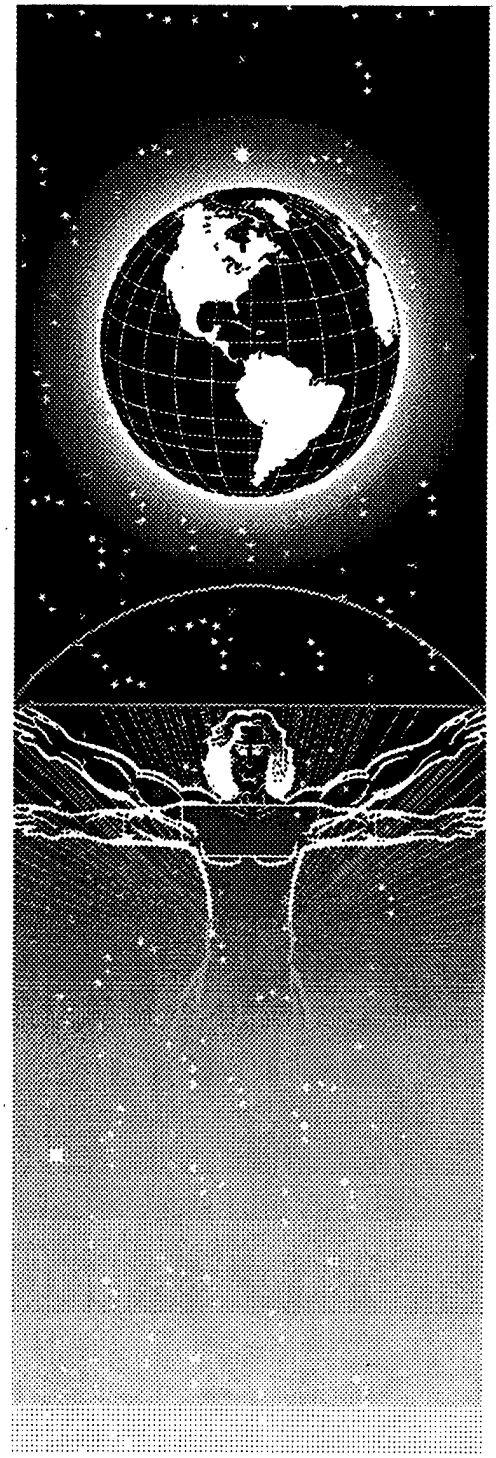
AFRL-HE-WP-TR-2000-0017



UNITED STATES AIR FORCE RESEARCH LABORATORY

DEVELOPMENT MANUAL FOR 3D WORLD VIRTUAL ENVIRONMENT SOFTWARE

Annette L. McCoy

SYTRONICS, INC.
4433 DAYTON-XENIA ROAD
DAYTON OH 45432

Susan K. Schnipke

OHIO STATE UNIVERSITY
DEPARTMENT OF PSYCHOLOGY
COLUMBUS OH 43210

DECEMBER 1999

INTERIM REPORT FOR THE PERIOD APRIL 1998 TO NOVEMBER 1999

Human Effectiveness Directorate
Crew System Interface Division
2255 H Street
Wright-Patterson AFB OH 45433-7022

NOTICES

When US Government drawings, specifications, or other data are used for any purpose other than a definitely related Government procurement operation, the Government thereby incurs no responsibility nor any obligation whatsoever, and the fact that the Government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication or otherwise, as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

Please do not request copies of this report from Air Force Research Laboratory. Additional copies may be purchased from:

National Technical Information Service
5285 Port Royal Road
Springfield, Virginia 22161

Federal Government agencies and their contractors registered with Defense Technical Information Center should direct requests for copies of this report to:

Defense Technical Information Center
8725 John J. Kingman Road, Suite 0944
Ft. Belvoir, Virginia 22060-6218

TECHNICAL REVIEW AND APPROVAL

AFRL-HE-WP-TR-2000-0017

This report has been reviewed by the Office of Public Affairs (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication.

FOR THE COMMANDER

MARIS M. VIKMANIS, DR-IV
Chief, Crew System Interface Division
Human Effectiveness Directorate
Air Force Research Laboratory

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	December 1999	Interim, April 1998 to November 1999

4. TITLE AND SUBTITLE

Development Manual for 3D World Virtual Environment Software

5. FUNDING NUMBERS

C: F41624-94-D-6000
PE: 62202F
PR: 7184
TA: 14
WU: 25

6. AUTHOR(S)

Annette L. McCoy*
Susan K. Schnipke**

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

*Sytronics, Inc.
4433 Dayton-Xenia Rd, Dayton OH 45432

** Ohio State University
Department of Psychology, Columbus OH 43210

8. PERFORMING ORGANIZATION

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Air Force Research Laboratory
Human Effectiveness Directorate
Crew System Interface Division
Air Force Materiel Command
Wright-Patterson AFB OH 45433-7022

10. SPONSORING/MONITORING

AFRL-HE-WP-TR-2000-0017

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This report documents a software package called **3DWorld**. The software provides the environment and scenario development tools necessary to create a virtual environment for human performance research. This report contains step-by-step instructions on how to develop and run virtual environments, as well as an in-depth description of the program structure.

14. SUBJECT TERMS
Virtual Environments Human Performance

15. NUMBER OF PAGES
60

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UNLIMITED

THIS PAGE INTENTIONALLY LEFT BLANK

List of Figures

THIS PAGE INTENTIONALLY LEFT BLANK

1.0 INTRODUCTION

3D World is a virtual environment software package created by scientists in Air Force Research Laboratory (AFRL) at Wright-Patterson Air Force Base, Ohio. The program, which runs under MS-DOS, allows users to design virtual environments, customize scenarios, navigate within the environments, and collect experimental data.

Gerald Dalley, a summer intern with AFRL, developed the original version of 3D World primarily for studying situation awareness issues. A general definition of situation awareness is "the perception of elements in the environment within a volume of time and space, the comprehension of their meaning, and the projection of their status in the near future," (Endsley, 1993). Using 3D World environments, we can study situation awareness by researching how people perceive their surroundings, navigate within those surroundings, and remember locations of objects. To date, 14 situation awareness studies have been conducted using 3D World environments. Results of six of the studies have been published (Colle & Reid, 1998; Colle & Reid, 1999), and the others are a series of studies which are near completion. In addition, 3D World environments were also used to study workload issues at Ohio State University (Nygren, Schnipke, & Reid, 1997) for which the environments were customized to measure time pressure, effort, and stress.

In this paper, we will be explaining how to build an environment, how to view and navigate within the environment, how to customize scripts or scenarios, and how to collect data while running the program. First, read the overview which will provide you with general information about the program and give you a better sense of what the 3D World program has to offer. Then go on to the more detailed sections of the manual for in-depth information about creating environments.

2.0 OVERVIEW

Building an environment in 3D World can be very simple or very complex, depending upon what you want. You may want a simple world which consists of a small building with a couple of rooms in which you are free to roam around, or you may want a multi-level, multi-room environment, equipped with stairs and elevators, and monitored movement. We will briefly overview what is involved in building a small, basic environment so you'll have an idea of what to expect.

To begin, take a look at the following illustrations to get an idea of what an environment may look like on the screen. Figures 1, 2, and 3 are a sequence of captured screen images as the operator is walking down a hallway.

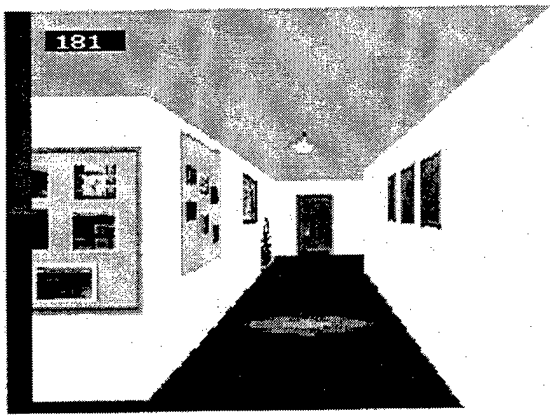


Figure 1

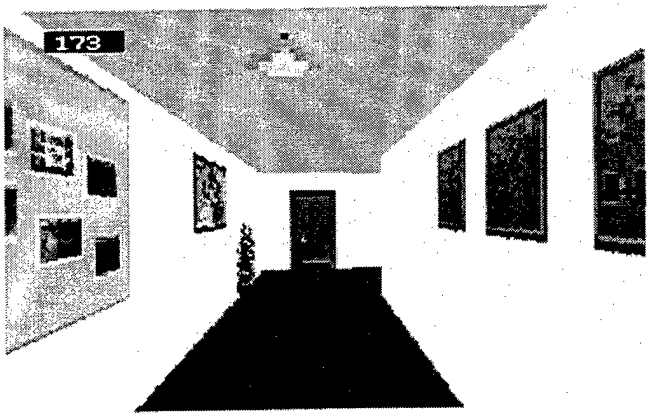


Figure 2

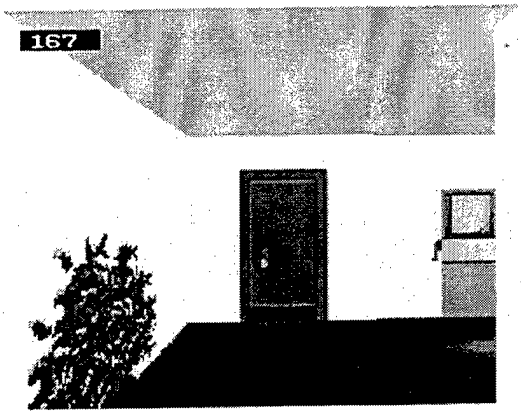


Figure 3

Figures 4 through 10 are various captured screen images of environments used in situation awareness (Colle, Reid, 1997) and workload (Nygren, Schnipke, 1997) studies.

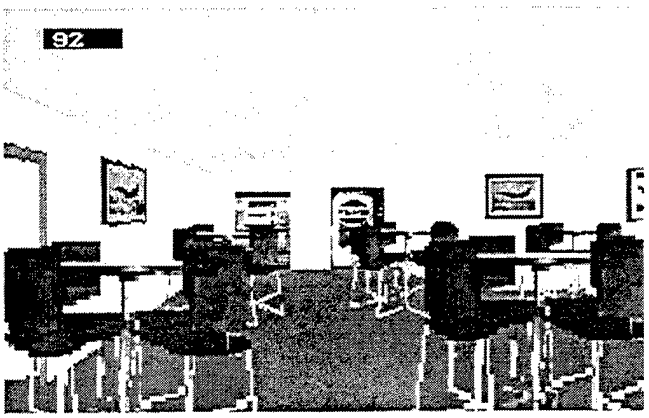


Figure 4

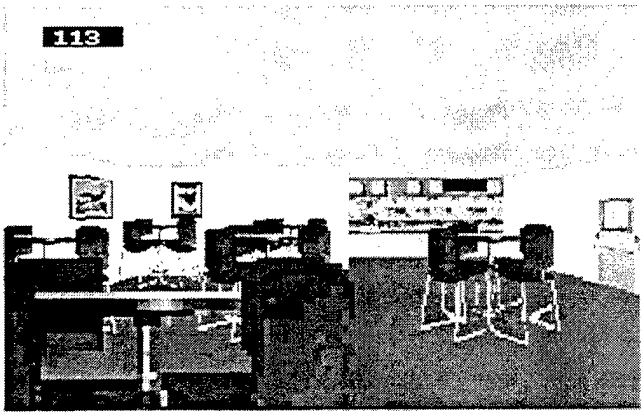


Figure 5

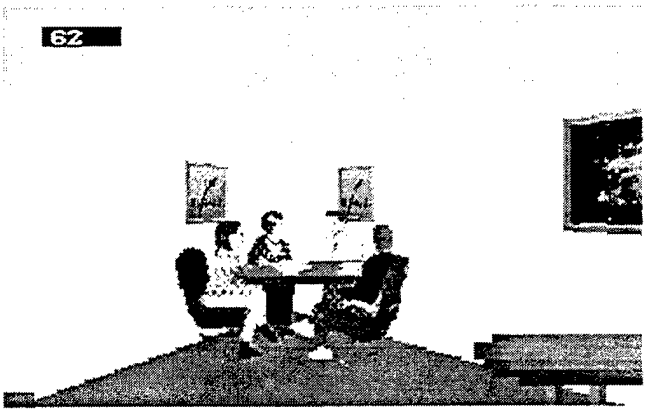


Figure 6

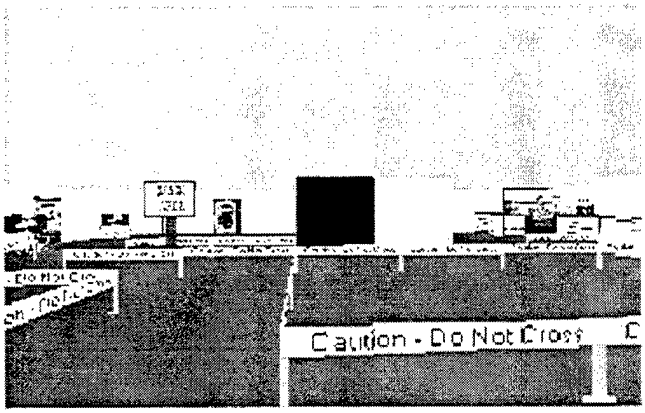


Figure 7

Figure 8

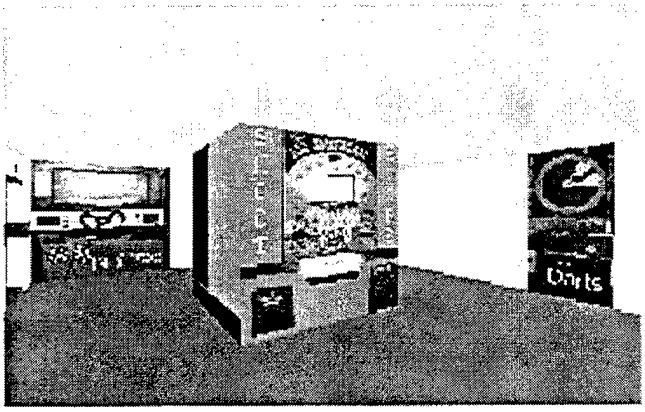


Figure 9

Figures 1-9 are examples of what you would see while 'viewing' the environment. The environment is comprised of several pictures called image files. Image files are typically .pcx files created in a paint program. Each picture (image file) represents a wall or an object in a room. For example, one picture could be a plain wall. If you organized pictures of a plain wall in the form of a square, you'd have a square room. To add a door, you would include a picture of a wall with a door on it, or you could add a wall with a mirror or window, etc. Organize these pictures to form rooms and hallways and you create an environment. All of these image files are gathered into one of two *Icon Description* files: mapdata.def or objectdata.def.. *Mapdata.def* contains the image files which represent *walls* of the enviroment. *Objectdata.def* contains images which represent *objects* within rooms, such as a chair. Both of these files are displayed on a drawing board called the *Map Editor*. The *Map Editor* is the tool used to build or create the environment. See Figure 10.

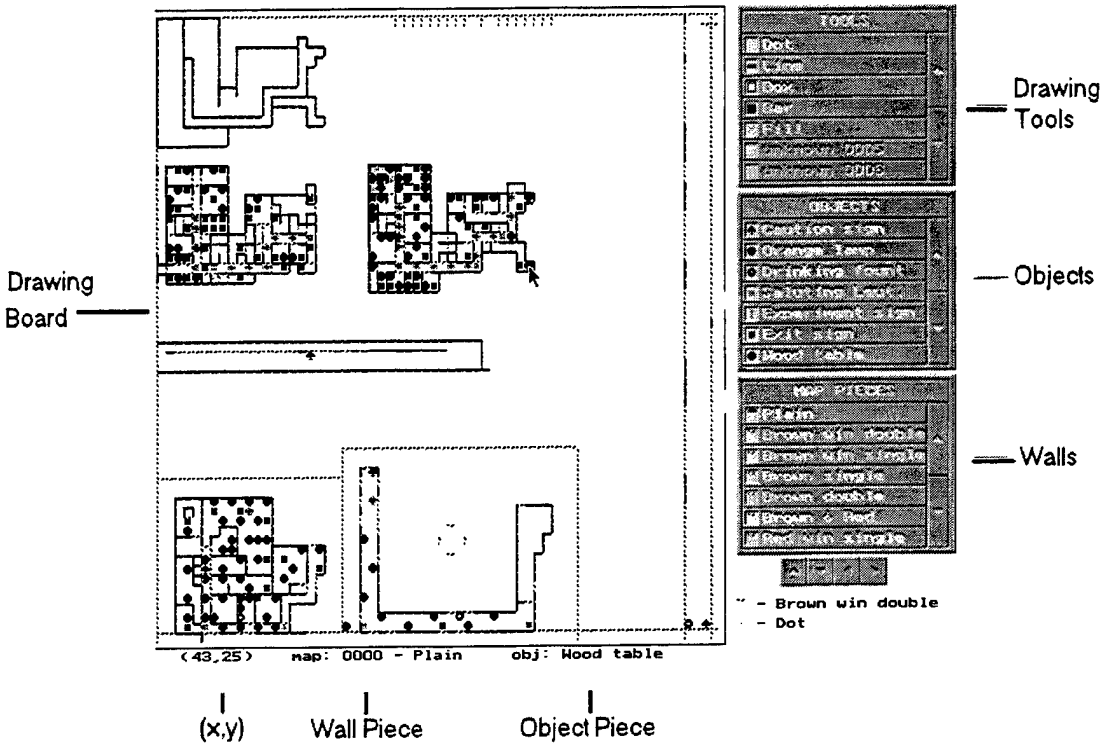


Figure 10. Map Editor drawing board

As you see in Figure10, the object and wall pieces are listed to the right of the actual drawing area. To create an environment, you simply select a map piece by clicking on it with the mouse, then click on the drawing board where you want to place it (more detailed instructions will be provided later). For example, the above drawing area shows five individual environment segments, which can represent different levels of one building, or different environments all-together. It also shows a long corridor which

appears about midway down the drawing board. The line segments on the board represent the walls of the environment and the circles, squares, etc. represent objects within the rooms and hallways. Listed in the object icon menu, there are also startpoint arrows to choose from which you select and place at the position you want to start operator control in the environment. Wherever the arrow is placed is where you will begin viewing the environment when running the 3D program. More information is provided on the Map Editor in Section 4.2.

To briefly summarize, to create and operate in a virtual environment, you 1) collect multiple image files which represent walls and objects you want in your environment (See Section 3.1), 2) list all images in the Icon Description files to be displayed on the Map Editor, 3) build the environment using the Map Editor drawing board, 4) create a scenario, 5) run the 3D program, and 6) navigate in the environment and collect data.

Of course, this is a very simplified overview. More detailed information is provided throughout the manual. For questions regarding this document or the 3D World Program, see the references at the end of the manual.

*Note: For the remainder of this manual, the term "user" refers to the person using the 3D World software to develop the environments; "operator" refers to the person who is navigating in the environment in the run mode.

3.0 UNDERSTANDING THE FILES YOU WILL USE

3D World requires that the following files be present within a directory in order to create a working environment. These files can be categorized into five different groups:

1) **Image Files** (.pcx files)

2) **Icon Description Files**
   a) objdata.def
   b) mapdata.def
   c) tools.def

3) **World Database File** (3d.map)

4) **World Development Files**
   a) editmap.exe
   b) editor.exe
   c) initmap.exe

5) **Loading and Running Files**
   a) 1.way
   b) egavga.bgi
   c) map.3dm
   d) 3d.exe

3.1 Image Files

In order to create an environment, you need image files which portray walls, doors, objects, etc. 3D World uses .pcx image files created with an independent paint program. All .pcx files must be 128 x 128 resolution, 256-color .pcx graphics files. In general, the same color palette should be used for all images. The maximum number of .pcx files you may use in any single environment is 254.

3.2 Icon Description Files

The Icon Description files define the map piece icons and drawing tools which will be used to build the environment on the Map Editor drawing board (see overview). There are three different Icon Description files: 1) mapdata.def, 2) objdata.def, and 3) tools.def. The mapdata.def and objdata.def files specify the name, color, and shape of the wall and object map pieces (respectively) to be used in the Map Editor. The Tools.def file defines the tools which are available to assist in building the environment. The format is similar for all three files.

3.2.1 Mapdata.def

An example of a mapdata.def file is as follows:

0000 eeff plain
0001 7aff window
0002 66cc door
0003 29ff mirror
0004 11ff clock
00FF 00ff empty

*Note: The following line should ALWAYS appear at the end of the mapdata.def file:

00FF 00FF Empty

The first field of numbers represents the order of the item in the file. The second field defines that item's icon image. The icon will appear as two lines on the drawing board map. The first two positions in the second field define the color of the icon; the last two

positions define the bit patterns for those lines (for more information on how bit patterns work, see setlinestyle documentation in either Borland Pascal or Turbo C manuals).

The line colors are defined as follows:

0 – Black	8 - Dark Gray
1 – Blue	9 - Light Blue
2 – Green	A - Light Green
3 – Cyan	B - Light Cyan
4 – Red	C - Light Red
5 – Magenta	D - Light Magenta
6 – Brown	E - Yellow
7 - Light Gray	F - White

The third field is the descriptive name for the wall icon. The descriptive name does not need to be the same as the .pcx file it represents. The icon and its descriptive name will appear in the Map Editor and is for identification purposes only.

Using the example mapdata.def file above, the mirror is 0003 (the fourth line) on the list. (*Note - the actual name of the mirror .pcx file **must also be listed fourth** in the 3d.map file discussed in Section 3.3). The icon that will appear on the map to represent this mirrored wall will be a pair of lines, one green (#2) and one light blue (#9).

The first and second fields are all hexadecimal numbers (all digits 0..f) and **MUST** be four characters long. Each field **MUST** be separated by one and only one space. Additionally, there **MUST** be **NO** blank lines or lines that do not follow the above conventions.

3.2.2 Objdata.def

There are two differences between mapdata.def and objdata.def: 1) specific starting point lines must be included in objdata.def, and 2) the images are presented as objects rather than walls in the environment. An example of an objdata.def file follows (the required starting point lines are in bold):

0000 1056 Box
0001 60f0 Tree
0002 b016 Chair
0003 5120 Person
00F7 20F0 Starting point
00F8 20F1 Starting point
00F9 20F2 Starting point
00FA 20F3 Starting point
00FB 20F4 Starting point
00FC 20F5 Starting point
00FD 20F6 Starting point

00FE 20F7 Starting point
00FF 0010 Empty
The object icons will appear on the map as symbols such as a circle or square. Here is the key for icons in Objdata.def:

1st position
0-F (see above): foreground color

2nd position
0-F (see above): background color

3rd position
0 - print character in 4th position
1 - solid
2 - half-tone
3 - solid w/decoration
4 - half-tone w/decoration
5 - circle
6 - horizontal door
7 - vertical door
8 - top half foreground, bottom back
9 - dot
A - tall upper left
B - short upper left
C - centered
D - x **(associated with 4th position)**
  0 - no background
  1 - show background bar
  2-f - reserved
E - outline
F - arrow (direction determined by 4th position)

4th Position
When D is in 3rd position:
  0 - no background
  1 - show background bar
  2-f - reserved
When F is in 3rd position:
  0 - north
  1 - northeast
  2 - east
  3 - southeast
  4 - south
  5 - southwest
  6 - west

7 – northwest
*Note: If D or F is not in 3rd position, the 4th position is ignored, but a number must be inserted.

3.2.3 Tools.def

The Tools.def file lists the 'drawing tools' and defines the shape and color of their corresponding icons which appear in the Map Editor. The "toolbar" is similar to those used in independent paint programs. **YOU SHOULD NOT EDIT THE TOOLS.DEF FILE.** There will be more information regarding the tools in Section 4.2.2.1.

3.3 World Database File (3d.map)

3d.map is considered to be the world's database. It calls up and defines all image files as walls or objects, it calls up the environment map, and it defines walking parameters, turnrate, and other miscellaneous parameters. It defines any overhead images (to be discussed later in 3.3.5) and Help screens to be used in the environment. There are six major sections to the 3d.map file: 1) map - defines the map for the environment, 2) parameters - defines the walking parameters, 3) pic - defines the wall images, 4) obj - defines the object images, 5) overhead - defines the overhead images, and 6) help - defines the help screens. Here is an example of an entire 3d.map file:

```
[map]
map.3dm

[parameters]
stepHeight .50
eyeLevel 5.0
speed 31
stepDist 3
turnRate 100

[pic]
wall1.pcx
basewall.pcx
poster.pcx
sign.pcx
window.pcx window
door.pcx door
windoor.pcx door window
opendoor.pcx opendoor window

[obj]
table.pcx
```

[overhead]
map.pcx
Cafeteria
Conference

[help]
help.pcx

3.3.1 [map] section

The [map] section calls up the map of the environment which is a file named map.3dm. This is the map that is created and edited in the Map Editor and it is **always saved** as map.3dm. Only the map.3dm file can be loaded in this section, therefore the following lines **must** appear in the 3d.map file:

[map]
map.3dm

If you would like to have multiple versions of maps, simply rename other versions (i.e., mall.3dm or kitchen.3dm) and keep them in the 3D directory. Just remember that the map you want to view in the map editor must be called map.3dm.

3.3.2 [parameters] section

This section sets up the walking parameters for 3D World. The **stepheight** controls the "bobbing" sensation when walking. If this value is increased, the "bobbing" sensation is increased. The **eyelevel** is the height of the field of view. In **USEKEYS** mode, the **speed** dictates how many "steps" it will take to cross a coordinate block of the map, which represents 8 square feet. The **stepdist** should be an estimate of how long the step is, however, changing the value does not seem to change the actual step distance. The **turnrate** is how many key presses it takes to move a certain number of degrees in a circle. Step height, distance, and eye level are measured in units of feet, walking speed in miles per hour, and turning rate in degrees per second. These units will vary according to the computer you are using and how the waypoint file is set up to navigate; i.e., in USEKEYS or USEZOOM mode (see 3.5.1.1). Using a 486 DX-4, 100mhz computer with an Intel processor, and the USEKEYS mode set in the waypoint command file, the following parameters "seem natural":

[parameters]
stepheight .50
eyelevel 5.0
speed 31
stepdist 3
turnrate 100

You may change any of these values to suit your needs. Remember, the values are highly dependent upon the computer you are using so there are no set values defined. If no values are entered or if any values exceed the 8-foot boundaries of a coordinate block, 3D World will give a short error message telling you what the problem is. The step distance must be entered after the speed, and if any values are changed, any .mov files (see 3.5.1.5) that are being used must be rerecorded. Here are some examples of how changing the above parameters effect the keystrokes in the **USEKEYS** mode:

Parameter	Approximate Number of Keystrokes
turnRate 100	≈ 71 keystrokes for a complete circle. ≈ 5° per keystroke.
turnRate 200	≈ 40 keystrokes for a complete circle. ≈ 9° per keystroke.
speed 31	3 blocks ≈ 6 keystrokes. 4 blocks ≈ 11 keystrokes. 7 blocks ≈ 16 keystrokes.
speed 12	3 blocks ≈ 13 keystrokes. 4 blocks ≈ 17 keystrokes. 7 blocks ≈ 45 keystrokes.

Figure 11. Examples of changing parameters in the USEKEYS **mode**

Changing parameters in the **USEZOOM** mode will not show significant differences. Although all parameters must be entered, the important parameters are the stepHeight and the eyeLevel in this mode.

3.3.3 [pic] section

The [pic] section defines the image (.pcx) files to be used for the environment's walls. **These files must be in the same order as the items in the mapdata.def file (section 3.2.1)** in order for the world editor to accurately represent the environment. The descriptive name in the mapdata.def file need not be identical to the actual .pcx filename, but the order should be exact. Here is an example of the [pic] section containing three image files.

```
[pic]
bluewall.pcx
bathroom.pcx
officwin.pcx
```

There are three features that can be added to a basic wall. The first feature is to make part of the wall piece **transparent**. The second feature is to have a **door** that can be opened and walked through. The last feature is to have an **open door** which can be walked through.

3.3.3.1 Partial Transparency

This feature allows part of the wall to be transparent. This is useful for making windows and open doorways. The part of the image file that should be transparent must be black. The .pcx filename must be followed by the word window in the 3d.map file to take on transparent qualities. For example, the image file officwin.pcx listed above refers to an office window. To make the black areas appear as a window, you must type the word window following the filename.

Example:
officwin.pcx window

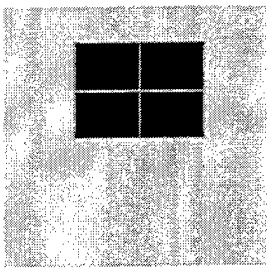


Figure 12. Officwin.pcx as it appears in a paint program

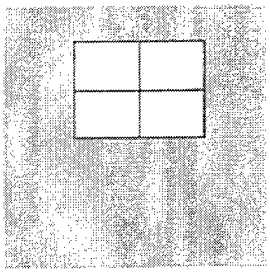


Figure 13. Officwin.pcx as seen as a wall in the Environment.

Wall.pcx would appear as a wall with a transparent window in the environment. Anything on the other side of the wall would be visible through the window.

3.3.3.2 Door

Normally you may not walk through a wall, however, the "door" feature of a wall provides the ability to pass through the wall piece. The .pcx filename must be followed by the word door in the 3d.map file. To pass through the door, the spacebar must be pressed

when the operator is next to the door. Generally, this is used for closed doors so that when the operator walks up to the door, they must stop and press the spacebar to get in, which would simulate stopping to open the door.

Example:
bathroom.pcx door

3.3.3.3 Open Door

The "open door" feature allows for passage through a wall piece without pressing the spacebar. Often, this feature is used with the "window" feature so the perception is a door standing open that can be looked through and walked through.

Example:
bathroom.pcx opendoor window

Examples as seen in a Paint Program:

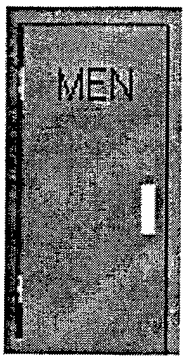


Figure 14 Bathroom.pcx door window

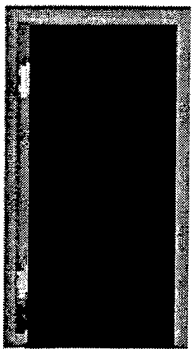


Figure 15 Bathroom.pcx opendoor

In the environment, if standing in front of Figure 13, you could press the spacebar and be placed on the other side of the door as if you had walked through the doorway. In the environment, the black area in Figure 14 would appear transparent and you could walk through the doorway.

3.3.4 [obj] section

The [obj] section defines the .pcx files to be used as objects in the environment. Objects are placed within rooms, and they do not hinder movement when navigating through an environment. One coordinate block on the map is considered to be an 8 x 8 foot room, and only one object can be placed 'within' a coordinate block. An object is actually flat, considering that it is an image file like the walls. Therefore, it only has one view, so no matter what angle it is viewed from, it will always appear the same. An example is that if

the image file of a person facing you is placed in a room as an object, the person will always appear to be facing you regardless of the viewing angle.

In summary: 1) objects are placed within a coordinate block on the map whereas walls define the perimeter of coordinate blocks, 2) the space surrounding the object in the image file should be painted black in order to appear transparent in the environment, 3) you can walk through objects, but not walls (unless designated as open doors). Note: If the space surrounding the object is not painted black, the object will appear to be a wall that you can walk through.

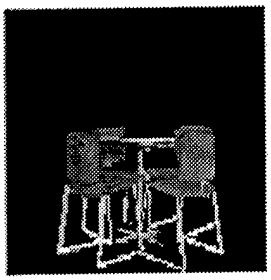


Figure 16. Table.pcx as it appears in paint program.

Figure 17. Table.pcx as seen as an object in the environment.

3.3.5 [overhead] section

This section gives information for displaying an 'overhead' picture in the environment. An overhead picture is an image file which can have movable text pieces.

The text pieces are originally displayed as white text listed down the left side of the image file. Each piece of text can be moved around within the image file using the drag and drop feature of a mouse. The first entry in the [overhead] section is the file name of the .pcx file to be displayed. All successive lines define the text pieces to be listed. In the following example, a map (map.pcx) will be loaded as an overhead image file, and Office, Cafeteria, Files, and Conference will be the movable text pieces.

```
[overhead]
map.pcx
Office
Cafeteria
Files
Conference
```

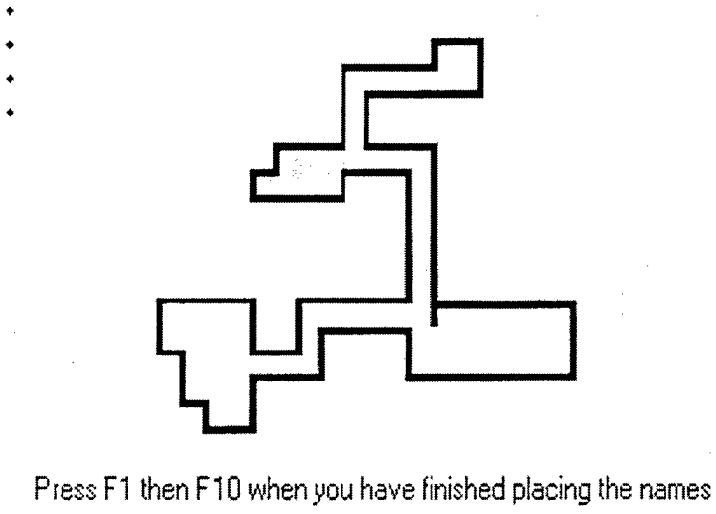


Figure 18. Overhead Image of a map

In this case, the user would select the text pieces with the mouse and move them to the desired location on the map. Another example of an overhead image would be a schedule like the one shown below.

[overhead]
schedule.pcx
Annette
Susan

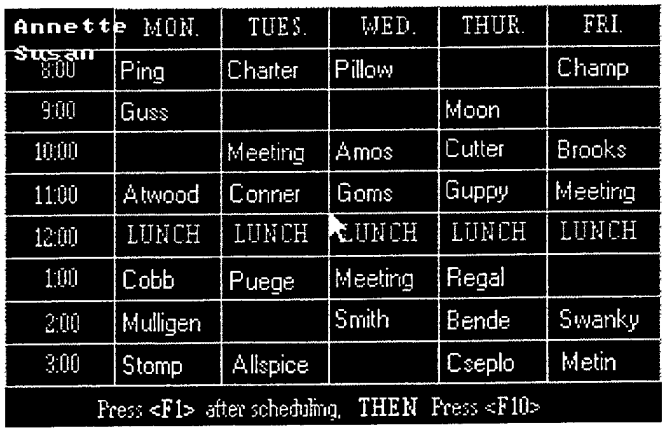


Figure 19. Schedule.pcx as an Overhead Image.

In this example, the two names in the top left corner, Annette and Susan are the moveable text pieces, and they can be selected and moved to the appropriate cell.

There are two ways to call up an overhead picture: 1) calling it up in the waypoint command file using the SHOWMAP command (Section 3.5.1.6), or 2) showing the picture on demand with the key combination of Alt-m.

To exit the overhead picture, <F10> must be pressed. If the picture is shown more than once, the pieces will appear wherever they were left in the previous display; however, the position is **not** saved in the data output file. Currently, the only known method to save the final positions of the names, is to run a screen-capturing program simultaneously with 3D World.

3.3.6 [help] section

This section is a one-line entry that displays a customized help screen when the key combination of Alt-h is pressed. The help screen is a .pcx image file no larger than 360 by 200 pixels. To exit the "help" screen, press any key.

[help]
help.pcx

3.4 World Development Executable Files

The World Development files are the World Editor and the Map Editor. These are the executable files that are used to actually build the environment. The World Editor is the primary interface between the user and environment, and the Map Editor is the drawing tool used to construct the environment. You will be instructed how to use the editors in Section 4.0.

3.4.1 Editor.exe

Executes the World Editor Menu System (Section 4.1).

3.4.2 Editmap.exe

Executes the Map Editor which is the drawing board and tools for building the environment (Section 4.2).

3.4.3 Initmap.exe

Executes the Map Editor, but creates a blank drawing board (Section 4.2).

3.5 Loading and Running Files

The following are files that are needed to run the 3D World program. The waypoint command file is the primary running file in that it is within the waypoint file (1.way) that you can define a scenario or story for the environment, designate sound files, provide instructions or define routes within the environment, etc.

3.5.1 The Waypoint Command File (1.way)

The waypoint command file is the central running file for 3D World. It is responsible for loading the map, initializing sound, providing interactive information, controlling all action in the environment by designating waypoints (positional goals), and defining data collecting procedures.

A positional goal is a coordinate block (waypoint) within the map of the environment that the operator 'triggers' once it is stepped on. For example, an operator may be asked to go to a particular room in an environment and find a specific object. When the operator walks up to the object, that waypoint may trigger, and he may get a text message on the screen and/or a voice message confirming that he has found the object. A waypoint is simply a place where instructions are provided, voice files are activated, overhead images are displayed, or some other action may occur.

The waypoint file is created using any ASCII text editor. The 1.way file is created and/or edited in the DOS editor by typing "edit 1.way" at the DOS prompt. It can also be created and/or edited in Windows using Notepad or any other editor. Regardless of the editor used, the file should always be saved as text with .way as the extension. You may have multiple waypoint files in a directory, for example. 1.way, 2.way, 3.way, etc. 3D World

only recognizes 1.way, however, unless you initiate the executable with the w = somenumber command when you start the program (See 3.2 for more information).

A waypoint file is partitioned into sections called functions. A function begins with a header (usually waypoint coordinates) and lists a set of commands to be carried out when that function is called. The first function in the waypoint file, however, is unique in that it provides instructions to the run module for setting up the environment.

3.5.1.1 The Start Function

The headers for all functions are enclosed in square brackets. The first function always has a "start" header rather than waypoint coordinates. Following the start header, commands are given to set up the environment. Typically, the commands in the [start] function include: 1) display a picture while the map is being loaded, 2) load the world database .map file, 3) activate the sound, 4) activate the input device to be used and 5) activate the first waypoint. Here is an example of the [start] function.

```
[start]
showpalpic logo.pcx
load 3d.map
usesoundeffects
usedigitizedvoice
backgroundsoundoff
usekeys
wait
activate (17,6,1)
play
```

The following is a list with explanations of the commands that can be used in the [start] function.

- **SHOWPALPIC**

Displays a picture centered on the screen while the .map file is being loaded. Typically, an introduction screen is displayed. The picture's palette is also being loaded at this time. In general, the same palette should be used for all images. If multiple palettes must be used, the palette can be reset by using the SHOWPALPIC command.

- **LOAD**

Loads the 3d.map file. This command is essential to run 3D World. The 3d.map file is the world database and calls up the images for the environment. Only one LOAD should be performed per waypoint file and the LOAD command should always precede a PLAY command. The file does not have to be called 3d.map, but does have to have a .map extension.

• **USESOUNDEFFECTS**
Enables the playing of sound effects to signal running into walls and walking through doors. If this option is turned on, ouch.voc and phaser8.voc must be present in the current directory.

• **USEDIGITIZEDVOICE**
Enables the playing of digitized sound files with the **SPEAK/SPEAKONLY** commands. This allows you to hear voice files of people talking.

• **BACKGROUNDSOUNDON/ BACKGROUNDSOUNDOFF**
BACKGROUNDSOUNDON turns on background sound. It causes **test.mid** to be played continuously, until the program exits, or BACKGROUNDSOUNDOFF is executed which simply turns off the background sound started by BACKGROUNDSOUNDON. Successive calls to BACKGROUNDSOUNDON will simply restart the background music.

Note*: To hear sound, the /sound command must be present when starting 3D World (see Starting the Program, Section 3.1). These features have only been successfully tested using true SoundBlaster cards.

• **USEKEYS / USEZOOM**
To navigate in an environment, the keyboard or the mouse can be used. The default navigation device is the keyboard. These commands specify the navigation device. None of the keyboard commands have associated parameters, therefore each should be entered on a line by itself.

USEKEYS changes the 3D viewer's input to the standard keyboard input. For slow machines like a 386DX/20, this mode is recommended since keystrokes will not be "missed" if the computer samples at the wrong time. The keyboard buffer is reduced down to one stroke, and additional keystrokes are ignored until the first is processed. This navigation device is also recommended when the feeling of "stepping" is desired.

USEZOOM is a faster implementation of USEKEYS. The "zoom" mode gives more of an arcade game feel, but requires more processor speed. If the processor is too slow and can't provide a reasonably high update rate, users may notice quick keystrokes being missed by the computer. This mode allows multiple keystrokes to be pressed for turns while walking, etc. The motion also tends to seem more fluid, and thus the sensation of "stepping" is reduced.

• **USEMOUSE / NOMOUSE**

USEMOUSE enables the mouse as a navigation device. This command must be used in conjunction with either USEKEYS or USEZOOM. The mouse is generally not used as a navigation device because it is difficult to control.

NOMOUSE disables the mouse as a navigation device.

• **WAIT**

The WAIT command used at the end of the first function will keep the image file that was loaded using the SHOWPALPIC command on the screen until the operator presses a key. A message reading "Press a key to continue" will blink on the bottom of the screen. Once a key is pressed, the environment will appear on the screen at the starting point. If this command is not used, the image file will only be on the screen long enough for the 3d.map file to be loaded, then will disappear and the environment will appear at the starting point. This usually happens vey quickly, so it is recommended to use the WAIT command in the start function.

• **ACTIVATE / DEACTIVATE**

When in play mode, the 3D World viewer checks to see if an active waypoint block has been intersected (stepped on) each time a movement takes place. When an active waypoint has been triggered, execution goes to the commands associated with that coordinate block.

The first waypoint should be activated in the [start] function. ACTIVATE activates a waypoint. The coordinates in (x,y,l) format of the waypoint should follow the word activate. The x and y are standard coordinates and the 'l' identifies the level of the waypoint. You may have multiple levels of a waypoint if you want it to do different things at different times.

When the ACTIVATE command is called, a search is conducted within the entire waypoint file to find the header containing that waypoint. Using the ACTIVATE example shown in the [start] function above, a line beginning with "[(17,6,1)]" would be searched for. If found, the commands associated with that waypoint would be executed.

DEACTIVATE simply deactivates a waypoint so it will not trigger if stepped on again. This allows the operator to visit a location more then once, but only have an action occur at the time that the waypoint is active. This command is essential when using multiple level waypoints. If a waypoint is deactivated within it's own function, DEACTIVATE MUST BE the last command before PLAY.

• **PLAY / END**

The final command in a function is either PLAY or END. Either of these commands signals the end of the function. PLAY is the command used at the end of all functions to return control to the operator. While in play mode, navigation of the environment is possible until a waypoint is stepped on. At that time, control is given to the function associated with that waypoint. END is used in the last function to end the 3D viewing program.

3.5.1.2 Waypoint Functions

All functions following the "start" function act as positional goals called waypoints. Each waypoint function contains a header which indicates the coordinates that activate the waypoint. After the waypoint is activated, commands following the header are executed. The waypoint is terminated when a **PLAY** or **END** command is encountered.

The header of every waypoint **must** be enclosed in brackets. The coordinates for a waypoint are in the (x,y,l) format. The x and y are standard coordinates and the 'l' is for the level of the waypoint. Each waypoint can have different levels so that each time the waypoint is crossed the program will display different information. The levels are numbered consecutively beginning with one. An example header would appear like this:

[(17,6,1)] x = 17, y = 6, Level = 1

There are a variety of commands which can be executed in a waypoint function. Here is an example of a function in a waypoint file.

```
[(17,6,1)]
record effort.mov
speak intro1.voc
echo "Please raise your hand and
echo "wait for the experimenter
echo "before beginning.
showmap schedule.pcx
showpic blank.pcx
speakonly 16min.voc
nop
speak intro2.voc
echo "You've just arrived at the
echo "Browning, Browning & Smith
echo "law firm. Enter the office to
echo "find out what you will be
echo "doing today.
clock_on 500
activate (15,15,1)
```

deactivate (17,6,1)
play

There are many different commands that can be used in the waypoint file. They have been categorized into four groups: 1) basic functions, 2) interactive commands, 3) data collection commands, and 4) overhead image commands. The various commands are defined below. The command interpreter is not case sensitive, however it is advised that all commands be lower case. Each command must start on a new line to be interpreted correctly, and each command must end with a hard return. Extra spaces are ignored.

3.5.1.3 Basic Commands

• **ACTIVATE** activates a waypoint. The 3d-world viewer checks to see if a waypoint coordinate block has been activated each time a movement takes place. When an activated waypoint is intersected, a search is conducted throughout the entire waypoint file to find the function header containing that waypoint. When in play mode, the execution goes to the first line following the header.

Example:
activate (17,6,1)
activates the function [(17,6,1)] shown above

• **DEACTIVATE** deactivates a waypoint. It is important to deactivate a waypoint once it's been activated if you do not want it to trigger again. By deactivating a waypoint, that point is removed from the list of waypoints to be checked against as described in **ACTIVATE**. If a waypoint is reached, but not deactivated before a **PLAY** command is executed, that waypoint will continue to trigger.

Example:
deactivate (17,6,1)

• **END** ends the 3D viewing program. The output file is closed, cleanup is performed, and control is returned to the DOS prompt.

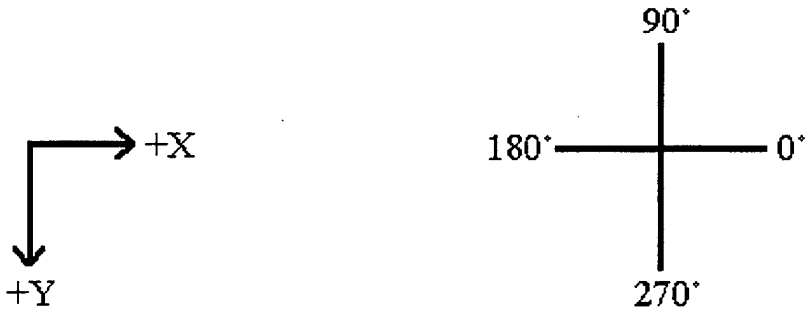
• **NOP** is short for No Operation. This is good for separating dialogue boxes when you have long segments of text.

• **PAUSE** stops the program for a specified amount of time. This command requires a parameter representing the number of seconds for the program to pause. This command can be used in conjunction with the **SHOWPIC** command to create a sense of animation.

Example:
pause 1

• **PLAY** returns control to the viewing program. While in play mode, navigation of the environment is possible until a waypoint is hit. At that time, control is given to the waypoint function associated with that point. A **PLAY** command will almost always be at the end of a waypoint function.

• **POSITION** teleports the operator to a specified location. The values given are the X and Y coordinates and the facing angle. The point (0,0) corresponds to the upper left-hand corner of the world. The X position moves positively to the right and the Y position moves positively downward (the standard quadrant system uses up as positive Y). The angle is measured in degrees where +X is 0°, -Y is 90°, -X is 180°, and +Y is 270°. Five squares to the right, ten squares down from the upper left-hand corner facing up would be (5,10,90).



Example:
position (5,10,90)

Figure 20. (X,Y) Coordinate System and Angles.

This is a useful command when it is necessary to take the subject out of one environment and put them in another, or position them on the other side of a door.

• **POSITIONWAIT** is identical in syntax to the **POSITION** command. The difference is that 3D World waits for a key to be pressed before continuing.

Example:
positionwait (5,10,90)

• **REM** is a remark statement or comment. Any line beginning with this is for internal documentation and is ignored by the run module. Additionally, blank lines and invalid commands are treated as comments.

• **WAIT** pauses until a key is pressed. It also writes a blinking message of "Press a key to continue" centered on the bottom of the screen.

• **WAITFORENTER** pauses until the <Enter> key is pressed. It also writes a blinking message of "Press the <Enter> key to continue" centered on the bottom of the screen.

• **YO** pauses until a key is pressed. The difference between this and the **WAIT** and **WAIFORENTER** commands is that you can customize a message to appear on the screen. The message can be up to 40 characters in length. The syntax is the same as the **ECHO** command (described below).

Example:
yo "<user defined message>

• **YOWAITFORENTER** pauses until the <ENTER> key is pressed. The difference between this and the **WAITFORENTER** command is that a typed message is included that is to be displayed on the screen. The message can be up to 40 characters in length. The syntax is the same as the **ECHO** command (described below).

Example:
yowaitforenter "<user defined message>

3.5.1.4 Interactive Commands

• **ECHO** types a set of text into a text (dialogue) window.

Example:
echo "<message>
echo "<message>

All text in the line following the ECHO command will be displayed in a window. If there is more than one line of text, simply continue to the next line beginning with the ECHO command again. Only the text will appear in the window, not the word 'echo.'

If more than one window is desired, the **NOP** command can be used between echo commands. This is useful for dividing long segments of text up into different boxes so operators aren't required to do as much scrolling. (See Using the Dialogue Windows, Section 3.3 , for further information).

• **SPEAK** plays a digitized voice file on a SoundBlaster card. The command must be followed by an **ECHO** command to hear the sound.

Example:
speak tada.voc

echo "hello

This feature has only been successfully tested on true SoundBlaster cards. SoundBlaster compatible cards that have been tested did not work.

The sound will only be played if the /sound command line option is used when running 3D World. Additionally, **USEDIGITIZEDVOICE** must be called before these sounds will play.

• **SPEAKONLY** plays a digitized voice file on a SoundBlaster card. This feature differs from **SPEAK** in that an **ECHO** command does not have to follow in order to hear the sound.

Example:
speakonly tada.voc

This feature has only been successfully tested on true SoundBlaster cards. Soundblaster compatible cards that have been tested did not work.

The sound will only be played if the /sound command line option is used when running 3D World. Additionally, **USEDIGITIZEDVOICE** must be called before these sounds will play.

3.5.1.5 Data Collection Commands

• **CLOCK_ON** starts an on screen countdown counter. The start time is specified in the command, and the clock counts down in seconds.

Example:
clock_on 500

• **CLOCK_OFF** stops the on screen countdown counter and erases it from the screen.

Example:
clock_off

• **INPUT** displays an input box on the screen that accepts input from the screen and saves it to the data file. The specified prompt to be displayed cannot exceed 40 characters of text.

Example:
input "<specified prompt>

• **PLAYBACK** plays back a series of keystrokes which were previously recorded as a .mov file.

Example:
playback move.mov

• **NOPLAYBACK** ends the playing back of a series of recorded movements before the set of movements have been completed.

• **POINTTO** Each occurrence of this command will read one line of the 'input file', so it is important that there are the same number of entries in the 'input file' as there are 'pointto' commands. The input file consists of the following:

-Position (x,y,angle)
-Question that the subject sees (256 characters max.)
-Correct heading
-Correct distance

A yellow cross hair will appear in the middle of the screen when the subject is asked to point to an object.

• **RECORD** records operator moves to a specified file having the extension .mov.

Example:
record move.mov

In order to end the move recordings, a waypoint must be positioned at the desired endpoint and contain **NORECORD**. The **END** command will also stop the recording process. The recorded commands include all keys active during a "PLAY" session - currently movement, overhead help, opening doors, and quitting the program without the use of an **END** command. The record command does not record <Enter>. The input for the moves is the same as in the **USEKEYS** mode. The RECORD command is useful for recording operator movements for research purposes. Playing back recorded movements can also be useful, for example, to take tours of an environment without operator interaction.

• **NORECORD** stops recording moves and closes the movement file from the last **RECORD** command.

• **RECORD_TIME** records the current time on the down counter to the data file. It will also include a specified comment if present.

Example:
record_time "<comment>

• **SPIN** allows for movement to spin in a circle. **SPIN** must be coupled with the **CLOCK_ON** command because forward and backward movement is not allowed until the clock is at zero. This command gives the sensation of looking around without moving.

Example:
clock_on 10
spin

• **SWAT** accepts input from the screen. This command is a more specific form of input. SWAT stands for Subjective Workload Assessment Technique and is a research tool used to study operator workload (ref%%%). It displays a prompt on the screen asking the operator to enter Time, Effort and Stress SWAT ratings then saves them to the data file.

Example:
swat

• **TIME** records the elapsed time from the 3D World internal timer.

• **TIMEROFF** disables the 3D World internal timer. This allows for the use of 3rd party software that may interfere with the timer in 3D World.

• **TIMERON** enables the 3D World internal timer.

• **YOCOMPASS** is used in conjunction with the command line argument '/comm'. It is identical to the **YO** command in syntax, but after displaying the message on the screen it sends a 'G' to COMM port 2 and then waits for a 'G' from the external computer before continuing. **WARNING**: ***This command has the ability to lock up the 3D World program if used in conjunction with the '/comm' argument and there is no external computer attached.***

3.5.1.6 Overhead Image Commands

• **SHOWMAP** displays a picture of the overhead picture defined in the 3d.map file. <F10> must be pressed to exit the **SHOWMAP** mode.

• **SHOWPIC** displays a .pcx file centered in the screen. For a better guarantee of positive results, make the picture be a multiple of four in width (i.e. 64 or 68 pixels wide, not 65 pixels). The picture should not be any larger than 320 pixels

wide by 200 high. Generally, this should be followed by a **WAIT** or **WAITFORENTER** command to allow the operator time to view the picture.

Example:
showpic map.pcx

• **SHOWPALPIC** is the same as **SHOWPIC**, but also loads the picture's palette. The previous palette is faded out, the picture is loaded, and the new palette is faded back in. This is useful if a palette file does not exist for the current images being used.

Example: showpalpic map.pcx

3.5.2 Egavga.bgi

The graphics driver.

3.5.3 Map.3dm

The actual map of the environment created in the Map Editor

4.0 CREATING AN ENVIRONMENT

In this section, we will be discussing how to actually build the environment using the World Editor and the Map Editor. The World Editor is not required to create an environment. Its purpose is to simplify the development process *especially* when creating the Icon files. If you choose not to use the editor, you must manually edit the Icon files as described in Section 3.2.

4.1 Understanding and Using the World Editor

The World Editor is a development tool which simplifies the creation of an environment by allowing the operator to select options from a menu. You are able to access the Map editor, the DOS editor, a Paint Program, and DOS Shell from within the editor. Perhaps the most important feature of the editor is the ease in which it allows you to create the icon description files (objectdata.def and mapdata.def). The World Editor is not required to create an environment. You can create an environment without ever using the World Editor, however, it can be very helpful in certain ways as we will explain. To access the World Editor, you must have the editor.exe file in your 3d directory, then type 'editor' and it will appear on the screen. You should be able to create a complete working

environment by using the selections from the World Editor menu. See Figure 20 for an example of how the World Editor appears on the screen:

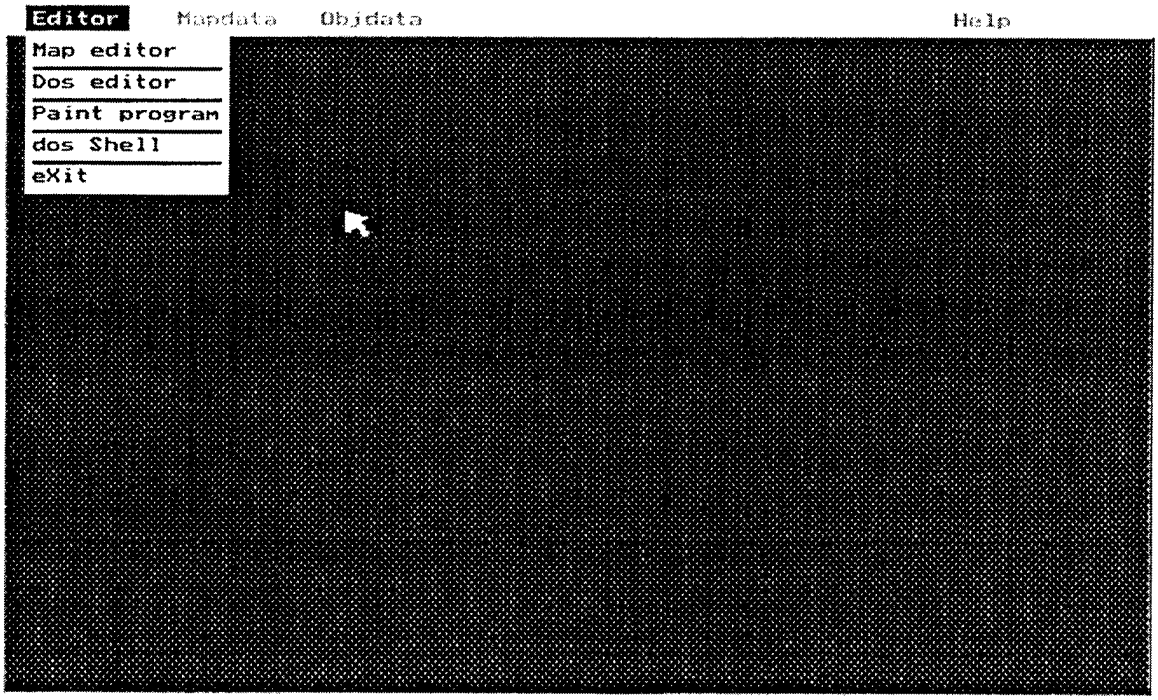


Figure 21. Example of World Editor screen

4.1.1 Editor Menu Selections

Map Editor: Selecting this will display the Map Editor for editing the drawing board.

DOS Editor: This will put you in edit mode in DOS.

Paint Program: This will bring up the Neopaint Paint program for creating/editing .pcx files.

DOS Shell: This puts you in MS DOS mode. You must exit to return to the editor.

4.1.2 Mapdata/Objdata Menu Selections

Selecting these will allow you to create/edit the Mapdata.def and Objectdata.def files and create icons for use in the Map editor. The icons represent the image (.pcx) files that comprise the environment. You will build the environment using these icons on the Map Editor drawing board. Using the World Editor is much simpler than manually entering the field values in the file as described in 3.2. To edit the Mapdata.def file (for example), you would click on mapdata.def, then choose Edit from the menu bar. The following illustration represents what you will see on the screen.

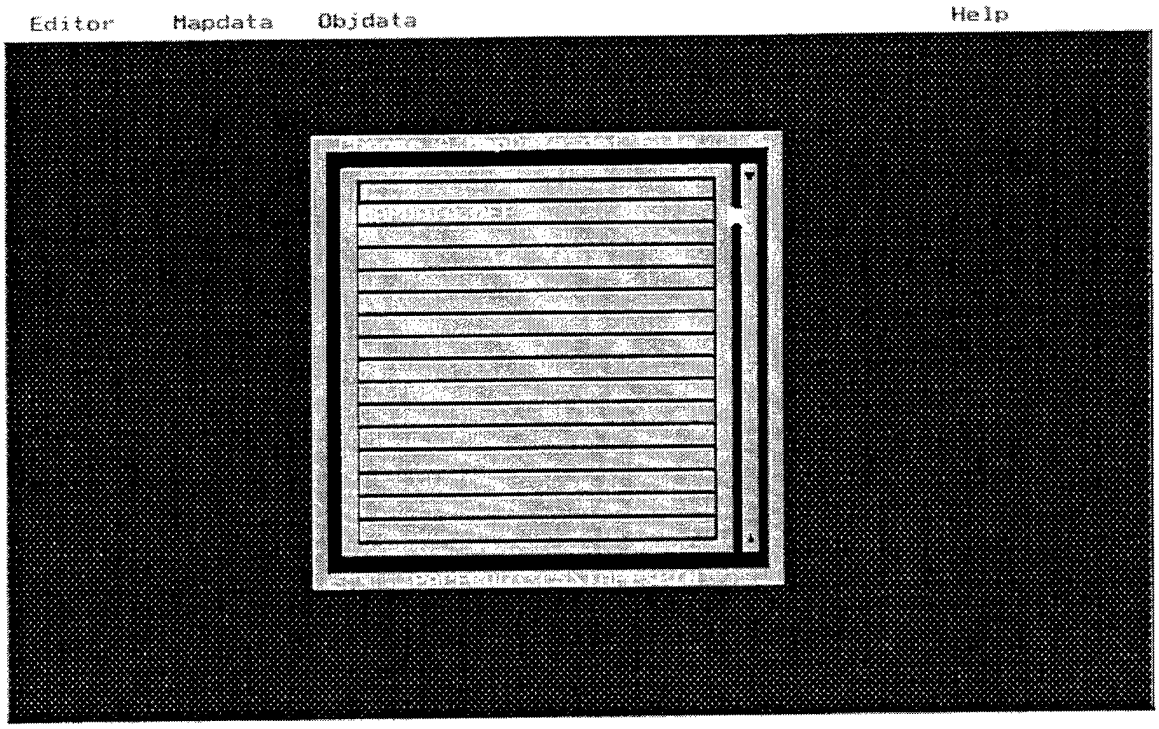


Figure 22. Example of mapdata.def menu

Although the instructions on the box say to choose a file, you should only have one file available. You should select mapdata.def using the RIGHT mouse button to open the file. Figure 22 is an example of how the screen will look once the file is opened.

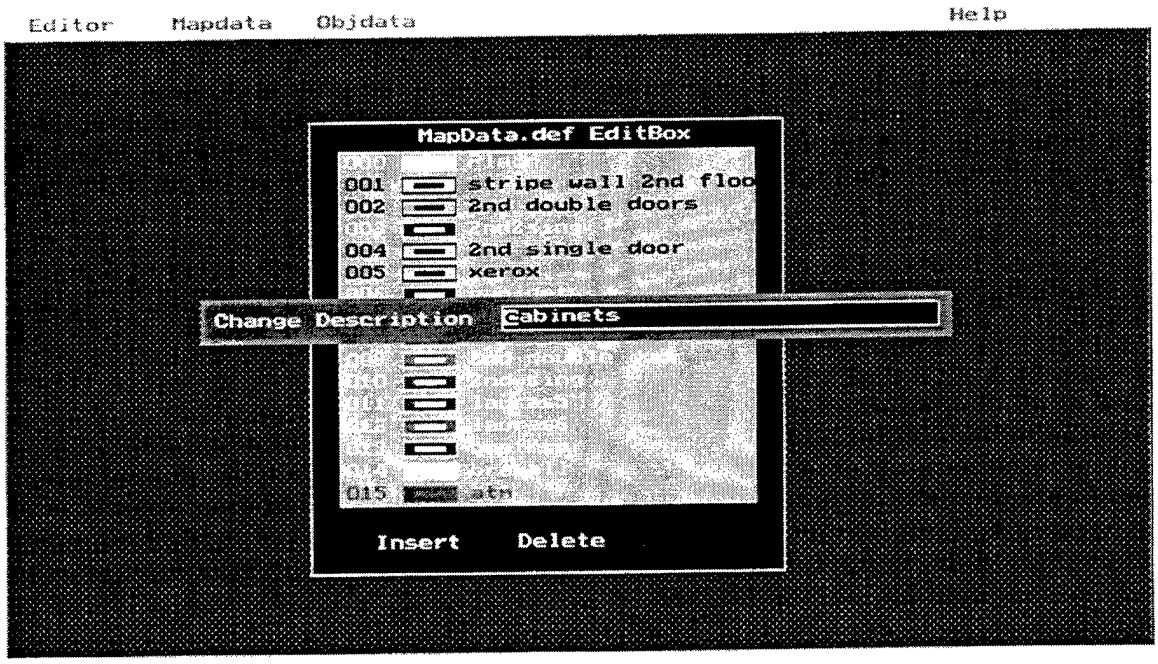


Figure 23. Example of mapdata.def menu

When using the World Editor, the mapdata.def file is organized as described in 3.2.1.: the first field is the item number, the second field is the map piece icon's image, and the third field is the item description. The difference is that the actual icon appears in the second field instead of a field value. You would edit this file to add wall icons, change an icon color, or change an icon's descriptive name. To edit this file, click on a line with the right mouse button. Once you click on a line, that item's description will appear on the screen as shown in Figure 22. You can edit the description by typing in the corresponding block. If you are creating a new icon, the block will be blank, and you should type in the name you choose. Once you have finished, press enter. The only editing you will do is to the second and third fields of the file; icon image (as discussed below) and item description. You will not edit the first field (item numbers) in the World Editor. This is done automatically.

IMPORTANT: *The icons in the mapdata.def and objectdata.def files will be used as map pieces and represent the .pcx image files in the World Database (3d.map) file. Therefore, they should be listed in the SAME order as the .pcx files in the 3d.map file. An image description DOES NOT have to be identical to the .pcx filename since it is for descriptive purposes only, but the .pcx filenames listed in 3d.map MUST be in the 3d directory. Note: You can change the name of a .pcx file while editing the .def files by pressing 'c'.*

Next you will be selecting the foreground and background colors for the item's map icon. The icon will represent the map piece that you will be placing on the drawing board so you may want to try and relate the icon colors to the .pcx image. To choose the colors, press either F for foreground or B for Background. See Figure 23.

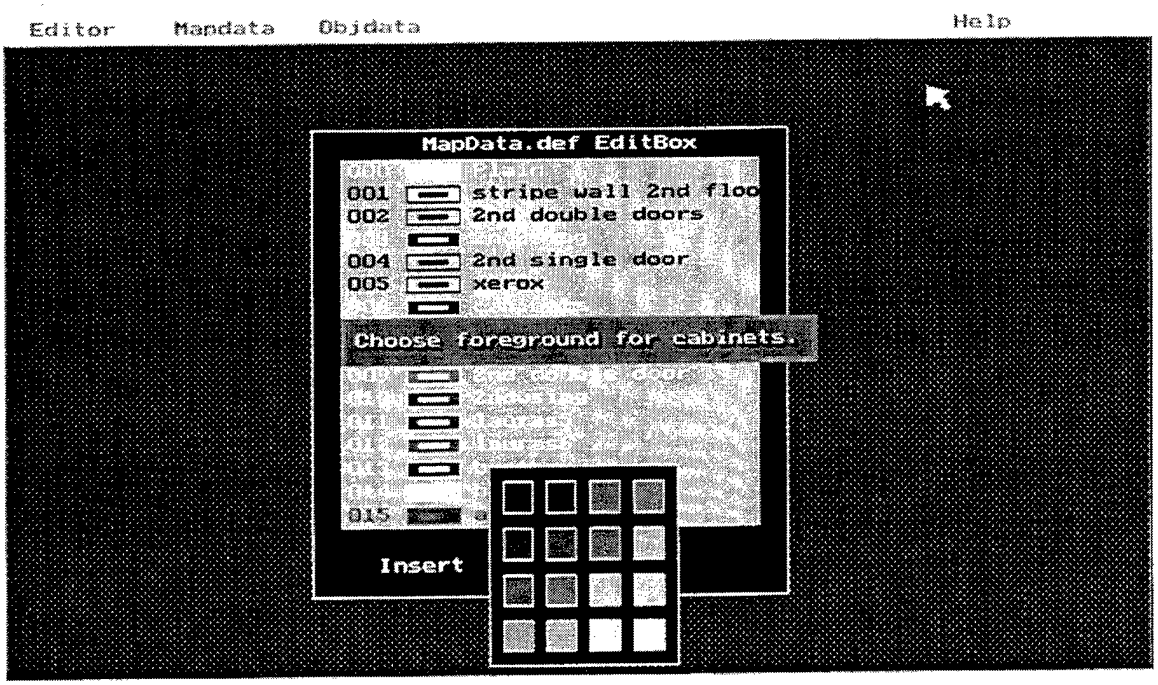


Figure 24. Example of editing the foreground color for the map piece icon in the mapdata.def file

Figure 23 shows how the screen will appear if you're editing the foreground. To choose a foreground color, simply click on a color with the left mouse button. Repeat the process for background color.

Editing the *Objectdata.def* file is identical to editing the Mapdata.def file, with the exception of the shape of the icons. You are permitted to change the shape of an icon in the objectdata.def file. Do this by pressing the 'x' key and choosing any of the patterns on the menu.

IMPORTANT: When you have completed editing a .def file, you must press <Enter> to save the file. ESC will NOT save the file.

4.1.3 INSERT and DELETE

At times, you will find it necessary to insert or delete an item from the .def files. As seen in Figure 22, there are two selections, insert and delete, which you may use to do just that. To insert an item, select the item in the file that you want directly below the inserted item. Click on it with the left mouse button. A line will be inserted in the file which reads "empty." Click on that file with the right mouse button and edit as usual. To delete an item, click on it with the left mouse button.

IMPORTANT: When you add or delete a file from the mapdata.def or objdata.def files, you MUST manually insert or delete the item in the 3d.map file in the same order as it appears in the .def file.

4.2 Understanding the Map editor

The world editor provides a graphical interface called the Map Editor for use in editing the 3D environment. It works similar to many of the standard commercial paint programs on the market using selection lists and multiple drawing tools.

For an example of what the Map Editor looks like, see Figure 25.

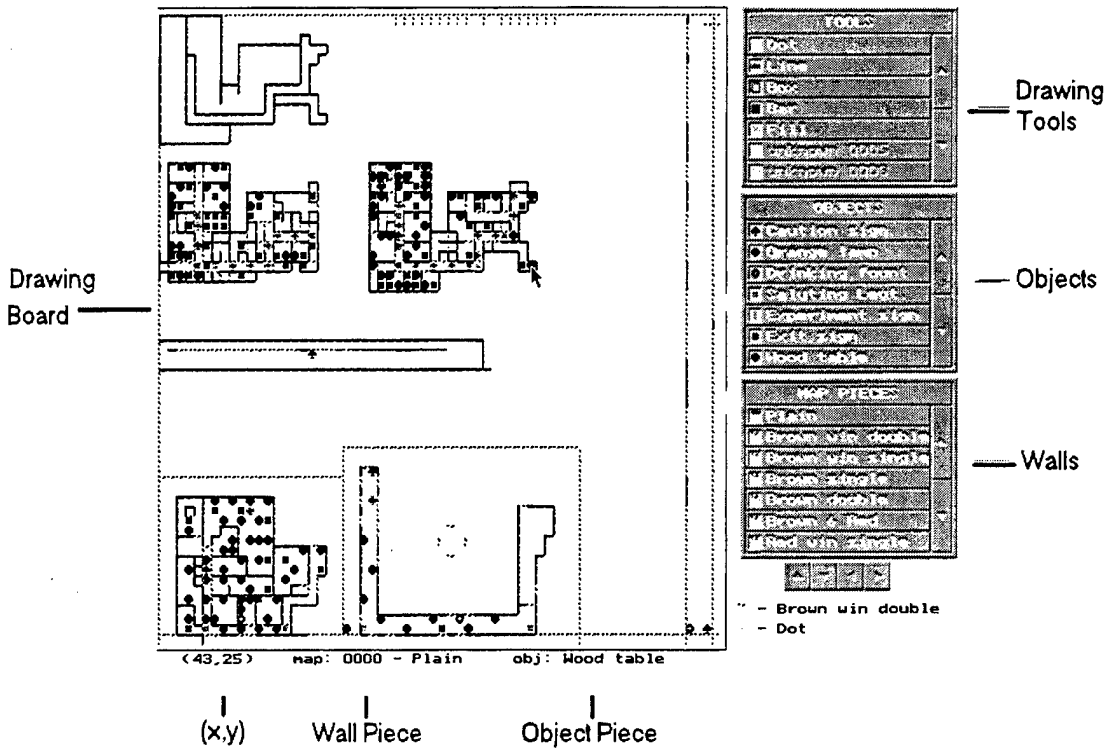


Figure 25. Map Editor

Drawing board: the bird's eye view drawing area representing the environment

Drawing Tools: Lists the tools available for drawing the environment

Objects: Lists the object image files and their icons as defined in the objectdata.def file

Walls: Lists the wall image files and their icons as defined in the mapdata.def file

(x,y): displays the x,y position of the mouse cursor on the drawing board. Note that the upper-left hand corner is (0,0) and both x and y increase toward the lower right corner

Wall Piece: Displays the name of the wall map piece that the cursor is pointed to on the drawing board. If pointed to an object or nothing at all, this will read 0000 - Plain.

Object Piece: Displays the name of the object map piece that the cursor is pointed to on the drawing board. If pointed to a wall or nothing at all, it will say nothing.

4.2.1 Understanding the Map Editor Coordinate System

4.2.1.1 The (X,Y) Plane

• X increases as you move to the right (east); Y increase as you move down (South):

Note that this is a left-handed coordinate system. As a result, when doing math to calculate coordinates, the Y value must often be negated since the standard math model uses up as positive Y.

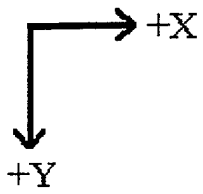


Figure 26. (X,Y) Plane.

4.2.1.2 Angles

• +X is 0; angle increases counter-clockwise, in degrees:

This is conceptually the same as the mathematical model where the angles start from east and increase as the angle goes in the direction of north. Note that when using the /showcoords command line option and **POSITION** in the waypoint command file, this is the system being used.

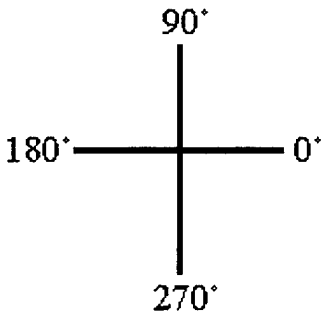


Figure 27. Angles in the Map.

To find the position of a block, always start with the mouse in the upper left-hand corner at (0,0). Move the mouse from this position to the desired position. If you do not start from the upper left-hand corner, you may get an incorrect reading.

4.2.2 Building the Environment Using the Map Editor

The Map editor, as described earlier, is basically a canvas where we draw, or build, an environment. Selection boxes allow the user to easily choose from a large number of drawing tools and images by using a mouse. The selection boxes are located to the right of the drawing board and are categorized into three groups: drawing tools, objects and walls. Below is an example of the Tools selection box.

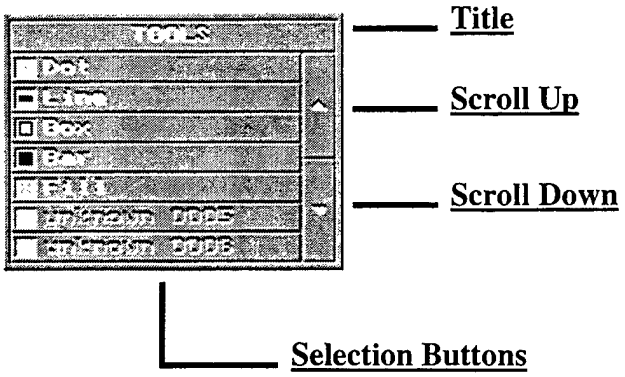


Figure 28. Map Editor Selection Box

The scrollers shift the list of items up or down so that all items can be viewed. You can select an item by simply clicking on it.

4.2.2.1 Tools

Tools affect the way that objects and map pieces are drawn on the screen with the mouse. They are provided to speed up the map drawing process.

The tool selection box is located on the top-right hand corner of the map editor's screen. Tools can be selected by moving the arrow cursor over the item and clicking on the cell of the desired tool.

The tools that are currently available are: dot, line, box, bar, and fill. The **dot** tool is used to place one item on the map at a time. When clicking on the map using the dot tool, one space is filled with either a wall or an object, depending on which type of piece is selected. When dragging the dot tool, a scribbling effect can occur if the mouse is moved too quickly. Drawing will continue as long as the mouse button is held down.

The **line** tool provides for simple line drawing. If the mouse button is clicked, this tool acts like the dot tool, but isn't as precise. The standard use is to drag the mouse from the starting position to the ending position of the line then release the mouse button.
The **box** tool creates a box. This is useful in initially creating rooms by setting up the outside walls. The clicking and dragging functions work similar to the way the line does.
The **bar** tool creates a filled box. This tends to be most useful as a large area eraser. The clicking and dragging functions work similar to the line and bar.

The **fill** tool is very common in professional paint packages. To fill an area, click on the area that should be filled. The fill will not cross wall boundaries. When filling in map pieces, all pieces connected to and of the same type as the original are changed to the currently selected map piece. If you begin a long fill and change your mind, you can terminate the fill by pressing the space bar or most any other key. Sometimes the fill tool will not fill an entire area due to memory constraints. This usually occurs only when filling areas of more than about half of the world. If this happens, simply follow up with additional fills or with the dot or bar tool.

4.2.2.2 Wall Pieces

The third selection box from the top is the Walls selection box. It contains a list of all of the wall pieces that can be used in designing a world. To create rooms or add walls to the map, select the type of tool you want to use from the Tools selection box. Then go to the Walls selection box, and click on the wall piece you want to place in the environment. You should then move the cursor over the map where you would like to place the wall, then press the left mouse button. You will see a map piece appear on the map.

There are two ways to erase a map piece. The first is to select an "Empty" item from the Walls or Objects selection boxes depending on what you want to erase. If you have many wall pieces, it may be toward the end of the list. To avoid scrolling endlessly down,

scroll up. The selection boxes allow scrolling in both directions and wrap around when the end or beginning of the list is found. Select and place the "Empty" piece on the map. Doing this will replace, and therefore erase the previous walls.

The second, and far more convenient, way to erase walls is with the right mouse button. Clicking on an item with the right mouse button performs the same as using the "Empty" piece with the left button. The advantage is that you don't have to move your mouse over to the selection bar, scroll up, then go back to where you were, instead, you just erase.

4.2.2.3 Objects

Like the walls, objects have their own selection box. Object pieces function almost identically to the map pieces. Just select an object and begin drawing with it, or use the right button to erase. The main difference between the two when creating the environment is that the walls are placed on the perimeter of a coordinate block, and are thus arranged as horizontal and vertical pieces only. Objects, on the other hand, are always placed in the center of a block. Therefore, using an object at a particular set of coordinates does not preclude the use of north-south and/or east-west wall(s) there. Just remember, only one object may occupy one block, and only one map piece may occupy one side of a block.

Just before the "Empty" object in the selection list, there are 8 arrows. By placing one of these on the map, the starting position of the operator is defined. The arrow specifies the direction the player will be facing when the 3D World viewer loads this world. If more than one of these arrows exist in the world, the one closest to the southeast corner is used. However, it is advised that only one be used to ensure future compatibility.

5.0 VIEWING THE ENVIRONMENT

5.1 Starting the Program

5.1.1 Hardware Requirements

3D World is written in Borland 'C' 3.1 and will run adequately on any IBM PC with at least an Intel 386 processor (or equivalent) with 8 megabytes installed RAM. 3D World is also capable of playing '.mid' and '.voc' files through an authentic Sound Blaster Pro sound card.

5.1.2 Configuring the Computer

3D World uses expanded memory, and the config.sys file must be configured to allow this. In the device line which contains emm386.exe, RAM and h=255 must be added. If the statement NOEMS occurs in this line, it must be removed.

Example:
Device=c:\dos\emm386.exe RAM h=255

5.2 Running 3D World

To begin the program with the default parameter files, simply type 3d<ENTER> at the command prompt. There are several parameters that can be changed when starting 3D World, and the following is a list of the additional available command line options. To use these when running 3D World, type:

3d /<command>

- **/comm**

  This tells 3D World that there is a computer hooked up to COMM port 2 and that whenever a 'yocompass' command is encountered in the waypoint file, the 3D World program will send a 'G' to the computer hooked to COMM port 2 and then will wait until the external computer sends a 'G'. **WARNING**: ***This command has the ability to lock up the 3D World program if used in conjunction with the 'YOCOMPASS' argument and there is no external computer attached.***

- **/data**

  This tells 3D World that subject data is to be collected. The information that is required is the subject's initials and a ten digit identification number. An output file is created based on the information given here. If that file already exists, a prompt will be displayed asking if the file should be overwritten. If the answer is yes, the old file will be deleted and all of its information will be lost.

- **/lowres**

  Using this option causes 3D World to run in a low-resolution mode. This results in significantly better performance on low-end machines. However, this mode is not recommended for machines capable of running at a normally acceptable frame rate because 3D World run in low resolution has severely degraded image quality.

- **/p=*SomeNumber***

  This tells 3D World what size to make the wall and object textures after they are loaded. The default size is 128 by 128 pixels. *SomeNumber* represents this size in pixels. The images must be square, have sides that are a multiple of two in length, and must be less than or equal to 128. So, 2, 4, 8, 16, 32, 64, and 128 are the only valid values. The only reason to use a value other than 128 would be to save EMS memory.

- **/showcoords**

  This has been made for debugging purposes. It displays the operator's position and facing angle while running 3D World. This can be useful for detecting misplaced waypoints.

- **/sound**

  By including this option, SoundBlaster support is enabled, therefore, this option requires a SoundBlaster Pro with the following drivers: SBFMDRV, SBMIDI, and SBSIM. If this option is used when a compatible card is not present, 3D World will generally exit with an error message. However, it may lock up the system with some cards, requiring a reboot.

- **/w=*SomeNumber***

  The default waypoint command file is 1.way. This command accesses other waypoint command files. *SomeNumber* matches the number used in the alternate waypoint file name.

5.3 Navigating in the Environment

Movement in the program is controlled by the number pad and/or the cursor keys. Figure 27 describes the function of the number pad keys:

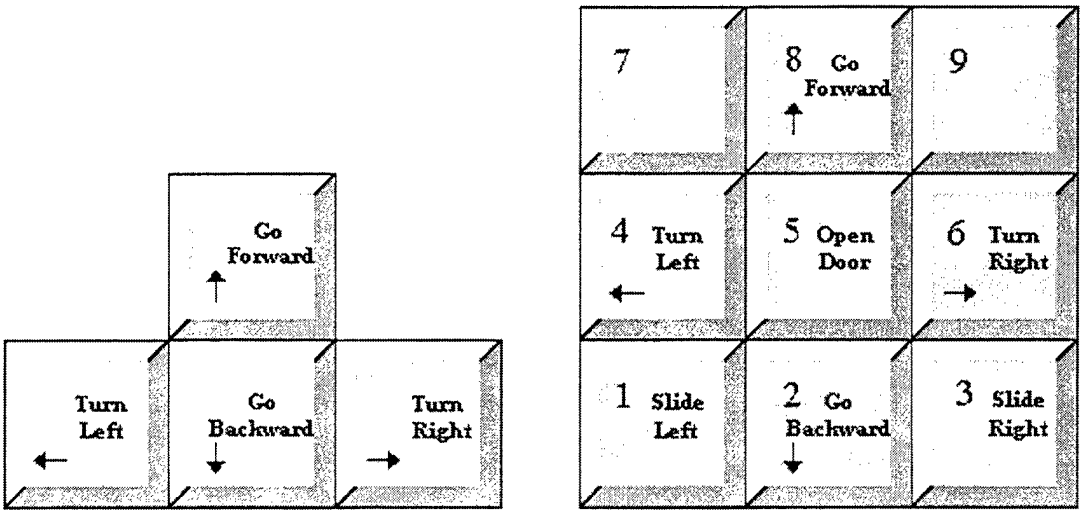


Figure 29. Navigation Keys

There are other keys and key combinations that are active during navigation as well.

SPACEBAR

The space bar is used to open a door. In the simulated environment, the operator must be facing the door (within 45 degrees, non-inclusive) and be closer than 4 feet away.

USEKEYS/SHIFT

In the **USEKEYS** mode, holding down <SHIFT> while walking causes the operator to move twice the normal speed.

ALT-H
Displays the HELP Screen

ALT-M
Displays the Overhead Pictue

ALT-Q
Quits the program and closes the output file.
While navigating through the environment, waypoints will be activated. Navigation control will shift to the waypoint commands then return to the navigation keys when all commands in the waypoint have been executed.

5.4 Using Dialogue Windows

Dialogue windows are used to present the operator with textual information. A dialogue window simply displays text on the screen and can be scrolled up and down with the arrow keys. When the operator has completed reading the text in the window, he or she can close it by pressing <ENTER>.

A Dialogue Window is created whenever the ECHO command is used (see Section 3.5.1.4). There is no limit to the amount of text you can put into a dialogue window, but if you have a lot of text, it is suggested that you divide it up into several windows. You would do this by using the NOP command between echo commands in the waypoint file.

Example:

echo "Today is your first day as an Office Assistant in the Psychology Department. You
echo "will be running errands, making copies, purchasing materials from the bookstore,
echo "and performing other miscellaneous tasks.
NOP
echo "You should see your supervisor to get a list of the tasks you are to perform.

In the above example, two dialogue windows will be displayed. The first one will contain all the text that follows the ECHO commands in the first three lines. The operator will then close that dialogue window and then the next window will appear on the screen. This screen will contain all the text following the next ECHO command. Figure 28 is an example of a dialogue box as it appears on the screen.

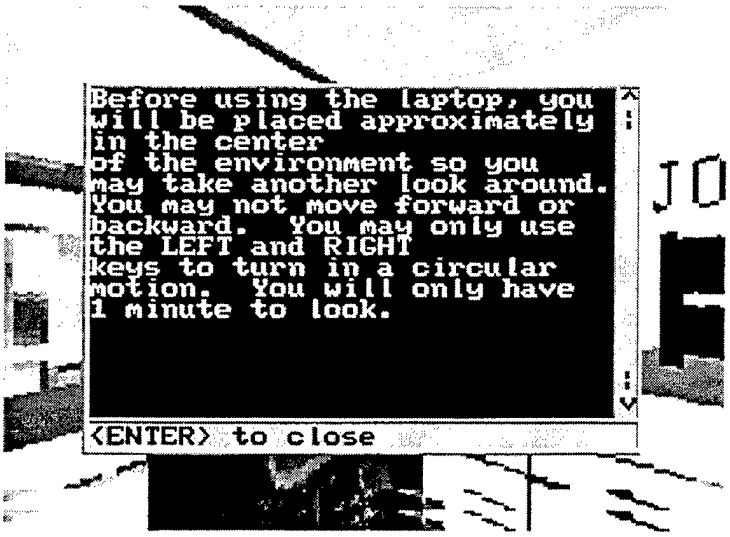


Figure 30. Example of Dialogue Window

6.0 DATA COLLECTION / OUTPUT FILES

6.1 .out file

3D World provides for automated data collection. Type 3d/data to start the 3D program. When 3D World begins, it will ask for the first and last initials, and identification number of the operator. This information is used to construct the name of the .OUT file as well as to give a short header for each line in the file. The following is the construction of the filename:

first and last initials + identification number + .out

Example:
ss0956.out

The .out file will be saved in the same directory as the program.

Each line in the .out file will start with the first and last initials and identification number. The next part of the .out file will be a description of what waypoint command was used. The final part will be the input. The line is divided by colons, so it is easy to import the file into a spreadsheet. The following is what each command will give as an output:

Command	First and Last Initials	ID Number	Message	Type of Input	Type of Input
input	Initials:	ID#:	Message given to operator:	Input from operator	
record_time	Initials:	ID#:	Message specified in command:	Time from counter	
swat	Initials:	ID#:	SWAT:	Input from operator	
time	Initials:	ID#:	Elapsed time:	Time	
copier*	Initials:	ID#:	Copier:	Number	
file_on*	Initials:	ID#:	File:	Name	
phone*	Initials:	ID#:	Phone:	Correct or Incorrect	
menu*	Initials:	ID#:	Menu:	Quantity: (default=1)	Food Item

*The copier, file_on, phone, and menu commands are described in Appendix A.

If none of these commands appear in a waypoint file, or if they were not activated, the .out file will be blank.

6.2 .mov file

All files ending in .mov are generated by commands in the waypoint command file. They define a series of moves and consist of a recording of the character codes written to the keyboard buffer. The recorded commands include all keys active during a "PLAY" session. In order to create a .mov file, you will need to run the program with the **RECORD** and **NORECORD** commands in the waypoint file. 3D World will record all movements between those two commands, beginning immediately following **RECORD** and ending with **NORECORD**. You would then remove those commands from the waypoint file, and insert the **PLAYBACK** command to play back the recorded movements. See Data Collection Commands under the Waypoint Running File, Section 2.1.3 for more information on these commands.

7.0 QUICK-STEP GUIDE TO CREATE A VIRTUAL ENVIRONMENT

Here are six easy steps to begin building an environment in 3D World. It is assumed that wall and object .pcx files have already been created or taken from a library. All files must be in the same directory to run the program. The 1.way file and the 3d.map file must be created before the program can run.

Step 1: Create the World Database (3d.map) file.
Step 2: Edit the mapdata.def and objdata.def files.
Step 3: Draw the environment.
Step 4: Create a waypoint (1.way) file
Step 5: Test the environment.
Step 6: Finish the waypoint file.

Files that are necessary to begin building an environment:

1.way	3d.exe
3d.map	editor.exe
objdata.def	editmap.exe
mapdata.def	initmap.exe
tools.def	egavga.bgi

Step 1:
The .map file must be created to set the parameters for the environment and to list the .pcx files that are to be used as walls and objects.

- Wall pieces should be listed under the [pic] heading.
- Object pieces should be listed under the [obj] heading.

The following is an example including how to add an open door, a wall, a closed door, and a table.

```
[map]
map.3dm

[parameters]
eyelevel 5.0
stepheight .50
speed 31
stepdist 3
turnrate 100
```

[pic]
thrudoor.pcx opendoor window
wall.pcx
bluedoor.pcx door

[obj]
blutable.pcx

[overhead]

[help]
help.pcx

Step 2:

There are two different methods of editing the mapdata.def and objdata.def files. It can be done manually or the editor can be used. The editor is the easier method; however, there are times when you may want to edit the files manually.

To use the editor:

- Type editor at the prompt.
- Left mouse click on the MAPDATA pull-down menu.
- Left mouse click on EDIT.
- Right mouse click on MAPDATA.DEF.
- Press <Enter> or right mouse click to change the description. Although it is referring to that specific file, the description does not have to be identical to the filename of the .pcx file.
- Press <Enter>.
- To change the color of the foreground, press <f>.
- To change the color of the background, press <b>. It is necessary to give each piece it's own color combination so the pieces can be differentiated when building the environment. Note: Changing the foreground and background colors do not affect the colors of the actual picture.
- Enter a description for each of the pictures found in the [pic] section of the .map file. Remember to keep them in the same order.
- When all pieces have been described, press q to save the file.
- Repeat this procedure for the objects, starting with the OBJDATA pull-down menu.
- To exit the editor, click on EDITOR.
- Click on exit.

Step 3:
Create a new map using initmap.exe. When this command is used, map.3dm will be created to give a new blank map.

- Type initmap at the prompt.

After map.3dm is created, it should be edited to place the walls and objects into the new environment. To edit the map, use editmap.exe.

- Type editmap at the prompt.
- Place walls and objects in the environment.
- Place a starting arrow in the environment.
- To quit from editmap.exe, type q, then y to save.

Step 4.
A rudimentary way point file must be created. This file will be modified and expanded later in development of the environment. For the purposes of beginning to build an environment the 1.way file must load the .map file. An example of this is as follows:

```
[start]
showpalpic keyboard.pcx
load 3d.map
play
```

The **SHOWPALPIC** command loads the picture to be displayed while the 3d.map file is being loaded. Note:* You can change the name of the 3d.map file so you can have multiple files in one directory. Simply load the one you want to use.

Step 5.
Test the environment.

- Type 3d at the DOS prompt.
- To open doors, press the space bar.
- To exit from the environment, press Alt-q.

Step 6.
Finish the waypoint command file.

8.0 TROUBLESHOOTING

• Memory is sometimes lost while running 3D World. Here is an example of an error message that may appear when 3D World is exited:

HEAP CORRUPT in DoneMapLocs!

With luck, the computer will run 3D World again, however, normally the system must be rebooted to recover the memory.

• If the palettes do not match on all of the pictures, the picture with the different palette may change the appearance of colors on the other pictures. Therefore, make sure that every picture uses the same palette.

• The .way, .map, and .def files can be edited from within Windows (using Notepad or another editor) or they can be edited in DOS by typing edit at the prompt. Editor.exe, editmap.exe, and 3d.exe must be run from the DOS prompt. 3d.exe usually does not work if Windows has already been started.

• Voice files should be saved in SoundBlaster format (11 mhz, Mono, 8-bit) as a .voc file.

• Make sure any given room has walls completely encircling it. Otherwise, you will be able to walk outside of the environment and the computer will probably lock up.

• If the editor is started from the MS-DOS prompt in Windows, the mouse may not be active. Exiting Windows and starting editor should solve this problem, however, the whole system may need to be rebooted to DOS.

• The editor will allow you to run other programs. However, it has been found that sometimes the other programs do not run correctly from editor. If this is happening, it is best to exit completely from editor and start the new program from the DOS prompt.

• There are times when the editor leaves blocks of color on the screen. This will occur when a foreground or background color is chosen. These can be ignored, they will not harm anything in the environment.

9.0 REFERENCES

Colle, H. A., & Reid, G. B. (1998). The Room Effect: Metric Spatial Knowledge of Local and Separated Regions. *Presence*, 7(2), 116-128.

Colle, H. A., & Reid, G. B. (1999). The Room Effect: Exploring Paths and Rooms with Objects Grouped Categorically and Spatially. Submitted to Ecological Psychology.

Endsley, M. R. (1993). A Survey of Situation Awareness Requirements in Air-to-Air Combat Fighters. *The International Journal of Aviation Psychology*, 3(2), 157-168.

Nygren, T. E., Schnipke, S. K., & Reid, G. B. (1998). The Effects of Workload on Groups formed by Subjective Ratings of the Subjective Workload Assessment Technique. *Proceedings of the 42nd Annual Meeting of Human Factors and Ergonomics Society*, 816-820.

Mailing Address:

Annette McCoy
Sytronics, Inc.
WPAFB, Area B, Bldg 197
Dayton, OH 45433
Annette.McCoy@wpafb.af.mil
937-255-5111

APPENDIX A. - ADDITIONAL WAYPOINT COMMANDS

Included in 3D World, there are additional waypoint commands that were used for a specific experiment. Each of these is designed to work with a particular .pcx file and for a specific purpose. If additional waypoint commands are desired, they must be programmed in the source code.

• **COPIER** accepts input from the screen via the mouse. The command works in conjunction with the copier.pcx file. The command format is: copier "<pcx file name>. See Figure 29.

Example:
copier "copier.pcx

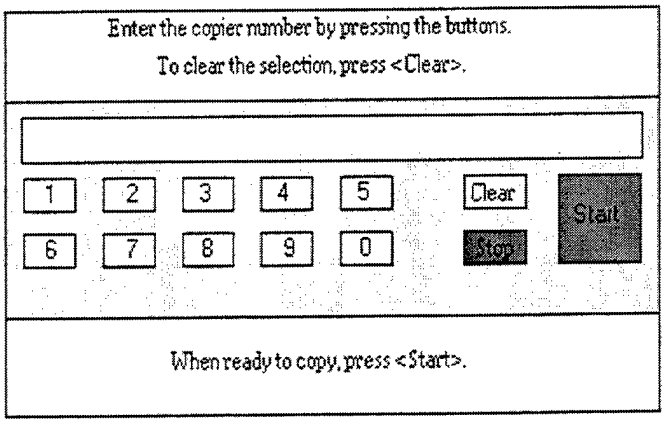


Figure 31. Copier.pcx

When a number is clicked with the mouse it appears in the copier window. When 'Clear' is clicked, the numbers are cleared. When 'Start' is clicked, the input is saved in the data file and the next command in the waypoint file is executed. To replicate this .pcx file with this function, the .pcx file must be 320x200 pixels large. The top left and bottom right corners for each of the buttons should be placed in the following configuration (in pixels):

Button	Top Left	Bottom Right
1	(9,84)	(34,99)
2	(48,84)	(73,99)
3	(87,84)	(112,99)
4	(126,84)	(151,99)
5	(165,84)	(190,99)
6	(9,110)	(34,126)
7	(48,110)	(73,126)
8	(87,110)	(112,126)
9	(126,110)	(151,126)
0	(165,110)	(190,126)
Clear	(225,84)	(255,99)
Start	(271,84)	(310,126)

Figure 32. Copier.pcx Replicate Configuration

• **FILE_ON** accepts input from the screen via the mouse. This command waits until the operator presses a specified key or they leave the current waypoint. If the correct key is pressed, an operator specified .pcx file is displayed, and the subject uses the mouse to make the selection. The command format is: file_on <user specified key>, <.pcx file to display>, <name1>, <name2>, <name3>, <name4>, <name5>, <name6>, <name7>, <name8>, <name9>.

Example:
file_on r, r_file.pcx, Karen Randall, Rowan Regal, Denny Renner, Jeff Richards, Tracy Rogers, Darrell Rolex, Sean Row, Thomas Russo, Daniel Rutski

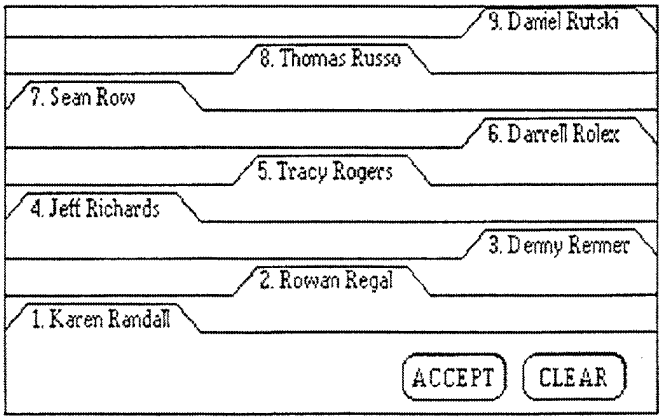


Figure 33. R_file.pcx

When a name is clicked, the output that is specified in the waypoint command appears in the bottom left hand corner. The 'Clear' button clears the name. The 'Accept' button saves the input in the data file and the next command in the waypoint file is executed. To replicate this .pcx file with this function, the .pcx file must be 320x200 pixels large. The top left and bottom right corners for each of the buttons should be placed in the following configuration (in pixels):

File/Button	Top Left	Bottom Right
1	(11,145)	(85,159)
2	(123,127)	(197,141)
3	(235,109)	(309,123)
4	(11,91)	(85,105)
5	(123,73)	(197,88)
6	(235,56)	(309,70)
7	(11,37)	(85,51)
8	(123,19)	(197,33)
9	(235,1)	(309,15)
Clear	(252,171)	(303,192)
Accept	(193,171)	(244,192)

Figure 34. R_file.pcx Replicate Configuration

• **PHONE** accepts input from the screen via the mouse. The command works in conjunction with the phone.pcx file. The command format is: phone "pcx file, <phone number 1>, <phone number 2>, <phone number 3>, <phone number 4>, <voice file for phone number 1>, <voice file for phone number 2>, <voice file for phone number 3>, <voice file for phone number 4>, <voice file for wrong number>. If one of the correct phone number is dialed, the corresponding .voc file will be heard. If a wrong number is dialed, the wrong number .voc file will be heard. When either all four correct numbers are dialed or the wrong number is dialed twice, the command is terminated and the next command is executed in the waypoint file.

Example:
phone "phone.pcx, 2911234, 2914567, 2917890, 2918765, message1.voc, message2.voc, message3.voc, message4.voc, wrong.voc

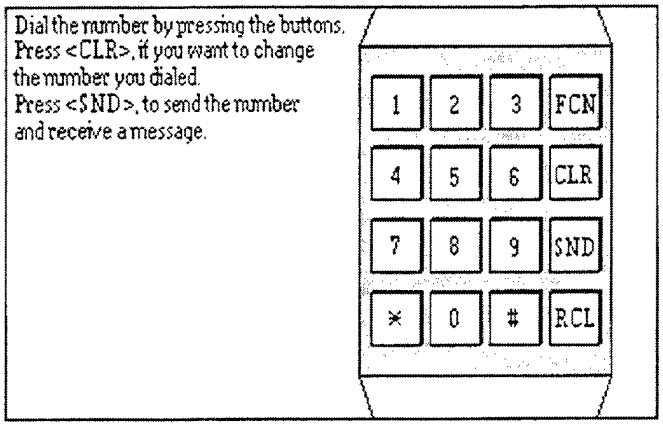


Figure 35. Phone.pcx

When a number is clicked, it appears in the bottom left hand corner. When the 'Clr' is clicked, the numbers are cleared. When 'Snd' is clicked, the input is saved in the data file and the number is cleared. The next command in the waypoint file will be executed when either all four correct numbers are dialed or two wrong numbers are dialed. To replicate this .pcx file with this function, the .pcx file must be 320x200 pixels large. The top left and bottom right corners for each of the buttons should be placed in the following configuration (in pixels):

Button	Top Left	Bottom Right
1	(181,35)	(204,59)
2	(209,35)	(233,59)
3	(238,35)	(262,59)
4	(181,69)	(204,93)
5	(209,69)	(233,93)
6	(238,69)	(262,93)
7	(181,103)	(204,128)
8	(209,103)	(233,128)
9	(238,103)	(262,128)
0	(209,139)	(233,163)
Clr	(267,69)	(291,93)
Snd	(267,139)	(291,163)

Figure 36. Phone.pcx Replicate Configuration

• **MENU** accepts input from the screen via the mouse. The command works in conjunction with the menu.pcx file. The command format is: menu "<pcx file name>.

Example:
menu "menu.pcx

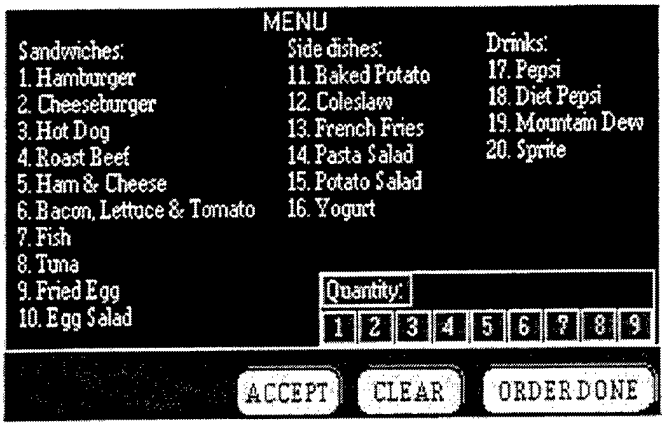


Figure 37. Menu.pcx

When a number is clicked on, the output appears next to the word 'Quantity'. When a food item is clicked on, the output appears in the bottom left hand corner. These particular food items are programmed in the source code. The 'Clear' button clears the order. The 'Accept' button saves the output of the quantity and food item in the data file. The 'Order Done' button exits this command, and the next command in the waypoint file is executed. To replicate this .pcx file with this function, the .pcx file must be 320x200 pixels large. The top left and bottom right corners for each of the buttons should be placed in the following configuration (in pixels):

Quantity/ Button	Top Left	Bottom Right
1	(155,149)	(171,164)
2	(173,149)	(189,164)
3	(191,149)	(207,164)
4	(209,149)	(225,164)
5	(227,149)	(243,164)
6	(245,149)	(261,164)
7	(263,149)	(279,164)
8	(281,149)	(297,164)
9	(299,149)	(315,164)
Accept	(113,176)	(162,198)
Clear	(172,176)	(222,198)
Order Done	(232,176)	(316,198)

Figure 38. Menu.pcx Replicate Configuration

Food Item Number	Top Left	Bottom Right
1	(5,28)	(67,38)
2	(5,42)	(75,52)
3	(5,55)	(53,65)
4	(5,67)	(62,77)
5	(5,81)	(80,91)
6	(5,93)	(123,103)
7	(5,107)	(33,117)
8	(5,120)	(37,130)
9	(5,132)	(58,142)
10	(5,146)	(63,156)
11	(137,28)	(208,38)
12	(137,42)	(189,52)
13	(137,55)	(205,65)
14	(137,67)	(200,77)
15	(137,81)	(204,91)
16	(137,93)	(182,103)
17	(234,26)	(270,36)
18	(234,40)	(290,50)
19	(234,53)	(274,63)
20	(234,65)	(274,75)

Figure 39. Menu.pcx Replicate Configuration (continued)