```
          +-----------------------------------------------------+
 Version  | SUPAPLEX SPEED-FIX, FOR ALL COMPUTERS, FAST AND SLOW! |
   6.3    |     WITH MANY EXTRA FEATURES LIKE DEMO RECORDING      |
          +-----------------------------------------------------+
```

NEW: For info about the two files TESTSIG.EXE and MYSPSIG.TXT, see below at the "full version history of changes and additions", at "v6.2 -> v6.3".

PLEASE READ FOLLOWING FEW LINES CAREFULLY.
IT IS NECESSARY FOR CORRECT OPERATION OF THE SPEEDFIX.

THANK YOU.

This program is a highly improved "replacement" for the executable file SUPAPLEX.EXE only: all other files from the original Supaplex package are necessary for the game to run, but are not included here.
The complete original (unimproved) Supaplex package SUPAPLEX.ZIP can be downloaded from the Supaplex WebPages (see below)!

Use of this program is at your own risk. We are using this program all the time, and have never had any damage, so your computer will probably not be damaged, but don't come crying to us when it is ...

First "install" the original Supaplex package if this is not already done, and keep that SUPAPLEX.ZIP somewhere safe: Don't delete it: You may need it. Do NOT use the Install program of that package: just create a subdirectory and unzip that package into that subdirectory. You're ready to run Supaplex. Find the executable file: This is either the original SUPAPLEX.EXE or the debugged and improved SPFIX63.EXE, which contains the complete SUPAPLEX.EXE.

(If you already played Supaplex, you may want to backup the Supaplex directory before using this program. Make sure you also backup your personal scores files, which are hidden files. Under DOS use "ATTRIB -h *.*" to reveal them. Under Windows change the explorer settings to show hidden files.)

SPFIX63.EXE is totally backward compatible with the original SUPAPLEX.EXE: Many bugs have been fixed, and many features have been added without changing the game! (The "new" demo's from the SpeedFixes before version 6.2 are not needed anymore: They can still be used, but they don't work with the original SUPAPLEX.EXE. Now the original game can also be used and without drawbacks.)

The best way to play Supaplex is using plain DOS, not any Windows DOS box and not the DOS from the Win9x shut-down menu! Use the F8 key (Win95) or Ctrl-key (Win98) at boot time, and start the plain DOS command line from the menu! If the mouse does not react, you may want to install a DOS mouse driver. (A mouse is not needed at all because it only works in the menu screen.) It is possible to use the (full screen) Win9x DOS box, but to prevent the jerky movements, you must change the properties of that DOS box. Win9x wants to do many things in the background which must be prevented! No guarantee!

Supaplex will not work with Windows NT, because this Windows version does not permit direct communication with the hardware at any time, to prevent crashes.

Supaplex/SpFix63 runs with Windows XP (SP2) with an NTFS partition, even without the extra program “DosBox”, but only with short file names.

Make sure you visit the Supaplex Pages at http://www.elmerproductions.com/sp/ and the forum at http://www.infordigital.com/ to find all about Supaplex.

+-------+
| INDEX |
+-------+

FIRST INTRODUCTION

This file contains lots of information about the original Supaplex, and also information about the SpeedFix. If you are interested, read it all, if not, just read the changes below (if you are upgrading from a previous version) or all about the instructions for use and the command line parameters (if you are new to using the SpeedFix).

If you are not smarter after reading it, you might be one of the authors of Supaplex yourself.

Except for the following version history at versions v5.x, almost all "I"s and "me"s etc. refer to me: Herman Perk.

To Philip Jespersen: thanks for your support, and for the game itself.
To Robin Heydon: thanks for the PC version of the game.
To Hilde Anita Hopen: thanks for spreading the game through the internet.
To Herman Perk: thanks for making the first SpeedFixes. <Maarten>
To Maarten (Elmer) Egmond: thanks for writing SPEDIT, a Supaplex level editor. and for making all SpeedFix 5 additions.
To Digital Integration: thanks for not making problems.
To Sergei Sinicyn: thanks for showing me some problems and solutions.
To Matrox Tech support: thanks for the repeated non-support and un-answers.
To Jens Randloev-Hansen: thanks for your contributions and beta testing.
To Frans Meulenbroeks: thanks for your contributions and beta testing.
To Yonatan Rozenshein: thanks for your contributions and beta testing.
To Kim Min-Soo: thanks for your contributions and beta testing.
(His family name is Kim)

- -

SUPAPLEX

You are Murphy, bug hunter extraordinaire, exploring deep inside a crazy computer. The only way out of each brain-teasing level is to collect the Infotrons and this is where the fun begins! Snik Snaks must be avoided at all costs...and falling Zonks will trap the unwary...exploding discs, Electrons and ports add up to the coolest action game around!

(Quote from the original manual)

==========

FULL VERSION HISTORY OF CHANGES AND ADDITIONS

History v1,v2,v3,v4 (Herman Perk):
- see SUPAPLEX SPEED-FIX: INSIDE INFO AND HISTORY below.

Changes from v4 to v5 (Maarten Egmond):
- Added a little check routine to avoid loading demo files that are too large for the segment.
- Command line option EGA (Capitalized) changed to EGA (any CaSe).
- :file.sp option added to command line options.
- Pressing F11 in the menu will replay the demo and F12 will play the level.
- In stead of ---DEMO LEVEL--- as level name of a demo, the real level name appears in the panel.
- : option will activate (FORCED) player.
- Added a credit line on the title screen for my work

v5 -> v5.1 (Maarten Egmond)
- fixed a small bug in demo-loading routine, which would crash if a file was being loaded that was too big (larger than about 45000 bytes).

Usually, this error would not occur (or you'd have to run the SpeedFix with a wrong :file.sp file, or maybe record a *very* long demo?)

v5.1 -> v5.2 (Maarten Egmond)

- Fixed a bug in the demo-loading routine. If a demo was wrong, spfix51 would return to DOS without clearing the screen (and possible waiting for a key press), so it appeared if the machine hung.
- Fixed demo format (again). Random events (like "bugs") would still be randomized after recording a demo. So recording a good solution of any level with important random effects (e.g. level 9) would always result in a wrong demo. The random number used at recording-start is now saved to the demo, and restored when it's being played, so these random effects will always be the same in the demo (the demo always looks the same). Demo's recorded by v5.2 are compatible with v5.0 and v5.1, and vice versa. (That means, they can be played, but can give true random events in stead of recorded random events).
  (Technical:) To accomplish this I've used the last two bytes of the 1536 byte level. These two bytes were (according to my experience) not used by anything else.

v5.2 -> v5.3 (Maarten Egmond)

- Fixed pan back to menu after playing a level at startup (using colon parameter). It would flash up the screen. Now it correctly pans back.
- Fixed demo bug: playing a demo by pressing the 'demo' button in the menu, or a random demo started after a while would not run correctly most of the time.
- Added 'DEMO SUCCESS', 'DEMO FAILED', 'LEVEL SUCCESS' and 'LEVEL FAILED' messages so you can easily see if the level was ended successfully (i.e. though the exit, with enough Infotrons collected).
- Added '@' command line option. Must be used together with the colon option and a demo. This will replay the demo at VERY high speed, and quit Supaplex right after the end of it. One of the above 'DEMO SUCCESS' or 'DEMO FAILED' messages will be shown. This is handy to check large amounts of demo files to see if they run correctly. (You can test your own solutions with this option too).
- Changed cheat mode ('#' option) to set all levels to SKIPPED in stead of DONE. This way, you can keep track of which levels you have done properly in one Supaplex session.
- Timing problem at demo recording (which has been in the original Supaplex too) solved. Sometimes, when recording a demo, Murphy would walk very close to danger (i.e. Snik Snak, falling Zonk, etc.), but still pass it. Then, when replaying the demo, the level failed. This has been fixed. All demo's should now run exactly as recorded. (If not, email me).
- While recording a demo, no debug keys may be used, you can still change the speed with the gray keys. This is to enable 'us' to see the REAL playing time, in stead of the time the demo takes at default speed. So if you're really good, you can record your demo at the highest speed, and set a new record easier. Demo's will still be played at the current speed, regardless of how they were recorded.
- Ctrl-F12 will stop demo recording, and lets you play on.
- The system time will have changed after Supaplex has been run (was also in the original Supaplex). This is normal Supaplex behavior, and will cause the time in your computer to change. Time will be reset to proper time after a reboot. To keep the correct time without rebooting, you can use the TIMEFIX.EXE program from Elmer Productions (see the Supaplex Web-Pages) to fix it. If you use it, run like this: TIMEFIX 50 0 SPFIX53.EXE and it should set the correct time after exiting the SpeedFix. You can add your favorite command line options to the back.
- You can now only start demo recording when the speed auto-detect routine is done. (Just wait about a second after starting a level for the first time).
- Cleaned up this documentation to make it easier for new users.
- When playing an .SP file yourself (by using the colon option and/or by pressing F12 in the menu) the currently selected level would be set to 'solved' when you correctly solved the level. This is now fixed.
  No matter how you end the .SP level play, it will never interfere with the normal playing.

- Now when playing an .SP file, the level number displayed on the screen will always be .SP in stead of just 'sometimes'
- Renamed this readme document to SPFIXnn.DOC, which helps it being identified as a SpeedFix document easily.

v5.3 -> v5.4 (Maarten Egmond)

- Removed time bug - apparently, the gray * key was not correctly tested to change the slowest demo speed. This would not affect game/demo play, but compromise the accuracy of the timing (duration) of the demo.
- Level names are finally shown correctly all the time (also when playing demo's and .SP files)
- Fixed bug: If a demo would run when a player was selected, the time it took would be added to the player's playing time.
- Removed bug that would cause ENTER to blow up Murphy in the game when no mouse driver was active, in stead of just switching panel on/off.
- Built in system time correction. It's accurate up to 0.05% of normal system time. No need to use TIMEFIX.EXE anymore, this is much better. (If you still use TIMEFIX, it will make the time wrong, since SPFIX54.EXE lets the system timer run at (almost) 18.2 Hz itself).

v5.4 -> v6.0 (Herman Perk)

Thanks to Maarten many extra features were added and he also created most of the "savegame" code, which he never released due to hard-to-find bugs. Furthermore there still were annoying bugs to fix, so time for 6.0.

- "savegame" code enabled and debugged, so it is possible now to save a game situation. See Ctrl-W and Ctrl-L in the key list below for details. (Looking through that code I found game options that were never used: Ports that can freeze Zonks and/or Snik Snaks & Electrons, and that same Zonk-freeze option can be set at game-start, like gravitation.)
- Option added to switch Debug mode on and off during a game. See Ctrl-ScrollLock and Alt-ScrollLock in the key list below for details.
- Demo recording still contained bugs that could make a recorded demo fail.
  - One problem was to let the recording always start at exactly the same frame (frame=1/35th of a second), seen from the play-back point of view.
  - Another problem was the exact timing of the BUG sprites: those bugs fired one frame too early in the playback of the recorded demo, which also could make a good demo fail. These problems have been solved at last

  (If there are still problems: PLEASE (e-)mail and send any bad demo to us!)
- If there was no DEMO0.B?? (or any of the DEMO?.B??) and waiting in the menu started a random demo, the SpeedFix would crash! This also happened if any standard demo was forced to start by clicking "Demo" in the menu etc. This bug has been removed. This was the most annoying bug: Crash after wait!
- If such a random demo (DEMO?.B??) contained BUG-sprites, those demo's could not finish: the stored start position of such demo was not fetched, which caused the bugs to react different each time the demo was started randomly. This has been fixed.
- The Ctrl-F12 key now also stops a demo PLAYBACK, so you can take over. I just thought it was a logical and handy thing to do.
- Screen "glitches" caused by writing menu text to the play field have been removed. There were some, especially in Debug mode using the white - and = keys to change levels during the game. Stuff needed to be updated, but in those update routines also menu text was written to screen. This is fixed. If there were any other such occasions, those cannot happen anymore.
- All sprites beyond the invisible wall are now invisible too. I changed this because due to a Supaplex bug (which we don't want to fix), Infotrons and Zonks can change into fixed objects that just won't fall. These objects cannot be reproduced if a saved game is loaded again (if the complete video memory is not saved too). Those objects became strange looking random blobs and that is why I made those invisible. If someone objects to this, I can easily change that back. Just try and see the difference between this and an older version SpeedFix especially with the debug "s" key (remove all Snik Snaks). (The invisible wall was not designed into Supaplex! It accidentally happened to be there without the programmer knowing about it.)
- I recently found out that there are notebooks with LCD displays that do not use the official VGA frame frequency 70Hz, but 60Hz instead. There goes another standard. (Thanks Compaq!) Nothing fixed here: that's life.

v6.0 -> v6.1 (Herman Perk)

- Oops, forgot about Maarten's checksum. I fixed the demo recording BUG-bug, but in cases that that fix had to do something (sometimes), the checksum was wrong. The demo was ok, but Maarten rejected it for his Hall Of Fame. Actually, those rejected demo's proved, that my BUG-bug fix works!
  Now also the checksum will be ok, and recording must be always perfect.
- The savegame option (Ctrl-W and Ctrl-L) can now be safely used during a recording: the recording is ended properly now when Ctrl-L is pressed, and also recording "scars" in the saved game are ignored now with Ctrl-L.
- After loading a saved game, now the random number is randomized, which means that each time the game is loaded, the bug-sprites fire differently. In 6.0 those bugs always reacted in the same manner.
- At last!: There was an original Supaplex problem with the Red Disk release. If Murphy was killed DURING a Red Disk release, he could NEVER release another Red Disk, not even in other levels, until Supaplex was completely restarted. This has been fixed.
- After loading a saved game, the actual level could not be skipped. Fixed.
- Changed our text in the author screen a little: extra line of text.

v6.1 -> v6.2 (Herman Perk)

While everyone was playing with version 6.1, I spend some time to improve the SpeedFix a lot. Here is the list of changes/improvements:

- Command line parameter collisions (spfix61 /n !m :fname first loaded the game without demo, and after you exited that game it played the demo of it. When that demo finished, the /n !m did their job).
  Now the demo is played first, then the /n !m start that level and the level pointer in the menu points to that level afterward.
  A major change was needed in Maarten's SP file control routines, where several minor bugs (which were "corrected" in his code) were removed too. This change was tricky, but now the code is much simpler.
- If command line had just : or @: strange things happened: A filename was recognized, but rejected after changing a byte in the SPFIX data space ...
- The level pointer was reset to level 1 by starting a game with F12. Fixed.
- I changed the name of the saved game from "SUPAPLEX.SAV" to "SAVEGAME.S??" which prevents possible wildcards problems with DOS file manipulations. Each level set has its own unique saved game now, but the old way of having only one saved game for all level sets can be forced with the new command line option "W", to always use "SAVEGAME.SAV" in stead of "SAVEGAME.S??". (If you have an old "SUPAPLEX.SAV" that you still want to use, you can rename it to any of the 101 different possible "SAVEGAME.S??" at will.)
- The saved game does not use the saved debug mode information anymore, so if you saved in normal mode, and are in debug mode when loading the saved game, you now stay in debug mode (vice versa). This prevents surprises.
- The panel on/off status after loading a saved game was bad: The panel was (in)visible as before but the viewing window was taken from the saved game. That panel on/off info from the saved game is now ignored.
- If a saved game was loaded during a first-time run of an ":SP_file", the player name appeared as "WIBBLE??". Fixed to show the initial name.
- If a saved game was loaded during a demo run, the last demo input "key" was not deleted, which sometimes moved Murphy from his saved location. Fixed.
- Changing from debug mode to normal mode didn't issue Q and 1 which entered the normal mode without resetting debug options M, D, 2,...,9,0. Fixed. The Q and 1 commands are now also issued when a game exits, and after loading a saved game.
- Debug S-command did not remove Snik Snaks properly. Fixed.
- Explosions in the top corners of the game field edge destroyed internal variables. (This happened with experimental levels only.) Fixed.
- If Murphy went into the top left 8 edge fields of the game border, he was beamed away to unknown places. This could only happen in levels where Murphy was allowed to walk on edges. Fixed because it was a bug anyway.
- Maarten removed by accident the "restore video mode" (SPFIX54) when the game was aborted due to an error. (This was already fixed in SPFIX60.) He corrected for it by staying in the game after a demo-file load error. The behavior was unpredictable: Restored to exit after such error.
- In debug mode an explosion cloud showed as red signal lamps and Sergei Sinicyn from Russia showed me that this was because of a mouse button bug.

This original bug was also responsible for the fact that Murphy was killed by the Enter key if no mouse driver was loaded. The mouse button reactions (during the game-play only!) are now exchanged and Maarten's bug fix of that "kill Murphy by Enter" is removed because it is not needed anymore. This exchange of button reactions is no problem since their reactions during the game are annoying and were of no value anyway.
Now also the right mouse button is the ESC key in the game too, not only in the menu, so actually I think it is better this way. (More consequent,)

- The "abort Supaplex on error" exit did not restore the int 24h vector and didn't switch off running music: A DOS boot could have been needed. Fixed.
- This "abort Supaplex on error" exit could have passed errorlevel 0 to DOS. In case of 0 it passes an errorlevel 255 to DOS now. (For batch "freaks".)
- SP-demo files longer than 50698 bytes are now rejected as legal demo's. This number includes 1536 bytes for the level, 48650 bytes for the maximum demo length (with the terminating 0FFh) and 512 bytes for the signature. This filters out many problems when using wildcards with the now enhanced command line option "@" to test demo's at warp speed (see below).
- The debug M command now reacts on the gray 6-key block as a "Goto". Their relative positions on the keyboard indicate which field part to show:
  Ins,Home,PgUp = Go to the top of the level: left,middle,right
  Del,End ,PgDn = Go to the bottom of the level: left,middle,right
- Loading a saved game now blacks the screen and fades in. This became necessary because it loads the "dull" game field, updates the panel, restores "fancy" stuff, which all would flash around the screen for a few moments. I think it looks better this way, and it gives an extra short pause during the fade-in, when immediate reaction to the saved game is needed, so you can rearrange your fingers on the keyboard in anticipation of the game.
  I made the screen dark without fading, so it doesn't take extra time.
- Supaplex had disabled code to 'shake the screen during an explosion', which is now enabled. Because it sometimes was too much shaking now, I enabled this feature (by default) only if Murphy dies.
  For changing this default, two new command line parameters have been introduced: "S" or "s" if the screen should Shake on every explosion, and "N" or "n" if the screen may Never shake, as in the original game.
  Too bad that this shaking still makes problems on the Matrox Millennium. (Sergei Sinicyn showed me a way to optimize code here, but because of my many command line switches, I could not use much of his optimized code.)
- Moving to the left when the debug M-command was active, the game field jumped when the left edge was reached. Now you cannot go beyond the field edges anymore so the outside of the game field cannot be viewed.
  To change this default setting, another command line parameter "M" or "m" has been introduced to be able to look beyond the field edges again, but this time without the described jump.
- The debug M key was affected by Murphy's demo moves during a demo run. This was not a logical M key reaction. Now it does not follow Murphy.
- The Alt-SysReq key now exits faster and more reliable.
  This makes some sense of this key combination and more attractive to use.
- Maarten's result text of the "@" command line option is now redirectable to a file, which allows unattended demo file checking with file-output to be evaluated later after all demo's have been tested.
  In order to use this feature, it was necessary to change the original idea a little: If there is no demo attached or no (or bad) ":file" parameter used, the program now exits immediately with an appropriate message.
  Only demo's can play now if the "@" command line option is used.
  The message now also includes the file name from the ":" parameter, even when a (bad) demo file is shorter than a level or larger than 65535 bytes.
  For even extra speed, fading has been disabled and the exit is done now without removing the panel first. See below for more details.
- LEVEL.L?? (LEVEL.LST) is now optional and not mandatory. This means that it is not needed anymore. SPFIX62 tries LEVEL.L?? first in case some extra speed is needed (if the game runs from floppy disk), and, if it does not exist, the information from LEVELS.D?? is used.
  A new command line parameter "C" or "c" will tell SPFIX62 to create LEVELS.L?? in case it does not exist.

(If it is not needed, it is not necessary to create it by default, and like this Supaplex can still be played from write-protected disks.)
Yes, I noticed the difference between the original LEVELS.DAT and the new created LEVELS.DAT: The original has a dash in the level name of level 042 between "LITTLE" and "PLEASURE". LEVELS.DAT has a space in that name.

- If the 4 byte long SUPAPLEX.CFG had as first byte anything else than "i", "s", "a", "b", "r" or "c", Supaplex did not load any sound driver as it should, and looking at the 'controls' screen it showed as if "s" was active, and going back to the menu without changing anything, "b" was put here in this CFG file. This could lead to a crash if this choice was bad. Now the code has been simplified to test only non-default values, otherwise a default is forced for any non-default byte. This fixed this 'bug'. The file is now also closed after a read error, to free the handle.
- Clicking on the "floppy 1<->2" in the menu was a useless thing to do. I changed this into changing level sets! All existing LEVELS.D?? are presented one by one in the message line, in ascending order. Doing the same with the shift key pressed, they show in descending order.
- Clicking this "floppy 1<->2" with the Alt key pressed, the old but improved floppy stuff is activated: It now also accepts drive letters up to F (without showing this in the screen text). After ESC-exit, the default drive is restored now as it was before pushing the disk button in the menu. If a drive was chosen where no Supaplex files were found, it now does not abort the whole program anymore, but still there has to be any formatted medium in the drive to be able to escape, but at least not to DOS anymore. In case someone does not want to hold the Alt key, I introduced the command line parameter 'F', which only inverts the Alt-key influence here.
- It has been possible to change "normalized" Hardware and RAM-chips back to the original "fancy" stuff, so saved games do not look dull anymore. (The size of a saved game is increased with the 1536 bytes for that.) Furthermore, all debug keys where something is removed, now leave that "fancy" stuff intact too, and prevent the level to successfully end with an update of the level pointer (and the time doesn't count either).
- The now left mouse button used to turn "fancy" stuff to "dull" stuff. It now toggles between "fancy" and "dull". The "normalize" or "change to dull" routine affected the (initial) movements of Snik Snaks and Electrons, because it was meant to be called at game-start only. This has been fixed.
- The auto-repeat of the Enter key for switching the panel on and off has been drastically slowed down, and it now immediately reacts after each new Enter key press.
- The auto-repeat of the debug keys F1,F2 and F3 was too fast: you never knew when gravity was on or off etc. This has been changed to key-actions without auto-repeat. The F2 key has also been changed to on-off-on-off.
- I finally managed to do something about that screen "glitch" that appeared when the game successfully ended (or when demo's were aborted), just before panning to the menu. This "glitch" was only observed when Murphy was on the right side of the game field, and was caused by overwriting the viewing window. The game field is used by panning to the menu etc. as follows: The viewing window is copied to the top right of the game field, (even extending beyond the normal game field, which accounts for the trash which is visible with the debug M key), then the screen jumps to that new window, then the menu is copied to the middle of the top of the game field, and finally the panning begins. See the section "Map of the video memory". Fix: If the viewing window is completely outside the far left viewing area, (otherwise we would introduce the same problem over there again,) a preliminary copy to the top left and jump to that copy is done first now. From there the rest is as it was: The original copy is done etc. (Sergei Sinicyn also did good work on this "glitch", but his solution did not work at all on the Matrox Millenium VGA card! Sorry Sergei!)
- At the start of each game, Murphy changes very little. The pictures of the not-moving Murphy in FIXED.DAT and MOVING.DAT are slightly different: The game field is initially painted with FIXED.DAT, and as soon as the game runs, only Murphy's picture from MOVING.DAT is used. Now the field is initialized too with Murphy's picture from MOVING.DAT.
- The menu messages "demo failed", "game successful" etc. are now updated before the menu appears, which means that you don't have to wait until

the panning or fading has been completed to see those messages.
(The SpeedFix version info on top in the author screen now appears faster too, but I left the names on the bottom as they were, because before those names appear, no key to proceed to the menu is accepted by Supaplex.)
- Short original demo's were opened, but never closed after rejection: The error "Too many open files" followed. Fixed, but also:
- The original old-style demo's are recognized now and can be used again: The NEWDEMOS.ZIP is now obsolete, so the original Supaplex can be used too without problems. (NEWDEMOS.ZIP cannot be used with the original game!) Also SPFIX62 :DEMOn.BIN works. Any original DEMO?.BIN will show BIN in stead of .SP in the panel. Any mix of new and old demo's can be used, and the new demo's will always show .SP in the panel, as before.
The original DEMO?.BIN files are recognized by their length and the first 4 bytes. (Knowing this, you can fool the SpeedFix, but that's life.)
The old demo's need the original LEVELS.DAT file (with the original name!) to play, even if the demo's are renamed and thus activated for a different level set (like DEMO?.B01). This way you can have the original demo's running with all different level sets by copying the demo's.
If the order of the demo files is changed by renaming (like DEMO6.BIN becomes DEMO2.BIN etc.) the demo's still work (as in the original game).
- Scrolling of game field was "poor" if not running at highest speed. This has been changed now! I finally found the way to fix it!
If not set at highest speed or half speed, it is still a bumpy ride, but smoother now. Only highest and exact half speed seem to be perfect now.
Murphy himself does not move that smooth, but that is also because we now need 16 different pictures of him moving, and they are just not available.
The soft scrolling is done pixel-wise, and Murphy (and all other moving or falling objects) moves with 2 pixels each time: The result is that he moves two pixels but the screen just 1 pixel, and the next frame another pixel, which (net) causes Murphy to blur. I cannot do anything about it!
- A bug has been fixed with horizontal scrolling: some VGA cards like the Matrox Millennium were affected by this bug. Just in case this bug fix now makes problems with VGA cards that used to run, I introduced the command line parameter 'H' (for Horizontal) to force the original Supaplex timing.
- Added an extra SpeedFix key: instant fastest speed with the gray "divide"-key, which is also the "/" (slash) key on the white US(!) keyboard.
- I made the SpeedFix speed auto-detect routine faster, which measures the original playing speed when the very first game or demo is played.
- The code has been optimized in several places and I am pretty sure that I did not introduce new bugs, but one can never know for sure ...
Please e-mail me if you find any bugs, or even if you think you found one!
- Besides changing this document to show the enhancements, several new parts have been added: The layout of several Supaplex and Speedfix files, a complete list of original protection codes, the video memory layout, and instructions for using the edges in levels properly.

v6.2 -> v6.3 (Herman Perk)
- If a demo was started from the menu and Ctrl-F12 was pressed during that demo, the current level was marked 'done' if the end was reached. Fixed.
(Bug reported by Yonatan Rozenshein from Israel)
- Loading saved games also "restored" saved interrupt vectors, which are there to release the interrupts when the SpeedFix returns to DOS ...
If those SAV files were saved on a different computer or configuration, the computer will crash while returning to DOS. This has been fixed by skipping those saved vectors. The SAV file format has not been changed.
(By the way: I can not explain the SAV file format without describing a lot of internal Supaplex variables, so I will not spill any time for it.)
- Restarting a level sometimes did not show Murphy's movements correctly in the beginning: "Multiple" Murphies could appear. This was again a bug due to a variable that was not re-initialized if a level started: Much the same as the Red Disk bug before. Fixed. (Vague bug reports by several.)
This happened when Murphy was killed inside a port, where he was drawn on screen in two parts. Restarting the level, this "split" was still active, but only in two initial directions: going up or left into a space sprite.
I also found out why only up and left etc. and fixed the problem there too.
- The joystick was always scanned, even if the keyboard was chosen in the

controls screen. Users had to disconnect their joysticks to play: Fixed. (Frans Meulenbroeks and Yonatan Rozenshein among many bug-reporters.)
- The speed-escape Alt-SysReq has been changed to Alt-X, because Windows interfered with this Alt-SysReq key-sequence. (For Yonatan Rozenshein.)
- Demo's were affected by the debug speed keys 1-9 and 0. Now these keys are disabled during a demo. (For Min-Soo.)
- Starting a demo recording would not automatically reset the debug speed-keys to 1, and did not automatically issue a "q" to exit any debug d and m. Now they do. (Also for Min-Soo.)
- The Hall-Of-Fame entry did not include the time of the finishing level. Now the Hall-Of-Fame entry equals the real total time. New total times are now accurate. (Bug reported by Richard Sharland from South Africa.)
- Once BIN displayed left in the panel, to show what kind of file is playing, anything else, like level number or .SP, did never appear anymore, with one exception: if an .SP demo was started with the F11 key, everything was returned to normal, until BIN reappeared again. This bug was because of a not properly handled variable in new version 6.2 code, to handle the 10 original old-style demo's. Fixed. (Reported by Kim Min-Soo, Korea.)
- If the debug R-key was pressed during a demo, the level of that demo was restarted, but the demo continued to play, without restarting. Fixed. Now a demo "rewinds" too with the debug R key. (By and for Kim Min-Soo.)
- If the debug R-key, '='-key or '-'-key were pressed, the accumulated player time was not updated to show how much time the user spent in a level. Now these keys update the player time (under normal conditions, of course).
- If the debug '='-key or '-'-key were pressed during a demo or an SP file, the current level pointer changed. Fixed. No more hidden level changes.
- There were more problems with the debug R-, '='- and '-'-keys: After a running demo was taken over with Ctrl-F12, old-style demo's failed to restart, new style demo's and the F11 demo would, but there were strange things going on. Also if you wanted to record a demo from that demo then. I hope I made everything working consequently for all those situations.
- If a Red Disk was released, and before its explosion, the debug R key was pressed to restart the level, that explosion still happened in the restarted level. This was still a remains of the original Red Disk bug, of which yet another variable was not re-initialized. Fixed. (Kim Min-Soo.)
- If a Red Disk was released, and before the explosion of it, the game-situation was saved with the Ctrl-W key, that Red Disk did not explode when that "savegame" was loaded with the Ctrl-L. I erroneously initialized in version 6.2 the same variable that originally caused the Red Disk bug. Fixed. In the loaded game the Disk now explodes, but with a small visual "error": That temporary Red Disk does not show on screen.
- If a Red Disk was released, it could be swallowed again and still explode. This is a cheat, which is considered forbidden. Now this Red Disk cannot be eaten anymore, but since this changes the game itself somehow, I made this cheat accessible again with a new command line switch T or t. (This cheat was reported by Tom Geelen, a long time ago. A fix at last.)
- New command line parameter added: T, which enables the now by default disabled Red Disk cheat trick for "backward compatibility".
- New recorded demo's are now named differently: iiSjjj$k.SP for standard recordings, and iiS---$k.SP for recordings from SP files, where ii stands for the active level set, jjj for the active level number and k for the known demo number as in the original DEMOk.B?? (=F-key minus 1).
  For the original level set, this xx becomes '00', as known from Maarten's solutions, and because the real '00' is a different legal set (reserved for instance for the working set in SupaShow in compatible mode), this original '00' becomes '--'. {I am aware that this is neither logical nor consequent, but it was Maarten, who decided to name the original demo's '00S???-?.SP', not me ... Well, this scheme can also be interpreted as a confirmation of my reservation of the real set '00' for scratch purposes.}
  Note that the file names differ from the 'official' solutions only by the $ (dollar sign), opposed to Maarten's '-' (minus sign), and can be addressed all-at-once by the wildcard structure "??S???$?.SP" for copying or renaming or anything else anyone want to do with all his/her demo's.
  Note also that loaded SP files (see command line option ":filename.ext") can also result in a recorded demo, but there can be 10*101 different

demo names generated: 10 for the Ctrl-Function keys, and the 101 for all possible active level sets when the SP was started with the F12 key: The active level set (which can be changed in the menu by clicking on the disk 1<->2 symbol at the bottom) is here also included in the name. (Idea for this enhancement by Frans Meulenbroeks.)
- New command line parameter added: O (the letter, not the digit zero), which forces the old demo names for recording demo's. Because the new demo names prevent instant replay with an Fn-key of just recorded demo's , I made this original way of recording still possible.
- Automatic addition of a prepared signature, BUT ONLY if the recording succeeded and the level was successful (Exit reached with the Infotrons). The signature has to be a special text file with the name: MYSPSIG.TXT SPFIX63 uses the text of that file until the first terminator byte (FF hex) or until the end if no terminator is available, or until the 511th byte if the file is longer than 511 bytes. Then it adds any missing terminator. This legalizes ANY text according to the definition of the SP signature. This new feature also works if the old demo names are forced with the 'O'. If you don't want it, just delete MYSPSIG.TXT from the Supaplex directory.
- A sample of MYSPSIG.TXT is included in the SPFIX63 package, which MUST be replaced by your own signature, or MUST be deleted if this automatic signature addition is not wanted. Use Maarten's SPSIG to strip unwanted signatures from demo's.
- I have included also a small program (TESTSIG.EXE) to check how your text will look like, when the SpeedFix appends it: If the text is too long, you can see what is left of it as a signature and where it is clipped. TESTSIG has no parameters and it always uses MYSPSIG.TXT. Just type 'testsig' and you'll see what the SpeedFix does with MYSPSIG.TXT. If you want the signature separately stored into a file, you can redirect the screen output to a file, like in: TESTSIG > MYSPSIG.SIG and then use it to append manually to a demo, like in: COPY /B MYDEMO.SP + MYSPSIG.SIG
- "Remarks about Supaplex game field calculations and resulting tricks" and "SpeedFix command line parameter overview" added to this doc.

v6.3 -> v6.4 (Herman Perk) NOT RELEASED
- Make it possible to instant replay the new SPFIX63+ demo's with Alt-Fn.
- Moving to the right, eating a Red Disk, the animated video sequence is wrong: It uses 9 pictures in stead of 8 as with all other sequences. Because existing demo's wouldn't run properly if the extra frame would be removed, the extra frame is untouched but without scrolling now. Because Kim Min-Soo used this as feature in one of his levels, this fix has been removed again.
  -> Not enough changes to justify the release of 6.4, especially after the release of Megaplex, which caused v6.3 to be the last official release!

* Ideas for future enhancements (JUST IDEAS!):
  - Fix the EGA panel problem if running with VGA cards. The EGA emulation on VGA cards needs some extra programming for this split-screen function to work properly.
    (I did some tests already but until now without any success.)
  - Level verification at load time: the first 1440 bytes may only contain hex values from 00h to 28h, and there are also some easy criteria for several other bytes. A bad level would not be loaded. This would make it possible to use the "@" parameter with 'wilder' wildcards and search for level and demo files without ".SP" extension, without having to wait for and abort illegal files (*.com, *.exe, *.bat).
  - Use the video frame frequency in calculating the demo times, for honesty.
  - Who knows what I can think of next.

* Known but unfixed problems:
  - There still seems to be a problem with the PLAYER.LST file: it sometimes is destroyed. Reason yet unknown but this was also in the original game. It did not happen to me for a long time. Not repeatable: hard to find. Maybe this bug has been fixed already? Please notify me when it happens!
  - There are sometimes some more screen "glitches" during the game, which is an original Supaplex "bug".
    (Especially during unforeseen disk-accesses like when SmartDrive writes

or reads something to/from disk without Supaplex asking for it.)
- Some problems with Matrox video cards still exist: moving to the left during the first few seconds and shaking in the right half of the game field still makes problems! (The left half shakes to the right, while the right half shakes to the left because otherwise we would see trash from outside the game field on the edges.
  Vertical shaking makes no problem: all black outside.)
- Murphy blurs during the soft-scroll movements. This cannot be changed because of missing pictures, since Murphy moves 2 pixels in MOVING.DAT. It only may be improved a little by calculating Murphy to move per pixel, but his background would blur then, and it takes a lot more programming!
- Several of the sound generating sub programs (*.SND) do not work properly anymore on our fast computers. Especially the SoundBlaster effects from BLASTER.SND use delay times from CPU-instructions, which is too fast.
- There is always one more bug.

Herman.

==========

HOW TO DEFINE SUPAPLEX SPEED, AND WHAT IS THE CORRECT SPEED

I took my 286/10 for the Supaplex speed reference. I use the standard demo which is started from the menu with the F2 key, and check the panel time at the end of that demo. (The panel clock always seems to run correctly.)
My speed reference is 2 minutes and 9 (+/-1) seconds for that demo.
On my Pentium 133 this time was exactly half of this: 1 minute 4(5) seconds.
Until now I do not have a definitive answer from Robin Heydon how fast Supaplex is supposed to run, except that the slower speed "feels right".

Does anyone object? Please correct me if I take the wrong speed reference!

Several selected statements from Robin Heydon to me:
...
>>Well the game should run at 60Hz. But on a PC it would run at 70Hz, due to the video graphics hardware. So it would run quicker than the original Amiga Version. Or though, not so quick that it makes it unplayable.<<
...
>>I have tried your version on this machine and it does 2min 10. BUT I have not tried an original version. I don't have one, and no way of getting one.<<
(Note by Herman Perk: this is about original packaged Supaplex, not the ZIP!)
>>However, it does "Feel" right to me, and that's what is important in all games. Anyway, go and release it. It should be excellent. I've certainly created a little interest on this side of the Atlantic.<<

SUPAPLEX SPEED-FIX: INTRODUCTION

On modern machines the original SUPAPLEX.EXE runs too fast.
This let me search for a fixed version. There wasn't any around, so I had to fix it myself, so my then 5 year old son could play Supaplex on my Pentium.
There were several problems to overcome (see HISTORY DETAILS below).

The speed fixed file SPFIXnn.EXE replaces the original SUPAPLEX.EXE.
The other necessary game files (the whole game) can be found in the Internet.
Any feedback is welcome, see my (E-mail) address below.

All my SpeedFix versions are completely based on the original Supaplex code, and all of them can be used as complete substitute of the original EXE file, solving the speed problem on fast machines, but also running on 'slow'

machines in the original speed.
I do not intend to unnecessarily change any of the original code.
I did however remove the protection routine after I saw what it actually did, and some code that was not even referenced at all: dead stuff, small leftovers of early tests by Robin Heydon.
I have made some additions to make life easier for several persons.
I let Maarten Egmond make some additions for the same reasons.

Have fun and spread the game,
Herman Perk. <- made in Holland and always will be Dutch.

SUPAPLEX SPEED-FIX: INSIDE INFO AND HISTORY

INSIDE INFORMATION ABOUT WHY THE GAME RUNS TOO FAST:

The game speed depends on the video frame frequency (vertical scan rate) in the 300*200 mode. Supaplex calculates changes in the game field, it updates the screen and it waits for a new frame to start.
On slow machines those varying calculations last beyond the, say, first synchronization pulse, and Supaplex was created to react that way.
On fast machines however, those calculations are ready before that same pulse shows up, and waiting for a pulse now is waiting for the wrong pulse: one pulse too early. The game speed is then exactly twice the original speed, depending on when those calculations are ready: before, close to, or after that first pulse. I expect on very few computers that the speed is neither the original speed, nor the exact double speed. Those computers must be matching the video sync pulses very closely.

HISTORY DETAILS:

- The original EXE file is packed (=blocks of the same byte are squeezed).
  My disassembler could not unpack this EXE file, so I had to do it myself.
  I created an unpacked EXE file first, and I used many tricks to do so.
  I then created a disassembly, which did cost me some time too, before the reassembled code ran even after shifting code around. I did this shifting for testing the quality of the disassembly. By this time my goals were not to just try and patch the original EXE file, but to reassemble the whole after changing the disassembly, since patching a packed EXE file can be very tricky!

  Then the biggest problem arose: where in the world did I have to add delay!
  The problem was, that whatever I tried, everything looked fine until the screen scrolled horizontally. It double shifted during a scroll.
  So where was that scroll routine? I found 3 of those, but not the one I was looking for: the horizontal screen movement when Murphy walked.

  I expected that I had to go through a lot of code, and cried for help in the Internet: no reply. At last I found THE routine, which does actually not scroll at all: it just changes screen pointers even if Murphy does not move. (I rejected this place as a candidate before, since I was searching for a scroll only routine and here my breakpoint in the debugger went bezerk.)

  The rest was done in less than half an hour, and several hours for testing.
  I released this version as my first SpeedFix through Maarten (Elmer).

  -

- During my search, I found strange things. Keys that were tested, but were disabled by a zero byte. This byte cannot be set by Supaplex itself.
  I added a method to set it and played with the extra keys.
  This second version I E-mailed to Maarten and I warned him for bugs.

- I already found out, that the Ctrl-function keys should enable the demo recording routines, but only on a few occasions I was able to create a working demo file, and I had many crashing 'too many opened files' errors. There they were: over 3 megabytes of lost clusters, all looked like the beginning and completed demo files. Now I knew why it was not recording properly, and after a quick look it was clear what the problem was. As long as those keys are pressed, it reopened the file each time the keys are found to be pressed. With a high auto repeat rate of Supaplex itself, and the wrong timing of when the message 'RECORDING DEMO0' appears on screen, it cannot be used reliably. I fixed that bug by closing the file first if it is opened, and then open the file again.
- By this time the 'RECORDING DEMO0' message never appeared on screen, so this was fixed too by just moving one instruction.
- Now it always showed DEMO0 for all demo's. I changed it by adding one instruction, and this SUPAPLEX.EXE was released as SpeedFix 3.

- -

- During my experiments with new levels, and Maarten's SPEDIT, I was annoyed by the fact that I needed to rename files every time, and together with Maarten I developed a few command line options for easy level access. With the command line option ! and two digits you can choose now from 101 level sets. See below for details.
- I had a big problem if files LEVELS.D* and LEVEL.L* of the requested group did not exist: The computer would hang in a loop of "INSERT DISK 2" and "INSERT DISK 1" screens, and the only escape was the reset button. I fixed this problem by adding code for aborting these screens with the ESC key.
- I added display of the SpeedFix info and the video frequency in the statistics screen if the debug mode is active.
- I changed the debug keys '-' and '=' to show the proper level names now, so in debug mode you can change levels properly in the game itself.
- Spedit shells to Supaplex with Ctrl-A. If the Ctrl key is held until big Murphy starts to show up, the release of that Ctrl key is never seen by DOS: this information is swallowed by Supaplex.
  This goes for all Ctrl, Alt, Shift, NumLock, CapsLock etc. keys.
  DOS reacts upon return as if that Ctrl-key is still pressed, until that same Ctrl key is hit again.
  It always is very annoying when the keyboard does not react normal.
  I fixed this by waiting for all such keys to be released just before Supaplex takes complete control of the keyboard. This is an endless loop without time-out, so if anyone has problems, please contact me!
- I know why Robin's slow-down code has problems with the demo's: he wanted Supaplex to react quickly upon key-press in his slowest mode.
  This means that he has to look at the keyboard more often than under normal conditions. This keyboard routine also takes the next demo step...
  I did not 'debug' this.
- The result is released as SpeedFix 4.

- -

- Maarten Egmond contacted me about making some additions to the SpeedFix, in order to play his new demo format *.SPD files. After discussing this, we came to the conclusion, that there was no need to create a new SPD format (where the level would have been glued after the demo information), but just extending the existing SP specifications would do even better.

  This is, where I thought, that there was no need for SpeedFix additions, and I demonstrated Maarten with a special (unreleased) version of my SupaShow, that it was possible to play the level and show the demo of a SP file using the SPFIX4. All the high level language program had to do, is creating a set of LEVELS.D00, LEVEL.L00 and DEMO0.B00.
  LEVELS.D00 would have been nothing more than 111 copies of the level part of the SP file. DEMO0.B00 would have been the demo part of the SP file, and LEVEL.L00 would have been very easy to create from the level part.

This set 00 was either to be reserved as special set, or backed up before, and restored after starting the special set, created from the SP file.

Since Maarten did not wanted to go that way, and since he offered to make the SpeedFix changes himself, and since I don't claim to be any creator of the game, and since I think that Maarten was very well capable of doing the job, I let Maarten struggle with my SPFIX4.ASM disassembly file. (Another reason: I had no time for that right away.)

INSTUCTIONS FOR USE, AND MORE

Starting with SpeedFix v4, the speed fixed SUPAPLEX.EXE executable is called SPFIXnn.EXE, where nn is the current version number, in stead of SUPAPLEX.EXE. (Now the fixed file can be recognized from the outside too.)
Just copy the SPFIXnn.EXE into your existing Supaplex directory and type SPFIXnn in stead of SUPAPLEX to start the game.
(I leave it up to you to copy SPFIXnn.EXE over the old SUPAPLEX.EXE, or not.)

You need all the other files from the original game too, which can be found on the Supaplex WebPages (see top). Just download the big ZIP file called SUPAPLEX.ZIP (if you don't have it yet).

Before version 6.2: Make sure you backup the original files DEMO?.BIN, and replace them by the new demo's which are in the NEWDEMOS.ZIP file included in this package, or the demo's will not work properly. This is obsolete now.

If you use SPEDIT 3.0 or higher, remember to change SPEDIT.CFG to shell to SPFIXnn in stead of SUPAPLEX.EXE.

If you use SupaShow, remember to change SUPASHOW.INI to shell to the correct SPFIXnn version (or to SUPAPLEX.EXE if you want it the hard way).

Now the main thing to get it working is just to change to your Supaplex directory, and run SPFIXnn.EXE. You should see the normal Supaplex intro screen coming up, and then the code screen. No code will be asked. In stead, the current SpeedFix version number will be shown. Press any key here to start the game. The rest works just like in the original game.

- -

Advanced users (or if you want to do more with the game):

If the initial screen with names appear, including the text "SPEED FIX (n) BY HERMAN PERK" (or similar), and Supaplex is waiting for a key, press the ScrollLock key to enable the debug features.
For the standard game: press any other key.

Up to version 6.0 this and a command line parameter were the only places where this debug mode could be switched on.

Version 6.0 introduced an extra key to switch the debug mode from within the game: Ctrl-ScrollLock = on, Alt-ScrollLock = off.

During the game there are extra debug keys to play with, but be aware that this is not without bugs. It was not meant for us to use! Use at own risk!

SPEEDFIX COMMAND LINE PARAMETERS

(The original game Supaplex does NOT interpret any command line parameters.
I introduced the use of the first few of these parameters in SpeedFix 4.)

All command line parameters can be used together in any combination and order.
Some parameters exclude each other mutually like the # and the & options.
I did not want to make an intelligent command line parser, so please excuse
the way these options are implemented.

Note that "SPFIXnn" should be replaced by the actual executable of this
version (e.g. SPFIX4, or SPFIX63). This nn is simply an abbreviation for
the version number.

First a command line example to show what it is all about:

```
SPFIX63 !01 # /3 d
```

Which means:
Start SpeedFix with level files *.?01, force cheat mode (do not use the files
PLAYER.L01 and HALLFAME.L01 and enable all levels), start level 3 immediately
and in debug mode (so demo files DEMO0.B01,...,DEMO9.B01 can be created, etc.)
Spaces between parameters are not needed (except after a file name).

Description of all defined parameters:

```
SPFIXnn *z   or   SPFIXnn *
==========        =========
```

where z=0,1,2,...,10 is a SpeedFix delay number as in: SPFIXnn *5

If the SpeedFix does not find the speed properly, you can force any speed
within the SpeedFix range. *0 (or just *) is the fastest Supaplex speed.
With *10 Supaplex runs the slowest, *5 is for computers that run exactly with
double original speed (most computers I know). This option fixes the speed
at the start of Supaplex, in stead of automatically find the right speed and
then fix it. Even if the * option is used, the SpeedFix keys are always
active. See also the statistics screen in debug mode for checking this value.

```
SPFIXnn !y
==========
```

where y is 00..99 is a level set number as in: SPFIXnn !23

The first 10 sets can be addressed with a single digit: !5 equals !05 etc.
Ever since new levels can be created, there is a need for handling those
levels. All files belonging to the original set of 111 levels are:
LEVELS.DAT, LEVEL.LST, PLAYER.LST, HALLFAME.LST, DEMO0.BIN, ..., DEMO9.BIN.
With this option 100 total new sets can be used by Supaplex, without renaming
the original files.
The file names of those new sets are different in the extensions only:
LEVELS.D00, LEVEL.L00, PLAYER.L00, HALLFAME.L00, DEMO0.B00, ..., DEMO9.B00,
etc. up to LEVELS.D99, LEVEL.L99, ..., DEMO9.B99.
This allows 100 extra sets of 111 levels each, plus the original set making
a total of 101 times 111 levels.
A complete new set can be renamed or copied with following wildcard file name:
*.?00 for the complete set 00, and *.?57 for complete set 57 etc.
Remember that PLAYER.L00,...,PLAYER.L99 and HALLFAME.L00,...,HALLFAME.L99 are
all hidden, like PLAYER.LST and HALLFAME.LST.
If option !y is not used, the original set is used.
Only LEVELS.D?? and LEVEL.L?? must exist for a set to work.

PLAYER.L?? and HALLFAME.L?? are created without names and privileges, if those files do not exist.

SPFIXnn /x
==========
where x=1,2,3,...,111 is a level number as in: SPFIXnn /78

Maarten wrote SPEDIT, a level editor. He already implemented a call to Supaplex in order to test the level under construction.
With this option SPEDIT can instruct Supaplex to enter a level without going through the menu. If '/' is seen on the command line, only the first picture of Supaplex is displayed (the opening curtain and author screen are skipped. If no legal level number is entered immediately after the '/' then the nearest level to this illegal number starts. (200 will start 111, 0 will start 1)

SPFIXnn &w
==========
where w=1,2,3,...,20 is a player number as in: SPFIXnn &2

This starts Supaplex with the player of choice already active, so this player does not have to be selected by pressing the player up and down buttons in the menu. His ranking is also selected.
Did you already know, that clicking on the player's name shows his ranking, so you do not have to scroll through the player AND ranking?

SPFIXnn #
=========
Force all levels playable as if those are all skipped. This was necessary for SPEDIT too. With this option the player list and the hall of fame list are never affected: Supaplex does not change the files PLAYER.LST and HALLFAME.LST
The player name is '(FORCED)' and cannot be changed or deleted.
This is a good test mode.

SPFIXnn S
=========
New in SPFIX62: The original Supaplex has (deactivated) code to shake the screen during all explosions. I thought this was too much, so now the default is that the screen shakes only when Murphy explodes. This 'S' or 's' lets the screen shake on all explosions. See also the 'N' for no shake at all.

SPFIXnn N
=========
New in SPFIX62: The original Supaplex has (deactivated) code to shake the screen during all explosions. I thought this was too much, so now the default is that the screen shakes only when Murphy explodes. This 'N' or 'n' disables also that shaking (like the old Supaplex). See also the 'S' for always shake.

SPFIXnn D
=========
Force debug mode. This 'D' can be 'd' too, so you don't need the shift key.
Since the '/x' option skips the screen where you can press the ScrollLock key for 'debug mode on', I implemented this option as command line parameter too.
Debug mode will enable you to record demo's (to show to your friends, or as help to explain someone how to solve a level), and use some more features.
See "COMPLETE LIST OF SUPAPLEX KEYS" below for what extra keys (features) you can use in debug mode.
SPFIX60 and above have an extra possibility to switch between normal and debug mode: see Ctrl-ScrollLock and Alt-ScrollLock in the complete key list below.

SPFIXnn M
=========
New in SPFIX62: The debug m key for moving the game field allowed the field to scroll beyond the edges. This is now clipped by default, and this 'M' or 'm' can be used to get the old non-clipping m-mode back, but now improved: The problem with the sudden jump on the left edge has been fixed.
See for this debug m key under "EXTRA DEBUG KEYS IN THE GAME ..." below.
(Check also the extra SPFIX62-keys Ins,Home,PgUp - Del,End,PgDn below!)

SPFIXnn C
=========
New in SPFIX62: The SpeedFix doesn't need LEVEL.L?? anymore! If that file is available, it is used. If it is missing, the file LEVELS.D?? is used and the missing information is extracted from it. Nothing is written by default, but this 'C' or 'c' tells the SpeedFix to create the LEVEL.L?? file in that case. If LEVEL.L?? is present, nothing will happen, so if you want to make a new file, DELETE the old file first before starting the SpeedFix with this 'C'. The missing file is then created, when the SpeedFix investigates that level set to display level names etc. in the menu. To re-create all new LEVEL.L?? files: delete all LEVEL.L?? before starting the SpeedFix, and scroll through the level sets in the menu by keeping the Floppy 1<->2 symbol near the bottom of the menu screen pressed, until all level set names were displayed in the message line. That's all. (If you don't use the "#" option, all missing hidden PLAYER.L?? and HALLFAME.L?? are initialized at the same time.)

SPFIXnn L
=========
New in SPFIX62: Start the SpeedFix with the saved game (if available).
Be aware that SPFIX62+ introduces different saved games for each level set!
(See also the 'W' and '!y' options.)

SPFIXnn W
=========
New in SPFIX62: Use only SAVEGAME.SAV as saved game, otherwise SAVEGAME.S?? is used, where ?? is either AV,00,..,99 depending on which level set is active when the snapshot is made (with the "write game" key "W" during the game), AND when it is loaded again!

SPFIXnn F
=========
New in SPFIX62: Clicking with the mouse on the floppy symbol "1<->2" in the menu now changes the level set (default): All available LEVELS.D?? are "scrolling" through the message line, the original first, then 00,01,..,99, wrapping back to the original. With the Shift key held down, this order is inverted. If the Alt key is pressed while clicking on the floppy symbol, the original (but enhanced) floppy routines are executed, as in the old days ... This 'F' or 'f' parameter just inverts the function of the Alt key here.

SPFIXnn E (Before version 6.2: SPFIXnn EGA)
=========
Since the VGA/EGA mode is tested, set and used by Supaplex before the keyboard routines are installed, I could only show the EGA mode through a command line parameter without affecting the style of the program too much.
This EGA mode shows Supaplex using the standard 16 colors only, and not changing the video palettes for better colors. Fading is not possible.
The Supaplex screen in this mode is calculated with a different length than with VGA. There is a problem with the location of the bottom panel in the

game, because this EGA mode is not meant to run on VGA, without precautions.
This option is only to show what unused potential is in this game, you probably don't want to use it a second time after seeing it...
(I changed the EGA into just E, because I did not see any reason to leave it the way it was, and I could remove my old code to check for this whole EGA.)

SPFIXnn H
=========
A bug has been fixed with Horizontal movements, so I think that now more new video cards are working properly, like the Matrox Millennium which I used to fix this bug. Especially VGA cards where shifted pictures (with a horizontal difference of half a sprite) were flashing during those horizontal movements.
This 'H' or 'h' forces the original Supaplex timing, just in case the bug-fix makes problems now with originally working VGA cards (which I do not expect!).
If you need to use this parameter PLEASE tell me about it!!!

SPFIXnn R
=========
I provided a command line option "R" or "r" in version 6.2 to refresh the video memory when the game returns to the menu. I did not make this refresh option default because it increases the time to return to the menu, especially when the game runs from floppy disk: the file MOVING.DAT is reloaded. This parameter is actually only needed when you are experimenting with walking on and beyond the top or bottom edges, which is not allowed anyway. For more about such experiments see the section "ABOUT THE GAME FIELD EDGES" below.

SPFIXnn O
=========
I provided a command line option "O" or "o" in version 6.3 to be able to use the original demo names when recording demo's. With the new demo names for recording, instant replay with the appropriate function key in the menu is not possible anymore. Maybe in the next SpeedFix version I might implement a way to instant replay of the recorded demo's with new names.

SPFIXnn T
=========
This command line option "T" or "t" in version 6.3 enables an old Supaplex bug, which I fixed: Eat a just released Red Disk, which will explode anyway, but you carry one extra Red Disk. Repeating this trick, you will never run out of Red Disks. This trick is officially declared to be an unallowed cheat.
Without this command line parameter, this cheat is not possible anymore.
Because here I changed the game itself a little by disabling this cheat, I created this option to still use this cheat, or at least demonstrate it.

SPFIXnn :filename.ext
=====================
where filename.ext is the name of a single-level-file: SPFIXnn :MYLEVEL.SP
or, on a different drive in a subdirectory: SPFIXnn :D:\HERE\IS\MYLEVEL.SP

Maarten wrote SPEDIT, a Supaplex level editor.
SPEDIT can create single-level-files, which normally have the extension SP.
With SPEDIT version 3.2, Maarten introduced an extra option for these so called SP-files: a demo (created with SPFIX3 or higher) can be attached to the level, and still it is an SP-file (called: extended SP file).
This option starts the SP file demo (if a demo is attached), or starts the level itself (if no demo is attached).
The Supaplex menu shows the current level set (chosen with the !-option), which is indeed active. All other command line options can be used too.
You will not find the SP level in the Supaplex menu.

Back in the menu, you can restart the demo with the F11-key, or (re-)start the SP-level with the F12-key.
With the SpeedFix v5.0 and upwards, the demo's that Supaplex records, are also extended SP files (i.e. you can record a demo, say DEMO3.BIN, and run it with SPFIXnn :DEMO3.BIN).

SPFIXnn @ :filename.ext

The "@" can only be used together with the colon option (see above).
The file used with the colon option must contain a demo. This option plays the demo at "lightning" speed, without any delays. On a fast 486, a demo that takes 5 minutes at default speed, will now be played in only a few seconds. Further, the program will exit immediately after playing, and state whether the demo failed or succeeded (to reach the exit with enough Infotrons collected). This is handy to check demo's fast.
(Maarten uses it to check entries to the Honored Solutions list very quickly)

The resulting screen text afterward is redirectable to a file (version 6.2+). (Which means that without it, every single file needs to be checked manually, because the result text could then not be put in a file for evaluation later.) Speedfix version 6.2+ allows no levels without demo's to load with the "@" option anymore, because it would prevent complete unattended bulk demo checks with the use of batch files: The ESC key would have to be pressed to exit. (Only the original 10 old-style demo files are recognized and evaluated, and all other old-style demo's are rejected as "file too short" by SpeedFix 6.2+.)

Such a DOS batch file could look like in this two-line example:

```
echo SPFIX63 demo results: > spresult.txt
for %%a in (*.sp demo?.b??) do call spfix63 @ :%%a >> spresult.txt
```

(If this second line is typed in as command on the DOS command line, both "%%a" must change into "%a": Each with only 1 percent sign.)
(NOTE: Under Windows it is necessary not to use long file names at all!)
The first line kills any existing old result file with the single ">".
The line with 'for' can be changed if the SP files are located in subdirectories. Example for files in subdir1, subdir2 and more:

```
for %%a in (subdir1\*.sp subdir2\*.sp d:\even\more\subdirs\*.sp) do ...
```

A more intelligent structure is of course possible, and also a possibility of aborting the batch can be built in ...

An example of such a resulting file (in our case spresult.txt) with all different expected results could be:

```
SPFIX63 demo results:
Demo successful: 03S087.SP
Demo failed:     NEWDEMO1.SP
SP without demo: A_LEVEL.SP                  (meaning: level only)
!! File < Level: OLDDEMO.SP                  (meaning: file too short)
!! File >> Demo: LEVELS.SP                   (meaning: file too long)
Demo successful: DEMO0.BIN                   (original short demo!)
```

3 other possible error messages are from unexpected errors, where level results in stead of demo results appear, which could only occur if there is a bug in my code, or from an error in the command line, which would not be not fault:

```
"@"-ERROR: Level(?) successful: UNKNOWN.SP
"@"-ERROR: Level(?) failed:     ERRORS.SP
"@"-ERROR: Bad or missing ":filename.ext"!
```

```
              +-----------------------------------------+
              | SPEEDFIX COMMAND LINE PARAMETER OVERVIEW |
              +-----------------------------------------+

     !nn        Force level SET number at start (nn=0...99), else original set
     /nnn       Force LEVEL number at start    (nnn=1...111)
     &nn        Force PLAYER number at start    (nn=1...20)
     *nn        Force SpeedFix SPEED at start   (nn=0...10, or empty(=0=fast))
      #         Force all levels skipped and no score updates: test mode
     A a        (not used, but reserved by "EGA" mode)
     B b        (not used yet)
     C c        If deleted, Create LEVEL.L?? file out of info from LEVELS.D??
     D d        Force Debug mode at start: needed to record demo's etc.
     E e        Force (buggy) EGA mode on VGA hardware
     F f        Force original Floppy 1<->2 symbol function (Invert Alt key)
     G g        (not used, but reserved by "EGA" mode)
     H h        Force original Supaplex Horizontal smooth-scroll timing
     I i        (not used yet)
     J j        (not used yet)
     K k        (not used yet)
     L l        Load and play the available saved game at start
     M m        View beyond the game field edges with the debug M key options
     N n        Never shake the screen during explosions (See also "S")
     O o        Record using Original demo names DEMO?.B?? (not ??S???$?.SP)
     P p        (not used yet)
     Q q        (not used yet)
     R r        Refresh video memory after each game: reload MOVING.DAT
     S s        Shake the screen during every explosion (See also "N")
     T t        Allow the use of the original infinite Red Disk (ch)eat Trick
     U u        (not used yet)
     V v        (not used yet)
     W w        Force Writing only one SAVEGAME.SAV (else use SAVEGAME.S??)
     X x        (not used yet)
     Y y        (not used yet)
     Z z        (not used yet)
 :filename.ext  Use SP-file to play at start and from menu with F11 and F12
@:filename.ext  Lightning speed SP-demo test, exits to DOS with message
```

COMPLETE LIST OF SUPAPLEX KEYS

Following is a COMPLETE list of keys, that are checked by Supaplex.
Since Supaplex tests the keyboard scan codes only, it is impossible to refer to national keyboard layouts. All keys are as located on the US keyboard.
BE AWARE that all keys can have a HIGH AUTO REPEAT RATE on fast machines, except for the 'p' and the NumLock pause keys (and all keys used in the protection code, which I threw away.) This time I did something about the debug function keys F1, F2 and F3. They really could do without fast repeat.
The mouse can only be used in the menu and controls screens. See also below.
Watch out: the mouse buttons are active in the game, and can destroy your effort!

ONLY IN THE PROTECTION CODE ENTRY (I THREW IT AWAY):

0,1,2,..,9 = (not on the keypad!) Digits to enter, so the (unpatched) game did not always say: DOS ERROR: Access Denied.

BackSpace = Correction key.

Enter = Usual Enter, and hope that the entered three digits were correct, unless you have a patched game.

THE 'DEBUG-ON' KEY I ADDED (AUTHOR SCREEN ONLY):

ScrollLock = Switch on the debug mode. This ONLY works in the author screen! This key is not available in my first SpeedFix release.

THE STANDARD KEY SET IN THE MENU:

Several game keys have a visual effect in the Controls screen too.

ESC = Leave Supaplex: back to DOS
Leave Controls, Gfx Tutor, Statistics etc. when in those screens.
The right mouse button in the menu: Leave Supaplex: back to DOS

Alt-SysReq = Back to DOS. As if ESC key is pressed in the game AND the menu.
From SPFIX63: THIS KEY COMBINATION HAS BEEN CHANGED TO Alt-X !!!

Alt-X = NEW IN SPFIX63! Used to be Alt-SysReq, but because Windows uses that combination too for its own purpose, I changed it to Alt-X.
What it does? See Alt-SysReq above.

a,b,..,z = Entering a new player name: needed for the name. (US keyboard!)

- = Entering a new player name: dash can be used in the name. (US!)
(This is not the keypad '-'!)

Space = Press the OK button, even if the cursor is somewhere else.
Leave Gfx Tutor, Statistics etc. when in those screens.
Entering a new player name: Space can be used in the name.

BackSpace = Entering a new player name: Correction key.

Enter = Press button under cursor, but only if no mouse driver is loaded.
Leave Gfx Tutor, Statistics etc. when in those screens.
The left mouse button in the menu: Press button under cursor.

Entering a new player name: Enter (as usual).

Arrow keys = If no mouse driver is installed: move cursor in that direction. The keypad digits 8 (up), 4 (left), 6 (right) and 2 (down) are functional too. This is not influenced by the state of NumLock. The mouse takes over, but only in the menu and the controls screen: nowhere else, although both mouse buttons function as Enter/ESC/Space key in several other screens, like Gfx Tutor, Controls, Statistics and other screens with that blue background.

FunctionKey= F1,..,F10 Start DEMO: its 'file number' is the function key number -1. Those demo files are DEMO0.BIN through DEMO9.BIN for the original level set, and DEMO0.B00..DEMO0.B99 through DEMO9.B00..DEMO9.B99 for the additional level sets, when such a level set is chosen. If any of those files are missing or really bad, then none of the following demo files with higher numbers are ever loaded by Supaplex, and therefore cannot be started.
In that case just fill the gap by copying any good demo file to the missing/bad file.
As with SpeedFix 6.3 the names of the demo's, which the player records himself, cannot be started with these keys anymore, unless the command line option 'O' was used when the demo's are recorded, or when those new demo's are renamed to the original names (DEMO0.BIN etc.), or by loading those SP files with the command line option ":filename.ext".
SPFIX63 and later: Changed to Alt-FunctionKey, unless command line option 'O' has been used.

left mouse button = in the menu: Press button under cursor.

right mouse button = in the menu: Leave Supaplex: back to DOS

```
+--------------------------------------+
| EXTRA MENU KEYS MAARTEN EGMOND ADDED: |
+--------------------------------------+
```

F11 = restart the demo, attached to the single-level-file, specified on the command line (colon option "SPFIXnn :file.sp"). This key only works in the main menu screen.

F12 = (re-)start the level of the single-level-file, specified on the command line. This key only works in the main menu screen.

```
+--------------------------+
| EXTRA MENU KEYS I ADDED: |
+--------------------------+
```

Shift and clicking on the floppy 1<->2 symbol: change level sets in descending order. Without Shift the order is ascending.
Changing level sets from the menu is introduced in SPFIX62.

Alt and clicking on the floppy 1<->2 symbol: do the old floppy stuff:
= load the LEVELS.DAT etc. files from a different floppy.
This used to be the default without Alt key before SPFIX62, but without improvements like drive letters up to F.
See also the command line option 'F'.

```
                    +----------------------------------+
                    | THE STANDARD KEY SET IN THE GAME: |
                    +----------------------------------+

ESC         = Quit play (give up).
              The left mouse button functions as ESC key during the game:
              do not touch the mouse and don't drop it while playing!
              This is changed in SPFIX62: it is now the RIGHT mouse button,
              which is now the same ESC mouse button as in the menu!

Alt-SysReq = Back to DOS.  As if ESC key is pressed in the game AND the menu.
              Any blue text screen has to be closed separately though.
              From SPFIX62: these blue screens are closed too, and the exit to
              DOS is faster and more reliable.
              From SPFIX63: THIS KEY COMBINATION HAS BEEN CHANGED TO Alt-X !!!

Alt-X       = NEW IN SPFIX63! Used to be Alt-SysReq, but because Windows uses
              that combination too for its own purpose, I changed it to Alt-X.
              What it does? See Alt-SysReq above.

Space       = Pressed together with arrow keys: do not move, but 'eat' the
              object in the chosen direction.
              Pressed alone for a few seconds: release a Red Disk.

Enter       = Toggle bottom panel on/off. (Display more of game field.)
              Before SPFIX54: If no mouse driver was loaded: KILL MURPHY!

Arrow keys = Move Murphy into the chosen direction.
              The keypad digits 8 (up), 4 (left), 6 (right) and 2 (down) are
              functional too.  This is not influenced by the state of NumLock.
              The mouse is without function in the game, except for the left
              button, which acts as ESC.
              In debug mode, after pressing the 'm': move game field that way.

j           = Recalibrate joystick on the fly.

p           = Pause on/off.  This is independent from the NumLock pause.

NumLock     = Pause on/off.  During this pause you can type in "cant sto".
              Type in those 4 characters, 1 space and 3 characters only:
              the pause AND the debug mode both end, without pressing NumLock.
              Remember that these keys refer to the US keyboard layout.

Pause       = (This key is not checked by Supaplex or any SpeedFix.
              DOS may still make its own pause with this key ...)

RIGHT Shift= Show in the bottom panel how many Red Disks Murphy has left.

left mouse button  = Before SPFIX62:  Kill Murphy!  From SPFIX62: no function.
                     Do not touch the mouse and don't drop it while playing!

right mouse button = Before SPFIX62: no function.  From SPFIX62:  Kill Murphy!
                     Do not touch the mouse and don't drop it while playing!
                     Now it is the same button as in the menu.

                   +----------------------------------------+
                   | THE SPEED-FIX KEYS I ADDED (GAME ONLY): |
                   +----------------------------------------+
(These 4 keys do their job everywhere in Supaplex, but the result is only seen
in the game.)
```

gray + = Speed up stepwise to the original speed. (see also gray '-')

gray - = Slow down stepwise, independent from the 1..9,0 keys: see below.
The debug 'd' command key will show a black-white flickering.
This means that the CPU does active waiting in the black periods.
Slowing down with these SpeedFix keys does NOT affect the
demo's. (The debug speed keys 0..9 do affect the demo's!)

gray * (x) = Re-adjust speed automatically. This is default at the start of
Supaplex, unless the command line option *... is used (Ver.4+).
Version 5.x and higher only adjust once, and remember the setting
for this * key. This is why after each fresh start of the Speed-
Fix, the very first moment of the first game is slow: Measuring.

gray / (:) = Fastest playing speed. This key is the same as the white slash
key ('/') on the (US) keyboard.
(This key is not available in SpeedFixes before version 6.2).

SpeedFix version 4 and up display additional information in the statistics screen if the debug mode is on. The normally dashed line shows delay info of the SpeedFix keys and the video frame frequency.
Example of this line: -------- <DLY:05> ------- <071HZ> --------
DLY:00 refers to the original Supaplex speed (fastest). DLY:10 is slowest. DLY:05 is the speed for computers that originally run exactly double speed. The video frame frequency information in this example shows 71 Hz, or 71 frames per second. This frequency is measured and displayed each time the statistics screen is called: you will notice a 1 second measuring delay.

```
+-----------------------------------+
| KEYS I ADDED IN 6.0 (GAME ONLY): |
+-----------------------------------+
```

Ctrl-W = Write game situation: make a snapshot which can be loaded with
Ctrl-L later. This makes it possible to proceed with a partly
solved level, beginning with the situation at the time the
snapshot was written.
The snapshot file name is fixed to SAVEGAME.S??, where the "??"
depends on the active level set. Any previous snapshot with the
same name is always overwritten and thus gone forever!
The lower right corner of the panel will flash "WR" if there
were no problems, otherwise it will flash "XX".
Ctrl-W does not affect anything: If you finish a level or a demo
making snapshots with Ctrl-W, that level counted the normal way,
and demo recording/playback is not aborted: The game situation
is written as if no recording or playback was running.
(Before SpeedFix version 6.2 there was only one saved game for
all level sets possible and was named "SUPAPLEX.SAV". These old
files can be renamed to any of the 101 different "SAVEGAME.S??".)
See also the command line options "W" and "L" above.

Ctrl-L = Load the snapshot file, written with Ctrl-W.
The lower right corner of the panel will flash "LD" if there
were no problems, otherwise it will flash "XX".
"File does not yet exist" is an example of such a problem.
It is not possible to complete a level by using this "cheat"
method: If you want to officially finish a level, you have to
play it from scratch, without using this snapshot Ctrl-L option!
See also the command line options "W" and "L" above.

Ctrl-ScrollLock = Enable Debug mode during the game.
The lower right corner of the panel will flash "DB".

Alt-ScrollLock = Disable Debug mode during the game. See Debug keys below.

Ctrl-F12 = Stop demo playback: you take over. You can finish that demo.
(This is the same key that also stops demo recording: see below.)

+---+
| EXTRA DEBUG KEYS IN THE GAME (US KEYBOARD LAYOUT!): |
+---+

Ctrl FunctionKey = (Re-)Start recording DEMOx (x=Function key number-1).
Remember that the demo files count from 0 and the function keys from 1. Therefore the -1 here.
You can't change the speed of the game when you are recording a demo. Only F1 through F10 can be used.

WARNING: THE ORIGINAL CODE LEAVES LOST CLUSTERS ON THE HARD DISK: do not wait pressing these keys until the message RECORDING DEMO0 appears since by then it has reopened the same file without closing it. This doesn't destroy data, but you fill your hard drive with trash if you do not remove those lost clusters. The resulting DEMO file is actually inside one of those lost clusters.

I fixed this bug in SpeedFix version 3.
You can now restart the demo recording as much as you want, since the previous file is now closed first.
Furthermore the RECORDING DEMOx message is put on screen immediately now, and not with the second open as in the original code, and it now shows the proper file name.

Maarten Egmond changed the format of the created demo's to include the level itself. (SpeedFix ver.5 and up)
See also remark at FunctionKey= above.

NEW in version 6.3: The names of the resulting files are only DEMOx.B?? when the command like option 'O' is used. With no 'O' parameter, new file names are used to store these recorded demo's: xxSyyy$z.SP, where xx is the active level number 00-99 (the original level set becomes '00' here, and because 00 is a different set, those original '00' demo's are renamed to '--'). And yyy is the active level number ('---' if the level was started with the F12 key), and z is the function key number minus 1, as in the original demo names.
The demo's with new names cannot be replayed instantly: They can be loaded and viewed with the command line option ":filename.ext".

Also NEW in version 6.3: If the file "MYSPSIG.TXT" exists in the Supaplex directory, the first 511 bytes (or all of the text when less than 511 bytes) in this file are used to append it after a successful(!) demo. SpeedFix automatically adds a terminator byte (FF hex). If Murphy failed to complete the level, no signature is appended. Use Maarten's SPSIG.EXE in this case if a signature is really wanted (like when showing a trick). (Using my SupaShow, bad or good demo's are easily found by looking whether a signature is appended or not.)

1,2,3,4,5,6,7,8,9,0 = (not on the keypad!) slow down: 1=normal 0=slowest.
This however does not slow down the timing of the DEMOs: The demo runs in normal speed, but the screen reacts in low speed. Poor Murphy ...
Use the 1 to get back to normal speed, and choose speed with the SpeedFix keys for the demo's to run properly in lower speed than normal.
From Version 5.3: During demo recording this is forced

to 1 and ignored until end of recording.
From Version 6.3: During demo playback this is forced to 1 and ignored until end of playback.

NumLock = see NumLock above for ending the debug mode with 'cant sto'.

Keypad 5 = Return to Murphy's position in the screen move mode (see 'm' key).

r = Restart the active level, without going through the menu.
From SpeedFix 6.3: This key also rewinds running demo's to the start.

m = Move the game field, not Murphy. q = Quit this function.
The arrow keys move the field into the chosen direction. The Keypad 5 returns the screen to Murphy's position, but the mode stays until 'q'.
This command has been improved in SPFIX62:
- The screen jumped on the left border: fixed, but now only the game field can be seen by default. Command line parameter "M" or "m" allows also to look outside of the game field (as before SPFIX62), but without jump.
- 6 extra keys for immediately show 6 different parts of the game field. See those new keys below: Ins, Home, PgUp, Del, End, PgDn.
- Using this during a demo playback, the screen movement was affected by Murphy's movements. Not anymore! (This was just annoying.) Keeping keypad 5 pressed, Murphy is followed without leaving the m mode.

From Ver 5.3: This key is ignored during demo recording.
From Ver 6.3: This key is reset with q when demo recording starts.

d = White background. q = quit (My guess: this shows how much time the CPU needs for its Supaplex calculations. Darker parts show calculation time.)
From Ver 5.3: This key is ignored during demo recording.
From Ver 6.3: This key is reset with q when demo recording starts.

q = Quit from m and d modes.

s = Remove all Snik Snaks
c = Remove all RAM-Chips
z = Remove all Zonks
b = Remove all Bases
h = Remove all Hardware

When something moves, and is removed, the screen will show strange objects. If hardware is removed, the field edges -especially the bottom edge- do very strange things with the Supaplex objects. I did not introduce these bugs! (see batch below) SPFIX62+ now completely removes the Snik Snaks and keeps the "Fancy" hardware and RAM chips!

- = Go to screen of previous level (not the keypad '-'. See also '='-key)

= = Go to screen of next level (-,= before SpeedFix 4: no level name update.)
('+' is the shifted '=' key on the US keyboard, so we deal with + and -)

F1= Toggle gravity on/off
F2= Prevent Zonks from falling: off/on/off. (Press twice for on, once for off)
From SPFIX62: Just toggle this on/off: no pressing twice anymore.
F3= Toggle 'freeze Snik Snaks and Electrons' on/off
(These 3 keys reflect all properties that special ports can change. Those special ports were thought to only affect gravity, until September 1997)

right mouse button = From SPFIX62: Kill Murphy! Before SPFIX62: "normalize" Hardware and RAM chips: "Fancy" stuff changes to dull stuff. Actually this happens with almost all debug keys which change things. Supaplex just repaints the screen from memory, which is already "normalized" for fast game reactions, so it does not always have to go through all different Hardware and RAM-chip stuff.

left mouse button = Before SPFIX62: Kill Murphy! From SPFIX62: Toggle normalized Hardware and RAM chips to fancy stuff and back. The debug keys which change things now do not normalize fancy stuff anymore!

```
+--------------------------------------+
| EXTRA DEBUG KEY MAARTEN EGMOND ADDED: |
+--------------------------------------+
```

Ctrl-F12 = Stop demo recording. This will close the demo file that is being recorded. The game will continue as if nothing happened. This key will only work when a game is being recorded. If you press this key in the main menu it will be regarded as a normal F12. (This is the same key that also stops demo playback: see above)

```
+--------------------------+
| EXTRA DEBUG KEYS I ADDED: |
+--------------------------+
```

Alt-ScrollLock = Switch Debug mode off during the game.
The lower right corner of the panel will flash "--".
If any of the "d" or "m" keys were used, an automatic "q" is invoked. (If you don't want to leave those d and m modes, you can also press NumLock followed by the "cant sto".)

Ins Home PgUp
Del End PgDn = During the "Move the game field" mode (see m key) it seemed convenient to me to show a specific part of the game field with an easy procedure. The arrangement of these six keys on the keyboard is the same as the corresponding part of the game field. Home is on top in the middle, and so on. As long as you don't use a notebook, you don't have to remember which key shows which part of the game field. Those 6 field parts just overlap (if the panel is switched off): The whole game field can be inspected extremely fast.

```
+--------------------------------+
| ABOUT SPEEDFIX DELAY FACTORS: |
+--------------------------------+
```

The original Supaplex runs at "maximum" speed on each machine. At the time Supaplex was programmed, it ran at only one speed on all machines (except on those very slow machines where calculation time was not acceptable).
The SpeedFix allows 11 different speeds, of which one is the only original Supaplex speed from the old days, and one is the (original) maximum speed of today's original Supaplex.

Let us number the 11 different speeds from 0 to 10, where 0 is maximum speed, and let us call those numbers "SpeedFix values".
These numbers are somehow related to the amount of extra delays that the SpeedFix inserts in-between the many separate Supaplex video frames:

As also written somewhere above, Supaplex takes some time for its own internal calculations, which used to last over 1/70th of a second for each new frame, but on present day machines this takes less than 1/70th of a second.

After those calculations are ready, Supaplex just sits and waits for the first occurrence of a video frame synchronization pulse, which come each 1/70th of a second.

Let us now forget about the calculations and just notice that in the old days it took 2 delays (of 1/70th of a second) for each new Supaplex frame, and now it takes 1 delay (of 1/70th of a second) for each new Supaplex frame on fast machines. (Original Supaplex!), which makes the game exactly twice as fast.
In other words: Originally it took 1/35th of a second per Supaplex frame, and now, with fast machines, it takes 1/70th of a second per frame.

The SpeedFix introduces extra (discrete) delays: it waits for extra video-

frame synchronization pulses (the 70 Hz pulses), and the amount of extra delays is (for each of the SpeedFix values 0..10):

0, 1/5, 1/4, 1/3, 1/2, 1, 2, 3, 4, 5, 6

(The fractional amounts of delays are created by inserting one delay every so many calculation cycles, and taking the mean delay over a large enough time interval.)

The multiplication factor for the playing time can be calculated from these extra(!) delays. The original Supaplex standard playing time from the old days, multiplied with such a factor equals the true playing time now.

For slow machines these can only be (depending on the SpeedFix value 0..10):

1, 11/10, 9/8, 7/6, 5/4, 3/2, 4/2, 5/2, 6/2, 7/2, 8/2

Time multiplication factors for fast machines (for "normalized" playing time):

1/2, 6/10, 5/8, 4/6, 3/4, 2/2, 3/2, 4/2, 5/2, 6/2, 7/2

Example calculation for slow machines for SpeedFix value 7:

Supaplex waits on slow machines for 2 delays per frame of its own.
We put in an extra 3 (from SpeedFix value 7) and we get 5 delays in total.
This 5 divided by the 2 of the original game from the old days makes 5/2,
which means that Supaplex waits 5/2 times the original time, which means
also that the playing time is prolonged to 5/2 times the original time,
which is calculated by counting frames of 1/35th of a second each.

The same calculation example for the present fast machines:

Supaplex waits on fast machines for 1 delay per frame of its own.
We put in an extra 3 (from SpeedFix value 7) and we get 4 delays in total.
This 4 divided by the 2 of the original game from the old days makes 4/2,
which means that Supaplex waits 4/2 times the original time, which means
also that the playing time is prolonged to 4/2 times the original time,
which is also calculated by counting frames of 1/35th of a second each.

Note: The "@" command line option tells the SpeedFix never to wait for any of those video synchronization pulses. Not even those waits from the original Supaplex are executed: The speed depends on calculations only.

The original VGA standard "frame rate" at the Supaplex screen resolution of 320 * 200 dots (16 colors) is 70Hz (= 70 frames per second).
In some notebooks this 70Hz is reduced to 60Hz, which makes things even more complicated ... The SpeedFix does not correct for this non-standard rate.

What is worse: Present day turbo video cards may have this low resolution badly implemented, which even can make it impossible to play Supaplex.
(Neglected because everyone cries for higher speeds and higher resolutions...)
Example: Several Matrox video cards, ...

How does the automatic SpeedFix speed-adaptation work?
==========

The SpeedFix uses the system clock (which Supaplex had speeded up to 50 Hz) to measure time intervals, and adapts the number of calculation cycles per time interval to the standardized expected amount: 7 cycles per 1/5th of a second. If the SpeedFix value ends up as 5, which means that exactly one extra delay was needed, we deal with a fast computer. If the SpeedFix value ends up as 0, which means that no extra delays are allowed, we have a slow computer at hand. I forced nearby SpeedFix values to 0 or 5: 1 becomes 0, and 4 or 6 become 5. In-between values have also popped up, at the video frame rate of 60 Hz ...

This machine dependent SpeedFix value is stored somehow in each recorded demo,

together with the SpeedFix value of the slowest recording speed. From these two values, together with the information in the demo itself, we calculate the total playing time of the demo. I know, that slowing down for a short moment during a demo recording, and then speeding up again, results in unrealistic and "unfair" long recording times, but that's life: Just don't slow down...

```
+-----------------------------------------------------------------+
| HOW TO SWITCH ON THE DEBUG MODE WITH THE ORIGINAL SUPAPLEX CODE: |
+-----------------------------------------------------------------+
```

The ORIGINAL Supaplex version can be switched into the debug mode with the following batch job (which needs the DOS program DEBUG.COM or DEBUG.EXE). (All "cracked" original Supaplex versions are considered "original" too.) This batch can be extracted from this file by deleting everything except the following 10 lines, and save it as SUP.BAT into the Supaplex directory:

```
@echo off
echo g33 > supaplex.deb
echo t >> supaplex.deb
echo g103 >> supaplex.deb
echo t >> supaplex.deb
echo ecs:a040 1 >> supaplex.deb
echo g >> supaplex.deb
echo q >> supaplex.deb
debug supaplex.exe < supaplex.deb
del supaplex.deb
```

NOTE: This batch is NOT to be used with any of my SpeedFix versions!
SEE ALSO MY WARNING ABOUT THE 'RECORDING DEMO' BUG! (Ctrl-FunctionKeys)

I use this batch job to compare the behavior of my recompiled version with the original code. I did never see any differences yet except for the improvements I made.

If the text "CODE: xyz" shows, there are three more ASCII characters on screen, just behind the text "ENTER: ???". These are, if converted to digits, the code to enter: Space=0, !=1, "=2, #=3, $=4, %=5, &=6, '=7, (=8,)=9. If the protection has been removed by a patch, this can be ignored.

```
+-------------------------------------------------------+
| HOW TO FORCE THE EGA MODE WITH THE ORIGINAL SUPAPLEX CODE: |
+-------------------------------------------------------+
```

With one additional line, the above batch job can also force the video mode on VGA cards to run as if it runs with an EGA card, and with debug mode on. The behavior and the colors of Supaplex are totally different now. Different code is used in the program, and the standard 16 colors are used in stead of using the VGA palettes for nicer colors.

```
@echo off
echo g33 > supaplex.deb
echo t >> supaplex.deb
echo g103 >> supaplex.deb
echo t >> supaplex.deb
echo ecs:5378 0 >> supaplex.deb
echo ecs:a040 1 >> supaplex.deb
echo g >> supaplex.deb
echo q >> supaplex.deb
debug supaplex.exe < supaplex.deb
del supaplex.deb
```

NOTE: This batch is NOT to be used with any of my SpeedFix versions!
I included a command line switch to force it on in my SpeedFix (v4+):
use the command line parameter EGA (capitalized in v4 only) as in:
SPFIXnn EGA

On VGA there is a quirk in location of the panel.
I did not test it with an original EGA card, so I do not know if it is a bug.
(VGA cards need a different approach when using EGA code, which may be why.)

The EGA mode is switched on with the line: echo ecs:5378 0 >> supaplex.deb
The debug mode is switched on with the line: echo ecs:a040 1 >> supaplex.deb
The lines with g33,t,g103,t are to persuade Supaplex to unpack itself.
Supaplex has to be unpacked, since the debug byte is inside of a packed block.

+--+
| EXTRA INFO ABOUT THE ORIGINAL SUPAPLEX CODE (FOR INSIDERS): |
+--+

I found two different protection patches of the original Supaplex game:

1- If any number can be entered, and the game starts anyway, the described 3 debug-mode ASCII characters can be ignored, but they are visible anyway. This was a single or double byte patch, which both ignore the result of the entered code, but still runs all of the protection code:

The most common patch is the single byte patch (also my original):

```
edit: SUPAPLEX.EXE
search: 3B 0E 34 DE 75 06 C6 06
change: -- -- -- -- -- 00 -- --      (-- = leave as it is)
```

(This patch changes a conditional JUMP to jump to the next instruction, and therefore this jump is disabled: it does nothing.)

I found the following text for the two byte patch in the Internet:

(Quote:)
```
edit: SUPAPLEX.EXE
search: 3B 0E 34 DE 75 06 C6 06
change: -- -- -- -- 90 90 -- --      (-- = leave as it is)
```

(This patch changes the same conditional JUMP into NOP's, which do nothing)

2- If the screen with author names is immediately followed by the menu, without waiting for a code, then Supaplex has been patched differently, and all of the remarks about 3 debug-mode ASCII characters can be ignored, since those will never appear on screen.

This was a 3 byte patch, which bypassed the complete protection:

I found this text somewhere else in the Internet too:

(Quote:)
```
Crack for SUPAPLEX by Satch.
File: SUPAPLEX.EXE
Search for  : E8 E9 05 E8 98 87
Replace with: -- -- -- 90 90 90      (-- = leave as it is)
```

(This patch changes the CALL to the protection routine into NOP's.)

Maarten wrote in his FAQ on his Supaplex pages, that there is another patch, which waits for just a single "any key" to proceed to the menu.
I have never seen any other reference to that patch.

(It wouldn't start on my computer, because I don't seem to have an "any" key.)

These texts were my verification for the reconstruction of the original protected code, that I made before I knew these texts. Why I wanted to reconstruct the protected code? I wanted an original disassembly, and sometimes such patches change the behavior of the program somehow.

(In 1998 this reconstruction has been verified: The SUPAPLEX.EXE from the official Supaplex version on CD is byte for byte identical! Only the installation program has changed and a TSR program has been added to run Supaplex from CD. There is also a manual in PDF format, but it uses a special keycaps font (PIXymbols.command) which is not included on CD! The printed result of that manual looks a bit weird with the font-substitute that the many different Adobe Acrobat Readers on that CD choose instead.)

- -

Together with the two byte patch text I found following text too: (Quote:)

> Edit the savefile with a HexEditor (it's a hidden file!) and edit this after your name (as entered when you first created a character in the game) enter a few 01 hex characters , ascii code 1 a small face (white) and this counts as one more completed level. Do this up till about 111 or as far as you like !

The referred savefile is PLAYER.LST, and everyone with a little bit of experience already did this automatically without ever reading this text.

There is also a utility on Maarten's pages to make these changes to this file.

- -

COMPLETE LIST OF PROTECTION CODES:

These codes are only needed to start the original unpatched, protected game.
(Don't think that I typed this code table ... I'm not that kind of crazy!)

000 864	063 303	126 766	189 205	252 668	315 107	378 570	441 009
001 465	064 928	127 367	190 830	253 269	316 732	379 171	442 634
002 066	065 529	128 992	191 431	254 894	317 333	380 796	443 235
003 691	066 130	129 593	192 032	255 495	318 958	381 397	444 860
004 292	067 755	130 194	193 657	256 096	319 559	382 511	445 461
005 917	068 356	131 819	194 258	257 721	320 160	383 623	446 062
006 518	069 981	132 420	195 883	258 322	321 785	384 224	447 687
007 119	070 582	133 021	196 484	259 947	322 386	385 849	448 288
008 744	071 183	134 646	197 085	260 548	323 505	386 450	449 913
009 345	072 808	135 247	198 710	261 149	324 612	387 051	450 514
010 970	073 409	136 872	199 311	262 774	325 213	388 676	451 115
011 571	074 010	137 473	200 936	263 375	326 838	389 277	452 740
012 172	075 635	138 074	201 537	264 500	327 439	390 902	453 341
013 797	076 236	139 699	202 138	265 601	328 040	391 503	454 966
014 398	077 861	140 300	203 763	266 202	329 665	392 104	455 567
015 511	078 462	141 925	204 364	267 827	330 266	393 729	456 168
016 624	079 063	142 526	205 989	268 428	331 891	394 330	457 793
017 225	080 688	143 127	206 590	269 029	332 492	395 955	458 394
018 850	081 289	144 752	207 191	270 654	333 093	396 556	459 509
019 451	082 914	145 353	208 816	271 255	334 718	397 157	460 620
020 052	083 515	146 978	209 417	272 880	335 319	398 782	461 221
021 677	084 116	147 579	210 018	273 481	336 944	399 383	462 846
022 278	085 741	148 180	211 643	274 082	337 545	400 504	463 447
023 903	086 342	149 805	212 244	275 707	338 146	401 609	464 048
024 504	087 967	150 406	213 869	276 308	339 771	402 210	465 673
025 105	088 568	151 007	214 470	277 933	340 372	403 835	466 274
026 730	089 169	152 632	215 071	278 534	341 997	404 436	467 899
027 331	090 794	153 233	216 696	279 135	342 598	405 037	468 500
028 956	091 395	154 858	217 297	280 760	343 199	406 662	469 101
029 557	092 510	155 459	218 922	281 361	344 824	407 263	470 726
030 158	093 621	156 060	219 523	282 986	345 425	408 888	471 327
031 783	094 222	157 685	220 124	283 587	346 026	409 489	472 952
032 384	095 847	158 286	221 749	284 188	347 651	410 090	473 553
033 504	096 448	159 911	222 350	285 813	348 252	411 715	474 154
034 610	097 049	160 512	223 975	286 414	349 877	412 316	475 779
035 211	098 674	161 113	224 576	287 015	350 478	413 941	476 380
036 836	099 275	162 738	225 177	288 640	351 079	414 542	477 502
037 437	100 900	163 339	226 802	289 241	352 704	415 143	478 606
038 038	101 501	164 964	227 403	290 866	353 305	416 768	479 207
039 663	102 102	165 565	228 004	291 467	354 930	417 369	480 832
040 264	103 727	166 166	229 629	292 068	355 531	418 994	481 433
041 889	104 328	167 791	230 230	293 693	356 132	419 595	482 034
042 490	105 953	168 392	231 855	294 294	357 757	420 196	483 659
043 091	106 554	169 508	232 456	295 919	358 358	421 821	484 260
044 716	107 155	170 618	233 057	296 520	359 983	422 422	485 885
045 317	108 780	171 219	234 682	297 121	360 584	423 023	486 486
046 942	109 381	172 844	235 283	298 746	361 185	424 648	487 087
047 543	110 503	173 445	236 908	299 347	362 810	425 249	488 712
048 144	111 607	174 046	237 509	300 972	363 411	426 874	489 313
049 769	112 208	175 671	238 110	301 573	364 012	427 475	490 938
050 370	113 833	176 272	239 735	302 174	365 637	428 076	491 539
051 995	114 434	177 897	240 336	303 799	366 238	429 701	492 140
052 596	115 035	178 498	241 961	304 400	367 863	430 302	493 765
053 197	116 660	179 099	242 562	305 001	368 464	431 927	494 366
054 822	117 261	180 724	243 163	306 626	369 065	432 528	495 991
055 423	118 886	181 325	244 788	307 227	370 690	433 129	496 592
056 024	119 487	182 950	245 389	308 852	371 291	434 754	497 193
057 649	120 088	183 551	246 507	309 453	372 916	435 355	498 818
058 250	121 713	184 152	247 615	310 054	373 517	436 980	499 419
059 875	122 314	185 777	248 216	311 679	374 118	437 581	
060 476	123 939	186 378	249 841	312 280	375 743	438 182	
061 077	124 540	187 501	250 442	313 905	376 344	439 807	
062 702	125 141	188 604	251 043	314 506	377 969	440 408	

```
        +-----------------------------------------------------+
        | INFO FOR PEOPLE WHO WANT TO MAKE PORTABLE DEMO FILES: |
        +-----------------------------------------------------+
```

First of all: SEE MY WARNING above with the Ctrl-FunctionKeys:
USE MY SPEED-FIX VERSION 3 (or higher) FOR RECORDING!
USE SPEED-FIX VERSION 6.1 (or higher) FOR NEVER-FAIL RECORDING!
(Never-fail: the demo always plays back as it was recorded!)

Note that there are roughly two types of demo's:
1) Original Supaplex demo format,
2) .SP format, used for SpeedFix 5.0 and higher, and SPEDIT v3.0 and higher.

The two formats are not compatible (you can't play demo's made with the SpeedFix v5.0 or higher with the original Supaplex version and vice versa), but they can be converted quite easily: The "new" format simply has the actual level prepended, and the rest is pretty much the same as the old format. Since the old demo's don't have the level prepended, they rely on the first byte (so it's the 1537th byte in the new format) which states the level number to be used when playing the demo. Obviously, the new format is very much preferred, since you don't need the correct level set for it: any loose .SP file will work.

Also note that from SpeedFix v5.3, the level byte (the 1537th byte in the .SP file) is actually increased by 128. This is to note that it's recorded with at least version 5.3. To get the correct level number (which is probably useless, since it's not used in the demo, and probably has to be changed for the old demo format anyway) you should ignore the highest bit (that is, you should subtract 128 if the value is higher than 128).

From SpeedFix version 6.2 the 10 original demo files are recognized and are the only exceptions on the above rules: they can be used with the SpeedFix! (I introduced this because I sometimes want to check with the original game, and the 10 demo's were just not compatible with the other EXE file.)

```
                    +-----------------------------+
                    | SUPAPLEX FILE ARCHITECTURE: |
                    +-----------------------------+
```

The LEVELS.D?? (LEVELS.DAT) architecture:
==========

- 111 single levels of 1536 bytes each, totaling 111 * 1536 = 170496 bytes.
  These levels are just glued together. See the "single level architecture".
  The original Supaplex uses only LEVELS.DAT. Additionally, for the SpeedFix,
  the ?? is a number from 00 to 99, for easy access to different level-sets.

- -

The single level architecture as used in LEVELS.D??, *.SP and DEMO?.B??:
==========

Bytes - Description

 1440 - Bytes 0000-1439
        The level itself (width * height = 60 * 24 = 1440)
        (See also below: "The level elements in the 1440 byte level")

    4 - Bytes 1440-1443
        unused (I found nothing in the program yet that uses these bytes!)

    1 - Byte 1444
        Initial gravitation: 0=off, anything else = on

1 - Byte 1445
20h = unused ASCII space character in the original SUPAPLEX.EXE and all SpeedFix versions up to v5.2. Used from versions 5.3 and up as: 20h + SpeedFix version number in hex format: v5.3 -> 73h, v6.2 -> 82h.

23 - Bytes 1446-1468
Level title in uppercase ASCII, stuffed with dashes ("-").

1 - Byte 1469
Initial "Freeze Zonks": 2=on, anything else (0) = off. (1=off too!)

1 - Byte 1470
Number of Infotrons needed. 0 means that Supaplex will count the total amount of Infotrons in the level, and use the low byte of that number. (A multiple of 256 Infotrons will then result in 0-to-eat, etc.!)

1 - Byte 1471
Number of special ("gravity") port entries below. Maximum 10 allowed!

60 - Bytes 1472-1531
Data-base of properties of up to 10 special ports, 6 bytes per port. Each of those 10 entries is formatted as:

[hi],[lo],[gravity],[freeze Zonks],[freeze enemies],[unused]

where:

[hi],[lo] : high and low byte of the location of a special port. If (x,y) are the coordinates of a port in the field and (0,0) is the top-left corner, the 16 bit value here calculates as 2*(x+(y*60)). This is twice of what you may have expected: Supaplex works with a game field in memory, which is 2 bytes per sprite. To calculate backward: Offset=(256*[hi]+[lo])/2 y=INT(Offset/60), x=Offset-60*y or: x=Offset modulo 60, y=(Offset-x)/60

[gravity] : 1 = turn on, anything else (0) = turn off

[freeze Zonks] : 2 = turn on, anything else (0) = turn off (1=off!)

[freeze enemies]: 1 = turn on, anything else (0) = turn off

[unused] : doesn't matter: is ignored.

The offset to entry N (=0..9) is calculated as 1472+N*6
The reverse calculation: N=INT((Byte-1472)/6), where byte number B (=0..5) into this entry is: B=Byte-1472-6*N
or: B=(Byte-1472) modulo 6, N=(Byte-1472-B)/6

4 - Bytes 1532-1535
Unused in the original SUPAPLEX.EXE and SpeedFix versions up to v5.1. Used by SpeedFix versions (5.2 and higher) to store some vital demo information. (Version 5.2 used only the last 2 bytes.)
Maarten and I did not want to disclose the exact use of these bytes, because it would make it easier to cheat with demo times for his Hall-Of-Fame. Now his Hall-Of-Fame is not updated anymore, so:
These last 4 bytes carry following information:
Byte 1532: scrambled Speed byte
Byte 1533: scrambled checksum byte
Byte 1534: low byte of 16 bit random number seed
Byte 1535: high byte of 16 bit random number seed

To byte 1532:
This byte carries the information of the slowest speed used during the demo recording. 0x00=fastest, 0x0A=slowest
This information is exclusive-ored with the high random number byte (byte 1535). (Each bit is toggled, where in byte 1535 a 1 appears.)
The result is the value of byte 1532 (and is used to scramble byte 1533).

To byte 1533:
All upper nibbles of each demo byte (without first level number byte and without ending 0xFF), each nibble incremented by 1, are added up. This total equals the total number of demo frames and reflects the normalized demo time with 35 frames per second.
To this total, of which only the lower 8 bits are used, the lower random number byte (byte 1534) is added.
The resulting lower 8 bits are exclusive-ored with the final contents of byte 1532. (Each bit is toggled, where in byte 1532 a 1 appears.)
The resulting lower 8 bits is the value of byte 1533.

Note: Megaplex does not put any information into bytes 1532 and 1533.

To bytes 1534 and 1535:
All Bugs are fired randomly, so in order to be able to make a recording of a level with Bugs, it is necessary to let them fire exactly at the same time in each playback of that recording. In order to guarantee that, we need a predictable random number generator and start it each playback with the same starting value (seed) as when the recording was started. When the sequence of all following random numbers is repeatable, all Bugs will always fire the same way during each playback as during the creation of the recording.
Luckily the original Supaplex uses a very simple random number generator for this purpose, which is not depending on external influences like date and time or a keypress. Start the random number generator with a random number seed and the next random number is calculated, which is also used as seed for the next calculation. A certain seed will always result in only one specific random number. The sequence of all following random numbers is thus fixed for each seed.
So at the start of each recording, we need to remember the starting random number as seed for the random number generator during each playback.

Each random number is a 16 bit number. After each random number calculation, only the lower 16 bits are kept as seed for the next calculation:
new_random_number_seed = ((old_random_number_seed * 1509) + 49) modulo 65536
This "modulo 65536" just signifies keeping only the lower 16 bits and reject all higher bits.

Note that since the SpeedFix has been used, there have been a few changes: The last 4 bytes are used now, and the '20h' byte is used for the version number. Also note that it may be possible to use the last 4 bytes for a gravity port, but that you shouldn't use it (you can't with SPEDIT) since it's already used in the SpeedFix.

Supaplex knows only how to display the ASCII characters 20h through 5Fh, which only are: { !"#$%&'()*+,-./013456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_}
but only those characters within and without the {} here.
This limits the level name to upper case, but all the other characters of this list can be used too. Maarten's SPEDIT doesn't accept the whole list (yet).

If Murphy passes a special port, the Number Of Special Ports at the end of the level is used as counter of how many data-base entries are to be checked. Is the address of an entry equals Murphy's location (inside the port), then the three properties are updated: Gravity, Freeze Zonks and Freeze enemies.
This way of handling the special ports means that:
- entries that are not pointing to a special port are ignored,
- unused data-base entries are ignored,

- special ports without database entries behave just like a normal port.
- when the port is blown up, nothing will happen if Murphy walks where the port used to be: without passing a special port, nothing happens.

- -

The level elements in the 1440 byte level:
==

The game field counts 24 lines of 60 sprites each, and these sprites are coded byte-wise, making the level size 60 * 24 = 1440 bytes. The level is stored the same way as you read this document: Starting from the left, top line first, then the next line, starting from the left, and so on. The last byte is the rightmost bottom corner.
Here are the sprite-code bytes (hex byte value, decimal byte value, sprite):

00h = 0	Space	
01h = 1	Zonk	
02h = 2	Base	
03h = 3	Murphy	
04h = 4	Infotron	
05h = 5	RAM chip (pins on all 4 sides, standard shape)	("Dull")
06h = 6	Hardware (gray dented pyramid, standard shape)	("Dull")
07h = 7	Exit	
08h = 8	Utility Disk, Orange	
09h = 9	Port left to right	
0Ah = 10	Port up to down	
0Bh = 11	Port right to left	
0Ch = 12	Port down to up	
0Dh = 13	Special port left to right (changing gravity etc.)	
0Eh = 14	Special port up to down (changing gravity etc.)	
0Fh = 15	Special port right to left (changing gravity etc.)	
10h = 16	Special port down to up (changing gravity etc.)	
11h = 17	Snik Snak (initially pointing upward)	
12h = 18	Utility Disk, Yellow	
13h = 19	Terminal for exploding yellow disk	
14h = 20	Utility Disk, Red	
15h = 21	Port vertical, bi-directional	
16h = 22	Port horizontal, bi-directional	
17h = 23	Port horizontal + vertical (cross)	
18h = 24	Electron (initially pointing upward)	
19h = 25	Bug	
1Ah = 26	RAM chip (horizontal left ("pin 1"))	("Fancy")
1Bh = 27	RAM chip (horizontal right)	("Fancy")
1Ch = 28	Hardware (radial blue circular cap + colored shapes)	("Fancy")
1Dh = 29	Hardware (green signal lamp)	("Fancy")
1Eh = 30	Hardware (blue signal lamp)	("Fancy")
1Fh = 31	Hardware (red signal lamp)	("Fancy")
20h = 32	Hardware (yellow/black diagonal stripes)	("Fancy")
21h = 33	Hardware (yellow resistor + blue + red shapes)	("Fancy")
22h = 34	Hardware (horizontal red capacitor + smd shape)	("Fancy")
23h = 35	Hardware (red + yellow + blue horizontal resistors)	("Fancy")
24h = 36	Hardware (3 red vertical resistors)	("Fancy")
25h = 37	Hardware (3 yellow horizontal resistors)	("Fancy")
26h = 38	RAM chip (vertical top ("pin 1"))	("Fancy")
27h = 39	RAM chip (vertical bottom)	("Fancy")
28h = 40	Invisible wall	

29h = 41 and above: Unused. Displayed trash. Invisible since version 6.0.

Supaplex scans the level from the beginning to find Murphy, and activates the first available Murphy. If there are more Murphies in the level, the leftmost of the topmost Murphies is therefore the lucky one, but the other Murphies

must be protected because when the program sees any attacked Murphy, it does not check if it is the active Murphy, which causes all Murphies to die. (My level "TRIVIAL" was also the first level to use this as a feature, but because of the problems with SPEDIT, it was only released after my "SUICIDE".)

In order not to test too many different byte-values, Supaplex changes the "Fancy" stuff to "Dull" stuff internally before the game starts. (During the game, each sprite is checked for all different possibilities, and many decisions have to be made this way. Supaplex does not use the faster and more intelligent method of "indexing", which would make these changes from "Fancy" to "Dull" unnecessary. I think that maybe the programmer decided for his way in an early stage of the game-design , where the amount of different sprites did not justify the use of "indexing".)

In the "normalized" or "Dull" game field, code 1Fh is used for explosions. This is why in previous versions of the SpeedFix explosions sometimes showed or flashed as red signal lamps in debug mode.

The invisible wall was not known to the programmers. Experimenting with "illegal" bytes, I found this wall and designed the two levels "TRIVIAL?" and "SURFIN'" with it. Maarten's SPEDIT (2.0 at that time) automatically changed this wall to some hardware, destroying those levels. I told Maarten about this wall and asked him to change his SPEDIT to not destroy that wall anymore. At first he protested, saying that you never know what might happen using it, but the result is: We now have this invisible wall "officially", and his SPEDIT 2.0 was updated to 3.x to include this wall.

- -

The LEVEL.L?? (LEVELS.LST) architecture:
==========

The LEVELS.D?? must be accompanied by their own LEVEL.L?? (LEVEL.LST) files, which is only used for speeding up the loading of the game (from floppy disk).

- 111 lines of 27 bytes of level information plus a 1-byte line-terminator, totaling 111 * (27 + 1) = 3108 bytes.
  Each of these entries MUST be exactly 28 characters long:

    3 - ASCII digits level number ("001" up to "111")

    1 - ASCII space character (" " or 20h)

   23 - Level name in uppercase ASCII, including the "-" stuffing.

    1 - ASCII line-feed character (0Ah), without carriage-return character!

Example: "055 -- THIS IS MY LEVEL ---" + <Line-Feed>

All information in this file is also available in the LEVELS.D??, so why should any missing LEVEL.L?? prevent Supaplex from starting the game? From SpeedFix version 6.2 these files became optional: If it exists, it is used as before, otherwise the information is extracted from LEVELS.D??. When the command line parameter "C" or "c" was used, the missing LEVEL.L?? is then created from that reconstruction. (It does not overwrite existing files.)

The original LEVEL.LST differs from the reconstructed file, because of the separator in the name of level 042 "LITTLE?PLEASURE": space versus dash.

- -

The original Supaplex DEMO?.BIN architecture:
==========

These files as described here can only be used with the original Supaplex and SpeedFix V4 and below, but this original demo format is exactly the demo part of the new DEMO?.B?? and *.SP files (with the exception of the highest bit of the very first byte)! The 10 original demo's from the original game are the only old-style demo's that now also work again with SpeedFix 6.2 and above.

- Level number byte of the level to be used with the demo.
  Legal values are hex 01h to 6Fh (=111 decimal). (New format: hex 81h to EFh)
  This is an important byte of the original Supaplex demo's because it tells which level to load with the demo.

- Any amount of demo bytes up to a maximum of 64008 bytes, of which each byte carries the information which key was pressed and how long the key was pressed, counted in Supaplex time units of 1/35th second (at normal speed). These time units are determined by the video frame frequency (and affected by the speed setting of the SpeedFix keys). The time units are a multiple of the official 70 Hz video-frame-frequency tics. Some notebooks use 60 Hz. Each demo byte is splitted into a high nibble and a low nibble:

  -- High nibble: Time-unit repeat count minus 1 for low nibble command key.
  This is the information from which the total demo duration is calculated, corrected with the (stored) SpeedFix setting. At normalized speed, these time units are 1/35th of a second each. Example: If this nibble shows 9, it means 9+1=10 time units for the command key in the lower nibble.

  -- Low nibble: Command key, equivalent with the keyboard game keys:
  0 = just wait: no key pressed
  1 = cursor up
  2 = cursor left
  3 = cursor down
  4 = cursor right
  5 = space + cursor up
  6 = space + cursor left
  7 = space + cursor down
  8 = space + cursor right
  9 = space only
  A = (not used)
  B = (not used)
  C = (not used)
  D = (not used)
  E = (not used)
  F = (not used! / low nibble of the end-of-demo byte)

- hex FFh byte, meaning: end of demo.

The maximum demo file length is 1+64008+1=64010 bytes, which is an arbitrary amount set by Robin Heydon (which could have been 65514 bytes maximum only).

This maximum total demo length of 64010 bytes has been reduced in the SpeedFix versions 5 and up to 48650 bytes because the same space is needed for all 10 demo levels of 1536 bytes each. (48650+10*1536=64010) See also just below.

- -

The *.SP and new-style DEMO?.B?? architecture:
==========

- A single level of 1536 bytes, according to the architecture description.
  If there is no demo attached to an .SP-file, the file ends here.
  In case there is a demo attached (*.SP with demo or new-style DEMO?.B??),
  5 bytes of this level are used to carry important information for the demo
  itself, for the recording speed, for the demo validation and to recognize
  the SpeedFix version used to record that demo (to be able to tell which
  of that other information is available and which were not yet stored!
  SpeedFix V5.0 used only 3 of those 5 bytes).
  (These 5 bytes are ignored by the game and overwritten with new information
  with each new demo recording, so it is safe to ignore the changes if any
  demo is removed from this level for placing it in a LEVELS.D?? file.)

- A complete demo as described in the original DEMO?.BIN architecture.
  In order to know that the demo was accompanied by valid extra information,
  stored inside the attached single level, Maarten changed the unused highest
  bit of the first byte from 0 to 1 (hex 81h to EFh). Because the level is
  now attached, this first byte lost its original function completely.
  In case there is no signature appended, the file ends with the hex FFh byte
  from the original demo architecture (see above: end of demo).

- Any text of up to 511 bytes long: the demo signature introduced by Maarten
  in October 1996, and the only restriction is, that it contains no ASCII 255
  (FF hex) byte, since that byte is used to signify the end of the signature.
  This signature is meant for comments, the name of person that played the
  demo, and anything else anyone can think of.

- FF hex byte, meaning: end of signature text and also end of file. This byte
  is defined as the last byte of the signature, totaling 512 bytes maximum.

The maximum new-style demo length is 1536 + 48650 + 512 = 50698 bytes.
(This 48650 is the total amount of reserved space for all 10 demo's, which is
just an arbitrary amount set by Robin Heydon in the first place, and changed
by Maarten as he stored the 10 demo levels in the same reserved space (Ver.5))

Up till now there is only 1 level (level 62 of level set 03, "Waiting for
Godot") which needs a (VERY MUCH!) larger space for a finishing demo than is
available, even if the original absolute maximum of 65514 bytes could be used.
("Now" is June 2007.) This demo would take about 700000 bytes, and it would
last for about 3.7 days in normalized speed, which is way beyond Supaplex's
capabilities.

Maarten's program SPSIG manages the signature: adds a text-file as signature
to SP files, strips signatures from SP files and shows SP file signatures.

- -

The Megaplex *.MPX architecture:
==========

MPX files have an extra 20 byte header, the rest equals SP demo files.

###{Start of MPX-header}###
4 bytes: "MPX " (0x4D 0x50 0x58 0x20) for recognition
2 bytes: Megaplex version number (0x0001, unused)
2 bytes: Number of levels in this file (0x0001, unused)
2 bytes: X size (for instance: 0x000D), width of the level, including border
2 bytes: Y size (for instance: 0x0010), height of the level, including border
4 bytes: biased offset to the level (0x00000015, unused)
4 bytes: total length of the file minus those first 20 bytes (for instance:
         0x00000130 for a specific level only, 0x000000030C with demo attached)
         This value is used by Megaplex to only fetch so many bytes. If this

```
         value is less than the actual level-plus-demo length, all bytes beyond
         this value are ignored and playback of the demo will not finish, even if
         the file contains the completed demo.
###{End of MPX header, start of level info}###
   (The level info is the same as described in the *.SP and new-style DEMO?.B??
   architecture above:)
X*Y bytes: game field (SP size is X=60=0x003C and Y=24=0x0018, total 1440 bytes)
96 bytes:  level info, same as standard SP files, except for 3 SpeedFix bytes:
           Bytes 1445, 1532 and 1533 as described in the level architecture.
           These are the specific SpeedFix recorder version byte (byte 1445) and
           the speed-info and checksum bytes near the end of the level, when a
           demo is attached. These are untouched by Megaplex
           (The last two level bytes <just before the demo bytes> with the random
           number generator start value is exactly the same as in SP demos, also
           the used random number generator itself is identical for maximum
           compatibility: Megaplex uses the exact code of SpeedFix 6.3+.)
   (It is clear, that with MPX files, the number of special ports is also limited
   to 10 maximum, as with SP files.)
###{End of level-only MPX-file, start of optional demo information.}###
   (MPX demo information is exactly the same as SP demo information:)
variable number of bytes: SP demo, ending with 0xFF
   The demo length is not as limited as with the SpeedFix: A demo of the infamous
   level "Waiting for Godot" is possible as MPX-Demo. The size-limit is mainly
   prescribed by the variable type of the total length value in the MPX-header.
###{Optional signature, after a demo, same as with SP demos:}###
1-512 bytes: SP signature, ending with 0xFF
###{End of all MPX files}###

Note: The present Megaplex versions seem to suffer from following bugs:
Original version from Frank Schindler:
- When a demo is recorded, one must wait several seconds before saving the demo,
  otherwise any previous attempt is saved, or otherwise only the level.
- When entering the level editor, the number of Infotrons is reset to 0 = all.
  This results mostly in bad information, not only because Electrons are not
  counted then, but also many levels are designed for a different number.
- When a smaller recording is made over a previous longer recording, the
  resulting file has the extra old bytes attached, which results in a bad display
  of the level info (signature).
Adapted version from Paulo Matoso with nicer high resolution graphics:
- Same bugs as the original Megaplex, plus:
- Large levels crash, like “SEEING YELLOW” from the Macintosh game Infotron
- Zoom function does not work properly anymore with these new graphics.

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

The PLAYER.L?? (PLAYER.LST) architecture:
=========================================

This file contains the results of all 20 possible players.

This is a hidden file, which can be revealed with "attrib -h PLAYER.L??".
(Don't bother to hide it again: this is done automatically by Supaplex.)
If this file is missing, Supaplex creates an empty new file, where all 20
names are initialized as "--------", and all other bytes are 0.

This file has 20 entries of 128 bytes each.  Each entry is:

  8 - Player name in upper case ASCII  (empty name = "--------")
  1 - end-of-name terminator: must be 0!
  3 - Total time played until now:
      - 1 byte: hours   (hex)
      - 1 byte: minutes (hex)
      - 1 byte: seconds (hex)
111 - Result for each level: 1 byte per level: 0=unplayed, 1=played, 2=skipped
  3 - unused (=0)
```

```
  1 - Ignored, but written by Supaplex as:
      Pointer to the first unplayed level, or if all levels are "done",
      to the first skipped level.  If all are played, this byte becomes 71h.
  1 - unused (=0)
```

- -

The HALLFAME.L?? (HALLFAME.LST) architecture:

This file shows the results of the 3 best players who finished all 111 levels.

This is a hidden file, which can be revealed with "attrib -h HALLFAME.L??".
(Don't bother to hide it again: this is done automatically by Supaplex.)
If this file is missing, Supaplex creates an empty new file, where all 3 names
are initialized as ASCII spaces and all other bytes are 0.

This file has 3 entries of 12 bytes each. Each entry is:

```
  8 - Player name in upper case ASCII  (empty name = "        ")
  1 - end-of-name terminator: must be 0!
  3 - Total time played for all 111 levels:
      - 1 byte: hours   (hex)
      - 1 byte: minutes (hex)
      - 1 byte: seconds (hex)
```

- -

The SUPAPLEX.CFG architecture:

This file contains the options, as chosen in the 'controls' screen.
It is always only 4 bytes long, in readable LOWER CASE(!) ASCII, and is only
written when leaving the 'controls' screen (just before panning to the menu).
The bytes are:

```
  1 - "i" = Internal speaker                  (=default)
      "s" = Samples (+ Internal speaker)
      "a" = Adlib
      "b" = Sound Blaster (+ Adlib)
      "r" = Roland
      "c" = Combined (Roland + Sound Blaster)
      (Any other value = default "i", but without loading the sound driver.
       Entering and leaving the 'controls' screen without changing anything,
       will then result in a "b" in this file, which could lead to a crash!
       This bug has been fixed in SPFIX62 to load the speaker driver too.)

  1 - "k" = keyboard                          (=default)
      "j" = joystick
      (Any other value = default "k")

  1 - "n" = music off                         (=default)
      "m" = music on
      (Any other value = default "n")

  1 - "y" = FX off (FX=sound effects)         (=default)
      "x" = FX on
      (Any other value = default "y")
```

If Supaplex crashes after you made bad changes in the 'controls' screen,
just delete this file and the Supaplex internal defaults "ikny" are used.

Any bytes attached with an editor (like Return/Line-Feed) are ignored.
Missing bytes (if the file is too short) result in defaults for those bytes.

Each of these 4 bytes has its own meaning and is not recognized if the correct value is on the wrong place: "rjmx" works, but "xmjr" defaults to "ikny".

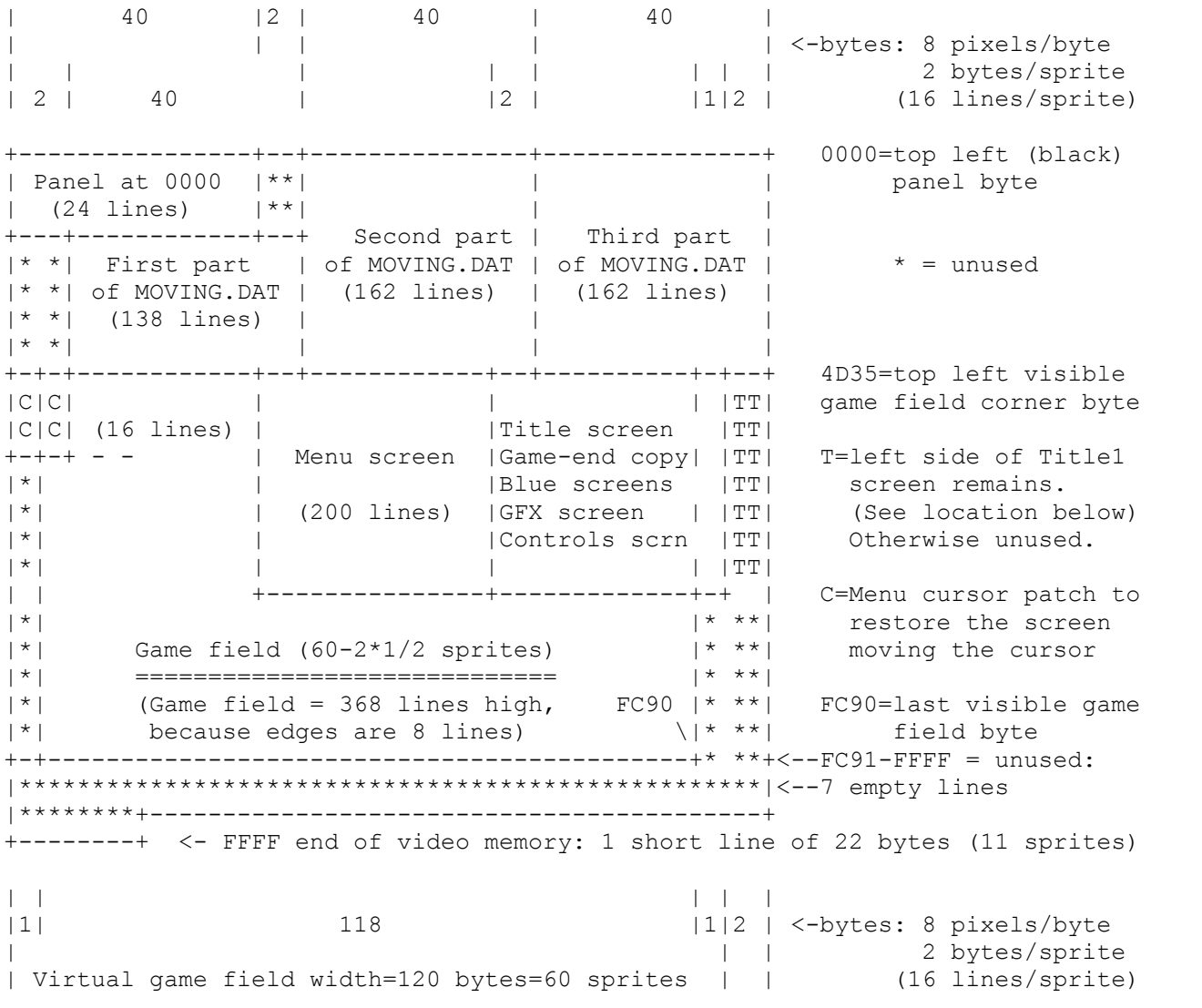
From SPFIX62, this file is not checked for the "i", "k", "n" and "y" anymore because those were defaults anyway. Any 'illegal' byte , which now also includes those "i", "k", "n" and "y", forces the default choice for that byte. (The file is still written with those "i", "k", "n" and "y".)

MAP OF THE VIDEO MEMORY:

If the debug M key is used to move the field in stead of Murphy, we see a lot of strange things outside the game field. (SpeedFix version 6.2 and above need the command line option 'M' to view beyond the game field borders!)

The video memory is used as follows (Use a fixed pitch font to view this!):

The 65536 bytes video RAM is divided in lines of 976 pixels horizontally, which means 976/8=122 bytes, making 537 lines plus 22 extra bytes in total. In 320*200 video mode, each picture line is written twice, which means that there are 400 horizontal lines on screen and each line is divided in 320 dots, just in case you want to count them.
Following graphic shows picture lines, and not these doubled screen lines.



A Supaplex sprite is 16*16 pixels, which means 2 bytes wide and 16 lines high.
The virtual game field measures 60*24 sprites but the real game field is
2 * 1/2 sprite less in both directions because the edges are half-sized only.
Horizontally those missing halves are reserved (but unused), but vertically
those missing 2*8 lines are not reserved and cause problems when Murphy walks
on those edges: The MOVING.DAT pictures in video RAM are overwritten then
because Murphy is always updated full size on screen.

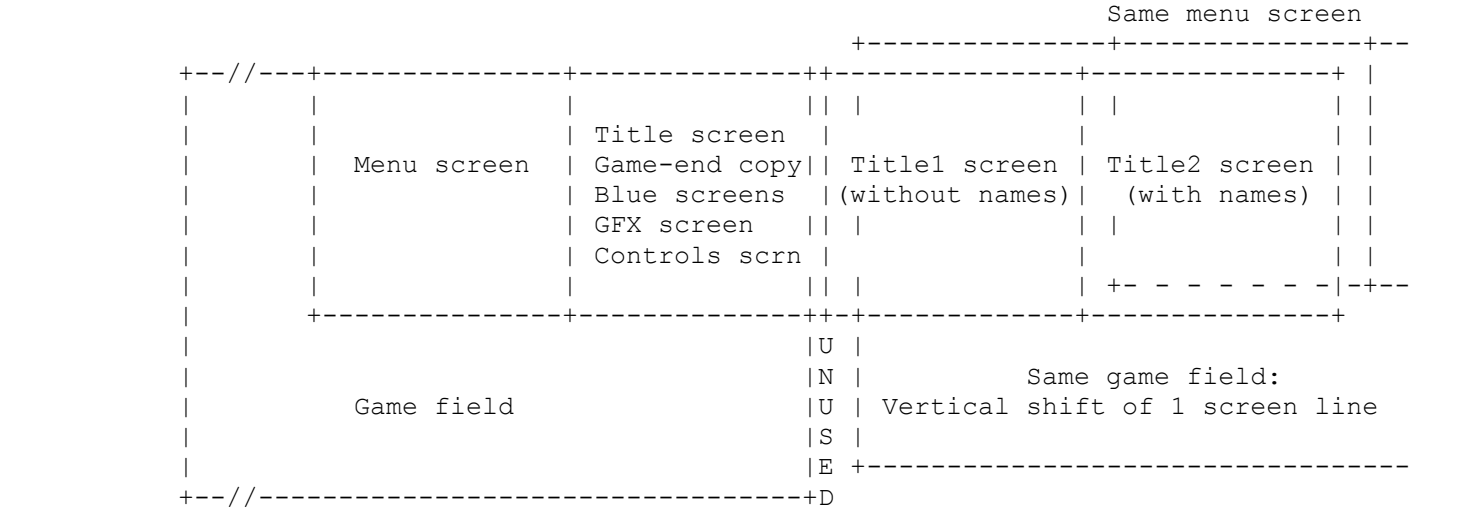
Hexadecimal addresses (for insiders):

0000 = Panel start address: must be here for the "split-screen" option.
4D34 = Game field start address, starting with 1 invisible (unused) byte.
4D34 = Buffer for menu cursor screen patch, destroyed by cursor graphics.
4D5C = Menu screen start address (top middle of game field)
4D84 = Special screens start address (Title (=BIG Murphy),GFX,Controls etc.)
4DAC = Title1 screen start address (before and during opening of the curtain)
4DD4 = Title2 screen start address (after curtain opened, with all names)

FC91h-FFFFh unused: 3 bytes on the last line + 7 lines + 22 bytes: 879 bytes
The 22 bytes cause the shift of 11 sprites from the field above/below.
The 7 lines account for the black stripe between those two game fields.
Note: The first line of the panel is a black line in the panel-graphics.

The menu cursor overwrites the menu graphics and therefore the part of the
menu graphics that is "destroyed" that way, is remembered at the beginning
of the game field. (The left part of the last copy shows there.)

The locations of the author screens (where the "curtain" opens) look a bit
strange, but with a different drawing those have logical positions too (as
if the authors planned extra panning, which is not possible because the
palette colors differ too much).



The "trash" in the unused area is thus also caused by the Title1 screen, and
from SpeedFix version 6.2 also by the extra copy of the viewing window before
the game pans back to the menu. (This is done to remove the screen "glitch".)

ABOUT THE GAME FIELD EDGES:

The game field edges (where Murphy is not supposed to walk on) are included in the level, which makes the actual playable level-size only 58 * 22 sprites. These edges are always substituted by Supaplex to show special edge sprites, which make it possible in experimental levels to hide sprites there.
(My level "EXIT?" was the first level to use this as a feature.)

For speed reasons, Supaplex does no border-checking, so everything may happen when Murphy (or other moving stuff) enters those edges. Because of the way the level is stored in memory, it is possible to walk out of the field on one side entering the field at the opposite side, but shifted one line. (This only applies to the left and right field edges, not the top and bottom edges.)
(I introduced this as a feature in my son's level "KRUEMMELKRAM", and my level "OVERFLOW" was exclusively designed to show what can be done with it.)

If level designers want to let Utility Disks explode on the edges (as my son wanted in "KRUEMMELKRAM"), they have to be careful not to put any explosives in the top left corner, one sprite below that, and 8 sprites in the top edge on the far right side. Explosions extend outside the game field and destroy some internal variables like the file name LEVELS.D?? and the pointer to the active level... The menu will show the wrong level and it is impossible to do anything but exit Supaplex. I showed those forbidden sprites in the level "KRUEMMELKRAM" as RED signal lamps. Although I fixed the problem in SPFIX62 (with a small buffer zone), those 10 sprites may not be used for explosions in new levels because of those problems in the original game! (Compatibility!)

If Murphy walks into any of the 8 sprites in the top edge on the far left, he is beamed to undefined places because the internal look-up table of screen-coordinates, which is used for the fast placement of Murphy on screen, was overwritten by an overlapping scratch buffer for fading calculations.
This screen coordinate table is defined at load-time and never refreshed.
(Fading is this optical effect of slowly making the screen dark or bright.)
I also fixed this in SPFIX62 by correctly sizing that scratch buffer.
It was an unimportant bug, but it was a bug anyway, therefore I fixed it.

Rules for level designers for safely using edges:
==========

- The left and right edges, without the corners, are safe for Murphy to exit, but the sprite just underneath the top left corner is not safe for any explosion, including Snik Snaks, Electrons or Murphy when he gets killed! For a demonstration of using left and right edges: See my level "OVERFLOW".

- No explosions are allowed in the top left and top right, as described above. (See the RED signal lamps in my son's level "KRUEMMELKRAM".)

- Almost anything may be hidden in all edges, as long as Murphy cannot remove it from the top or bottom edge (or destroy it by an explosion!), because:

- Murphy may not be allowed to walk on or outside the top or bottom edges!

  Even those 2 edges have side-effects. Just try entering the 20th edge field to the right of the top left corner, or the 8th edge field to the left of the bottom right corner: The master of Murphy's image is mutilated: Either the top of his head is missing or he has a black belt when he is not moving, or when he is pushing a Disk. And Murphy did not even walk outside of the whole game field this time ...
  This is caused by the fact that the top and bottom edges are really half-sized on screen, but Murphy is painted full-size into the MOVING.DAT area.

  (Exception of items which you may put on the top or bottom edges: Terminals! You can activate all terminals by hitting one of them. This first terminal

you hit, is completely painted on screen. This also applies to the top and bottom edges, with the side-effect that the MOVING.DAT area is mutilated, the same way as described above, where Murphy enters the edges...)

Murphy will really leave the memory image of the game field, with unknown side-effects! There are three different aspects to observe. Remember that the video memory is the graphical image of the real game field in memory:

1: - Video memory (What is actually visible):
This mutilation of Murphy is caused by the destruction of the video memory contents. The panel and all the moving sprites from the file MOVING.DAT (which includes the not-moving Murphy during the game) are stored there. Murphy popping up in the panel with such "experiments" is another example of this aspect: Although Murphy is somewhere else, his ghost is in the panel. Time to restart Supaplex, because MOVING.DAT is only copied to the video-memory once. (The panel is refreshed all the time so this doesn't matter.) (I provided a command line option "R" or "r" in version 6.2 to refresh MOVING.DAT in the video memory when the game returns to the menu. I did not make this refresh option default because it increases the time to return to the menu, especially when the game runs from floppy disk, and it is actually only needed after such experiments. Besides that, it only refreshes the video memory, but more could have happened ...)

2: - Location coordinates (Table of video memory locations):
The now fixed top left corner (see above) was a special example of destroyed coordinates, but Murphy can walk outside of the memory copy of the game-field, which also means outside of that coordinate table. There are no valid coordinates defined there and so he can pop up everywhere and nowhere, at random.
The panel is a preferred place since it is situated at the start of the video-memory, which is hit when such invalid coordinates are near 0.
Nothing is destroyed because of those missing random coordinates alone (except for the video memory contents), but:

3: - Working memory (The actual game field that matters):
The main aspect is what happens with the real game field in memory:
When Murphy exits the top edge and moves around beyond the top edge near a certain point, he cannot come back or the screen turns white or worse.
The table with the keyboard compare values or the byte with the debug E-key flag has been destroyed. Other things may happen too...
This is problematic because much of that stuff is only defined at load-time and is NEVER refreshed. The worst thing what theoretically can happen, is that the only way-out could be a cold boot. Time to RESTART SUPAPLEX!

(The bottom edge is not that problematic, because Murphy cannot destroy too much over there, at least not that I know of.)

An easy way to experiment with this: start any level in debug mode and remove most stuff with the debug keys "C", "Z", "S" and "H" ...

Through the bottom edge you can also find some Infotrons, and also an Exit. This is not because of what is in video memory (and what can be seen with the debug M key), but because the working memory accidentally has some RAM locations outside of the game field filled with "appropriate" values. Kim Min-Soo released a level (07S062: "Hidden track") which exploits this, but I cannot guarantee (although I'll try), that this level will be solvable in new SpeedFix versions, because the outside of the game field can change! Be careful: This level may not be seen as a "standard" level, and all warnings above about not using top and bottom edges still apply!

+--+
| REMARKS ABOUT SUPAPLEX GAME FIELD CALCULATIONS AND RESULTING TRICKS: |
+--+

(This list is certainly not a complete list of strange Supaplex behavior...)

Supaplex re-calculates the whole game field each frame, which happens each 1/35th of a second at normalized speed (or 1/70th of a second in high speed). Supaplex scans the frame from top-left to bottom-right, line by line, and remembers the location and type of each moving object in the field, without changing anything in the game field yet.

The type of the object determines the animation procedure-to-call for it. These moving objects are Snik Snaks, Electrons, Zonks, Infotrons, Orange Disks, Murphy, Explosions, and all different pushing Murphies with Zonks, Orange Disks and Yellow Disks.

After this scan, this memorized table of locations and procedures-to-call is executed, item for item, in the same order of the scan.

By first scanning and moving afterward, none of the movements are influenced by any other movement-calculations in the same time frame.

This way of delayed calculating also leads to imperfections, which we can use as special tricks.

Examples of tricks due to delayed calculations:
==========

Murphy is found next to a Zonk, and starts pushing against it. At the same time the Zonk does not know about that fact (because of the buffering), and decides to fall in a space beneath it. The two of these movements together result in the Zonk-cloning trick (together with the fact that the Zonk-movement calculations don't stop where they should.)

Exactly the same happens with Orange Disks, with the exception, that the falling Disk cannot fall any further if the top part has been pushed away, because the calculations are slightly different. The net result is, that we have some remains of a falling Orange Disk in the space below, which can be either invisible or visible, depending on how far that Disk fell, or when Murphy started pushing against it. These remains are known as the deadly invisible wall, which is not recognized later by a next scan as moving object, and hence works as (deadly) obstacle.

These delayed calculations are also responsible for the well known trick of Murphy moving sideways from the middle of a pile of three Zonks: Both middle and top Zonk fall in the direction of where Murphy was before. What happens here is that during a scan, the field where Murphy moved from, is recognized as space, and the top Zonk cannot fall down, so it decides to slide into that space. During the same scan, the middle Zonk sees the space too, and decides to slide off from the bottom Zonk. Now both Zonks already started to slide, and cannot change their minds to do something else now. The result is known. This trick can also be explained by the way Zonks reserve space to fall into.

Examples of a few other imperfections:
==========

Explosion clouds can remove part of a falling object, or remove a reservation of the space below a falling object. Those objects cannot fall any further, and are not seen as a normal object anymore, because something is missing. This explains why sometimes complete (or parts of) Zonks and Infotrons "hang in the air", and cannot be moved or eaten.

Sliding of a Zonk or Infotron implies, that the first movement is sideways, and then downward a few frames later. Because this empty space below is not

reserved, Murphy can enter that unreserved space below without being killed, or an object can be pushed into that empty space, before the downward movement starts. The sliding Zonk/Infotron will hang frozen, partly rotated, above that now occupied space, until that space is emptied again.
Anything falling onto such a partly moved Zonk/Infotron does not see an ordinary Zonk/Infotron, and will not slide off from such an obstacle.
This can be very helpful to reserve a passage for later, or to prevent a pile of objects from falling.

When Murphy pushes against an object, the scan calculations check if there is an empty space on the other side of that pushed object.
This empty space immediately changes its status to "reserved", but the pushing movement itself is delayed on purpose. Murphy may decide to stop pushing, which then releases the space on the other side of that pushed object.
This space-reservation results in the trick, where Murphy can change the movement of Snik Snaks and Electrons just by pushing against an object, and then abort this push.

More examples of "strange calculations":
==========

Such a half-pushed Zonk does not explode, if it is pushed this way underneath a falling Orange Disk ... (This is a Supaplex bug, which I will never "fix".)

The first frame after a Zonk (etc.) decides to move, only the status of that Zonk (etc.) has changed, without performing a real movement. The space-reservation for the movement starts one frame after this status change.
Here we have the explanation of several other tricks:

Murphy can catch Zonks and Infotrons by moving quickly from underneath it and back again, because the field underneath these objects stay empty for one frame after they found the space beneath them to fall into.
Murphy's reactions are faster, and can immediately re-occupy that field, before that space is reserved by the Zonk (etc.) which would kill Murphy.
Now the Zonk (etc.) sees that space occupied again and cannot enter that space where Murphy moves into, and also fails to kill Murphy.
Because that falling Zonk (etc.) initiated its movements already, Murphy cannot do this trick twice, because the next scan will let both the Zonk and Murphy occupy the same space, killing Murphy.

Any Zonk or Infotron immediately above such a Zonk, which changed its status, does not see a normal resting Zonk beneath it, and therefore will not slide off. And because they also see no empty space beneath them, they did not start falling either, which makes them pushable or eatable.

A Red Disk, released underneath a Zonk or Orange Disk, lets those objects pass right "through" it, and occupy temporarily the same space.
If we initiated the descent of a Zonk in a Zonk pile before that Red Disk release, we can destroy more Zonks than otherwise, because the resulting explosion cloud will happen to include more Zonks. (Lauterkranz trick.)

If a Zonk needs to fall faster because of a time-critical maneuver, its fall can be initiated on beforehand by creating an empty space underneath it.
We must use this trick in the level "Romancing The Stones!" (Level 01S011) in the top-right side of that level: Murphy holds a mixed pile of Zonks and Infotrons, eats a base sideways from underneath another Zonk, immediately passes underneath that Zonk and as soon as this Zonk fell, pushes it into the falling pile, to be able to eat all Infotrons from that pile. The timing for pushing the Zonk into that pile is so critical, that it is impossible to push the Zonk into the pile at the desired place, without speeding up the fall of that Zonk first. (Alexei Boreisho was the first person to officially solve that particular level, which was considered impossible at that time and was therefore almost deleted from that level set.)

If a Red Disk is eaten, Supaplex does not test if it has just been released.

This leads to the forbidden Red Disk cheat (by Tom Geelen), which I 'fixed' in version 6.3, but because it changes the game itself, I made it possible to enable this cheat again with an extra command line switch "T".

DISCLAIMER: We cannot be held responsible in any form for any damage etc. that might occur using software and knowledge presented by us. Use at own risk.

Contact information:
==========

If you still have any questions, please take a look at the Supaplex WebPages at http://www.elmerproductions/sp/ and also check the Supaplex forum at http://www.infordigital.com
We can be reached via email (preferred) or normal mail at:

Normal mail:	E-mail:	Phone:
Herman Perk Spessartstrasse 15 D-14197 Berlin Germany	Herman_Perk@hotmail.com Herman_Perk@compuserve.com	Home: +49-30-8222255 Mobile: +49-160-8454107
Maarten Egmond (Old address:) Schoutstraat 24 NL-5663 EZ GELDROP The Netherlands	ep@elmerproductions.com	