Defendroid Help Contents

I can be contacted at:
gsummers@physics.ox.ac.uk
WWW Page:
http://sable.ox.ac.uk/~symons/glen/glen.html

Introduction

This program is a Windows based version of the classic old arcade game Defender. It uses DISPDIB.DLL for direct access to the VGA hardware for really fast graphics under Windows. There is also a WinG option for running the game in a window, but performance will be significantly reduced.

Please e-mail me any bugs, comments or suggestions you may have about the game. And send me your high score tables and Demo Mode DLLs if you like.

Reasons for Writing Defendroid

There must be many versions of Defender around already, I wrote this one to see if a fast playable game could be written easily using the Windows environment.

Game Controls

The game can be controlled with keyboard, mouse or joystick.
Control type is selected on the Options.Control sub-menu.
The controls can be configured on the Options.Control Config sub-menu.

Keyboard

Keys control ship movement up/down, reverse, laser fire, thrust, smart bomb and hyperspace.

Mouse

The ship will move towards the mouse pointer. Key/button presses control laser fire, thrust, smart bomb and hyperspace.

Joystick

Joystick movement controls up/down/left/right. Key/button presses control laser fire, smart bomb and hyperspace. The thresholds for movement/thrust can be modified in the joystick configuration dialog.

Additional keys

F2	Start Game
'P'	Pause/Unpause
ctrl+'Q'	Quit current Game

(The game will pause if you switch to another application, or open a dialog box.)

Future of this program

Defendroid will most likely propagate to Win32. Stargrate and Defendroid 2001 versions may materialize. I will try to get hold of the original graphics/sound effects. The gameplay will be tweaked to make it more like the original. A network version might be a possibility. I also intent to incorporate a graphical rather than script based customization utility.

Defendroid Internals

The program is written mostly in C++ with all the graphics routines in assembler. The game kernel took about 2 weeks to write and consists of about 5000 lines of code. However, tweaking the game into a releasable form and adding the customization facilities took rather longer.

DEFEND.DAT is a LZW compressed file container that is the receptacle for all the files necessary for running the game, including data files which describe the game objects, how they interact and a level database. In principal this file could be replaced and the entire nature of the game could be changed. The game allows additional DAT files to be merged in enabling customization of the graphics and sounds. (See customization)

Game objects are handled as C++ classes, more complex objects are derived from the simple base object class. Classes exist for special purposes like target acquisition, a object derived from these additional classes automatically gains its functionality.

How to Play

Press F2 to start a game.

Use the controls to control your ship. Shoot the aliens, save the humanoids. When everything is dead you go on to the next level. The game is over when you run out of ships.

The ship's status indicators shows score, lives and smart bombs left.

Notes on Performance

There is not much you can do to speed up the DISPDIB.DLL version of the program (buy a faster computer) except by reducing the number of background stars. However while running with WinG, the speed of the program is likely to be limited by the time taken copying regions of the off screen buffer to display memory. It is thus essential to have WinG correctly installed on your system and to be running in a suitable graphics mode (i.e. 256 colors).

For some reason WinG seems to handle many small objects very poorly. The background stars are an example of this. You will probably want to disable these stars when using running under WinG. See Display Dialog

For local bus graphics systems (32 bit, 25 or 33MHz) the program should run quite smoothly. However, slower ISA graphics cards (16 bit, 8 or 11MHz) could cause the program to run very slowly. The overall scaling factor applied when copying the image to the screen can be modified, the default value is 2 but by setting this to 1 the game will run significantly faster.

For slower machines I suggest you use the lowest resolution 256 color screen mode available (e.g. 640x480), and maybe use a scale factor of 1 (this will however make the playing area appear quite small).

For faster machines any 256 color mode should be sufficient.

If WinG seems to be running sluggishly, check that the line: device=dva.386 appears in the [386 Enhanced] section of your system.ini.

WinG

WinG is Microsoft's high performance graphics library enabling bandwidth limited memory transfers to the display.

Display Dialog

The display dialog allows you to select whether the game will run in fullscreen VGA using DISPDIB.DLL or whether it will use WinG. Also the number of background stars can be modified.

See Notes on Performance

Sound Configuration

The game incorporates my own sound mixing code as I could not get WAVEMIX.DLL to work at all on my system. The sound configuration dialog is on the Options.Sound Config menu.

Experiment with the number and size of buffers used. The default number was sufficient to eliminate clicks on my computer.

On some computers using WaveOutReset causes load clicks or pauses. If this happens try the remix option. However this may cause distortion or slight timing problems.

If you do not have a sound card then PC speaker sounds can be enabled (simple tone based sounds). The synchronous speaker driver *can* be used but the game will hang during playback.

Configuration

Demo Dialog

The demo mode dialog, accessible from the File.Demo Control menu item, allows you to add/remove Demo Mode DLLs and view the games played and average score achieved. The best score achieved should appear in the High Score Table (an asterisk in the table indicates demo mode scores).

The currently selected demo mode in the list box indicates the next demo to play.

Add
Opens a file dialog to load a new Demo Mode DLL.

Remove
Removes the currently selected Demo Mode.

Info
Gets information on the currently selected Demo Mode. The number of games played and average score achieved is shown.

Terminate demo on key/mouse.
This option indicates whether key presses/mouse actions terminate demo mode play.

Sound on in demo.
This option allows sound in demo mode play. You must first have sounds enabled in the Sound Dialog

No frame sync in demo mode
This option allows fast demo mode play for the purpose of faster statistical analysis of scores attained.

See also Writing your own Demo Mode DLL

Patch Dialog

The patch dialog, accessible from the File.Patch Control menu item, allows you to add remove custom patches from the game. This enables sounds and graphics in the game to be fully customized.

The order the patches appear in the list box indicates the order they were loaded in. This is important when they overwrite each other.

Add
Opens a file dialog to add a new Patch file. The new patch file patches the current state of the game, so the effect of previously loaded patched may be overwritten.

Remove
Removes the currently selected Patch. All the patches will be removed and then reloaded in the order they appear the list box. This ensures consistency.

Info
The author, description and the number of sounds and graphics modified by the patch file are indicated.

See Creating a Patch File

Writing your own Demo Mode DLL

If you installed the customization tools you will find the files **DEMO.H** and **NEWDEMO.CPP** in the **CUSTOM** subdirectory to the main Defendroid directory.

These files are the basis of a DLL which can be loaded by the game to service control requests for Demo Mode play.

The following functions should be exported by the DLL:

DemoInfo * _export CALLBACK Initialize(DemoProc First,DemoProc Next);

The Initialize function is called once when the DLL is loaded. The function pointers First and Next should be saved for later use. The function should return a pointer to a DemoInfo structure containing game target, version, demo name and demo high score entry *e.g.*

```
DemoInfo Info={"Defendroid",VERSION,"Test Demo","Test Demo"};
```

void _export CALLBACK GameInit();

The optional game initialization function is called when a game is started. You can use this to reset internal structures in the DLL.

void _export CALLBACK ControlDemo(GameInfo *G);

The ControlDemo function is called once per game frame when control information is required. See GameInfo. This function should have some code to receive information on the objects in the game *e.g.*

```
GameObject O;
First(&O);
while (O.Valid)
{
// make decisions based on contents of structure O
Next(&O);
}
```

Where *First* and *Next* are the function pointers passed by Initialize().

void _export CALLBACK GameOver(DWORD Score,WORD Smarts);

This (optional) function is called at the end of each game informing the DLL of the attained score and smartbombs left over. This information could be used to try and improve the performance of the Demo by modifying some internal parameters i.e. in a learning algorithm.

You can load a compiled Demo Mode DLL info the game in the Demo Control Dialog.

GameInfo

```
struct GameInfo
{
        int Lives,SmartBombs;
        DWORD Score,Frame;
        BYTE *Landscape;
        BOOL Scape;
        GameObject Ship;

        Controls *C;
};
```

Lives	The number of lives your demo mode has left.
SmartBombs	The number of smartbombs left.
Score	The current score.
Frame	This starts a zero and is incremented once per game frame.
Landscape	Points to an array of 4096 bytes representing the landscape. The values go from 0 at the top of screen to 240 at the bottom.
Scape	True if Landscape exists.
Ship	GameObject structure containing information about your ship.
C	Pointer to a Controls structure to fill with desired control actions.

GameObject

```
struct GameObject
{
        int Flags,Type,User;
        int x,y,w,h;
        long xv,yv;
        BOOL Valid;
};
```

Flags Various flags containing information about the object.
Type The type of the object.
User Depends on the Type of the Object
x,y The x,y location of the object.
w,h The width and height of the object.
Valid A flag indicating if the structure contains valid infomation. This is used by the First() Next() functions to indicate the last valid object.

Notes

The object types currently n the game are defined thus:
enum {SHIP1=0, HUMANOID, LANDER, MUTANT, PULSAR, POD, PODBIT, BAITER, LASER, BULLET, PULSBUL };

The object flags in GameObject.Flags
#define O_DEAD 1
#define O_MATERIALISE 2
#define O_SMALLWEAP 4
#define O_EXPL 8
#define O_SHIP 16

The humanoid states in GameObject.User when Type==HUMANOID
#define H_NORM 0
#define H_HELD 1
#define H_FALLING 2

The lander states in GameObject.User when Type==LANDER
#define L_NORM 0
#define L_GRABBING 1
#define L_LIFTING 2

Controls Structure

```
union Controls
{
        BYTE byte;
        struct BITS
        {
                BYTE Up:1;
                BYTE Down:1;
                BYTE Thrust:1;
                BYTE Dir:1;
                BYTE Fire:1;
                BYTE Smart:1;
                BYTE Hyperspace:1;
        } bits;
};
```

Pass back control information to the game in this structure.

e.g.

```
        void _export CALLBACK ControlDemo(GameInfo *G)
        {
        Controls::BITS &B=G->C.bits;
        ...
        B.Up=TRUE;   // to go up
        ...
        }
```

Creating a Patch File

The GENERATE.EXE program which is optionally installed when you install Defendroid allows you to create add-in patch files. There should also be an example file CUSTOM.GEN and the various bitmap and wave files associated with it.

Generate accepts generate files (*.GEN) with the following syntax:

```
; customized generate file

[General]
Author=G.M.Summers
Description=Example dat File

BaseFile=..\defend.dat
Palette=defend.pal

; modified game parameters
[Parameters]
ShipLaserDist=1
ShipThrustDist=2

[Graphics]
identifier=filename
...

[Sounds]
identifier=filename
...
```

Currently only 8 bit 11 kHz WAV files are allowed and graphic files should be 256 color uncompressed BMP files.

See the help file accompanying the Generate program for more information.

You can load a compiled patch file info the game in the Patch Control Dialog.

Customization

Selecting merge data file from the file menu allows you to merge in customized sounds and graphics.

The generate.exe program which is optionally installed when you install Defender allows you to create add-in dat files.

Generate accepts generate files (*.GEN) with the following syntax:

```
; customized generate file

[General]
Author=G.M.Summers
Description=Example dat File

BaseFile=..\defend.dat
Palette=defend.pal

; modified game parameters
[Parameters]
ShipLaserDist=1
ShipThrustDist=2

[Graphics]
identifier=filename
...

[Sounds]
identifier=filename
...
```

See the CUSTOM.GEN file for more information.

Currently only 8 bit 11 kHz WAV files are allowed and graphic files should be 256 color BMP files with their width a factor of 4 pixels.