты в компьютерной программе».

С помощью данного продукта практически любой пользователь, даже никогда ранее не знакомый с программированием, сумеет создать полноценное графическое приложение: игру с трехмерной графикой, видеозаставку, «хранитель экрана» и т. п. Причем ему не придется вникать в тонкости устройства ОС и способов отображения графики, не надо будет думать о механизмах работы графических ускорителей и учить мудреные слова наподобие «анизотропная фильтрация», – достаточно просто придумать алгоритм программы и реализовать его на довольно понятном и легком языке.

Что касательно меня, то за неполных 2 месяца изучения этого языка, я смогла создать нечто красочное, реалистичное, что могу показать друзьям и родным со словами: «Это я сделала!». И надеюсь к следующей конференции уже будет готовая работа по данной теме, чтобы можно было бы представить сразу в двух направлениях: «Программирование» и «Астрономия».

Приложения

Приложение 1. Данные о планетах

Planets(1).Name="Меркурий"
Planets(1).Diametr=4879
Planets(1).Distans=58
Planets(1).Day=1408
Planets(1).Year=88
Planets(1).Speed=48
Planets(1).Angle=0
Planets(2).Name="Венера"
Planets(2).Diametr=12104
Planets(2).Distans=108
Planets(2).Day=-5833
Planets(2).Year=225
Planets(2).Speed=35
Planets(2).Angle=0
Planets(3).Name="Земля"
Planets(3).Diametr=12756
Planets(3).Distans=150
Planets(3).Day=24
Planets(3).Year=365
Planets(3).Speed=30
Planets(3).Angle=0
Planets(4).Name="Марс"
Planets(4).Diametr=6794
Planets(4).Distans=228
Planets(4).Day=25
Planets(4).Year=687
Planets(4).Speed=24
Planets(4).Angle=0
Planets(5).Name="Юпитер"
Planets(5).Diametr=142984
Planets(5).Distans=779
Planets(5).Day=10
Planets(5).Year=4331
Planets(5).Speed=13
Planets(5).Angle=0
Planets(6).Name="Сатурн"
Planets(6).Diametr=120536
Planets(6).Distans=1434
Planets(6).Day=11
Planets(6).Year=10747
Planets(6).Speed=10
Planets(6).Angle=0
Planets(7).Name="Уран"
Planets(7).Diametr=51118
Planets(7).Distans=2873
Planets(7).Day=-17
Planets(7).Year=30589
Planets(7).Speed=7
Planets(7).Angle=0
Planets(8).Name="Нептун"
Planets(8).Diametr=49528
Planets(8).Distans=4495
Planets(8).Day=16
Planets(8).Year=59800
Planets(8).Speed=5
Planets(8).Angle=0

Приложение 2. Планеты