3D CONSTRUCTION KIT™

IBM PC, AMIGA and ATARI ST

CONTENTS

INTRODUCTION

Welcome to the 3D Construction Kit. We had often been asked when a Freescape creator would be made, so here it is! It represents a total of four and a half years of actual development, and many more man-years.

The program uses an advanced version of the Freescape 3D System, and will allow you to design and create your own 3D Virtual Worlds. These could be your living room, your office, an ideal home or even a space station.

You may then walk or fly through the three dimensional environment as if you were actually there. Look around and up and down, move forward and back, go inside buildings and even interact with moveable or animating objects. The facilities to make a fully fledged action adventure game are even included: just add imagination...

Most of all, though, just have fun creating, experimenting, colouring and playing in 3D - you can easily lose all track of time !

I hope you enjoy using the 3D Construction Kit as much as we enjoyed creating it.

Have fun !

Ian Andrew

Ian Andrew

REGISTRATION

It is essential to register as a 3D Construction Kit user, as support can only be given to registered owners. The registration form is included with the package.

All correspondence should be sent to Mandy Rodrigues, at the address shown below. If a reply is required a stamped addressed envelope must be enclosed.

THE 3D CONSTRUCTION KIT USERS CLUB

The Club is provided to offer additional help and advice for users of the 3D Construction Kit and will consist of a bi-monthly newsletter packed full of news, information, hints and tips on the system to allow everyone to use it to its full potential. It will also act as a forum for users to exchange ideas and information. To apply for membership of the club, just fill in the relevant section of the registration card and further details and information will be sent to you. All registration forms should be sent to:

Mandy Rodrigues, 67 Lloyd Street,
Llandudno, Gwynedd, LL30 2YP.

ACKNOWLEDGEMENTS

2

NOTICE

DISCLAIMER

VIRUSES

The disks included with the 3D Construction Kit are guaranteed to be in correct working order and free from virus programs. It is the purchaser's responsibility to prevent infection of this product with a virus, which may cause the program to cease working. Incentive Software will in no way accept liability or responsibility for virus infection or damage which can always be avoided by switching off the machine for at least 30 seconds before trying to use this product. It is also useful to bear the following points in mind, which will reduce the possibility of infection:

* Always switch off before loading a new program
* Use a virus checker program regularly
* NEVER use pirated disks
* NEVER boot from suspect disks
* Write protect your original disk

CONDITIONS

OBJECT

AREA

GENERAL

FILE

INTRODUCTION

LOADING INSTRUCTIONS

AMIGA

Booting from the program disc - insert Program disc in drive DFO: and reset the machine, the Construction Kit will autoload.

Running from Workbench - with the Program disc in drive DFO: double click on the icon for Drive DFO: to open its directory window and then double click on the Construction Kit icon to startup the Construction Kit.

Running from CLI - make sure the current CLI directory is the directory in which the Construction Kit program is stored, type 3DKIT to start up the Construction Kit. Note - it is essential that the following files are in the root directory - 3DKIT.RSC and SAMPLES.BNK.

Installing on Hard Disc - Run the INSTALL program from the CLI and answer the on screen prompts to install the Construction Kit on to a specified hard disc.

ATARI ST

Booting from Program disc - insert Program disc in drive A: and reset the machine, the Construction Kit will autoload.

Running from Desktop - with the Program disc in drive A: double click on the icon for drive A: to open its directory window then double click on the 3DKIT.PRG icon to start the Construction Kit.

Note - it is essential that the following files are in the same directory as the main program - 3DKITRSC and SAMPLES.BNK and that the program is called while the current directory is the same as the one containing the Construction Kit.

Installing on Hard Disc - Run the INSTALL.TOS program from the desktop and answer the on screen prompts to install the Construction Kit on to a hard disc.

PC

Boot up from DOS, then insert the relevant disc in the current drive. Type 3DKIT (Return) at the DOS prompt to load. Follow any on screen instructions.

Installing on Hard Disc - Follow the instructions for loading, but type INSTALL (Return) at the DOS prompt. Follow any on screen instructions. Type README (Return) at the DOS prompt to see any extra instructions or amendments to the manual (if any).

INTRODUCTION TO THE EDITOR

The 3D Construction Kit is designed to be user-friendly with icons and pull-down menus enabling the user to quickly understand the working environment.

Upon loading the program you will see the Main Screen (Figure 1) which is divided up into the following areas:

MENU SELECTOR: This is the top text line which contains the headings for the various menus. To access one of the menus simply move the mouse pointer over the desired heading and the relevant menu will open below the heading. Moving the mouse pointer over the options within the menu will highlight them and then pressing the mouse button will select the option currently highlighted. Moving the pointer out of the boundary of the menu will cause it to retract.

The PC version of the construction kit has the option not to use a mouse. In this case, the cursor keys (on the numeric keypad) will move the mouse cursor around the screen in the desired direction. To move faster, press either shift key at the same time. The Insert and Delete keys, also on the numeric keypad, act exactly like the left and

CONDITIONS

OBJECT

AREA

GENERAL

FILE

INTRODUCTION

MENU SELECTOR   **VIEW WINDOW**   **INFO BAR**

FILE   GENERAL   AREA   OBJECT   FREE:83970

AREA 001   POS:4000,0300,4000 ROT:000,000,000

EXCL HIGHL   WALK USER   MODE STEP

SELECT   COPY   CREATE   EDIT   TEST

RESET   CONDIT   DELETE   ATTRIB   COLOUR

MAIN SCREEN
SHORT CUT ICONS   *Figure 1*   **FREESCAPE ICONS**

MODE ICONS
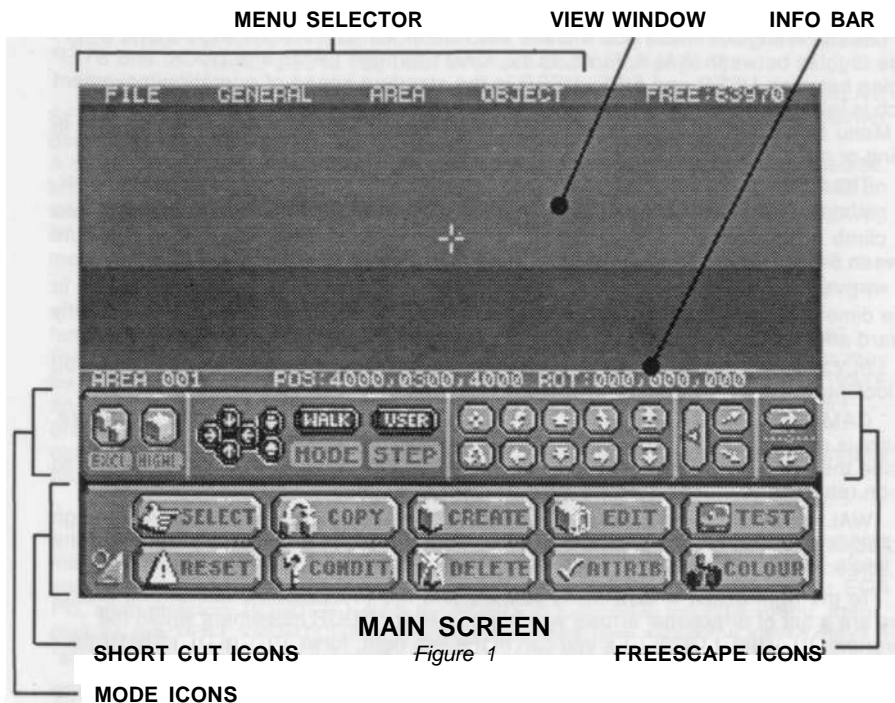
right mouse buttons. The joystick options act in similar manner; the stick moves the cursor, the buttons correspond to the mouse buttons.

Below the Menu Selector you will see the main VIEW window. This area is always used to display the current FREESCAPE view as seen from whichever camera is currently selected.

Below the VIEW window is the INFORMATION BAR. This initially reads AREA 001 POS:4000,0300,4000 ROT:000,000,000. This shows the current area, your present viewpoint coordinates (shown as X,Y,Z), and the angle of view (yaw, pitch and roll). When in edit mode this line will change to read the object name you are editing, its position in the environment and its size. The information will be especially useful when animation or other more advanced uses of the system are required.

Below the Information Bar you will see a series of Icons. These are the MODE and FREESCAPE icons. The MODE icons are on the left of the screen. EXCL (Exclude) is useful when editing objects. Clicking on this icon will EXCLude all background information and leave the currently selected object to be edited. Just to the right of this you will see HIGHL or Highlight which when activated will. HIGHLight the selected object for ease of identification during work. Just to the right of these you will see a set of small icons in the form of arrows. These icons are very useful. When an object is selected, eg. for editing, if these arrows are

5

activated they will lock onto the current object from the front, rear, either side, or top and bottom. Alongside these you will see two further icons which are MODE and STEP. Mode toggles between WALK, FLY1, FLY2, CAM1 through CAM5 and LOCK, and STEP toggles between USER and FINE. USER is the standard speed of operation/movement which is initially set by the PREFERENCES menu which is found under GENERAL on the Menu Bar. FINE is used for fine work when only a small movement is required in editing or movement.

The different modes selected by the MODE icon affect your movement as follows:

WALK allows you to move along the ground, with the restriction of gravity - you can climb onto objects and fall off them. Your height above the floor is restricted to between 64 and 280 units, corresponding to a crouched and standing position.

FLY1 removes restrictions on gravity. You can now fly with complete freedom in three dimensions. Forward motion is restricted to a horizontal plane, so that you can fly forward and look down at the same time.

FLY2 is very similar to FLY1, except that you now fly in exactly the direction you are looking.

CAM1 through CAM5 control five "cameras"[1] which can be placed anywhere. Control is similar to FLY1, except that the cameras are allowed inside objects and outside the area. When you change to another view the camera's position is saved, so that on returning to that camera the view position is retained.

WALK, FLY1 and FLY2 have collision detection built in; they will not travel through solid objects. These modes are the only three possible within a runnable program or the test screen.

To the right of the MODE icons you will find the FREESCAPE icons. The first of these are a set of directional arrows which are used for YOUR movement within the environment. Using these arrows you can move left, right, forwards, backwards, rotate left, rotate right, make a complete u-turn, move yourself up or down and toggle the cross-hair cursor on and off. To the right of these you will see the rest of the Freescape Icons which control your view movement. These allow you to look up, look down, roll and clicking on the centre "eye" icon will return your view to the centre view once more.

Note that the EDIT and FREESCAPE icons remain on the screen and can be used at most times during editing.

Below the MODE and FREESCAPE icons you will see the SHORTCUT icons. These icons are marked SELECT, COPY, CREATE, EDIT, TEST, RESET, CONDITION, DELETE, ATTRIBUTES and COLOUR. These are shortcut icons which duplicate the more commonly used functions which are also available from the Menus as follows (from left to right):

DRTCUT ICON	IN MENU
SELECT	OBJECT
COPY	OBJECT
CREATE	OBJECT
EDIT	OBJECT
TEST	GENERAL
RESET	GENERAL
CONDITION	OBJECT
DELETE	OBJECT
ATTR (Attributes)	OBJECT
COLOUR	OBJECT

GETTING TO KNOW THE MOVEMENT AND VIEWPOINT CONTROLS
(Refer also to The User Interface Section on page 9)

First load in a datafile from the disc. Move the mouse pointer to the MENU SELECTOR and move it to the left until the FILE menu appears. Move the pointer down until LOAD DATAFILE is highlighted and click the left mouse button. A dialogue box will appear showing all the available files and directories on thedisc. Move the mouse pointer to the file named KITGAME (CGAGAME or EGAGAME on the PC) and click the left mouse button. The name will appear next to the heading FILE. Next, click on the OK button and the datafile will load, and after a few moments will appear in the VIEW window.

Now using the FREESCAPE icons experiment with moving around the new environment. Move in all the directions you can until you become completely familiar with how to "move yourself around within the FREESCAPE landscape. Press the left mouse button within the VIEW window to see how some objects may be SHOT. Pressing the right mouse button within the VIEW window to ACTIVATE an object. Activating an object in the VIEW window will appear to have no effect unless conditions have been entered which are triggered by the ACTIVATED? condition. Try this on the door to the building. (Note that activating objects can only be done within a finite range).

THE 3D KIT GAME
This has been included as an example to illustrate some of the environments that are possible. This is supplied as a datafile and can be played as a stand alone game. First load KGBORDER (or KGCGAPIC for CGA PC, or KGEGAPIC for EGA PC, all in directory BORDER), from the BORDER function in the RLE menu, then click on the TEST SHORTCUT icon to play the game from within the kit. To make a "stand alone" game or environment use the MAKE function found in the FILE menu

The object of the game is to escape from the mysterious world in which you find yourself, and return to Earth. Some sort of space vehicle will probably come in handy (large clue). F1 will return you to the Editor.

Advanced use has been made of animations and conditions, and these can be examined and edited using the relevant functions.

See if you can complete the game without cheating !

CREATING AND EDITING YOUR FIRST OBJECT
First the existing datafile must be cleared from the VIEW window. To do this, move the mouse pointer up to the MENU SELECTOR and move along to the FILE menu. Move the pointer down until the CLEAR ALL is highlighted and press the mouse button. An ALERT BOX will appear warning that all current data will be lost if the operation continues. Click on OK and after a few moments the VIEW window will clear revealing an empty area.

Now move the mouse pointer to the SHORTCUT icons and click on CREATE. These icons will now be replaced with a further set of icons each showing a particular type of object for you to select. Move the mouse pointer to the CUBE icon and click the mouse button. A grey cube will now appear in the VIEW window. Note that the SHORTCUT icons reappear once the cube has been created.

Next select the COLOUR icon and you will see that a list of objects appears on the lower half of the screen. At present it should show:

> 001 CUBOID 001
> 002 CUBOID 002

7

CONDITIONS

OBJECT

AREA

GENERAL

FILE

NTRODUCTION

Click on cuboid 002 to select this object. Move the mouse pointer to the small "tick" at the top left of the selector and click on this. An alternate method would be to click directly on the cube in the VIEW window. The screen should now change to show the colour panel.

To the left of the colour bar you will see six small squares which represent the six sides of the cube and show their current colours, opposite sides of the cube are linked by square brackets. At the moment there should be two white, two medium grey and two dark grey. These may vary depending on the computer.

Colouring the cube can be done in two ways: using these squares to colour all sides of the cube in one easy movement, or using the image of the cube on the screen. To colour the cube the easy way, move the mouse pointer over any colour you wish to use and select that colour by clicking the left mouse button. Note that the colour you have selected will appear in the small window above the UNDO and OKAY icons to the right of the colour bar. Now move the mouse pointer over one of the squares on the left of the colour bar and click the right mouse button to transfer the chosen colour. The left mouse button acts as a "get colour" and the right mouse button as a "put colour" Repeat this process until all six of the squares are coloured to your choice. You will also note that at the same time the cube in the VIEW window is also being coloured. Selecting UNDO will undo the last colour changes. Also note that pointing at an area of the VIEW window and clicking the left mouse button will select this colour.

The ST and Amiga versions allow colouring of the cube directly on the VIEW window. Just move the mouse pointer to the colour you require on the colour bar and press the left mouse button, check that the box on the right has changed to the chosen colour (also note that the chosen colour will be highlighted on the colour bar). Now move the mouse pointer to the face of the cube and click the right mouse button to transfer the chosen colour to the face of the cube.

For obvious reasons the first method of colouring the object is preferable as all sides may be coloured at once. The second method would involve either turning the cube, walking to the other side of the cube to view the hidden side, or using the View Lock arrows.

The Horizon colours can only be altered (on the ST and Amiga only) in the colour areas above and below the S and G letters. Use the Sky and Ground Icons on PC.

Now we will edit the cube. Move the pointer to the OKAY icon to the right of the colour bar and press the mouse button. The SHORTCUT icons will now reappear. Move the mouse pointer to the EDIT icon and press the mouse button to select it. Now either select CUBOID 2 from the object selector list or click on the cube in the VIEW window.

The EDIT Window shows five different groups of icons, POINT, TURN, SHRINK, STRETCH and MOVE. Depending on which type of object you are editing, only the editing groups available for your selected object are shown. As we are editing a CUBE the POINT icons are dimmed to show that they are not available.

Note that when an object is first created it is positioned above the "ground" so we will remedy this now. Move the mouse pointer to the MOVE icons and position the pointer over the icon shown as an arrow pointing down with a small line above it and press the mouse button. The cube in the VIEW window will now begin to move downwards. Keep pressing this icon and watch the INFORMATION BAR to see how the position coordinates change. The bottom of the cube may disappear from view as the cube is moved downwards. When you have moved the cube down as far as it will go

move the mouse pointer to the FREESCAPE icons and select the arrow pointing downwards. Press the left mouse button to "move" yourself backwards from the cube until all areas of the cube are visible.

Now move the mouse pointer to the STRETCH icons and click the left mouse button over the icon represented by an arrow pointing to the right. The cube will now stretch towards the right. SHRINK has the opposite effect to STRETCH.

Experiment a little with these icons until you are completely familiar with stretching, shrinking and turning/flipping the cube. Then try to bring the cube back to its original size (200,200,200).

When you have done this, move the mouse pointer to the OKAY icon and the SHORTCUT icons will reappear. Now move the mouse pointer to the COPY icon. The item selector will appear in the usual way.

Select the cube by clicking on this in the VIEW window. A DIALOGUE BOX will appear requesting that you select where you wish to copy the object to. Click on LEFT and click on OK. You will now see that the cube has been copied to the left of the existing cube. This will be called CUBOID 003. The new cube can be edited in the same way by selecting the cube from the item selector in the usual way.

THE USER INTERFACE

FILE SELECTOR

The file selector (See Figure 2) will appear when SAVE DATA, LOAD DATA, LOAD OBJECT or LOAD BORDER is selected from the FILE menu at the top of the VIEW window. The first files in the current directory will be displayed. The arrows
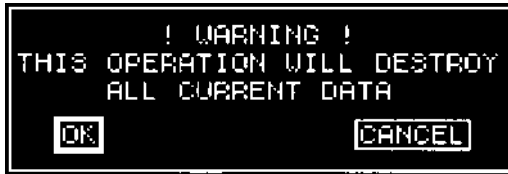
FILE SELECTOR
Figure 2

9

on the right of the file selector will scroll up or down through the files in the current directory if there are more files than can be shown in the space available. Select the required file by clicking on the filename with the mouse button and the selected filename will appear to the right of the FILE heading. Alternatively, you can click on the FILE heading and type in the file name using the keyboard, pressing RETURN when satisfied. To load or save the selected file click on OK. Just above the FILE name is the PATH which shows which drive is currently being used, along with the current position within directories. This can be changed to another drive or hard disc by clicking on it and typing the new path name in similar fashion to the filename. (ST and AMIGA only).

ALERT BOXES

During environment creation there will be instances where something you have attempted to do will be either not possible or irrevocable. In these instances an ALERT BOX (See Figure 3) will appear with information about the function requested. If the function is not possible the alert box will simply inform you of this and then wait for confirmation before cancelling the function. If the function is irrevocable i.e. CLEAR ALL the alert box will give you the chance to reconsider the action and continue with or stop the function. ALERT BOXES are also used to inform you that an otherwise invisible function has executed correctly. In this case the alert box will simply require input from you to confirm the message has been received.

```
┌─────────────────────────────────┐
│       ! WARNING !               │
│  THIS OPERATION WILL DESTROY    │
│      ALL CURRENT DATA           │
│                                 │
│  [OK]              [CANCEL]     │
└─────────────────────────────────┘
```

ALERT BOX
Figure 3

DIALOGUE BOXES

There are various parts of the environment creation which will require input from you to set parameters relating to the current function. These parameters will usually be set within a DIALOGUE BOX (See Figure 4).

DIALOGUE BOXES are simply windows which will open at a set place (usually the centre of the screen) these are similar to ALERT BOXES but with the added ability of user interaction. Interaction takes place in one of two ways, buttons or text/numerical input. Buttons can be one of three types. TOGGLE buttons which toggle between selected and unselected when activated. RADIO buttons which are part of a group of at least two buttons, when one radio button in a group is selected all other members of the group are unselected. FUNCTION buttons which do a specific task directly i.e. OK or CANCEL.

Text boxes may be edited by first clicking with the mouse over the text to be edited and if the text may be edited it will become inversed and a cursor will be displayed at the first character. You may then type your text in using the normal functions. DEL deletes a character under the cursor, BACKSPACE deletes the character before the cursor and other keys enter the desired character over the current content of the cursor position. Some text lines will restrict you to either NUMERICAL or ALPHA

10

CONDITIONS

OBJECT

AREA

GENERAL

FILE

INTRODUCTION

DIALOGUE BOX
Figure 4

characters only. To end editing of a particular text item, simply press the RETURN key whereupon the text will return to normal print and any restrictions on numerical values will be applied i.e. if you were to type in the number 9000 for an object position, as the maximum area coordinate is 8192, it will automatically be restricted to 8192 on pressing RETURN. Note that while editing a text or numerical item it is impossible to exit the DIALOGUE BOX or edit any other fields until you have finished editing the current text item by pressing RETURN.

TEXT EDITING

Text editing takes place in the lower half on the screen directly below the VIEW window after selecting any of the icons which bring up the text editing window. An inverse square will indicate your current cursor position. This position may be changed by either the keyboard or the control icons. (See Figure 5)
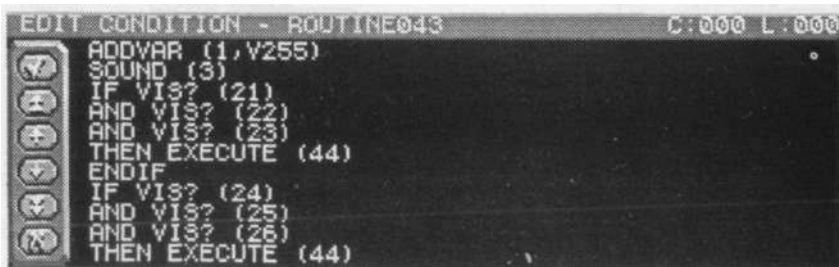
TEXT EDITOR
Figure 5

11

TEXT EDITING CONTROLS

ST and AMIGA

[←] MOVE CURSOR LEFT
ONE CHARACTER

[→] MOVE CURSOR RIGHT
ONE CHARACTER

[↑] MOVE CURSOR UP
ONE LINE

[↓] MOVE CURSOR DOWN
ONE LINE

[SHIFT] + [↑] MOVE CURSOR
UP 1 PAGE

[SHIFT] + [↓] MOVE CURSOR
DOWN 1 PAGE

[RETURN] INSERT A LINE

[B.SPACE] DELETE THE CHARACTER
BEFORE THE CURSOR

[DELETE] DELETE THE CHARACTER
UNDER THE CURSOR

[HELP] TOGGLE BETWEEN, HALF
& FULL SCREEN MODES

[ESC] CANCEL EDIT

Then select the TICK icon to enter your text into memory or the CROSS icon to cancel.

PC

[←] MOVE CURSOR LEFT
ONE CHARACTER

[→] MOVE CURSOR RIGHT
ONE CHARACTER

[↑] MOVE CURSOR UP
ONE LINE

[↓] MOVE CURSOR DOWN
ONE LINE

[Pg Up] MOVE CURSOR UP
ONE PAGE

[Pg Dn] MOVE CURSOR
DOWN ONE PAGE

[RETURN] INSERT A LINE

[B.SPACE] DELETE THE CHARACTER
BEFORE THE CURSOR

[DELETE] DELETE THE CHARACTER
UNDER THE CURSOR

[HOME] MOVE CURSOR TO START
OF TEXT

[END] MOVE CURSOR TO END
OF TEXT

[ESC] CANCEL EDIT

[SHIFT] + [RETURN] ACCEPT
EDIT

[F1] to [F10] INSERT SOME OF
THE MORE COMMON
FUNCTIONS

Then select the TICK icon to enter your text into memory or the CROSS icon to cancel.

12

FILE MENU OPTIONS

CONDITIONS

OBJECT

AREA

GENERAL

FILE

INTRODUCTION

FILE	GENERAL	AREA	OBJECT

Name: **SAVE DATA**

Function: To save all the data in memory to disc as a datafile.

Action: When File Selector appears, enter a name for the datafile, press RETURN, then click on OK or press RETURN again.

Response: The current datafile will be saved to disc. (See the Appendix for list of Error Messages).

Note 1: This function actually saves all FREESCAPE data to the disc including - Datafile, colours, sound data (not samples), and name data.

Note 2: On the PC, the file extension will be forced to .KIT, and an second file with extension .NAM will also be saved which contains the object and area names.

Note 3: On both the PC and ST, the maximum name length is 8 characters.

Note 4: Possible errors include Disk write protected, Disk full, No disk in drive.

Note 5: For a full list of Amiga and Atari ST error messages, see Appendix.

SAVE DATA
LOAD DATA
SAVE OBJECT
LOAD OBJECT
LOAD BORDER
MAKE
CLEAR ALL
DELETE FILE
ABOUT
QUIT

FILE	GENERAL	AREA	OBJECT
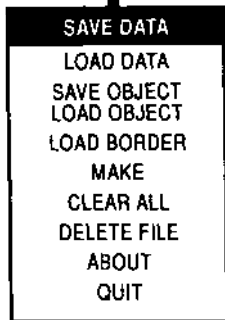
Name: **LOAD DATA.**

Function: To load datafile from disc.

Action: When File Selector appears, select datafile from the file selector and click on OK or press return.

Response: The datafile will be loaded from disc.

Note 1: Any data previously in memory will be over-written.

Note 2: On the PC, a file with extension .NAM will also be loaded. If this is not found, default area and object names will be used.

Note 3: See also notes for SAVE DATA.

SAVE DATA
LOAD DATA
SAVE OBJECT
LOAD OBJECT
LOAD BORDER
MAKE
CLEAR ALL
DELETE FILE
ABOUT
QUIT

13
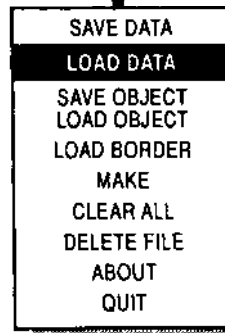
FILE	GENERAL	AREA	OBJECT

Name: **SAVE OBJECT**

Function: To save a single Object or Group to disc for reloading later (or into another area or datafile).

Action: Enter name to save object, press RETURN, then click on OK or press RETURN again.

Response: The object will be saved to disc.

Note 1: Neither conditions nor names will be saved along with the object.

Note 2: On the PC, the file extension will be forced to .OBJ .

Note 3: See also LOAD OBJECT.

```
SAVE DATA
LOAD DATA
SAVE OBJECT
LOAD OBJECT
LOAD BORDER
MAKE
CLEAR ALL
DELETE FILE
ABOUT
QUIT
```

FILE	GENERAL	AREA	OBJECT

Name: **LOAD OBJECT**

Function: Load a specified Object or Group into the current data.

Action: Click on object name you wish to load. Click on OK or press RETURN.

Response: The object will be loaded from disc.

Note: First an attempt will be made to position the object at the position at which it was saved. If this fails the object will be moved up and further attempts to position will be made. If the top of the area is encountered the operation will be aborted and an alert box will be displayed showing the required coordinates and size of the object, it is then up to you to ensure that enough free space exists for the object to be placed before retrying.

```
SAVE DATA
LOAD DATA
SAVE OBJECT
LOAD OBJECT
LOAD BORDER
MAKE
CLEAR ALL
DELETE FILE
ABOUT
QUIT
```

14

FILE	GENERAL	AREA	OBJECT

Name: **LOAD BORDER**

Function: Load a picture file from disc to use as a border in the TEST screen.

Action: ST and AMIGA only: When a DIALOGUE BOX appears requesting Format, select DEGAS, IFF or NEOchrome. Then click on OK.

Response: A file selector will appear.

Action: Click on the file to be loaded from disc. Then click on OK or press RETURN.

Response: The border will be loaded from disc into the TEST Screen.

```
SAVE DATA
LOAD DATA
SAVE OBJECT
LOAD OBJECT
LOAD BORDER
MAKE
CLEAR ALL
DELETE FILE
ABOUT
QUIT
```

Note 1: Borders can be created and loaded into memory from other programs providing that they are in one of the acceptable formats i.e.
ST/AMIGA: 320 x 200 pixels, 16 colours, Lo-Res and NTSC format.
PC CGA: 320 x 200 pixels, 4 colours.
PC EGA: 320 x 200 pixels, 16 colours.

Note 2: On the PC, borders must be in IFF format only, and have an extension of .LBM .

FILE	GENERAL	AREA	OBJECT

Name: **MAKE** (ST and AMIGA)

Function: To make a "stand alone" environment from the data in memory.

Action: When a file selector appears asking for the "RUNNER" (which is a program supplied on the Construction Kit Disc). Find the "RUNNER" program either on the Main Program Disc or on your Hard Disc, if you have copied it onto there). Click on OK once it is located.

Response: The File Selector will reappear asking for the MAKE PATH.

```
SAVE DATA
LOAD DATA
SAVE OBJECT
LOAD OBJECT
LOAD BORDER
MAKE
CLEAR ALL
DELETE FILE
ABOUT
QUIT
```

Action: Choose the directory you wish your environment to be created on

15

and type a name for the environment into the FILE name area (any extender will be ignored). If the drive with the "RUNNER" disc in and the drive with the selected environment disc are the same i.e. both A or DFO: you will be asked to ensure that the correct disc is inserted at various intervals during the "MAKE" process.

Note: When completed there will be four new files in the chosen directory, <NAME> (or <NAME>.PRG), <NAME>.DAT, <NAME>.SAM and <NAME>.BDR as follows:

<NAME> - Tne main program. (<NAME>.PRG on the ST)
<NAME>.DAT - The encrypted datafile.
<NAME>.SAM - The sample bank.
<NAME>.BDR - The border screen.

Where <NAME> is the environment title given in stage two.

Run <NAME> to run your environment.

Free distribution of stand alone runnable data files is permitted providing that the 3D Construction Kit is acknowledged on screen and in any accompanying documentation.

FILE	GENERAL	AREA	OBJECT

Name: **MAKE** (PC only)

Function: To make a stand alone environment from the data in memory.

Action: Enter name for the runnable datafile in the file selector, then press RETURN. Click on OK or press RETURN again.

SAVE DATA
LOAD DATA
SAVE OBJECT
LOAD OBJECT
LOAD BORDER
MAKE
CLEAR ALL
DELETE FILE
ABOUT
QUIT

Response: The datafile will be saved.

Note 1: The file extension will be forced to .RUN .

Note 2: In order to run the "stand alone" environment you will need a datafile saved using the MAKE option as detailed above, and a copy of the appropriate runner program from the RUNNERS directory on the master disk. This will be either RUNCGA.EXE, or RUNEGA.EXE depending on the video mode required. To run the environment, enter at the DOS prompt:

　　　　RUNCGA<NAME>.RUN

where <NAME> is the name of the required datafile. For an EGA datafile, type RUNEGA instead of RUNCGA.

RUNEGA will only run datafiles saved from the EGA version of the Construction Kit, and similarly for CGA.

Note 3: On running a stand-alone environment, a control option menu will appear.

16

FILE	GENERAL	AREA	OBJECT

		SAVE DATA
Name:	**CLEAR ALL**	LOAD DATA
Function:	To clear the current Data from memory and replace with the default area.	SAVE OBJECT
		LOAD OBJECT
Response:	Alert Box will appear requesting confirmation of the action.	LOAD BORDER
		MAKE
Action:	Select OK or CANCEL from the Alert Box.	**CLEAR ALL**
Response:	If OK selected the current data will be cleared. If CANCEL selected the Data will be left as it was.	DELETE FILE
		ABOUT
		QUIT

FILE	GENERAL	AREA	OBJECT

		SAVE DATA
Name:	**DELETE FILE**	LOAD DATA
Function:	Deletes an unwanted Datafile from the disc.	SAVE OBJECT
		LOAD OBJECT
Response:	A file selector will appear.	LOAD BORDER
Action:	Select the Datafile to be deleted and select OK.	MAKE
		CLEAR ALL
Response:	The Datafile will be deleted from the disc.	**DELETE FILE**
Note:	This function will allow deletion of old files from the disc. This can be used to free space on a disc prior to saving.	ABOUT
		QUIT

17

FILE	GENERAL	AREA	OBJECT

Name: ABOUT

Function: To display credits and release number.

SAVE DATA
LOAD DATA
SAVE OBJECT
LOAD OBJECT
LOAD BORDER
MAKE
CLEAR ALL
DELETE FILE
ABOUT
QUIT

FILE	GENERAL	AREA	OBJECT

Name: QUIT

Function: To exit 3D Construction Kit.

SAVE DATA
LOAD DATA
SAVE OBJECT
LOAD OBJECT
LOAD BORDER
MAKE
CLEAR ALL
DELETE FILE
ABOUT
QUIT

FILE	GENERAL	AREA	OBJECT

Name: **PREFERENCES**

Function: Set up specific Parameters used in the Construction Kit to your own preference.

Response: A Dialogue Box will appear.

Action: Enter the required numerical input.

Response: The Preferences will be set to those selected whenever possible.

Note 1: This function allows you to set up the step sizes for the vehicles/cameras used during editing. There are three step sizes: move step size, angle step size and object step size. Each step size has two entries, one for the left mouse button and one for the right button when clicking on the icons. The keys will use the left mouse button step size.

Note 2: On the ST and AMIGA, you may set the current edit buffer size. When the buffer size is changed ALL CURRENT DATA IS LOST. If an attempt is made to allocate more memory than is available the system will allocate as much as it can. After allocating a buffer of a different size than that requested, an ALERT BOX will be displayed showing the amount requested and the actual amount allocated.

```
PREFERENCES
RESET
CREATE CONDITION
EDIT CONDITION
DELETE CONDITION
DEFAULTS
CONTROLS
CREATE INSTRUMENT
EDIT INSTRUMENT
SET VIEW WINDOW
TEST
```

FILE	GENERAL	AREA	OBJECT

Name: **RESET**

Function: Resets the game/environment to the initial position as set in the defaults.

Response: The game/environment will reset.

Note: This also resets all objects/animators to their initial status and clears all variables except Variable 255.

```
PREFERENCES
RESET
CREATE CONDITION
EDIT CONDITION
DELETE CONDITION
DEFAULTS
CONTROLS
CREATE INSTRUMENT
EDIT INSTRUMENT
SET VIEW WINDOW
TEST
```

19

PREFERENCES
RESET
CREATE CONDITION
EDIT CONDITION
DELETE CONDITION
DEFAULTS
CONTROLS
CREATE INSTRUMENT
EDIT INSTRUMENT
SET VIEW WINDOW
TEST

Name: **CREATE CONDITION**

Function: Create a new GENERAL condition.

Response: A new GENERAL condition will be allocated ready for editing.

Note: GENERAL conditions are conditions which are executed each FREESCAPE frame regardless of the players position with the exception of the Initial" condition specified in the Defaults section.

PREFERENCES
RESET
CREATE CONDITION
EDIT CONDITION
DELETE CONDITION
DEFAULTS
CONTROLS
CREATE INSTRUMENT
EDIT INSTRUMENT
SET VIEW WINDOW
TEST

Name: **EDIT CONDITION**

Function: To edit a GENERAL condition.

Response: A list of existing General Conditions will be displayed in the Item Selector.

Action: Select a condition from the Item Selector. The selected condition will then be displayed below the VIEW Window. This can be edited using normal text editing.

Note-   See also CONDITIONS.

FILE	GENERAL	AREA	OBJECT

Name: **DELETE CONDITION**

Function: Deletes a General Condition.

Response: A list will be displayed, as above, and once a selection is made from the list the specified condition will be erased from memory.

```
PREFERENCES
RESET
CREATE CONDITION
EDIT CONDITION
DELETE CONDITION
DEFAULTS
CONTROLS
CREATE INSTRUMENT
EDIT INSTRUMENT
SET VIEW WINDOW
TEST
```

FILE	GENERAL	AREA	OBJECT

Name: **DEFAULTS**

Function: Set up the default game variables.

Action: Within this DIALOGUE BOX you can alter:

1. The climb ability
2. The "safe" fall distance
3. The activate range
4. The Timer frequency
5. The start area
6. The start entrance
7. The initial mode
8. The initial General Condition number

```
PREFERENCES
RESET
CREATE CONDITION
EDIT CONDITION
DELETE CONDITION
DEFAULTS
CONTROLS
CREATE INSTRUMENT
EDIT INSTRUMENT
SET VIEW WINDOW
TEST
```

Note: RESET should be selected to set these Defaults.

21

Name: **CONTROLS**

Function: To set up the controls that are used in the Test Screen.

Response: A list of the available controls will be displayed.

Action: Select a control by clicking left mouse button over the name. Once a control is selected any key pressed will become the key associated with that control.

Response: The key name will be displayed next to the control.

Next to the key name you will notice a small tick or cross. This indicates whether an icon in the Test Screen has been defined for the control or not.

Action: To define an icon for the control, click right mouse button over the name.

Response: The currently loaded BORDER (if any) will replace the Construction Kit Screen.

Action: To set the position of the icon, move the mouse pointer to the top left corner of the desired area and click the left mouse button. You will notice a box appear, this box will follow your mouse pointer movements growing and shrinking accordingly. Move the mouse to the bottom right corner of the desired area so that the box encompasses the area of the icon (note although the icon area must be rectangular this does not mean that the image of the icon on the border must be, as the rectangle of the icon definition is invisible anyway), then click on the mouse button to set the definition.

Response: A Dialogue Box will appear.

Action: Select whether the icon should be activated by the left, right, either, or both mouse buttons in the Dialogue Box. Once the icon is set the list of controls will be redisplayed and you may continue to edit the controls. When editing is completed select the TICK icon to set the controls and the Construction Kit screen will be redisplayed.

Note 1:   Selecting "either" will register both left and right mouse buttons. For movement the right button will default to a step size five times that of the left button, and turning will step by 30 degrees.

Note 2:   Using any of the function keys as an associated key will set that control to undefined - it cannot be used, neither from the keys nor using an icon.

Note 3:   Icon   controls can only be activated if an associated key is defined.

Note 4:   See appendix for default controls.

The side menu lists:

PREFERENCES
RESET
CREATE CONDITION
EDIT CONDITION
DELETE CONDITION
DEFAULTS
CONTROLS
CREATE INSTRUMENT
EDIT INSTRUMENT
SET VIEW WINDOW
TEST

22

CONDITIONS

OBJECT

AREA

GENERAL

FILE

INTRODUCTION

FILE	GENERAL	AREA	OBJECT

Name: **CREATE INSTRUMENT**

Function: Allocate a new Instrument.

Response: A new Instrument will be added.

Note: When created, an instrument will default to UNDEFINED and will therefore not be displayed at all. The new instrument can be edited using the EDIT INSTRUMENT function detailed below.

PREFERENCES
RESET
CREATE CONDITION
EDIT CONDITION
DELETE CONDITION
DEFAULTS
CONTROLS
CREATE INSTRUMENT
EDIT INSTRUMENT
SET VIEW WINDOW
TEST

FILE	GENERAL	AREA	OBJECT

Name: **EDIT INSTRUMENT**

Function: To Edit the various parameters associated with Instruments.

Response: A list of the current Instruments will be displayed.

Action: Select the Instrument from the Item Selector.

Response: A dialogue box will be displayed.

Note: Each parameter required to define an instrument will now be dealt with in turn.

TYPE: Each instrument must have a type, these include HORIZONTAL (bar), VERTICAL (bar), NUMERICAL and TEXT WINDOW or UNDEFINED.

PREFERENCES
RESET
CREATE CONDITION
EDIT CONDITION
DELETE CONDITION
DEFAULTS
CONTROLS
CREATE INSTRUMENT
EDIT INSTRUMENT
SET VIEW WINDOW
TEST

The two bars are thermometer style sliding indicators, they can be various sizes and combinations of colours, also the direction of the bar can be set using min. max. values explained later, a sliding bar must have associated with it a VARIABLE number as this is where the instrument will get its current setting from.

The NUMERICAL type is simply a text area where the value of its associated instrument is displayed in decimal. This can include negative numbers. If the minimum value is negative, the instrument will use a sign if necessary to display the value.

23

The TEXT WINDOW type is an area on the screen definable as both height and width in which messages may be printed using the FCL command PRINT. (See Conditions Section).

POSITION and SIZE define the screen position and size of the instrument, all instrument types are positioned to pixel boundaries, horizontal and vertical bar sizes are defined in increments of 1 pixel while text window and numerical types are defined in steps of 8 pixels (1 character).

VARIABLE NUMBER contains the number of the variable (if required) that the instrument will fetch its value from.

LEFT/BOTTOM contains the leftmost/bottommost value for a sliding bar, or numeric instrument, by making this value lower than the RIGHT/TOP value the bar will either go down or to the left or vice versa.

RIGHT/TOP contains the uppermost/rightmost value for a sliding or numeric bar, see also LEFT/BOTTOM, the step change for a sliding bar will be automatically scaled according to the difference between LEFT/BOTTOM and RIGHT/TOP values and the size of the bar.

FG/BG COLOUR contains the two colour numbers (for the Foreground and Background) in which the bar/text will be printed.

The following are legal colours for instruments:

PC CGA	- Alpha/Numeric instruments	0..3
	- Horizontal/Vertical bars	0..15
PC EGA	- Alpha/Numeric instruments	0..15
	- Horizontal/Vertical bars	0..255
ST/AMIGA	- All instruments	0..15

Instruments associated with a variable may be updated in two ways, either by altering the contents of the associated variable, in which case the instrument is automatically updated or by calling the FCL command UPDATEI with the relevant number. The text window type instrument can be updated only by using the FCL command PRINT.

FILE	GENERAL	AREA	OBJECT

PREFERENCES
RESET
CREATE CONDITION
EDIT CONDITION
DELETE CONDITION
DEFAULTS
CONTROLS
CREATE INSTRUMENT
EDIT INSTRUMENT
SET VIEW WINDOW
TEST

Name: **SET VIEW WINDOW**

Function: To set the size and position of the FREESCAPE view window in the Test screen.

Response: The 3D Construction Kit screen will be replaced by the alternate Test window. This will be black if no border has been loaded.

Action: Position the mouse pointer at the top left of your required window and drag the box to surround the area you wish to be included. Click the left mouse button once more at the bottom right and the window will be set.

Note: PC only - the width is limited to 256 pixels.

24

FILE	GENERAL	AREA	OBJECT

Name: **TEST**

Function: Go to the Test screen allowing the environment to be tested.

Note 1: This performs the same function as the F1 key which toggles between the two screens. Pressing the F1 key is necessary to return to the editor.

Note 2: Cameras are not allowed in the test area. The mode will default to WALK, FLY1 or FLY2 if using a camera when the test screen is accessed.

PREFERENCES
RESET
CREATE CONDITION
EDIT CONDITION
DELETE CONDITION
DEFAULTS
CONTROLS
CREATE INSTRUMENT
EDIT INSTRUMENT
SET VIEW WINDOW
TEST

AREA MENU OPTIONS

FILE	GENERAL	AREA	OBJECT

Name: **CREATE AREA**

Function: Create a new area.

Response: A new Area will be created and the viewpoint will be moved to this new Area.

Note: All new Areas contain an Entrance near the centre (Entrance 001) and a base (Cuboid 001). If these are not required they may be deleted.

CREATE AREA
EDIT AREA
DELETE AREA
GOTO AREA
AREA COLOURS
CREATE CONDITION
EDIT CONDITION
DELETE CONDITION
CREATE ENTRANCE
EDIT ENTRANCE
DELETE ENTRANCE
GOTO ENTRANCE
CREATE ANIMATION
EDIT ANIMATION
DELETE ANIMATION

Name:	**EDIT AREA**
Function:	Displays a list of existing areas and allows the user to edit the area specific information.
Response:	A list of all existing areas is displayed in the Item Selector.
Action:	Select an area to edit from the Item Selector.
Response:	A dialogue box will appear. This shows the area name, the number of definitions in the area (including OBJECTS, ENTRANCES and ANIMATORS), the area scale and whether or not the horizon is active. All of these elements may be edited in the usual dialogue box fashion, except for the number of definitions.
Note:	Only set horizon to OFF if the viewpoint is restricted to move only within a completely enclosed environment. (All four walls, floor AND ceiling.)

CREATE AREA
EDIT AREA
DELETE AREA
GOTO AREA
AREA COLOURS
CREATE CONDITION
EDIT CONDITION
DELETE CONDITION
CREATE ENTRANCE
EDIT ENTRANCE
DELETE ENTRANCE
GOTO ENTRANCE
CREATE ANIMATION
EDIT ANIMATION
DELETE ANIMATION

Name:	**DELETE AREA**
Function:	Delete a specified area.
Response:	A list of existing areas will be displayed in the Item Selector.
Action:	Select an area from the Item Selector.
Response:	The entire contents of the selected area including objects and local conditions will be removed from memory.
Note:	This function is irreversible so use carefully! Also note that you cannot delete the Area you are currently in.

CREATE AREA
EDIT AREA
DELETE AREA
GOTO AREA
AREA COLOURS
CREATE CONDITION
EDIT CONDITION
DELETE CONDITION
CREATE ENTRANCE
EDIT ENTRANCE
DELETE ENTRANCE
GOTO ENTRANCE
CREATE ANIMATION
EDIT ANIMATION
DELETE ANIMATION

CONDITIONS

OBJECT

AREA

GENERAL

FILE

INTRODUCTION

FILE	GENERAL	**AREA**	OBJECT

Name: **GOTO AREA**

Function: To move viewpoint to another area.

Response: A list of existing areas will be displayed.

Action: Select an area to go to.

Response: Will move the viewpoint to the new area selected.

Note: For Atari ST and Amiga: Area 000 is the Global Area and is only accessible from here. See also LIST GLOBALS in OBJECT MENU.

CREATE AREA
EDIT AREA
DELETE AREA
GOTO AREA
AREA COLOURS
CREATE CONDITION
EDIT CONDITION
DELETE CONDITION
CREATE ENTRANCE
EDIT ENTRANCE
DELETE ENTRANCE
GOTO ENTRANCE
CREATE ANIMATION
EDIT ANIMATION
DELETE ANIMATION

FILE	GENERAL	**AREA**	OBJECT

Name: **AREA COLOURS** (Atari ST and Amiga)

Function: To re-colour the 16 solid base colours in the current area.

Response: The COLOUR AREA panel will appear. (See Figure 6 on the next page). The panel contains three slider bars, one for each element of the colour red, green and blue. In each of these bars is a marker showing the current level of each element, also beside the bars the level is indicated by a number from 0 to 15. The levels of each element may be changed by either clicking on the arrow icons to either side of the sliders which will increase/ decrease the level in steps of 1, or click on the slider bar itself will move the pointer to the mouse position directly.

CREATE AREA
EDIT AREA
DELETE AREA
GOTO AREA
AREA COLOURS
CREATE CONDITION
EDIT CONDITION
DELETE CONDITION
CREATE ENTRANCE
EDIT ENTRANCE
DELETE ENTRANCE
GOTO ENTRANCE
CREATE ANIMATION
EDIT ANIMATION
DELETE ANIMATION

To the right of the sliders are a number of icons, these include:

27
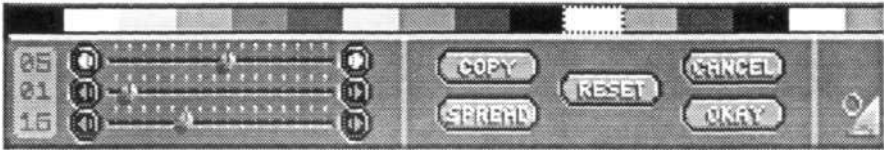
AREA COLOUR (ST & Amiga)
Figure 6

RESET: To reset the colours to their original values before any changes were made.

CANCEL: Exit and ignore any changes.

OKAY: Exit and save the new changes.

SPREAD: Will wait for you to select another colour from the colour bar and will approximate a smooth graduation between the two selected colours.

COPY: Will wait for you to select another colour and will then copy the original selected colour to the new position.

Above the panel is a display of the current 16 colours. To select a colour to edit simply click the mouse button over it. The flashing box will move to the new colour and its values will be displayed in the slider bars.

| FILE | GENERAL | **AREA** | OBJECT |

Name: **AREA COLOUR** (PC CGA)

Function: To toggle between the two available palettes.

Action: Select any object from the list.

Response: The COLOUR OBJECT panel will appear.

Action: Use palette icon to toggle area colours.

CREATE AREA
EDIT AREA
DELETE AREA
GOTO AREA
AREA COLOURS
CREATE CONDITION
EDIT CONDITION
DELETE CONDITION
CREATE ENTRANCE
EDIT ENTRANCE
DELETE ENTRANCE
GOTO ENTRANCE
CREATE ANIMATION
EDIT ANIMATION
DELETE ANIMATION

| FILE | GENERAL | **AREA** | OBJECT |

Name: **AREA COLOUR** (PC EGA)

Function: To re-colour the solid base colours in the current area.

Response: The COLOUR AREA panel will appear. (See Figure 7).

Action: Click on the colour to edit.

Response: Selected colour is displayed on the left of the screen.

Action: Click on the [R] [G] [B] [I] icons to toggle the colour values.

Function: COPY: Select colour to copy current colour to.

RESET: Sets all colours to their default values.

CANCEL: Exit without registering any changes.

OK: Return to main edit screen.

Note 1: Each area has its own set of colours.

Note 2: It is advisable not to change the colours used by the editor.

CREATE AREA
EDIT AREA
DELETE AREA
GOTO AREA
AREA COLOURS
CREATE CONDITION
EDIT CONDITION
DELETE CONDITION
CREATE ENTRANCE
EDIT ENTRANCE
DELETE ENTRANCE
GOTO ENTRANCE
CREATE ANIMATION
EDIT ANIMATION
DELETE ANIMATION
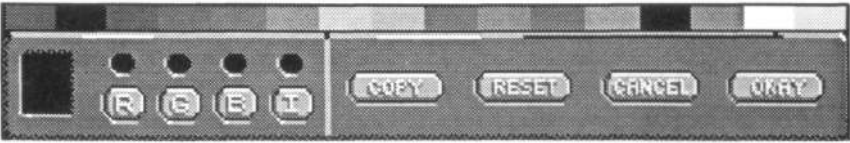
AREA COLOUR EGA
Figure 7

29

Name: **CREATE CONDITION**

Function: To create a new area condition.

Response: A new area condition will be created for editing.

Note: Area conditions are executed each frame when in the current area.

```
CREATE AREA
  EDIT AREA
DELETE AREA
 GOTO AREA
AREA COLOURS
CREATE CONDITION
 EDIT CONDITION
DELETE CONDITION
CREATE ENTRANCE
 EDIT ENTRANCE
DELETE ENTRANCE
 GOTO ENTRANCE
CREATE ANIMATION
 EDIT ANIMATION
DELETE ANIMATION
```

| FILE	GENERAL	AREA	OBJECT

Name: **EDIT CONDITION**

Function: Edit an Area condition.

Response: A list of the current area conditions will be displayed.

Action: Select the condition for editing.

Response: The condition will be displayed for editing.

Action: Edit the condition in the normal manner (see TEXT EDITING).

```
CREATE AREA
  EDIT AREA
DELETE AREA
 GOTO AREA
AREA COLOURS
CREATE CONDITION
 EDIT CONDITION
DELETE CONDITION
CREATE ENTRANCE
 EDIT ENTRANCE
DELETE ENTRANCE
 GOTO ENTRANCE
CREATE ANIMATION
 EDIT ANIMATION
DELETE ANIMATION
```

| FILE | GENERAL | AREA | OBJECT |

Name: **DELETE CONDITION**

Function: Delete a selected Area Condition.

Response: A list of conditions will be displayed in the Item Selector.

Action: Select a condition.

Response: The condition will be deleted from memory.

CREATE AREA
EDIT AREA
DELETE AREA
GOTO AREA
AREA COLOURS
CREATE CONDITION
EDIT CONDITION
DELETE CONDITION
CREATE ENTRANCE
EDIT ENTRANCE
DELETE ENTRANCE
GOTO ENTRANCE
CREATE ANIMATION
EDIT ANIMATION
DELETE ANIMATION

| FILE | GENERAL | AREA | OBJECT |

Name: **CREATE ENTRANCE**

Function: Create a new entrance in the current area.

Response: A new entrance will be created at your present position.

Note: The new entrance will contain the position and viewdirection of the viewpoint at the time of its creation, therefore to set up an entrance to a specific view simply move to that position and look in the desired direction. Then select CREATE ENTRANCE and the view will be stored as the last Entrance.

CREATE AREA
EDIT AREA
DELETE AREA
GOTO AREA
AREA COLOURS
CREATE CONDITION
EDIT CONDITION
DELETE CONDITION
CREATE ENTRANCE
EDIT ENTRANCE
DELETE ENTRANCE
GOTO ENTRANCE
CREATE ANIMATION
EDIT ANIMATION
DELETE ANIMATION

31

FILE	GENERAL	**AREA**	OBJECT

Name:	**EDIT ENTRANCE**
Function:	Allows you to edit an existing entrance.
Response:	A list of current entrances will be displayed.
Action:	Select the entrance to be edited in the usual manner.
Response:	A dialogue box will appear on the screen. Within the dialogue box will be details of the entrance; NAME, POSITION and ROTATION. These can be edited.
Action:	Edit the entrance in the dialogue box in the usual manner.

CREATE AREA
EDIT AREA
DELETE AREA
GOTO AREA
AREA COLOURS
CREATE CONDITION
EDIT CONDITION
DELETE CONDITION
CREATE ENTRANCE
EDIT ENTRANCE

DELETE ENTRANCE
GOTO ENTRANCE
CREATE ANIMATION
EDIT ANIMATION
DELETE ANIMATION

FILE	GENERAL	**AREA**	OBJECT

Name:	**DELETE ENTRANCE**
Function:	Deletes a specified entrance from memory.
Response:	A list of the current entrances will be displayed in the Item Selector.
Action:	Select an entrance in the usual manner.
Response:	The selected entrance will be deleted from memory.
Note:	This operation is irreversible, use with care!

CREATE AREA
EDIT AREA
DELETE AREA
GOTO AREA
AREA COLOURS
CREATE CONDITION
EDIT CONDITION
DELETE CONDITION
CREATE ENTRANCE
EDIT ENTRANCE

DELETE ENTRANCE

GOTO ENTRANCE
CREATE ANIMATION
EDIT ANIMATION
DELETE ANIMATION

FILE	GENERAL	**AREA**	OBJECT

Name: **GOTO ENTRANCE**

Function: Move to a specified entrance within the current area.

Response: A list of available Entrances will be displayed.

Action: Select an entrance in the usual manner.

Response: The viewpoint will be moved to the selected entrance.

CREATE AREA
EDIT AREA
DELETE AREA
GOTO AREA
AREA COLOURS
CREATE CONDITION
EDIT CONDITION
DELETE CONDITION
CREATE ENTRANCE
EDIT ENTRANCE
DELETE ENTRANCE
GOTO ENTRANCE
CREATE ANIMATION
EDIT ANIMATION
DELETE ANIMATION

FILE	GENERAL	**AREA**	OBJECT

Name: **CREATE ANIMATION**

Function: Create a new animator.

Response: A new animator will be created ready for editing.

CREATE AREA
EDIT AREA
DELETE AREA
GOTO AREA
AREA COLOURS
CREATE CONDITION
EDIT CONDITION
DELETE CONDITION
CREATE ENTRANCE
EDIT ENTRANCE
DELETE ENTRANCE
GOTO ENTRANCE
CREATE ANIMATION
EDIT ANIMATION
DELETE ANIMATION

33

CONDITIONS

OBJECT

AREA

GENERAL

FILE

INTRODUCTION

Name:	**EDIT ANIMATION**	
Function:	Allows editing of animation commands.	
Response:	A list of existing animators will be displayed in the Item Selector.	
Action:	Select the desired animator.	
Response:	The commands for that animator will be decompiled and displayed for editing.	
Action:	Edit or add to these commands in the same way as all conditions. (see TEXT EDITING). Also see CREATING AN ANIMATION.	

```
CREATE AREA
EDIT AREA
DELETE AREA
GOTO AREA
AREA COLOURS
CREATE CONDITION
EDIT CONDITION
DELETE CONDITION
CREATE ENTRANCE
EDIT ENTRANCE
DELETE ENTRANCE
GOTO ENTRANCE
CREATE ANIMATION
EDIT ANIMATION
DELETE ANIMATION
```

Name:	**DELETE ANIMATION**	
Function:	Delete a specified animation from memory.	
Response:	A list of all existing animators will be displayed.	
Action:	Select an animator in the usual manner.	
Response:	The selected animator will be deleted from memory.	
Note:	This operation is irreversible, use with care!	

```
CREATE AREA
EDIT AREA
DELETE AREA
GOTO AREA
AREA COLOURS
CREATE CONDITION
EDIT CONDITION
DELETE CONDITION
CREATE ENTRANCE
EDIT ENTRANCE
DELETE ENTRANCE
GOTO ENTRANCE
CREATE ANIMATION
EDIT ANIMATION
DELETE ANIMATION
```

OBJECT MENU OPTIONS

CONDITIONS

OBJECT

AREA

GENERAL

FILE

INTRODUCTION

FILE	GENERAL	AREA	**OBJECT**

Name: **CREATE OBJECT**

Function: Create a new object in the current area.

Response: A panel (See Figure 8) will be displayed over the SHORTCUT Icons showing the type of object available.

Action: Select an object type.

Response: The new object will be created in front of the current view.

Note: The new object name will default to its type followed by its number. These can be changed using the ATTRIBUTES function.

> **CREATE OBJECT**
> EDIT OBJECT
> DELETE OBJECT
> SELECT OBJECT
> COPY
> CONDITION
> ATTRIBUTES
> COLOUR
> EDIT GROUP
> LIST GLOBALS

A GROUP of objects can be created by selecting GROUP. The Item Selector will appear showing all the Objects currently created. Any objects for inclusion within the group will be highlighted when selected (or deselected) with the mouse button. When all the Objects have been selected, click on the TICK in the Item Selector and all the highlighted objects will be included within the new GROUP.

CREATE OBJECT
Figure 8

FILE	GENERAL	AREA	**OBJECT**
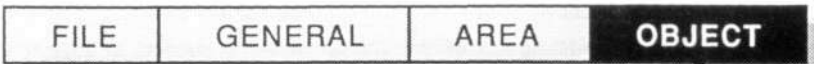
Name: **EDIT OBJECT**

Function: Edit a specified object.

Response: A list of the existing objects will be displayed.

Action: Select an object in the usual manner.

Response: A new bank of icons (See Figure 9 on the next page) will be displayed over the SHORTCUT icons. The icons are split into five groups:

> CREATE OBJECT
> **EDIT OBJECT**
> DELETE OBJECT
> SELECT OBJECT
> COPY
> CONDITION
> ATTRIBUTES
> COLOUR
> EDIT GROUP
> LIST GLOBALS

POINT: Alters to position of the point

35

EDIT OBJECT
Figure 9

number displayed in the INFO BAR. This function only applies to non rectangular facets and pyramids, in the case of facets all points may be moved whilst in the case of pyramids the two diagonally opposite apex points can be altered to change the point of the pyramid. The NEXT button is used to move to the next point to be edited.

TURN: Rotates the object in the direction of the arrows on the icons through 90 degrees.

SHRINK: Decreases the size of the object in the direction of the arrows.

STRETCH: Increases the size of the object in the direction of the arrows. As with MOVE the object cannot be stretched beyond the boundary of the area.

MOVE: Move the object in the direction of the arrows, left and right mouse button on these icons will have different effects depending on the values set in the PREFERENCES menu. If an object being moved hits another object of the edge of the area it will be butted against the obstruction.

To the right of the EDIT icons are three further icons as follows:

UNDO: This function will undo any editing made on an object prior to selecting another object or using the OKAY icon.

SELECT: This provides the option to select another object for editing as an alternative to clicking on another object within the VIEW window.

OKAY: Selecting this will commit all editing to memory and return to the main screen once more.

Note: Only triangular facets may be non-orthogonal; ie they may lie on a plane which is not aligned along one of the major axes (north/south, east/west, up/down).

36

FILE	GENERAL	AREA	OBJECT

Name: **DELETE OBJECT**

Function: Delete a specified object from memory.

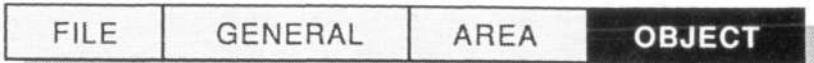*Response:* A list of objects will be displayed in the Item Selector.

Action: Select an object from the Item Selector in the usual manner.

Response: The object will be deleted from memory.

Note: This operation is irreversible, use with care!

```
CREATE OBJECT
EDIT OBJECT
DELETE OBJECT
SELECT OBJECT
COPY
CONDITION
ATTRIBUTES
COLOUR
EDIT GROUP
LIST GLOBALS
```

FILE	GENERAL	AREA	OBJECT

Name: **SELECT OBJECT**

Function: Choose a new selected Object for use in Highlight, Exclude or Lock.

Response: A list of objects will be displayed.

Action: Select one of the objects from the list.

```
CREATE OBJECT
EDIT OBJECT
DELETE OBJECT
SELECT OBJECT
COPY
CONDITION
ATTRIBUTES
COLOUR
EDIT GROUP
LIST GLOBALS
```

Name: **COPY**

Function: Create a duplicate of a specified object or group of objects.

Response: A list of objects will be displayed.

Action: Select the object from the Item Selector.

Response: A DIALOGUE BOX will be displayed.

Action: Select where the object is to be positioned in relation to the original object or "View" to place the object in front of your viewpoint.

Response: The new object will be created.

Note 1:   The name of the new object will default to its type followed by its number.

Note 2:   Any conditions entered for the original object will be copied to the new object also.

```
CREATE OBJECT
EDIT OBJECT
DELETE OBJECT
SELECT OBJECT
COPY
CONDITION
ATTRIBUTES
COLOUR
EDIT GROUP
LIST GLOBALS
```

Name: **CONDITION**

Function: To edit the conditions on a specified object.

Response: A list of all objects in the current Area will appear.

Action: Select an object in the usual manner.

Response: Any conditions on the object will be decompiled and displayed for editing in the usual manner (see TEXT EDITING).

```
CREATE OBJECT
EDIT OBJECT
DELETE OBJECT
SELECT OBJECT
COPY
CONDITION
ATTRIBUTES
COLOUR
EDIT GROUP
LIST GLOBALS
```

Name: **ATTRIBUTES**

Function: Alter the position and status of a specified object.

Response: A list of objects in the current Area will be displayed.

Action: Select an object from the list.

Response: A Dialogue Box will appear showing various information about the selected object - NAME, SIZE, POSITION, CURRENT STATUS, INITIAL STATUS and ANIMATED.

> CREATE OBJECT
> EDIT OBJECT
> DELETE OBJECT
> SELECT OBJECT
> COPY
> CONDITION
> **ATTRIBUTES**
> COLOUR
> EDIT GROUP
> LIST GLOBALS

NAME, POSITION and SIZE can be altered in the usual manner.

CURRENT STATUS alters the status of the object between VISIBLE, INVISIBLE and DESTROYED. An invisible object may be made visible at some other point in the environment whereas a destroyed object is gone until the environment is restarted using RESET.

INITIAL STATUS sets the state of the object when the environment is RESET, either VISIBLE or INVISIBLE.

MOVEABLE marks the object as being able to be animated, if you have any intention of animating this object it must be marked as MOVEABLE as this allows the START POSITION for the object to be set.

Sensors have additional attributes as follows:

RANGE shows the maximum distance at which you can be detected by the sensor.

SPEED alters the frequency at which checks are made for sensing or shooting.

DIRECTION shows from which directions the sensor can detect you.

EFFECT can be either SENSE, where the sensor will simply detect your presence, or SHOOT, when the sensor will shoot at you at the rate set by SPEED.

Action: Select OK to confirm changes or CANCEL to leave unchanged.

CONDITIONS

OBJECT

AREA

GENERAL

FILE

INTRODUCTION

39
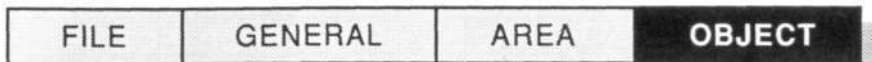
Name: **COLOUR**

Function: Colour objects in the current area.

Response: Initially a list of existing objects will be displayed.

Action: Select an object from the Item Selector.

CREATE OBJECT
EDIT OBJECT
DELETE OBJECT
SELECT OBJECT
COPY
CONDITION
ATTRIBUTES
COLOUR
EDIT GROUP
LIST GLOBALS

Response: A colour editing panel (See Figures 10, 11 & 12) will be displayed at the bottom of the screen displaying available colours. Base colours are combined to give various shades. The small "I" in the box for colour 0 indicates that colour 0 is invisible. Invisible facets are not drawn. Sides of objects that can never be seen should be coloured invisible to increase efficiency. To the right of the palette is a larger box showing the selected colour. Selecting the UNDO icon will undo the last colour change made. To the left of the screen is a display of all the selected object's colours.
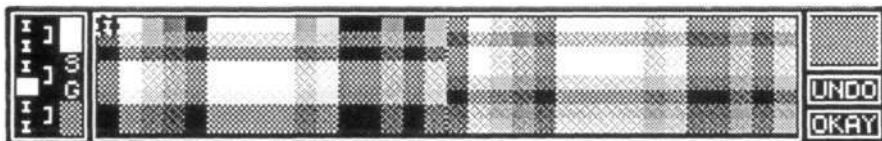
OBJECT COLOUR (ST & Amiga)
Figure 10
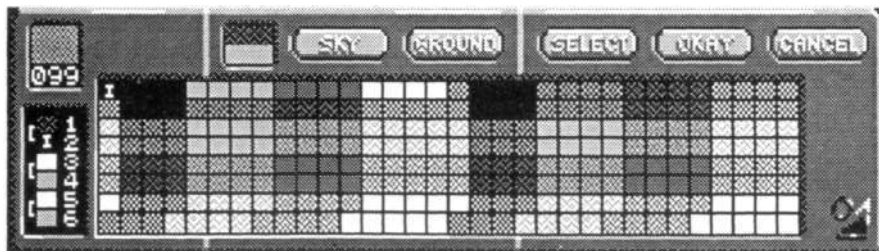
OBJECT COLOUR (PC EGA)
Figure 11

Action: Editing the colours of an object can be done in two ways. Firstly select a colour in the palette by clicking on it with the left mouse button. A flashing box will surround the colour to indicate it has been selected. Now move the cursor into the FREESCAPE VIEW window and click with the right

40

CONDITIONS

OBJECT

AREA

GENERAL

FILE

INTRODUCTION
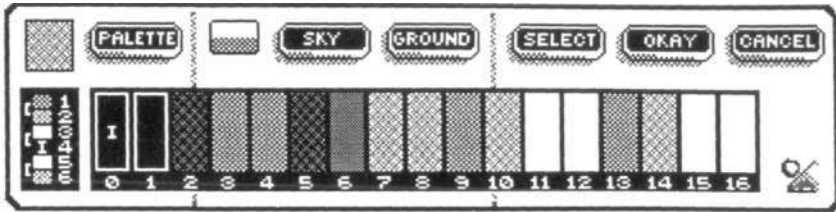
OBJECT COLOUR (PC CGA)
Figure 12

mouse button on the facet to be coloured. If this is on the currently selected object then it will simply change colour. If it is not, then that object will automatically become the selected object and the display to the left of the palette will change accordingly.

The second method is to select a colour in the palette as before and then click the right mouse button over the desired colour in the box to the left of the palette. The facet will change colour accordingly. This method allows the change of colour of objects which cannot be seen In the VIEW window either because they are out of view or because they have been made invisible. When colour editing is complete, click on the OKAY icon to return to the main screen.

Summary: The left mouse button = "Get colour"
            The right mouse button = "Put colour"

Note 1:   PC only - Colouring objects directly in the VIEW window is not possible.

Note 2:   PC CGA only - the palette icon will toggle between the two available palettes.

FILE	GENERAL	AREA	**OBJECT**
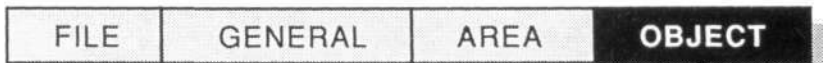
Name: **EDIT GROUP**

Function: To alter the contents of a specified group.

Response: A list of current group specifiers will be displayed.

Action: Select a group to be edited.

Response: A list of all the objects in the current area will be displayed. Members of the selected group will be highlighted.

Action: These objects can be removed or others included by clicking on the relevant object, either within the list or in the VIEW window.

CREATE OBJECT
EDIT OBJECT
DELETE OBJECT
SELECT OBJECT
COPY
CONDITION
ATTRIBUTES
COLOUR
EDIT GROUP
LIST GLOBALS

41

FILE	GENERAL	AREA	**OBJECT**

Name: **LIST GLOBALS** (Atari ST and Amiga only).

Function: Displays a list of defined objects available from the GLOBAL area. To view and possibly alter the presence of certain Global object in the current area.

Response: A list of the defined objects available from the GLOBAL area will be displayed with all displayed objects highlighted.

Action: These may be de-selected or you may select members of the list by clicking on them. When OK is selected all members highlighted (selected) will become visible in the current area.

Note 1: Global objects may only be edited when in Area 0.

Note 2: Global objects are a group of objects defined in area 0 that can be used in any number of areas using very little memory.

CREATE OBJECT
EDIT OBJECT
DELETE OBJECT

SELECT OBJECT
COPY
CONDITION
ATTRIBUTES
COLOUR

EDIT GROUP

LIST GLOBALS

THE FREESCAPE COMMAND LANGUAGE (FCL)

The FREESCAPE system contains a simple language definition allowing functions to be performed when certain conditions occur within the FREESCAPE environment. These commands can be used in any of 3 places:

OBJECT CONDITIONS: These commands are executed when some sort of interaction with the specified object takes place. The interaction options are: SHOT? - the object is pointed to by the mouse cursor and the left mouse button pressed, a series of lines will be seen from the corners of the screen to the object being shot, these lines depict a weapon (laser).

ACTIVATED? - the object is activated in a similar way to shooting an object except that the right mouse button is used and there is no visible effect. (The object has to be within the Activate distance as defined in the Defaults).

COLLIDED? - the object is collided with by either the player or an animated object.

AREA CONDITIONS: These commands are executed each frame while the viewpoint is within the confines of the specified area.

GENERAL CONDITIONS: These commands are executed every frame regardless of the viewpoint position.

In the following list, P1,P2 and P3 refer to parameters 1,2 and 3 respectively. These can be either a literal number or a variable. Variables are specified as V followed by a variable number 0..255 (eg V23 for variable 23). In this case the contents of the variable will be used as the parameter value. eg LOOP (P1) can be, for example: LOOP (6) (or any other number) or LOOP (V23) which uses the value stored in variable 23.

Parameters which must be variables are referred to as V1,V2,V3; eg:

SETVAR(P1.V2)

shows that the second parameter must be a variable.

Optional parameters or commands are surrounded by square brackets [].

A list of the available commands follows along with a description of the required parameters and their functions.

CONDITIONS

ACTIVATED? (ACT?)   Class - Trigger Interrogator.

Format: IF ACTIVATED?
     THEN commands...
     ELSE commands...
     ENDIF

Function :

This command checks whether the selected object has been activated. This happens when the cursor is over the selected object and the right mouse button is pressed.

Note: The selected object must be within the default activate range to be affected. This is set in the GENERAL menu, DEFAULT function.

Example IF ACTIVATED?
     THEN INVIS(4)
     ENDIF

43

CONDITIONS

OBJECT

AREA

GENERAL

FILE

INTRODUCTION

This condition simply informs the system that if the object is ACTIVATED then make object 4 invisible.

See also   IF, THEN, ELSE, SHOT?, COLLIDED?

ADDVAR (ADD)   Class - Variable Command

Format: ADDVAR (P1 ,P2)

Function:

This command performs an addition on the two supplied values, the value P1 is added to the value P2, If P2 is a variable specifier then the result of the addition is stored in the variable otherwise the result is lost but the CCR flags are still altered according to the result of the addition. Meaning that if an ADDVAR command is preceded by an IF and followed by a THEN/ELSE ENDIF combination, conditions may be executed depending on whether the result was zero or not without altering the value of any variables.

See also   SUBVAR

AGAIN   Class - Loop Command

Format: LOOP(P1)
             commands...
             AGAIN

Function:

This command serves to terminate a LOOP section. Upon reaching an AGAIN command the command processor will decrement the relevant LOOP counter and if the result is greater than zero, jump to the relevant LOOP address (the command immediately following the associated LOOP command).

Example   INCLUDE (1)
           START
           LOOP (20)
           MOVE (40,0,0)
           AGAIN
           RESTART

This is a simple example of using the LOOP command in animation. The rest of the commands will be explained fully later but for now the commands simply say: Include object 1 in the animation, Start the sequence when triggered, LOOP 20, move the object to the coordinates, AGAIN and restart.

See also   LOOP.

AND   Class - Condition Command

Format: IF condition
             AND condition
             THEN commands...
             [ELSE Commands...]
             ENDIF

Function:

This command combines the result of two or more condition checking commands and

44

returns TRUE only if all of the specified checks are TRUE otherwise a FALSE result is returned.

See also   IF, THEN, ELSE, ENDIF, OR

ANDV   Class - Variable Command

Format: ANDV (P1.P2)

Function:

This command performs a logical AND on the two values specified, the value P1 is ANDed with the value P2 and if P2 is a variable specifier the result is stored in the specified variable. CCR flags are set accordingly.

See also   ORV, NOTV

COLLIDED? (COL?)   Class - Trigger Interrogator

Format: IF COLLIDED?
                THEN commands...
                [ELSE Commands...]
                ENDIF

Function:

This command checks the COLLIDED flag in the status byte of the current object, a TRUE result is returned if a collision has occurred with this object since the last check, otherwise a FALSE result is returned. The COLLIDED flag on the current object is cleared upon executing this command.

Example   IF COLLIDED?
                THEN INVIS (4)
                VIS (5)
                ENDIF

In this condition the system checks if the object has been collided with. If it has then object 4 becomes invisible and object 5 becomes visible. This could be used to remove a door (object 4) and replace it with an open doorway (object 5).

See also   IF, THEN, ELSE, ENDIF, ACTIVATED?, SHOT.

DELAY   Class - Time Command

Format: DELAY (P1)

Function:

This command halts all FREESCAPE functions for the specified time. The specified time (P1) is in 50ths of a second.

Example   DELAY (50) would halt execution for 1 second.

See also   WAIT.

DESTROY   Class - Object Commands

Format: DESTROY (P1 [,P2])   {object [,area]}

Function:

This command sets the DESTROYED flag on the specified object (P1) in the

45

CONDITIONS

OBJECT

AREA

GENERAL

FILE

INTRODUCTION

specified area (P2). If no area is specified the command processor presumes that the specified object is in the current area. Note - Once an object has been destroyed it is then impossible to get the object back short of reseting.

Example   IF SHOT?
               THEN DESTROY (4,2)
               ENDIF

This simply asks if the current object has been shot and if so destroy object 4 in area 2.

See also   DESTROYED?

DESTROYED?   Class - Object Interrogator

Format: IF DESTROYED? (P1 [,P2])   { object [,area]}
               THEN Commands...
               [ELSE Commands...]
                ENDIF

Function:

This command checks the status of the specified object and returns a TRUE result if the object has been DESTROYED.

See also   IF, THEN, ELSE, ENDIF, DESTROY

ELSE   Class - Conditional Statement

Format   IF condition
               THEN commands...
               ELSE Commands...
               ENDIF

Function:

This command exists only as part of an IF/THEN/ELSE/ENDIF combination. It marks the start of commands to execute only if the result of a previous condition was FALSE. The effectiveness of the command relies on the correct usage of the IF and THEN

commands. For any Condition checking to work it is essential that the Condition be preceded by an IF command and followed by a THEN and (if required) an ELSE statement.

See also   IF, THEN, ENDIF

END   Class - Condition Command

Format: IF condition
               THEN Commands...
               END
               [ELSE Commands...]
               ENDIF
               Commands............

Function:

This command exits command processing before the end of the command list is reached, it allows the user to cut short the command execution on a particular

condition being TRUE or FALSE. Used in the above format, if the result of the Condition is true only the commands following the THEN statement will be executed and upon reaching the END command the processor would stop processing commands from this list. Were there no END command the processor would continue executing from the command following the ENDIF statement.

Note: If END is used within an animator the execution of the current animation frame is ENDed and execution continues on the next frame beginning with the command following the END command.

See also   IF, THEN, ELSE, ENDIF

ENDGAME   Class - Player Command

Format: ENDGAME

Function:

This command serves to reset the environment. This can be executed on a particular condition being TRUE or FALSE, i.e. if a counter being used to store game time reaches zero then the game finishes.

Example: IF COLLIDED?
             THEN ENDGAME
             ENDIF

This condition simply states that if the player or another animated object collides with the selected object then end the game and reset all the flags etc.

ENDIF   Class - Condition Statement

Format: IF Condition
             THEN commands...
             [ELSE Commands...]
             ENDIF

Function:

This command terminates a conditional section. Upon reaching an ENDIF command, execution continues as normal before the IF/THEN/ELSE combination. If the result of a Condition is TRUE the commands after the THEN statement are executed and those between the ELSE statement and the ENDIF are ignored. If the result is FALSE the commands between the THEN and the ELSE are ignored and those between the ELSE and the ENDIF are executed. In either case unless an END command has been issued, command processing will continue after the ENDIF statement.

See   IF, THEN, ELSE

EXECUTE (EX)   Class - Branch Command

Format: EXECUTE (P1)   {object}

Function:

This command terminates command execution on the current object and continues with the command list on object (P1). The status flags and the position of the original object are still used for Object Interrogator commands.

47

GOTO   Class - Player Command

Format: GOTO (P1 [,P2])   {entrance [.area]}

Function:

This command is used to allow player movement between the various defined areas. Upon reaching this command the player will be moved to the ENTRANCE P1 in the AREA P2. If no area is specified the entrance is presumed to be in the current area. If a new area is specified, command processing will cease at this point otherwise normal command processing will continue.

Example   IF COLLIDED?
               THEN GOTO (1,2)
               ENDIF

The above example would be quite useful if it was desired that the player, upon colliding with a doorway (the object selected) would then be transported to Entrance 1 in Area 2.

IF   Class - Condition Statement

Format: IF Condition
                 THEN commands...
                 [ELSE Commands...]
                 ENDIF

Function:

This command marks the start of a condition section. Immediately following the IF statement should be one or more condition commands separated by either AND or OR statements. The IF command simply serves to clear the CCR flags and prepare for the following condition. To have any effect at all the Condition should be followed by a THEN/ELSE combination otherwise execution will continue after the Condition regardless of the result.

See also   THEN, ELSE, ENDIF, AND, OR

INCLUDE   Class - Animation Command

Format: INCLUDE (P1)   {object}

Function:

This command is animation specific. Any attempt to execute it on an OBJECT or in LOCAL/GLOBAL conditions will have no effect. The command includes the specified object (if it is not already animated) into the animation list for the current animator. This command should be used at the very beginning of an animation before the START command so that is is only called once at the start of the animation and never again until the environment is restarted.

See also   MOVE, START, RESTART

INVIS (IV)   Class - Object Command

Format: INVIS (P1 [,P2])   {object [.area]}

Function:

This command sets the INVISIBLE flag on object P1 in the specified AREA P2. If no

48

area is specified the object is presumed to be in the current area.

Example   IF SHOT?
             THEN INVIS (9)
             ENDIF

A simple condition which states that if the specified object is shot then object 9 will become invisible.

See also   INVIS? VIS? VIS

INVIS?   Class - Object Interrogator

Format: IF INVIS? (P1 [,P2])   {object [.area]}
              THEN Commands...
              [ELSE Commands...]
              ENDIF

Function:

This command checks the INVISIBLE flag in the status byte of OBJECT P1 in AREA P2. If no area is specified then the object is presumed to be in the current area. The command returns a TRUE result if the specified object is INVISIBLE, otherwise a FALSE result is returned.

See also   INVIS, VIS, VIS?

LOOP   Class - Loop Command

Format: LOOP (P1)   { loop count}

Function:

This command marks the start of a LOOP section. The commands between the LOOP and the corresponding AGAIN command will be executed P1 times.

See also   AGAIN

MODE   Class - Player Command

Format: MODE (P1)   {movement mode}

Function:

This command alters the current movement mode of the player. In the game the player is restricted to WALK, FLY1 and FLY2. The CAMERA modes and LOCK modes are only available in the EDITOR, therefore the value of the new mode P1 must be in the range 1-3. Any value above this will be interpreted as 3 and any less than 1 will be interpreted as 1.

See also   GOTO

MOVE   Class - Animation Command

Format: MOVE (P1 ,P2,P3)   {x,y,z coordinates}

Function:

This command is animation specific, any attempt to execute this command on an OBJECT or LOCAL/GLOBAL conditions will have no effect. The command MOVEs the members of the current animation (specified at the beginning using

49

the INCLUDE command) by the specified amount in the X, Y and Z axis.

See also   INCLUDE, MOVETO

MOVETO   Class - Animation Command

Format: MOVETO (P1 ,P2,P3)   { x,y,z coordinates}

Function:

This command is animation specific, any attempt to execute this command on an OBJECT or LOCAL/GLOBAL conditions will have no effect. The command MOVEs the members of the current animation (specified at the beginning using the INCLUDE command) to the specified position in the X,Y and Z area.

Example   INCLUDE (3)
             START
             MOVETO (2900,0260,4760)
             RESTART

This condition, when triggered will move object 3 to the coordinates specified in the brackets following the command MOVETO.

See also   INCLUDE, MOVE

NOTV   Class - Variable Command

Format: NOTV(P1)

Function:

This command performs a logical NOT on the value specified, the value P1 and the result is stored in the specified variable. CCR flags are set accordingly.

See also   ANDV, ORV

OR   Class - Condition Command

Format: IF Condition
             OR Condition
             THEN Commands...
             [ELSE Commands...]
             ENDIF

Function:

This command combines the result of two or more condition checking commands and returns TRUE if any of the specified checks are TRUE otherwise a FALSE result is returned.

See also   IF, THEN, ELSE, ENDIF, AND

ORV   Class - Variable Command

Format: ORV(P1,P2)

Function:

This command performs a logical OR on the two values specified, the value P1 is ORed with the value P2 and if P2 is a variable specifier the result is stored in the specified variable. CCR flags are set accordingly.

Example   IF SHOT?
          THEN ORV(2,V21)
          ENDIF

This uses Bit 2 of Variable V21 as a flag to say that an object has been shot. Using this method it is possible to use a Variable to store a number of ON/OFF flags. The flags can be checked using the ANDV command.

Example   IF ANDV (V21,2)
          THEN Commands...
          [ELSE Commands...]
          ENDIF

By "ANDing" V21 with 2 and not the other way round the AND is executed without storing the result, therefore it is possible to check the state of the flags without affecting them.

See also   ANDV, NOTV

GETXPOS,GETYPOS,GETZPOS   Class - Object Interrogator

Format: GETXPOS (V1 ,P2 [,P3])   {variable,object[,area]}
          GETYPOS(V1,P2[,P3])
          GETZPOS(V1,P2[,P3])

Function:

These commands store the position of the specified object P2, in area P3 along the X, Y or Z axis in the specified variable V1. If no area is specified, the current area is assumed.

Example   GETXPOS (V21,2)
          IF VAR=?(V21,1000)
          THEN SOUND (2)
          ENDIF

This will get Object 2's X position and will perform a sound only if Object 2 is at position 1000 in the X axis.

PRINT   Class - Instrument Command

Format: PRINT   ("message...",P1)   {message,instrument}

Function:

This command allows the user to print a message to a defined TEXT WINDOW type instrument (see INSTRUMENTS). The message between the quotation marks is printed to the instrument number P1 if the instrument exists and if it is a TEXT WINDOW type. The message can be split into several lines (if the TEXT WINDOW is big enough) by using \N to begin a new line.

RESTART   Class - Animation Command

Format: RESTART

Function:

This command is animation specific, any attempt to execute it on an OBJECT or in LOCAL or GLOBAL conditions will have no effect. After executing this command execution of the animation will continue at the position set by the

51

START command. If no START command has been executed the RESTART command will set execution to continue from the start of the animation.

See also   START

REDRAW   Class - Instrument Command

Format: REDRAW

Function:

This command will force an immediate redraw of the FREESCAPE view window. Any objects whose status have changed since the last frame update will be displayed in their new state.

Example   LOOP (10)
              TOGVIS *(2)*
              REDRAW
              AGAIN

This will toggle the visibility of Object 2 ten times and REDRAW the FREESCAPE view each frame.

REMOVE   Class - Animation Command

Format: REMOVE (P1)   {object}

Function:

This command works in the opposite way to INCLUDE. The object specified P1 will be removed from the animation. This command can be incorporated into the animation controller e.g. to remove objects from the animation one at a time during animation. This command may only be used in animation.

SOUND   Class - Sound Command

Format: SOUND (P1)   {sound number}

Function:

This command will immediately perform the sound P1.

See also   SYNCSND

SETVAR (SET)   Class - Variable Command

Format: SETVAR (P1 ,V2)

Function:

This command sets the variable V2 to the value P1. If V2 is not a variable specifier then the command has no effect.

SHOT?   Class - Trigger Interrogator

Format: IF SHOT?
                THEN Commands...
                [ELSE Commands...]
                ENDIF

Function:

This command checks the SHOT flag in the status byte of the current object. If the object has been shot since the last time checked then the command returns a TRUE result otherwise a FALSE result is returned. Execution of this command also clears the SHOT flag on the current object.

See also   ACTIVATED?, COLLIDED?

START   Class - Animation Command

Format: START

Function:

This command is animation specific, any attempt to execute it on an OBJECT or LOCAL or GLOBAL conditions will have no effect. The command marks the start of the animation command list. The instruction after the START command will be the point at which the RESTART command will continue execution from. The START command should be placed after any INCLUDE command as INCLUDES after the START will be executed each time through the animation loop, this wastes time and has no useful effect.

See also   INCLUDE, RESTART

STARTANIM   Class - Animation Command

Format: STARTANIM (P1[,P2J)   {animator [,area]}

Function:

This command will start an animation controller going. At the beginning of a game all animation controllers are marked as STOPPED. To begin the animation a STARTANIM command must be executed. The STARTANIM command will also re-enable an animation controller which has been stopped using the STOPANIM command.

Example   IF COLLIDED?
           THEN STARTANIM (2)

This condition was placed on a selected object. If the object is collided by the player then start the second animation controller (2).

See also   STOPANIM, TRIGANIM, WAITTRIG

STOPANIM   Class - Animation Command

Format: STOPANIM (P1 ,[P2])   {animator [,area]}

Function:

This command will stop an animation controller, no commands will be executed on the controller until is is started using the STARTANIM command. Upon receiving a STARTANIM command the animation controller will continue execution from the point at which the STOPANIM command was received.

See also   STARTANIM, TRIGANIM, WAITTRIG

SUBVAR (SUB)   Class - Variable Command

Format: SUBVAR (P1,P2)

Function:
This command performs a subtraction on the two supplied values, the value P1 is subtracted from the value P2. If P2 is a Variable specifier then the result of the subtraction is stored in the variable otherwise the result is lost but the CCR flags are still altered according to the result of the subtraction. Therefore if a SUBVAR command is preceded by an IF and followed by a THEN/ELSE ENDIF combination, conditions may be executed depending on whether the result was zero or not without altering the value of any variables.
See also   ADDVAR, SETVAR

SYNCSND   Class - Sound Command
Format: SYNCSND (P1)   {sound number}
Function:
This command will execute the specified sound P1 in sync with the next complete frame update. Note the REDRAW command will also perform a synchronised sound.
See also   SOUND

THEN   Class - Condition Statement
Format: IF Condition
               THEN Commands...
               [ELSE Commands...]
               ENDIF
Function:
This command checks the status of the ZERO flag in the CCR. If the contents are TRUE then the commands following the THEN statement are executed until either an ELSE or ENDIF statement is found. If an ELSE is found the commands following it are ignored up until an ENDIF or the end of the command list. If an ENDIF is found then normal command execution will continue with the following command. The THEN command is the only command which examines the result of a condition, so an IF ELSE ENDIF combination without a THEN command will produce incorrect results.
See also   IF, ELSE, ENDIF, AND, OR

TIMER?   Class - Trigger Interrogator
Format: IF TIMER?
               THEN Commands...
               [ELSE Commands...]
               ENDIF
Function:
This command checks the TIMER flag, the command returns a TRUE result if a timelapse of the amount specified in the defaults setup section has passed, otherwise a FALSE result is returned. This command is only really useful in LOCAL and GLOBAL conditions as these are the only conditions which are executed each frame, any TIME commands on OBJECTS will only be checked when some form of interaction takes place with the object.

TOGVIS (TOG)   Class - Object Command
Format: TOGVIS (P1 [,P2])   { object [,area]}

Function:

This command toggles the status of the VISIBLE flag in the status byte of object P1 in area P2. If no area is specified the object is presumed to be in the current area.

See also   VIS, INVIS, VIS?, INVIS?

TRIGANIM   Class - Animation Command
Format: TRIGANIM (P1)   {animator}
Function:

This command sets the TRIGGER flag in the status byte of animation controller P1. A WAITTRIG command within the animation controller will register this trigger. If no WAITTRIG commands exist in the animation controller a TRIGANIM command will have no effect on this animator.

See also   STARTANIM, STOPANIM, WAITTRIG

UPDATEI   Class - Instrument Command
Format: UPDATEI (P1)   {instrument}
Function:

To Update Instrument (P1) in the Test Screen.

VAR=? (V=?)   Class - Variable Command
Format: IF VAR=? (P1.P2)
                 THEN Commands...
                 [ELSE Commands...]
                 ENDIF

Function:

This command will compare the values of the P1 and P2 and return a TRUE result if the values are equal otherwise a FALSE result is returned.

See also   SETVAR, ADDVAR, SUBVAR, VAR>?, VAR <?

VAR>? (V>?)   Class - Variable Command
Format: IF VAR>? (P1,P2)
                 THEN Commands...
                 [ELSE Commands...]
                 ENDIF

Function:

This command will compare the values of P1 and P2 and return a TRUE result if the value of P1 is greater than that of P2, otherwise a FALSE value is returned.

See also   SETVAR, ADDVAR, SUBVAR, VAR=?, VAR<?

VAR<? (P1,P2)   Class - Variable Command
Format: IF VAR<? (P1,P2)
                 THEN Commands...
                 [ELSE Commands...]
                 ENDIF

55

CONDITIONS

OBJECT

AREA

GENERAL

FILE

INTRODUCTION

Function:
This command will compare the values of P1 and P2 and returns a TRUE result if the value of P1 is less than that of P2, otherwise a FALSE result is returned.

See also   SETVAR, ADDVAR, SUBVAR, VAR>?, VAR=?

VIS (V)   Class - Object Command
Format: VIS (P1 [,P2])   {object [,area]}
Function:
This command clears the INVISIBLE flag on OBJECT P1 in the specified AREA P2, making it visible. If no area is specified the object is presumed to be in the current area.

See also   INVIS?, VIS?, INVIS, TOGVIS

VIS?   Class - Object Interrogator
Format: VIS? (P1 [,P2])   {object [,area]}
Function:
This command checks the INVISIBLE flag in the status byte of OBJECT P1 in AREA P2. If no area is specified then the object is presumed to be in the current area. The command returns a TRUE result if the specified object is VISIBLE, otherwise a FALSE result is returned.

See also   VIS, INVIS, TOGVIS, INVIS?

WAIT   Class - Time Command
Format: WAIT
Function:
This command halts processing of the current command list and stores information about the current command list on an internal stack. The FREESCAPE processing is then allowed to continue, processing any more required conditions, animations and player movements, when the next frame comes round execution of the command list will continue from the command following the WAIT command.

See also   DELAY

WAITTRIG   Class - Animation Command
Format: WAITTRIG
Function:
This command is animation specific, any attempt to execute it on an OBJECT or in LOCAL or GLOBAL conditions will have no effect. The command will check the TRIGGER flag in the status byte of the animation controller. If the flag has been set by use of the TRIGANIM command, the flag will be cleared and execution will continue as normal, otherwise execution will be stopped at the WAITTRIG command and the execution of the animation command list will be stopped. Upon reaching the current animation controlleron the next frame the WAITTRIG command is the first to be executed, therefore the execution of the animation command list is halted at the point of the WAITTRIG command until a TRIGANIM command sets the TRIGGER flag.

See also   TRIGANIM, STARTANIM, STOPANIM

THE ANIMATION CONTROLLER

In addition to the COMMAND language, the the FREESCAPE system includes a further system for object control, namely the ANIMATION OBJECT CONTROLLER. The ANIMATION OBJECT CONTROLLER (AOC) provides a means of joining a number of objects together and performing movement and animation functions on these objects. This means that to move a car (for example) it is not necessary to move each element of the car individually for each stage of the movement path but simply to join all relevant objects that make up the car in an AOC and then every MOVE command in the AOC command list after the INCLUDE list will affect all of the objects.

To animate an object the object must first be marked as MOVEABLE, this can be done by entering the OBJECT ATTRIBUTES dialogue box, toward the bottom of the dialogue box is a button titled ANIMATION. The initial state of this button will be STATIC. Below the ANIMATION field iN the ATTRIBUTES dialogue box is a START POSITION text field, while the object is marked as static this field will simply contain the message REFER POSITION, this means that as the object is not MOVEABLE it can never be moved, therefore its start position is equal to its current position. To mark the object as MOVEABLE simply click once on the STATIC button, the button will change to show the message MOVEABLE and the START POSITION field will change to show the object's current position. The START POSITION of a MOVEABLE object can be changed in the same way you would its position and size.

Any attempt to animate a STATIC object will be ignored (if there are problems animating an object for any reason it is always advisable to check the animation state of the object first).

AOC commands are executed every frame, all commands in the AOC list will be executed in order until either the end of the command list is encountered or a redraw is requested. Upon encountering a redraw request program execution will stop and the current program position will be stored. Program execution on the AOC will then recommence from that position on the next frame. If the end of the command list is encountered the AOC is marked as STOPPED and can only be used again if a STARTANIM command resets its internal program counter and marks it as STARTED.

Certain commands will, if called from an AOC, force a redraw i.e. MOVE, MOVETO, and END. REDRAW should not be used within an AOC, since it will do a redraw, then exit, forcing another redraw. Therefore any other commands you wish to have executed before the next frame update must be placed before that command. A description of the commands available from within an AOC may be found in the FREESCAPE COMMAND LANGUAGE section of this manual.

It is worth noting that although the animated objects will collide with other objects in the dataset, a group of objects will behave like a single large object, even if they occupy only a relatively small area. For example, an animator controlling two small objects, one at each edge of an area, will not be able to move them past a tall object in between them. The objects are effectively grouped together in a large object that stretches between them, and this will collide with the object in the centre. In this case, it is necessary to use two animation controllers to move the objects individually.

57

EXAMPLES

TO GO TO ANOTHER AREA

As an example we will use object 3 which is our DOOR and object 4 which is our DOORWAY. For simplicity the doorway is a black RECTANGLE which is placed close against a wall and the door is a red CUBE which has been "flattened" by the use of the EDIT OBJECT tools and placed close up in front of the doorway. The DOORWAY (rectangle) should be set to INVISIBLE via the OBJECTS ATTRIBUTES function both on START STATUS and PRESENT STATUS. We will use the ACTIVATED? command to "open" the door and reveal the doorway as follows:

Enter the following condition commands for OBJECT 3 by selecting the CONDITION icon and selecting OBJECT 3 from the list by clicking with the mouse button until object 3 is highlighted and then selecting the TICK icon. Now enter the following:

```
            IF ACTIVATED?
            THEN INVIS (3)
            VIS (4)
            ENDIF
```

Now experiment by clicking the RIGHT MOUSE BUTTON on the door in the VIEW window. The door (object 3) should vanish and be replaced by the doorway (object 4).

Now enter the following condition commands for OBJECT 4 in the same way as above and enter the following:

```
            IF COLLIDED?
            THEN GOTO (1,2)
            ENDIF
```

Now try walking towards the "doorway" until you collide with it. You will be transported instantly to ENTRANCE 1 in AREA 2.

TO MAKE AN OBJECT INVISIBLE OR VISIBLE

As can be seen by the previous example, making objects vanish and reappear is a very simple matter. If, for example, we wish an object to become invisible when it is shot we would select the object by clicking on the CONDITION icon, selecting the object from the list (or by clicking on the object in the VIEW window if it is visible). Then the following conditions should be entered:

```
            IF SHOT?
            THEN INVIS (o)
            ENDIF
```

The number in the brackets following the INVIS is the object number.

TO MAKE A SOUND

There are several different sound effects within the FREESCAPE system. One example would be to have a sound effect when a piece of treasure is picked up by the player. The treasure we will refer to as OBJECT 4. The following commands will play a sound as the player activates the object and the object vanishes from the VIEW window. Select the CONDITION icon and select OBJECT 4 from the list (having, of course, previously created the object).

Now enter the following conditions:

> IF ACTIVATED?
> THEN INVIS (4)
> SOUND (5)
> ENDIF

Experiment with different sounds by changing the number in the brackets following the SOUND command and clicking the right mouse button on object 4 in the VIEW window to hear the effect.

TO USE A LOOP

At various times during game creation the LOOP command is needed. One example of its use would be within an ANIMATION. The following example will show how to animate an object and the use of the LOOP command. Clear all data from the VIEW window and create a cube. This will be OBJECT 2 (Cuboid 2) as CUBOID 1 is the base of the area. Select CREATE ANIMATION from the AREA MENU at the top of the screen. There will be no visible response but if EDIT ANIMATION is then selected it will be seen that ANIMATION 1 has been created ready for use. For the moment select ATTRIBUTES from either the OBJECT menu or from the SHORTCUT icons then select OBJECT 2. A dialogue box will appear showing the ATTRIBUTES for OBJECT 2. Click on the window where you see STATIC until the word MOVEABLE appears. Note that to animate an object it must always be defined as moveable first. Now select EDIT ANIMATION, select animator number 1 from the list shown and enter the following conditions:

> INCLUDE (2)
> LOOP (20)
> MOVE (40,0,0)
> AGAIN

We now need something to trigger the animation so we will select the floor which is CUBOID 1. Select the CONDITIONS icon and select CUBOID 1 in the usual manner. Now enter the following:

> IF SHOT?
> THEN STARTANIM (1)
> ENDIF

Now shoot the floor to see the results!

TO CREATE AN ANIMATION

As in the previous example HOW TO USE A LOOP it will be seen that animation is a very simple procedure. For this example we will attempt to move the cube directly to another position on the VIEW window. Create the cube and define it as MOVEABLE as in the previous example and create an animator for our commands. Now select EDIT ANIMATION and enter the following:

> INCLUDE (2)
> START
> MOVETO (4560,0200,4760)
> RESTART

This will be activated in the same way as the previous example by shooting

59

CONDITIONS

OBJECT

AREA

GENERAL

FILE

INTRODUCTION

the floor. Try shooting the floor to see the cube transported to another position to the right of the VIEW window and slightly lower.

Now we will attempt something a little more sophisticated. We will attempt to animate the cube to glide from one side of the VIEW window to the other and back again. To save time we will edit the existing conditions. Select EDIT ANIMATION and reselect the same animator to edit. Edit the conditions to read as follows:

```
INCLUDE (2)
START
LOOP (20)
MOVE (40,0,0)
AGAIN
LOOP (20)
MOVE (-40,0,0)
AGAIN
RESTART
```

Now shoot the floor and see the result! Note that the LOOP is repeated after the first AGAIN command and that the MOVE has been modified by a minus before the 40. This is to move the cube in the reverse direction to the first loop. Experiment a while with the coordinates after the MOVE Command and see what happens.

HOW TO USE VARIABLES

The format for using a VARIABLE can be handled in the same way through various types of conditions, on an OBJECT condition we could, for example, arrange for a variable to be increased to hold a higher value when it is shot, as follows:

```
IF SHOT?
THEN ADDVAR(25,V21)
```

Thus adding 25 to the VARIABLE number 21. In a similar way a value can be deducted from a VARIABLE using the following example:

```
IF SHOT?
THEN SUBVAR(15,V21)
```

To set a VARIABLE to hold a specified number we could use the following GENERAL condition Commands:

```
SETVAR (600,V21)
```

This same process can be incorporated into slightly morecomplicated conditions where we want to check the value of the variable and then if TRUE to set the variable to hold another value as follows:

```
IF VAR>?(0,V21)
THEN SETVAR (3000,V21)
```

Thus if Variable 21 holds a value greater than 0, Variable 21 will be set to hold the value 3000.

MORE ABOUT VARIABLES

The use of VARIABLES enables you to create a wide range of conditions, from the very simple to the complicated. The system has 256 VARIABLES available for use by the COMMAND LANGUAGE. These VARIABLES are 32 bit storage areas (that is they can hold numbers in the range -2147483646 to +2147483647) which can be used

CONDITIONS

OBJECT

AREA

GENERAL

FILE

INTRODUCTION

to store and manipulate various numerical values within the environment, eg player score, object position, fuel supply or a timer. The first 30 (0-29) of these VARIABLES are used by the FREESCAPE II system. The contents of these VARIABLES are updated each frame by the system, and any changes to the VARIABLES are so noted by the system. Ie. if a variable command were to change the value stored in Variable V0 (the viewpoint X position) the next displayed frame would move the player to the new specified X position. A list of the contents of the system VARIABLES follows:

                    00 Viewpoint X position
                    01 Viewpoint Y position
                    02 Viewpoint Z position
                    03 Viewpoint X rotation
                    04 Viewpoint Y rotation
                    05 Viewpoint Z rotation
                    06 Current vehicle type
                    07 Current height (WALK only)
                    08 Current Area number
                    09 Number of last Area visited
                    10 Distance fallen above max ability
                    11 Number of times shot
                    12 Number of times crushed
                    13 Number of last SENSOR (detect only) to find you
                    14 Number of times SENSED (detect only)
                    15 ASCII code of last key pressed
                    16 Button status at last press (1-LEFT,2-RIGHT,3-BOTH)
                    17 Mouse X position at last press
                    18 Mouse Y position at last press
                    19 50Hz counter for accurate timing
                    20 Player firing control (see below)
                    21 Number of shots fired
                    22..29 Reserved

The player firing control variable allows optional control of the player's ability to shoot. Putting 0 into this variable disables the player's shooting completely. A value of 1 enables shooting. Adding 2 draws lines from the edges of the screen to the point of firing, and adding 4 enables the firing sound.

Finally, adding 8 allows rapid fire - holding down the firing button releases a continuous stream of shots. So, to enable firing and rapid fire, but with no lines or sound, a value of 1 + 8 (enable + rapid fire) = 9 should be placed in this variable. At the start of the game, it is set to 15 (enabled, lines, sound and rapid fire), so it would be necessary to include a SETVAR(V20,9) in the startup condition to override this.

Parameters are passed to the commands in brackets following the command itself, the number of parameters required by the command varies and some have optional parameters, in the case of commands with optional parameters (mostly object commands, where the area number is usually optional) the optional parameter is usually the last one in the list. All numeric parameters may be given as either an absolute value in the range -16383 to +16384 or as a variable specifier in which case the value must have a V preceding it and is restricted to the range 0 to 255. If a parameter is given as a variable specifier then the contents of the given variable are used as the parameter.

Note 1:

In the defaults setup dialogue box there is an option to set an initial condition number, this number refers to GENERAL conditions and allows the user to have any of the defined global conditions executed only once immediately after reset. This condition will then be ignored from then on.

Note 2:

VARIABLE 255 is not cleared when the environment is reset and as such could be very useful when a Hi-Score counter is required.

SOUND EFFECTS
(PC)

The sound bank on the PC consists of 20 sounds, defined as follows:

       1: Beep
       2: Buzz
       3: Sensor shooting
       4: Player shooting
       5: Step up
       6: Step down
       7: Fall too far
       8: Door open
       9: Door close
      10: Double tone
      11: Rising tone
      12: Activate
      13: Bonus 1
      14: Bonus 2
      15: Bonus 3
      16: Anti-bonus
      17: Chink
      18: Drone
      19: Lift up
      20: Bonus 4

SOUND EFFECTS
(Atari ST and Amiga)

The Standard Sound Bank allows for 32 sounds. Sounds 0 to 6 are already defined:

       0. Laser Out   **(PLAY SAMPLE 1)**
       1. Shooter   **(PLAY SAMPLE 2)**
       2. Bump   **(PLAY SAMPLE 3)**
       3. Explosion   (PLAY SAMPLE 4)
       4. Ping   **(FIXED SOUND)**
       5. Smash   (PLAY SAMPLE 5)
       6. Clang   (PLAY SAMPLE 6)
       7. UNDEFINED   **(PLAY SAMPLE 7)** If exists!

      31   UNDEFINED   **(PLAY SAMPLE 31)** If exists!

CREATING A NEW SAMPLE BANK

(ST and AMIGA only)

The format for the FREESCAPE II samples are as follows:

> 0000   Length
> 0004   Playback rate (ignored)
> 0006   8 bit sample data

Samples saved in RAW 8 bit sample format from most samplers is similar to the above (note the contents of the playback rate is not important as it is ignored anyway). The sample BANK is made up of a number of these samples joined together as a linked unit (one immediately following another). A small utility is supplied called JOIN (JOIN.TTP on the Atari ST) to concatenate a number of sample files together to create a new sample bank. The command line supplied to the JOIN command must be of the format:

<Sample_1_name>+<Sample_2_name>+<Sample_?_name> <Sample_BANK_name>

Note there are no spaces between each sample name but the space between the sample list and the BANK name is essential.

To pass a command line to the program on the Amiga, simply type

JOIN [Command Line] from the CLI.

On the Atari ST, double click on the JOIN.TTP icon and when a text input box appears requesting the parameters for the program type in the command line at the prompt.

Once a new sample bank has been created it can be used in the CONSTRUCTION KIT by passing the KIT a command line of the form:

3DKIT -s<Sample Bank Name>

The sample bank name must follow the -s option immediately with no spaces.

Note:
To be able to pass a command line to the Atari ST version of the CONSTRUCTION KIT it will be necessary to RENAME the 3DKIT.PRG file to 3DKITTTP. Having done this, when you double click on the 3DKIT.TTP icon a text input requester will appear and you may then type in the command line.

If there is not enough memory to load a new, larger sample bank within the CONSTRUCTION KIT it may still be possible to use the new bank on a runnable game (as there is more available memory in a runnable environment as the EDITOR is not used). This can be achieved by MAKEing the runnable environment with the standard sample bank, there will be a file called GAME.SAM (where GAME is the name of the stand alone environment). Replace this file with the new sample bank (giving it the same name i.e. GAME.SAM) and it will then be loaded as the environment's sample bank.

63

APPENDIX

DEFAULT KEY CONTROLS

O MOVE FORWARD	**Q** TURN LEFT	**K** MOVE BACK
H MOVE LEFT	**P** LOOK UP	**J** MOVE RIGHT
R MOVE UP (RISE)	**N** TILT LEFT	**F** MOVE DOWN (FALL)
W TURN RIGHT	**I** FACE FORWARD	**1** SAVE GAME POSITION
L LOOK DOWN	**U** U-TURN	**2** LOAD GAME POSITION
M TILT RIGHT	**A** ACTIVATE OBJECT	**ESC** QUIT GAME
B SELECT MODE	**V** CHANGE MODE (WALK/FLY)	**C** CENTRE CURSOR ON/OFF

SPACE FIRE!

RECOMMENDED ART PACKAGES

AMIGA,ST and PC - DeLuxe Paint I, II and III.
ST Only - Degas Elite, Neochrome.

HINTS AND TIPS

1 Save regularly.
2 Have blank formatted disks ready for saving data.
3 Always mention the Construction Kit release number in any correspondence. This can be accessed from the ABOUT function in the FILE menu.
4 Colour sides of objects that can never be seen to colour 0 (Invisible) to increase performance.
5 Care should be used when entering Conditions as an infinte loop could be created effectively causing a crash. If in doubt save your data to disc before testing a procedure you are unsure of.

DISK CONTENTS

The disk contents are described in the README file on the actual disk. To view this file, type README at the DOS prompt, or run README or README.PRG from the desktop. This file will also detail any changes to the manual, errata, etc.

POSSIBLE ATARI ST DISK ERROR MESSAGES

ERROR -2: Drive not ready.   There is no disk in the specified drive.
ERROR -7: Not a valid disk.   The disk is not an ATARI DOS disk.
ERROR -10: Write error.   Basic error in writing to the disk.
ERROR -11: Read error.   Basic error in reading from the disk.
ERROR -12: General error.   Usually write protected disk.
ERROR -13: Write protect.   Disk is write protected.
ERROR -33: File not found.   The file specified does not exist.
ERROR -34: Path not found.   The path specified does not exist.
ERROR -36: Access not possible.   The file has been protected against deletion or writing, or it is a directory.
ERROR -46: Invalid drive.   The specified disk drive does not exist.

POSSIBLE AMIGA DISK ERROR MESSAGES

ERROR 202: In use.   Another program is already using this file or directory.
ERROR 204: Path not found.   The path specified does not exist.
ERROR 205: File not found.   The file specified does not exist.
ERROR 210: Invalid filename. The path or filename specified contains invalid characters.
ERROR 212: Wrong type.   The filename you specified is a directory, or vice versa.
ERROR 213: Disk not valid.   The disk either has not yet been validated by the system, or is defective.
ERROR 214: Write protected.   The disk is write protected.
ERROR 218: Disk not mounted. The disk named is not in one of the disk drives.
ERROR 221: Disk full.   The disk is full.
ERROR 222: Cannot delete.   The requested file is protected from deletion.
ERROR 223: Cannot write.   The requested file is protected from writing.
ERROR 224: Cannot read.   The requested file is protected from reading.
ERROR 225: Not a DOS disk.   The disk is not a valid Amiga DOS disk.
ERROR 226: No disk.   There is no disk in the drive.

RANGES OF ALLOWED VALUES

Object X,Y,Z positions:	Q..8192
sizes:	0..8192
Viewpoint X,Y,Z positions:	0..8192
rotations:	0..359
Numbers in conditions:	-16384..+16383
Variables:	0 255
which can store:	-2147483646..+2147483645

CONDITIONS

OBJECT

AREA

GENERAL

FILE

INTRODUCTION

PC - ADDITIONAL INFORMATION FOR VGA

As EGA except for:-

SAVE DATA and LOAD DATA - VGA only - an extra file will be saved to disc
containing colour palette information - this will
have the extension .PAL

LOAD BORDER - VGA format is 320 x 200 pixels, 256 colours

MAKE (PC) - The runner program is called RUNVGA.EXE

EDIT INSTRUMENT - The instrument colour range is from 0 to 256
for all types

AREA COLOURS - This will display the COLOUR OBJECT option
as all areas use the same palette

COLOUR OBJECT - 256 solid colours are available in VGA mode.
The last 16 colours are used for the editor and
cannot be changed. All other colours can be
altered using the red, green and blue slider bars.

INDEX

CONDITIONS

OBJECT

AREA

GENERAL

FILE

INTRODUCTION

IBM PC, AMIGA

AND

ATARI ST

3D CONSTRUCTION KIT™ ENGLISH
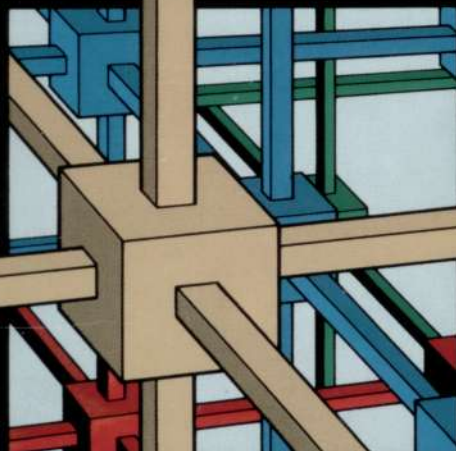
incentive
The Award Winners

D⬭MARK