Monte Carlo Modeling for Electron Microscopy and Microanalysis

David C. Joy

Monte Carlo Modeling for Electron Microscopy and Microanalysis

OXFORD SERIES IN OPTICAL AND IMAGING SCIENCES

1. D. M. Lubman (ed.). *Lasers and Mass Spectrometry*
2. D. Sarid. *Scanning Force Microscopy With Applications to Electric, Magnetic, and Atomic Forces*
3. A. B. Schvartsburg. *Non-linear Pulses in Integrated and Waveguide Optics*
4. C. J. Chen. *Introduction to Scanning Tunneling Microscopy*
5. D. Sarid. *Scanning Force Microscopy With Applications to Electric, Magnetic, and Atomic Forces, revised edition*
6. S. Mukamel. *Principles of Nonlinear Optical Spectroscopy*
7. A. Hasegawa and Y. Kodama. *Solitons in Optical Communications*
8. T. Tani. *Photographic Sensitivity: Theory and Mechanisms*
9. D. Joy. *Monte Carlo Modeling for Electron Microscopy and Microanalysis*

Monte Carlo Modeling for Electron Microscopy and Microanalysis

DAVID C. JOY

New York *Oxford*
OXFORD UNIVERSITY PRESS
1995

Oxford University Press

Oxford New York
Athens Auckland Bangkok Bombay
Calcutta Cape Town Dar es Salaam Delhi
Florence Hong Kong Istanbul Karachi
Kuala Lumpur Madras Madrid Melbourne
Mexico City Nairobi Paris Singapore
Taipei Tokyo Toronto

and associated companies in
Berlin Ibadan

Published by Oxford University Press, Inc.,
200 Madison Avenue, New York, New York 10016

Oxford is a registered trademark of Oxford University Press

Library of Congress Cataloging-in-Publication Data
Joy, David C., 1943—
Monte Carlo modeling for electron microscopy and microanalysis / David C. Joy.
p. cm. — (Oxford series in optical and imaging sciences : 9)
Includes bibliographical references.
ISBN 0-19-508874-3
1. Electron microscopy—Computer simulation. 2. Electron probe microanalysis—Computer simulation. 3. Monte Carlo method.
I. Title. II. Series.
QH212.E4J67 1995
502′.8′25—dc20 94-35642

A disk, 3½-inch MS-DOS format, containing all the source code in this book is available by mail from David C. Joy, P.O. Box 23616, Knoxville, TN 37933-1616, U.S.A. The cost, including postage and handling, is $10.00 (U.S. and Canada) or $15.00 (elsewhere). Payments accepted in the form of a check or a money order.

1 3 5 7 9 8 6 4 2

Printed in the United States of America
on acid-free paper

PREFACE

Electron microscopy and electron beam microanalysis are techniques that are now in daily use in many scientific disciplines and technologies. Their importance derives from the fact that the information they generate comes from highly localized regions of the specimen, the data produced are unique in scope, and the images and spectra produced can be quantified to give detailed numerical data about the sample. For quantification to be possible, however, it is necessary to be able to describe how a beam of electrons interacts with a solid specimen—and such a description is difficult to provide because of the very varied and complex nature of the interactions between energetic electrons and solids.

The purpose of this book is to demonstrate how this interaction can be accurately simulated and studied on a personal computer, by applying simple physical principles and the mathematical technique of Monte Carlo (or random-number) sampling. The aim is to provide a practical rather than a theoretical guide to this technique, and the emphasis is therefore on how to program and subsequently use a Monte Carlo model. The bibliography lists other books that cover the mathematics of Monte Carlo sampling and the physical theory of electron scattering in detail. To make the programs developed here as accessible as possible, a disk—for use with MS-DOS–compatible computers—has been made available; it contains all of the source code described in this book together with executable (i.e., runnable) versions. To order see facing copyright page.

This book would not have been possible without the generous cooperation of many other people. I am especially grateful to Dr. Dale Newbury of N.I.S.T., for first introducing me to Monte Carlo models, and to him and Dr. Robert Myklebust, also of N.I.S.T., for sharing their code with me. Dr. Hugh Bishop of A.E.R.E. Harwell, whose Ph.D. work produced the first electron beam Monte Carlo programs, kindly lent me a copy of his thesis and provided some invaluable background information. Dr. Peter Duncumb, of the University of Cambridge and Tube Investments Ltd., whose pioneering work on Monte Carlo modeling using minicomputers ultimately made this book possible, lent me copies of his early reports and papers and has been unfailingly helpful in answering many questions about the development of the technique. The programs given in this volume have been refined and improved through the efforts of many colleagues who have used them over the past few years. Vital improvements in science, substance, and style have been made by

Drs. John Armstrong (California Institute of Technology), Ed Cole (Sandia), Zibigniew Czyzewski (University of Tennessee), Raynald Gauvin (Université de Sherbrooke), David Howitt (University of California), David Holt (Imperial College, London), Peggy Mochel (University of Illinois), John Russ (North Carolina State University), and Oliver Wells (IBM). To them, to my students Suichu Luo, Xinlei Wang, and Xiao Zhang, and to many others who have given of their time and expertise, I am deeply grateful. Any errors and problems that remain are strictly my own responsibility.

Finally, I dedicate this book to my wife Carolyn, without whose love and encouragement this manuscript would have remained just another pile of floppy discs.

Knoxville D. J.
August 1994

CONTENTS

Monte Carlo Modeling for Electron Microscopy and Microanalysis

This page intentionally left blank

1

AN INTRODUCTION TO MONTE CARLO METHODS

1.1 Electron beam interaction—the problem

The interaction of an electron beam with a solid is complex. Within a distance of a few tens to a few hundreds of angstroms of entering the target, the electron will interact with the sample in some way. The interaction could be the result of the attraction between the negatively charged electron and the positively charged atomic nucleus (and equally the repulsion between the negatively charged atomic electrons and negative charge on the incident electron), in which case the electron will be deflected through some angle relative to its previous direction of travel, but its energy will remain essentially unaltered. This is called an *elastic* scattering event. Alternatively, the incident electron could cause the ionization of the atom by removing an inner-shell electron from its orbit, so producing a characteristic x-ray or an ejected Auger electron; it could have a collision with a valence electron to produce a secondary electron; it could interact with the crystal lattice of the solid to generate phonons; or, in one of several other possible ways, it could give up some of its energy to the solid. These types of interactions in which the electron changes both its direction of travel and its energy are examples of *inelastic* scattering events. After traveling a further distance, the electron will then again be scattered, either elastically or inelastically as before, and this process will continue until either the electron gives up all of its energy to the solid and comes to thermal equilibrium with it or until it manages to escape from the solid in some way.

While at a sufficiently atomistic level this train of scattering events is presumably quite deterministic—given sufficient information about the electron and the parameters describing it—to an observer able to watch the electron as it travels through the solid, the sequence of events making up the trajectory for any given electron would appear to be entirely random. Every electron would experience a different set of scattering events and every trajectory would be unique. Since, in a typical electron microscope, there are actually about 10^{10} electrons impinging on the sample each second, it is clear that there is not likely to be any simple or compact way to describe the spatial distribution of the innumerable interactions that can occur or the various radiations resulting from these events. At best it will be possible to assign probabilities to specific events, such as the chance of an electron being

backscattered (i.e., being scattered through more than 90°) or of being transmitted through the target; but any more detailed analysis of the interaction will be impossible. The Monte Carlo method described here uses such probabilities, together with the idea of sampling by using random numbers, to compute one possible set of scattering events for an electron as it travels into the solid. By repeating this process many times, a statistically valid and detailed picture of the interaction process can be constructed.

1.2 The Monte Carlo method

One of the very earliest published papers on "Monte Carlo" methods (Kahn, 1950) provides an excellent statement of the basis of the method—"By applying random sampling techniques to the problem [of interest] deductions about the behavior of a large number of [electrons] are made from the study of comparatively few. The technique is quite analogous to public opinion polling of a small sample to obtain information concerning the population of the entire country." The use of random sampling to solve a mathematical problem can be characterized as follows. A game of chance is played in which the probability of success P is a number whose value is desired. If the game is played N times with r wins then r/N is an estimate of P. The "game of chance" will be a direct analogy, or a simplified version, of the physical problem to be solved. To play the game of chance on a computer, the roulette wheel or dice are replaced by random numbers. The implication of a "random" number is that any number within a specified range (usually 0 to 1) has an equal probability of being selected, and all the digits that make up the number have an equal probability (i.e., 1 in 10) of occurring. Thus to take a simple and relevant example, consider an electron that can be scattered elastically or inelastically, the probability of either occurrence being determined by its total cross section. If the probability that a given scattering event is elastic is p_e and p_i is the probability of an inelastic scattering event (and the sum of p_e and p_i is unity), then a choice could be made between the two alternatives by picking a random number RND ($0 < \text{RND} < 1$) and specifying that if $\text{RND} \leq p_e$, then an elastic event occurs, otherwise an inelastic event is assumed to have occurred. If this selection procedure is applied a large number of times, then the predicted ratio of elastic to inelastic events will match the expected probability derived from the given probabilities p_e and p_i, since a fraction p_e of the random numbers will be $\leq p_e$. Random numbers can also be used to make other decisions. For example, if the probability $p(\theta)$ of the electron being scattered through some angle θ is known, either experimentally or from some theoretical model, then a specific scattering angle α can be obtained or picking another random number and solving for α the equation:

$$\text{RND} = \frac{\int_0^{\alpha} p(\theta)\, d\theta}{\int_0^{\pi} p(\theta)\, d\theta} \tag{1.1}$$

which equates RND to the probability of reaching the angle α given the known distribution of $p(\theta)$. By repeating these random-number sampling processes each time a decision must be made, a Monte Carlo simulation of one particular electron trajectory through the solid can be produced. The result of such a procedure is not necessarily or even probably a trajectory representing one that could be observed experimentally under equivalent conditions. However, by simulating a sufficiently large number of such trajectories, a statistically significant mixture of all possible scattering events will have been sampled and the *composite result* will be a sensible approximation to experimental reality.

1.3 A brief history of Monte Carlo modeling

The first published example of the use of random numbers to solve a problem is probably that of Buffon, who—in his 1777 volume *Essai d'Arithmetique Morale*—described an experiment in which needles of equal length were thrown at random over a sheet marked with parallel lines. By counting the number of intersections between lines and needles, Buffon was able to derive a value for π. Subsequently, other mathematicians and statisticians followed Buffon's lead and made use of random numbers as a way of testing theories and results. Because many of the phenomena of interest to physicists in the early twentieth century, such as radioactive decay or the transmission of cosmic rays through barriers, displayed an apparently random behavior, it was also an obvious step to try to use random numbers to investigate such problems. The procedure was to model, for example, a cosmic ray interaction by permitting the "particle" to play a game of chance, the rules of the game being such that the actual deterministic and random features of the physical process were exactly imitated, step by step, by the game and in which random numbers determined the "moves."

During the Manhattan Project, which led to the development of the first atomic bomb, John von Neumann, Stanislav Ulam, and others made innovative use of both random-number sampling and game-playing situations involving random numbers as a way of studying physical processes as diverse as particle diffusion and the probability of a missile striking a flying aircraft. It was during this period that these techniques were first dubbed "Monte Carlo methods" (Metropolis and Ulam, 1949; McCracken, 1955). Because the Monte Carlo method needs a large supply of random numbers as well as much repetitious mathematical computation, the later development of the technique was closely geared to the development of "automatic computing machines." As first mechanical and then electronic machines became available during the 1950s, the technique found increasing application to problems ranging in scope from diffusion studies in nuclear physics to the modeling of population growth by the Bureau of the Census. A valuable bibliography of these early papers and techniques can be found in Meyer (1956).

Although Monte Carlo methods had been applied to many other phenomena, it was not until the work of Hebbard and Wilson (1955) that the method was suc-

cessfully employed for charged particles. Later work by Sidei et al. (1957) and Leiss et al. (1957) led to a major paper by Berger (1963) that laid the groundwork for future developments. Simultaneously, in England, M. Green, a physics graduate student in Cambridge working for V. E. Cosslett, was persuaded to investigate the application of von Neumann's Monte Carlo method, and the university's EDSAC II computer, to the scattering of electron beams. Taking experimental data on the scattering of electrons in a 1000-Å film as a starting point, Green (1963) and later Bishop (1965) were able to derive the electron backscattering coefficient, and the depth dependence of characteristic x-ray production, from a bulk sample and to demonstrate good agreement with measured values. However, while this approach showed the validity of the technique, it was limited in its application to those situations where the suitably detailed initial experimental data were available. At the 4th International Conference on X-ray Optics and Microanalysis in Paris in 1965, however, two independent papers (Bishop, 1966; Shimizu et al., 1966) demonstrated how theoretically based electron scattering distributions could replace and so generalize experimental distributions; within a short time, groups in Europe, Japan, and the United States had produced working programs based on this concept. One of the most important of these was the one produced by Curgenven and Duncumb (1971) working at the Tube Investments Laboratory in England. This program introduced several new concepts, including the so-called multiple scattering approximation discussed in Chapter 4 of this book, and was optimized to run on a relatively small scientific computer. Copies of the FORTRAN code were generously made available to interested laboratories throughout the world for their own use; as a result, this program came into widespread use and made a significant contribution to popularizing the idea that electron-solid interactions could be modeled conveniently and accurately by computer.

By 1976, the use of this technique was sufficiently common for a conference entitled "Use of Monte Carlo Calculations in Electron Probe Microanalysis and Scanning Electron Microscopy" to be held at the National Bureau of Standards (NBS) in Washington, D.C. The proceedings of that meeting (Heinrich et al., 1976) still form one of the basic resources for information in this field, and programs, algorithms, and procedures developed by the NBS group have formed the starting point for many of the programs in current use, including those described in this volume.

Apart from the very earliest examples, which were run by hand, using random numbers generated by spinning a "wheel of chance" (Wilson, 1952) or on mechanical desk calculators (e.g., Hayward and Hubbell, 1954), Monte Carlo programs were designed to be run on the main-frame computers then becoming available in most government and industrial laboratories and universities. Although such machines were both large and expensive, their capabilities were very limited, and considerable ingenuity was necessary to produce workable programs within the limits set by the available memory (often as little as 2000 words) and operating time between crashes

of the system. Nevertheless Monte Carlo programs were often cited as a prime example of the new analytical power made available through electronic computing. With the advent of personal computers (PCs), this power is now available to anyone who needs it. Monte Carlo programs are, in most cases, relatively short in length and can readily be run on any modern PC without encountering any problems with the lack of memory. The programs are also computationally intensive, in the sense that once the program has obtained all the necessary data, it performs calculations continuously until its task is finished. This is not the ideal situation for a program that is run on a time-shared main-frame computer because it means that the actual computing time will depend directly on the number of users working on the machine at any given time (unlike programs such as word processing, where the majority of the computer's time is spent in waiting for the operator to enter the next character and multiple users produce little apparent drop in response speed). Consequently, even on relatively powerful time-share systems, the computational speed experienced by a user when running a Monte Carlo program can seem very slow; thus to calculate a sufficient number of trajectories to produce an accurate result can cost a lot of both time and money. While PCs do not, in general, perform the individual computations as rapidly as the main frame, they *are* dedicated to one task; as a result, their effective throughput can easily rival that of much larger machines. Also, since access to PCs is often free or at least very cheap, over-lunch or even overnight runs are no financial burden to the user. Finally, the interactive nature of PCs and the ready access to graphical presentation that they provide offers the chance to make programs that are both more accessible and more immediately useful.

1.4 About this book

This volume is not intended to replace standard textbooks on the general theory of Monte Carlo sampling (such as those by Hammersley, 1964, and Schreider, 1966), nor is it a substitute for a comprehensive guide to electron beam microscopy and microanalysis (such as Goldstein et al., 1992). Rather, it is intended to provide electron microscopists, microanalysts, and anyone concerned with the behavior of electrons in solids with ready access to the power of the Monte Carlo method. It therefore provides working Monte Carlo simulation routines for the modeling of electron trajectories in a solid and discusses procedures to deal with associated phenomena such as secondary electron and x-ray production. These procedures can then be added to the basic simulation as required to produce a program customized to tackle particular problems in image interpretation or microanalysis. The goal is to make available simulations whose accuracy is at least as good as that likely to be achieved in a comparable measurement or experiment on an electron microscope. The programs have been developed and have been designed to be run on personal computers rather than on scientific minicomputers or full size main-frame machines. Even given the advanced PC designs now available, this has occasionally made it

necessary to compromise between the completeness of the model and the speed of execution. In most cases the choice has been for the version that is fast in operation, since a good approximation available rapidly is much more useful than an exact result that takes a day to compute. No claim is made that these programs represent the best or even the only way to do the job. Indeed, a large number of other approaches are cited in the text. This book will have achieved its purpose if you—the user—feel ready, willing, and able to use the printed programs given here, or those available on the accompanying disk, as the basis for your own experimentation and development.

2
CONSTRUCTING A SIMULATION

2.1 Introduction

In this chapter, we will develop a Monte Carlo simulation of a random walk (sometimes called a "drunken walk," after its most popular mode of experimental investigation). Although this particular problem is only loosely related to the studies of electron beam interactions that follow, the model that we will develop provides a convenient way of establishing a framework for those subsequent simulations. It also illustrates the general principles of programming to be followed in this book and introduces some of the important practical details associated with constructing and running such models on a personal computer.

2.2 Describing the problem

The random walk problem can be stated as follows: "How far from the starting point would a walker be after taking N steps of equal length but in randomly chosen directions?" In order to simulate this problem, we must break it down to a sequence of instructions, or algorithms, that allow us to describe it mathematically. Figure 2.1 shows the situation for one of the steps making up the walk. It commences from the coordinate (x, y) reached at the conclusion of the previous step and is made at some random angle A with the X-axis, so that:

$$A = 2\pi*\text{RND} \tag{2.1}$$

where RND is a random number between 0 and 1. The coordinates xn, yn of the end of the step are then:

$$xn = x + \text{step}*\cos(A) \tag{2.2}$$

$$yn = y + \text{step}*\sin(A) \tag{2.3}$$

Equations (2.2) and (2.3) can be cast in a more symmetrical form by writing

$$B = (\pi/2) - A \tag{2.4}$$

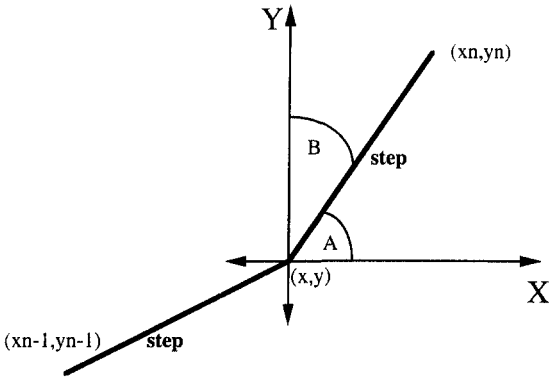

Figure 2.1. Coordinate system for the random walk simulation.

so that

$$xn = x + \text{step}*\cos(A) \tag{2.5}$$

$$yn = y + \text{step}*\cos(B) \tag{2.6}$$

cos (*A*) and cos (*B*), the cosines of the angles between the vector representing the direction of the step and the axes, are called the direction cosines and will in future be abbreviated to CA and CB.

Although Eqs. (2.1), (2.5), and (2.6) give an accurate mathematical description of one step of the random walk, the axes of the coordinate system are constantly changing as the walker moves from one step to the next. It would intuitively be more satisfactory to describe the progress of the walker with respect to a fixed reference frame of axes (such as the walls of the room), because this makes it possible to predict when, for example, a collision might occur. With this description then, as shown in Fig. 2.2:

$$\theta = 2\pi*\text{RND} \tag{2.7}$$

$$A = X + \theta \tag{2.8}$$

$$B = (\pi/2) - A = Y - \theta \tag{2.9}$$

where *X*, *Y* are the angles described by the direction cosines *CX*, *CY* for the previous step, and *A*, *B* are the angles for the new direction cosines *CA*, *CB*. As before

$$xn = x + \text{step}*CA \tag{2.10}$$

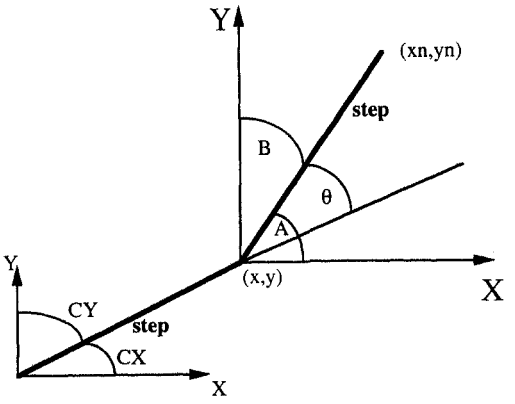

Figure 2.2. Modified coordinate description using fixed axes for the random walk.

$$yn = y + \text{step}*CB \tag{2.11}$$

and from the usual trigonometric expansions we get

$$CA = \cos(X + \theta) = CX*\cos(\theta) - CY*\sin(\theta) \tag{2.12}$$

$$CB = \cos(Y - \theta) = CY*\cos(\theta) + CY*\sin(\theta) \tag{2.13}$$

using the result that $\sin(X) = \cos(Y)$ and $\sin(Y) = \cos(X)$. Equations (2.7), (2.10), (2.11), (2.12), and (2.13) now describe how to calculate the end point of a step, given its starting point. The next step can similarly be computed using the identical equations but resetting the coordinates so that the old end point becomes the new starting point and the exit direction cosines become the entry direction cosines:

$$x = xn,\ y = yn,\ CX = CA,\ CY = CB \tag{2.14}$$

The recipe for simulating the random walk is therefore as follows:

Given a starting point (x_0, y_0) and a starting direction (CX, CY), then
Repeat the sequence
Find the deviation angle θ [Eq. (2.7)]
Calculate the new direction cosines CA,CB [Eqs. (2.12) and (2.13)]
Calculate the new coordinates xn,yn [Eqs. (2.10) and (2.11)]
Then reset the coordinates for the next step
$x = xn,\ y = yn,\ CX = CA,\ CY = CB$
until the required number of steps has been taken
Then distance s from starting point is $s = \sqrt{(x - x_0)^2 + (y - y_0)^2}$

2.3 Programming the simulation

To carry out the simulation on a computer, the recipe given above must be expressed in a form that the computer can understand. This requires that we choose one computer language, from among the many now available, in which to code our program. Unfortunately, any discussion of programming languages is liable to lead to acrimony, because anyone who regularly uses a particular language and has become used to its syntax and particular strengths and weaknesses can always find sufficient reasons to prove that any other language is deficient in power or convenience. However, stripped of the theological overtones so often accompanying this sort of debate, the truth is that any of the languages now in common use on personal computers could be used to code this and the following, Monte Carlo simulations without a noticeable effect on the quality of the final product. But since it is not practical to provide equivalent code for all possible languages, it is necessary to choose just one arbitrarily, for whatever reasons seem appropriate, and work with that.

Even though it might not have been your first or even your second choice, the decision here has been to use PASCAL. The reasons for this decision were principally as follows:

1. PASCAL is a good example of a modern language. It allows for structured and modular programming; it has a powerful yet simple syntax; and—since it is a compiled language—it is fast in execution.

2. The style of a PASCAL program, in particular the use of indenting and the availability of long descriptive variable names, leads to code that is easy to read and understand.

3. Variants of PASCAL are commercially available for all computers likely to be encountered in current use. Although there are slight differences between them, the original definition of the language was sufficiently precise that these variations rarely pose a problem if a program must be moved from one version of PASCAL to another.

4. Finally, if you cannot take PASCAL at any price, then—since other modern languages such as QUICKBASIC, ADA, MODULA-2, or C now share so many of the features of PASCAL—conversion from PASCAL to any other language of your choice is straightforward. In fact, software is available that can effect such transformations automatically in many cases.

The programs in this book are written in TURBO PASCAL™ (version 5.0), which is perhaps the most widely used form of PASCAL for MS-DOS computers. These programs will also compile and run, without any changes being necessary, in

Microsoft QUICK PASCAL™ version 1.0 and higher. Other variants of PASCAL and other types of computers may require some modifications to the code, especially for the graphics commands. A disk (IBM/MS-DOS format) containing all of the code discussed in this text, as both source code and as compiled and executable code, is available from the author. Ordering details are given on the copyright page of this book.

2.4 Reading a PASCAL program

The PASCAL program that implements the mathematical description derived above for the random walk simulation is as follows:

```
Program Random_walk;
   {this stimulates a simple random walk with equal-length steps}

uses CRT,DOS,GRAPH; {resources required}

var
   CA,CB,CX,CY:extended;                    {direction cosines}
   x,xn,y,yn,theta,distance:extended;       {step variables}
   step:extended;                           {display variables}
   hstart,vstart:integer;                   {screen center}
   i,tries:integer;                         {counter variables}
   GraphDriver:Integer;                     {which graphics card?}
   GRAPHMODE:Integer;                       {which display mode?}
   ErrorCode:Integer;                       {is there a problem?}
   Xasp,Yasp:word;                          {aspect ratio of screen}

cons
      twopi=6.28318;                        {2π constant}

Procedure set_up_screen;
   {gets the required input data to run the simulation}

  begin                                     {ensures screen is clean}
     ClrScr;
      GoToXY(10,5);
      writeln('Random Walk Simulation');

      GoToXY(33,5);                         {get number of steps}
         write('. . . . . . . . . how many steps?');
           readln(tries);
  end;

Procedure initialize;
   {identify which graphics card is in use and initialize it}
```

```
var
   InGraphicsMode:Boolean;
   PathToDriver:String;

  begin
   DirectVideo:=False;
    PathToDriver:='';
     GraphDriver:=detect;
    InitGraph(GraphDriver,GRAPHMODE,PathToDriver);
   SetViewPort(0,0,GetMaxX,GetMaxY,True);    {clip view port}

   hstart:=trunc(GetMaxX/2);     {horizontal midpoint of screen}
   vstart:=trunc(GetMaxY/2);     {vertical midpoint of screen}
   step:=(GetMaxX/50);           {a suitable increment}

   GetAspectRatio(Xasp,Yasp);    {find aspect ratio of this display}

  end;

Procedure initialize_coordinates;
   {set up the starting values of all the parameters}

  begin
               x:=0;
               y:=0;
              CX:=1.0;
              CY:=0.0;

              randomize; {and reset the random-number generator}
  end;

Procedure new_coord;
   {computes the new coordinates xn,yn given x,y, CX,CY and theta}

var
   V1,V2:extended;

  begin

   V1:=cos(theta);
   V2:=sin(theta);

     CA:=CX*V1 - CY*V2;    {new direction cosines}
     CB:=CY*V1 + CX*V2;

   xn:= x + step*CA;
   yn:= y + step*CB;       {new coordinates}

  end;
```