CREATING
POLITICAL & MILITARY SIMULATION GAMES
ON YOUR MICRO

Mike Rose

Now you can take control! Full instructions on how to develop your own simulation games. Complete with listings for ten ready-to-run games. Suitable for AMSTRAD, SPECTRUM+ and SPECTRUM, BBC MICRO, APPLE, all MSX machines, plus any machine furnished with BASIC.

Interface Publications
LONDON · MELBOURNE

CREATING POLITICAL AND MILITARY SIMULATION GAMES ON YOUR MICRO

INCLUDES TEN READY-TO-RUN MAJOR GAMES

Mike Rose

First published in the UK by:
Interface Publications,
9–11 Kensington High Street,
London W8 5NP, UK

First printing August 1985

ISBN 0 947695 20 6

Books published by Interface Publications are distributed in the UK by WHS Distributors, St. John's House East Street, Leicester LE1 6NE (0533 551196) and in Australia and New Zealand by PITMAN PUBLISHING. Any queries regarding the contents of this volume should be directed by mail to Interface Publications, 9–11 Kensington High Street, London W8 5NP.

Cover design – Richard Kelly
Printed and Bound by Short Run Press Ltd, Exeter

CONTENTS

NOTE: The listings in the book are written in a common subset of BASIC, and will run with minimal modification on the following computers: AMSTRAD, SPECTRUM + and SPECTRUM, COMMODORE 64, BBC MICRO, APPLE and all MSX machines. The listings were created on a BBC Micro and some changes may be needed to run the programs on your computer. If the program includes a line like A = RND (5) it means the variable A is to be set equal to a randomly-chosen integer between 1 and 5. You may need to change the line to A = INT(RND(1)* 5)+ 1. If your computer does not have GET$, replace it with INKEY$. A REPEAT/UNTIL construction simply makes the computer cycle over and over again until a particular condition has been met. If your computer does not have REPEAT/UNTIL, just check the input, or condition to be met, and GOTO the start of that section over and over again until the right input or condition is met in order to 'leap out of' this loop.

FOREWORD – TIM HARTNELL

A whole new world awaits you. A world where **you** make the decisions, where **you** give the orders, where **you** snap your fingers and people run to do your bidding.

This is the world revealed for you by Mike Rose in this major book which will teach you how to create and play strategy games on your computer – games which recreate political and military situations which you can control.

As well as the extremely detailed instructions which explain how you can write such games, Mike has included ten complete program listings which you can get up and running on your computer immediately.

This book is your key to this new world. It is time to turn it.

Tim Hartnell,
London, 1984.

Tim Hartnell is managing director of *Interface Publications.* He is the author of more than 50 books on various aspects of computer operation. His latest published works include *Tim Hartnell's QL Handbook* and *Exploring Artificial Intelligence on your Microcomputer* (both Interface Publications, 1984); *The Giant Book of Computer Games* (Fontana, UK, 1983; Ballantine Books, USA, 1984) and *Executive Games for your IMB PC and XT* (Ballantine Books, USA, 1984).

INTRODUCTION

Strategy games – or what is meant here by strategy games – are to "space invader" games what chess is to – well, arm wrestling. Arcade games may improve your reaction time, but where strategy games have the edge is in increasing your concentration, and powers of clear thinking. It is possible to envisage a future society where the qualification to enter, say, politics and the army at high rank is to attain a certain grade in a particular strategy game. I would say they definitely have a bright future.

From the reader's point of view, this book requires only a relatively elementary knowledge of computing – only enough to put logical flowcharts into BASIC.

The essence of good programming is to be able to define exactly what is required of a program, and to be able to translate the requirement into a flowchart form without losing the object. The final translation into "computer language" is the relatively easy part. This book helps by helping you decide what is required in a program, by giving instructions for building up flowcharts and translating them.

As you become skilled at the art – or science? – of writing strategy game programs, many of the intermediate steps can be carried out automatically, without thought. The exhaustiveness of this text is partly insurance against failing imagination at any stage, partly to familiarise the reader with the distinct stages of, and the reasoning behind, the writing process, and partly to ensure the book works with any computer language with its own structures and so on.

The logical progression of the book is the creation of a "real-life" game, constructing an objective flowchart from this and eventually the translating of it into computer language. The chapters fit together best in their numerical order, but may be useful in other orders for some tasks.

Note that the way to write strategy games presented here is not the only way. It does, undoubtably work, and I personally find it very effective. However, you may well find other variations on the same theme better for them.

Although the actual structure of the book is intended for strategy games, many of the component techniques – such as display management – can readily be applied to all types of program, from puzzles, "Adventure" games and Arcade games through to educational programs to data processing and business software.

Mike Rose,
Dunblane,
Perthshire,
1984.

CHAPTER 1
INVENTING A GAME

Most computer enthusiasts have played strategy games. By "strategy game", I mean one where you find out quite quickly that planning is required to get anywhere. Experience usually helps as well. In fact, there is a vast number of strategy games possible. "Adventure" games, board games and card games are all games of strategy, and all can be programmed into a computer. However, the ones we are concerned with here are games simulating political and military situations.

There are two parts involved in writing a strategy game program. One is the composing of the program. The other, which concerns us now, is the actual invention of a viable game. This is the single greatest area of possible innovation in the writing of a game program.

How the game works

The game is basically a network of factors which have to be manipulated correctly to win. The factors may represent "pieces", such as tanks, squadrons of aircraft or catapults, which are moved around during the game. On the other hand, they may represent numerical variables such as money, friendliness and inflation. The object of the game must be related to the factors involved, as must failure situations.

In strategy games, several of the most important factors must make conflicting demands on the player. The strategy – the essence of the game – is introduced in making compromises between conflicting demands. Strictly speaking, any game representing conflicting demands is a strategy game. However, the more simple games can

be won in much the same way each time they are played. For this reason, good strategy games are more complex and flexible. One way of achieving this is to have a larger number of conflicting factors. Another way is to introduce **random factors.** These may be in the form of small, random changes in variables such as popularity, or in the form of fairly randomly occurring "happenings" like the "Chance Pile" in "Monopoly". Here, comparatively large changes in some factors may be caused from time to time.

Yet another way of expanding strategy games is to introduce one or more **subgames,** outside the main game. A subgame is really an expanded "happening"-type random factor, with a greater proportion of interaction with the player. A crude example would be in a space-war, where most time is spent directing ship movements to counter enemy attacks and press home advances. Occasionally, your home planet itself comes under attack and you rush your command ship home to engage in a wild graphics fight, shooting enemy ships yourself.

As I said, this is a crude and rather unrealistic example, but it illustrates the point, that subgames are a change – often of pace and setting – from the main game. They are linked to the main game by common factors, but are games in their own right to some extent. In fact, from the game writing point of view, it is probably easiest to deal with them in the same way as an entire game.

TYPE OF GAME	RANDOM FACTORS	SUBGAMES
Military	Sudden suicide attack Equipment failure New equipment Unexpected resistance Deserters	Escalation Retreat New front opens
Political	Strikes Scandal Disasters	Negociations Outbreak of war Elections

TABLE 1 POSSIBLE RANDOM FACTORS AND SUBGAMES

So you see that a strategy game should be fairly complex to be satisfying and different each time it is played. The only physical limit to the complexity – and hence the program length – is the computer memory available. The games in this book are not particularly long, so that they fit into all but the smallest computer memories. If they are not complex enough for your taste, it is easy enough to expand them using the techniques in the book.

Types of game

As you know, the games in this book come under the general heading "political and military". To be more exact, they fit the types "military, political, diplomatic and economic". Political and diplomatic are almost identical – diplomacy can be regarded as international politics – so these will be handled together under "political". Economic games deal with the actual mechanism of the country and its control, as opposed to persuading the populace that you can run the country and dealing with other countries, which is politics. Therefore, games will be divided into the following categories: military, political and economic.

There are basically two types of military game. "True military" covers games which accurately trace the course of a confrontation using the precise co-ordinate positions and movement. These games can be extremely long and complex, but, more importantly as far as this book is concerned, sophisticated graphics are necessary. Since most sophisticated graphics are highly machine-specific, the only "true military" games considered in this book use a simple board-game type display. The other type of military game is like "The Siege of Doune Castle" at the end of this book. You do not move men and equipment to exact locations of the "2 miles W, 3 miles N" ilk; instead, you direct them to specific destinations. This method is much easier to write to and understand and is used several times here.

Setting and variables

Now we come to the start of composing the actual game. Presumably, you have some idea for a game, some basic concept. Different

people's first concepts vary considerably in details. For the same game, one person might think "a battle in a Tolkein setting" while someone else might think "a confrontation between an army of elves and men and an army of goblins and dwarves, each moves differently and has different weapons". This difference does not really matter at this stage – the more basic idea fills out as you go along.

In making up a game, you have to know what the **variables** are – the characters and equipment and amounts of substances of any degree of abstraction. You must also know the **laws** of the game – the "natural" reactions of the game to any changes. The detailed laws are considered in the next chapter. Another thing you need to know is the **rules** – what the player is allowed to do in terms of the format accepted by the computer. This comes under Input and is dealt with in detail in Chapter 5.

But the first thing to determine is the **setting,** i.e. when and where the game takes place. The number of detailed settings for each type of game is probably infinite, but they can be generalised, as the following table shows.

GAME TYPE	GENERALISED SETTING	EXAMPLES
Military	1. Evenly matched battle situation 2. Unevenly matched situation (besieging/defending)	"Battle of Britain" game Defending an oilrig
Political	1. Government of country/ colony 2. Diplomacy between countries 3. "Getting to the top"	19th Century Britain simulation Modern Lebanon East-West Relations Grecian city-state
Economic	Any government	Galactic Empire

TABLE 2 SETTINGS AND EXAMPLES

Note that a game may equally be a hybrid of more than one type, say, military and political.

Of course, **all** the material from the setting can never be used – that would be far too complicated. Instead, you have to identify the subset of the setting you will use. This process produces the characters and material for the game. If the necessary information does not spring to mind, there are two ways to set about it:

1. Take the complete setting – the entirety of life at the time and place of the game – and cut out all details which have no bearing on the game;
2. Start at the "top" – determine the most important and prominent features of the setting. If these are not sufficient to build your game on, work "down" to less prominent features until you have enough.

No matter how you go about it, you must end up knowing:

(a) The player's role;
(b) All the major and minor characters and objects involved; and the various other quantities such as "trust", in political games.

The list of variables you end up with is not necessarily complete. Particularly in political and economic games, where variables are more abstract, you may find you need to add further variables as references to the past, and coefficients in order to relate the main characters in the desired ways.

GAME TYPE	COMMON VARIABLES
Military	Men, Ships, Food, Fuel, Bases, Territory held
Political	Popularity, Allies, Enemies, Trustworthiness, Promises
Economic	Popularity, Money, Tax, Army size, Wages

TABLE 3 COMMON VARIABLES

The opponent

Having established the setting you will use, the next step is to decide on the form the opponent takes. In true military games and political games which are a straight fight between two sides, the opponent can be considered as another player for the time being. The workings of the opponent's "mind" are the subject of Chapter 4, on the Reply Move. In true military games, the opponent has to be very sophisticated, since it meets the player on equal terms. In military games with an unevenly matched situation – as in "The Siege of Doune Castle" – an opponent is not needed to such an extent, and may be much simpler. In fact, it may be practically dispensed with and replaced by more random functions to decide on the distribution of forces or whatever. And, of course, the main opponent in economic games is a hard-to-manage set of variables rather than a real enemy.

GAME TYPE	WIN SITUATION	LOSS SITUATION	RELEVANT VARIABLES
Military	Destroy enemy forces Invade far enough	Lose your forces You yourself killed	Your men Enemies Existence of a "piece"
Political	Some achievement Win election Triumph against threatening situation (war etc.)	Fail to achieve object Lose election or kicked out Situation occurs (war breaks out)	Rank, Honour Relationship Popularity Provocation of opponent
Economic	Succeed in running country	Country bankrupt Lose election	Balance of payments Popularity Standard of living

TABLE 4 WIN AND LOSS

Win and loss

Now we have the raw materials for the game itself. The object of the game and the loss situations must be defined, based on the characters and variables already found. It is normal to base these win and loss situations on major variables like "Men left" and "Popularity". However, the choice is not restricted as you will see from the game "Road to Valhalla", where the variable "Honour" is used to decide your fate in the afterlife. The value of this variable is never indicated directly during the game – you just have to guess. Table 4 gives some examples of win and loss situations together with the variables they are based on.

In economic games, it is not usually possible to lose until a general election or similar, unless the player does spectacularly badly. So in this sort of case, there are two separate loss situations – total economic failure and low popularity in an election. Needless to say, when you choose the win and loss situations for any game, you take the particular setting into account – the table is only for examples.

At this point, a distinction should be made between **direct** loss situations as shown in the table, and **incidental** loss situations. In some games, it is better to add some extra loss situations of a more "personal" nature to the player's role. For example, in political games, family problems caused by overwork may bring about the player's downfall.

For most games, win and loss should come in different "packages". In other words, the player and opponent should be able to win and lose in several ways. In a battle game, there could be the option of the opponent surrendering or fleeing, as well as being totally wiped out. This is more like real-life, and it introduces variety.

The way the package is selected in the game is to use one variable and link each package to a range of values. Take "The National Hero Game" as an example. When the general election comes, if your popularity is over 90%, you stay on as dictator. If your popularity is between 70 and 90%, you are head of the next government, and if it is between 50 and 70%, you are given a less important position in the government. If it is below 50%, you lose.

Starting point

Next, the starting point of the game should be determined, based on as many of the variables as possible. In general, military games start with peace, both sides' forces in formation or at base. Political and economic games, however, often start with a more hectic situation. In all cases, the start situation should not be **too** chaotic or unstable. Remember that if you want other people to play and enjoy the game, they will be put off if they lose in the first couple of turns for no readily apparent reason.

In some games, it is desirable to have a variety of possible starting-points, each different in some way. For important variables, it can be a good idea to indicate the starting value in the introduction, if the value is not to be constantly displayed. In "The Road to Valhalla", your lifespan is randomly determined at the beginning and its value is hinted at in the introduction.

Game plan

We now come to the main product of this chapter – the complete game plan. The object of this is to link together all possible situations for building the program around later on. Since computers need precise detail rather than vague ideas to work with, we have to structure the information gathered so far to produce a flowchart covering the entire game. This will be in effect a "program" for the player.

The production of this crucial flowchart is carried out by considering the starting-point of the game and deciding what happens next. At all stages, all the information given, the decisions to be taken by the player and the opponent(s) and the "natural" happenings as a result of these decisions must be thought out. This process is repeated ad nauseam until all the win and loss situations have been reached.

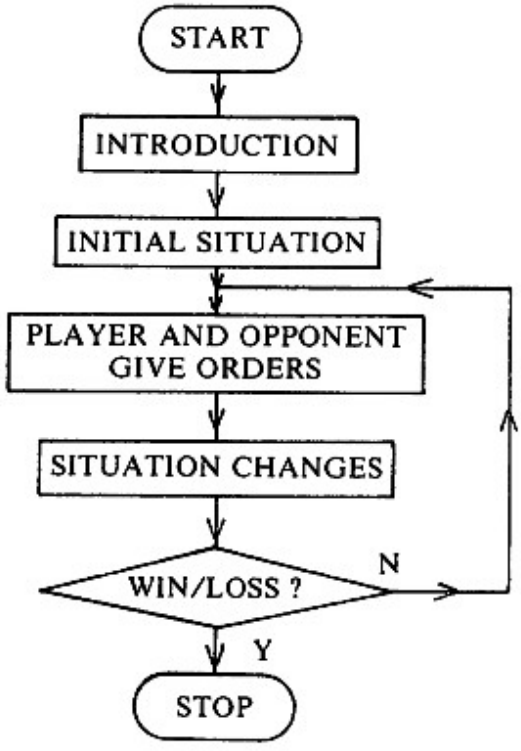
A fairly important decision regarding the opponent's turn should be made now. You have to choose between simultaneous and alternate turns. Alternate turns are like in chess, where one moves after the other and can react to the opponent's last move knowing

that the opponent can do no more until the move is made. Simultaneous moves are more like real-life in that, though the computer cannot do two things at once, namely receiving your own move and making its own decisions, it can **carry out** the moves at much the same time.

However better from the point of realism, simultaneous moves are harder to program for the computer's decisions, since a greater degree of powers of prediction are needed. You just have to weigh up the odds. But do not be put off from simultaneous moves straight away. Note that it is possible to make simple predictions for the computer – see "The Siege of Doune Castle", where the computer keeps a tally of where you have attacked before and allocates its forces accordingly, with no knowledge of your move. The relevance of the decision between simultaneous and alternate moves will become clear when ordering the parts of the game.

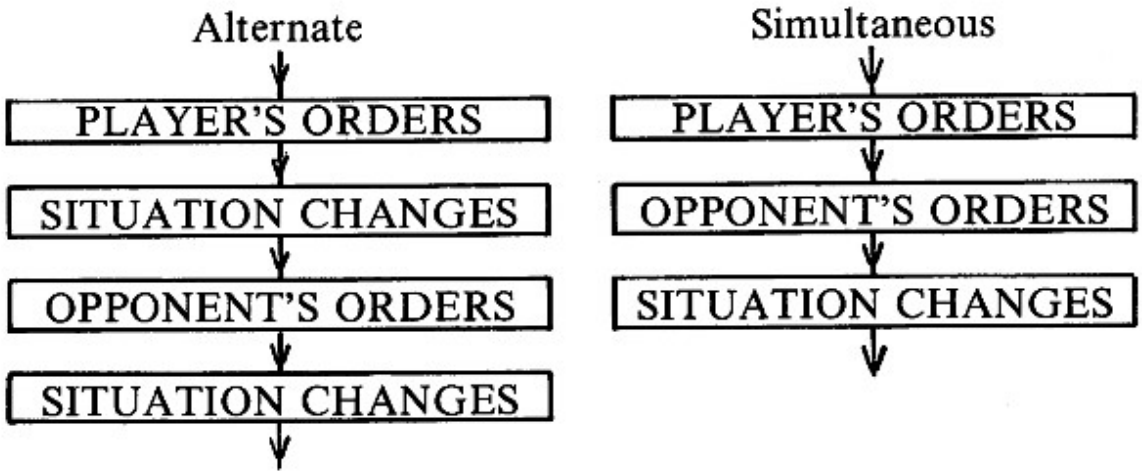
When writing the complete flowchart of the game, it is of course possible to start at the beginning and work through in intricate detail, but you'll find cross-referencing can be quite a headache while the flowchart is still incomplete.

Now, ignoring win and loss situations, and considering the opponents as other "real" players, all games follow this general pattern:



The loop represents the passage of each day (or whatever), so any other details such as the arrival of replacement equipment must be incorporated into the "schedule" as best fits the game.

The "Orders" and "Situation Changes" boxes are altered according to whether you selected simultaneous or alternate moves:



The next object is to decide the loop period – the period between consecutive order-givings by the player. For military programs, this is typically one day, though it can be less. For political and economic games, it can be almost anything up to a year. Of course, you can get round the problem by setting the game on another planet and calling the time unit the "Fleg" or "Yad".

As I have said, the diagrams summarise the general course of the game. Now, it is necessary to fill each box in in greater detail.

The first thing to do is to apply any subgames and random factors you have and fit them into the general plan at suitable places. Usually, that will be immediately before or after the player's orders are given, or just before or after the win/loss detection. The advantage in putting them after the win/loss detection is that the player is given a chance to redeem any fatal mistakes in the next loop.

After that, the Orders boxes can be tackled. In general, the more real-life options of possible orders there are, the more realistic the game. You have to decide **exactly** what the player is allowed to do – the variables the player controls and just what can be done with them. You may wish to have options to be answered with yes/no

responses, numerical decisions, or choices of fairly broad-based courses of action.

In political and economic games, the orders section usually covers a basically simple, narrow range of actions like setting taxes and giving or withdrawing financial aid. In military games, the situation is easily made more complicated. The orders can be divided into two types – deployment of existing material and requests for more material.

Check that each order has a potentially bad effect on some variable as well as some good effect. Also, in military games, there must be ways of limiting requests for more material – either by setting a maximum production rate, or by linking production to another variable such as energy or manpower.

The following table shows a selection of common inputted (player – or at this stage – opponent controlled) variables:

GAME TYPE	INPUT VARIABLES
Military	Procurement of each type of equipment Deployment/movement of men, equipment
Political	Y/N answers to courses of action Amounts of aid, etc, given Choices of actions and targets
Economic	Tax rate, expenditure on Defence, Medicine, etc

TABLE 5 POSSIBLE VARIABLES INPUT

The opponent's orders can be treated in exactly the same way, bearing in mind that the computer will have to make feasible decisions.

The next part is deciding how the situation changes as a result of the orders given. In military games, the situation changes through battles, broadly speaking. Thus in this case, the variables

controlling the battles and the effects of the battles on the variables – the number of men lost and so on – must be chosen. In political games the variable changes are basically alterations of allegiances and respective friendliness, backed up by, say, material and money changing hands in raids and agreements. In economic games, of course, the situation changes are in the physical state of the economy.

You must be fairly detailed in identifying effects, although exact relationships are not essential to decide on at this stage.

For instance, an attack involving men and tanks against enemy men and field-guns could have the following effects:

1. Advance of positions;
2. Reduction in numbers of men, tanks and guns;
3. Start of a battle – outcome decided by relative strengths, and final positions of both sides may well be altered.

In a political game, sending aid to a country could have the following effects:

1. Increase of home popularity;
2. Increase of other country's friendliness;
3. Increase of other country's allies' friendliness to lesser extent;
4. Decrease of other country's enemies' friendliness to lesser extent;
5. Decrease of balance of payments by appropriate amount.

Once this procedure has been completed for all possible orders, the win and loss detection can be introduced into the flowchart.

It is usual for loss detection to come before win detection so the player cannot win from an impossible situation – a sort of concession to the computer's probable lesser skill in planning. However, the order of detection is a minor matter and can be reversed with little effect, according to the dictates of personal choice.

For example, the player may be able to "cheat" by, say, diverting the entire forces to one front. If the opponent retaliates by attacking on another front, both attacks could succeed. Unless you

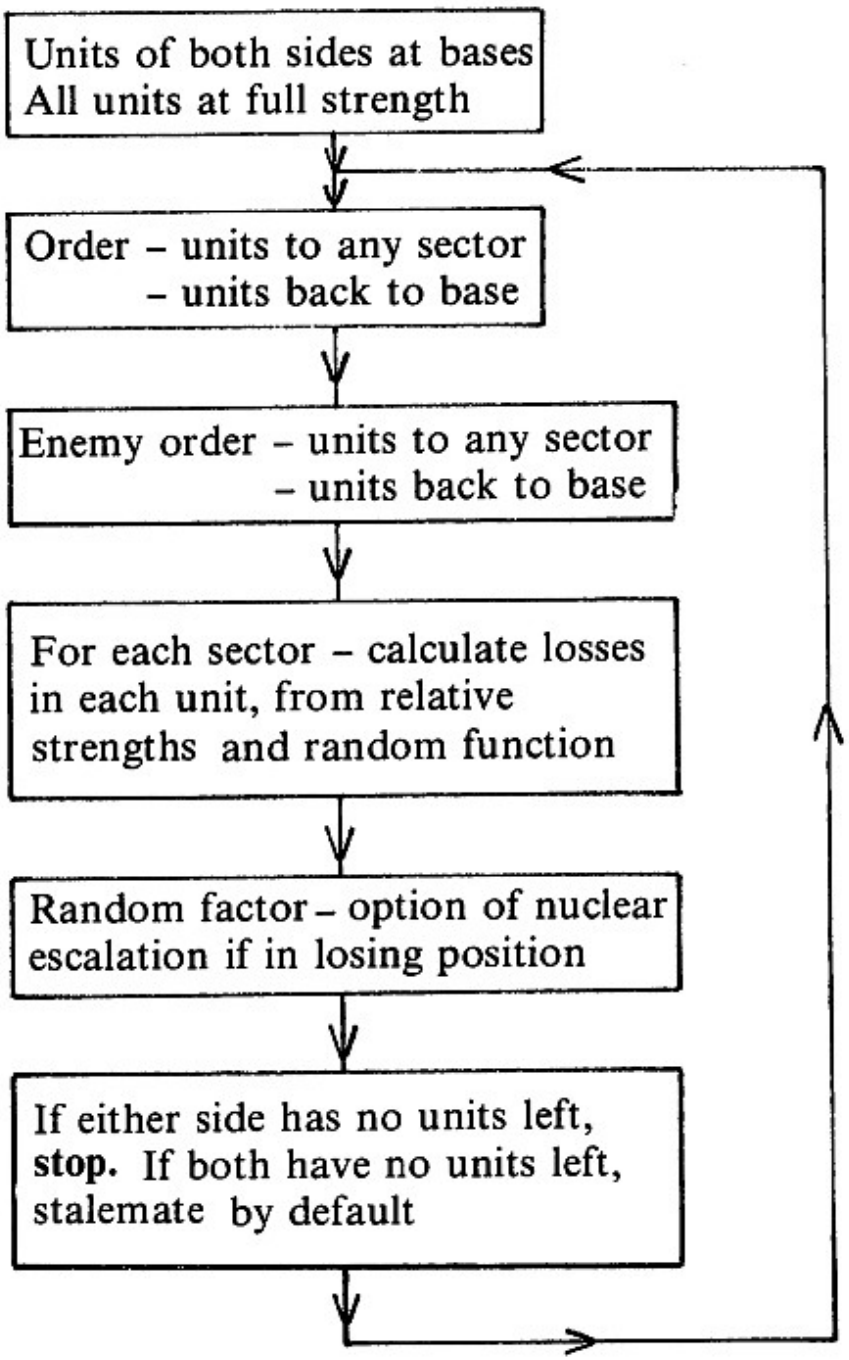
want to include an extra routine for dealing specifically with this situation (a stalemate by default), you have to decide which side gets priority.

In win and loss detection, of course, include all possible "packages" when incorporating into the flowchart.

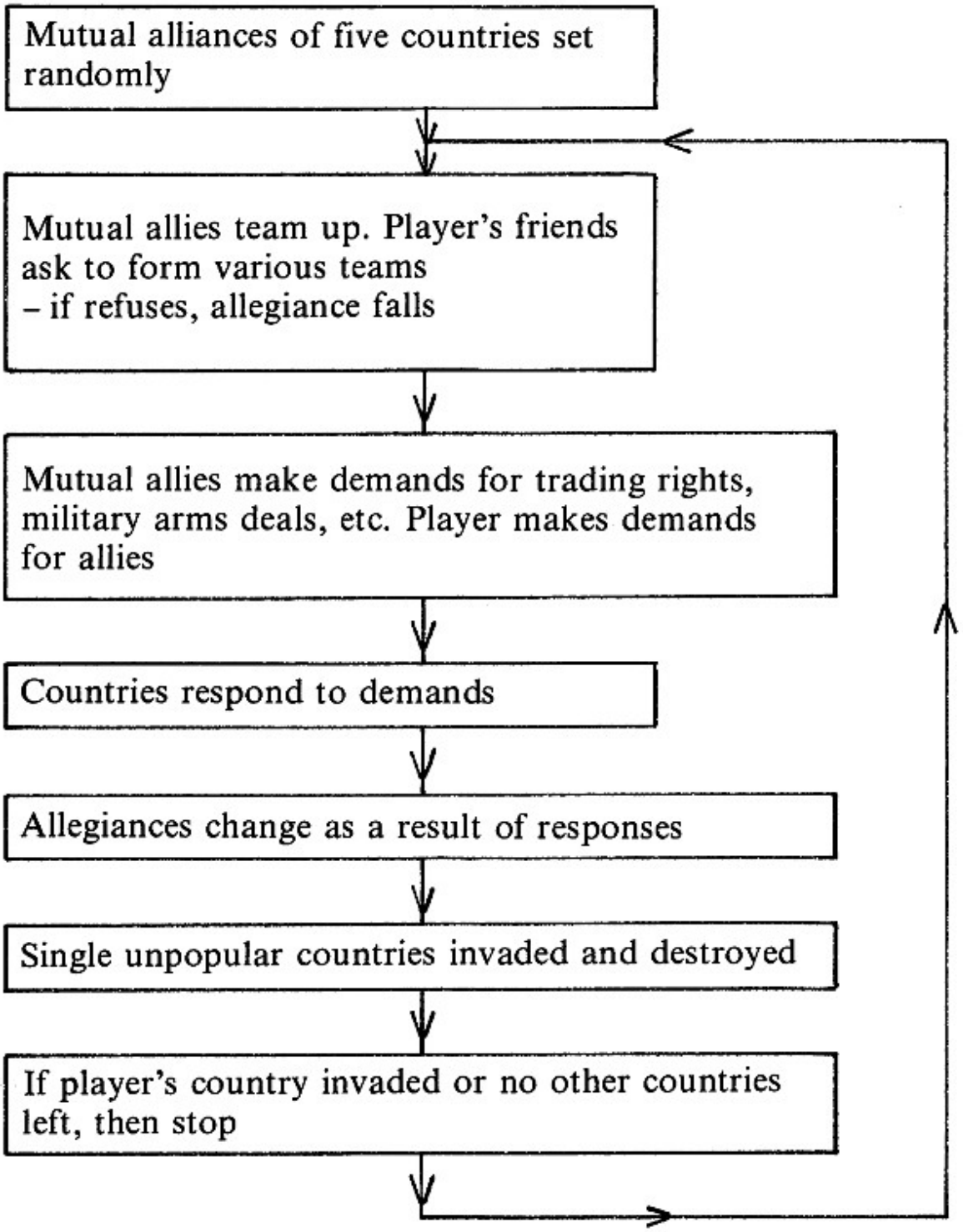
There are only two more things to do with the planning at this stage. The first is the establishment of the start situation, which speaks for itself. The other is to decide whether or not the game does actually end at the "Stop" box, or if there is to be a "Replay" option. You may also want to include a "Hi-score" part, where the "scores" are usually best times to victory in strategy games. The list of best scores should be given at the start or just before the replay option.

By the end of this chapter, you should have a flow-chart encompassing the entire game from the point of view of the player. It should look something like one of the following examples, though as the examples are of pure varieties of game, your flowchart may have sections out of each.

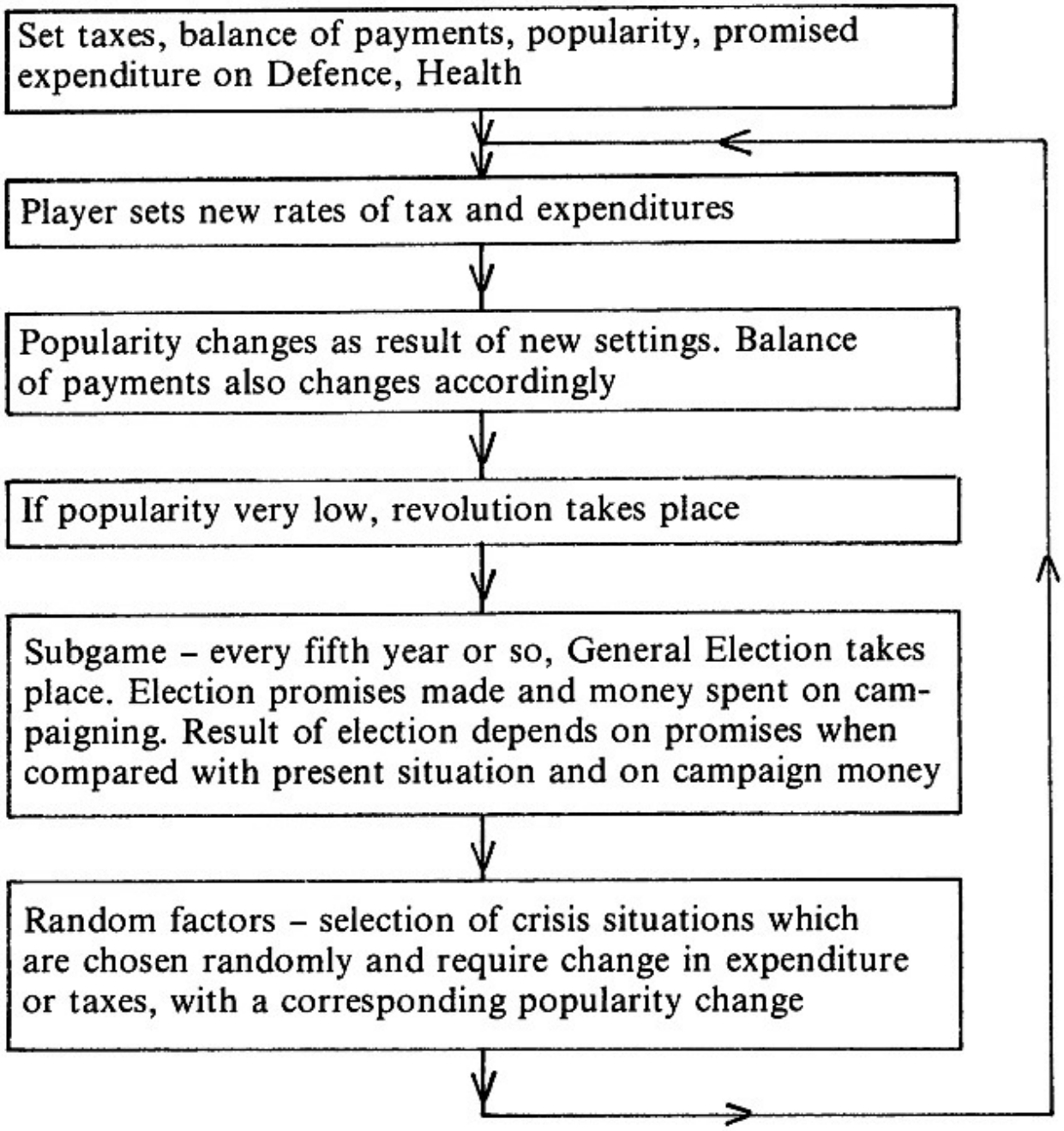
Military


Units of both sides at bases
All units at full strength
Order – units to any sector
– units back to base
Enemy order – units to any sector
– units back to base
For each sector – calculate losses in each unit, from relative strengths and random function
Random factor – option of nuclear escalation if in losing position
If either side has no units left, **stop.** If both have no units left, stalemate by default

Political


Mutual alliances of five countries set randomly
Mutual allies team up. Player's friends ask to form various teams – if refuses, allegiance falls
Mutual allies make demands for trading rights, military arms deals, etc. Player makes demands for allies
Countries respond to demands
Allegiances change as a result of responses
Single unpopular countries invaded and destroyed
If player's country invaded or no other countries left, then stop

Economic


Set taxes, balance of payments, popularity, promised expenditure on Defence, Health
Player sets new rates of tax and expenditures
Popularity changes as result of new settings. Balance of payments also changes accordingly
If popularity very low, revolution takes place
Subgame – every fifth year or so, General Election takes place. Election promises made and money spent on campaigning. Result of election depends on promises when compared with present situation and on campaign money
Random factors – selection of crisis situations which are chosen randomly and require change in expenditure or taxes, with a corresponding popularity change

CHAPTER 2
THE MODEL

In this chapter, the idea for the game from Chapter 1 is analysed to produce a reasonably sophisticated model of the interactions between the variables. This is an essential step on the road to writing the final program.

Function of the model

The model represents the variables involved and the relationships between them at all levels. Where the game plan is concerned, it comes into greatest effect in the situation-change part, serving as the "laws" of the game. It can be considered as the part of the game containing the "real-life" structure and behaviour to be imitated, in a form which a computer can potentially understand. As such, the model is the basis for the mathematical, string-handling and algorithmic lines in the final program.

Separating the variables

The first step in producing the model is to differentiate the types of variable in the game. Variables fall into five generalised categories:

1. Simple variables, e.g. money, men – displayed as they are and controlled relatively easily;
2. Complex variables, e.g. trust, honour – displayed in a "wordy" way rather than as a figure, but still controlled by fairly simple processes;
3. Indirect variables, e.g. popularity, inflation – displayed in a simple **or** a complex way, but controlled by complicated, convoluted processes depending on many factors;

4. Position variables – co-ordinates – displayed as numbers, or, in a graphic program, by screen position;
5. Position-related variables, e.g. speed, acceleration – affecting the position variables, displayed as numbers and controlled simply.

Also, any variable may be "hidden" – that is, not displayed to the player. This is one way guesswork and estimation are brought into the game.

Finding the type of variable helps when building up the model because it highlights the areas where additional work needs to be done. Simple variables can be incorporated into the model as they stand, but other types may require considerable extra treatment – converting particularly complicated ones into combinations of more simple variables, for instance.

Military games generally use simple and position variables. Position-related variables only occur in very detailed games. Political games always use some complex variables. Economic games have a large proportion of simple variables. Both political and economic games use indirect variables.

Laws

Once the variables have been differentiated, the next stage is to decide what general trends control the situation changes in the game. The trends in military games are based on the confrontation of forces of various strengths, aided by their equipment. Also, position variables have some effect in determining the effectiveness of fire at different ranges, and whether battle can take place at all at extreme ranges. In economic games, the trends represent the flow of money to the various consumers. Political games are the most challenging to compose laws for, since they represent interactions between people and affect such abstract variables as trust, allegiance and friendliness. The trends in political games are the changes in these variables as a result of various courses of action. Thus "allegiance" could be increased by a gift of soldiers in a crisis. The object of this chapter is to convert these laws into relationships between the variables.

Table 6 shows some typical laws for strategy games:

GAME TYPE	LAWS
Military	Men and equipment can be moved to/from the reserves and sectors, but not between sectors In battle, each man kills Ø.5 men per loop Tanks each destroy one tank per loop
Political	Countries get friendlier when offers/gifts made Their allies get friendlier too, but less so Their enemies get slightly less friendly These effects exaggerated for helping a country involved in war
Economic	If taxes raised, popularity falls If public expenditure cut, popularity falls If wages cut, popularity falls Balance of payments is total tax from wages, minus expenditures (simplest case)

TABLE 6 SIMPLE LAWS FOR STRATEGY GAMES

With the laws as a starting point, the variables can be introduced. The laws covering the change of variables fall into two categories:

1. Changes ordered directly, e.g. decrease in balance of payments due to payment of Defence budget;
2. Changes based on other variables, e.g. change in popularity because of tax rate and Defence budget.

In military games, new equipment supplied as ordered and decrease of food stocks when rations are supplied are type 1 laws. Battle losses and, say, the number of men deserting because of dissatisfaction of some sort fall in type 2. In political games, any simple variables usually change in type 1, whereas the indirect variables change in type 2.

In actual fact, type 1 involves changes in the inputted variables – or variables closely related to them (as closely as "men" and "men not used"). By contrast, type 2 laws involve changes in variables which are more complex functions of the inputted variables.

Type 1 laws are identified by considering each variable in turn and asking the question "What changes can the orders make?". For example, supplies of equipment may be increased here by orders for more. Ensure that there is a way of limiting orders for more equipment, by having limiting variables, for instance, which keep a tab on the value of equipment ordered at any time, so the limit can be enforced. Table 7 shows some typical type 1 laws.

GAME TYPE	TYPE 1 LAWS
Military	Equipment can be allocated to any sector No more equipment can be allocated than exists Equipment can be withdrawn to the reserves Equipment cannot be moved between sectors
Political	Friendliness coefficient towards any other country can be increased by aid and agreements Friendliness coefficient can be decreased by aiding enemies any having a strong military presence
Economic	Values of percentage taxes can be altered between Ø and 6Ø% Values of expenditures can be altered between Ø and any positive figure

TABLE 7 TYPE 1 LAWS

Type 2 laws are identified by considering each variable in turn and saying "Which other variables have an effect on it, and how?". In this way, the laws controlling battle losses and outcome are revealed in military games, the laws controlling the behaviour of

the economy in economic games, and the laws controlling mutual status in political games. Table 8 shows some typical type 2 laws.

GAME TYPE	TYPE 2 LAWS
Military	Chemical and biological warfare destroy a fixed proportion of a force Each piece of equipment destroys a fixed proportion of each type of equipment Advances and retreats are found from the relative numbers of men and tanks (each tank is worth five men)
Political	When your country is least loved by the others and you do nothing about it, your credibility falls (so other countries are even less likely to give you aid and support in war) Popularity falls in country if there are no trade agreements; it also falls if too much military support is given to other countries
Economic	Popularity changes based on obvious variables such as tax rate and pay, with different weightings for relative importance Cost of living changes according to "economic climate" composed of size of industry, foreign trade, etc Industrial growth related to industrial aid and "market potential" – amount of money left over after people have paid the cost of living

TABLE 8 TYPE 2 LAWS

Note that position and position-related variables are almost always controlled directly, i.e. by type 1 laws.

Before going on, check that the laws originally made up are covered by the new laws. This is necessary because the second set of

laws – types 1 and 2 – do not come from the first set directly, and so there could be omissions in the final set.

Once the variables have all been connected adequately with laws, the exact mathematical relationships can be worked out. Generally, type 1 laws are simple additions and subtractions, possibly with some weightings, whereas type 2 laws are more complicated. Some examples of type 2 laws are shown in Table 9. It was not considered necessary to give detailed examples of type 1 laws since they are mainly simple transfers and reallocations of fairly simple variables.

GAME TYPE	TYPE 2 LAWS – EXPRESSED MATHEMATICALLY
Military	Men lost in battle = original number × ⅛ proportion of chemical warfare shells + ½ enemy tanks + ¼ enemy field guns Similar expressions for tanks and field guns lost
Political	Friendliness of a country = original friendliness + £ of aid/1 million + £ of aid to the country's friends /100 million – £ of aid to enemies /10 million
Economic	Popularity = original popularity – % of income tax/100 + % pay rise/50 + % increase in Health budget/200

TABLE 9 EXAMPLES OF LAWS EXPRESSED MATHEMATICALLY

Note that quite often, the results of some mathematical operations (e.g. those working out men lost in battle) may have to be altered if they have made the figure for the variable incorrect (e.g. "men left" could be negative in some cases – this would have to be converted to read zero) – see also the section on "Ranges" in this chapter.

Complex laws

You may find that in some games, it is impossible to express one or more of the type 2 laws straight off. This is a common problem in more complicated political games, involving close-knit alliances and threats. The only way to deal with these is to analyse the basis of the law in greater detail – to penetrate to a more fundamental level. Eventually, you should find that introducing a new variable supplies the missing link. The new variable is often very abstract, usually requiring several separate values to cover all the "players", for mutual feelings of some sort.

The analysing process is not easy to explain – instinct plays a large part in it. But to help you have inspirations, Table 1Ø shows possibilities for dealing with particularly complex laws.

LAW (IN WORDS)	METHOD OF SIMPLIFICATION
In a time of prolonged peace, if the armed forces are above a certain size, popularity falls	Create a new variable to keep track of the time since the last war or major crisis; this is used to refer to when comparing force size and length of peacetime
Unrest in conquered territories depends on background of the conquering empire's expansion (reasons)	Create a new variable for rate of expansion, and another for peacefulness of the empire's dealings with the territory before expansion, and another for strength of enemy empire beforehand

TABLE 1Ø SIMPLIFIED COMPLEX LAWS

It can be seen from the table that the essence of solving the problem of complex laws is to express any particularly awkward or abstract quantities as a series of simpler variables. Then a relationship must be created between them as for simple laws.

Another method of finding mathematical relationships

If the generalised basic laws you formulated in Chapter 1 were detailed enough, there is another possible way of finding the precise relationships for some laws. If you take the generalised laws, it should be possible to express some of them mathematically straight away.

Here is an example from "The National Hero Game":

> When the general election is called, if popularity is low, no change. If it is higher, it falls in proportion to the time till the election.

This could be re-expressed as:

> If popularity above Ø.5, popularity falls by time till election/ 1ØØ when the election is called.

Such a group of lines can therefore often be condensed into a single mathematical line.

Frequently, it is possible to convert the initial set of lines into a mathematical line with a slightly different meaning (i.e. it works in a slightly different way). This has the advantage that the writer of the game may not be able to predict the outcome quite so easily, which makes the game more fun to play. Also, having more complex lines means that players who are inclined to "cheat" by looking at the listing are less likely to understand the laws, unless they are prepared to spend quite a lot of time working them out.

Remember, however, that this method of establishing the laws is not sufficiently exhaustive for the other method to be dispensed with – it only deals with the checklist of laws made in Chapter 1, and rarely includes all the type 1 and 2 laws found in this chapter.

Ranges

Having fixed the relationships between variables, the ranges and possible values for the variables must be decided on. Of course, most variables will be integers – in military and economic games at least – and most will also be positive numbers. Position variables in games using a "board" fall in a relatively small range (Ø-7 for each of 2 variables in chess). The main task is to ensure that all equipment variables can only be positive integers, or zero, that popularity can only fall between Ø and 1ØØ% and that position variables fall within any limits. Note that in economic games, many variables can be negative (representing % decreases in expenditures), or fractional to one or two decimal places (more accurate popularity figures). If the limits for the variables and the number of decimal places they are taken to are noted down, you can check all relationships involving them produce acceptable results in all circumstances. As many relationships can produce negative values for variables such as "men left", it is possible to note that a routine must be added to convert negative numbers to zero in the appropriate places. Also, go over all mathematical relationships which should produce whole numbers and – if they contain decimal factors (e.g. × Ø.1) or division processes – add an INT!.

Sequencing the laws

Next, the laws can be put in order of use. Orders of each type are grouped together, as type 1 are used in the "players' orders" sections of the game plan, and type 2 are used in the "situation changes" sections. Sequencing is particularly important in military games, where for example replacement equipment as calculated at the "Orders" stage is not added on until the end of the loop – in real life, supplies would not be available until later.

Now, the model to date can be added to the game plan from the last chapter – the "situation changes" box can be filled in in greater detail, and the orders given by the players "carried out". At this stage, the precise effects of subgames and random features should be worked out in much the same way as for any other law. Once this is done, they ought to be practically finished with, as they are ready-sequenced.

Stabilising the model

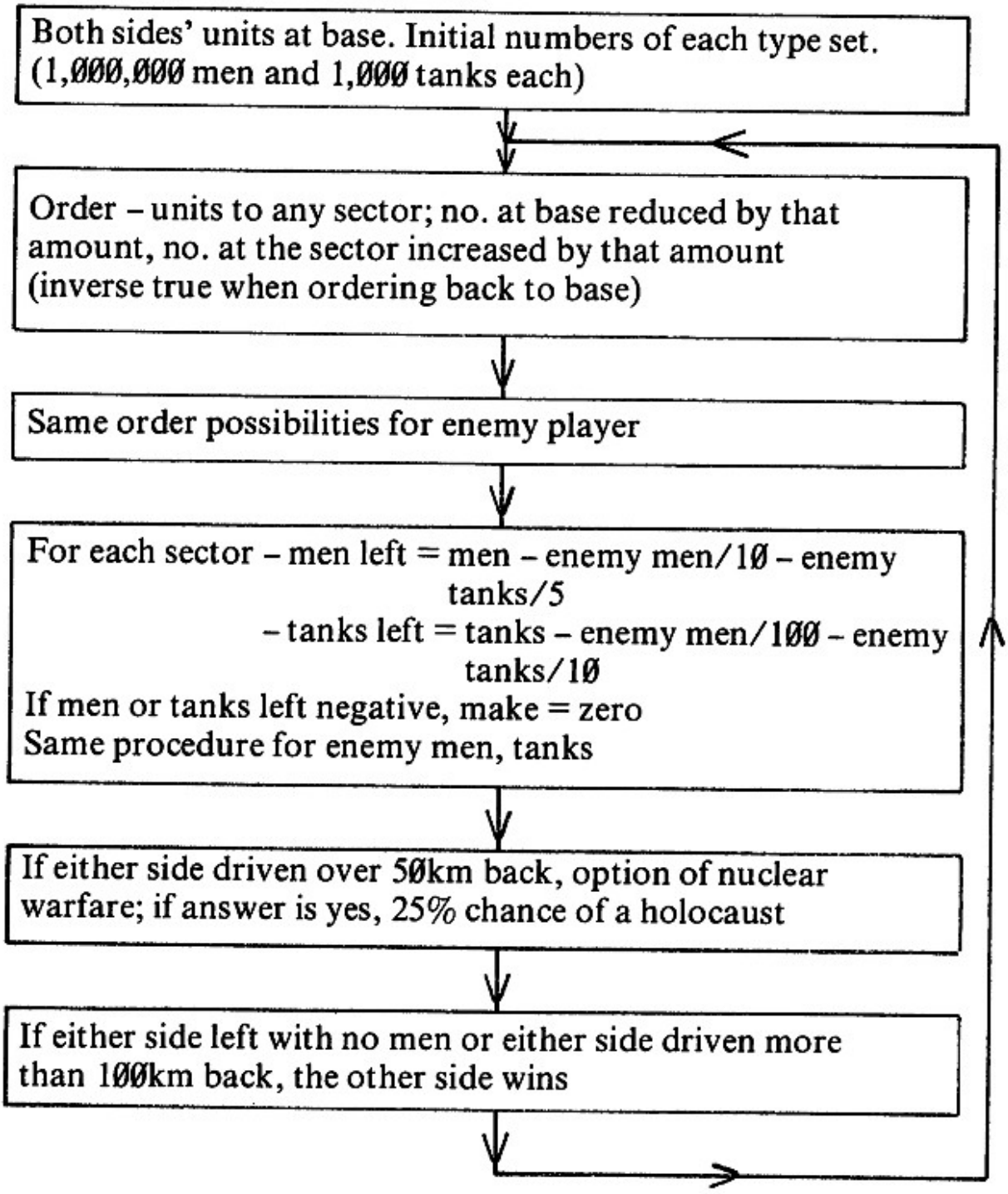
There are probably minor points about the model in its present condition which could be improved. These are most likely – and potentially most serious – in economic games, but military games have such faults too. For instance, it is possible that, in a game where a certain area of enemy territory must be occupied to win, it could take a very long time with the forces depleted by battle. I found such a fault in "Third World War" – though by that time the program was written, and had to be altered in its final form.

As was said, this problem is greatest in economic games. This is largely because there is no opponent, so the model is all the more crucial. In economic games, the main source of trouble is often a tendency for drastic changes to take place in the state of the economy. It is all too easy to attempt an accurate copy of a real economy, with all details such as inflation, growth, recessions, stock exchanges and so on, but the task of correctly evaluating the relationships between these quantities is of Herculean proportions. As a result, inadequate laws are used and the game shows such alarming traits as swings in inflation between —1Ø% and 3ØØ% – in one loop. After all, if this was not so, anyone could solve the world's economic problems and all the economists would be out of a job.

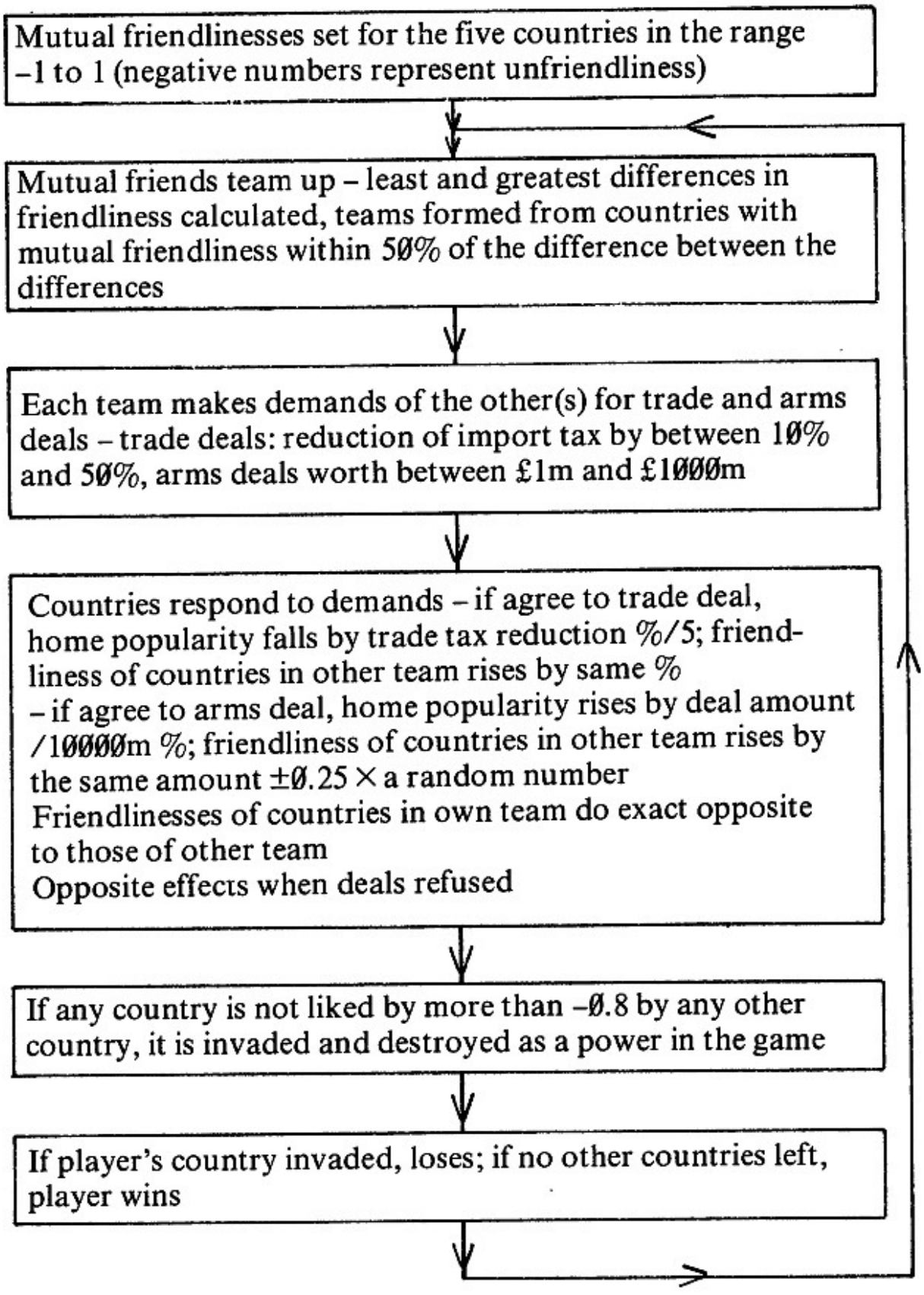
But problems of numerical instability can be caused in any game. They can be minimised by increasing the "inertia" of the variables – making it impossible for them to change by more than so much in one loop. On the other hand, when a player gives drastic orders, the game should be able to respond equally drastically – by having a revolution or a nuclear war, say.

Once these problems of stability have been sorted out, ensuring win and loss, and a realistic model created, the game plan can be considered complete. It should look something like the following simple plans which are, again, for pure games.

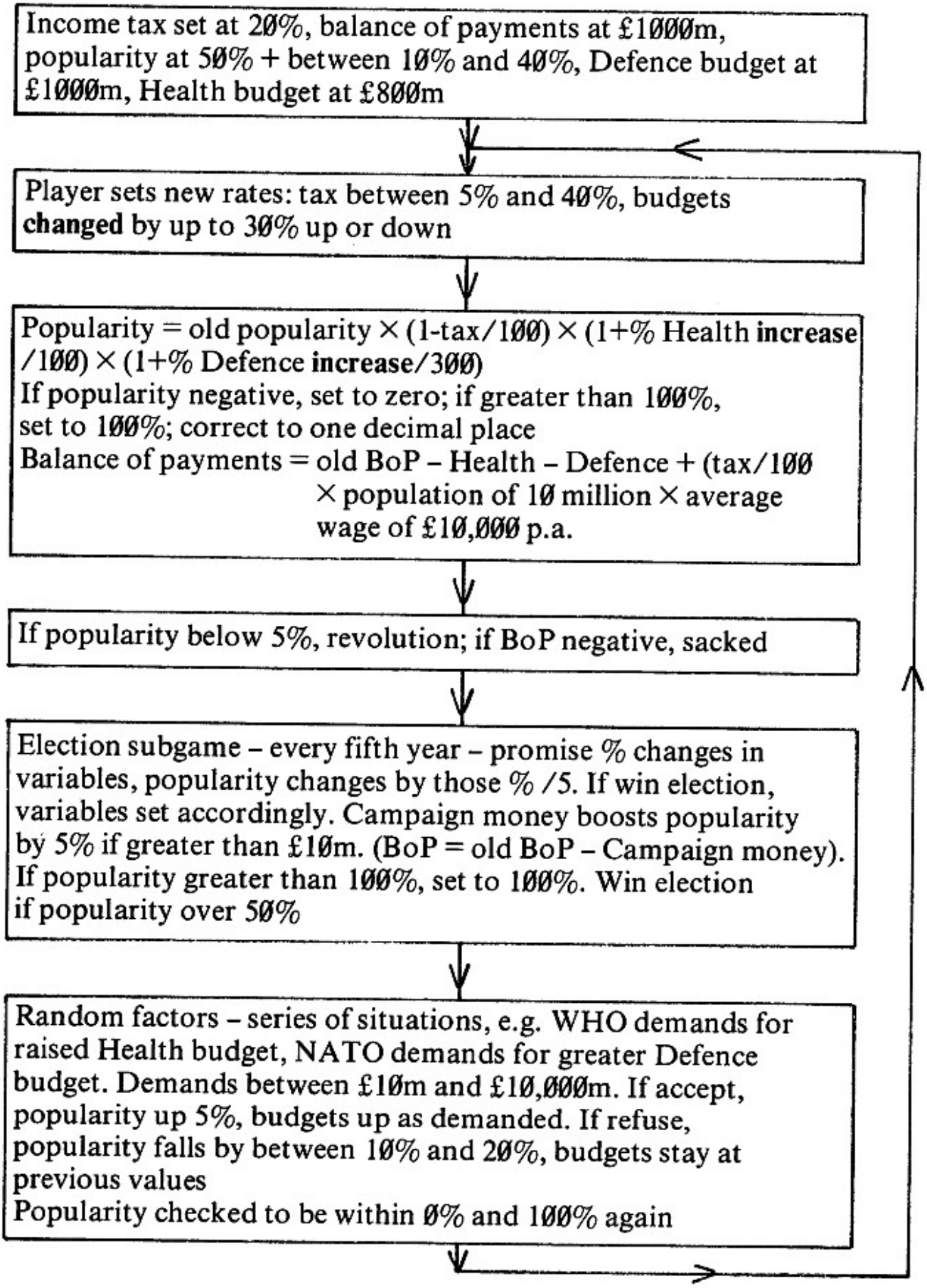
Military


Both sides' units at base. Initial numbers of each type set. (1,ØØØ,ØØØ men and 1,ØØØ tanks each)
Order – units to any sector; no. at base reduced by that amount, no. at the sector increased by that amount (inverse true when ordering back to base)
Same order possibilities for enemy player
For each sector – men left = men – enemy men/1Ø – enemy tanks/5
– tanks left = tanks – enemy men/1ØØ – enemy tanks/1Ø
If men or tanks left negative, make = zero
Same procedure for enemy men, tanks
If either side driven over 5Økm back, option of nuclear warfare; if answer is yes, 25% chance of a holocaust
If either side left with no men or either side driven more than 1ØØkm back, the other side wins

Political


Mutual friendlinesses set for the five countries in the range –1 to 1 (negative numbers represent unfriendliness)
Mutual friends team up – least and greatest differences in friendliness calculated, teams formed from countries with mutual friendliness within 5Ø% of the difference between the differences
Each team makes demands of the other(s) for trade and arms deals – trade deals: reduction of import tax by between 1Ø% and 5Ø%, arms deals worth between £1m and £1ØØØm
Countries respond to demands – if agree to trade deal, home popularity falls by trade tax reduction %/5; friendliness of countries in other team rises by same %
– if agree to arms deal, home popularity rises by deal amount /1ØØØØm %; friendliness of countries in other team rises by the same amount ±Ø.25 × a random number
Friendlinesses of countries in own team do exact opposite to those of other team
Opposite effects when deals refused
If any country is not liked by more than –Ø.8 by any other country, it is invaded and destroyed as a power in the game
If player's country invaded, loses; if no other countries left, player wins

Economic


Income tax set at 20%, balance of payments at £1000m, popularity at 50% + between 10% and 40%, Defence budget at £1000m, Health budget at £800m
Player sets new rates: tax between 5% and 40%, budgets changed by up to 30% up or down
Popularity = old popularity × (1-tax/100) × (1+% Health increase /100) × (1+% Defence increase/300)
If popularity negative, set to zero; if greater than 100%, set to 100%; correct to one decimal place
Balance of payments = old BoP – Health – Defence + (tax/100 × population of 10 million × average wage of £10,000 p.a.
If popularity below 5%, revolution; if BoP negative, sacked
Election subgame – every fifth year – promise % changes in variables, popularity changes by those % /5. If win election, variables set accordingly. Campaign money boosts popularity by 5% if greater than £10m. (BoP = old BoP – Campaign money). If popularity greater than 100%, set to 100%. Win election if popularity over 50%
Random factors – series of situations, e.g. WHO demands for raised Health budget, NATO demands for greater Defence budget. Demands between £10m and £10,000m. If accept, popularity up 5%, budgets up as demanded. If refuse, popularity falls by between 10% and 20%, budgets stay at previous values
Popularity checked to be within 0% and 100% again

CHAPTER 3
GAME PLAN TO PROGRAM FLOWCHART

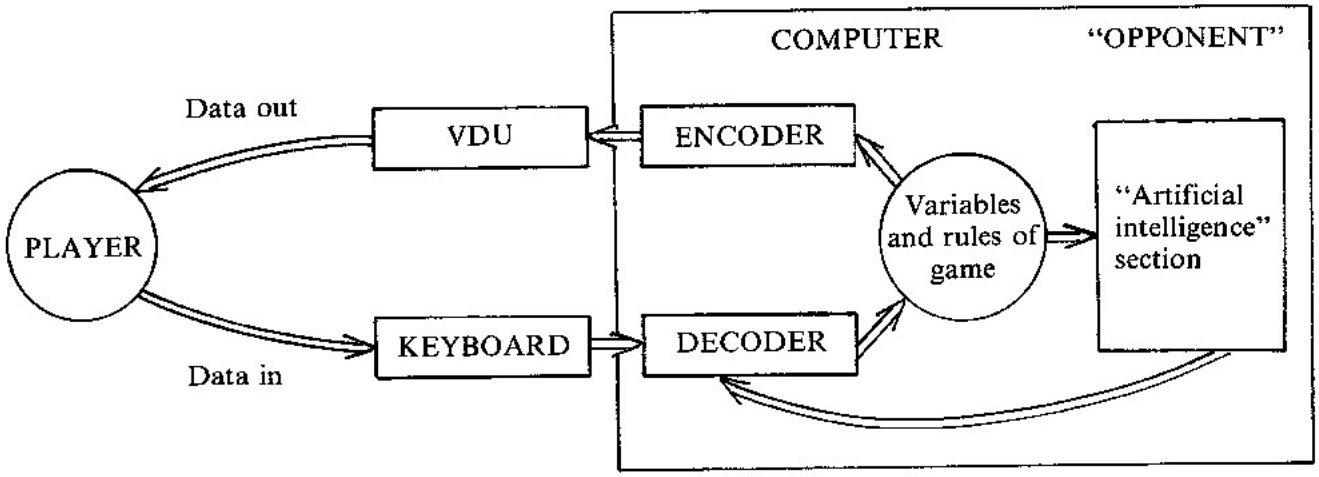
In this chapter, we convert the game plan built up in the last two chapters into a flowchart more suited to the computer's view of the game. In fact, we have to incorporate the game plan into a greater plan including the instructions and information necessary for the computer to generate the game. From the computer's point of view, the game plan to date has the major shortcoming that it assumes the player knows what is going on – the values of the variables – at the appropriate times without being told. Unfortunately, as most people cannot magically see what a computer's circuits are reading, the computer has to go out of its way and display the relevant information at intervals.

Function of the computer

As the whole strategy game is based on the computer, it would be a good idea to establish what exactly it does.

Basically, it generates the entire situation. It is the whole world in a political program, everything on a battlefield in a military game, the country's economy in an economic game. It contains all the factors and rules involved, and knows when to implement them.

In contrast, the player merely takes the part of **one** character, linked to the game via the VDU and keyboard. The computer feeds information to the player on the VDU and receives instructions from the keyboard. The situation is in effect as shown below:


COMPUTER
"OPPONENT"
Data out
VDU
ENCODER
PLAYER
Variables and rules of game
"Artificial intelligence" section
KEYBOARD
DECODER
Data in

It is an inescapable fact that computers can only deal with numbers, simple mathematical relationships, and to a very limited extent, strings of characters. Yet, any strategy game must appear realistic to the player if it is to be enjoyable. The answer to this lies in the arrows in the diagram labelled "Data out" and "Data in". By the way data is presented and instructions inputted, an **atmosphere** of realism is created.

Data is presented in a way which matches the subject of the game. This may be achieved by using graphics or even sound effects, but details of these techniques are beyond the scope of this book. So we shall concern ourselves only with I/O (Input/Output) languages and the basic layout of the display. These are dealt with in Chapter 5.

Constituent parts of a program

Program lines form sections which fall into three basic categories:

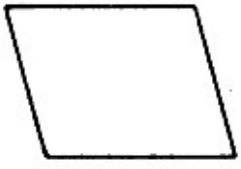
1. **Structural sections** – sections which order the execution of commands and routines. Structural commands in simple BASIC are GOTO, GOSUB, FOR – NEXT, and IF – THEN.
2. **Input/Output sections** – screen addressing, decoding inputs, encoding outputs. Note that en- and de-coding require the third type of section, especially where the display uses graphics.
3. **Mathematical sections** – covering any line where operations are carried out on variables.

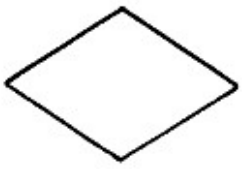
The game plan up to the present really only covers the mathematical sections from the model.

In the rest of the flowcharts in this book, the following convention will be used:

Structural – branches

Input/Output – main routes – arrows in program



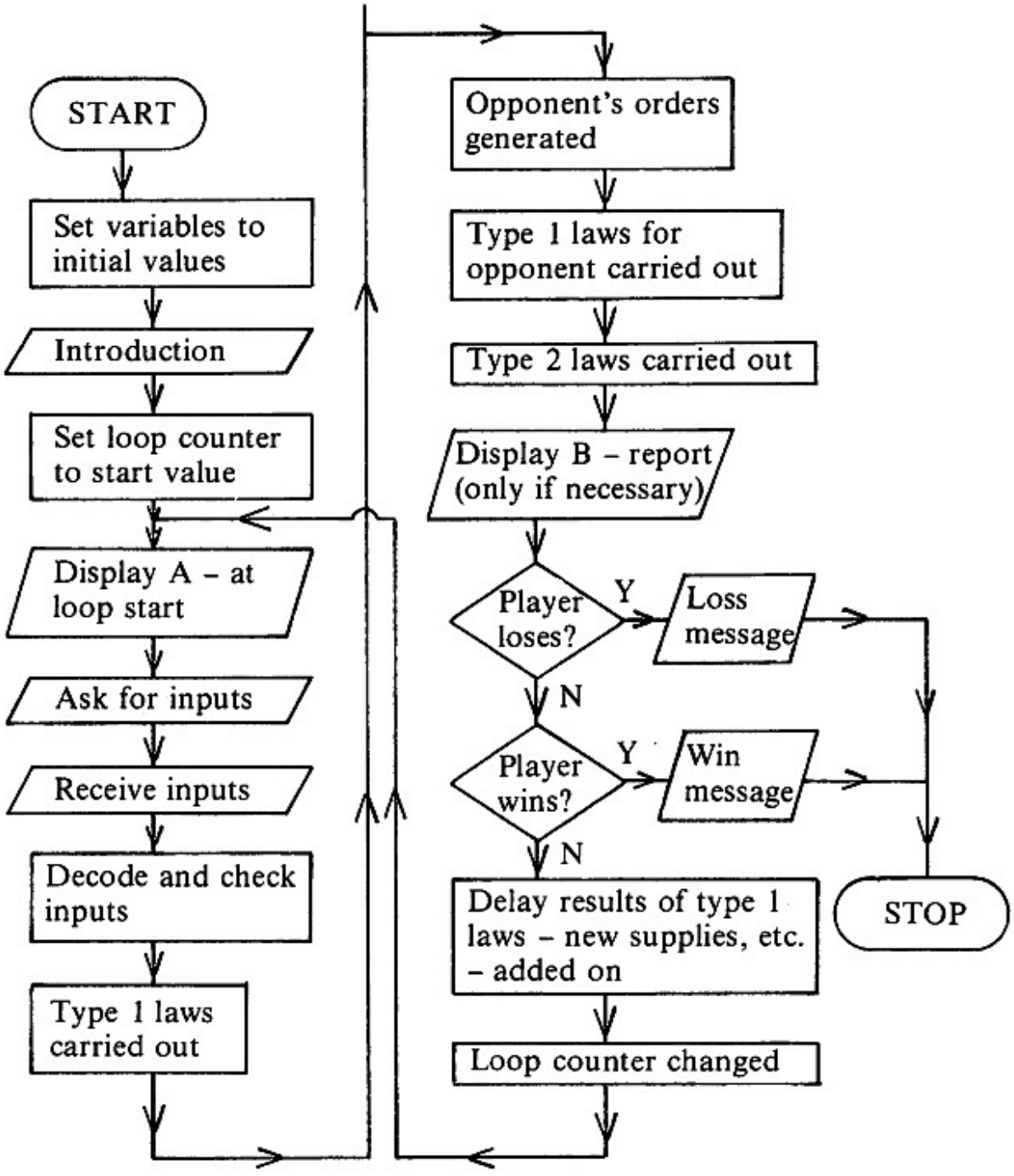


Mathematical, etc.

By the end of this chapter, all the structural sections should be present in the diagram, and all the mathematical sections except for the opponent's move should be completed. The I/O information should be known in general terms, though detailed I/O management is dealt with in Chapter 5.

Program structure

Although the exact structure of a program depends fundamentally on the game, we can make a generalised flowchart (leaving subgames and random factors aside for a while):



The above diagram is for simultaneous moves – for alternate moves, the type 2 laws would also be carried out after the player's type 1 laws are. Of course, for games with no effective opponent (i.e. most economic games) all opponent-dedicated boxes would be omitted. If more than one opponent is used, the same laws can be repeated for each one, using a FOR-NEXT type construction.

Notepad variables

Incidental "notepad" variables can be added after you have adapted the program flowchart to your particular game plan. These are used either as a part-way stage to shorten repeated calculations, e.g.:

```
LET Danger Factor = Radiation Level + Poison Concentration
LET Men = Men - Danger Factor
LET Food = Food - Danger Factor/2Ø
```

or to keep a tally of an incidental quantity, such as the time since the last election, or to store old values of a variable so the change in the value can be worked out, or to remember an outstanding value while a whole list is checked – as in finding the winner of an election, where the best so far is stored and each subsequent result compared with it. Notepad variables can be added to any such cases at this stage of the planning.

Display content

The content of displays A and B in the program flowchart is relatively similar in all the types of program. Display A generally shows the information the player uses directly to decide on the orders – numbers and dispositions of unit, immediate political situation, current tax, popularity, expenditure and balance of payment values and so on through the principal variables of the game. Display B on the other hand functions to report the results of the orders – battles, confrontations, industrial reports and so on.

An additional display, display C, can be added just before the loop variable is changed. This is most important in military games – and

rarely used in the other types – and these contain reports on matters not directly concerned with the main orders, such as equipment replacement and the number of men who deserted or died from disease.

The precise phrasing and screen location of output information is dealt with in Chapter 5. At this stage, only the basic information should be identified.

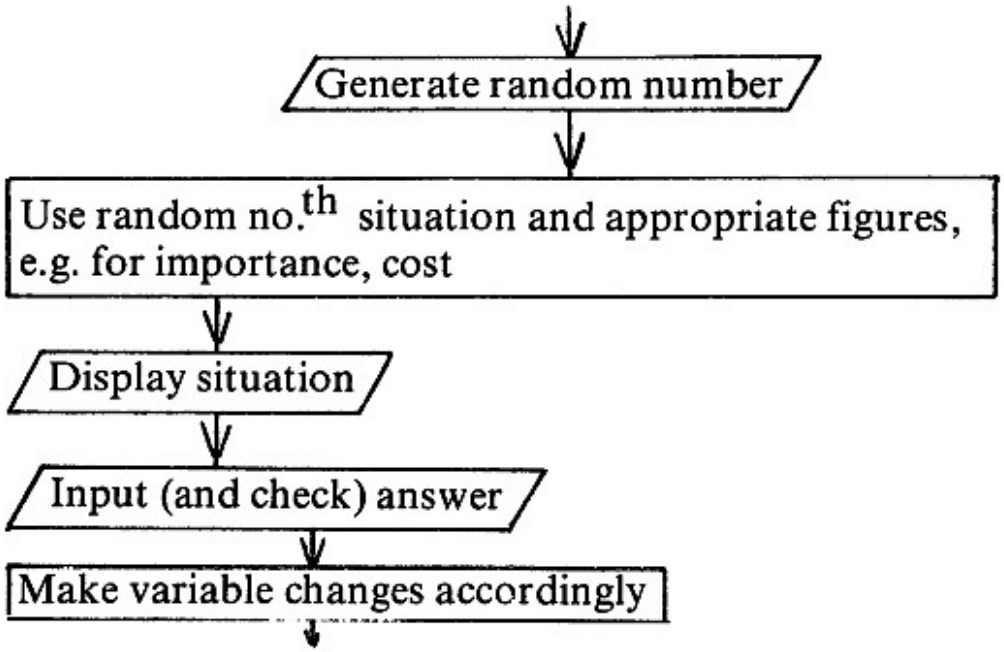
In addition, the information to be given in the win and loss displays can be chosen. For instance, the length of time you took to win/lose and your final position are possibilities. Messages commenting on your handling of the game (i.e. insults) are also good.

Flowcharts for random factors and subgames

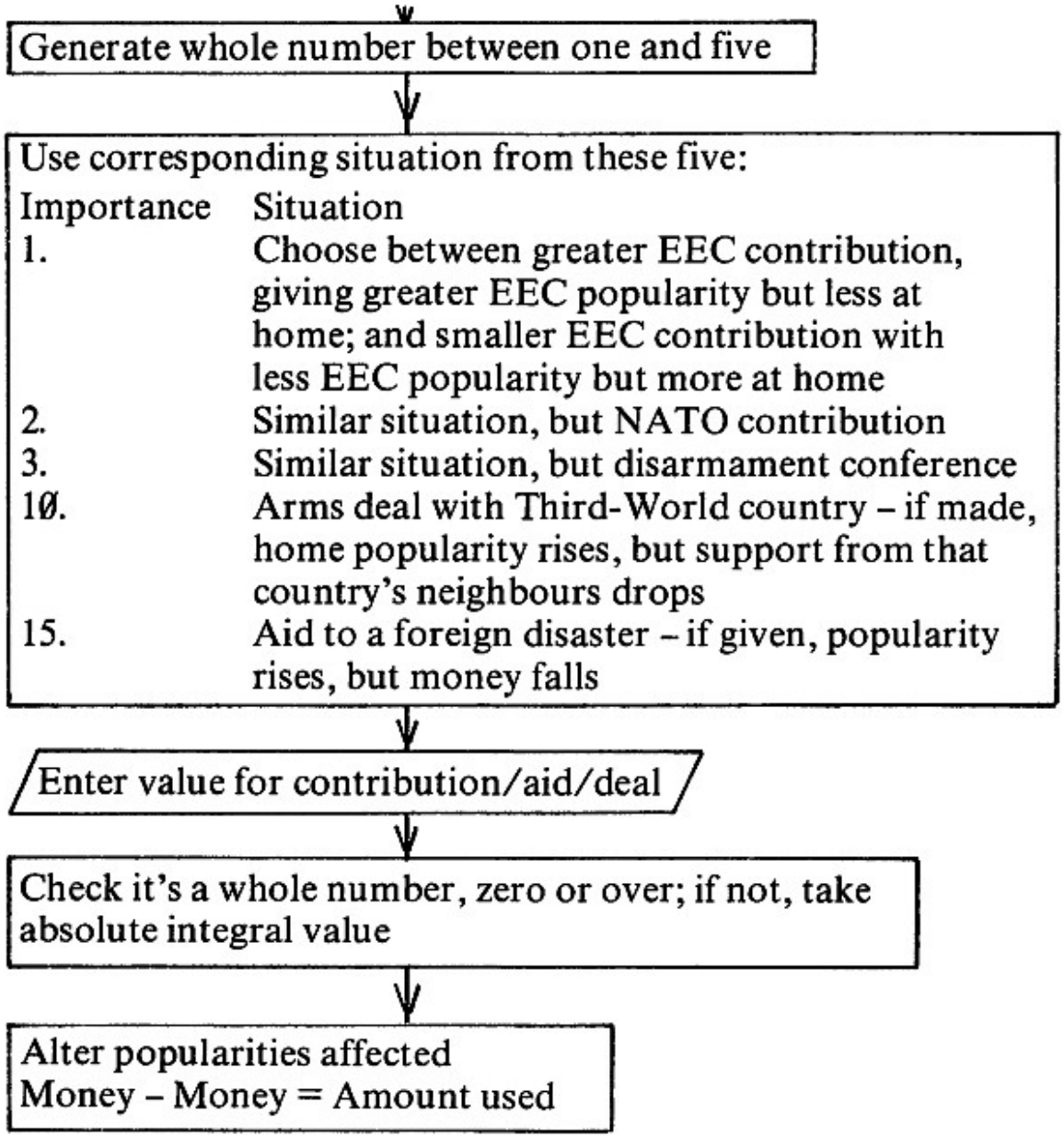
Next, we should complete the subgames and random factors in greater detail. Unfortunately, the total number of possibilities means that only a relatively small example section can be included to deal with these.

As we established before, subgames and random factors affect fairly important variables, such as men and popularity. Random factors do this in two ways – by presenting a "chance pile" of situations, each of which affects the relevant variables in a slightly different way, or by adding an extra complication to the game, such as a supplementary rule, e.g. if men remain in camp, a certain proportion of them will die of a disease.

The first sort of random factor is generated by the following general flowchart:

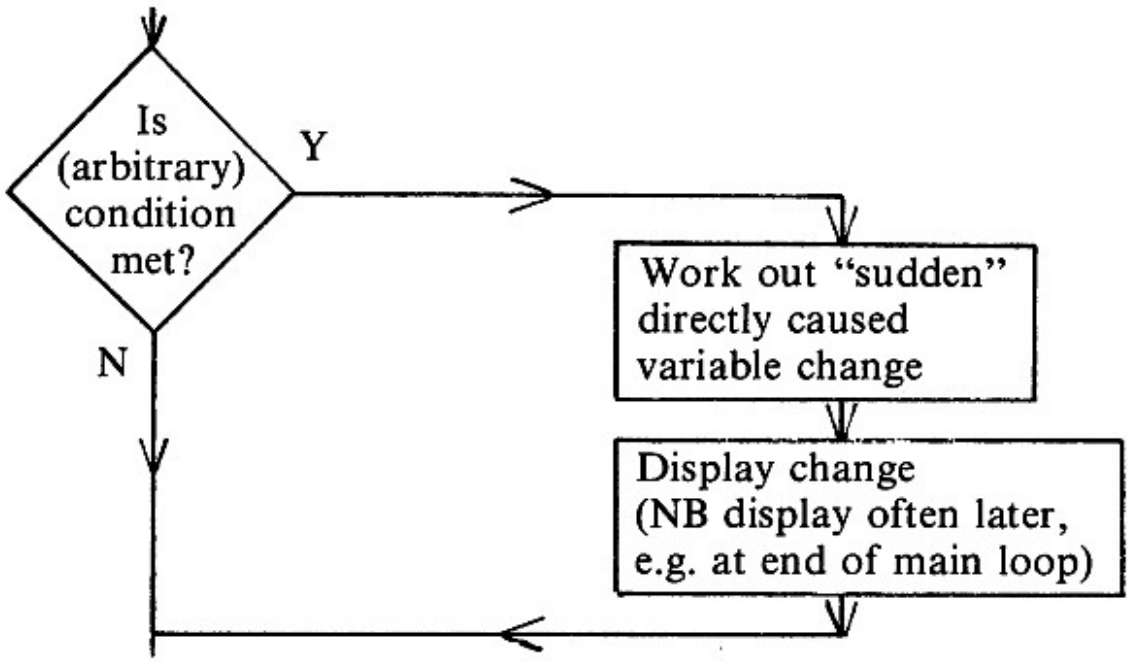


Here is an example of this type of random factor:

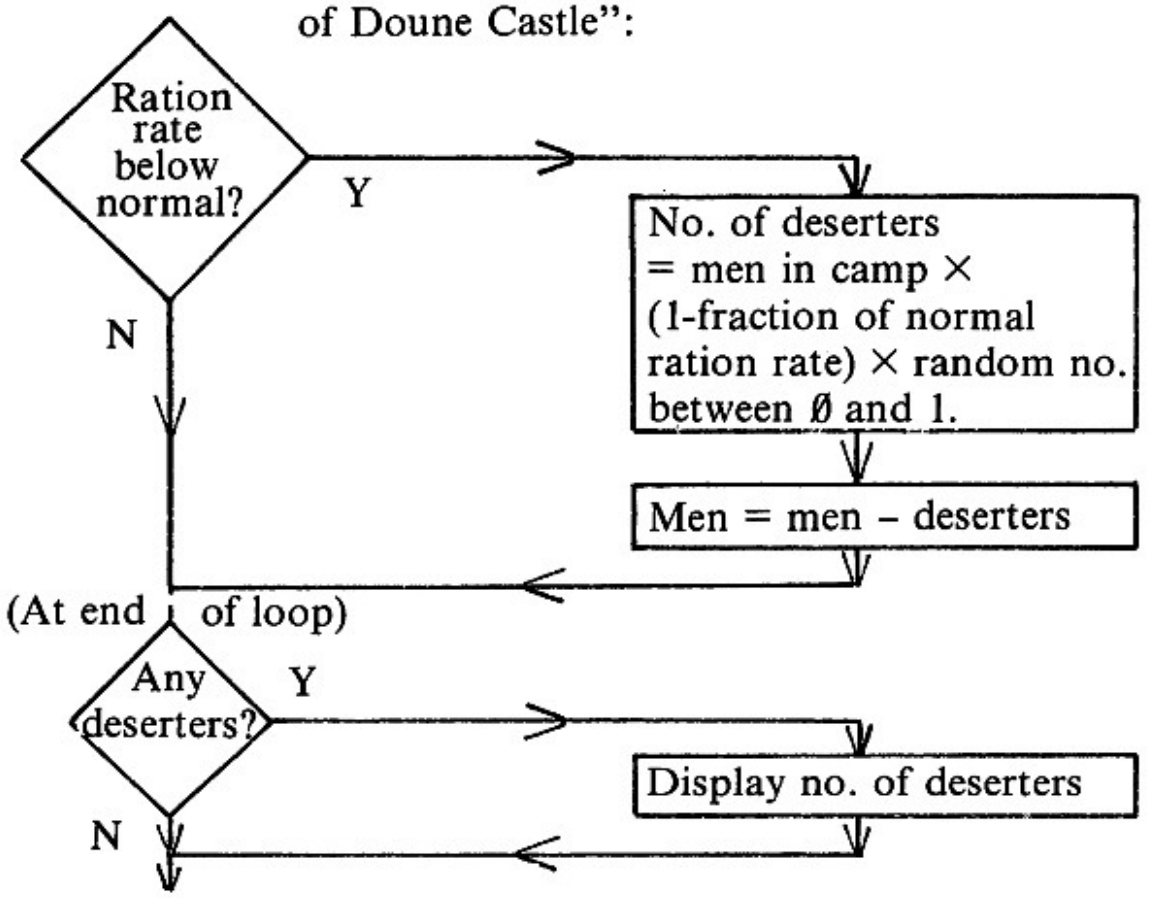


The random factor can be made so that the same two or three variables are affected in each case. Then, only the wording of the display need be changed; and perhaps numerical weightings or importance.

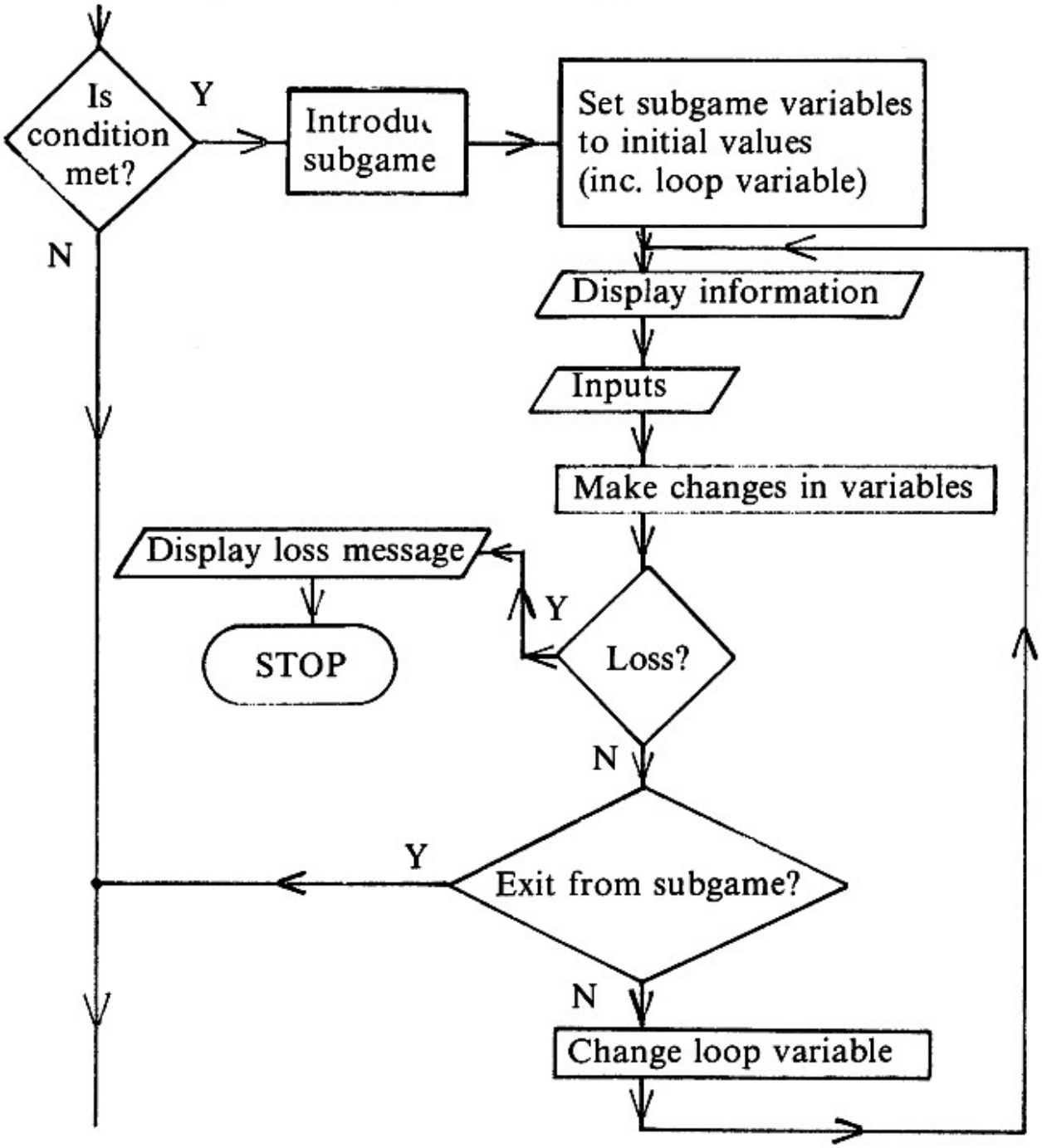
The second sort of random factor is generated by the following flowchart:



Here is an example, based on the "deserters" routine in "The Siege of Doune Castle":



Subgames have features from both types of random factor, but are more complicated and (should) contain a loop. An average subgame could be generated by the following routine:

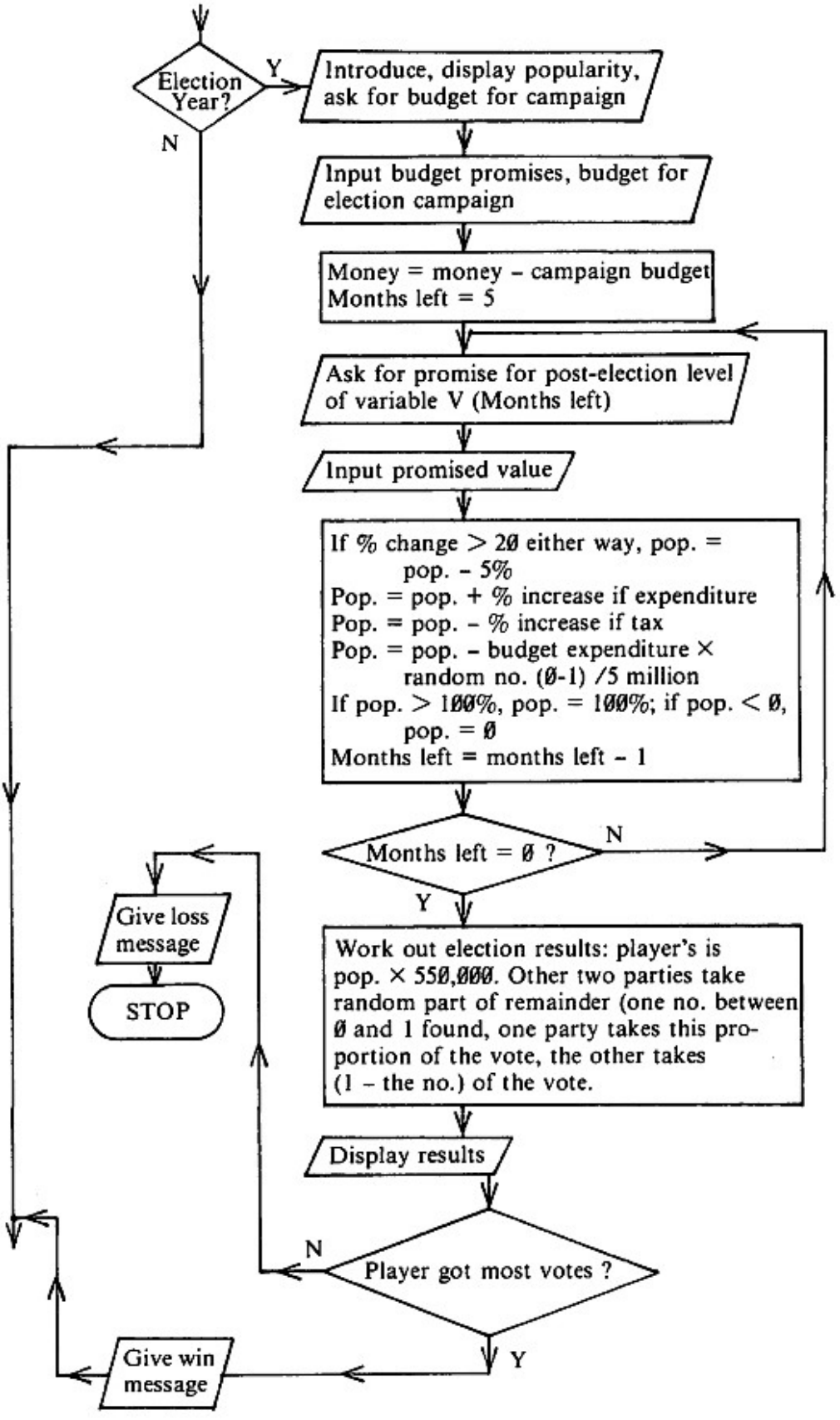


This diagram shows a subgame with a structure similar to the main game, though simpler. It is related by some of its variables – rarely all, though it does not have as many as there are in the main game.

Other subgames are made even simpler by missing out any looping. In the general election routines in, say "Corridors of Power", the procedures are not repeated – the player makes one set of election

promises and the result is decided on that, with no political infighting in the run-up to the election.

The following diagram shows a flowchart for an election routine **with** a repeating loop:



Once you have added the various parts of your game to the general flowchart, completed the lists of display information and connected up the subgames and random factors, you will notice that one box of the flowchart has been untouched. The detailed analysis of this, the opponent's orders box, is the subject of the next chapter.

There is no longer any point in considering the program flowchart in its entirety. The full diagrams would be extremely long. From now on, the constituent sections will be dealt with separately. If you need fuller examples, you are advised to refer to the programs themselves, at the end of the book. They should be relatively clear, and all the important routines are titled.

CHAPTER 4
THE OPPONENT'S ORDERS

The object of this chapter is to produce a method for the computer to give its own orders in its capacity of opponent. In alternate move games such as chess, the opponents' orders are termed the **reply move.** Since this name is short and descriptive, we will use it to refer to all types of opponent move, including simultaneous moves.

The reply move is homologous to the player's inputs, and indeed the process the computer uses must be similar. The problem of reply move generation is closely allied to the field of artificial intelligence – programming computers to solve complex problems and handle tasks on their own. It is probably the most interesting and challenging part of strategy game programming.

Product of reply move generation

It would be best to consider exactly what is required of the computer in making its choices.

1. It must make a move in **all** circumstances. It must not get stuck forever in a routine – a real danger when it is in a losing situation and cannot select a move which actually improves its position, or it cannot select equipment when it has none left.
2. Its decisions must be based on the situation, but should not be entirely predictable – a random function can be brought in so that it does not always make exactly the same move in the same circumstances. This is a relatively minor point in most games.

3. The decisions it makes should be "good in the circumstances" – they need not be the best available, but they should be well above the worst.

The third point is not as simple as it seems. The decisions must give the computer a fair chance of winning, but without making victory a certainty – or the game would be impossible for the human to win. This fine balance is difficult to achieve because it is so hard to make the computer perform as well as a human at strategy games that when enough effort is made to produce a really good reply move generater, it is far better than the human.

This problem must be at least partially solved before the computer's "intelligence" can be tackled in detail.

Matching abilities

In fact, the source of the problem is the unmatched abilities of the player and computer. These must be matched as closely as possible so that in your game, the competition is fair. We shall go about this by first considering the differences between the abilities of player and computer as far as strategy games are concerned.

Computers retain and remember large quantities of information with perfect accuracy. They can evaluate mathematical functions and perform string operations with unimaginable speed. However, the only information they deal with is in the form of numbers (and strings, which are incomprehensible to the computer), and the functions used and problems solved can only be of a precise, numerical nature.

Humans can handle complex problems, but by estimation, guesswork and comparison with personal experiences. Humans can also **visualise** problems, which is useful in battle situations. But the shortcoming of the average human brain is in its untrained inability to accurately process and remember large quantities of information.

Given that estimation and guesswork are integral parts of strategy games in particular, we see that the player has a great advantage – assuming the model works and the player understands it. For this

reason, the computer needs all the help it can get. Therefore, the reply move generator should maximise the use of the computer's stronger points as against those of the human player.

The upshot of all this is that the following axioms should be applied in supplying the reply move:

1. Store information, e.g. player's previous moves as "records" as an aid in prediction;
2. Analyse possibilities in large numbers, to eliminate poor ones;
3. Use precise mathematical formulae.

The final point in ability matching is, how sophisticated does the opponent need to be? If your game has a large number of variables, random factors and subgames, or if the sides are very unevenly matched (equipment-wise) to start with, the opponent is not such a critical matter. In this case, simple mathematical formulae would suffice, perhaps with random functions added. On the other hand, if the game is in the "Chess" league, something much more complicated is required.

Types of reply move generator

Assuming that you prepared a list of decisions or possible orders for the opponent in Chapter 1, it is now possible to consider reply move generation in greater detail.

There are two approaches to "reasoned" reply moves – checking all the legal possibilities and choosing the one producing the best results, and generating moves which would improve the situation and checking to make sure the best is legal. The first method looks ahead at possibilities, using **algorithms** to produce successive parallel positions for comparison. Since the algorithms are the major feature of this method, we will refer to it as algorithmic. The other approach uses mathematical functions and simple choices between differing courses of action based on the mathematical results to give a situation nearer the target (or, the situation moving the least further away), so it is **heuristic.**

Algorithmic methods compare alternative future situations as generated by all possible moves. This has two main consequences:

one, that a phenomenal amount of memory may be needed to store a set of situations for comparison, and two, that when the game has a number of variables which can possess many values (unlike position variables in board games), it is practically impossible to go through all the individual moves. In fact, pure algorithmic methods are only really useful for simple board games (or for games which can be reduced to board games for the purposes of the reply move). Chess is a borderline case.

Heuristic methods are an exact complement to algorithmic methods. They come into their own with a large number of possible actions to take, involving many variables with a wide range of possible values.

Both methods are similar in terms of skill required to write them. But heuristics are more adaptable for very simple reply move generation. Combinations of the two are often possible. Algorithms can be made more efficient and faster by heuristics elimination of particularly bad moves at an early stage.

A variation on the heuristic theme is to use mathematical function results to select "behaviour patterns" for the computer – it finds the most appropriate course of action by analysing the situation and then makes its final decisions in accordance with the correct behaviour pattern for the situation. A reply move generator similar to this is used in "Nuclear Crisis".

Which mechanism to use

The choice of reply move generator depends also on how effective you want the opponent to be. If you happen to like writing championship chess programs, you will need all the sophisticated intricacies of advanced algorithms and shortcuts, the latest heuristic techniques, and behaviour pattern files compiled from painstaking analysis of countless grandmaster games. If, on the other hand, the strategy game you wish to create is in the more "casual interest" field – enjoyable and interesting to play, but not necessarily a fantastically developed international sport – there are much easier techniques of programming for the reply move.

Your reply move generator can range from a carefully planned, reliable and sophisticated thing using the mechanisms as already described to a much simpler hybrid of elementary heuristics with perhaps some basic behaviour patterns. In any case, the first criterion to decide on is the type of mechanism you will use – or the particular hybrid. The second criterion is how effective the mechanism needs to be (which depends on how evenly matched the two sides are), and the third criterion is how large you are prepared to make the reply move generator, remembering that when debugging your program, faults in the reply move generator are usually hard to detect. The longer it is, the more likely are faults to occur.

The following table gives the usual methods of reply move generation for the game types.

GAME TYPE	USUAL TYPE OF REPLY MOVE GENERATOR
Military (board)	Algorithmic – possibly with behaviour patterns for certain circumstances, e.g. surrender when few men left
Military (not board)	Heuristic (with behaviour patterns). Algorithmic if can be considered as or approximated to a board game, e.g. by taking every five units of the position variables as one "square"
Political	Heuristic with behaviour patterns
Economic	Heuristic (if there is an opponent)

TABLE 11 TYPE OF REPLY MOVE GENERATOR TO USE.

Non-standard reply move generation

All the methods of reply move generation considered so far have tried to imitate human actions. It is often simpler and easier to

"cheat" a little when the computer is making its move and take shortcuts with the method.

Usual shortcuts are made in the following ways:

1. Not analysing the situation properly. In "Laserfight in the OK Space Zone", the enemy fleets decide where to move by selecting two points at random and moving towards the one with fewer of your ships; or to the second if both contain the same.

2. Allocating non-existent money, equipment, etc. In "Nuclear Crisis", when at the stage of launching a single weapon, the computer chooses a type at random, checks to see if there are any left, and if there are none found for a (random) number of selections, it uses a non-existent one.

3. Totally random behaviour. The orders the computer gives are generated by random numbers, with **no** analysis of the position.

4. Fixed pattern of behaviour. In "Drake's Return", the enemy ships just head straight for your flagship. They will not collide with one another, but do not pay any attention to the positions of any other of your ships.

5. Giving the opponent an unfair advantage. Often, especially in military games, it is shorter in program length and helps the opponent if "hidden" variables are available for use. In "The Siege of Doune Castle", the opponent has access to the player's dispositions while making a "simultaneous" move, whereas the player has no information on the opponent's dispositions. This also meant that no extra variables had to be created for the "opponent's conception" – what it was "allowed" to know.

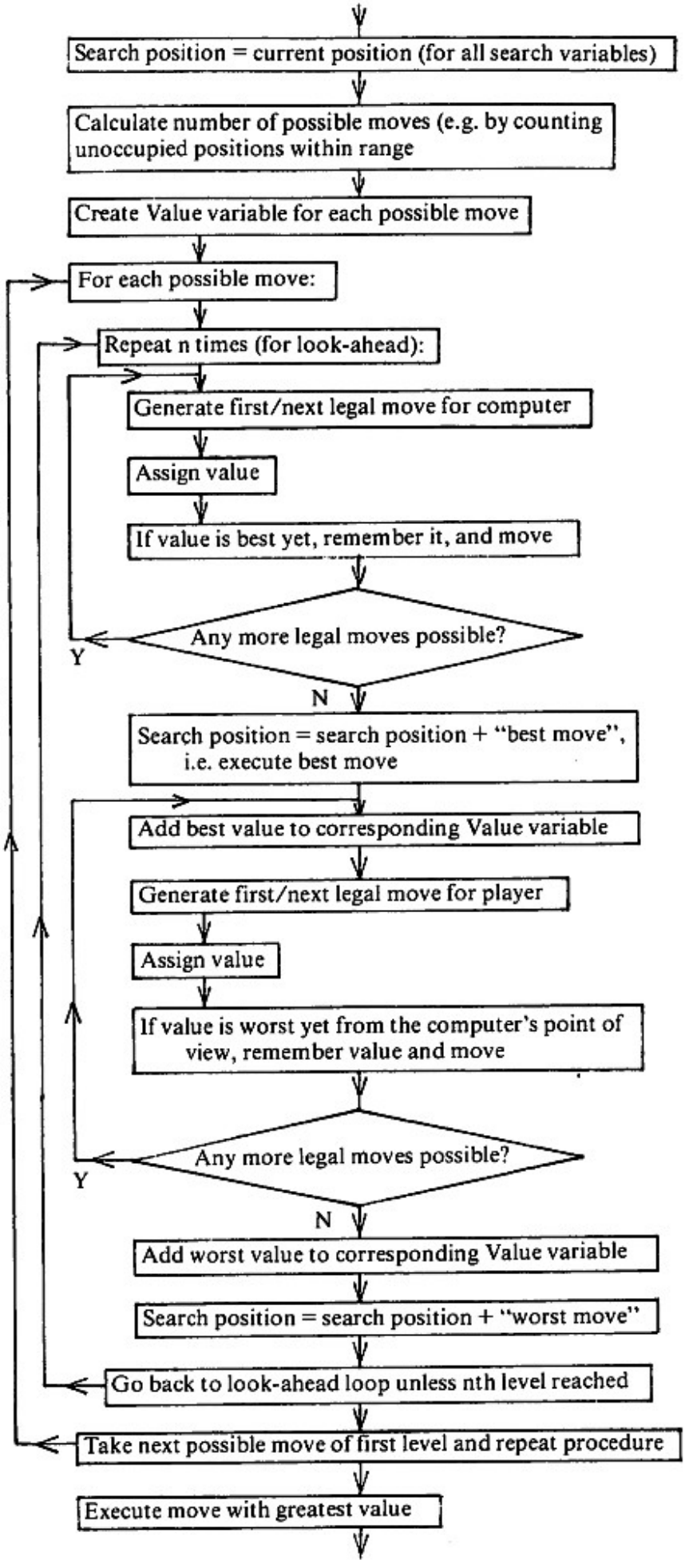
Using "cheating" shortcuts like these helps by putting the sides on more even terms for competition, and usually simplifies the program. As you see, the programs in this book make use of the shortcuts.

Note too that the programs use simple mechanisms, as complicated ones need machine-specific language to be most effective.

The mechanisms in detail

(i) **Algorithmic:** This type of reply move generator consists of two parts: the **legal move generator** and the **move selector.** The legal move generator takes any possible situation and, using simple rules, produces all possible moves one after the other, in logical, repeatable order. The move selector can work in a variety of ways. All analyse the situation produced by the latest legal move and assign a value to it. The simplest selectors wait until the legal move generator has gone through all the moves for the current situation and choose the move leading to the situation with the best value. This is practically a heuristic method. Better selectors record the position values produced by the moves from the current situation, then send all but the worst situations back to the legal move generator, producing a new series of moves, looking further ahead. This process can be repeated, say, n times, rejecting some moves at each stage, until the end of the game is found or a simple comparison of the position values is made. This is called **n-move look-ahead.** It has the disadvantage that for anything other than small values of n, the memory required to hold the position values of the situations at the current level of look-ahead can be phenomenal. The method is further complicated by the need to predict the player's moves, which may well be less rational than the computer's. The way round this in terms of safety is to consider **all** the player's possible moves. As this would consume too much memory for most purposes, a practical solution is to eliminate the worst moves as before, remembering that the scoring system must be **reversed** – the worst positions for the player, not the computer, must be eliminated.

The generalised flowchart for an algorithmic reply move generator looks like this:


Search position = current position (for all search variables)
Calculate number of possible moves (e.g. by counting unoccupied positions within range
Create Value variable for each possible move
For each possible move:
Repeat n times (for look-ahead):
Generate first/next legal move for computer
Assign value
If value is best yet, remember it, and move
Any more legal moves possible?
Y
N
Search position = search position + "best move", i.e. execute best move
Add best value to corresponding Value variable
Generate first/next legal move for player
Assign value
If value is worst yet from the computer's point of view, remember value and move
Any more legal moves possible?
Y
N
Add worst value to corresponding Value variable
Search position = search position + "worst move"
Go back to look-ahead loop unless nth level reached
Take next possible move of first level and repeat procedure
Execute move with greatest value

The major points to decide on are as follows:

1. How far to look ahead? This has a direct bearing on the memory consumed and the search time, since for each move, there will, on average, be a certain number of possible moves, and the total number investigated will be this average, the "branching factor", raised to the power of the look-ahead number.
2. Does the generator operate any differently in producing the player's forecast move? In the flowchart shown above, the player's move is chosen by finding the best according to the analysis of the resulting position. This is only an approximation, since it takes no direct account of the player looking ahead.
3. How is the value of each situation assigned? In military games, the numbers of each side's pieces and their positional advantages are usually analysed. This can easily become unbelievably complicated where formation, ranges and protection are concerned, so to save excessive brain-strain, simple evaluation functions are recommended, at least at first.
4. Is the generator purely algorithmic, or does it incorporate heuristics, etc? Any such refinements either filter out moves at the legal move stage, by making undesirable moves illegal, or alter evaluation functions.

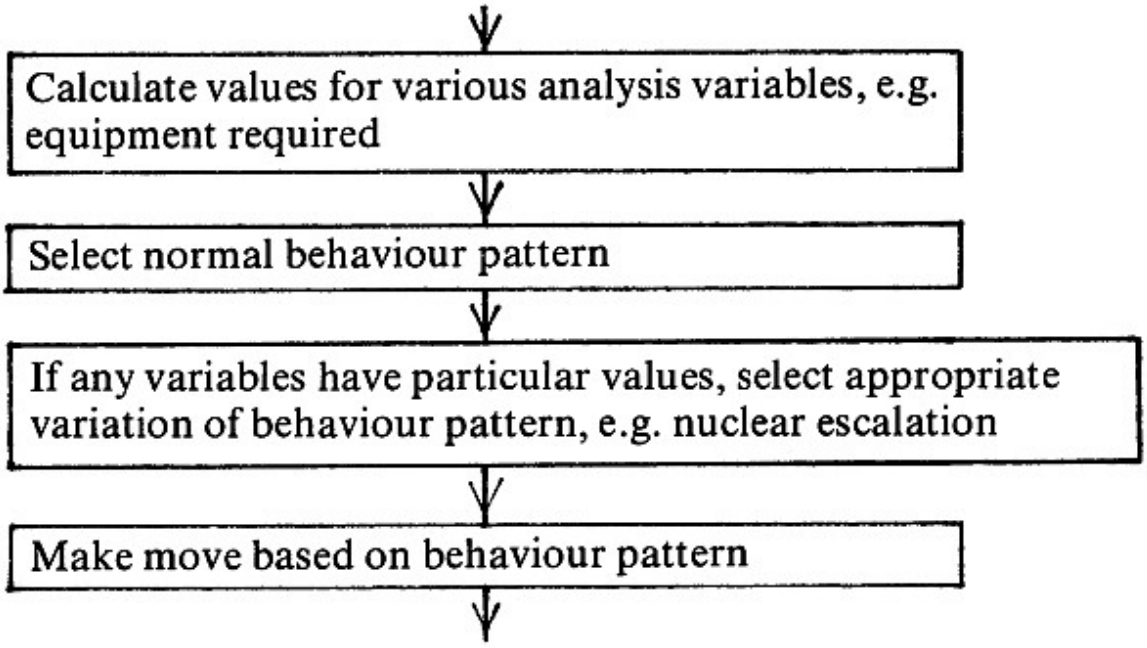
(ii) **Heuristic** This type of reply move generator uses an agglomeration of mathematical lines and string functions, with a few Choice lines, to produce an output in the required variables by carrying out operations on the current position variables. The equations point to a situation nearer the target situation. In a way, **any** reply move generator (except wholly random ones) can be regarded as heuristic, but the epithet best fits the mathematical types described here.

It is extremely difficult to produce a heuristic generator for games with discrete moves, like board games, and games with very specific rules (chess falls in both these categories), but for the less rule-infested military games, and for most political and economic games, heuristics are quite practicable. The telling part in making the generator is deciding how the general response you want in all possible circumstances can be related to the situation variables and

a handful of mathematical lines. When very elementary responses are required, as in "Drake's Return", the matter is simple. The computer takes each ship in turn and calculates the nearest it can move towards your flagship. It then checks to see that it is not going to collide with any of its own ships. Simple applications like this are reasonably acceptable for military programs, though of course they can be greatly improved by adding a few more possible "behaviour patterns" – such as taking evasive action when outnumbered.

In this case, the behaviour pattern effectively selects the target, which is then aimed for by the heuristic generator. As with most things in life, this is not a perfect description, and the distinctions between the types of generator are blurred anyway.

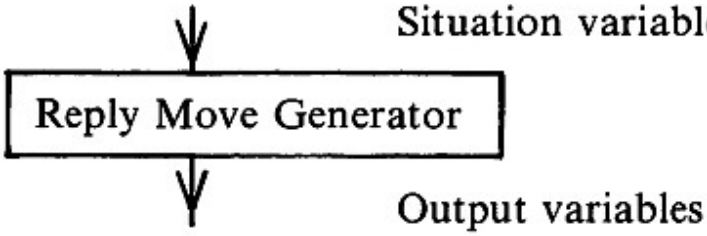
The following flowchart generalises heuristic reply move generators:



This flowchart also shows the part played by behaviour patterns. An example of this type of generation is in "Nuclear Crisis", where simple heuristics are used to buy new equipment, and behaviour patterns are used to decide on the general diplomatic line taken.

Integrating the reply move generator

The reply move generator, whatever its mechanism, may be represented by the following diagram:



When integrating the reply move generator, you must check that:

1. It produces a reply move in all circumstances.
2. None of the equations in it can produce an error, e.g. by dividing by, or taking the log of, zero, or by taking roots of negative numbers.
3. The output variables are of the appropriate types (integer/real and correct range).

If the game is written using the methods in this book, it will be very modular. Different reply move generators could be substituted for experimental purposes, and the above advice might well come in useful.

Finally, please note that not all the intricacies and refinements possible in reply move generation are covered here – not by any means. This chapter is intended only as an introduction to the subject, to explain some of the factors involved. In fact, the games in this book use only the simplest forms of reply move generation themselves. For more information, you should refer to a more specialised book, such as "Computer Gamesmanship" by David Levy (Century Publishing Company Ltd.).

CHAPTER 5
INPUT/OUTPUT

The object of the input and output sections is of course to transfer information between the computer and the player. To do this most effectively, the output should be concise and easy to understand. If the game is just for your own amusement, you do not have to lay the display out carefully and identify all the variables unmistakeably. But if you want other people to see it, the display should be well laid-out and comprehensible. Input should be logical and obvious – the computer supplies clear instructions for what is required. Illegal moves must be rejected – the inputs repeated, or default values automatically chosen.

All the above applies to any computer program. As far as strategy games in particular are concerned, the input/output sections have also to create the **atmosphere** for the game – to bring it to life. In the games in this book, this is accomplished by careful wording of output and instructions. However, it is often even more effective if graphics displays and sound are used too, especially in military programs. These are unfortunately beyond the scope of this book, since the relevant commands vary greatly from machine to machine. But the same rules apply (to graphics especially) and it is a straightforward task to take the I/O sections of any program in the book and add graphics displays and sound.

Output

Any output section can be broken down into **translation** and **display.** In translation, the relevant information in the computer is used to access different sentences or phrases, for example, which will then be displayed. Of course, some variables are displayed just as they are. In display, the information is organised on the screen, to look neat and orderly.

Computers can store data as strings of characters and as numbers. In simpler displays, these can be output as they are. But when more realism is required, some variables – such as people's opinions and damage reports – are better not displayed numerically. Instead, they should be translated by using the value of the variable to access different strings – phrases and sentences – to produce a "wordy" report.

The usual candidate variables for this type of display are: opinions, conditions, rates (e.g. of work), strengths of forces, and any factor of an "on/off" nature – for instance industries either nationalised or privatised, represented by a variable with possible values of Ø or 1.

Particularly in games with a lot of numerical variables, the layout of the display is crucial. The general format for the main display is often as follows:

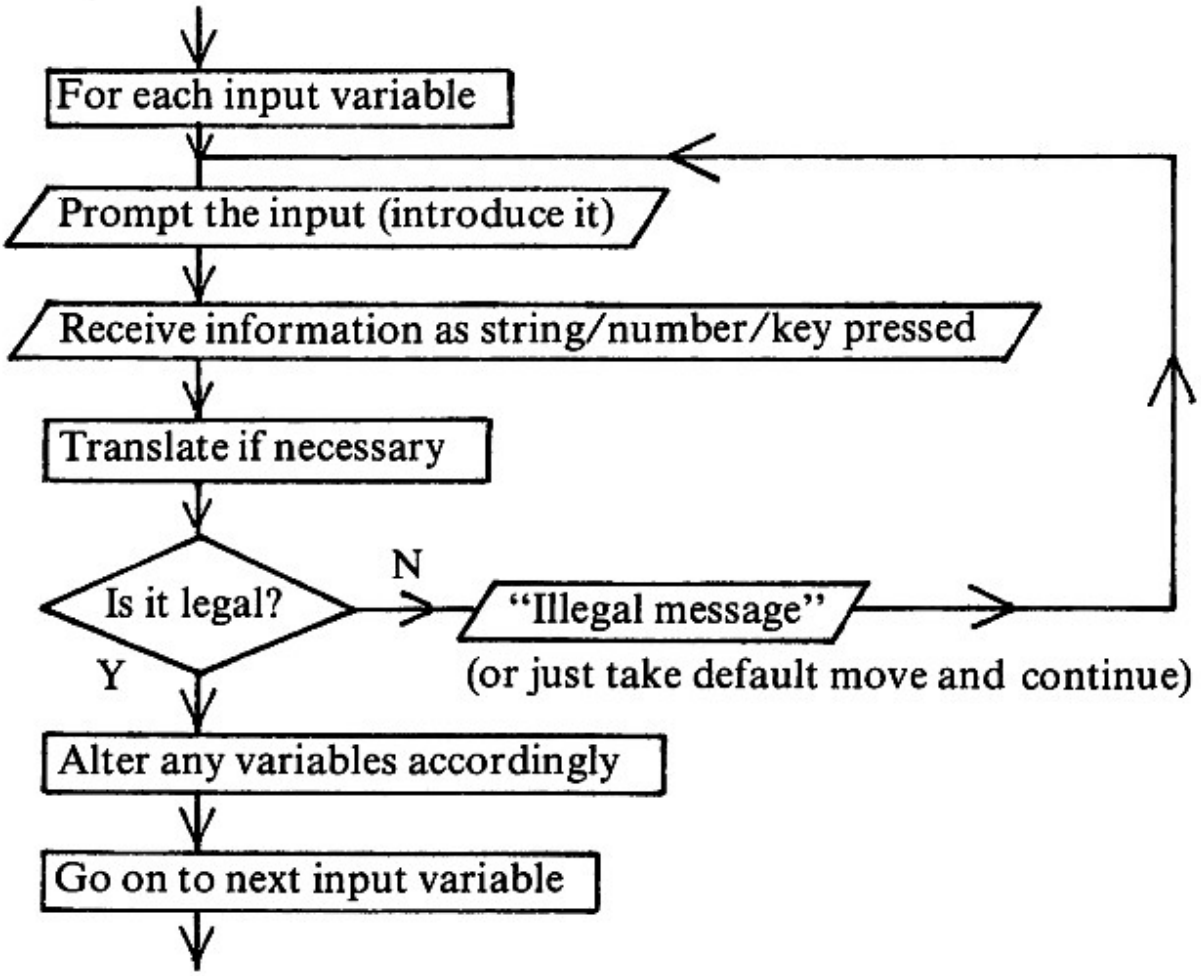
Value of loop variable Title
Information in order of importance, in groups separated by blank lines (graphics display if applicable)
Space for input instructions

This pattern is achieved either by printing the lines in order, top to bottom, or by printing to each screen line directly, using the screen addressing commands discussed in the next chapter. The second way is useful when the information must be updated during the input process, to send new figures to the correct positions and to delete selected lines, such as old input instructions, making way for the next set.

The output information should now be added to the game flowchart. Make sure that the flowchart takes all possible outputs into account, displaying the correct number of significant figures, giving units and adding full stops to the end of sentences.

Input

The input section can be broken down into the following flowchart:

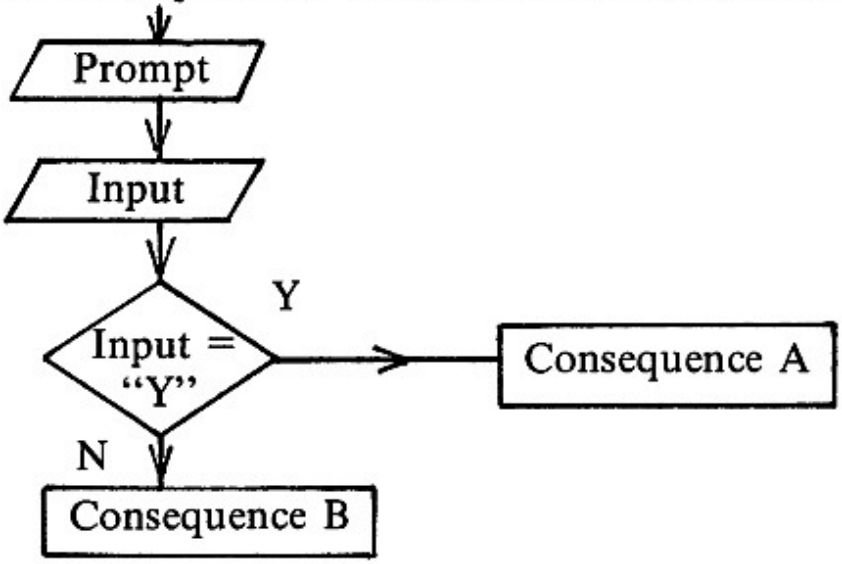


The input prompting is where the computer asks for the information. The wording should fit in with the atmosphere, but as for all computer wording, it should not be too incongruous with the keyboard and VDU. For instance, the prompt "Arrr captain, therr be Zpanishe shippes on t'horizon. What be we to do?" is more ridiculous than sublime.

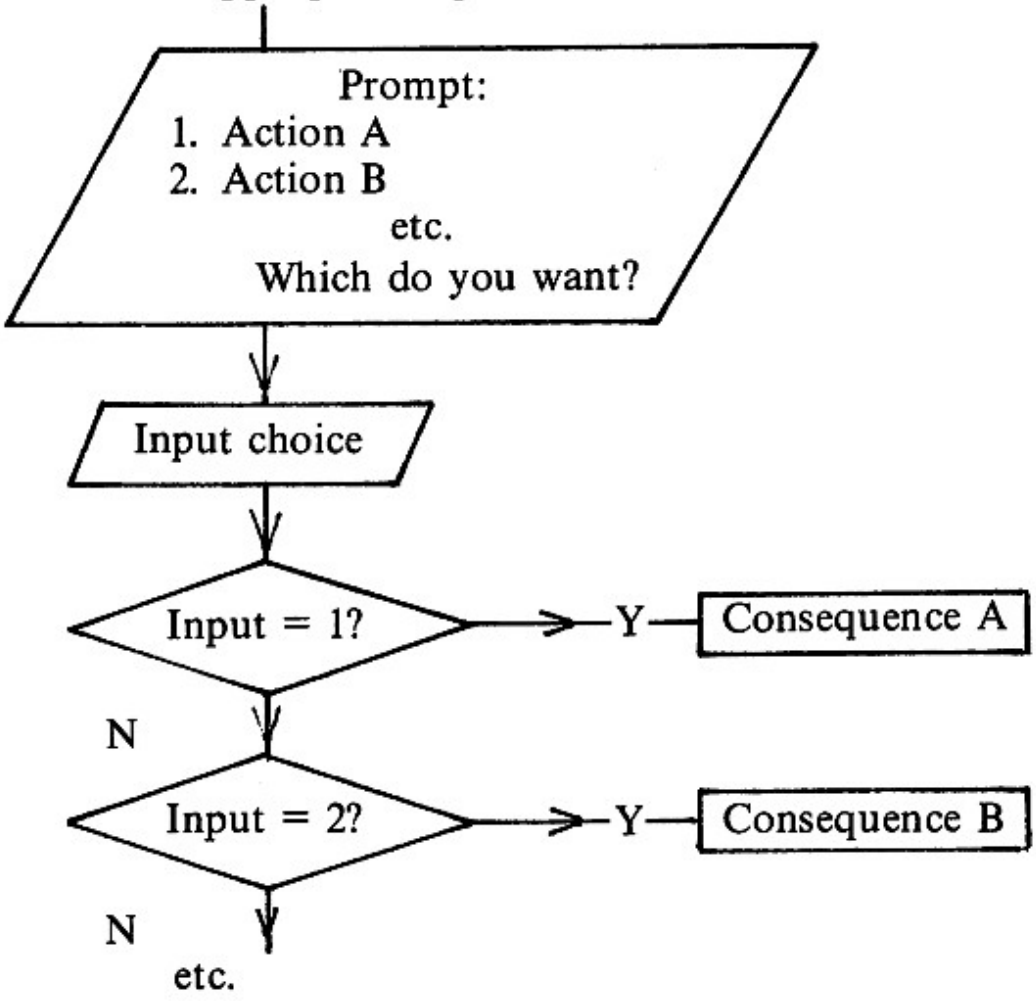
If several inputs are required on the basis of a display of information, it may be best to display the prompts at the same place on the screen each time, as in "The Siege of Doune Castle". To do this, you must remember to erase the previous prompt and answer, and to address each prompt to the correct position.

Computers receive information in three general ways in this sort of game – as a number, as a string, and as a single key pressed. Strategy games require information either in a Yes/No form, as one or more choices from several options for courses of action, or as a quantity – like tax.

Yes/No questions are best structured like this:



Choices can be structured in two basic ways. The simpler, used in the programs in this book, is to number the options and have the appropriate option number input:



The other way to engineer choice inputs is by the use of an input "language", as in "Adventure" games. This method is, however,

impossible to produce in a sophisticated form without having to lapse into the dialects of particular computers.

Numerical quantities are of course input as numbers. It is wise not to require that units be added to the input the player makes – "£" for instance. This should be taken as read.

Yes/No questions and multiple options are usually used to branch the game on to a new path for a while – a general election, say. As such, they are used to access routines, or particular places in the program. Numerical inputs, on the other hand, are used to provide the quantities for use in equations, and do not affect the execution of the program **directly.**

An essential part of Input is the checking stage. Inputs can be fouled up in a considerable number of ways, sometimes maliciously, often accidentally. When an illegal input is made, it can either be rejected and the input process repeated from the prompting stage, or the input can be set at a default value. The following tables show what can go wrong, and how to deal with it.

YES/NO INPUTS (AND ONE CHARACTER OR WORD)

Illegal entry	Check – remedy
RETURN only	Reject/take as "No"
Unknown word/letter	Reject
Starts/ends with spaces	Suppress spaces/reject
Contains spaces	Reject

OPTIONS INPUTS BY NUMBER

Illegal entry	Check – remedy
Negative number	Take absolute value
Fractional number	Take integral/rounded value
Out of range	Take number at nearest end of range

OPTION INPUTS BY LANGUAGE

Illegal entry	Check – remedy
RETURN only	Reject/take to be "No action"
Unknown word	Query word only – retain rest of sentence.
Ungrammatical sentence	Reject
Extra spaces between words/at ends	Suppress spaces if possible

NUMERICAL INPUTS

Illegal entry	Check – remedy
RETURN only	Reject/take as zero
Negative where should be positive	Take absolute value
Fractional where should be whole	Take integral value
Out of range	Reject/adjust to nearest end of range
Strings input	Reject/take as zero

These tables summarise everything that can go wrong in an input. Not all the precautions listed are necessary – checking for strings in numerical inputs can only be done indirectly, and is also an unlikely error. However, out of range errors are more common. Make sure that all **likely** errors are taken into account in your planning. A balance has to be struck between rejecting **all** slightly ambiguous or wrong inputs, and taking default values left, right and centre. The latter may mean the player spends less game-time correcting errors, but if the limits possible in the inputs are unknown, the player may be in the dark as to what is going on. A possible compromise would be for the computer to display the value eventually taken, which the player can compare with the value input.

The input section can now be added to the program flowchart.

Screen management

At this stage, there are still a very few final things to do to the flowchart before it can be translated for the computer. These are to do with the clearing of the screen, prompting for continuation, and pauses.

In general, the screen should be cleared after each set of information has been displayed, giving the player time to see it and, of course, to make appropriate inputs. Additional screen clearing may be required if there is too much information to fit on to the screen in one go. In this final case, and in others if desired, continuation prompting can be added. This is of the form of an instruction – "PRESS ANY KEY TO CONTINUE" – and a loop which waits until a key is pressed, then clears the screen and continues with the program. The same effect can be achieved with a timed pause – either using any "PAUSE" or "WAIT" command the computer may have, the computer's internal timer (again only if it has one), or a FOR-NEXT loop which takes a sufficient time to execute.

Care should be taken that in optional displays – such as battles, which are often not used – the continuation routine or pause routine is also missed out, as it can cause unnecessary and irritating delays.

CHAPTER 6
TRANSLATING THE FLOWCHART

So far, we have made up an operational plan of the strategy game in full detail. A human being reading it through should be able to understand perfectly how the game works. However, it is totally incomprehensible to computers – at least of the current generation. In this chapter, the flowchart is finally translated into something the computer can understand.

Up until this chapter, everything that has been said applies equally well to any conceivable computer. It therefore holds for future generations of computers just as well as for the Third and Fourth generations. Yes, 8, 16, 32 bit and more, artificially intelligent Fifth generation computers, all can be used to play strategy games, in **any** computer language (including machine code), with appropriate translation. But since by far the most widely used language is BASIC, we shall restrict ourselves to this one language, and a subset of it at that.

Computer languages

High level computer languages like Forth, Pascal and BASIC are designed to make the program comprehensible to the computer while changing it as little as possible from the original version. One thing which is sacrificed is the structural look of the flowchart – with shaped boxes and interconnecting lines. Instead, instructions are expressed in separate, single ordered lines, numbered in most languages, for cross-referencing.

A "universal" BASIC

Specifying BASIC as the language used is not enough at the present time. Every type of computer has its own dialect of BASIC. Our object at this point is to select a suitable subset, present in all BASIC-using computers, so that the programs in this book could be used with little or no modification by any owner of a personal computer.

In researching the most common BASICs, it quickly became clear that although the vast majority of important BASIC words are the same on all the computers, a few remain which are either slightly different, or even totally absent. Where differences were found, the appropriate alternatives have been given, and where commands are absent, equivalent routines have been given where possible.

Subroutines

BASIC has one structure in particular not directly illustrated by the game flowchart: that of subroutines. Their principal use is to enable a particular set of instructions to be executed at several stages in the program without having to repeat the sequence each time. They are also used in conditional lines where a sequence of instructions will not fit in the conditional line itself.

In the following sections, these abbreviations are used:

n – number
v – numeric variable
v$ – string variable
V – numeric or string variable
S – string

Structure

The general structural parts of the flowchart are: repetition of a routine, branching if a condition is satisfied, going to a particular part of the flowchart, calling a subroutine and stopping. In addition, the program may be re-run from the start. The following

table shows the BASIC translations for the various structural options:

Flowchart Structure	BASIC equivalent
Branch to a section (at line n)	GOTO n
Call a subroutine (at line n)	GOSUB n
Return from the subroutine	RETURN
Repeat a procedure n times	FOR v = 1 TO n at start of procedure NEXT v at end of procedure
Stop – end of program	END or STOP
Execute program	RUN
Branch if condition satisfied (diamond-shaped box)	IF (condition) THEN

Note that IF-THEN **GOTO** or **GOSUB** need only be used when the consequence is more than a couple of instructions. Otherwise, the instructions can be continued within the IF-THEN line, especially if the computer uses multi-statement lines (see later). Example: IF DES>O THEN PRINT DES; "men deserted.".

Data management

These commands are concerned with the storage and transfer of information within the computer. As you can see, they presented more of a problem than the structural commands. While most computers have READ-DATA-RESTORE, it is noticeably absent on the Sinclair ZX81. A moderately simple routine can be used as a replacement, as the printout shows.

Flowchart command	BASIC equivalent
Create n numbered variables	DIM V(n) *
Assign value n or S to V	LET V = n or S ("LET" often optional)
Store of data: series of values	DATA n or S, n or S, . . .
Go back to start of data (at line n)	RESTORE n
Assign the next data value to V	READ V

* Note that on some computers, such as the ZX81, DIM creates n variables, numbered 1 to n. On other computers, such as the BBC Micro, it creates n+1 variables, numbered Ø to n. Since the first version is contained in the second, the first is used in the programs.

SIMPLE ZX81 READ/DATA/RESTORE ROUTINE:

DATA info, info, info	:	LET D$ = "info, info, info,"
RESTORE	:	LET DC = Ø
READ	:	see subroutine, i.e. GOSUB 95ØØ

```
9500 LET I$=""
9510 FOR X=DC+1 TO LEN D$
9520 LET DC=X
9530 IF D$(X)="," THEN RETURN
9540 LET I$=I$+D$(X)
9550 NEXT X
9560 RETURN
```

This works with any length of a piece of data, as long as every piece ends in a comma and all are contained in the one line. As shown, it reads to a string variable, I$. If a numeric is required, add a line:

LET I = VAL I$

which reads the data into I on returning to the main routine.

Input/Output

For putting information into the computer, INPUT V was used throughout as a standard.

The simplest display commands are virtually identical on all machines. The major exception is in addressing to a part of the screen. The convention adopted in the book is BBC BASIC's TAB(x,y) where the origin is at the top left of the screen, and x represents the number of characters along, y the number of lines down (starting at TAB (Ø,Ø)). The corresponding Sinclair system is AT y,x, i.e. the order of the numbers is reversed. Dialect translation between the two is simple. However, in the simple Commodores VIC 2Ø and 64 (without extended BASICs), complicated "POKE"

Random numbers

An essential function is the random number generator. The convention used in the programs is RND(1) which generates a random number between Ø and 1 **inclusive.** Any corresponding function replacing it will work perfectly. Watch out that the function used can produce both Ø and 1. Some, such as the RND function in the ZX81, produce Ø to Ø.999999999, which is sometimes not quite good enough.

Multi-statement lines

Multi-statement lines are used in all the programs in the book, as almost all computers can take them. If they occur anywhere other than in an IF-THEN line, they can be replaced by consecutive lines:

e.g. 1Ø LET A = 1:LET D = 5Ø → 1Ø LET A = 1
15 LET D = 5Ø

If the multi-statement part does occur at the end of an IF-THEN line, it must be replaced by a subroutine

e.g. 9Ø IF A$ = "Y" THEN
CLS:PRINT "Goodbye":PAUSE 1ØØ:CLS
→ 9Ø IF A$ = "Y"
THEN GOSUB 1ØØØ
1ØØØ CLS
1Ø1Ø PRINT "Goodbye"
1Ø2Ø PAUSE 1ØØ
1Ø3Ø CLS
1Ø4Ø RETURN

Cases of letters

In printed material enclosed within inverted commas, various characters and letters in upper and lower case are used. If your computer lacks any symbol, improvise. Small letters look nicer, but capitals display the information just as well.

REM

A final command to consider is REM, enabling remarks to be inserted in the program. It is functionally useless, and wastes memory and typing time. However, it enables explanatory text to be included, which is useful when checking and writing programs for other people.

Preparing the flowchart

Before translation, some final alterations may need to be made to the game flowchart.

Firstly, it should be in linear form, with all the branches and outlying routines reached by arrows, weaving in and out of each other if necessary.

Secondly, major subroutines should be identified at this stage. The first type to look for is a repeated procedure – like several damage reports. Depending on its length and the number of times used, it may or may not be worthwhile making it into a separate subroutine. The second type may not be evident until translation is under way. It involves IF-THEN questions followed by lengthy consequences. You should decide as soon as possible whether these need to be put into subroutines.

Having done all this, the flowchart is ready to be translated.

Translation

The object of translating the flowchart is, of course, to make it comprehensible to the computer. But the matter should not rest there – translation should produce an economical, well-structured program, so that its workings are clearly understood for improvements later.

As far as structuring goes, I am a great believer in so-called "structured BASIC". This uses commands such as PROC to access what are effectively named subroutines. The ideal program layout

from this is to have the main routine simple and uncluttered, accessing its procedures as subroutines, as shown below:

```
 10 REM MASTER ROUTINE
 20 GOSUB 1000:REM INITIALISE
 30 CLS
 40 GOSUB 1500:REM INTRO
 50 GOSUB 9000
 60 LET H=1
 70 CLS
 80 GOSUB 2000:REM DISPLAY A
 90 GOSUB 2500:REM INPUTS
100 GOSUB 3000:REM ENEMY ORDERS
110 CLS
120 GOSUB 3500:REM COLLISIONS
130 GOSUB 4000:REM FIGHTING
140 GOSUB 4500:REM DAMAGE REPORTS
150 GOSUB 5000:REM SINKINGS
160 GOSUB 5500:REM WIN/LOSS
170 GOSUB 9000
180 LET H=H+1
190 GOTO 70
```

There is nothing wrong with having all the routines stuck together in a long chain, but structured BASIC is easier to plan, and clearer – at least, that is what I find.

Furthermore, I like to use a highly modular system in programming, where major routines are more or less self-contained. This makes debugging easier, and means that different routines can be substituted relatively easily – new ways of producing the reply move, say.

Simple "universal" routines are grouped together at the end of the program. Some examples of these routines are given at the end of the chapter.

Through experience, the best way to set about translation seems to be to go through the flowchart, giving major routines round line numbers – 1ØØØ, 15ØØ, 2ØØØ, etc. Then, write the master routine to link these together. The major routines can be tackled individually at will. I always use an interval of 1Ø between line numbers in a

routine. This is neat, and leaves enough space to add forgotten lines in between.

When writing the program proper, it is a good idea to make a paper copy at the same time as typing the program in, if not before. When, as I have, you have left the program sitting during dinner and come back to find it crashed, or you have typed DELETE 4Ø,5ØØØ instead of DELETE 4ØØØ,5ØØØ, you really learn the meaning of the word "frustration". Saving parts of a program as you go along is also a good idea – this means you avoid typing and lost information in one go.

Translation itself is done using the appropriate commands, as shown in the tables, for the programs in this book. There are some cardinal rules of translation:

1. Be sure that the ending is accessed by a GOTO routine and not a GOSUB.
2. Be sure that each GOSUB has a RETURN.
3. Be sure that FORs always lead to a NEXT – only use GOSUBs to access routines from inside such repeating procedures, or GOTOs leading back inside.
4. Be sure that each routine ends correctly – with a GOTO or a RETURN.

Conventions used

The naming of variables is a major part of translation. Where possible, use single letters, preferably standing for the variable (e.g. Fuel – F). Where an array of numbered variables is used, remember to keep a note of the order of variables in the array – it is all too easy to forget, particularly with a large number. In some cases, such as in "Corridors of Power", the variables are best given longer, more explanatory names, for clarity.

It is useful to use the following convention for control and data transfer variables:

X, Y and Z for repeating a procedure – FOR X = 1 TO 5, etc.
I/I$ for information from DATA, etc, and J/J$ and K/K$ if more are needed.

A/A$ for answers from the player (i.e. INPUTS), and B/B$ and C/C$ if more are needed simultaneously.

Be very careful that where the same control variable names are used in different routines, they cannot come into contact. When writing "Nuclear Crisis", I used X as a repeat variable in the player's escalation section. This section accesses the battle section as a subroutine. Originally, both X and Y were used as control variables here. While testing, I noticed that the program always misbehaved strangely here, and it took a rather long and mentally-disturbing time to track down the problem and correct it by altering one control variable.

Once the program has been written, record it at once. Resist the urge to play it, particularly if long, because if it crashes, you are bound to have to type a good bit back in.

Then – debug it!

UNIVERSAL ROUTINES

```
8000 REM INPUT QUANTITY CHECK
8010 INPUT A
8020 LET A=ABS(INT(A)):REM FOR POSITIVE WHOLE NUMBER
8030 IF A<lowerlimit THEN LET A=lowerlimit
8040 IF A>upperlimit THEN LET A=upperlimit
8050 RETURN
```

```
8500 REM INSUFFICIENT QUANTITY REPORT
8510 INPUT A
8520 LET quantity=quantity-A
8530 IF quantity>=0 THEN RETURN
8540 PRINT"YOU DO NOT HAVE ENOUGH equipment."
8550 LET quantity=quantity+A
8560 GOSUB key press routine
8570 GOTO 8510
```

```
>
 9000 REM REPLAY ROUTINE
 9010 PRINT
 9020 PRINT"DO YOU WANT ANOTHER GAME ?"
 9030 INPUT A$
 9040 IF A$="Y" THEN RUN
 9050 END

>
 9900 REM KEY PRESS ROUTINE
 9910 PRINT
 9920 PRINT"PRESS ANY KEY TO CONTINUE"
 9930 REPEAT UNTIL GET$<>""
 9940 CLS
 9950 RETURN
```

CHAPTER 7
DEBUGGING AND TROUBLESHOOTING

So, you have finished the "first draft" of your program. It is all typed in and ready to use. If you are experienced, you will be chattering your teeth, for you know that – **no program is perfect!**

Even the most carefully designed and typed-in programs can have faults. They may be caused by tiredness; they may be caused by miscalculation. If – when – you start to find them, do not commit suicide. This chapter is about detecting, locating and curing them.

Types of fault

There are four general types of bug a program can have:

1. Computer detected mistake – obvious.
2. Program not working right – structural error.
3. Display not working right – design wrong.
4. Game not working right – model wrong.

Typical computer-detected mistakes are arithmetic overflows (e.g. from division by zero), missed-out characters and mis-spelt commands.

Structural errors are caused by routines accessed wrongly, or not at all, and control variables interfering, e.g. by "nesting" FOR-NEXT loops with the same variable.

Display errors are usually insufficient or too many screen clearances, not pausing to enable the display to be read, faulty

screen addressing (e.g. getting the x and y co-ordinates the wrong way round) and run-on lines. They are relatively easy to track down, being displayed.

Model errors are difficult to pin down, as the program runs as it should. The mistakes are invariably in the mathematical lines.

Often, the exact type of mistake cannot be found at this stage. A program typing error – say, a RETURN missed out at the end of a subroutine – could cause model-fault type symptoms, and strange values for the variables.

Causes of faults

Faults are caused by two sorts of root mistake – Composition Error and Typing Error.

If you made a hard copy of the program, typing errors can be identified by the laborious, mindless but simple process of comparing the listing in memory with all relevant parts of the hard copy. Typing errors are usually missed-out characters, mis-spelt words and mis-copied numbers.

Composition Errors are more insidious – more difficult to detect and cure. They are often made when the game was still in flowchart form, or when basic ideas for translation were made. As such, they usually occur in mathematical lines, where you did not think carefully enough while planning. At their worst, a whole routine may need to be re-written. At best, a single mathematical operation changed. To cure them, you really only need to check mathematical lines. But, these must be studied with the utmost care to see how they behave in the circumstances. It may be necessary to trace the values of one or more variables throughout the game. In summary, this type of error can be very frustrating, which is why great care should be taken with the planning of the program.

When a fault occurs . . .

1. Take a deep breath. Do NOT (a) smash the keyboard or (b) jump out of a window. Have a break for a few minutes if

possible. Remember – most faults are caused by tiredness.

2. Decide what type of fault has occurred (for **each** mistake, of course).
3. Find the relevant part(s) of the listing and display it (them).
4. Inspect the listing to see if a fault is obvious.
5. Compare with the hard copy if one was made.
6. Check the mathematical parts to make sure it is fundamentally correct.

If, after all this, you have failed to find the fault, have a break, then try again, and again, and again. Only when you are 100% sure there is no apparent fault should you consider destroying the computer, suicide, or even leaving it till tomorrow!

INTRODUCTION TO THE PROGRAMS

The programs in the remainder of this book are all strategy programs as defined in Chapter 1. Some deal with things military, some with political confrontations, and some with the economic management of a country. Most are not purely of one type – they are hybrids to some extent.

The BASIC they are written in was developed primarily on a BBC Micro. It will, however, run on any BASIC-using computer with little or no adaptation. Multi-statement lines are used sparsely, and features of the currently-evolving extended BASICs are omitted altogether.

That being said, the following explanations may be necessary for owners of some computers:

TAB(x,y)	Move print position to x along, y down relative to origin at top left.
REPEAT UNTIL GET$<>""	Wait until a key is pressed.
RND(1)	Random number Ø to 1 **inclusive.**
FOR Q = 1 TO 6ØØØ:NEXT (and other variations on Q and 6ØØØ)	Pause – for about three seconds.

Since on almost all computers, the use of "LET" in assigning values to variables is optional, it has been omitted in some of the programs. If your computer requires the LETs, adding them makes no difference whatsoever to the running of the programs.

The games run on any screen of 25 x 4Ø character lines easily, and usually in much less space. If even this is too large for your computer, either abbreviate or relocate information to the screen, make use of automatic scrolling, or add screen clearing routines like the KEY PRESS ROUTINE given at the end of Chapter 6.

The programs are in modular form, so routines such as reply generation can be altered easily for experimenting.

So, plug in the old computer and go to it!

Drake's Return

This game represents a naval battle in Elizabethan times, the object being to sink the enemy fleet by gunfire or ramming before your flagship is sunk.

The game is carried out on a "board", generated a square at a time. Reports are given of encounters after your moves are made.

The reply move generator is very simple – the enemy ships head straight for your flagship!

The variables containing the ships are two arrays, holding each ship's x and y position and condition. Another array is used to set up the display, so that direct screen addressing is not used. The ships's positions are read into this, and it is then printed bit by bit.

The reply move generator is easily improved if you want to experiment. Another possible field for experimentation is the model itself, as regards damage due to fighting. There is no random function as the game stands, and you may like to try adding one to make winning easier/impossible.

```
 10 REM MASTER ROUTINE
 20 GOSUB 1000:REM INITIALISE
 30 CLS
 40 GOSUB 1500:REM INTRO
 50 GOSUB 9000
 60 LET H=1
 70 CLS
 80 GOSUB 2000:REM DISPLAY A
 90 GOSUB 2500:REM INPUTS
100 GOSUB 3000:REM ENEMY ORDERS
110 CLS
120 GOSUB 3500:REM COLLISIONS
130 GOSUB 4000:REM FIGHTING
140 GOSUB 4500:REM DAMAGE REPORTS
150 GOSUB 5000:REM SINKINGS
160 GOSUB 5500:REM WIN/LOSS
170 GOSUB 9000
```

```
 180 LET H=H+1
 190 GOTO 70
 997 :
 998 :
 999 REM INITIALISATION
1000 DIM Y(5,3)
1010 DIM E(5,3)
1020 DIM D(100)
1030 LET P1=1
1040 LET P2=100
1050 FOR X=1 TO 5
1060 LET Y(X,1)=1
1070 LET E(X,1)=1
1080 LET DF1=RND(5)+1
1090 LET DF2=RND(5)+1
1100 LET P1=P1+DF1
1110 LET P2=P2-DF2
1120 LET E(X,2)=INT(P1/10)
1130 LET Y(X,2)=INT(P2/10)
1140 LET E(X,3)=P1-E(X,2)*10
1150 LET Y(X,3)=P2-Y(X,2)*10
1160 NEXT X
1170 RETURN
1497 :
1498 :
1499 REM INTRO
1500 PRINT"DRAKE'S RETURN"
1510 PRINT
1520 PRINT"YOU ARE THE COMMANDER OF A P
ATROL OF"
1530 PRINT"5 SHIPS OF THE LINE IN THE R
EIGN OF"
1540 PRINT"ELIZABETH I. YOU DIRECT THE
MOVEMENTS"
1550 PRINT"OF YOUR SHIPS TO OVERCOME A
SPANISH"
1560 PRINT"PATROL. THE SHIPS FIRE THEIR
"
1570 PRINT"BROADSIDES WHEN IN REASONABL
E RANGE."
1580 PRINT
```

```
 1590 PRINT"YOUR SHIPS ARE NUMBERED 1-5
ON THE"
 1600 PRINT"SIMPLE DISPLAY, ENEMY SHIPS
ARE"
 1610 PRINT"MARKED 'E'"
 1620 PRINT"BE CAREFUL NOT TO LOSE YOUR
FLAGSHIP,"
 1630 PRINT"SHIP 1..."
 1640 PRINT
 1650 PRINT"WHEN YOU GIVE YOUR ORDERS, U
SE THE"
 1660 PRINT"NORMAL ABBREVIATIONS.(N,S,NE
 ETC)"
 1670 RETURN
 1997 :
 1998 :
 1999 REM DISPLAY A
 2000 PRINT"DISPOSITIONS:"
 2010 FOR X=1 TO 100
 2020 LET D(X)=0
 2030 NEXT X
 2040 FOR X=1 TO 5
 2050 IF Y(X,1)<0 THEN GOTO 2070
 2060 LET D(Y(X,2)*10+Y(X,3)+1)=X
 2070 NEXT X
 2080 FOR X=1 TO 5
 2090 IF E(X,1)<0 THEN GOTO 2110
 2100 LET D(E(X,2)*10+E(X,3)+1)=-1
 2110 NEXT X
 2120 FOR X=1 TO 100
 2130 IF D(X)=0 THEN PRINT".";
 2140 IF D(X)>0 THEN PRINT STR$(D(X));
 2150 IF D(X)=-1 THEN PRINT"E";
 2160 IF X/10=INT(X/10) THEN PRINT
 2170 NEXT X
 2180 PRINT
 2190 RETURN
 2497 :
 2498 :
 2499 REM INPUTS
 2500 FOR X=1 TO 5
```

```
 2510 IF Y(X,1)<0 THEN GOTO 2640
 2520 PRINT"WHAT ARE YOUR ORDERS FOR SHI
P ";X;"?"
 2530 INPUT A$
 2540 LET S=Y(X,2)
 2550 LET T=Y(X,3)
 2560 IF A$="N" AND S>0 THEN Y(X,2)=S-1
 2570 IF A$="S" AND S<9 THEN Y(X,2)=S+1
 2580 IF A$="E" AND T<10 THEN Y(X,3)=T+1
 2590 IF A$="W" AND T>0 THEN Y(X,3)=T-1
 2600 IF A$="NE" AND S>0 AND T<10 THEN G
OSUB 9500
 2610 IF A$="SE" AND S<9 AND T<10 THEN G
OSUB 9550
 2620 IF A$="SW" AND S<9 AND T>1 THEN GO
SUB 9600
 2630 IF A$="NW" AND S>0 AND T>1 THEN GO
SUB 9650
 2640 NEXT X
 2650 RETURN
 2997 :
 2998 :
 2999 REM ENEMY ORDERS
 3000 LET S=Y(1,2)
 3010 LET T=Y(1,3)
 3020 FOR X=1 TO 5
 3030 IF E(X,1)<0 THEN GOTO 3110
 3040 LET M=SGN(S-E(X,2))
 3050 LET N=SGN(T-E(X,3))
 3060 FOR C=1 TO 5
 3070 IF E(C,1)<0 OR C=X THEN GOTO 3090
 3080 IF E(C,2)=E(X,2)+M AND E(C,3)=E(X,
3)+N THEN GOSUB 9700
 3090 NEXT C
 3100 LET E(X,2)=E(X,2)+M
 3110 LET E(X,3)=E(X,3)+N
 3120 NEXT X
 3130 RETURN
 3497 :
 3498 :
 3499 REM COLLISION TESTING
```

```
 3500 FOR X=1 TO 4
 3510 IF Y(X,1)<0 THEN GOTO 3600
 3520 FOR Y=X+1 TO 5
 3530 IF Y(Y,1)<0 THEN GOTO 3550
 3540 IF Y(X,2)=Y(Y,2) AND Y(X,3)=Y(Y,3)
 THEN GOSUB 6000
 3550 NEXT Y
 3560 FORY=1 TO 5
 3570 IF E(Y,1)<0 THEN GOTO 3590
 3580 IF Y(X,2)=E(Y,2) AND Y(X,3)=E(Y,3)
 THEN GOTO 6100
 3590 NEXT Y
 3600 NEXT X
 3610 RETURN
 3997 :
 3998 :
 3999 REM FIGHTING TESTS
 4000 FOR X=1 TO 5
 4010 IF Y(X,1)<0 THEN GOTO 4060
 4020 FOR Y=1 TO 5
 4030 IF E(Y,1)<0 THEN GOTO 4050
 4040 IF ABS(Y(X,2)-E(Y,2))<=2 AND ABS(Y
(X,3)-E(Y,3))<=2 THEN GOSUB 6500
 4050 NEXT Y
 4060 NEXT X
 4070 RETURN
 4497 :
 4498 :
 4499 REM DAMAGE REPORTS
 4500 FOR X=1 TO 5
 4510 IF Y(X,1)<-5 THEN GOTO 4530
 4520 IF Y(X,1)<.5 THEN PRINT"SHIP ";X;"
 REPORTS SEVERE DAMAGE."
 4530 NEXT X
 4540 RETURN
 4997 :
 4998 :
 4999 REM SINKING TEST
 5000 FOR X=1 TO 5
 5010 IF Y(X,1)<=RND(1)/10 AND Y(X,1)<>-
100 THEN GOSUB 7000
```

```
 5020 NEXT X
 5030 FOR X=1 TO 5
 5040 IF E(X,1)<=RND(1)/10 AND E(X,1)<>-
100 THEN GOSUB 7100
 5050 NEXT X
 5060 RETURN
 5500 LET T=0
 5510 FOR X=1 TO 5
 5520 IF E(X,1)>0 THEN T=1
 5530 NEXT X
 5540 IF T=0 THEN GOTO 7500
 5550 IF Y(1,1)<0 THEN GOTO 7600
 5560 RETURN
 5988 :
 5989 :
 5990 REM MINOR SUBROUTINES
 5999 :
 6000 PRINT"SHIPS ";X;" AND ";Y;" HAVE C
OLLIDED."
 6010 LET Y(X,1)=-100
 6020 LET Y(Y,1)=-100
 6030 RETURN
 6099 :
 6100 PRINT"SHIP ";X;" HAS COLLIDED WITH
 AN ENEMY SHIP."
 6110 LET Y(X,1)=-100
 6120 LET E(Y,1)=-100
 6130 RETURN
 6499 :
 6500 LET RF=1/(.01+(ABS(Y(X,2)-E(Y,2))
^3+(ABS(Y(X,3)-E(Y,3)))^3)
 6510 LET P1=Y(X,1)*RF
 6520 LET P2=E(Y,1)*RF
 6530 LET Y(X,1)=Y(X,1)-P2/20
 6540 LET E(Y,1)=E(Y,1)-P1/20
 6550 PRINT"SHIP ";X;" HAS ENGAGED THE E
NEMY."
 6560 RETURN
 6999 :
 7000 PRINT"SHIP ";X;" HAS SUNK."
 7010 LET Y(X,1)=-100
```

```
 7020 RETURN
 7099 :
 7100 PRINT"AN ENEMY VESSEL HAS SUNK."
 7110 LET E(X,1)=-100
 7120 RETURN
 7499 :
 7500 PRINT
 7510 PRINT"YOU HAVE DESTROYED THE ENEMY
 FLEET IN"
 7520 PRINT H;" HOURS."
 7530 GOTO 8000
 7599 :
 7600 PRINT
 7610 PRINT"YOUR FLAGSHIP HAS BEEN DESTR
OYED."
 7620 GOTO 8000
 7999 :
 8000 PRINT
 8010 PRINT"DO YOU WANT ANOTHER GAME?"
 8020 INPUTA$
 8030 IF A$="Y" THEN RUN
 8040 END
 8999 :
 9000 PRINT
 9010 PRINT"PRESS A KEY TO CONTINUE"
 9020 REPEAT UNTIL GET$<>""
 9030 RETURN
 9497 REM:
 9498 REM:
 9499 REM MOVEMENT SUBROUTINES
 9500 LET Y(X,2)=S-1
 9510 LET Y(X,3)=T+1
 9520 RETURN
 9550 LET Y(X,2)=S+1
 9560 LET Y(X,3)=T+1
 9570 RETURN
 9600 LET Y(X,2)=S+1
 9610 LET Y(X,3)=T-1
 9620 RETURN
 9650 LET Y(X,2)=S-1
 9660 LET Y(X,3)=T-1
```

```
9670 RETURN
9700 LETM=0
9710 LETN=0
9720 RETURN
```

The Siege of Doune Castle

This is a challenging and complicated game involving the capture of a castle, using various types of equipment. You have to take care that enough food is available for the men, as well as paying attention to matters of military strategy.

The main daily report gives a list of your equipment, and asks for your orders. These include requests for new supplies, from your supporting teams. If you request more than they can produce, they scale the demands down. As well as battle reports, the figures for new equipment supplied are displayed at the end of each day.

The computer's orders are worked out in a simple but effective manner, using tallies of your attacking distribution and the state of the castle's walls to decide on the allocation of men, archers and cauldrons of boiling oil.

All equipment and men are stored in arrays, the appropriate names sorted out from the DATA statements at the end of the program.

Incidentally, Doune Castle is the one used by Monty Python in the Trojan Rabbit sequence of one of their films.

```
 10 REM MASTER ROUTINE
 20 GOSUB 1000:REM INITIALISATION
 30 CLS
 40 GOSUB 1500:REM INTRO
 50 GOSUB 9200
 60 LET DAY=1
 70 GOSUB 2000:REM DISPLAY A
 80 GOSUB 2500:REM RATION RATES
 90 GOSUB 3000:REM CONSTRUCTION INPUTS
100 GOSUB 3500:REM BATTLE INPUTS
110 GOSUB 4000:REM COMPUTER'S ORDERS
120 GOSUB 4400:REM BOMBARDMENT TEST
130 GOSUB 4500:REM BATTLE TEST
140 GOSUB 4700:REM RANDOM DEATHS
150 GOSUB 5000:REM WIN/LOSS
160 IF RND(1)<.3 THEN GOSUB 5200:REM
ENEMY DESERTER
```

```
  170 GOSUB 5300:REM ENEMY REPAIRS
  180 GOSUB 9200
  190 GOSUB 5500:REM REPORT+VARS CHANGE
  200 GOSUB 9200
  210 LET DAY=DAY+1
  220 GOSUB 8300:REM VARIABLE FLUSH
  230 GOTO 70
  997 :
  998 :
  999 REM INITIALISATION
 1000 DIM A(8):DIM P(6,6):DIM N(6)
 1010 DIM E(3):DIM D(6,4):DIM T(6)
 1020 DIM C(6)
 1030 DIM L(6)
 1040 RESTORE 9900
 1050 FOR X=1 TO 8
 1060 READ I
 1070 LET A(X)=I
 1080 IF X>2 THEN LET A(X)=INT(RND(1)*I)
 1090 NEXT X
 1100 LET E(1)=300:LET E(2)=50:LET E(3)=
INT(RND(1)*10+1)
 1110 FOR X=1 TO 6
 1120 LET C(X)=1:LET T(X)=1
 1130 NEXT X
 1140 RETURN
 1497 :
 1498 :
 1499 REM INTRO
 1500 PRINT"THE SIEGE OF DOUNE CASTLE"
 1510 PRINT
 1520 PRINT"It is the 14th century. You
command"
 1530 PRINT"an army of 300 infantrymen,
laying"
 1540 PRINT"siege to Doune Castle, Scotl
and."
 1550 PRINT"In your role of general, you
 request"
 1560 PRINT"the quantities of supplies a
nd"
```

```
 1570 PRINT"equipment required from your
 supporting"
 1580 PRINT"tradesmen who will deliver a
ll they can."
 1590 PRINT
 1600 PRINT"In calculating the food you
will use,"
 1610 PRINT"remember that each man and a
rcher in"
 1620 PRINT"the army eats one tenth of a
 sack of"
 1630 PRINT"food in a day."
 1640 RETURN
 1997 :
 1998 :
 1999 REM DISPLAY A
 2000 PRINT"DAY ";DAY;" OF THE SIEGE"
 2010 PRINT
 2020 RESTORE 9600
 2030 FOR X=1 TO 8
 2040 READ I$
 2050 PRINT I$;":";TAB(15);A(X)
 2060 NEXT X
 2070 RESTORE 9700
 2080 FOR X=1 TO 6
 2090 READ I$
 2100 PRINT I$;":";
 2110 LET Z=C(X)
 2120 GOSUB 6000
 2130 NEXT X
 2140 RETURN
 2497 :
 2499 REM RATION RATES
 2500 PRINT TAB(0,17);"Do you wish to is
sue full rations?"
 2510 INPUTA$
 2520 IF A$="Y" THEN LET R=1
 2530 IF A$<>"Y" THEN GOSUB 6200
 2540 LET A(8)=INT(A(8)-(A(1)+A(2))*R/10
)
 2550 RETURN
```

```
 2997 :
 2998 :
 2999 REM CONSTRUCTION INPUTS
 3000 RESTORE 9610
 3010 FOR X=1 TO 6
 3020 GOSUB 9500
 3030 READ I$
 3040 PRINT"HOW MANY ";I$;" DO YOU NEED?
"
 3050 INPUT A
 3060 LET A=ABS(INT(A))
 3070 LET N(X)=A
 3080 NEXT X
 3090 LET TL=1
 3100 RESTORE 9800
 3110 FOR X=1 TO 6
 3120 READ I
 3130 LET TL=TL+N(X)*I
 3140 NEXT X
 3150 FOR X=1 TO 6
 3160 LET N(X)=INT(500*N(X)/TL)
 3170 NEXT X
 3180 RETURN
 3397 :
 3497 :
 3498 :
 3499 REM BATTLE INPUTS
 3500 RESTORE 9700
 3510 FOR X=1 TO 6
 3520 LET L(X)=0
 3530 GOSUB 9500
 3540 READ I$
 3550 PRINT"Do you wish to attack the ";
I$;"?"
 3560 INPUT A$
 3570 IF A$="Y" THEN LET L(X)=1
 3580 NEXT X
 3590 :
 3600 FOR X=1 TO 6
 3610 IF L(X)=0 THEN GOTO 3800
 3620 RESTORE 9700
```

```
 3630 FOR Z=1 TO X
 3640 READ I$
 3650 NEXT Z
 3660 RESTORE 9600
 3670 FOR Y=1 TO 6
 3680 READ J$
 3690 IF X<6 AND Y=6 OR X=6 AND Y>2 AND
Y<6 OR A(Y)=0 THEN GOTO 3780
 3700 GOSUB 9500
 3710 PRINT I$;" attack:"
 3720 PRINT"HOW MANY ";J$;"?"
 3730 INPUT A
 3740 LET A=ABS(INT(A))
 3750 IF A(Y)-A<0 THEN GOTO 6400
 3760 LET A(Y)=A(Y)-A
 3770 LET P(X,Y)=A
 3780 NEXT Y
 3790 IF P(X,1)=0 AND P(X,2)=0 THEN LET
L(X)=-1
 3800 NEXT X
 3810 CLS
 3820 PRINT"You left ";A(1);" men on gua
rd."
 3830 RETURN
 3997 :
 3998 :
 3999 REM COMPUTER'S ORDERS
 4000 LET C=0:LET T=0
 4010 FOR X=1 TO 6
 4020 LET T=T+T(X)
 4030 LET C=C+2-C(X)
 4040 NEXT X
 4050 FOR X=1 TO 6
 4060 FOR Y=1 TO 3
 4070 LET D(X,Y)=INT(E(Y)*T(X)/T)
 4080 LET E(Y)=E(Y)-D(X,Y)
 4090 NEXT Y
 4100 LET D(X,4)=INT(50*(2-C(X))/C)
 4110 NEXT X
 4120 RETURN
 4397 :
```

```
 4398 :
 4399 REM BOMBARDMENT TESTING
 4400 RESTORE 9700
 4410 FOR X=1 TO 5
 4420 READ I$
 4430 IF P(X,3)>0 AND X<6 THEN GOSUB 700
0
 4440 NEXT X
 4450 GOSUB 9200
 4460 RETURN
 4497 :
 4498 :
 4499 REM BATTLE TESTS
 4500 RESTORE 9700
 4510 FOR X=1 TO 6
 4520 READ I$
 4530 IF L(X)>0 THEN PRINT I$;" attack:"
 4540 IF L(X)<0 THEN PRINT"You can't att
ack without people, idiot."
 4550 IF L(X)>0 THEN GOSUB 7500
 4560 NEXT X
 4570 GOSUB 9200
 4580 RETURN
 4697 :
 4698 :
 4699 REM RANDOM DEATHS
 4700 PRINT
 4710 LET DIS=INT(RND(1)*A(1)/2)
 4720 LET A(1)=A(1)-DIS
 4730 IF DIS>0 THEN PRINT"Today, ";DIS;"
 men died of disease."
 4740 :
 4750 LET DES=0
 4760 IF R<1 THEN LET DES=INT(RND(1)*A(1
))
 4770 IF DES>0 THEN PRINT"Because of low
 rations, ";DES;" deserted."
 4780 LET A(1)=A(1)-DES
 4790 LET E(1)=E(1)+DES
 4800 RETURN
 4997 :
```

```
 4998 :
 4999 REM WIN/LOSS
 5000 FOR X=1 TO 6
 5010 FOR Y=1 TO 2
 5020 LET A(Y)=A(Y)+P(X,Y)
 5030 LET E(Y)=E(Y)+D(X,Y)
 5040 NEXT Y
 5050 LET E(3)=E(3)+D(X,3)
 5060 NEXT X
 5070 :
 5080 IF A(1)<20*RND(1) THEN GOTO 8000
 5090 IF A(8)<0 THEN GOTO 8100
 5100 :
 5110 IF E(1)<0 OR E(1)<50 AND E(1)<A(1)
 THEN GOTO 8200
 5120 RETURN
 5197 :
 5199 REM ENEMY DESERTER
 5200 PRINT"An enemy has deserted. He cl
aims that"
 5210 PRINT"there are ";E(1)+INT(10*(RND
(1)-RND(1)));" of them left."
 5220 LET E(1)=E(1)-1
 5230 LET A(1)=A(1)+1
 5240 RETURN
 5297 :
 5298 :
 5299 REM ENEMY REPAIRS
 5300 FOR X=1 TO 6
 5310 LET C(X)=INT(C(X)+D(X,4)/300)
 5320 IF C(X)>1 THEN LET C(X)=1
 5330 NEXT X
 5340 RETURN
 5497 :
 5498 :
 5499 REM REPORT+VARIABLE CHANGE
 5500 PRINT"The labourers have supplied
you with:"
 5510 RESTORE 9610
 5520 FOR X=1 TO 6
 5530 READ I$
```

```
 5540 PRINT I$;":";TAB(15);N(X)
 5550 LET A(X+2)=A(X+2)+N(X)
 5560 NEXT X
 5570 :
 5580 FOR X=1 TO 6
 5590 IF L(X)=1 THEN LET T(X)=T(X)+1
 5600 FOR Y=3 TO 6
 5610 LET A(Y)=A(Y)+P(X,Y)
 5620 NEXT Y
 5630 NEXT X
 5640 RETURN
 5988 :
 5989 :
 5990 REM SUBROUTINES
 5998 :
 5999 REM CONDITION ROUTINE
 6000 IF Z=1 THEN PRINT"un";
 6010 IF Z<1 AND Z>=.9 THEN PRINT"hardly
 ";
 6020 IF Z<.9 AND Z>=.7 THEN PRINT"sligh
tly ";
 6030 IF Z<.5 AND Z>=.3 THEN PRINT"badly
 ";
 6040 IF Z<.3 THEN PRINT"seriously ";
 6050 PRINT"damaged."
 6060 RETURN
 6197 :
 6198 :
 6199 REM DIFFERENT RATION RATE
 6200 PRINT"What decimal fraction of nor
mal rations"
 6210 PRINT"do you wish to issue?"
 6220 INPUT A
 6230 LET R=ABS(A)
 6240 RETURN
 6398 :
 6399 REM INSUFFICIENT QUANTITIES
 6400 PRINT"YOU DO NOT HAVE ENOUGH ";J$
 6410 FOR Q=1 TO 3000:NEXT:REM PAUSE
 6420 GOSUB 9500
 6430 GOTO 3700
```

```
 6997 :
 6998 :
 6999 REM BOMBARDMENT
 7000 LET H=0:LET TH=0
 7010 FOR Y=1 TO P(X,3)
 7020 IF A(7)-1<0 THEN GOTO 7070
 7030 LET A(7)=A(7)-1
 7040 LET H=INT(30*RND(1))
 7050 LET C(X)=C(X)-H/100
 7060 LET TH=TH+H
 7070 NEXT Y
 7080 PRINT
 7090 PRINT"The bombardment on the ";I$
 7100 PRINT"scored ";TH;" hits."
 7110 RETURN
 7497 :
 7498 :
 7499 REM BATTLE ROUTINE
 7500 LET BF=(15*P(X,4)+P(X,5)+3*P(X,6)+
2*P(X,2)/(1+D(X,2)))/(C(X)+.1)
 7510 LET M=P(X,1)
 7520 LET N=D(X,1)
 7530 GOSUB 9000
 7540 LET P(X,1)=M
 7550 LET D(X,1)=N
 7560 LET P(X,2)=INT(P(X,2)*(1-D(X,2)/50
))
 7570 LET D(X,2)=INT(D(X,2)*(1-P(X,2)/50
))
 7580 LET DC=INT(RND(1)*10*D(X,3))
 7590 IF DC>P(X,1) THEN LET DC=0
 7600 IF DC>0 THEN PRINT "The cauldrons
of boiling oil killed ";DC;"."
 7610 LET P(X,1)=P(X,1)-DC
 7620 IF X=6 THEN LET C(6)=C(6)-P(6,6)/2
0
 7630 FOR W=4 TO 6
 7640 LET P(X,W)=INT(RND(1)*P(X,W)/2)
 7650 NEXT W
 7660 PRINT
 7670 IF P(X,1)>(4+RND(1)*3)*D(X,1) THEN
```

```
 GOTO 8200
 7680 RETURN
 7995 :
 7996 :
 7997 REM ENDINGS
 7998 :
 7999 REM LOSS-NO MEN
 8000 PRINT
 8010 PRINT"Your few remaining men have
mutinied."
 8020 PRINT"You have lost on day ";DAY;"
."
 8030 GOTO 8500
 8098 :
 8099 REM LOSS-NO FOOD
 8100 PRINT
 8110 PRINT"You have run out of food. Yo
u have to"
 8120 PRINT"surrender on day ";DAY;"."
 8130 GOTO 8500
 8198 :
 8199 REM WIN-ENEMY SURRENDER
 8200 PRINT
 8210 PRINT"The castle has surrendered-v
ictory!"
 8220 PRINT"You won in ";DAY;" days."
 8230 GOTO 8500
 8297 :
 8298 :
 8299 REM VARIABLE FLUSH
 8300 FOR X=1 TO 6
 8310 LET N(X)=0
 8320 FOR Y=1 TO 4
 8330 LET D(X,Y)=0
 8340 IF Y>4 THEN GOTO 8360
 8350 LET P(X,Y)=0
 8360 NEXT Y
 8370 NEXT X
 8380 RETURN
 8498 :
 8499 REM REPLAY
```

```
 8500 PRINT
 8510 PRINT"Do you want another game?"
 8520 INPUT A$
 8530 IF A$="Y" THEN RUN
 8540 END
 8997 :
 8998 :
 8999 REM DEATHS SUBROUTINE
 9000 LET AD=INT(RND(1)*M/(2*LN(ABS(BF)+
10)))
 9010 IF AD>M THEN LET AD=INT(RND(1)*M)
 9020 LET ED=INT(RND(1)*N)
 9030 PRINT"You lost ";AD;" men. The ene
my lost ";ED;"."
 9040 LET M=M-AD
 9050 LET N=N-ED
 9060 RETURN
 9197 :
 9198 :
 9199 REM KEY PRESS ROUTINE
 9200 PRINT
 9210 PRINT"PRESS A KEY TO CONTINUE"
 9220 REPEAT UNTIL GET$<>""
 9230 CLS
 9240 RETURN
 9497 :
 9498 :
 9499 REM SPACES ROUTINE
 9500 PRINT TAB(0,17);
 9510 FOR Q=1 TO 5
 9520 PRINT"
              ";
 9530 NEXT Q
 9540 PRINT TAB(0,17);
 9550 RETURN
 9599 :
 9600 DATA MEN,ARCHERS
 9610 DATA CATAPULTS,SIEGE TOWERS,LADDER
S,BATTERING-RAMS,SACKS OF ROCK,SACKS OF
FOOD
 9700 DATA North Wall,South Wall,East Wa
```

```
11,West Wall,Keep,Gates
 9800 DATA 50,100,10,30,2,1
 9900 DATA 300,50,5,4,10,6,10,200
```

Third World War

This is the game of a war in Western Europe, the object being for either side to penetrate 1ØØkm into the other side's territory. It is possible to use nuclear, biological and chemical weapons, but the computer is likely to retaliate and, if nuclear exchange reaches a high level, a holocaust breaks out.

The game uses a menu to enable the player to select reports on any sector of the front, to move men and equipment, and to change the levels of non-conventional warfare. The advances made in each sector are reported at the end of each loop.

The computer uses two strategies in its movements. It attacks your highest concentrations of men and tanks with appropriate counter-measures (missile launchers and field guns – which are the most effective), and it pours most men and tanks into the sectors with the greatest advance or retreat, which is supposed to be the actual Soviet policy, by the way.

All equipment is contained in arrays, with the names held in DATA statements. DATA statements also contain the effectivenesses of each type of equipment against all the other types (see lines 494Ø – 497Ø). This program is thus easily adapted for other types of equipment, if you want to try any.

```
   5 REM MASTER ROUTINE
  10 CLS
  20 GOSUB 1000:REM INITIALISE
  30 GOSUB 1500:REM INTRO
  40 LET D=1
  50 GOSUB 2000:REM MENU
  60 GOSUB 4000:REM ENEMY ACTION
  70 GOSUB 4500:REM SITUATION CHANGE
  80 GOTO 5000:REM END CHECK
  90 LET D=D+1
 100 GOTO 50
 997 :
 998 :
 999 REM INITIALISE
1000 DIM P(6,7)
```

```
1010 DIM E(6,7)
1020 DIM F(5)
1030 DIM G(5)
1040 FOR X=1 TO 4
1050 READ I
1060 LET P(6,X)=I
1070 LET E(6,X)=I
1080 NEXT X
1090 RETURN
1099 :
1100 DATA 100000,400,500,150
1497 :
1498 :
1499 REM INTRO
1500 PRINT"THIRD WORLD WAR"
1510 PRINT
1520 PRINT"You are the commander of NAT
O forces in"
1530 PRINT"Europe at the outbreak of wa
r. You"
1540 PRINT"direct land forces between t
he reserves"
1550 PRINT"and the 5 front-line sectors
. You can"
1560 PRINT"not send forces between sect
ors"
1570 PRINT"directly."
1580 PRINT
1590 PRINT"If either side penetrates to
 a cepth of"
1600 PRINT"100 km inside enemy territor
y with"
1610 PRINT"sufficient force, the battle
 is over."
1620 PRINT
1630 PRINT"It is also possible to start
 an all-out"
1640 PRINT"nuclear war, destroying the
whole world"
1650 PRINT"if you are not careful."
1660 GOSUB 9900
```

```
1670 RETURN
1997 :
1998 :
1999 REM MENU
2000 CLS
2010 PRINT"Day ";D;" of the battle"
2020 PRINT
2030 PRINT"Options:"
2040 PRINT"1. Display situation in a se
ctor."
2050 PRINT"2. Give movement orders."
2060 PRINT"3. Give escalation orders."
2070 PRINT"4. End of day's orders."
2080 PRINT"Which one ?"
2090 INPUT A
2100 LET A=INT(ABS(A))
2110 IF A>4 THEN LET A=4
2120 IF A<1 THEN LET A=1
2130 IF A=4 THEN RETURN
2140 IF A=2 THEN PRINT"To which ";
2150 IF A<>2 THEN PRINT"Which ";
2160 PRINT"sector ? (6 for reserves)"
2170 INPUT S
2180 LET S=ABS(INT(S))
2190 IF S>6 OR S<1 THEN RETURN
2200 IF A=1 THEN GOSUB 2500
2210 IF A=2 THEN GOSUB 3000
2220 IF A=3 THEN GOSUB 3500
2230 GOSUB 9900
2240 GOTO 2000
2498 :
2499 REM SITUATION DISPLAY
2500 CLS
2510 IF S<6 THEN PRINT"Sector ";S
2520 IF S=6 THEN PRINT"Reserves"
2530 PRINT
2540 RESTORE 2800
2550 FOR X=1 TO 4
2560 READ I$
2570 PRINT I$;TAB(20);P(S,X)
2580 NEXT X
```

```
 2590 IF S=6 THEN RETURN
 2600 PRINT"Proportion of special shells
:"
 2610 FOR X=5 TO 7
 2620 READ I$
 2630 PRINT I$;TAB(20);P(S,X);"%"
 2640 NEXT X
 2650 PRINT
 2660 PRINT"The front line has ";
 2670 IF F(S)>=0 THEN PRINT"advanced";
 2680 IF F(S)<0 THEN PRINT"retreated";
 2690 PRINT;" ";ABS(F(S));" km"
 2700 PRINT"altogether."
 2710 RETURN
 2799 :
 2800 DATA Men,Tanks,Field guns,Missile
launchers
 2810 DATA Biological,Chemical,Nuclear
 2997 :
 2998 :
 2999 REM MOVEMENT ORDERS
 3000 IF S=6 THEN GOSUB 3300
 3010 PRINT
 3020 PRINT"1. Men"
 3030 PRINT"2. Tanks"
 3040 PRINT"3. Field guns"
 3050 PRINT"4. Missile launchers"
 3060 PRINT"Which one ?"
 3070 INPUT E
 3080 LET E=INT(ABS(E))
 3090 IF E>4 OR E<1 THEN LET E=1
 3100 PRINT"How many ?"
 3110 INPUT Q
 3120 IF S<6 THEN LET T=6
 3130 IF Q>P(T,E) THEN LET Q=P(T,E)
 3140 LET P(S,E)=P(S,E)+Q
 3150 LET P(T,E)=P(T,E)-Q
 3160 RETURN
 3299 :
 3300 PRINT"From which sector ?"
 3310 INPUT T
```

```
 3320 LET T=INT(ABS(T))
 3330 IF T>5 OR T<1 THEN LET T=6
 3340 RETURN
 3498 :
 3499 REM ESCALATION
 3500 PRINT
 3510 PRINT"1. Biological"
 3520 PRINT"2. Chemical"
 3530 PRINT"3. Nuclear"
 3540 PRINT"Which type ?"
 3550 INPUT T
 3560 IF T>3 OR T<1 OR T<>ABS(T) THEN RE
TURN
 3570 PRINT"To what % of shells fired ?"
 3580 INPUT Q
 3590 LET Q=ABS(INT(Q))
 3600 IF Q>30 THEN LET Q=30
 3610 LET P(S,4+T)=Q
 3620 RETURN
 3997 :
 3998 :
 3999 REM ENEMY ACTION
 4000 LET FC=0:LET MC=0:LET TD=0
 4010 FOR X=1 TO 5
 4020 LET TD=TD+ABS(G(X)-F(X))
 4030 LET FC=FC+P(X,1)
 4040 LET MC=MC+P(X,2)
 4050 FOR Y=1 TO 4
 4060 LET E(X,Y)=INT(E(X,Y)/2)
 4070 LET E(6,Y)=E(6,Y)+E(X,Y)
 4080 NEXT Y
 4090 NEXT X
 4100 FOR X=1 TO 5
 4110 FOR Y=1 TO 2
 4120 LET C=INT(ABS(G(X)-F(X)+1)/(TD+5)*
E(6,Y))
 4130 LET E(X,Y)=E(X,Y)+C
 4140 NEXT Y
 4150 LET C=INT((P(X,1)+1)/(FC+5)*E(6,3)
)
 4160 LET E(X,3)=E(X,3)+C
```

```
 4170 LET C=INT((P(X,2)+1)/(MC+5)*E(6,4)
)
 4180 LET E(X,4)=E(X,4)+C
 4190 FOR Y=5 TO 7
 4200 LET E(X,Y)=P(X,Y)+INT((RND(1)-RND(
1))*5)
 4210 IF P(X,1)>E(X,1) AND E(X,1)>0 THEN
 LET E(X,Y)=20+INT(RND(1)*10)
 4220 IF E(X,Y)>30 THEN LET E(X,Y)=30
 4230 IF E(X,Y)<0 THEN LET E(X,Y)=0
 4240 NEXT Y
 4250 NEXT X
 4260 FOR Y=1 TO 4
 4270 LET E(6,Y)=0
 4280 NEXT Y
 4290 RETURN
 4497 :
 4498 :
 4499 REM SITUATION CHANGE
 4500 FOR X=1 TO 5
 4510 CLS
 4520 LET DP=E(X,5)+E(X,6)+E(X,7)
 4530 LET DE=P(X,5)+P(X,6)+P(X,7)
 4540 FOR Y=1 TO 4
 4550 LET P(X,Y)=INT(P(X,Y)*(1-DP/300))
 4560 LET E(X,Y)=INT(E(X,Y)*(1-DE/300))
 4570 NEXT Y
 4580 RESTORE 4940
 4590 FOR Y=1 TO 4
 4600 FOR Z=1 TO 4
 4610 READ I
 4620 LET P(X,Y)=INT(P(X,Y)-I*E(X,Z))
 4630 IF P(X,Y)<0 THEN LET P(X,Y)=0
 4640 LET E(X,Y)=INT(E(X,Y)-I*P(X,Z))
 4650 IF E(X,Y)<0 THEN LET E(X,Y)=0
 4660 NEXT Z
 4670 NEXT Y
 4680 LET G(X)=F(X)
 4690 LET F(X)=INT(F(X)+(P(X,1)+P(X,2)*5
-E(X,1)-E(X,2)*5)/2E3*(1+(RND(1)-RND(1))
/5))
```

```
 4700 RESTORE 2800
 4710 PRINT"Sector ";X
 4720 PRINT
 4730 PRINT"Reports show the enemy have
"
 4740 FOR Y=1 TO 4
 4750 IF P(X,Y)<0 THEN LET P(X,Y)=0
 4760 IF E(X,Y)<0 THEN LET E(X,Y)=0
 4770 READ I$
 4780 PRINT;I$;TAB(20);E(X,Y)
 4790 NEXT Y
 4800 PRINT"in the area."
 4810 PRINT"The front line has ";
 4820 IF F(X)-G(X)>=0 THEN PRINT"advance
d";
 4830 IF F(X)-G(X)<0 THEN PRINT"retreate
d";
 4840 PRINT" by ";ABS(F(X)-G(X));" km"
 4850 PRINT
 4860 PRINT"Proportion of special shells
:"
 4870 FOR Y=5 TO 7
 4880 READ I$
 4890 PRINT;I$;TAB(20);E(X,Y);"%"
 4900 NEXT Y
 4910 GOSUB 9900
 4920 NEXT X
 4930 RETURN
 4939 :
 4940 DATA .1,.3,.5,.3
 4950 DATA 1E-3,.1,.1,.4
 4960 DATA .01,.2,.2,.3
 4970 DATA .1,.2,.2,.3
 4997 :
 4998 :
 4999 REM END CHECK
 5000 CLS
 5010 FOR X=1 TO 5
 5020 IF P(X,7)+E(X,7)>=50 AND RND(1)<.4
 THEN GOTO 5500
 5030 IF F(X)>100 THEN GOTO 6000
```

```
 5040 IF F(X)<-100 THEN GOTO 6500
 5050 NEXT X
 5060 GOTO 90
 5497 :
 5498 :
 5499 REM NUCLEAR HOLOCAUST
 5500 PRINT"A nuclear holocaust has brok
en out. The"
 5510 PRINT"world has been destroyed";
 5520 GOTO 7000
 5997 :
 5998 :
 5999 REM VICTORY
 6000 PRINT"You penetrated 100 km inside
 enemy"
 6010 PRINT"territory in sector ";X;". T
he enemy"
 6020 PRINT"surrendered";
 6030 GOTO 7000
 6497 :
 6498 :
 6499 REM LOSS
 6500 PRINT"The enemy penetrated 100 km
inside your"
 6510 PRINT"territory in secto  ";X;". Y
ou lost"
 6520 GOTO 7000
 6997 :
 6998 :
 6999 REM REPLAY
 7000 PRINT" on day ";D;"."
 7010 PRINT
 7020 PRINT"Do you want another game ?"
 7030 INPUT A$
 7040 IF A$="Y" THEN RUN
 7050 END
 9897 :
 9898 :
 9899 REM KEY PRESS ROUTINE
 9900 PRINT
 9910 PRINT"PRESS A KEY TO CONTINUE"
```

```
9920 REPEAT UNTIL GET$<>""
9930 CLS
9940 RETURN
```

Laserfight in the OK Space Zone

This game shows you how much ability you have in picturing situations in three dimensions. You direct your ships to find the enemy fleets as they invoke their incredible tactics to avoid you.

In actual fact, the tactics the computer uses rely heavily on chance. The computer picks two sectors at random, and heads for the one containing less of your ships. Since most of the sectors will be empty, the enemy fleets head for the second position investigated. You can guess with fair accuracy that this makes the fleets head towards the centre most of the time.

The daily display begins by going through all the squares in turn and identifying ones containing your ships. The area surrounding these is then searched for enemy ships, whose presence is reported. The same procedure is repeated to find ships for you to move. When – finally – you manage to force an encounter with the enemy fleets, the results are also reported.

As usual, equipment is located in arrays.

The workings of the program are very simple and straightforward, but long and laborious. The two types of ship are dealt with separately, which increases the program length considerably.

A tip on tactics: post picket ships to report on the whereabouts of enemy ships, covering the entire region. This uses eight ships.

```
  5 REM MASTER ROUTINE
 10 CLS
 20 GOSUB 1000:REM INTRO
 30 GOSUB 1500:REM INITIALISE
 40 LET T=1
 50 GOSUB 2000:REM POSITION REPORTS
 60 GOSUB 2500:REM MOVEMENT
 70 GOSUB 3000:REM ENEMY MOVEMENTS
 80 LET S=0
 90 LET ES=0
100 GOSUB 3500:REM BATTLES & COUNTING
110 IF S>10*ES AND RND(1)<.3 THEN GOTO
```

```
 4000:REM ENEMY SURRENDER
  120 IF ES>10*S THEN GOTO 4500:REM ASK
IF WITHDRAW
  130 IF S=0 THEN GOTO 6000:REM LOSE
  140 IF ES=0 THEN GOTO 6500:REM WIN
  150 LET T=T+1
  160 GOTO 50
  997 :
  998 :
  999 REM INTRO
 1000 PRINT"LASERFIGHT IN THE OK SPACE Z
ONE"
 1010 PRINT
 1020 PRINT"You are the intrepid admiral
 of space-"
 1030 PRINT"fleet 5, comprising 400 ship
s, of two"
 1040 PRINT"types. Your thankless task i
s to"
 1050 PRINT"destroy the forces of the ev
il Prince"
 1060 PRINT"Orion, arch-enemy of Justice
, and a"
 1070 PRINT"pretty mean hand at space ta
ctics. The"
 1080 PRINT"aforementioned prince is in
possession"
 1090 PRINT"of a similar fleet."
 1100 GOSUB 9900
 1110 PRINT"The region of space where yo
u finally"
 1120 PRINT"confront him comprises 216 s
ectors -"
 1130 PRINT"6x6x6. Although all weaponry
 is only"
 1140 PRINT"effective within the same se
ctor as the"
 1150 PRINT"ship, the detectors have a r
ange of 1"
 1160 PRINT"sector in all directions, an
d all the"
```

```
 1170 PRINT"ships in your fleet report w
hat they"
 1180 PRINT"have detected to you by hype
r-wave, so"
 1190 PRINT"you can tell tell what is go
ing on far"
 1200 PRINT"away. Also, type 2 ships are
 about"
 1210 PRINT"twice as strong and destruct
ive as type"
 1220 PRINT"1s."
 1230 PRINT
 1240 PRINT"Happy hunting!"
 1250 GOSUB 9900
 1260 RETURN
 1497 :
 1498 :
 1499 REM INITIALISE
 1500 DIM S(6,6,6,2)
 1510 DIM E(6,6,6,2)
 1520 LET S(1,1,1,1)=300
 1530 LET S(1,1,1,2)=100
 1540 DIM F(5,3)
 1550 FOR F=1 TO 5
 1560 LET X=INT(RND(1)*5)+1
 1570 LET Y=INT(RND(1)*5)+1
 1580 LET Z=INT(RND(1)*5)+1
 1590 LET F(F,1)=X
 1600 LET F(F,2)=Y
 1610 LET F(F,3)=Z
 1620 LET E(X,Y,Z,1)=60
 1630 LET E(X,Y,Z,2)=20
 1640 NEXT F
 1650 RETURN
 1997 :
 1998 :
 1999 REM POSITION REPORTS
 2000 CLS
 2010 PRINT"Standard Day ";T;" of the en
counter."
 2020 PRINT"Standby for reports from the
```

```
 fleet..."
 2030 PRINT
 2040 FOR X=1 TO 3000:NEXT
 2050 FOR X=1 TO 6
 2060 FOR Y=1 TO 6
 2070 FOR Z=1 TO 6
 2080 IF S(X,Y,Z,1)>0 OR S(X,Y,Z,2)>0 TH
EN GOSUB 2150
 2090 NEXT Z
 2100 NEXT Y
 2110 NEXT X
 2120 RETURN
 2149 :
 2150 PRINT"Fleet detachment at (";X;","
;Y;",";Z;"),"
 2160 PRINT"consisting of ";S(X,Y,Z,1);"
 type 1 ships"
 2170 PRINT"and ";S(X,Y,Z,2);" type 2 sh
ips reports:"
 2180 LET TE=0
 2190 FOR XP=X-1 TO X+1
 2200 IF XP<1 OR XP>6 THEN GOTO 2290
 2210 FOR YP=Y-1 TO Y+1
 2220 IF YP<1 OR YP>6 THEN GOTO 2280
 2230 FOR ZP=Z-1 TO Z+1
 2240 IF ZP<1 OR ZP>6 THEN GOTO 2270
 2250 IF E(XP,YP,ZP,1)>0 THEN PRINT" ";E
(XP,YP,ZP,1);" type 1s at (";XP;",";YP;"
,";ZP;")":LET TE=TE+1
 2260 IF E(XP,YP,ZP,2)>0 THEN PRINT" ";E
(XP,YP,ZP,2);" type 2s at (";XP;",";YP;"
,";ZP;")":LET TE=TE+1
 2270 NEXT ZP
 2280 NEXT YP
 2290 NEXT XP
 2300 IF TE=0 THEN PRINT"No enemy ships
within detector range."
 2310 GOSUB 9900
 2320 RETURN
 2497 :
 2498 :
```

```
 2499 REM MOVEMENT
 2500 PRINT"The fleet is waiting for you
r orders -"
 2510 GOSUB 9900
 2520 FOR X=1 TO 6
 2530 FOR Y=1 TO 6
 2540 FOR Z=1 TO 6
 2550 IF S(X,Y,Z,1)>0 OR S(X,Y,Z,2)>0 TH
EN GOSUB 2650
 2560 NEXT Z
 2570 NEXT Y
 2580 NEXT X
 2590 RETURN
 2649 :
 2650 PRINT"In sector (";X;",";Y;",";Z;"
), you have:"
 2660 PRINT" ";S(X,Y,Z,1);" type 1 ships
"
 2670 PRINT" ";S(X,Y,Z,2);" type 2 ships
"
 2680 PRINT
 2690 FOR SX=1 TO 2
 2700 IF S(X,Y,Z,SX)=0 THEN GOTO 2850
 2710 PRINT"How many type ";SX;" ships d
o you want to"
 2720 PRINT"move ?"
 2730 INPUT A
 2740 IF A<1 THEN GOTO 2850
 2750 LET A=ABS(INT(A))
 2760 IF A>S(X,Y,Z,SX) THEN LET A=S(X,Y,
Z,SX)
 2770 PRINT"Where do you want to move th
em to ?"
 2780 PRINT"(Enter coordinates in usual
order)"
 2790 INPUT DX
 2800 INPUT DY
 2810 INPUT DZ
 2820 IF ABS(X-DX)>1 OR ABS(Y-DY)>1 OR A
BS(Z-DZ)>1 OR DX<1 OR DY<1 OR DZ<1 OR DX
>6 OR DY>6 OR DZ>6 THEN GOTO 2850
```

```
 2830 LET S(DX,DY,DZ,SX)=S(DX,DY,DZ,SX)+
A
 2840 LET S(X,Y,Z,SX)=S(X,Y,Z,SX)-A
 2850 NEXT SX
 2860 CLS
 2870 RETURN
 2997 :
 2998 :
 2999 REM ENEMY MOVEMENTS
 3000 FOR F=1 TO 5
 3010 LET X=F(F,1)
 3020 LET Y=F(F,2)
 3030 LET Z=F(F,3)
 3040 IF E(X,Y,Z,1)=0 THEN GOTO 3140
 3050 LET SD=0
 3060 FOR P=1 TO 2
 3070 LET NX=INT(RND(5)+1)
 3080 LET NY=INT(RND(5)+1)
 3090 LET NZ=INT(RND(5)+1)
 3100 IF P=1 THEN LET SD=S(NX,NY,NZ,1)+2
*S(NX,NY,NZ,2)
 3110 IF P=2 AND S(NX,NY,NZ,1)+2*S(NX,NY
,NZ,2)<=SD THEN LET SD=-1
 3120 NEXT P
 3130 IF SD=-1 THEN GOSUB 3200
 3140 NEXT F
 3150 RETURN
 3199 :
 3200 LET DX=SGN(NX-X)
 3210 LET DY=SGN(NY-Y)
 3220 LET DZ=SGN(NZ-Z)
 3250 LET E(X+DX,Y+DY,Z+DZ,1)=E(X+DX,Y+D
Y,Z+DZ,1)+E(X,Y,Z,1)
 3260 LET E(X+DX,Y+DY,Z+DZ,2)=E(X+DX,Y+D
Y,Z+DZ,2)+E(X,Y,Z,2)
 3270 LET E(X,Y,Z,1)=0
 3280 LET E(X,Y,Z,2)=0
 3290 LET F(F,1)=X+DX
 3300 LET F(F,2)=Y+DY
 3310 LET F(F,3)=Z+DZ
 3320 RETURN
```

```
 3497 :
 3498 :
 3499 REM BATTLES & COUNTING
 3500 FOR X=1 TO 6
 3510 FOR Y=1 TO 6
 3520 FOR Z=1 TO 6
 3530 IF S(X,Y,Z,1)+S(X,Y,Z,2)>0 AND E(X
,Y,Z,1)+E(X,Y,Z,2)>0 THEN GOSUB 3650
 3540 LET S=S+S(X,Y,Z,1)+S(X,Y,Z,2)
 3550 LET ES=ES+E(X,Y,Z,1)+E(X,Y,Z,2)
 3560 NEXT Z
 3570 NEXT Y
 3580 NEXT X
 3590 RETURN
 3649 :
 3650 LET S1=S(X,Y,Z,1)
 3660 LET S2=S(X,Y,Z,2)
 3670 LET E1=E(X,Y,Z,1)
 3680 LET E2=E(X,Y,Z,2)
 3690 LET SL1=INT((E1/2+E2)*(1+RND(1)/4)
)
 3700 LET SL2=INT((E2/4+E2/2)*(1+RND(1)/
4))
 3710 LET EL1=INT((S1/2+S2)*(1+RND(1)/4)
)
 3720 LET EL2=INT((S2/4+S2/2)*(1+RND(1)/
4))
 3730 IF SL1>S1 THEN LET SL1=S1
 3740 IF SL2>S2 THEN LET SL2=S2
 3750 IF EL1>E1 THEN LET EL1=E1
 3760 IF EL2>E2 THEN LET EL2=E2
 3770 LET S1=S1-SL1
 3780 LET S2=S2-SL2
 3790 LET E1=E1-EL1
 3800 LET E2=E2-EL2
 3810 LET S(X,Y,Z,1)=S1
 3820 LET S(X,Y,Z,2)=S2
 3830 LET E(X,Y,Z,1)=E1
 3840 LET E(X,Y,Z,2)=E2
 3850 PRINT"Engagement at (";X;",";Y;","
;Z;"):"
```

```
 3860 PRINT
 3870 PRINT"We lost ";SL1;" type 1 ships
."
 3880 PRINT"We have ";S1;" left."
 3890 PRINT"We lost ";SL2;" type 2 ships
."
 3900 PRINT"We have ";S2;" left."
 3910 PRINT
 3920 PRINT"The enemy lost ";EL1;" type
1 ships."
 3930 PRINT"They have ";E1;" left."
 3940 PRINT"The enemy lost ";EL2;" type
2 ships."
 3950 PRINT"They have ";E2;" left."
 3960 GOSUB 9900
 3970 RETURN
 3997 :
 3998 :
 3999 REM ENEMY SURRENDER
 4000 PRINT"Prince Orion has announced h
e is"
 4010 PRINT"surrendering. He had only ";
ES;" ships"
 4020 PRINT"left, and did not want to wa
ste his"
 4030 PRINT"few remaining sailors."
 4040 PRINT"Congratulations, you have wo
n, on"
 4050 PRINT"Standard Day ";T;" of the en
counter."
 4060 GOTO 7000
 4497 :
 4498 :
 4499 REM WITHDRAW ?
 4500 PRINT"You have only ";S;" ships le
ft, and the"
 4510 PRINT"Prince still has ";ES;"."
 4520 PRINT"Do you want to withdraw ?"
 4530 INPUT A$
 4540 CLS
 4550 IF A$<>"Y" THEN GOTO 130
```

```
 4560 PRINT"The encounter lasted ";T;" S
tandard"
 4570 PRINT"Days. Never mind, he who fig
hts and"
 4580 PRINT"runs away..."
 4590 GOTO 7000
 5997 :
 5998 :
 5999 REM LOSE
 6000 PRINT"Your forces have been wiped
out by"
 6010 PRINT"Prince Orion."
 6020 PRINT"You survived for ";T;" Stand
ard Days."
 6030 GOTO 7000
 6497 :
 6498 :
 6499 REM WIN
 6500 PRINT"You have decimated Prince Or
ion's"
 6510 PRINT"fleet, leaving ";S;" of your
 own ships."
 6520 PRINT"The Prince managed to surviv
e for only"
 6530 PRINT" ";T;" Standard Days. Well done!"
 6540 GOTO 7000
 6997 :
 6998 :
 6999 REM REPLAY
 7000 PRINT
 7010 PRINT"Do you want to try again ?"
 7020 INPUT A$
 7030 IF A$="Y" THEN RUN
 7040 END
 9898 :
 9899 REM KEY PRESS ROUTINE
 9900 PRINT
 9910 PRINT"PRESS ANY KEY TO CONTINUE"
 9920 REPEAT UNTIL GET$<>""
 9930 CLS
 9940 RETURN
```

Galactic Empire

The object of this game is to build up an empire by taking over regions occupied by another empire. You do this by a combination of tact and military strength – if you build up too many ships in a region, your popularity falls there.

The menu method of display and input control is very similar to that in "Third World War". Enemy equipment movement is also worked out in a similar way – by finding the regions showing the greatest advance or retreat and pouring ships in there.

As usual for the numbers of ships and regions involved, arrays are used for storage, though DATA statements are unnecessary for holding names since numbers for regions are sufficient, and there is only one type of ship.

```
   5 REM MASTER ROUTINE
  10 CLS
  20 GOSUB 1000:REM INITIALISE
  30 GOSUB 1500:REM INTRO
  40 LET Y=1
  50 GOSUB 2000:REM MENU
  60 GOSUB 3500:REM ENEMY DECISIONS
  70 FOR X=1 TO 4
  80 GOSUB 4000:REM BATTLES
  90 NEXT X
 100 FOR X=1 TO 4
 110 IF F(X)=0 THEN GOTO 6000:REM LOSE
 120 IF F(X)=10 THEN GOTO 6500:REM WIN
 130 NEXT X
 140 GOSUB 5500:REM ALTER VARIABLES
 150 LET Y=Y+1
 160 GOTO 50
 997 :
 998 :
 999 REM INITIALISE
1000 DIM T(5):DIM D(5)
1010 DIM P(4,2):DIM F(4)
1020 DIM E(5)
1030 FOR X=1 TO 4
```

```
 1040 LET F(X)=4+SGN(RND(1)-RND(1))
 1050 LET D(X)=INT(RND(1)*100)
 1060 LET E(X)=INT(RND(1)*70)
 1070 FOR Y=1 TO 2
 1080 LET P(X,Y)=RND(1)-RND(1)
 1090 NEXT Y
 1100 NEXT X
 1110 LET D(5)=1000:LET E(5)=800
 1120 RETURN
 1497 :
 1498 :
 1499 REM INTRO
 1500 PRINT"GALACTIC EMPIRE"
 1510 PRINT
 1520 PRINT"You are the general of the s
pace-fleet"
 1530 PRINT"of an expanding galactic emp
ire. It"
 1540 PRINT"currently occupies 40% of th
e galaxy."
 1550 PRINT"Using military strategy and
political"
 1560 PRINT"acumen, you must beat back t
he forces"
 1570 PRINT"of another, larger empire wh
ile taking"
 1580 PRINT"care not to alarm planets of
 the"
 1590 PRINT"frontier regions of your mil
itaristic"
 1600 PRINT"intentions."
 1610 PRINT
 1620 PRINT"When a territory is occupied
, be sure"
 1630 PRINT"to allocate enough ships to
guard"
 1640 PRINT"against rebellion from natio
nalist"
 1650 PRINT"elements among the populace.
"
 1660 GOSUB 9000
```

```
 1670 RETURN
 1997 :
 1998 :
 1999 REM MENU
 2000 CLS
 2010 PRINT"Here are your options:"
 2020 PRINT"1. Do nothing."
 2030 PRINT"2. Display situation on one
front."
 2040 PRINT"3. Transfer ships."
 2050 PRINT"Which one do you want ?"
 2060 INPUT A
 2070 LET A=INT(A)
 2080 IF A>3 THEN LET A=3
 2090 IF A<=1 THEN RETURN
 2100 IF A=2 THEN PRINT"W";
 2120 IF A=3 THEN PRINT"To w";
 2130 PRINT"hich front ? (1 to 4 or 5 fo
r"
 2140 PRINT"reserves)"
 2150 INPUT B
 2160 LET B=ABS(INT(B))
 2170 IF B>5 OR B<1 THEN LET B=1
 2180 IF A=2 THEN GOSUB 2500
 2190 IF A=3 THEN GOSUB 3000
 2200 GOTO 2000
 2498 :
 2499 REM DISPLAY SITUATIONS
 2500 CLS
 2510 IF B<5 THEN PRINT"Front ";B
 2520 IF B=5 THEN PRINT"Reserves"
 2530 PRINT
 2540 PRINT"We have ";D(B);" ships here.
"
 2550 PRINT"The enemy is thought to have
 about"
 2560 PRINT" ";INT(ABS(E(B)+(RND(1)-RND(
1))/5*E(B)));" ships."
 2570 IF B=5 THEN GOTO 2710
 2580 PRINT
 2590 PRINT"The front is ";F(B);" region
```

```
s from"
 2600 PRINT"our home planet."
 2610 PRINT
 2620 FOR X=1 TO 2
 2630 PRINT"The population on ";
 2640 IF X=1 THEN PRINT"our";
 2650 IF X=2 THEN PRINT"the enemy";
 2660 PRINT" side is"
 2670 IF P(B,X)<.4 THEN PRINT"rebellious
."
 2680 IF P(B,X)>=.4 AND P(B,X)<=.7 THEN
PRINT"fairly calm."
 2690 IF P(B,X)>.7 THEN PRINT"content."
 2700 NEXT X
 2710 GOSUB 9000
 2720 RETURN
 2998 :
 2999 REM TRANSFER
 3000 PRINT
 3010 PRINT"From which front ?"
 3020 INPUT C
 3030 LET C=ABS(INT(C))
 3040 IF C>5 OR C<1 THEN LET C=5
 3050 PRINT
 3060 PRINT"How many ships ?"
 3070 INPUT D
 3080 LET D=ABS(INT(D))
 3090 IF D>D(C) THEN LET D=D(C)
 3100 LET D(C)=D(C)-D
 3110 LET T(B)=T(B)+D
 3120 RETURN
 3497 :
 3498 :
 3499 REM ENEMY DECISIONS
 3500 LET TD=0
 3510 FOR X=1 TO 4
 3520 LET TD=TD+ABS(5-F(X))
 3530 LET E(5)=E(5)+INT(E(X)/2)
 3540 LET E(X)=INT(E(X)/2)
 3550 NEXT X
 3560 FOR X=1 TO 4
```

```
 3570 LET E(X)=E(X)+INT(ABS(5-F(X))/(TD+
1)*E(5))
 3580 NEXT X
 3590 LET E(5)=0
 3600 RETURN
 3997 :
 3998 :
 3999 REM BATTLES
 4000 LET SL=ABS(INT(E(X)/5+(RND(1)-RND(
1))*20))
 4010 LET TL=ABS(INT(D(X)/5+(RND(1)-RND(
1))*20))
 4020 IF SL>D(X) THEN LET SL=D(X)
 4030 IF TL>E(X) THEN LET TL=E(X)
 4040 LET D(X)=D(X)-SL
 4050 LET E(X)=E(X)-TL
 4060 LET NP=F(X)+SGN(INT((D(X)-E(X))/25
))
 4070 IF NP>10 THEN LET NP=10
 4080 IF NP<0 THEN LET NP=0
 4090 CLS
 4100 PRINT"Report from front ";X
 4110 PRINT
 4120 PRINT"We lost ";SL;" ships."
 4130 PRINT"The enemy lost ";TL;" ships.
"
 4140 PRINT
 4150 IF NP>F(X) THEN PRINT"We have take
n the region."
 4160 IF NP<F(X) THEN PRINT"We have lost
 the region."
 4170 IF F(X)<NP THEN GOSUB 4300
 4180 LET F(X)=NP
 4190 PRINT"The front is now ";F(X);" re
gions from"
 4200 PRINT"our home planet."
 4210 LET P(X,1)=P(X,1)+(RND(1)-RND(1))/
5
 4220 LET P(X,2)=P(X,2)+(RND(1)-RND(1))/
5
 4230 IF P(X,2)>.3 AND F(X)<10 THEN GOSU
```

```
B 4500
 4240 IF P(X,2)<-.7 AND F(X)>0 THEN GOSU
B 5000
 4250 GOSUB 9000
 4260 RETURN
 4298 :
 4299 REM OCCUPYING FORCE
 4300 PRINT
 4310 PRINT"How many ships will occupy t
he region ?"
 4320 INPUT A
 4330 LET A=INT(ABS(A))
 4340 IF A>D(X) THEN LET A=D(X)
 4350 LET D(X)=D(X)-A
 4360 RETURN
 4498 :
 4499 REM REGION JOINS YOU
 4500 PRINT
 4510 PRINT"The next region outward has
rebelled"
 4520 PRINT"against the enemy."
 4530 IF D(X)<E(X) THEN PRINT"The rebell
ion was crushed.":RETURN
 4540 PRINT"The system has thrown out th
e enemy and"
 4550 PRINT"joined you."
 4560 LET F(X)=F(X)+1
 4570 LET P(X,2)=P(X,2)+(RND(1)-RND(1))/
3
 4580  LET P(X,1)=P(X,2)
 4590 RETURN
 4998 :
 4999 REM REGION JOINS ENEMY
 5000 LET F(X)=F(X)-1
 5010 LET P(X,2)=P(X,1)
 5020 PRINT
 5030 PRINT"The next region inward has r
ebelled and"
 5040 PRINT"joined the enemy."
 5050 RETURN
 5497 :
```

```
 5498 :
 5499 REM ALTER VARIABLES
 5500 FOR X=1 TO 5
 5510 LET D(X)=D(X)+T(X)
 5520 LET T(X)=0
 5530 NEXT X
 5540 FOR X=1 TO 4
 5550 LET P(X,1)=P(X,1)+SGN(D(X)-E(X))/5
 5560 LET P(X,2)=P(X,2)+SGN(E(X)-D(X))/5
 5570 NEXT X
 5580 LET NS=INT(RND(1)*500)
 5590 LET NT=INT(RND(1)*400)
 5600 PRINT"The factories have delivered
 ";NS;" new"
 5610 PRINT"ships to our reserves. The e
nemy have"
 5620 PRINT"received ";NT;" according to
 our spies."
 5630 LET D(5)=D(5)+NS
 5640 LET E(5)=E(5)+NT
 5650 GOSUB 9000
 5660 RETURN
 5997 :
 5998 :
 5999 REM LOSS
 6000 PRINT
 6010 PRINT"The enemy fleet from front "
;X;" has"
 6020 PRINT"reached our home planet and
taken it in"
 6030 PRINT"Galactic Standard Year ";Y;"
."
 6040 GOTO 7000
 6497 :
 6498 :
 6499 REM WIN
 6500 PRINT
 6510 PRINT"The fleet from front ";X;" h
as taken the"
 6520 PRINT"enemy's home planet in Galac
tic"
```

```
6530 PRINT"Standard Year ";Y;". Nothing
now"
6540 PRINT"prevents you from taking the
rest of"
6550 PRINT"the galaxy."
6560 GOTO 7000
6997 :
6998 :
6999 REM REPLAY
7000 PRINT
7010 PRINT"Do you want another game ?"
7020 INPUT A$
7030 IF A$="Y" THEN RUN
7040 END
8998 :
8999 REM KEY PRESS ROUTINE
9000 PRINT
9010 PRINT"PRESS ANY KEY TO CONTINUE"
9020 REPEAT UNTIL GET$<>""
9030 CLS
9040 RETURN
```

Nuclear Crisis

This is a fairly sophisticated representation of a nuclear blackmail confrontation, with a variety of weaponry and defences. This is responsible for the length of the program.

The object of the game is to persuade the other countries to give you their land. To speed this process up, you can destroy some land with nuclear devices if you wish.

The other countries analyse the situation with respect to numbers of equipment, and make demands or escalate accordingly. They also escalate in response to you. Where you have the edge over them is in buying equipment – they do not save money, with the result that they cannot afford the more expensive types after the first loop.

The main display uses the menu technique, so you can select reports on any of the countries. Then you make decisions regarding each country in turn. Your options in the decisions are generated using the present stage of relations with the country – the state accesses a set of options from the subroutines between lines 329Ø and 373Ø.

The initial values for the numbers of equipment are produced using basic quantities from the DATA statement at line 165Ø and a randomising factor to vary the figures slightly.

The reply move generation is a prime candidate for experimentation if you do not agree with the behaviour programmed in at present.

```
  5 REM MASTER ROUTINE
 10 CLS
 20 GOSUB 1000:REM INTRO
 30 GOSUB 1500:REM INITIALISE
 40 W=1
 50 GOSUB 2000:REM REPORT
 60 GOSUB 2500:REM ACQUIRE
 70 GOSUB 3000:REM ESCALATE?
 80 FOR C=2 TO 3
```

```
   90 IF C(C,1)>0 THEN GOSUB 4500:REM RE
PLIES
  100 NEXT C
  110 FOR X=1 TO 3
  120 IF C(X,2)<1E6 THEN C(X,1)=0
  130 NEXT X
  140 IF C(1,1)=0 THEN GOTO 8000
  150 IF C(2,1)=0 AND C(3,1)=0 THEN GOTO
 8500
  160 GOSUB 5000:REM DUD EQUIPMENT
  170 W=W+1
  180 GOTO 50
  997
  998
  999 REM INTRO
 1000 PRINT"NUCLEAR CRISIS"
 1010 PRINT
 1020 PRINT"It is the year 1992. The sup
erpowers"
 1030 PRINT"are selling nuclear weaponry
 to the old"
 1040 PRINT"Third World countries, as th
ey them-"
 1050 PRINT"selves are defended by satel
lite"
 1060 PRINT"weapons."
 1070 PRINT
 1080 PRINT"You are the leader of one su
ch country,"
 1090 PRINT"striving to protect yourself
 and"
 1100 PRINT"expand your territories. Aga
inst you"
 1110 PRINT"are two similar countries, w
ith the"
 1120 PRINT"same objectives - and means.
"
 1130 GOSUB 9900
 1140 PRINT"You buy in nuclear weaponry
to replace"
 1150 PRINT"broken (and used) equipment.
```

```
 The money"
 1160 PRINT"comes from your own country
as taxes,"
 1170 PRINT"and from the other countries
, in"
 1180 PRINT"blackmail payments of land."
 1190 PRINT
 1200 PRINT"As the amount of tax you can
 collect"
 1210 PRINT"depends on the size of your
country,"
 1220 PRINT"expansion increases your pot
entials."
 1230 PRINT
 1240 PRINT"You can also buy anti-missil
e missiles"
 1250 PRINT"against missiles, anti-aircr
aft"
 1260 PRINT"missiles against bombers, an
d"
 1270 PRINT"interceptors against Cruise
missiles."
 1280 GOSUB 9900
 1290 PRINT"Blackmailing is carried out
as follows:"
 1300 PRINT"First, you claim land as rig
htfully"
 1310 PRINT"yours. After that, if the vi
ctim"
 1320 PRINT"country does not agree, you
threaten"
 1330 PRINT"with nuclear attack. If the
country"
 1340 PRINT"still does not agree, you ca
n launch"
 1350 PRINT"an attack with one weapon. I
f this"
 1360 PRINT"fails, you can launch an all
-out"
 1370 PRINT"attack."
 1380 GOSUB 9900
```

```
1390 RETURN
1497
1498
1499 REM INITIALISE
1500 DIM C(3,10)
1510 FOR X=1 TO 3
1520 RESTORE 1650
1530 C(X,1)=1
1540 FOR Y=2 TO 10
1550 READ I
1560 C(X,Y)=INT(I+I/5*(RND(1)-RND(1)))
1570 NEXT Y
1580 NEXT X
1590 DIM T(3,6)
1600 DIM S(2)
1610 DIM F(3)
1620 MI=0:MO=0
1630 RETURN
1649
1650 DATA 1E7,1E9,10,15,5,15,5,10,0
1997
1998
1999 REM MAIN DISPLAY
2000 CLS
2010 PRINT"Week ";W
2020 PRINT
2030 PRINT"Which do you want ?"
2040 PRINT" 1. Report on your country."
2050 PRINT" 2. Report on Eprabia."
2060 PRINT" 3. Report on Faulenland."
2070 PRINT" 4. Leave report stage."
2080 INPUT A
2090 A=INT(A)
2100 CLS
2110 IF A<1 OR A=4 THEN RETURN
2120 PRINT"Report on ";
2130 RESTORE 2300
2140 FOR X=1 TO A
2150 READ C$
2160 NEXT X
2170 PRINT C$;":"
```

```
 2180 PRINT
 2190 IF C(A,1)=0 THEN PRINT"This countr
y no longer exists.":GOTO 2280
 2200 RESTORE 2310
 2210 FOR X=2 TO 9
 2220 IF A>1 AND X>2 THEN PRINT"Est. ";
 2230 READ I$
 2240 N=C(A,X)
 2250 IF A>1 AND X>2 THEN N=ABS(INT(N+N/
3*(RND(1)-RND(1))))
 2260 PRINT I$;TAB(25);:IF X>3 THEN PRIN
T;N
 2265 IF X<4 THEN PRINT;INT(N/1E6);"m"
 2270 NEXT X
 2280 GOSUB 9900
 2290 GOTO 2000
 2299
 2300 DATA your country,Eprabia,Faulenla
nd
 2310 DATA Land area (sq. km),Money ($)
 2320 DATA B1 bombers
 2330 DATA Cruise missiles
 2340 DATA Pershings
 2350 DATA A.A.M.s
 2360 DATA Interceptors
 2370 DATA A.M.M.s
 2497
 2498
 2499 REM ACQUIRE EQUIPMENT
 2500 IF C(1,3)<2E6 THEN RETURN
 2510 CLS
 2520 PRINT"Which do you want ?"
 2530 PRINT" 1. Leave acquiral stage.".
 2540 RESTORE 2320
 2550 FOR X=2 TO 7
 2560 PRINT" ";X;". Buy ";
 2570 READ I$
 2580 PRINT I$;"."
 2590 NEXT X
 2600 INPUT A
 2610 A=INT(ABS(A))
```

```
 2620 IF A<2 OR A>7 THEN RETURN
 2630 RESTORE 2800
 2640 FOR X=2 TO A
 2650 READ I
 2660 NEXT X
 2670 PRINT"How many do you want (@ $";I
;"m) ?"
 2680 INPUT B
 2690 B=INT(ABS(B))
 2700 IF C(1,3)<B*I*1E6 THEN B=INT(C(1,3
)/I/1E6)
 2710 PRINT"The suppliers will deliver "
;B;"."
 2720 T(1,A-1)=B
 2730 C(1,3)=C(1,3)-B*I*1E6
 2740 GOSUB 9900
 2750 GOTO 2500
 2799
 2800 DATA 100,15,65,2,25,10
 2997
 2998
 2999 REM ESCALATE?
 3000 FOR C=2 TO 3
 3010 RESTORE 2300
 3020 FOR Z=1 TO C
 3030 READ C$:NEXT Z
 3040 IF C(C,1)=0 THEN GOTO 3345
 3050 CLS
 3060 PRINT"Current situation with ";C$;
":"
 3070 PRINT
 3080 S=S(C-1):P$="We"
 3090 FOR X=1 TO 2
 3100 IF S=0 AND X=2 THEN PRINT"They are
 at peace with us."
 3110 IF S=1 THEN PRINT P$;" have offici
ally claimed some land."
 3120 IF S=2 THEN PRINT P$;" threatened
nuclear war."
 3140 IF S=3 THEN PRINT P$;" made an att
ack with one weapon."
```

```
 3160 IF S=4 THEN PRINT P$;" started an
all-out nuclear war."
 3170 S=C(C,10):P$="They"
 3180 PRINT
 3190 NEXT X
 3200 PRINT
 3210 S=S(C-1)
 3220 FOR Q=0 TO 1
 3230 IF Q=1 AND S=0 THEN GOTO 3340
 3240 GOSUB 9900
 3250 PRINT"Action on ";C$;":"
 3260 IF Q=0 THEN PRINT"(Your claims)"
 3270 IF Q=1 THEN PRINT"(";C$;"'s claims
)"
 3280 PRINT
 3290 PRINT"Here are your options:"
 3300 PRINT" 1. Do nothing."
 3310 GOSUB (3350+400*Q+100*S)
 3320 IF S>2 THEN GOSUB (4000+S*500)
 3330 IF Q=0 THEN S(C-1)=S
 3340 S=C(C,10):NEXT Q
 3345 NEXT C:RETURN
 3350 PRINT" 2. Claim some land."
 3360 PRINT" 3. Launch a nuclear attack.
"
 3370 INPUT A
 3380 IF A=2 THEN S=1
 3390 IF A=3 THEN S=4
 3400 RETURN
 3449
 3450 PRINT" 2. Withdraw the claim."
 3460 PRINT" 3. Threaten nuclear war."
 3470 PRINT" 4. Launch a nuclear attack.
"
 3480 INPUT A
 3490 IF A=2 THEN S=0
 3500 IF A=3 THEN S=2
 3510 IF A=4 THEN S=4
 3520 RETURN
 3549
 3550 PRINT" 2. Withdraw the claim."
```

```
3560 PRINT" 3. Launch a single weapon."
3570 PRINT" 4. Launch a full attack."
3580 INPUT A
3590 IF A=2 THEN S=0
3600 IF A=3 OR A=4 THEN S=A
3610 IF A=4 THEN S=4
3620 RETURN
3649
3650 PRINT" 2. Withdraw the claim."
3660 PRINT" 3. Launch a single weapon."
3670 PRINT" 4. Launch a full attack."
3680 INPUT A
3690 IF A=2 THEN S=0
3700 IF A=4 THEN S=4
3710 IF A=1 THEN S=2
3720 IF A=3 THEN S=3
3730 RETURN
3749
3750 PRINT" 2. Withdraw the claim."
3760 PRINT"Doing nothing is the same as
"
3770 PRINT"continuing the attack."
3780 INPUT A
3790 IF A=2 THEN S=0
3800 RETURN
3849
3850 PRINT" 2. Accept the claim."
3860 INPUT A
3870 IF A=2 THEN GOSUB 4250
3880 RETURN
3949
3950 PRINT" 2. Accept the claim."
3960 INPUT A
3970 IF A=2 THEN GOSUB 4250
3980 RETURN
4049
4050 PRINT" 2. Accept the claim."
4060 PRINT" 3. Launch a weapon in retal
iation."
4070 PRINT" 4. Launch a full attack."
4080 INPUT A
```

```
 4090 IF A=3 OR A=4 THEN S=A
 4100 IF A=2 THEN GOSUB 4250
 4110 IF A=1 THEN S=0
 4120 RETURN
 4149
 4150 PRINT" 2. Accept the claim."
 4160 PRINT" 3. Launch a full attack."
 4170 INPUT A
 4180 IF A=3 THEN S=4
 4190 IF A=2 THEN GOSUB 4250
 4200 IF A=1 THEN S=0
 4210 RETURN
 4249
 4250 CLS
 4260 PRINT"How many sq. km of land will
 you give"
 4270 PRINT"them ?"
 4280 INPUT A
 4290 A=ABS(INT(A))
 4300 IF A>C(1,2) THEN A=C(1,2)
 4310 IF A>=RND(1)^2*C(1,2)^2/C(C,2) THE
N GOTO 4400
 4320 PRINT"It was not enough. Will you
reconsider ?"
 4330 INPUT A$
 4340 CLS
 4350 IF A$="Y" THEN GOTO 4250
 4360 RETURN
 4399
 4400 PRINT"It was enough to satisfy the
m."
 4410 C(C,10)=0
 4420 S(C-1)=0
 4430 C(1,2)=C(1,2)-A
 4440 C(C,2)=C(C,2)+A
 4450 GOSUB 9900
 4460 RETURN
 4497
 4498
 4499 REM REPLIES
 4500 MI=0:MO=0
```

```
 4510 FOR X=3 TO 6
 4520 MI=MI+C(1,X)-.7*C(C,X+3)
 4530 MO=MO+C(C,X)-.8*C(1,X+3)
 4540 T(C,X)=C(1,X)-C(C,X+3)
 4550 IF T(C,X)<0 THEN T(C,X)=0
 4560 NEXT X
 4565 IF MI<0 THEN MI=0
 4570 FOR X=1 TO 3
 4580 T(C,X)=C(C,X+3)-C(1,X+6)+INT(RND(1
)*5)
 4590 IF T(C,X)<0 THEN T(C,X)=0
 4600 NEXT X
 4610 RESTORE 2800
 4620 M=0
 4630 FOR X=4 TO 9
 4640 READ I
 4650 M=M+T(C,X-3)*I*1E6
 4660 NEXT X
 4670 M=C(C,3)/(M+1):C(C,3)=0
 4680 FOR X=1 TO 6
 4690 T(C,X)=ABS(INT(T(C,X)*M))
 4700 NEXT X
 4710 IF S(C-1)>1 AND MI>10 THEN GOTO 48
00
 4720 IF S(C-1)>2 THEN C(C,10)=S(C-1)
 4730 IF S(C-1)<4 AND MO>MI*(1.5+RND(1))
 AND C(C,10)<4 THEN C(C,10)=C(C,10)+1
 4740 IF C(C,2)<RND(1)*8E6 THEN C(C,10)=
1:GOSUB 6750
 4750 IF C(C,10)>2 THEN GOSUB (5000+500*
C(C,10))
 4760 IF MI>10 AND MO<MI AND S(C-1)<2 AN
D C(C,10)>2 THEN C(C,10)=C(C,10)-1
 4770 RETURN
 4799
 4800 CLS
 4810 RESTORE 2300
 4820 FOR X=1 TO C
 4830 READ I$
 4840 NEXT X
 4850 PRINT I$;" has decided to give you
```

```
 some"
 4860 PRINT"land if you will cease the a
ttack."
 4870 L=INT(RND(1)^3*C(C,2))
 4880 PRINT"They will give you ";L;" sq.
 km."
 4890 PRINT"Is this enough ?"
 4900 INPUT A$
 4910 IF A$="Y" THEN GOTO 4950
 4920 IF RND(1)>.5 THEN GOTO 4870
 4930 RETURN
 4949
 4950 S(C-1)=0
 4960 C(C,10)=0
 4970 C(1,2)=C(1,2)+L
 4980 C(C,2)=C(C,2)-L
 4990 RETURN
 4997
 4998
 4999 REM DUD EQUIPMENT, ETC
 5000 CLS
 5010 PRINT"Equipment gone unserviceable
:"
 5020 RESTORE 2320
 5030 FOR X=4 TO 9
 5040 D=INT(RND(1)*C(1,X)/5)
 5050 READ I$
 5060 PRINT I$;TAB(20);D
 5070 C(1,X)=C(1,X)-D
 5080 NEXT X
 5090 FOR C=2 TO 3
 5100 FOR X=4 TO 9
 5110 C(C,X)=C(C,X)-INT(RND(1)*C(C,X)/5)
 5120 NEXT X
 5130 NEXT C
 5140 PRINT
 5150 FOR C=1 TO 3
 5160 T=C(C,2)*2
 5170 IF C=1 THEN PRINT"Tax collected: $
";INT(T/1E6);"m"
 5180 C(C,3)=C(C,3)+T
```

```
 5190 FOR X=1 TO 6
 5200 C(C,X+3)=C(C,X+3)+T(C,X):T(C,X)=0
 5210 NEXT X
 5220 NEXT C
 5230 GOSUB 9900
 5240 RETURN
 5497
 5498
 5499 REM SINGLE WEAPON
 5500 CLS
 5510  PRINT"You wanted to launch 1 weap
on."
 5520 PRINT"This is your arsenal:"
 5530 RESTORE 2320
 5540 FOR X=1 TO 3
 5550 READ I$
 5560  PRINT" ";X;". ";I$;TAB(22);C(1,X+
3)
 5570 NEXT X
 5580 PRINT"Which type will you use ?"
 5590 INPUT A
 5600 A=ABS(INT(A)):IF A>6 OR A<1 THEN A
=1
 5610 IF C(1,A+3)<1 THEN RETURN
 5620 C(1,A+3)=C(1,A+3)-1
 5630 F(A)=1
 5640 GOSUB 7000
 5650 RETURN
 5997
 5998
 5999 REM FULL ATTACK
 6000 RESTORE 2320
 6010 PRINT"You ordered a full attack."
 6020 PRINT"Enter the numbers of each ty
pe to use:"
 6030 FOR X=1 TO 3
 6040 READ I$
 6050 PRINT I$;TAB(20);"(";C(1,X+3);" th
ere)"
 6060 INPUT A
 6070 A=ABS(INT(A)):IF A<0 OR A>C(1,X+3)
```

```
 THEN A=C(1,X+3)
 6080 F(X)=A
 6090 C(1,X+3)=C(1,X+3)-A
 6100 NEXT X
 6110 GOSUB 7000
 6120 RETURN
 6497
 6498
 6499 REM ENEMY SINGLE WEAPON LAUNCH
 6500 N=INT(RND(1)*3+.9)
 6510 IF C(C,N+3)=0 AND RND(1)>.01 THEN
GOTO 6500
 6520 F(N)=1
 6530 IF C(C,N+3)>0 THEN C(C,N+3)=C(C,N+
3)-1
 6540 GOSUB 7600
 6550 RETURN
 6647
 6648
 6649 REM FULL ENEMY ATTACK
 6650 FOR X=1 TO 3
 6660 F(X)=INT(C(C,X+3)/2+.5)
 6670 C(C,X+3)=C(C,X+3)-F(X)
 6680 NEXT X
 6690 GOTO 7600
 6747
 6748
 6749 REM LAST DITCH ATTACK
 6750 FOR X=1 TO 3
 6760 F(X)=C(C,X+3)
 6770 C(C,X+3)=0
 6780 NEXT X
 6790 GOSUB 7600
 6800 RETURN
 6997
 6998
 6999 REM ATTACK
 7000 CLS
 7010 LL=0
 7020 RESTORE 2300
 7030 FOR Z=1 TO C
```

```
 7050 NEXT Z
 7060 PRINT"Target: ";C$;". You launched
:"
 7070 RESTORE 2320
 7080 FOR X=1 TO 3
 7090 READ I$
 7100 PRINT I$;":";TAB(20);F(X)
 7110 NEXT X
 7115 GOSUB 9900
 7120 RESTORE 2320
 7130 FOR X=1 TO 3
 7140 READ I$
 7150 IF F(X)=0 THEN GOTO 7240
 7160 N=0
 7170 FOR Y=1 TO F(X)
 7180 IF C(C,X+6)>0 AND RND(1)<.75 THEN
N=N+1:C(C,X+6)=C(C,X+6)-1
 7190 NEXT Y
 7200 PRINT I$;":"
 7205 PRINT"The enemy defences destroyed
 ";N;"."
 7210 F(X)=F(X)-N
 7220 PRINT" ";F(X);" reached the target
."
 7230 LL=LL+INT(RND(1)*F(X)*1E4)
 7240 NEXT X
 7250 IF LL>C(C,2) THEN LL=C(C,2)
 7260 C(C,2)=C(C,2)-LL
 7270 PRINT" ";LL;" sq. km of land were,
lost."
 7280 C(1,4)=C(1,4)+INT(RND(1)*F(1))
 7290 FOR X=1 TO 3
 7300 F(X)=0
 7310 NEXT X
 7320 GOSUB 9900
 7330 RETURN
 7597
 7598
 7599 REM ENEMY ATTACK
 7600 N=0:LL=0
 7610 RESTORE 2300
```

```
 7620 FOR Z=1 TO C
 7630 READ C$
 7640 NEXT Z
 7650 CLS
 7660 PRINT C$;" launched:"
 7670 RESTORE 2320
 7680 FOR X=1 TO 3
 7690 READ I$
 7700 PRINT I$;":";TAB(20);F(X)
 7710 NEXT X
 7720 RESTORE 2320
 7730 FOR X=1 TO 3
 7740 READ I$
 7750 IF F(X)=0 THEN GOTO 7850
 7760 N=0
 7770 FOR Y=1 TO F(X)
 7780 IF C(1,X+6)>1 AND RND(1)<.75 THEN
N=N+1:C(1,X+6)=C(1,X+6)-1
 7785 NEXT Y
 7790 PRINT I$;":"
 7800 PRINT"Our defences destroyed ";N;"
."
 7810 F(X)=F(X)-N
 7820 PRINT" ";F(X);" reached the target
."
 7830 LL=LL+INT(RND(1)*F(X)*1E4)
 7840 IF LL>C(1,2) THEN LL=C(1,2)
 7850 NEXT X
 7860 C(1,2)=C(1,2)-LL
 7870 IF C(1,2)<0 THEN C(1,2)=0
 7880 PRINT" ";LL;" sq. km of land were
lost."
 7890 C(C,4)=C(C,4)+INT(F(1)-RND(1)*F(1)
)
 7900 FOR X=1 TO 3
 7910 F(X)=0
 7920 NEXT X
 7930 GOSUB 9900
 7940 RETURN
 7997
 7998
```

```
 7999 REM LOSE
 8000 CLS
 8010 PRINT"Your country has been given
away or"
 8020 PRINT"blown out of existence after
 ";W
 8030 PRINT"weeks."
 8040 GOTO 9000
 8497
 8498
 8499 REM WIN
 8500 CLS
 8510 PRINT"Well done! You have succeede
d in"
 8520 PRINT"vanquishing the other two co
untries,"
 8530 PRINT"leaving yourself to continue
 expanding"
 8540 PRINT"throughout the continent."
 8550 GOTO 9000
 8997
 8998
 8999 REM REPLAY
 9000 PRINT
 9010 PRINT"Do you want to try again ?"
 9020 INPUT A$
 9030 IF A$="Y" THEN RUN
 9040 END
 9898
 9899 REM KEY PRESS ROUTINE
 9900 PRINT
 9910 PRINT"PRESS ANY KEY TO CONTINUE"
 9920 REPEAT UNTIL GET$<>""
 9930 CLS
 9940 RETURN
```

The Road to Valhalla

The behaviour of people in this game is based on the description of Viking (or Icelandic) custom described in "Njal's Saga". The object is to gain honour and wealth so that on your death, you go to Valhalla. Common-sense laws such as not harming your friends and finishing vendettas should point the way to success.

Reply moves consist of fairly random actions on the part of your rival settlers, which indicate your standing with them and start vendettas. Owing to the slightly random nature of the behaviour-generating lines, they tend to do unlikely things from time to time – such as giving you presents after you have attacked them. The response to your actions is linked to your popularity with each settler. The analysis of your actions is fairly complete – each settler considers what you have done in the light of their opinion of the person you have done it to, if it is not the same settler.

Numbered choices are used throughout for your decisions. An input language would have been more interesting, but it proved impossible to produce in a "universal" BASIC.

Mutual popularities are stored in the array A. Other arrays are used for your possessions, etc. (NB The array F$(4) contains four **strings,** not characters, so on some computers it may be necessary to identify the position of each character. For this reason, all the rival settlers have four-letter names, so the correction will be minor: DIM F$(4,4). This applies to the Sinclair ZX81 for one.

```
 10 REM MASTER ROUTINE
 20 GOSUB 1000:REM INITIALISATION
 30 CLS
 40 GOSUB 1500:REM INTRO
 50 GOSUB 9000
 60 LET YEAR=1:LET SE=1
 70 GOSUB 2000:REM DISPLAY A
 80 GOSUB 9000
 90 GOSUB 2500:REM INPUTS
100 GOSUB 9000
110 GOSUB 3000:REM ENEMY DECISIONS
120 GOSUB 9000
```

```
  130 IF SE=2 THEN GOSUB 4000:REM MOOT
  140 LET SE=SE+1
  150 IF SE=4 AND YEAR=L THEN GOTO 5000:
 REM END OF GAME
  160 IF SE=4 THEN GOSUB 9900
  170 GOTO 70
  997 :
  998 :
  999 REM INITIALISATION
 1000 DIM A(4,5):DIM F(4):DIM P(4):DIM F
$(4)
 1010 FOR X=1 TO 4
 1020 FOR Y=1 TO 5
 1030 LET A(X,Y)=(RND(1)-RND(1))/5
 1040 NEXT Y
 1050 LET P(X)=INT(RND(1)*50)
 1060 NEXT X
 1070 LET L=INT(RND(1)*6)+2
 1080 DIM N$(4):DIM P$(4)
 1090 RESTORE 9500
 1100 FOR X=1 TO 4
 1110 READ I$
 1120 READ J$
 1130 LET N$(X)=I$
 1140 LET P$(X)=J$
 1150 NEXT X
 1160 RETURN
 1497 :
 1498 :
 1499 REM INTRO
 1500 PRINT"THE ROAD TO VALHALLA"
 1510 PRINT
 1520 PRINT"You have emigrated from the
kingdom of"
 1540  PRINT"Norway to Iceland. Landing
at Helmsdale"
 1550 PRINT"you found a settlement in tr
ue Viking"
 1560 PRINT"fashion."
 1570 PRINT
 1580 PRINT"After the long, stormy voyag
```

```
e, you are"
 1590 IF L>6 THEN PRINT"still in the bes
t of health."
 1600 IF L<=6 AND L>5 THEN PRINT"in mode
rate health."
 1610 IF L<=5 AND L>4 THEN PRINT"rather
the worst for wear."
 1620 IF L<=4 THEN PRINT"in a poor state
 of health."
 1630 PRINT
 1640 PRINT"You know the time has come t
o think"
 1650 PRINT"seriously of the afterlife.
You must"
 1660 PRINT"increase your honour so you
can reach"
 1670 PRINT"Valhalla when the time comes
."
 1680 PRINT"You alter your honour in you
r dealings"
 1690 PRINT"with the other four clans of
"
 1700 PRINT"settlers, led by Thor, Odin,
 Olaf and"
 1710 PRINT"Lars. You also gain honour b
y"
 1720 PRINT"increaing your worldly wealt
h."
 1730 RETURN
 1997 :
 1998 :
 1999 REM SEASONAL DISPLAY
 2000 PRINT
 2010 IF SE=1 THEN PRINT"Spring";
 2020 IF SE=2 THEN PRINT"Summer";
 2030 IF SE=3 THEN PRINT"Autumn";
 2040 IF SE=4 THEN PRINT"Winter";
 2050 PRINT" of year ";YEAR
 2060 PRINT
 2070 PRINT"Your possessions are:"
 2080 FOR X=1 TO 4
```

```
 2090 PRINT P(X);" ";P$(X)
 2100 NEXT X
 2110 PRINT
 2120 LET N=0
 2130 FOR X=1 TO 4
 2140  IF A(X,1)>0 THEN GOSUB 5500
 2150 NEXT X
 2160 IF N>0 THEN GOSUB 5600
 2170 PRINT
 2180 RETURN
 2497 :
 2498 :
 2499 REM INPUTS
 2500 CLS
 2510 PRINT"These are your options:"
 2520 PRINT"1.Give away some of your pos
sessions"
 2530 PRINT"2.Do nothing"
 2540 PRINT"3.Steal somebody's possessio
ns"
 2550 PRINT"4.Kill somebody's possession
s"
 2560 PRINT"Which do you want?"
 2570 INPUT J
 2580 IF J<1 OR J>4 OR J<>INT(J) THEN GO
TO 2500
 2590 LET J=J-2
 2600 LET N=0
 2610 LET K=0
 2620 IF J<>0 THEN GOSUB 5800
 2630 LET F(N)=ABS(F(N)-J)
 2640 IF J=-1 THEN GOSUB 6000
 2650 IF J=1 THEN GOSUB 6100
 2660 IF J=2 THEN GOSUB 6200
 2670 FOR X=1 TO 4
 2680 IF X=N THEN GOTO 2710
 2690 LET A(X,1)=A(X,1)-J*K*A(X,N+1)/5
 2700 LET H=H-SGN(A(X,1)*J*K*A(X,N+1))/2
 2710 NEXT X
 2720 RETURN
 2997 :
```

```
 2998 :
 2999 REM ENEMY DECISIONS
 3000 FOR X=1 TO 4
 3010 LET S=INT(RND(1)*3)+1
 3020 GOSUB 6500
 3030 FOR Y=1 TO 4
 3040 IF T<>1 THEN LET A(T-1,S)=A(T-1,S)
-J*K*A(Y,T)
 3050 NEXT Y
 3060 IF T=1 THEN LET F(X)=ABS(F(X)+J)
 3070 NEXT X
 3080 LET S=INT(RND(1)*3)+1
 3090 GOSUB 7000
 3100 RETURN
 3997 :
 3998 :
 3999 REM ANNUAL MOOT
 4000 CLS
 4010 PRINT
 4020 PRINT"Do you want to go to the moo
t this year?"
 4030 INPUT A$
 4040 IF A$<>"Y" THEN GOSUB 9000
 4050 IF A$<>"Y" THEN RETURN
 4060 CLS
 4070 PRINT"At the moot,you can settle y
our"
 4080 PRINT"differences with any one cla
n."
 4090 PRINT
 4100 FOR X=1 TO 4
 4110 PRINT X;".";N$(X)
 4120 NEXT X
 4130 PRINT"Which one do you want?"
 4140 INPUT A
 4150 IF A<1 OR A>4 OR A<>INT(A) THEN GO
TO 4060
 4160 LET F(A)=0
 4170 LET A(A,1)=RND(1)*2
 4180 PRINT
 4190 PRINT"It is the end of the moot. T
```

```
he chief"
 4200 PRINT"of the elected council from
Rekjavik"
 4210 PRINT"approaches you and tells you
 that as"
 4220 PRINT"far as your honour is concer
ned, you"
 4230 PRINT"are doing ";
 4240 IF INT(H)>0 THEN PRINT"fine."
 4250 IF INT(H)=0 THEN PRINT"averagely w
ell."
 4260 IF INT(H)<0 THEN PRINT"badly."
 4270 GOSUB 9000
 4280 RETURN
 4997 :
 4998 :
 4999 REM END OF LIFE
 5000 FOR X=1 TO 4
 5010 LET H=H-ABS(F(X))+P(X)/15
 5020 NEXT X
 5030 PRINT
 5040 PRINT"You have reached the end of
your life."
 5050 LET D=1+RND(1)*2
 5060 IF H>D THEN PRINT"Valhalla welcome
s you."
 5070 IF H<=0 THEN PRINT"Long may you ro
t in hell!"
 5080 PRINT
 5090 PRINT"Do you wish to play again?"
 5100 INPUT A$
 5110 IF A$="Y" THEN RUN
 5120 END
 5497 :
 5498 :
 5499 REM FRIENDS
 5500 LET N=N+1
 5510 IF N=1 THEN LET F$=N$(X)
 5520 IF N=2 THEN LET F$=N$(X)+" and "+F
$
 5530 IF N>2 THEN LET F$=N$(X)+","+F$
```

```
 5540 RETURN
 5597 :
 5598 :
 5599 REM FRIEND DISPLAY
 5600 PRINT F$;
 5610 IF N=1 THEN PRINT" has";
 5620 IF N>1 THEN PRINT" have";
 5630 PRINT" sent good wishes."
 5640 RETURN
 5797 :
 5798 :
 5799 REM SPECIFIC INPUTS
 5800 FOR X=1 TO 4
 5810 PRINT X;".";N$(X)
 5820 NEXT X
 5830 PRINT"Which one do you want?"
 5840 INPUT N
 5850 IF N<1 OR N>4 OR N<>INT(N) THEN LE
T N=INT(RND(1)*3)+1
 5860 FOR X=1 TO 4
 5870 PRINT X;".";P$(X)
 5880 NEXT X
 5890 PRINT"Which type of possession?"
 5900 INPUT K
 5910 IF K<1 OR K>4 OR K<>INT(K) THEN LE
T K=INT(RND(1)*3)+1
 5920 RETURN
 5997 :
 5998 :
 5999 REM GIFT
 6000 LET NG=INT(RND(1)*P(K))
 6010 PRINT
 6020 PRINT N$(N);" thanks you for the "
;NG;" ";P$(K);"."
 6030 LET P(K)=P(K)-NG
 6040 IF A(N,1)<0 THEN LET H=H-1
 6050 IF A(N,1)>=0 THEN LET H=H+7.5
 6060 LET A(K,1)=A(K,1)+K
 6070 RETURN
 6097 :
 6098 :
```

```
 6099 REM THEFT
 6100 LET NS=INT(RND(1)*40)
 6110 PRINT
 6120 PRINT"We have stolen ";NS;" of ";N
$(N);"'s ";P$(K);"."
 6130 LET P(K)=P(K)+NS
 6140 IF A(N,1)>=0 THEN LET H=H-(A(N,1)+
.2)
 6150 IF A(N,1)<0 THEN LET H=H+1.5
 6160 LET A(N,1)=A(N,1)-1.5
 6170 RETURN
 6197 :
 6198 :
 6199 REM KILL
 6200 LET NK=INT(RND(1)*50)
 6210 PRINT
 6220 PRINT"We have killed ";NK;" of ";N
$(N);"'s ";P$(K);"."
 6230 IF A(N,1)>=0 THEN LET H=H-2*(A(N,1
)+.5)
 6240 IF A(N,1)<0 THEN LET H=H+2.5
 6250 LET A(N,1)=A(N,1)-3
 6260 RETURN
 6497 :
 6498 :
 6499 REM ENEMY ACTION
 6500 LET T=INT(RND(1)*3)+1
 6510 IF T=S+1 THEN GOTO 6500:REM REPEAT
 TILL DIFFERENT PERSON
 6520 LET D=A(S,T)+(RND(1)-RND(1))/10
 6530 IF D<.01 THEN LET J=1
 6540 IF D<-3 THEN LET J=2
 6550 IF D>=.01 THEN RETURN
 6560 IF J=1 THEN LET S$="stole"
 6570 IF J=2 THEN LET S$="killed"
 6580 LET K=1+INT(ABS(A(S,T)))
 6590 IF K>4 THEN LET K=4
 6600 IF T=1 THEN GOSUB 7400
 6610 IF T<>1 THEN GOSUB 7500
 6620 PRINT N$(S)+" "+S$+" "+STR$(Q)+" o
f "+W$+P$(K)+"."
```

```
 6630 RETURN
 6997 :
 6998 :
 6999 REM ENEMY GIFT
 7000 IF A(S,1)+(RND(1)-RND(1))/10<=0 TH
EN RETURN
 7010 LET Q=INT(RND(1)*50)+1
 7020 LET K=INT(RND(1)*3)+1
 7030 LET P(K)=P(K)+Q
 7040 PRINT N$(S);" sends you ";Q;" ";P$
(K);"."
 7050 LET F(S)=F(S)-1
 7060 RETURN
 7400 LET Q=INT(RND(1)*P(K))
 7410 LET P(K)=ABS(P(K)-Q)
 7420 LET W$="your "
 7430 RETURN
 7500 LET Q=INT(RND(1)*50)+1
 7510 LET W$=N$(T-1)+"'s "
 7520 RETURN
 8997 :
 8998 :
 8999 REM KEY PRESS ROUTINE
 9000 PRINT
 9010 PRINT"PRESS A KEY TO CONTINUE"
 9020 REPEAT UNTIL GET$<>""
 9030 CLS
 9040 RETURN
 9498 :
 9499 :
 9500 DATA Thor,sheep
 9510 DATA Odin,cattle
 9520 DATA Olaf,horses
 9530 DATA Lars,servants
 9898 :
 9899 REM CHANGE OF YEAR
 9900 LET SE=1
 9910 LET YEAR=YEAR+1
 9920 RETURN
```

Downing Street

This is not so much a political or economic acumen game as a test of general knowledge in politics. You choose your political party, and are expected to follow a stereotyped pattern of behaviour with regard to nationalisations (no complaints from party activists please). You make generalised decisions on the economy which affect your popularity in the different sections of the population. Your popularities here are indicated in the annual report at the start of each loop. Various international events including conferences and wars come up, which also make changes to your popularity.

At election time, the outcome is decided by your total popularity, related to the size of each section of the population. "Election results" are generated randomly (except for yours').

Most variables can be either 1 or Ø, which makes the program very simple from the mathematical point of view.

```
  5 REM MASTER ROUTINE
 10 CLS
 20 GOSUB 1000:REM INTRO
 30 GOSUB 1500:REM PARTY
 40 GOSUB 2000:REM INITIALISE
 50 LET Y=1
 60 GOSUB 2500:REM MAIN DISPLAY
 70 GOSUB 3000:REM DECISIONS
 80 FOR X=1 TO 3
 90 GOSUB 3500:REM CONFERENCES
100 NEXT X
110 IF WAR=0 THEN GOSUB 4000
120 IF WAR=1 THEN GOSUB 4500
130 GOSUB 5000:REM CHANGE VARIABLES
140 IF P1+P2+P3<5 THEN GOTO 5500
150 IF Y/5=INT(Y/5) THEN GOTO 6000
160 LET Y=Y+1
170 GOTO 60
399 REM START WAR?
997 :
998 :
```

```
  999 REM INTRO
 1000 PRINT"DOWNING STREET"
 1010 PRINT
 1020 PRINT"Your party, which you choose
 shortly,"
 1030 PRINT"has been elected to power in
 a country"
 1040 PRINT"very similar to Britain."
 1050 PRINT
 1060 PRINT"In this simulation, you make
 basic"
 1070 PRINT"policy decisions rather than
 numerical"
 1080 PRINT"analyses, to remain as popul
ar as"
 1090 PRINT"possible. You are only requi
red to give"
 1100 PRINT"the desired trend of taxes,
etc."
 1110 PRINT
 1120 PRINT"In your decisions, take acco
unt of your"
 1130 PRINT"backers in the different cla
sses in the"
 1140 PRINT"country. Their individual op
inions of"
 1150 PRINT"you can be gauged from the a
ppropriate"
 1160 PRINT"popularity-indicating activi
ties such"
 1170 PRINT"as strikes, lobbies and riot
s."
 1180 GOSUB 9900
 1190 RETURN
 1497 :
 1498 :
 1499 REM PARTY
 1500 PRINT"In choosing your political p
arty, you"
 1510 PRINT"must decide between Conserva
tive,"
```

```
 1520 PRINT"Liberal-SDP and Labour."
 1530 PRINT"Enter the name in full. (Blo
ck letters)"
 1540 INPUT P$
 1550 IF P$<>"CONSERVATIVE" AND P$<>"LIB
ERAL-SDP" AND P$<>"LABOUR" THEN GOTO 154
0
 1560 IF P$="CONSERVATIVE" THEN LET P=1
 1570 IF P$="LIBERAL-SDP" THEN LET P=2
 1580 IF P$="LABOUR" THEN LET P=3
 1590 CLS
 1600 IF P=1 THEN GOTO 1700
 1610 IF P=2 THEN GOTO 1800
 1620 IF P=3 THEN GOTO 1900
 1699 :
 1700 LET P1=20
 1710 LET P2=20+INT(RND(1)*10)
 1720 LET P3=5+INT(RND(1)*45)
 1730 RETURN
 1799 :
 1800 LET P1=10+INT(RND(1)*10)
 1810 LET P2=30
 1820 LET P3=15+INT(RND(1)*35)
 1830 RETURN
 1899 :
 1900 LET P1=INT(RND(1)*20)
 1910 LET P2=INT(RND(1)*30)
 1920 LET P3=40+INT(RND(1)*10)
 1930 RETURN
 1997 :
 1998 :
 1999 REM INITIALISE
 2000 DIM E(7)
 2010 DIM C(7)
 2020 FOR X=1 TO 7
 2030 IF RND(1)>.5 THEN LET C(X)=1
 2040 NEXT X
 2050 LET F=INT(RND(1)*15)
 2060 LET U=INT(RND(1)*5E6)
 2070 LET G=INT((RND(1)-RND(1))*15)
 2080 LET C=INT((RND(1)-RND(1))*15)
```

```
 2090 LET WAR=0
 2100 LET WS=0
 2110 RETURN
 2497 :
 2498 :
 2499 REM MAIN DISPLAY
 2500 CLS
 2510 PRINT"YEAR ";Y;" OF ";P$;" POWER"
 2520 PRINT
 2530 PRINT"National popularity:";P1+P2+
P3;"% support"
 2540 PRINT"Inflation rate ";F;"%"
 2550 PRINT"Unemployment: ";U;" people"
 2560 PRINT"Industrial growth: ";G;"%"
 2570 PRINT"Crime rate ";
 2580 IF C<0 THEN PRINT"down";
 2590 IF C>=0 THEN PRINT"up";
 2600 PRINT" by ";ABS(C);"%"
 2610 PRINT
 2620 IF P1<10 THEN PRINT"There is an in
crease in lobbying."
 2630 IF P2<20 THEN PRINT"White-collar w
orkers are striking more."
 2640 IF P3<40 THEN PRINT"There are anti
-policy demonstrations."
 2650 IF P3<30 THEN PRINT"Strikes of man
ual workers are on the    increase."
 2660 IF P3<5 THEN PRINT"Riots are takin
g place in the capital."
 2670 GOSUB 9900
 2680 RETURN
 2997 :
 2998 :
 2999 REM DECISIONS
 3000 RESTORE 3300
 3010 FOR X=1 TO 7
 3020 CLS
 3030 READ I$
 3040 PRINT"At present, ";I$;" is ";
 3050 IF E(X)=0 THEN PRINT"low."
 3060 IF E(X)=1 THEN PRINT"high."
```

```
 3070 PRINT"Do you want to change this t
rend ?"
 3080 INPUT A$
 3090 IF A$="Y" THEN LET E(X)=E(X)-1
 3100 IF E(X)<0 THEN LET E(X)=1
 3110 NEXT X
 3120 RESTORE 3350
 3130 FOR X=1 TO 7
 3140 CLS
 3150 READ I$
 3160 PRINT"The ";I$;" is "
 3170 IF C(X)=0 THEN PRINT"under private
 ownership."
 3180 IF C(X)=1 THEN PRINT"nationalised.
"
 3190 PRINT"Do you want to alter this ?"
 3200 INPUT A$
 3210 IF A$="Y" THEN LET C(X)=C(X)-1
 3220 IF C(X)<0 THEN LET C(X)=1
 3230 NEXT X
 3240 RETURN
 3299 :
 3300 DATA the Police Budget,the Defence
 Budget
 3310 DATA Tax,Industrial Subsidy
 3320 DATA the Health Budget,the Educati
on Budget
 3330 DATA the average salary
 3349 :
 3350 DATA Broadcasting Corporation,Nati
onal Railway
 3360 DATA Steel Corporation,National Ai
rline
 3370 DATA Coal Board,Electricity Genera
ting Board
 3380 DATA Shipbuilding Corporation
 3497 :
 3498 :
 3499 REM CONFERENCES
 3500 CLS
 3510 PRINT"Common Market Conference, Se
```

```
ssion ";X
 3520 PRINT
 3530 PRINT"The motion is that your coun
try should"
 3540 LET MOTION=3700+INT(RND(1)*5)*10
 3550 RESTORE MOTION
 3560 READ I$,I
 3570 PRINT I$;"."
 3580 PRINT"Will you accept ?"
 3590 INPUT A$
 3600 LET D=-I
 3610 IF A$="Y" THEN LET D=I
 3620 LET P1=INT(P1+RND(1)*D*15)
 3630 LET P2=INT(P2+RND(1)*D*15)
 3640 LET P3=INT(P3+RND(1)*D*15)
 3650 RETURN
 3699 :
 3700 DATA import more dairy produce,-1
 3710 DATA contribute more to the budget
,-1
 3720 DATA reduce the size of its steel
industry,-1
 3730 DATA receive more rebates,1
 3740 DATA receive an agricultural grant
,1
 3750 DATA supply the Market's armies,1
 3997 :
 3998 :
 4000 IF RND(1)>.15 THEN RETURN
 4010 CLS
 4020 PRINT"A Third-World country has at
tacked one"
 4030 PRINT"of your dependencies. Will y
ou launch"
 4040 PRINT"a counter-attack ?"
 4050 INPUT A$
 4060 IF A$="Y" THEN GOTO 4200
 4070 LET P1=INT(P1/2+RND(1)*P1/4)
 4080 LET P2=INT(P2/3+RND(1)*P2/3)
 4090 LET P3=INT(P3/4+RND(1)*P3/4)
 4100 RETURN
```

```
 4199 :
 4200 LET WAR=1
 4210 LET P1=INT(P1/2+RND(1)*P1/4)
 4220 LET P2=INT(P2/2+RND(1)*P2)
 4230 LET P3=INT(P3+RND(1)*P3/5)
 4240 RETURN
 4497 :
 4498 :
 4499 REM WAR PROGRESS
 4500 LET WS=WS+E(2)-.5
 4510 CLS
 4520 PRINT"PROGRESS OF FOREIGN WAR:"
 4530 PRINT
 4540 PRINT"According to the latest repo
rts from"
 4550 PRINT"the front lines, we are ";
 4560 IF WS>0 THEN PRINT"winning";
 4570 IF WS=0 THEN PRINT"bogged down";
 4580 IF WS<0 THEN PRINT"losing";
 4590 IF ABS(WS)<1 THEN PRINT"."
 4600 IF ABS(WS)>1 THEN PRINT:PRINT"by a
 wide margin."
 4610 PRINT"Do you want to withdraw and
call for a"
 4620 PRINT"diplomatic settlement ?"
 4630 INPUT A$
 4640 IF A$="Y" THEN GOTO 4700
 4650 IF WS<-1 AND RND(1)<.5 THEN GOSUB
4750
 4660 IF WS>1 AND RND(1)<.5 THEN GOSUB 4
850
 4670 GOSUB 9900
 4680 RETURN
 4699 :
 4700 LET WAR=0
 4710 LET P3=INT(RND(1)*P3)
 4720 LET WS=0
 4730 CLS
 4740 RETURN
 4749 :
 4750 PRINT
```

```
4760 PRINT"We have lost the war."
4770 LET WAR=0
4780 LET WS=0
4790 LET P1=INT(RND(1)*P1)
4800 LET P2=INT(RND(1)*P2)
4810 LET P3=INT(RND(1)*P3)
4820 RETURN
4849 :
4850 PRINT
4860 PRINT"We have won the war."
4870 LET WAR=0
4880 LET WS=0
4890 LET P1=INT(P1/2+RND(1)*P1)
4900 LET P2=INT(P2/2+RND(1)*P2)
4910 LET P3=INT(P3+RND(1)*20)
4920 RETURN
4997 :
4998 :
4999 REM CHANGE VARIABLES
5000 CLS
5010 LET F=INT(F+(E(3)+E(7)-1)*10+(RND(
1)-RND(1))*5)
5020 LET G=INT(G+RND(1)*5*(E(4)-RND(1))
/(I+.1))
5030 LET U=INT(ABS(U-RND(1)*G*1E5))
5040 LET C=INT(C+RND(1)*U/5E5-RND(1)*10
*E(1))
5050 LET P1=P1+RND(1)*(G-C-F-U/5E5+WS*5
)
5060 LET P2=P2+RND(1)*(G-C-F-U/5E5+WS*5
)
5070 LET P3=P3+RND(1)*(G-C-F-U/5E5+WS*5
)
5080 RESTORE 5300
5090 FOR X=1 TO 7
5100 READ I,J,K
5110 LET P1=INT((P1+I*E(X)+(RND(1)-RND(
1))*5))
5120 LET P2=INT((P2+J*E(X)+(RND(1)-RND(
1))*5))
5130 LET P3=INT((P3+K*E(X)+(RND(1)-RND(
```

```
1))*5))
 5140 NEXT X
 5150 RESTORE (5400+10*(P-1))
 5160 FOR X=1 TO 7
 5170 READ I
 5180 LET P1=INT((P1+I*(C(X)-.5)+(RND(1)
-RND(1))*5))
 5190 LET P2=INT((P2+I*(C(X)-.5)+(RND(1)
-RND(1))*5))
 5200 LET P3=INT((P3+I*(C(X)-.5)+(RND(1)
-RND(1))*5))
 5210 NEXT X
 5220 IF P1>20 THEN LET P1=20
 5230 IF P2>30 THEN LET P2=30
 5240 IF P3>50 THEN LET P3=50
 5250 RETURN
 5299 :
 5300 DATA 55,-10
 5310 DATA -5,0,10
 5320 DATA 0,-5,-10
 5330 DATA -5,0,15
 5340 DATA -5,10,15
 5350 DATA -5,10,15
 5360 DATA -5,10,15
 5399 :
 5400 DATA 7,7,7,7,7,7,7
 5410 DATA 7,7,0,0,7,7,0
 5420 DATA 0,0,0,0,0,0,0
 5497 :
 5498 :
 5499 REM REVOLUTION
 5500 CLS
 5510 LET V=P1+P2+P3
 5520 IF V<0 THEN LET V=0
 5530 PRINT"Your popularity has fallen t
o ";V;"%."
 5540 PRINT"There has been a revolution
and you"
 5550 PRINT"have been ousted. You lasted
 ";Y;" years."
 5560 GOTO 6500
```

```
 5997 :
 5998 :
 5999 REM GENERAL ELECTION
 6000 CLS
 6010 PRINT"It is year ";Y;", time for t
he ";
 6020 IF Y>5 THEN PRINT"next";
 6030 PRINT:PRINT"General Election."
 6040 PRINT
 6050 PRINT"The results are coming throu
gh..."
 6060 PRINT
 6070 PRINT
 6080 PRINT"PARTY";TAB(20);"VOTES"
 6090 PRINT
 6100 PRINT"COMMUNIST PARTY";TAB(20);INT
(RND(1)*150+1)
 6110 PRINT"ECOLOGY PARTY";TAB(20);INT(R
ND(1)*400+2)
 6120 RESTORE 6350
 6130 LET VL=100-P1-P2-P3
 6135 LET VM=0
 6140 FOR X=1 TO 3
 6150 READ I$
 6160 PRINT I$;" PARTY";
 6170 LET V=INT(RND(1)*VL)
 6180 IF (X=2 AND P=3) OR (X=3 AND P<3)
THEN LET V=VL
 6190 IF V>VM AND X<>P THEN LET V=VM
 6200 IF X=P THEN LET V=P1+P2+P3
 6210 IF X<>P THEN LET VL=VL-V
 6220 LET V=INT(5E5*V/(1+RND(1)/150))
 6230 PRINT TAB(20);V
 6240 NEXT X
 6250 PRINT
 6260 PRINT
 6270 IF P1+P2+P3>=VM THEN GOTO 6400
 6280 PRINT"You lost the election."
 6290 PRINT
 6300 GOTO 6500
 6349 :
```

```
 6350 DATA CONSERVATIVE,LIBERAL-SDP,LABO
UR
 6399 :
 6400 PRINT"You won the General Election
. Do you"
 6410 PRINT"want to continue as leader o
f your"
 6420 PRINT"party ?"
 6430 INPUT A$
 6440 IF A$="Y" THEN GOTO 160
 6497 :
 6498 :
 6499 REM REPLAY
 6500 PRINT
 6510 PRINT"Do you want another game ?"
 6520 INPUT A$
 6530 IF A$="Y" THEN RUN
 6540 END
 9898 :
 9899 REM KEY PRESS ROUTINE
 9900 PRINT
 9910 PRINT"PRESS ANY KEY TO CONTINUE"
 9920 REPEAT UNTIL GET$<>""
 9930 CLS
 9940 RETURN
```

The National Hero Game

This is a cross between a simple military game and an economic game. The country's economy is represented simply by a set of named variables, which are controlled to remain popular while retaining sufficient armed forces to deal with any wars.

The economic reports include percentage changes since the previous year. Economic inputs are checked by a subroutine (lines 38ØØ to 383Ø) to set the limits on action.

The other two countries are tested in turn to see whether an attack is to be made by either side. The outcome of any war is, of course, directly related to the size of the player's armed forces, though there is an option for a diplomatic solution.

The variables are not stored in arrays; they are named as in English (or abbreviated).

```
 10 REM MASTER ROUTINE
 20 CLS
 30 GOSUB 1000:REM INITIALISE
 40 GOSUB 1500:REM INTRO
 50 GOSUB 9000
 60 LET TW=1
 70 LET GEC=-1
 80 LET YEAR=1
 90 CLS
100 GOSUB 2000:REM DISPLAY
110 IF GEC<0 THEN GOSUB 2500
120 GOSUB 9000
130 GOSUB 3000:REM SAVE OLD VALUES
140 GOSUB 3500:REM INPUTS
150 GOSUB 9000
160 GOSUB 4000:REM VARIABLE CHANGE
170 GOSUB 4500:REM ENEMY ATTACK?
180 GOSUB 4600:REM PLAYER ATTACKS?
190 IF GEC=0 THEN GOTO 6000
200 IF Pop<RND(1)*.05 THEN GOTO 6500
210 IF YEAR>RND(1)*20 THEN LET Pop=Pop
-RND(1)*YEAR/50
```

```
 220 IF Pop<0 THEN LET Pop=0
 230 IF Pop>1 THEN LET Pop=1
 240 LET YEAR=YEAR+1
 250 LET GEC=GEC-1
 260 LET TW=TW+1
 270 GOTO 90
 997 :
 998 :
 999 REM INITIALISATION
1000 LET BP=1E7
1010 LET Tax=.2
1020 LET Wages=5E3
1030 LET Services=5E7
1040 LET DB=5E7
1050 LET FS=1E3
1060 LET Pop=.25+RND(1)*.5
1070 LET Tax1=Tax
1080 LET Wages1=Wages
1090 LET Services1=Services
1100 LET DB1=DB
1110 RETURN
1497 :
1498 :
1499 REM INTRO
1500 PRINT"THE NATIONAL HERO GAME"
1510 PRINT
1520 PRINT"After combined civil war and
 invasion,"
 1530 PRINT"you are appointed to lead yo
ur"
 1540 PRINT"country to recovery."
 1550 PRINT"You take all the important d
ecions"
 1560 PRINT"on the economy and border di
sputes"
 1570 PRINT"with your neighbours."
 1580 PRINT
 1590 PRINT"People expect you to call th
e first"
 1600 PRINT"election when conditions per
mit."
```

```
 1610 PRINT"You ultimate future depends
on the"
 1620 PRINT"outcome of that election."
 1630 RETURN
 1997 :
 1998 :
 1999 REM DISPLAY
 2000 PRINT"YEAR ";YEAR;" OF YOUR ADMINI
STRATION."
 2010 PRINT
 2020 PRINT"Popularity ";INT(Pop*100);"%
"
 2030 PRINT"Balance of payments $";BP
 2040 PRINT"Income tax ";INT(Tax*100);"%
-";
 2050 LET Q=Tax
 2060 LET Q1=Tax1
 2070 GOSUB 2300
 2080 PRINT"Basic wage $";Wages;"-";
 2090 LET Q=Wages
 2100 LET Q1=Wages1
 2110 GOSUB 2300
 2120 PRINT"Services $";Services;"-";
 2130 LET Q=Services
 2140 LET Q1=Services1
 2150 GOSUB 2300
 2160 PRINT"Defence Budget $";DB;"-";
 2170 LET Q=DB
 2180 LET Q1=DB1
 2190 GOSUB 2300
 2200 PRINT"Size of forces:";FS;" men."
 2210 RETURN
 2299 :
 2300 LET CH=INT(((Q+.01)/(Q1+.01)-1)*10
0)
 2310 IF CH<0 THEN LET I$="down "
 2320 IF CH>0 THEN LET I$="up "
 2330 IF CH=0 THEN LET I$="no change."
 2340 IF CH<>0 THEN LET I$=I$+STR$(ABS(C
H))+"%"
 2350  PRINT I$
```

```
 2360 RETURN
 2497 :
 2498 :
 2499 REM SET ELECTION DATE
 2500 PRINT
 2510 PRINT"Do you wish to call an elect
ion?"
 2520 INPUT A$
 2530 IF A$<>"Y" THEN RETURN
 2540 PRINT"How many years do you want i
t in?"
 2550 INPUT A
 2560 LET GEC=ABS(INT(A))
 2570 IF GEC>20 THEN LET GEC=20
 2580 IF Pop>.5 THEN LET Pop=Pop-RND(1)*
GEC/10
 2590 RETURN
 2997 :
 2998 :
 2999 REM SAVE OLD VARIABLE VALUES
 3000 LET Tax1=Tax
 3010 LET Wages1=Wages
 3020 LET Services1=Services
 3030 LET DB1=DB
 3040 RETURN
 3497 :
 3498 :
 3499 REM INPUTS
 3500 PRINT"The Budget for year ";YEAR
 3510 PRINT
 3520 PRINT"What rate of Income Tax?"
 3530 INPUT A
 3540 GOSUB 3800
 3550 LET Tax=INT(A)/100
 3560 PRINT"What % change in the basic w
age?"
 3570 INPUT A
 3580 GOSUB 3800
 3590 LET Wages=INT(Wages*(1+A/100))
 3600 PRINT"What % change in Services?"
 3610 INPUT A
```

```
 3620 GOSUB 3800
 3630 LET Services=INT(Services*(1+A/100
))
 3640 PRINT"What % change in the Defence
 Budget?"
 3650 INPUT A
 3660 GOSUB 3800
 3670 LET DB=INT(DB*(1+A/100))
 3680 RETURN
 3799 :
 3800 IF A<-90 THEN LET A=-90
 3810 IF A>90 THEN LET A=90
 3820 LET A=A+3*(RND(1)-RND(1))
 3830 RETURN
 3997 :
 3998 :
 3999 REM CHANGE VARIABLES
 4000 LET PM=Pop
 4010 LET Pop=Pop+Wages/(Wages1+1)/100
 4020 LET Pop=Pop+Services/(Services1+1)
/200
 4030 LET Pop=Pop*(1-Tax)
 4040 LET BP=BP-Services-DB+Tax*Wages*1E
6
 4045 IF BP<1E8 THEN LET BP=INT(BP)
 4050 IF BP>1E9 THEN LET Pop=Pop*.7
 4060 IF TW*FS>RND(1)*2000 THEN LET Pop=
Pop/(RND(1)*TW+1)
 4070 LET FS=INT((FS+DB/5E4)/2)
 4080 IF BP<0 THEN LET Pop=Pop*.6
 4090 LET Pop=(2*Pop+PM)/3
 4100 RETURN
 4497 :
 4498 :
 4499 REM ENEMY ATTACK TESTS
 4500 FOR X=1 TO 2
 4510 IF X=1 THEN LET N$="Slovenlia"
 4520 IF X=2 THEN LET N$="Bilharzia"
 4530 IF RND(1)<.1 THEN GOSUB 4700
 4540 NEXT X
 4550 GOSUB 9000
```

```
 4560 RETURN
 4597 :
 4598 :
 4599 REM PLAYER ATTACKS?
 4600 FOR X=1 TO 2
 4610 IF X=1 THEN LET N$="Slovenlia"
 4620 IF X=2 THEN LET N$="Bilharzia"
 4630 PRINT
 4640 PRINT"Do you wish to attack ";N$;"
?"
 4650 INPUT A$
 4660 IF A$="Y" THEN GOSUB 5500
 4670 NEXT X
 4680 GOSUB 9000
 4690 RETURN
 4697 :
 4698 :
 4699 REM ENEMY ATTACK
 4700 PRINT
 4710 PRINT N$;" has attacked the border
."
 4720 PRINT
 4730 PRINT"What do you want to do?"
 4740 PRINT"1.Nothing."
 4750 PRINT"2.Call for a diplomatic solu
tion."
 4760 PRINT"3.Fight back."
 4770 PRINT"Which one?"
 4780 INPUT A
 4790 IF A=1 THEN GOSUB 4900
 4800 IF A=2 THEN GOSUB 5000
 4810 IF A=3 THEN GOSUB 5500
 4820 LET TW=0
 4830 RETURN
 4897 :
 4898 :
 4899 REM DO NOTHING
 4900 LET Pop=Pop*RND(1)/2
 4910 PRINT"The enemy has occupied the a
rea."
 4920 RETURN
```

```
 4997 :
 4998 :
 4999 REM DIPLOMATIC SOLUTION
 5000 LET C=RND(1)*Pop
 5010 IF C<.25 THEN GOSUB 5100
 5020 IF C>=.25 THEN GOSUB 5200
 5030 RETURN
 5099 :
 5100 PRINT"No solution could be found."
 5110 PRINT"This means war."
 5120 LET Pop=Pop-RND(1)/3
 5130 GOSUB 5500
 5140 RETURN
 5199 :
 5200 PRINT"A diplomatic solution has be
en found."
 5210 LET Pop=Pop+RND(1)/5
 5220 RETURN
 5497 :
 5498 :
 5499 REM FIGHTING
 5500 LET S=0
 5510 IF FS*RND(1)>300 THEN LET S=1
 5520 PRINT
 5530 IF S=0 THEN PRINT"You lost the bat
tle-some land is held."
 5540 IF S=1 THEN PRINT"You won the batt
le and took some land."
 5550 LET Pop=Pop+RND(1)*(S-.5)
 5560 LET FS=INT(FS+RND(1)*FS*(S-2)/2)
 5570 LET TW=0
 5580 RETURN
 5997 :
 5998 :
 5999 REM GENERAL ELECTION
 6000 CLS
 6010 PRINT"The General Election you cal
led is"
 6020 PRINT"imminent. How much money do
you wish"
 6030 PRINT"to spend on the campaign?"
```

```
 6040 INPUT A
 6050 LET A=ABS(A)
 6060 LET BP=BP-A
 6070 LET Pop=Pop*(1+RND(1)/5-A/(BP+1))
 6080 IF Pop>1 THEN LET Pop=1
 6090 IF Pop<0 THEN LET Pop=0
 6100 CLS
 6110 DIM R(3)
 6120 LET VC=1-Pop
 6130 FOR X=1 TO 3
 6140 LET R(X)=INT(VC*RND(1)*1E6)
 6150 LET VC=VC-R(X)/1E6
 6160 NEXT X
 6170 CLS
 6180 PRINT"The results of the voting ar
e:"
 6190 PRINT
 6200 PRINT"Republican Party ";R(1);" vo
tes"
 6210 PRINT"Ecology Party ";R(2);" votes
"
 6220 PRINT"Democratic Party ";R(3);" vo
tes"
 6230 PRINT"You ";INT(Pop*1E6);" votes"
 6240 PRINT"Communist Party ";INT(VC*1E6
);" votes"
 6250 PRINT
 6260 IF Pop<.5 THEN PRINT"You have lost
."
 6270 IF Pop<.5 AND Pop>.2 THEN PRINT"A
coalition formed to beat you."
 6280 IF Pop>=.5 AND Pop<.7 THEN PRINT"Y
ou have a place in the next government."
 6290 IF Pop>=.7 AND Pop<.9 THEN PRINT"Y
ou will head the next government."
 6300 IF Pop>=.9 THEN PRINT"The people w
ant you to stay as dictator!"
 6310 PRINT
 6320 IF Pop<.1 THEN PRINT"You should ne
ver be let near power."
 6330 IF Pop>.8 THEN PRINT"You should ta
```

```
ke up politics."
 6340 GOTO 7000
 6497 :
 6498 :
 6499 REM REVOLT
 6500 LET S=0
 6510 PRINT"The people have rebelled aga
inst you."
 6520 PRINT"Do you want to call in the a
rmy?"
 6530 INPUT A$
 6540 IF A$="Y" THEN LET S=FS-2E3
 6550 IF S<=0 THEN PRINT"You have been r
emoved from office."
 6560 IF Pop<.01 AND A$="Y" THEN PRINT"T
he army has mutinied."
 6570 IF S>0 AND Pop>=.01 THEN PRINT"You
 have a military dictatorship now."
 6580 PRINT
 6590 PRINT"Advice-never accept power!"
 6600 GOTO 7000
 6997 :
 6998 :
 6999 REM REPLAY
 7000 PRINT
 7010 PRINT"Do you want another game?"
 7020 INPUT A$
 7030 IF A$="Y" THEN RUN
 7040 END
 8997 :
 8998 :
 8999 REM WAIT TILL KEY PRESSED
 9000 PRINT
 9010 PRINT"PRESS A KEY TO CONTINUE"
 9020 REPEAT UNTIL GET$<>""
 9030 CLS
 9040 RETURN
```

Corridors of Power

This is a fairly comprehensive economic game, with many real-life variables. The object is, of course, to win the General Election but, to gain a sufficiently high popularity, the variables must be manipulated with care.

The state of the economy is displayed as a list of variables every "six months". The changes made are percentage increases or decreases. Interactions like strikes are also included, and your popularity at General Election time is affected by your manifesto promises, as well as your performance.

The variables are given full names or initials where a full name would have been too long. This method, as opposed to using arrays, was preferable in view of the complications involved in relating the variables while writing the game.

Various extreme loss situations are possible: revolution and total economic collapse. This provides more fitting epitaphs for people who make outrageous decisions.

```
  10 REM MASTER ROUTINE
  20 GOSUB 1000:REM INITIALISE
  30 CLS
  40 GOSUB 1500:REM INTRO
  50 GOSUB 9900
  60 GOSUB 2000:REM MANIF INIT
  70 LET YEAR=1:LET HALF=1:LET GEC=4
  80 GOSUB 9900
  90 GOSUB 2500:REM MAIN DISPLAY
 100 GOSUB 9900
 110 GOSUB 3000:REM INPUTS
 120 GOSUB 3500:REM VARS ALTERED+REPORT
 130 IF Pay*(1-Intax)<Costs OR UB<Costs
THEN GOTO 8000
 140 IF SIE<-1E10 THEN GOTO 8200
 150 GOSUB 9900
 160 GOSUB 4000:REM STRIKES?
 170 FOR X=1 TO 2
```

```
  180 LET CN=INT(RND(1)*4)+1
  190 IF RND(1)*X>.6 THEN GOSUB (4500+CN
*200)
  200 NEXT X
  210 PRINT
  220 PRINT"Do you want to call a Genera
l Election"
  230 PRINT"for sooner than at present?"
  240 INPUT A$
  250 IF A$="Y" THEN GOSUB 6000
  260 CLS
  270 IF GEC=0 AND HALF=1 THEN GOTO 7000
  280 LET HALF=HALF+1
  290 IF HALF>2 THEN GOSUB 500
  300 IF Pop>1 THEN LET Pop=1
  310 IF Pop<0 THEN LET Pop=0
  320 GOTO 80
  499 :
  500 LET HALF=1
  510 LET YEAR=YEAR+1
  520 LET GEC=GEC-1
  530 RETURN
  997 :
  998 :
  999 REM INITIALISATION
 1000 LET Pop=.5+RND(1)/3
 1010 LET BP=1E10
 1020 LET NDB=1E10
 1030 LET FS=1E5
 1040 LET SB=7.5E9
 1050 LET Pay=2000
 1060 LET UB=1500
 1070 LET Prod=2
 1080 LET Costs=1000
 1090 LET Unemp=INT(RND(1)*5E6)
 1100 LET Aid=1E6
 1110 LET Intax=.2
 1120 LET Ptax=.4
 1130 LET Eptax=.25
 1140 LET Iptax=.3
 1150 LET TPS=1000
```

```
 1160 LET VAT=.15
 1170 RETURN
 1397 :
 1498 :
 1499 REM INTRO
 1500 PRINT"CORRIDORS OF POWER"
 1510 PRINT
 1520 PRINT"You are the leader of the De
mocratic"
 1530 PRINT"Party in a country of 50 mil
lion. In"
 1540 PRINT"the last General Election, y
ou won with"
 1550 PRlNT"a majority of ";INT(Pop*1000
)/10;"%."
 1560 PRINT
 1570 PRINT"As Prime Minister, you take
decisions"
 1580 PRINT"on policy and any important
occurences"
 1590 PRINT"which appear."
 1600 PRINT
 1610 PRINT"Every 5 years or whenever yo
u choose,"
 1620 PRINT"there is a General Election.
 If you win"
 1630 PRINT"more than 98% of the vote, a
ll the"
 1640 PRINT"other parties join you to fo
rm a one-"
 1650 PRINT"party state under you."
 1660 RETURN
 1997 :
 1998 :
 1999 REM MANIFESTO GENERATION
 2000 LET MIntax=.15
 2010 LET MUB=UB+INT(RND(1)*UB/10)
 2020 LET MPay=Pay+INT(RND(1)*Pay/10)
 2030 LET MSB=SB+RND(1)*SB/10
 2040 LET MUnemp=Unemp-INT(RND(1)*Unemp/
2)
```

```
 2050 GOSUB 7500
 2060 RETURN
 2497 :
 2498 :
 2499 REM MAIN DISPLAY
 2500 IF HALF=1 THEN PRINT"FIRST";
 2510 IF HALF=2 THEN PRINT"SECOND";
 2520 PRINT" HALF OF YEAR ";YEAR
 2530 PRINT
 2540 PRINT"Popularity ";INT(Pop*1000)/1
0;"%"
 2550 PRINT"Balance of Payments $";INT(B
P/1E6);" million"
 2560 PRINT"Public Services Budget $";IN
T(SB/1E6);" million"
 2570 PRINT"Defence Expenditure $";INT(N
DB/1E6);" million"
 2580 PRINT"Size of Armed Forces ";FS;"
men"
 2590 PRINT"Industrial Aid $";INT(Aid/1E
5)/10;" million"
 2600 PRINT
 2610 PRINT"Basic weekly wage $";INT(Pay
/25)
 2620 PRINT"Weekly Unemployment Benefit
$";INT(UB/25)
 2630 PRINT"Weekly Cost of Living $";INT
(Costs/25)
 2640 PRINT"Number of unemployed ";Unemp
 2650 PRINT"Income Tax ";INT(Intax*100);
"%"
 2660 PRINT"Tax on Profits ";INT(Ptax*10
0);"%"
 2670 PRINT"Import Tax ";INT(Iptax*100);
"%"
 2680 PRINT"Export Tax ";INT(Eptax*100);
"%"
 2690 PRINT"VAT ";INT(VAT*100);"%"
 2700 RETURN
 2997 :
 2998 :
```

```
 2999 REM INPUTS
 3000 PRINT"BUDGET FOR THIS SEASON"
 3010 PRINT
 3020 PRINT"What % Income Tax do you wan
t?"
 3030 GOSUB 3400
 3040 LET Intax=ABS(A)
 3050 PRINT"What % change in Public Serv
ices?"
 3060 GOSUB 3400
 3070 LET SB=SB*(1+A)
 3080 PRINT"What % change in Defence Exp
enditure?"
 3090 GOSUB 3400
 3100 LET NDB=NDB*(1+A)
 3110 PRINT"What % change in Unemploymen
t Benefit?"
 3120 GOSUB 3400
 3130 LET UB=INT(UB*(1+A))
 3140 PRINT"What % change in the Basic W
age?"
 3150 GOSUB 3400
 3160 LET Pay=INT(Pay*(1+A))
 3170 PRINT"What % Profit Tax do you wan
t?"
 3180 GOSUB 3400
 3190 LET Ptax=ABS(A)
 3200 PRINT"What % Import Tax do you wan
t?"
 3210 GOSUB 3400
 3220 LET Iptax=ABS(A)
 3230 PRINT"What % Export Tax do you wan
t?"
 3240 GOSUB 3400
 3250 LET Eptax=ABS(A)
 3260 PRINT"What % VAT do you want?"
 3270 GOSUB 3400
 3280 LET VAT=ABS(A)
 3290 PRINT"What % change in Industrial
Aid?"
 3300 GOSUB 3400
```

```
 3310 LET Aid=Aid*(1+A)
 3320 RETURN
 3399 :
 3400 INPUT A
 3410 IF A>40 THEN LET A=40
 3420 IF A<-40 THEN LET A=-40
 3430 LET A=INT(A)/100
 3440 RETURN
 3497 :
 3498 :
 3499 REM VARIABLES ALTERED+REPORT
 3500 LET FS=INT((FS+NDB/5E4)/3)
 3510 LET BP=BP-NDB-SB-UB*Unemp-Aid+(5E7
-Unemp)*Pay*Intax+5E7*Costs*VAT
 3520 LET PM=ABS((5E7-Unemp)*(Pay*(1-Int
ax)-Costs))
 3530 LET SIE=(1-VAT)*UB*Unemp+(5E7-Unem
p)*(TPS*Prod-Pay)+Aid
 3540 LET IG=INT(SIE/20/Costs)
 3550 IF IG>Unemp THEN LET IG=Unemp
 3560 LET Unemp=Unemp-IG
 3570 LET EM=ABS(SIE/4)+SIE/4
 3580 CLS
 3590 PRINT"INDUSTRIAL REPORT FOR THE SE
ASON"
 3600 PRINT
 3610 PRINT"Growth:";IG;" men"
 3620 PRINT"Exports:$";INT(EM/1E6);" mil
lion"
 3630 PRINT"Imports:$";INT(PM/1E6);" mil
lion"
 3640 LET Costs=INT(Costs+Costs*(RND(1)-
RND(1))/20)
 3650 LET BP=BP+SIE*Ptax+PM*Iptax+EM*Ept
ax
 3660 LET Pop=Pop*(1+SGN(SB-MSB)/10)
 3670 LET Pop=Pop*(1+SGN(MIntax-Intax)/1
0)
 3680 LET Pop=Pop*(1+SGN(Pay-MPay)/8)
 3690 LET Pop=Pop*(1+SGN(UB-MUB)/10)
 3700 LET Pop=Pop*(1+SGN(MUnemp-Unemp)/1
```

```
0)
 3710 IF Pop>1 THEN LET Pop=1
 3720 IF Pop<0 THEN LET Pop=0
 3730 IF BP<-1E6 THEN LET Pop=Pop*.6
 3740 IF SIE<0 THEN LET Pop=Pop*.7
 3750 RETURN
 3997 :
 3998 :
 3999 REM STRIKES ROUTINE
 4000 IF MPay-Pay>Pay/10 THEN GOSUB 4100
 4010 IF RND(1)<.2 THEN GOSUB 4150
 4020 RETURN
 4099 :
 4100 PRINT"Strikes have broken out beca
use the"
 4110 PRINT"Government has not kept its
payrise"
 4120 PRINT"promises."
 4130 GOSUB 4200
 4140 RETURN
 4149 :
 4150 PRINT"Unions have called a strike
for a"
 4160 PRINT"shorter working week."
 4170 GOSUB 4200
 4180 RETURN
 4199 :
 4200 PRINT
 4210 PRINT"Here are your choices:"
 4220 PRINT" 1.Do nothing."
 4230 PRINT" 2.Increase working hours."
 4240 PRINT" 3.Insist on a wage cut."
 4250 PRINT" 4.Offer a pay increase."
 4260 PRINT" 5.Offer to reduce working h
ours."
 4270 PRINT" 6.Agree if productivity is
improved."
 4280 PRINT"Which do you want?"
 4290 INPUT B
 4300 IF B>6 OR B<1 THEN LET B=1
 4310 IF B=6 THEN GOTO 9500
```

```
 4320 LET B=B-3.5
 4330 LET Pop=Pop*(1+RND(1)*B/20)
 4340 IF B=-2.5 THEN RETURN
 4350 PRINT"By how many % ?"
 4360 GOSUB 3400
 4370 IF ABS(B)=1.5 THEN LET TPS=INT(TPS
*(1+SGN(B)*A))
 4380 IF ABS(B)<>1.5 THEN LET Pay=INT(Pa
y*(1-SGN(B)*A))
 4390 LET Pop=Pop*(1-A/5)
 4400 GOSUB 9900
 4410 RETURN
 4497 :
 4498 :
 4499 REM "CHANCE PILE"
 4500 LET C=INT(RND(1)*50)
 4510 PRINT"NATO has called for an incre
ase of ";C;"%"
 4520 PRINT"in military spending. Will y
ou comply?"
 4530 INPUT A$
 4540 IF A$="Y" THEN LET NDB=NDB*(1+C/10
0)
 4550 IF A$<>"Y" THEN GOSUB 4600
 4560 RETURN
 4599 :
 4600 LET S=RND(1)
 4610 IF S>.5 THEN PRINT"Your refusal ha
s been accepted."
 4620 IF S>.5 THEN RETURN
 4630 PRINT"You have been forced to acce
pt."
 4640 LET Pop=Pop*.8
 4650 LET NDB=NDB*(1+C/100)
 4660 RETURN
 4699 :
 4700 LET C=INT(RND(1)*50)
 4710 PRINT"The World Health Organisatio
n has"
 4720 PRINT"called for an increase of ";
C;"% in"
```

```
 4730 PRINT"Public Services expenditure.
 Will you"
 4740 PRINT"comply?"
 4750 INPUT A$
 4760 IF A$="Y" THEN LET SB=SB*(1+C/100)
 4770 IF A$<>"Y" THEN GOSUB 4800
 4780 RETURN
 4799 :
 4800 LET S=RND(1)
 4810 IF S>.5 THEN LET Pop=Pop*.8
 4820 IF S>.5 THEN RETURN
 4830 PRINT"You have been forced to comp
ly."
 4840 LET Pop=Pop*.95
 4850 LET SB=SB*(1+C/100)
 4860 RETURN
 4899 :
 4900 PRINT"There has been a disaster in
 the Third"
 4910 PRINT"World. How much aid will you
 send?"
 4920 INPUT A
 4930 LET A=ABS(A)
 4940 LET BP=BP-A
 4950 IF A<1E4 THEN LET Pop=Pop*.9
 4960 IF A>1E6 THEN LET Pop=Pop*1.05
 4970 RETURN
 5099 :
 5100 PRINT"There has been an outbreak o
f an"
 5110 PRINT"epidemic disease in the coun
try."
 5120 LET S=-.1
 5130 IF SB>1E10*RND(1) THEN LET S=.1
 5140 PRINT"The Health Services were ";
 5150 IF S=0 THEN PRINT"un";
 5160 PRINT"able"
 5170 PRINT"to fight the disease effecti
vely."
 5180 LET Pop=Pop*(1+RND(1)*S)
 5190 RETURN
```

```
 5299 :
 5300 PRINT"A Disarmament agreement has
been made."
 5310 LET C=INT(RND(1)*60)
 5320 PRINT"Will you reduce Defence Expe
nditure"
 5330 PRINT"by ";C;"% ?"
 5340 INPUT A$
 5350 IF A$="Y" THEN GOSUB 5400
 5360 IF A$<>"Y" THEN LET Pop=Pop*(1+RND
(1)/25)
 5370 RETURN
 5399 :
 5400 LET NDB=NDB*(1-C/100)
 5410 LET Pop=Pop*(1+(RND(1)-RND(1))/10)
 5420 RETURN
 5997 :
 5998 :
 5999 REM GEN ELEC TIME
 6000 PRINT"How many years do you want t
he"
 6010 PRINT"General Election in?"
 6020 INPUT A
 6030 LET A=INT(ABS(A))
 6040 IF A>GEC THEN LET A=GEC
 6050 IF A=0 AND GEC>0 THEN LET A=1
 6060 PRINT"It has been arranged for ";A
;" years."
 6070 LET GEC=A
 6080 GOSUB 9900
 6090 RETURN
 6997 :
 6998 :
 6999 REM MANIFESTO AND ELECTION
 7000 PRINT"The General Election is due.
 You must"
 7010 PRINT"prepare your maniesto."
 7020 PRINT
 7030 PRINT"What are your promises for t
he"
 7040 PRINT"following:"
```

```
 7050 PRINT" Income Tax (%)"
 7060 INPUT A
 7070 LET MIntax=ABS(INT(A))/100
 7080 PRINT" Weekly Unemployment Benefit
 ($)"
 7090 INPUT A
 7100 LET MUB=INT(ABS(A))*25
 7110 PRINT" Public Services Budget ($)"
 7120 INPUT A
 7130 LET MSB=ABS(A)
 7140 PRINT" Unemployment (people)"
 7150 INPUT A
 7160 LET MUnemp=INT(ABS(A))
 7170 PRINT" Weekly Pay ($)"
 7180 INPUT A
 7190 LET MPay=INT(ABS(A))*25
 7200 LET Pop=Pop*(1+SGN(MUB-Costs-2E3)*
RND(1)/10)
 7210 LET Pop=Pop*(1+SGN(MSB-SB)*RND(1)/
10)
 7220 LET Pop=Pop*(1-MIntax/2)
 7230 LET Pop=Pop*(1-SGN(MUnemp-Unemp+1E
4)*RND(1)/10)
 7240 LET Pop=Pop*(1+SGN(MPay-Pay)*RND(1
)/5)
 7250 IF Pop>1 THEN LET Pop=1
 7260 IF Pop<0 THEN LET Pop=0
 7270 CLS
 7280 PRINT"GENERAL ELECTION-RESULTS"
 7290 PRINT
 7300 LET PL=1-Pop:LET MP=Pop
 7310 PRINT"Republican Party ";
 7320 GOSUB 7450
 7330 PRINT"Nationalist Party ";
 7340 GOSUB 7450
 7350 PRINT"Official Loony Party ";
 7360 GOSUB 7450
 7370 PRINT"Democratic Party ";INT(Pop*5
E7);" votes."
 7380 PRINT"Communist Party ";INT(PL*5E7
);" votes."
```

```
 7390 PRINT
 7400 IF Pop>=MP THEN PRINT"You won."
 7410 IF Pop>=MP AND Pop<=.98 THEN LET G
EC=5:GOTO 280
 7420 IF Pop>.98 THEN PRINT"You may now
form a one-party state."
 7430 IF Pop<MP THEN PRINT"You lose."
 7440 GOTO 8500
 7449 :
 7450 LET P=RND(1)*PL
 7460 IF P>MP THEN LET MP=P
 7470 PRINT" ";INT(P*5E7);" votes."
 7480 LET PL=PL-P
 7490 RETURN
 7497 :
 7498 :
 7499 REM MANIFESTO DISPLAY
 7500 PRINT"MANIFESTO OF THE DEMOCRATIC
PARTY"
 7510 PRINT
 7520 PRINT"We undertake to create the f
ollowing"
 7530 PRINT"situation in our country:"
 7540 PRINT"Income Tax ";INT(MIntax*100)
;"%"
 7550 PRINT"Weekly Unemployment Benefit
$";INT(UB/25)
 7560 PRINT"Public Services Budget $";IN
T(2*MSB/1E6);" million"
 7570 PRINT"Unemployed:less than ";MUnem
p
 7580 RETURN
 7997 :
 7998 :
 7999 REM REVOLUTION
 8000 GOSUB 9900
 8010 PRINT"The workers have revolted be
cause they"
 8020 PRINT"do not have enough money to
buy food."
 8030 GOTO 8500
```

```
 8197 :
 8198 :
 8199 REM ECON. COLLAPSE
 8200 GOSUB 9900
 8210 PRINT"The country's economy has co
llapsed."
 8220 PRINT"Because of your policies, th
e"
 8230 PRINT"industries were sucked dry a
nd ruined."
 8240 GOTO 8500
 8497 :
 8498 :
```

A new world is waiting for you. A world where *you* make the decisions, where *you* give the orders, where *you* snap your fingers and people run to do your bidding.

This is the world of strategy computer games; this is the world of political and military simulation programs. And, in this book, Mike Rose gives you the key to this world.

Step by step, Mike leads you through the tasks you need to carry out in order to create major programs of your own. By the end of the book, you'll be bursting with ideas to turn into programs, and you'll have acquired the skills which will enable you to convert those ideas into reality.

The book ends with ten major, ready-to-run programs, including:

- **DRAKE'S RETURN** (a naval battle in Elizabethan times)
- **LASERFIGHT IN THE OK SPACE ZONE**
- **DOWNING STREET**
- **THE SIEGE OF DOUNE CASTLE**

another great book from
Interface

ISBN 0-947695-20-6

9 780947 695200

£7.95