4. Оцінка потенційного ступеня розрізнення імпульсних зондуючих радіосигналів з урахуванням кутової нестабільності генератора НВЧ / О.А. Мясіщев, І.І. Чесановський, Л.В. Карпова // Вісник Хмельницького національного університету. Технічні науки. – 2010. – № 2. – С. 72– 78.

5. Передающие устройства СВЧ : [учебное пособие для радиотехнических спец. вузов] / Вамберский М.В., Казанцев В.И., Шелухин С.А.; под общ. ред. М.В. Вамберского. – М. : Высш. шк., 1984. – 448 с.

6. Справочник по радиолокации : том 3. Радиолокационные устройства и системы / [под ред. А.С. Винницкого]. – М. : Сов. радио, 1978. – 528 с.

Надійшла 11.11.2012 р.
Рецензент: д.т.н. Шинкарук О.М.

УДК 681.327.12.001.33

Н.С. СВИРНЕВСКИЙ
Хмельницкий национальный университет

ИМИТАЦИЯ ПОЛЕТА КРЫЛАТОЙ РАКЕТЫ НА ОСНОВЕ ОБЪЕКТНО-ОРИЕНТИРОВАННОЙ МЕТОДОЛОГИИ ПРОЕКТИРОВАНИЯ ПРОГРАММНОЙ СИСТЕМЫ

Поставлена и реализована на основе объектно-ориентированной методологии проектирования программной системы задача имитации полета крылатой ракеты.

Ключевые слова: ракета, траектория, ландшафт, архитектура, программа, графика.

Formulated and implemented on the basis of object-oriented design methodology of a software system problem simulate flying cruise missile.

Keywords: missile trajectory, the landscape, the architecture, the program schedule.

Анализ исследований и публикаций

С развитием информационных технологий все более актуальной становится проблематика разработки программных систем с графическим отображением происходящих процессов [1]. Такие системы строятся на основе соответствующих математических методов [2], методологий программирования [3], с использованием графических библиотек [4].

Формулирование целей

Целью статьи является отображение процесса разработки программной системы, имитирующей полет крылатой ракеты, включая: анализ предметной области, алгоритмизацию задачи, выбор языка и среды программирования, создание архитектуры программы.

Изложение основного материала

Для управления полетом крылатой ракеты используются цифровые картины (карты) предварительно отснятых районов местности по маршруту полета. Комплект из лазерного, инфракрасного и спутникового оборудования определяют положение ракеты и цели на карте, а бортовая ЭВМ прокладывает текущий курс ракеты с учетом рельефа местности. Для решения задачи разработки программной системы, обеспечивающей имитацию полета крылатой ракеты над пересеченной местностью к двигающейся цели необходимо создать модели (рис. 1):

ландшафта земли;

траектории перемещения ракеты и цели над поверхностью ландшафта;

сцены, обеспечивающей взаимное положение объектов при их движении и изменении ракурса.

Рис. 1. Имитация полета крылатой ракеты

Моделирование ландшафта

Ландшафт моделируется на основе карты высот. Для хранения карт высот используется необработанный формат файлов RAW. Этот формат просто читать, поскольку он не содержит заголовков с какой-либо информацией об изображении, такой как размер или тип изображения. Размер карты может быть произвольный, но удобней использовать квадратную, с размером стороны кратным числу степени двойки: 128x128, 256x256... 1024x1024.

Файлы RAW являются двоичными файлами, содержащими данные о высотах ландшафта

(аналогично формату BMP-файла). В 8-разрядных картах высот каждый байт внутри файла RAW представляет высоту вершины. Значения высот варьируются от 0 до 255, где 0 (черный цвет) представляет самую низкую высоту вершины, а 255 (белый цвет) представляет максимально возможную высоту вершины. Можно расширить этот интервал, используя коэффициент масштабирования, который умножается на заданное значение высоты.

Чтобы построить ландшафт из карты высот сперва надо построить сетку вершин той же размерности, что и у карты высот, а затем использовать значение высоты точки (пикселя) из карты высот как высоту для вершины в сетке вершин (рис. 2). Точки выбираются через определенный шаг. Как известно, любая поверхность может быть представлена треугольниками. Поэтому любые четыре расположенные рядом точки карты высот задают прямоугольник, который мы разбиваем на два треугольника и рисуем. В OpenGL для этой цели можно использовать примитив типа **GL_TRIANGLE_STRIP.**

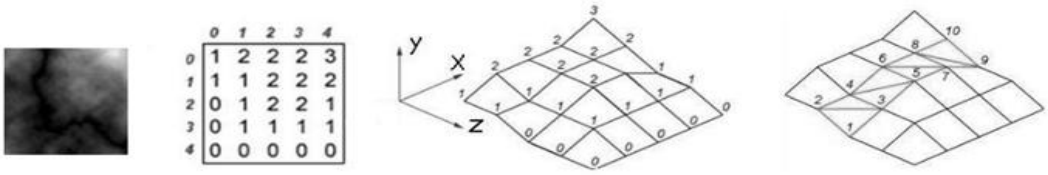

Рис. 2. Модель ландшафта

Для раскраски ландшафта можно применить интерполяцию цветов. Так, рисуя треугольник, необходимо каждой вершине задать разный цвет. При этом, OpenGL сам выполнит интерполяцию. Как известно в RGB режиме одна точка изображения описывается тремя байтами – значение Red, Green, Blue компоненты цвета. Если взять все три или одну из компонентов, как функцию высоты, то можно получить более качественное представление ландшафта – высокие точки будут более светлыми.

Моделирование траектории движения объектов

Над ландшафтом двигаются 2-а объекта – крылатая ракета и цель. Траектория движения каждого из них определяется 4-я точками: *pstart* – начальная; *ptarget* – конечная; *pcurrent* – текущая; *pnext* – следующая после текущей. Координата ***y*** точек определяется как функция высоты рельефа и координат *x, z* точки: *y= Height (x, z).* Начальное положение цели и точка, куда она перемещается, определяются при помощи генератора случайных чисел. Координаты *x* и *z* точек цели *pstart* и *ptarget* определяются при помощи генератора случайных чисел: *x= rand (); z= rand ().* Точка *pstart* ракеты находится в начале системы координат (при этом y= Height (0,0)), конечная точка ракеты совпадает с точкой *pnext* цели: *ракета.ptarget = цель.pnext.* Ракета и цель двигаются синхронно в пошаговом режиме – положение точек *pcurrent* и *pnext* определяется из параметрического уравнения отрезка прямой:

$$pcurrent.x = pstart.x + (ptarget.x - pstart.x) * i / Ik;$$
$$pcurrent.z = pstart.z + (ptarget.z - pstart.z) * i / Ik;$$
$$0 <= i <= Ik$$
$$pcurrent.y = Height\ (pcurrent.x, pcurrent.z).$$

Поражение цели происходит при *i=Ik.*

Моделирование сцены

Задача решается на основе математического аппарата аффинных преобразований [4]. Используются две системы координат (рис. 3). В глобальной *XYZ* моделируется ландшафт, определяются точки траектории объектов и положение локальной системы координат. В локальной *xyz* создается модель ракеты. Модель ракеты представляется двумя взаимно-перпендикулярными треугольниками, модель цели – точкой большого размера. Положение локальной системы координат относительно глобальной определяется текущей точкой траектории ракеты, а также углами *pci* **(**рыскания), *teta* (тангажа) и *gama* (крена).

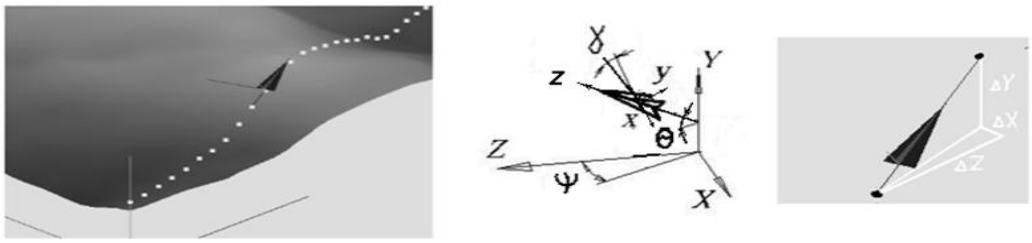

Рис. 3. Модель сцены

Текущие параметры определяются точками текущего положения ракеты (*pcurrent* и *pnext*):

dx= pnext.x – pcurrent.x;
dy= pnext.y – pcurrent.y;
dz= pnext.z – pcurrent.z;
dl =*sqrt* (pow (dx, 2) +pow (dy, 2)+pow (dz, 2))
pci = atan (dx/dz)*180/3.14159;

teta = asin (dy/dl)*180/3.14159;

gama=k* d_pci.

d_pci = pci (i) – pci (i-1), коеффициент k задается, чтобы gama < 45

Запуск ракеты инициализируется нажатием на клавишу P (пуск). Параметры начального положения ракеты определяются начальным положением цели. Геометрические преобразования реализуются *в OpenGL с помощью функций:*

glTranslate (x, y, z) – производит перенос, прибавляя к координатам значения своих параметров;

glRotate (angle, x, y, z) – производит поворот объекта на угол angle вокруг вектора (x, y, z);

gluLookAt (eyeX, eyeY, eyeZ, centerX, centerY, centerZ, upX, upY, upZ) – направление взгляда на сцену.

Взрыв, когда ракета достигает цели, имитируется сферой или точкой большого размера.

Проблематика создания архитектуры программы

При создании архитектуры программы наряду с обеспечением функциональности программы мы преследуем цель – преодоление сложности программных систем. Способ управления сложными системами был известен еще в древности – divide et impera (разделяй и властвуй). При проектировании сложной программной системы необходимо разделять ее на все меньшие и меньшие подсистемы, каждую из которых можно совершенствовать независимо. В этом случае мы не превысим пропускной способности человеческого мозга: для понимания любого уровня системы нам необходимо одновременно держать в уме информацию лишь о немногих ее частях (отнюдь не о всех). Сложные системы содержат два типа иерархий. В самолете, например, можно выделить несколько систем: питания, управления полетом и т.д. Такое разбиение дает структурную иерархию типа "быть частью". Эту же систему можно разложить совершенно другим способом. Например, турбореактивный двигатель – особый тип реактивного двигателя, а "Pratt and Whitney TF30" – особый тип турбореактивного двигателя. С другой стороны, понятие "реактивный двигатель" обобщает свойства, присущие всем реактивным двигателям; "турбореактивный двигатель" – это просто особый тип реактивного двигателя со свойствами, которые отличают его, например, от прямоточного. Эта вторая иерархия представляет собой иерархию типа "is-a". В объектно-ориентированном программировании обе иерархии находят свою реализацию (рис. 4) в структуре объектов и структуре классов. Наиболее успешны те программные системы, в которых заложены хорошо продуманные структуры классов и объектов [3].

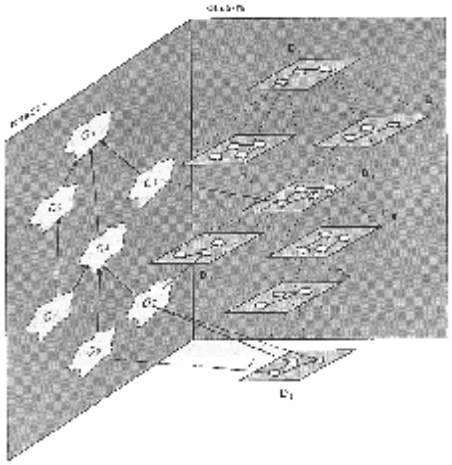

Рис. 4. Структура классов и объектов – архитектура системы

Создание архитектуры системы, иммитирующей полет крылатой ракеты.

Любое приложение с графическим интерфейсом должно иметь: окно для вывода изображения; возможности для создания в окне графических примитивов; возможности для взаимодействия с приложением.

Основой для создания такого приложения могут быть следующие языки программирования – C#.NET, C++ (WinApi или MFC) и другие. Остановим свой выбор на оконном WinAPI приложении на C++. Приложение, написанное на C++ без каких-либо библиотек классов, только с использованием WinAPI сделает приложение наиболее производительном при наименьшем размере исполняемого файла в Windows поскольку работа через Windows API – это наиболее близкий к системе способ взаимодействия с ней из прикладных программ.

Основными структурными компонентами оконного WinAPI приложении являются функция WinMain и оконная процедура. Первая отвечает за создание окна и организацию взаимодействия с окном (регистрация, бесконечный цикл), вторая – за обработку событий. События идентифицируется именем константы (`WM_PAINT, WM_DESTROY` и др.). Такая структуризация позволяет программисту легко вносить изменения в программу. Выполнив изначальные настройки приложения в функции WinMain в дальнейшем он работает только с функцией окна, добавляя те или иные события и прописывая реакцию на них. Для решения задач рисования в окне не будем портить файл main.cpp, а создадим новые блоки.

В данной программе реализованы следующие блоки:

- блок, в котором происходит создание окна и обрабатываются сообщения операционной системы, реализован в файле main.cpp;
- графический движок, выполняющий отрисовку изображения, класс Engine.

Движок – специализированный модуль для решения определенного класса задач в различных средах. В данном случае – это класс, который объединяет функцию рисования (Draw) и объекты, обеспечивающие вспомогательные функции. Графические возможности оконного WinAPI приложения ограничены элементарными действиями по созданию примитивов на плоскости типа отрезка, окружности и т.п. Для их расширения необходимо создать аппарат афинных преобразований. Не будем создавать

велосипед и воспользуемся библиотекой OpenGL, в функциях которой реализован этот аппарат. В обычном WinAPI приложении (без использования OpenGL) рисование осуществляется через объявление объектов дескриптора контекста устройства (экрана) и структуры paintstruct для информации в рабочей области окна. При использовании для рисования в Windows окне возможностей библиотеки OpenGL необходимо обеспечить взаимосвязь контекста устройства Windows с контекстом воспроизведения OpenGL. Команды, вырабатываемые конвейером OpenGL, поступают в контекст воспроизведения OpenGL, который в понятном (для контекста устройства Windows) виде, предоставляет фрагменты, которые надо нарисовать на поверхности окна или устройства. Дальше Windows сама распоряжается, что и как делать.

Один из вариантов архитектуры системы представлен на рис. 5. Здесь в классе Engine, инкапсулированы данные и функции, устанавливающие взаимосвязь контекста устройства Windows с контекстом воспроизведения OpenGL, а также функции, обеспечивающие задание модели объекта ее преобразование и отрисовку. В классе **Track** инкапсулированы текущие координаты и параметры объектов (ракеты и цели). Методы класса **Land** генерирует 3D-рельеф по точкам из карты высот.

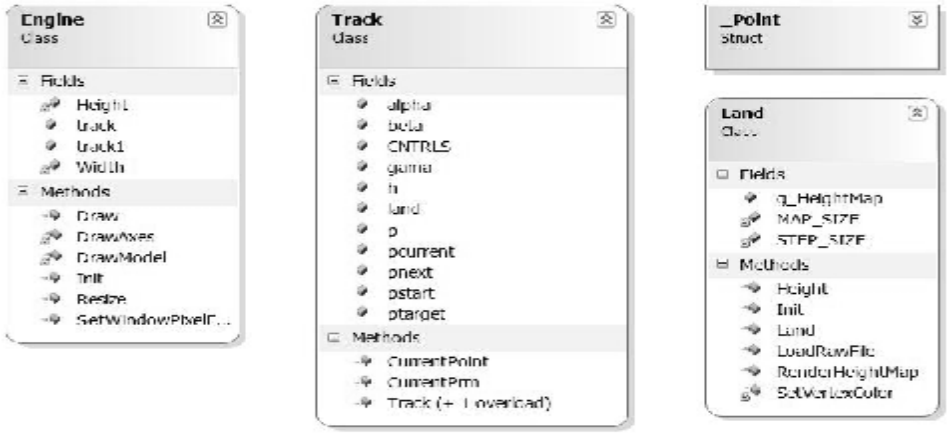

Рис. 5. Первый вариант архитектуры системы

Недостатки такого разбиения:

- объект цели не использует большое количество параметров и методов класса Track.
- каждый из объектов класса Track имеет свой объект класса Land

Первый из недостатков можно ликвидировать благодаря схеме (рис. 6):

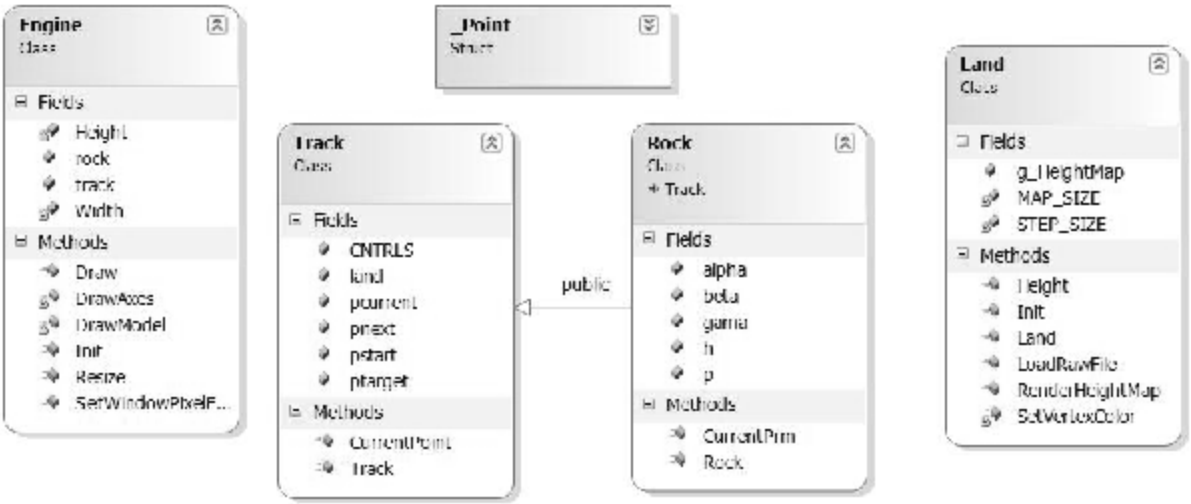

Рис. 6. Второй вариант архитектуры системы

Оба недостатка можно ликвидировать благодаря схеме:

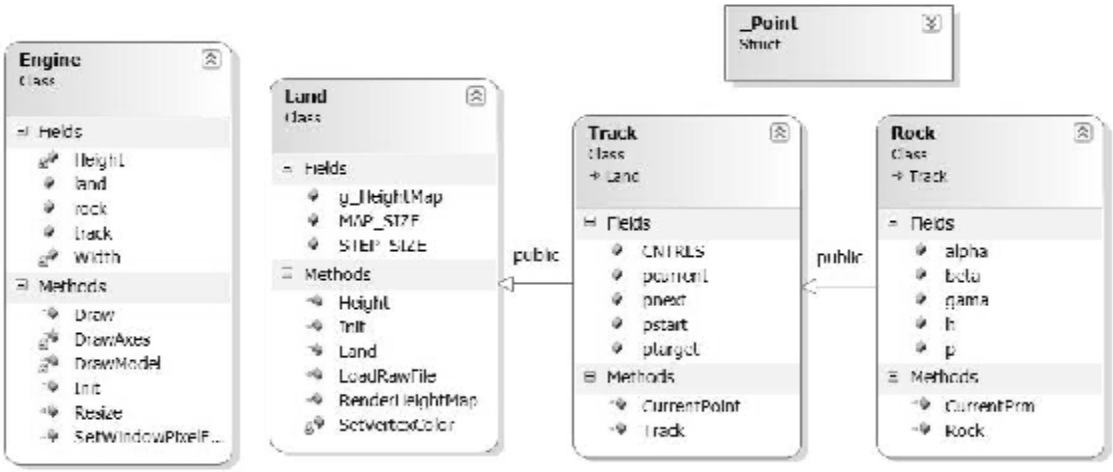

Рис. 7. Третий вариант архитектуры системы

Выводы

В данной статье описан процесс разработки программной системы, имитирующий полет крылатой ракеты. Проведены анализ предметной области, алгоритмизация задачи, отражен процесс создания архитектуры программы. Изложенный материал может иметь практическое применение при создании систем управления движением объектов по пересеченной местности, а также при создании игровых приложений.

Литература

1. Свирневский Н.С. Моделирование задачи автоматизированного проектирования изделия / Н.С. Свирневский // Вісник Хмельницького національного університету. Технічні науки. – 2006. – № 5. – С. 50– 54.

2. Михайленко В.Е., Анпилогова В.А. Справочник по машинной графике в проектировании / Михайленко В.Е., Анпилогова В.А. – К. : Будівельник, 1984. – 183 с.

3. Буч Г. Объектно-ориентированный анализ и проектирование / Буч Г. – М. : Бином, 1998. – 560 с.

4. OpenGL. Руководство по программированию / Шрайнер Д.и др. – Питер, 2006. – 624 с.

Надійшла 8.11.2012 р.
Рецензент: д.т.н. Сорокатий Р.В.

УДК 621.321

О.С. ПИВОВАР, С.Р. ПАВЛІНСЬКИЙ
Хмельницький національний університет

МОДЕЛЮВАННЯ ГЕНЕРАТОРІВ ХАОСУ ДЛЯ ВИКОРИСТАННЯ В UWB СИСТЕМАХ ОБРОБКИ ІНФОРМАЦІЇ

У статті розглядається ітераційна модель генератора хаосу із метою визначення особливостей спектрів хаотичних сигналів під час зміни керуючого параметру. Показано можливість використання спектральних особливостей для введення інформаційного сигналу у хаотичний носійний процес.

Ключові слова: генератор хаосу, хаотичний сигнал, спектр, біфуркація, UWB.

In the article the iteration model of generator of chaos is examined with the aim of determination of features of spectrums of chaotic signals during a change managing to the parameter. Possibility of the use of spectral features is shown for introduction of informative signal to the chaotic process.

Keywords: generator of chaos, chaotic signal spectrum, bifurcation, UWB.

Вступ

Зростаючі обсяги інформації мобільних систем телекомунікацій постійно потребують збільшення пропускних спроможностей каналів зв'язку. Особливо гостро стоїть проблема розробки радіосистем, що могли б функціонувати із вже існуючими системами мобільних комунікацій в одному і тому самому частотному діапазоні. Можливість спільного існування різних систем в межах одного частотного ресурсу потребує суттєвої відмінності застосованих сигналів у різних системах. Якщо традиційні системи використовують модульовані гармонічні коливання, то має сенс у нових системах використовувати хаотичні сигнали.

Основна частина

Хаотичні коливання утворюються в хаотичних нелінійних нестійких динамічних системах, тобто