MonkeyCantab

A Whisker client

---

by Rudolf Cardinal

www.whiskercontrol.com

Distributed by Campden Instruments Ltd (www.campden-inst.com)

MonkeyCantab

Printed: July 2009 in Cambridge, UK

Creator (Whisker)

Rudolf N. Cardinal

Design and Programming (Whisker)

Rudolf N. Cardinal

Michael R. F. Aitken

Legal Advisor (CUTS)

Adjoa D. Tamakloe

Sales (Campden)

Julie Gill

Contacting the authors:

For information about Whisker, visit http://www.whiskercontrol.com/.

If you have sales enquiries about Whisker, contact Campden Instruments Ltd at http://www.campden-inst.com/.

If you have comments or technical enquiries that cannot be answered by the sales team, contact the authors:

Rudolf Cardinal (rudolf@pobox.com)
Mike Aitken (m.aitken@psychol.cam.ac.uk)

Table of Contents

Foreword

WARNING

Whisker is a system designed for research purposes only, and should never be used to control medical apparatus or other devices that could endanger human life.

DISCLAIMER

The authors, copyright holders, and distributors disclaim all responsibility for any adverse effects that may occur as a result of a user disregarding the above warning.

1 MonkeyCantab

1.1 About MonkeyCantab

Purpose

Cognitive test battery for animals.
Written for the University of Cambridge.
The battery includes:

- Reinforcement familiarization
- Touchscreen training
- Visual discriminations (simple and compound) with reversal and intradimensional and extradimensional set-shifting
- Reversal learning
- Delayed matching and non-matching to sample
- Self-ordered search (a.k.a. spatial working memory)
- Multiple-choice serial reaction time
- Paired-associates learning
- Simple schedules of reinforcement
- Impulsive choice (delayed/probabilistic reinforcement choice task)

Software requirements

Requires Whisker v2.7 or greater (www.whiskercontrol.com).

Data storage

- Text-based output to disk.
- ODBC data storage to a database (supplied).

Author

Rudolf Cardinal (rudolf@pobox.com).

Acknowledgements

Thanks to Mike Aitken, Shibley Rahman, Hannah Clarke, Susannah Walker, and Laurence Argyle for helpful discussions regarding the ID/ED, PAL, SWM, and reversal learning tasks.

Copyright

Revision history

- 27 March 2003. Started (version 0.1).
- 8 April 2003. Version 0.1 released for spatial working memory researcher. Currently up and running: general parameters, reinforcement familiarization, schedules of reinforcement, spatial working memory. Other tasks disabled.
- 11 April 2003. Version 0.2: RF, TT, VD-predefined, VD-superimposed, D(N)MTS, SWM,

MCSRTT, schedules.

- 6 May 2003. Version 0.3: reversal learning in; general controller bug fix (punishment sound Y/N mixed up with reward sound Y/N); SDK bug fix (filenames not enclosed by quotes); visual discriminations bug fix (rewarded stimuli left up for too long); PAL in. Display caching added to all tasks. Released within University of Cambridge.
- 7 May 2003. Version 0.4: changes requested by CamCog. Two additional output lines, REINFORCEMENT_INFO and STIMULUS_INFO, added (see Required Devices); REINFORCEMENT_INFO gives 10ms pulses on reward, 20ms pulses on punishment; STIMULUS_INFO is high when stimuli are being displayed in ID/ED tasks (both varieties) and reversal task (during choice phase only, not when stimuli are left up during reward); option to enforce that an empty box is the last to be displayed in the sample phase of PAL. New configuration file version number. Option to use offset grids with central dispenser (but actual grids for this not yet implemented). Modifications to PAL (4-way edges grid; option to shuffle presentation order when repeating trials). Background touches recorded in SWM.
- 24 July 2003. Added predefined stimulus sets. Now requires WhiskerServer 2.6.8 or greater (with CamcogQuadStimulus built in to the server).
- 30 July 2003. Version 0.5: DMTS and TouchTraining (and, indeed, pretty much all tasks) now compatible with previous CamCog as well as University of Cambridge tasks.
- ...
- 11 March 2004 (approx). Version 0.9. Numerous bug fixes and improvements; see **www.whiskercontrol.com/version_tracker/MonkeyCantab.txt**
- 6 May 2004. Version 0.91. Changes to PAL stimulus order and WAV file volume selection.
- May 2004. Version 0.92. SWM bug fixed (all touches were erroneously punished if there was no "delaying tactic" between consecutive stimuli).
- 4 June 2004. Version 0.93. For full details see the version tracker (as above). Summary of changes follows. **(1)** Visual discriminations: if there is an SR (simple reversal) stage, the compound discrimination (+/- compound reversal) stimuli are altered such that there is no longer a reversal upon shifting from SR to CD. Nor is there a reversal moving from SD to CD if no SR stage is used. **(2)** Visual discriminations and Reversals: ITI touches are recorded, and (optionally) punished. **(3)** D(N)MTS: task variant with simultaneous presentation of phase 1/2 stimuli. **(4)** SWM: probe trial facility.
- 25-29 June 2004. Version 0.94. Several new facilities added to the MCSRT task.
- 6 July 2004. Version 0.95. Now informs user of problems encountered when saving configuration files.
- 13 July 2004. Version 0.96. Choice of background colour.
- 20-31 August 2004. Version 0.97. Record *all* events (or, at least, all that seem even vaguely relevant, with a generic event-recording facility so if someone decides something else is important it's easy to add!).
- 15 October 2004. Version 0.98. Bug fixed in DMTS location recording. Allows zero distractors in phase 2 of DMTS (for training). Bug fixed in DMTS: correction procedure failed to wait for finger to be removed from touchscreen (so always registered incorrect touch). Now it requires finger to be removed after end of phase 2 (but will still start correction procedure, i.e. phase 3, if subject removes finger and retouches before start of correction procedure). SWM allows reward to be delivered after every correct touch (not just on completion of trial). Bug fix, MCSRTT: if lever was used as centring response, background touches were treated as if the lever had been released, even with "Ignore other responses..." ticked. Task alteration, MCSRTT: "absent stimulus" markers appear right from the start of the trial, not just at the end of Phase 1. Bug fix, DMTS: correction procedure for simultaneous presentation didn't remove inert target copy from the screen during punishment. Task modification, DMTS: new colour-variation scheme (Seven Colours Per Trial) to mimic Weed et al. (1999) *Cognitive Brain Research* 8: 185.
- Version 0.99 (24 Oct - 16 Nov 2004). ET_REV_OMISSION and ET_VD_OMISSION events added. VDS/VDP, EDSHIFT stage: text instance of "B5A6(+)B6A6(-)" should have read "B5A6(+)B6A5(-)" (no execution bug, just descriptive). Database changed: MonkeyCantab_IndividualEvents.Box now long int, not byte (allows relationship from MonkeyCantab_GeneralConfig); CRecordSetIndividualEvents also changed; tested with database still using BYTE format: works. Comma-delimited output for MCSRT misaligned (no

comma between Phase1Given and Phase1Responded in header). MCSRT params shrunk to fit an 800x600 screen. MCSRT bug fix: if premature touches were not punished by ending the trial, movements across/within a premature stimulus were counted incorrectly as multiple premature responses. MCSRT bug fix: "ignore responses during Phase 1" flag partially backwards. DMTS task has draw-without-replacement option for Phase 1 and Phase 2 Target locations (to give pseudorandom rather than random location assignment). DMTS: phase 1 location selection when adjusting for a central feeder was not as advertised (it just alternated between middle-left/ middle-right positions, ignoring user preference); fixed. DMTS: explicit recording of Phase 1 and Phase 2 Target locations in database.

- Version 1.0 (17 Nov 2004). Up to 7 distractors in DMTS (but note: since the automatic stimulus-varying procedures available to DMTS generate stimuli in groups of 4, the stimuli generated by these procedures with other total numbers of stimuli per trial are not guaranteed to follow the same rules as with 4 total stimuli per trial). Draw-without-replacement system for MCSRT target location. Response location recorded for DMTS.
- Version 1.1 (9 Dec 2004). Improved bitmap centring function. Reinforcement Familiarization: background colour now works; bug fixes ("maximum no. reinfs" didn't work with the lick-contingent FR1 option, and spurious fall-though to main schedule caused by ==/= error). MCSRT: option to measure Phase 2 maximum response time from the end (rather than the start) of the stimulus; bugfix (distractor code erroneously removed a location from viNontargetLocations - so the final non-target pulled from an empty vector).
- Version 1.2 (26 Jan 2005). DMTS: option to draw levels without replacement. MCSRT: option to draw without replacement for (1) pre-stimulus delay; (2) stimulus duration. Option to have trials initiated by a lever/switch response, for use with dogs (see Use with dogs).
- Version 1.3 (10 Feb 2005). Bugfix to DMTS level DWOR.
- Version 1.4 (22 Mar - 3 Apr 2005). New DMTS stimulus generation methods to improve performance with >4 stimuli per trial; option to rotate Phase 2 stimuli in DMTS; option to jumble variant order; bugfix for simultaneous option. Attempt at bugfix to MCSRT - lever reported to be responsive at times that it shouldn't have been. Option to allow mouse input as well as touchscreen input.
- Version 1.5 (21 Apr - 4 May 2005). Cosmetic/convenience change to ID/ED configuration dialogue. Changes to SimpleSchedules (relating to what was displayed during reinforcement timeouts or during reinforcer-device-busy states.
- Version 1.6 (18 May 2005). When levers are used to initiate trials, the lever response now terminates all ongoing sounds, notably the Marker 1 sound.
- Version 1.7 (30 May 2005, 4 Sep 2005). 3, not 2, decimal places in DMTS/MCSRT dialogue boxes displaying floating-point values. Lick latency in Touch Training task.
- Version 1.8 (21 Sep 2005). ReinfFamil was inappropriately proceeding out of "totally free juice" to FR1.
- Version 1.9 (10 Oct 2005). Visual discrimination tasks had a bug in the Harsh correction procedure, such that no stimuli were shown on correction trials.
- Version 1.91 (30 Oct 2005). MonkeyCantab_GeneralConfig.ModuleList changed from String (255) to Memo to accommodate very long module lists.
- Version 2.0 (7 Jan 2006). Clock on main display. // End-of-task summaries on screen, from Reversals, VDP, and VDS. // Support for extra "debug" views showing touches, with WhiskerServer v2.12.1 and higher. // Ability to specify stimulus locations manually in MCSRT. // Marker 2 sound option in MCSRT. // Option for alternative targets within a single test in MCSRT. // Background rectangles added to Camcog ID/ED stimuli.
- Version 2.1 (10 Jan 2006). Option to leave stimuli up during reward in TouchTraining. // Option to punish ITI touches in TouchTraining. // "Copy module" button.
- Version 2.2 (11 Jan 2006). Option to punish, rather than reward, touches during TouchTraining. (A rather specialized option!)
- Version 2.3 (13 Feb 2006). Option to deliver free rewards manually.
- Version 2.4 (16 Feb 2006). Command-line automated execution.
- Version 2.5 (21 Feb - 2 March 2006). Option to repeat entire DMTS trial following errors (either at phase 1 or phase 2) until the subject gets the trial right.
- Version 2.6 (8 Apr 2006). Bug fixed that could cause delayed non-matching to lock up ("waiting

for finger release before starting correction procedure") under some circumstances (responding incorrectly in phase 2 and releasing finger while phase 2 incorrect-target stimulus still on screen). // Option to make S+/S- stimulus selection wholly predictable in DMTS.

- Version 2.7 (29 Apr 2006). DMTS problem described above still not fixed - may be a hardware fault. As a workaround, implemented an optional feature: "wait for finger release before proceeding but never for more than X seconds (user can specify X)".
- Version 2.8 (24 May 2006). Now autoupdates data file name on first entering subject ID. // Improved error message if data file already exists.
- Version 2.9 (July-August 2006). Improvements to DMTS in terms of ability to vary the number of distractor/nontarget objects within a session.
- Version 3.0 (9 Sep 2006). Option improvements to reversal learning (limiting the number of reversals possible; improved draw-without-replacement method for spatial randomization).
- Version 3.1 (22-28 Dec 2006). In TouchTraining, the stimulus eventually moved when it was the final size, regardless of whether the "Move" option was ticked; this bug has been fixed. // An option has been added to query the size of a stimulus (in units, and as proportions of the active screen size). // "Strict touches" option for all tasks (though mandatory for SimpleSchedules). // Time since last subject response is shown on the main display. // Separate specification of maximum response times for phase 1 and phase 2 in DMTS.
- Version 3.2 (22-23 Jan 2007). New monochrome option (monochrome, shape-only discrimination, fixed colour, i.e. colour fixed by the experimenter) for DMTS.
- Version 3.3 (27 Jan 2007). Bug fixed in v3.2: monochrome option didn't work properly with shape shuffling.
- Version 3.4 (8 March 2007). Easier compilation for users. Option to stop ID/ED tasks after a criterion has been met (not just to increase the stage).
- Version 3.5 (30 March 2007). "Side task" (location discrimination) option in Reversal Learning task.
- Version 3.6 (21 May 2007). Bug fixed in Spatial Working Memory (SWM or SOSS) task: sometimes, double touches to a stimulus were erroneously ignored because the system failed to notice all finger removals. Fixed.
- Version 3.7 (24 July 2007). Minor bugfix: SimpleSchedules miscalculated the cosmetic version of its end time.
- Version 3.8 (18 Aug 2007). Made the "assume finger released but the touchscreen missed it" setting easier to configure for SimpleSchedules (for which it is mandatory), and added explicit recording of this event.
- Version 3.9 (14 Sep 2007). Bugfix to ensure no attempt is made to play zero-length sounds (they may have created an undesirable "pop"). Also, option to disable houselight entirely.
- Version 4.0 (from 25 Oct 2007). Delayed reinforcement ("impulsive choice") task added.
- Version 4.1 (21 Dec 2007). Bug fix in SimpleSchedules - "assume finger removed" was not always detected correctly.
- Version 4.2 (26 Dec 2007). Two-choice option for multiple-choice serial reaction time task.
- Version 4.3 (7 March 2008). Visual stimulus options for reward/punishment.
- Version 4.4 (from 30 Apr 2008). Pellet collection latency via the MAGAZINE_DOOR device. Some additional reward collection latency measures. Minor bugfix in text file out from SWM (see version tracker).
- Version 4.5 (11 May 2008). MAGAZINE_LAMP device.
- Version 4.6 (24 Sep 2008). Enhanced options for leaving stimuli on screen after choice in Reversals, VisualDiscrimPredefined, VisualDiscrimSuperimposed.
- Version 4.7 (19 Oct 2008). Option to specify "false feedback" trials more precisely (pseudorandomly, not randomly) in Reversals.
- Version 5.0 (12 Jan 2009). (1) Server default now "localhost", not "loopback" (for Windows Vista compatibility and more general standardization). (2) Touch events no longer requested for nontouchable object components. (The check that at least one component is touchable remains as a warning in the "Configure visual objects" dialogue.)
- Version 5.1 (26 Apr 2009). Recompilation on Visual Studio 2008, with a couple of very minor bug fixes (see version tracker). Also, change of random number generator (to use the Boost libraries). There was a previous bug requiring, we think (1) MonkeyCantab to use its previous

Mersenne Twister library; (2) it to be compiled with Visual C++ 6.0; (3) particular sound card drivers to be active. We can only guess that this was some fault in low-level floating-point code: on about 14 per 100,000,000 iterations, there were errors in converting float to int values, but only when multiple sound cards were being accessed (and the compiler settings as above). The bug manifested itself as a silly number coming back from the random number generator, and MonkeyCantab seeming to wait for ever.

- Version 5.2 (8 May 2009). Further work on random numbers versus funny sound cards. Further workaround installed; see version tracker. Also, XML format for CamcogQuadPattern changed for compatibility with other XML parsers; see version tracker.
- Version 5.3 (11 May 2009). Option for darkness to accompany reward.
- Version 5.4 (11 May 2009-8 June 2009). RVIP task.
- Version 5.5 (26 June 2009). Non-target option in five-choice task.
- Version 5.6 (27 June 2009-19 July 2009). List-length DMTS task. Also, changed memory delay timing for DMTS in the specific case when Phase 1 responding is rewarded: the memory delay used to begin at the end of the reward, and now it begins at the start of the reward (with an extra check that the memory delays must all exceed the maximum reward time). Also, an off-by-one error in DMTS (in the "avoid stimuli used in the last N trials" option) fixed. And DMTS modification: memory delay now begins at *start* (not end) of phase 1 reward, if that is used (not that phase 1 rewards are a very sensible option!). See version tracker for full details.

1.2 Required devices

The program requires to claim devices in groups named **box0, box1, box2...** with device names as listed below in bold. (Note that the line numbers themselves are arbitrary and depend on how your apparatus is wired up.)

```
# ---------------- Box 0 definition
# INPUTS
# Lick sensors are used with peristaltic pump reinforcement (e.g. with
marmosets).
# A levers is used for the rhesus version of the five-choice task, and for many
tasks in dog testing apparatus.
# The "magazine door" sensor detects entry to the magazine into which pellets are
delivered,
# and so is used for pellet reward collection latency (added May 2008).

line     0      box0    LICKSENSOR
line     3      box0    LEVER
line     6      box0    MAGAZINE_DOOR

# OUTPUTS
# The houselight is on during the tasks.
# Peristaltic pumps are used in marmoset testing.
# "Pump" delivers nice juice; "Pump2" delivers punishment (e.g. saline).
# Pellet dispensers are used in rhesus testing.
# The "extra reward device" might be a tone generator installed in the box, to be
associated with reward.
# The "extra punishment device" can be activated when the subject is punished.
# The lever control line is intended for use with retractable levers (see
"Lever", above).
# Two other lines are used by Porton Down for interfacing to other hardware:
#   REINFORCEMENT_INFO delivers 10ms high pulses upon reward and 20ms high pulses
on punishment.
#   STIMULUS_INFO is high when stimuli are being presented.
# MAGAZINE_LAMP is a light inside the pellet magazine.

line    24      box0    HOUSELIGHT
```

```
line    27      box0    PUMP
line    30      box0    PELLET
line    33      box0    EXTRAREWARDDEVICE
line    36      box0    EXTRAPUNISHMENTDEVICE
line    39      box0    LEVERCONTROL
line    42      box0    PUMP2
line    45      box0    REINFORCEMENT_INFO
line    48      box0    STIMULUS_INFO
line    51      box0    MAGAZINE_LAMP

# DISPLAY
# This is the monitor with a touchscreen attached to it.

display  0      box0    SCREEN

# AUDIO
# This is one of the computer's sound device (typically connected to a
loudspeaker in the box).
# Used to reward successful monkeys with Mozart.

audio    0      box0    SOUND

# ... and so on for other boxes
```

Please ensure that these devices are available and listed in the device definition file in use by the server. (The snippet above shows an extract from a typical definition file.) **If you do not have a particular device, and do not need it, then you should configure "fake" lines on the Whisker server, and assign the non-existent device to a fake line.** For example, marmoset testing boxes might not have a pellet dispenser or a lever, while rhesus testing boxes might not have a pump.

1.3 Configuring Whisker for particular hardware

- Cambridge Cognition legacy hardware
- Cambridge Cognition 2004 hardware

1.3.1 Cambridge Cognition legacy hardware

Computers supplied by Cambridge Cognition to run their DOS-based Monkey CANTAB product have the following characteristics:

- one ICS input/output ISA card at 0x280 [to 0x283]. It should be configured for **reversed inputs** and **reversed outputs;** port A is for input; ports B and C are for output.
- one display
- one touchscreen
- one audio card
- no fake lines or failsafes
- a device definition file as follows:

INSERT DEVICE DEFINITION FILE HERE AS SUPPLIED BY CAMCOG

There is a **known problem** with this hardware. When an ICS card is initialized, it turns all its outputs off. There is no way to avoid this; it's a feature of the 82C55 controller chip used. Since CamCog wire up their boxes with reversed outputs (i.e. when the card thinks something's off, it's

actually on, and vice versa), initializing the card actually turns all the outputs *on*. WhiskerServer fixes this problem as quickly as it can, so that the outputs are on for no longer than a few microseconds, but devices that are sensitive to such short 'blips' may trigger. Some users have noticed this in that **a pellet is delivered** when WhiskerServer first starts. There are no problems subsequently, when WhiskerServer is running. The only way to fix this would be to rewire the boxes so that outputs are not "reversed".

1.3.2 Cambridge Cognition 2004 hardware

Computers supplied by Cambridge Cognition to run Whisker and MonkeyCantab for Whisker have the following characteristics:

- one Advantech PCI digital I/O card with **reversed inputs**
- three displays (one for the experimenter, two for subject testing) with a Matrox multimonitor card
- two touchscreens
- two audio cards
- no fake lines or failsafes
- a device definition file as follows:

Sample device definition file #1 - two-box MonkeyCantab system

```
WhiskerServer v2.0 - DEVICE DEFINITION FILE - DO NOT ALTER THIS LINE
##############################################################################

# This file defines device names used by the WhiskerServer program.
# Lines beginning with a hash (#) are comments and are ignored.
#
# Each line takes the following format:
#
#       <device_type> <device_number> <group_name> <device_name>
#
# where <device_type> may be
#       line            = digital I/O line
#       display         = display device (monitor)
#       audio           = audio device (sound card, or half-sound card; see
manual)
#
# The <device_number> is the number of the line/display/audio device that you
see
# on the server's console - the number that you would otherwise claim.
#
# The COMBINATION of the <group_name> and <device_name> must be unique.
# If the server encounters non-unique device group/name pairs in this file,
# all but the first will be ignored.
# Neither the <group_name> nor the <device_name> may start with a number.

##############################################################################

# File for double Monkey CANTAB test system

# ---------------- Box 0 definition

# INPUTS

line     0    box0    LEVER
line     1    box0    LICKSENSOR
line     2    box0    EXTRAREWARDDEVICE_INPUT
```

```
line     3    box0    MAGAZINE_DOOR
line     4    box0    FEEDER_REPORT
line     5    box0    PUMP2_FEEDER2_REPORT
line     6    box0    LEVERHOME
line     7    box0    SPARE_INPUT_BOX0

line    16    box1    LEVER
line    17    box1    LICKSENSOR
line    18    box1    EXTRAREWARDDEVICE_INPUT
line    19    box1    MAGAZINE_DOOR
line    20    box1    FEEDER_REPORT
line    21    box1    PUMP2_FEEDER2_REPORT
line    22    box1    LEVERHOME
line    23    box1    SPARE_INPUT_BOX1

# OUTPUTS

line    8     box0    HOUSELIGHT
line    9     box0    PUMP
line    10    box0    EXTRAREWARDDEVICE
line    11    box0    MAGAZINE_LAMP
line    12    box0    PELLET
line    13    box0    PUMP2
line    14    box0    LEVERCONTROL
line    15    box0    EXTRAPUNISHMENTDEVICE
line    42    box0    STIMULUS_INFO
line    43    box0    REINFORCEMENT_INFO

line    24    box1    HOUSELIGHT
line    25    box1    PUMP
line    26    box1    EXTRAREWARDDEVICE
line    27    box1    MAGAZINE_LAMP
line    28    box1    PELLET
line    29    box1    PUMP2
line    30    box1    LEVERCONTROL
line    31    box1    EXTRAPUNISHMENTDEVICE
line    44    box1    STIMULUS_INFO
line    45    box1    REINFORCEMENT_INFO

# DISPLAY

display  0    box0    SCREEN
display  1    box1    SCREEN

# AUDIO

audio    0    box0    SOUND
audio    1    box1    SOUND

# SAFETY RELAYS

line    40    R1
line    41    R2
```

Sample device definition file #2 - four-box MonkeyCantab system

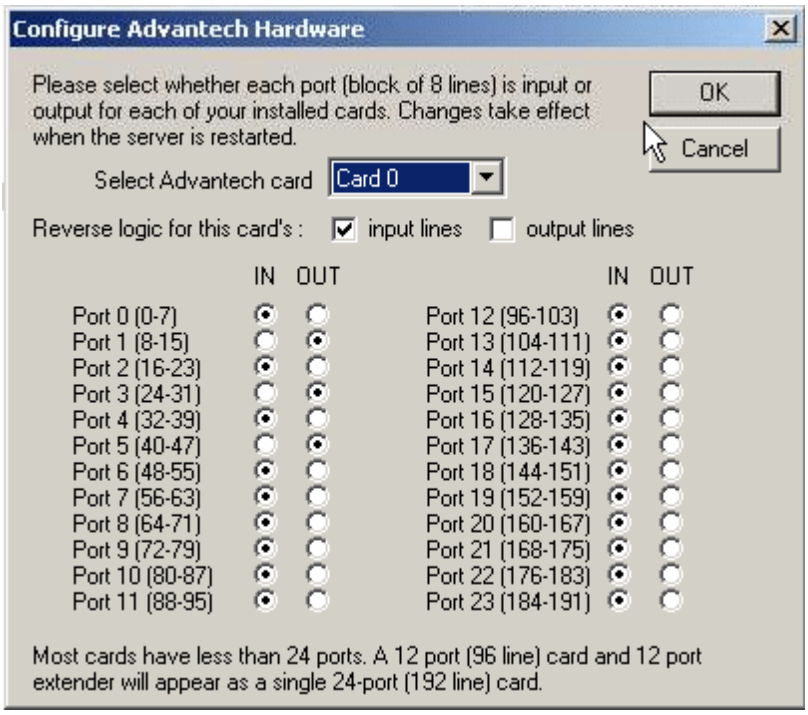
```
WhiskerServer v2.0 - DEVICE DEFINITION FILE - DO NOT ALTER THIS LINE
##########################################################################

# This file defines device names used by the WhiskerServer program.
#lines beginning with a hash (#) are comments and are ignored.
#
# Eachline takes the following format:
```

```
#
#    <device_type> <device_number> <group_name> <device_name>
#
# where <device_type> may be
#    line    = digital I/Oline
#    display    = display device (monitor)
#    audio    = audio device (sound card, or half-sound card; see manual)
#
# The <device_number> is the number of theline/display/audio device that you
see
# on the server's console - the number that you would otherwise claim.
#
# The COMBINATION of the <group_name> and <device_name> must be unique.
# If the server encounters non-unique device group/name pairs in this file,
# all but the first will be ignored.
# Neither the <group_name> nor the <device_name> may start with a number.
#
# CREATED ON: 29-September-2004
# BY: Simon Gow
################################################################################

# File for 4-Box Monkey Cantab boxes

# ---------------- Box 0 & 1 definition

# INPUTS
line     0    box0    LEVER
line     1    box0    LICKSENSOR
line     2    box0    EXTRAREWARDDEVICE_INPUT
line     3    box0    MAGAZINE_DOOR
line     4    box0    FEEDER_REPORT
line     5    box0    PUMP2_FEEDER2_REPORT
line     6    box0    LEVERHOME
line     7    box0    SPARE_INPUT_BOX0
line    16    box1    LEVER
line    17    box1    LICKSENSOR
line    18    box1    EXTRAREWARDDEVICE_INPUT
line    19    box1    MAGAZINE_DOOR
line    20    box1    FEEDER_REPORT
line    21    box1    PUMP2_FEEDER2_REPORT
line    22    box1    LEVERHOME
line    23    box1    SPARE_INPUT_BOX1

 # OUTPUTS
line     8    box0    HOUSELIGHT
line     9    box0    PUMP
line    10    box0    EXTRAREWARDDEVICE
line    11    box0    MAGAZINE_LAMP
line    12    box0    PELLET
line    13    box0    PUMP2_FEEDER_2
line    14    box0    LEVERCONTROL
line    15    box0    EXTRAPUNISHMENTDEVICE
line    72    box0    STIMULUS_INFO
line    73    box0    REINFORCEMENT_INFO
line    24    box1    HOUSELIGHT
line    25    box1    PUMP
line    26    box1    EXTRAREWARDDEVICE
line    27    box1    MAGAZINE_LAMP
line    28    box1    PELLET
line    29    box1    PUMP2_FEEDER_2
line    30    box1    LEVERCONTROL
line    31    box1    EXTRAPUNISHMENTDEVICE
line    74    box1    STIMULUS_INFO
line    75    box1    REINFORCEMENT_INFO
```

```
# DISPLAY
display  0    box0    SCREEN
display  1    box1    SCREEN
display  2    box2    SCREEN
display  3    box3    SCREEN

# AUDIO
audio    0    box0    SOUND
audio    1    box1    SOUND
audio    2    box2    SOUND
audio    3    box3    SOUND

# SAFETY RELAYS
line    64    R1    SAFETY_RELAY1
line    65    R2    SAFETY_RELAY2

# ---------------- Box 2 & 3 definition
# INPUTS
line    32    box2    LEVER
line    33    box2    LICKSENSOR
line    34    box2    EXTRAREWARDDEVICE_INPUT
line    35    box2    MAGAZINE_DOOR
line    36    box2    FEEDER_REPORT
line    37    box2    PUMP2_FEEDER2_REPORT
line    38    box2    LEVERHOME
line    39    box2    SPARE_INPUT_BOX0
line    48    box3    LEVER
line    49    box3    LICKSENSOR
line    50    box3    EXTRAREWARDDEVICE_INPUT
line    51    box3    MAGAZINE_DOOR
line    52    box3    FEEDER_REPORT
line    53    box3    PUMP2_FEEDER2_REPORT
line    54    box3    LEVERHOME
line    55    box3    SPARE_INPUT_BOX1

# OUTPUTS
line    40    box2    HOUSELIGHT
line    41    box2    PUMP
line    42    box2    EXTRAREWARDDEVICE
line    43    box2    MAGAZINE_LAMP
line    44    box2    PELLET
line    45    box2    PUMP2_FEEDER_2
line    46    box2    LEVERCONTROL
line    47    box2    EXTRAPUNISHMENTDEVICE
line    76    box2    STIMULUS_INFO
line    77    box2    REINFORCEMENT_INFO
line    56    box3    HOUSELIGHT
line    57    box3    PUMP
line    58    box3    EXTRAREWARDDEVICE
line    59    box3    MAGAZINE_LAMP
line    60    box3    PELLET
line    61    box3    PUMP2_FEEDER_2
line    62    box3    LEVERCONTROL
line    63    box3    EXTRAPUNISHMENTDEVICE
line    78    box3    STIMULUS_INFO
line    79    box3    REINFORCEMENT_INFO


# ---------------- Box User Inputs
# INPUTS
line    80    box0    USER_INPUT_1
line    81    box0    USER_INPUT_2
line    82    box1    USER_INPUT_1
```

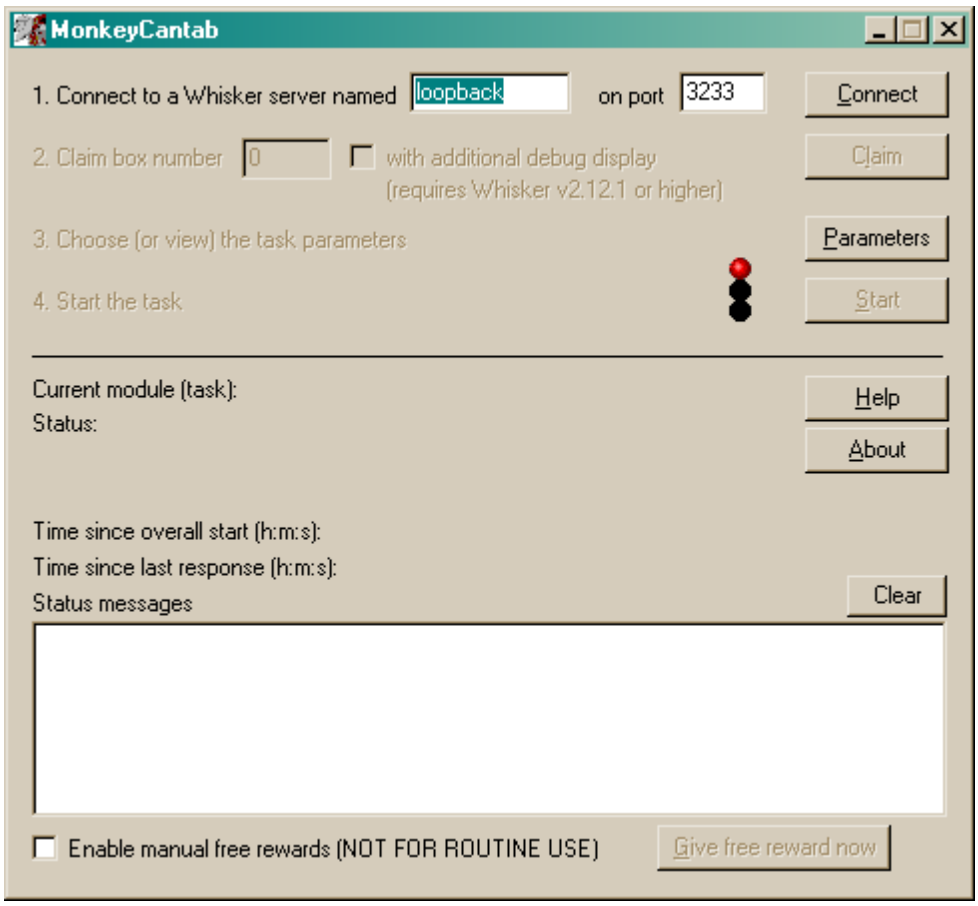
```
line     83     box1     USER_INPUT_2
line     84     box2     USER_INPUT_1
line     85     box2     USER_INPUT_2
line     86     box3     USER_INPUT_1
line     87     box3     USER_INPUT_2
```

Configure the Advantech card like this:



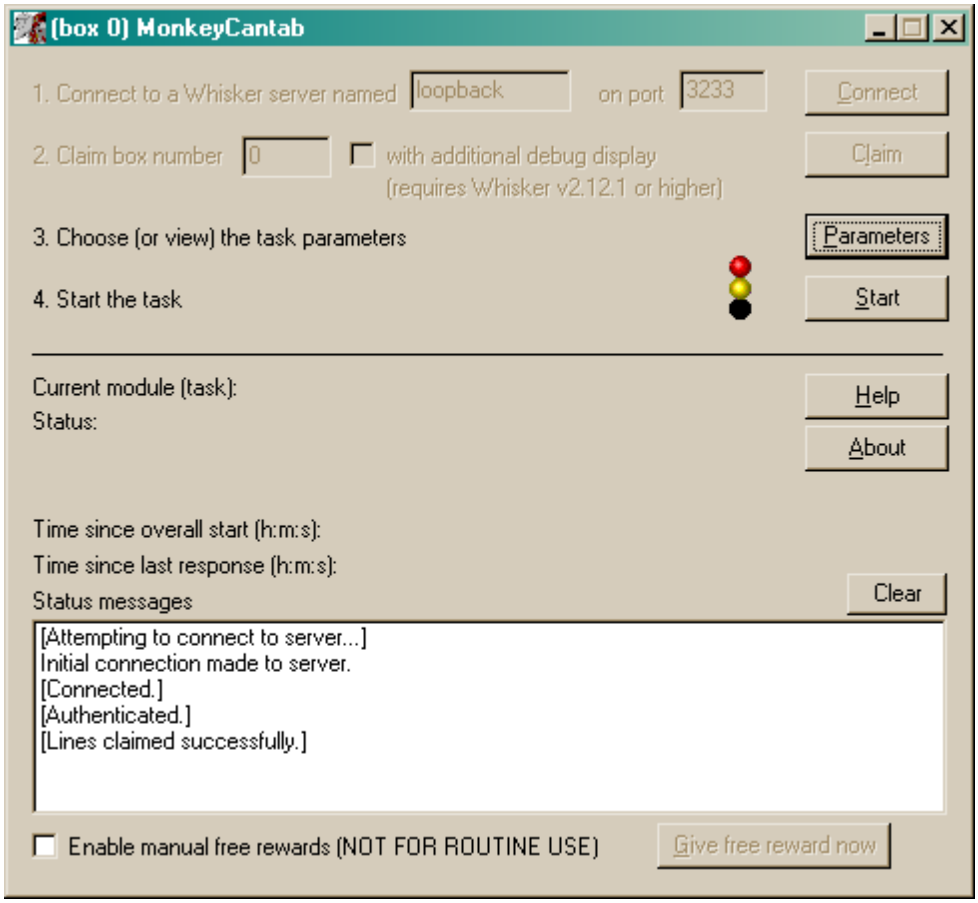
1.4 Using MonkeyCantab

When you run the task, the main screen looks as follows:



You must connect to a Whisker server, claim an operant chamber (box), and set up the parameters for your tasks. Configuration is explained on a separate page. Once that's done, the traffic lights will turn amber.

With **MonkeyCantab v2.0** and higher, a further option is presented: to claim a box **with additional debug display (requires Whisker server v2.12.1 or higher).** If you tick this option before claiming the box, an additional window will open on the server computer as you claim the box. This window shows a copy of what the subject sees. You can also see what the subject is seeing by browsing the WhiskerServer console, but this option allows you to run multiple copies of MonkeyCantab (e.g. with 6 touchscreen-equipped boxes) and see copies of all your subjects' screens and their touch responses simultaneously in extra windows. *If you tick this option and are not using WhiskerServer v2.12.1 or higher, an error message will appear, though MonkeyCantab will still operate normally.*

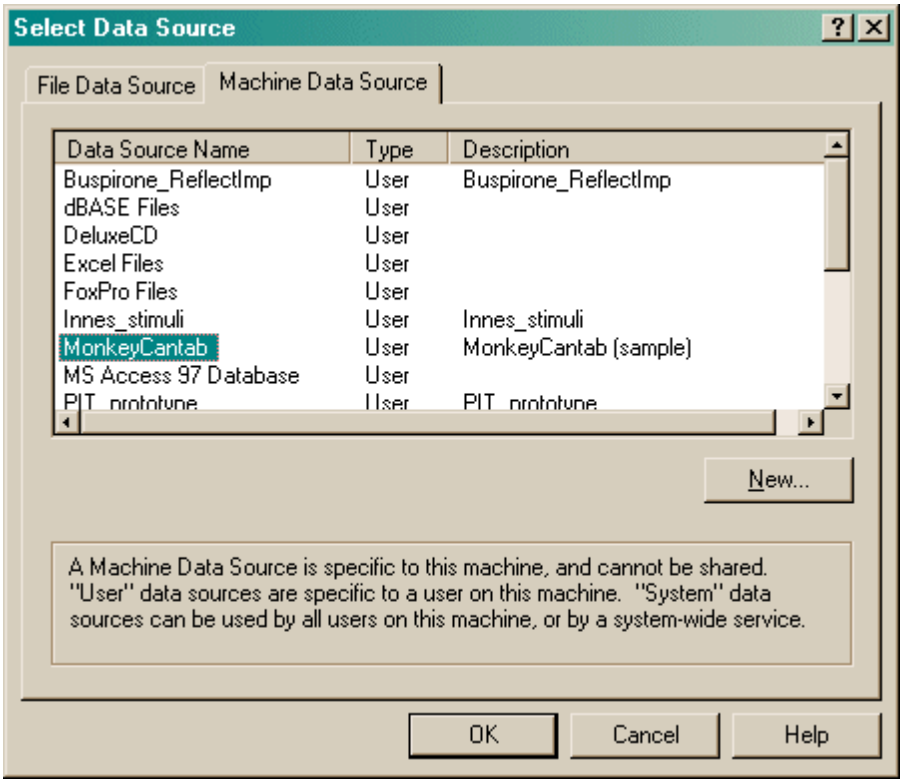


When you are ready, press *Start* to begin MonkeyCantab.

The traffic lights will turn green and MonkeyCantab will then work through all the modules (tasks) in your task list.

While the task is running, from **MonkeyCantab v2.3** and higher, a tick box offers you the opportunity to enable **manual delivery of free rewards.** When this box is ticked, clicking the button marked "Give free reward now" will deliver a reward immediately to the subject. The nature of that reward is defined in the General parameters. Note that this option may well interfere with the contingencies of the task in progress; its use is not generally recommended.

When the task finishes, it saves data to disk and pops up a new dialogue box for you to select a database to store the data to. (The data sources are configured under *Control Panel → ODBC*.) If you previously specified an ODBC data source in the parameters, that data source is used automatically and you will only see a dialogue box if something goes wrong and the program needs your input.



Your data will be saved and MonkeyCantab has then finished.

From MonkeyCantab v2.4, it is also possible to automate execution by running MonkeyCantab from the command line.

1.5 Automatic command-line execution

MonkeyCantab v2.4 and above supports command-line execution.

To execute commands from the command-line, you will need to run a **Command Prompt**. Windows has one under *Start / Programs / Accessories / Command Prompt*. Alternatively, you can click *Start / Run* and type `cmd` into the "Run" box. You will see a prompt such as

```
Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-2000 Microsoft Corp.

C:\Documents and Settings\Rudolf>
```

I'll show things from the computer in blue, and things that you type in red. Suppose you have installed MonkeyCantab into C:\Program Files\MonkeyCantab, as is common. You could then type

```
"c:\program files\monkeycantab\monkeycantab" -?
```

for syntax help. **(The quotes are necessary because "program files" has a space in it.)** This would produce something like the following:

```
MonkeyCantab v2.4 - DEBUG build compiled on Feb 16 2006 at 12:57:24
Usage:
  monkeycantab [-?] [-help] [-autoexecute] [-server SERVER] [-port
PORT] [-box BOX] [-debugdisplay] [-config CONFIG]
```

```
Options:
                  When run with no options, starts MonkeyCantab in GUI
mode
  -?              Prints this syntax message
  -help           Prints this syntax message
  -autoexecute    Starts MonkeyCantab in auto-execution mode.
                  This requires you to specify a configuration file.
  -server SERVER  Specifies a Whisker server to use (default
"localhost").
  -port PORT      Specifies an IP port to use (default 3233).
  -box BOX        Specifies a box (operant chamber) to use (default
0).
  -debugdisplay   Creates a debugging display (default is not to do
this).
  -config CONFIG  Specifies a configuration file to load.

Example:
  monkeycantab -autoexecute -box 1 -config "d:\my
configs\subject1_DMTS.xml"
```

So, if you wanted to run a MonkeyCantab session from a pre-saved configuration file, **(1) ensure that WhiskerServer is running,** and **(2)** type a command such as

```
"c:\program files\monkeycantab\monkeycantab" -autoexecute -box 1 -
config "d:\my configs\subject1_DMTS.xml"
```

Note the use of quotes to surround all filenames with spaces in them. This will run MonkeyCantab, connect to the default Whisker server ("loopback", or the same computer that MonkeyCantab is running on) with the default IP port (3233), and then try to claim box 1 and run a session based on the configuration file named "d:\my configs\subject1_DMTS.xml".

Log files for the session will be stored in the current working directory.

To run multiple consecutive sessions for the same subject, using different configuration files (e.g. for different reward parameters), you could create a **batch file**. Suppose you use your favourite text editor to create a batch file called **Subject7SWM_IDEDRun.BAT** that contains the following text:

```
@echo off
REM the "echo off" command is optional, but has the effect of
preventing remarks and commands from being shown on the screen
REM The initial "@" suppresses output from the first "echo off"
command
REM "echo" allows you to print things to the screen
REM As you may have noticed, all lines starting with "REM(ark)" are
ignored!

echo About to run subject 7 in box 0, first session (spatial working
memory task)...
REM we want the output files to go somewhere sensible, so we change
directory to that place first
cd "d:\my output files"
REM OK, now we run MonkeyCantab
"c:\program files\monkeycantab\monkeycantab" -autoexecute -box 0 -
config "d:\my configs\subject7_SWM.xml"

echo About to run subject 7 in box 0, second session (ID/ED task)...
```

```
"c:\program files\monkeycantab\monkeycantab" -autoexecute -box 0 -
config "d:\my configs\subject7_IDED.xml"

echo Finished.
```

As long as WhiskerServer is running, you could then type

```
Subject7SWM_IDEDRun
```

from your command prompt and the batch file would execute, running two MonkeyCantab sessions in sequence. The command-line window would show:

```
About to run subject 7 in box 0, first session (spatial working memory
task)...
```
(pause while MonkeyCantab runs in a separate window)
```
About to run subject 7 in box 0, second session (ID/ED task)...
```
(pause while MonkeyCantab runs in a separate window, again)
```
Finished.
```

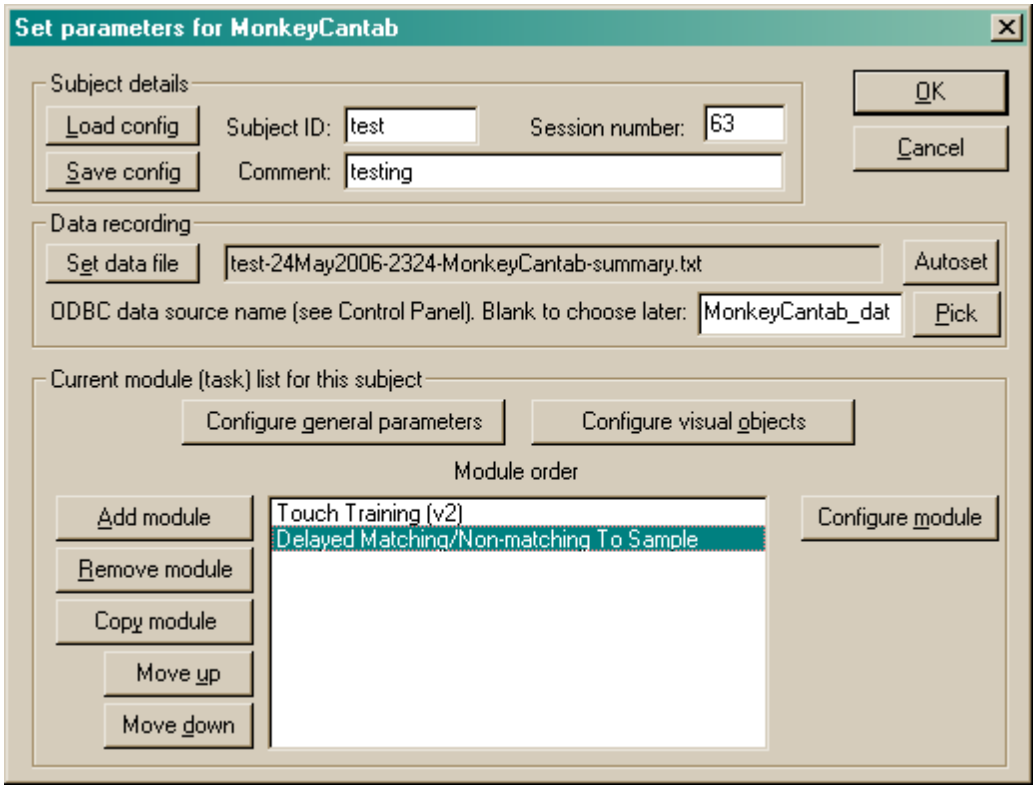
If need be, you can press **CTRL-C** or **CTRL-BREAK** to interrupt batch files.

Technical note: *To allow this to work on as many operating systems as possible, this facility is implemented with a small program called MonkeyCantab.COM, which provides syntax help, checks syntax, and passes on the message to MonkeyCantab.EXE, which it expects to be in the same directory as itself. The .EXE file is the main graphical user interface (GUI) program and behavioural task suite. This is necessary because the GUI program may not have a command line to provide syntax help to. When you type* ***monkeycantab*** *at the command line, Windows prefers to execute the .COM version, rather than the .EXE.*

1.6 Configuring MonkeyCantab (general settings)

To configure MonkeyCantab, choose **Parameters** from the main screen.

MonkeyCantab allows you to run a selection of tasks. You must therefore tell MonkeyCantab which tasks you want to run, and in what order.

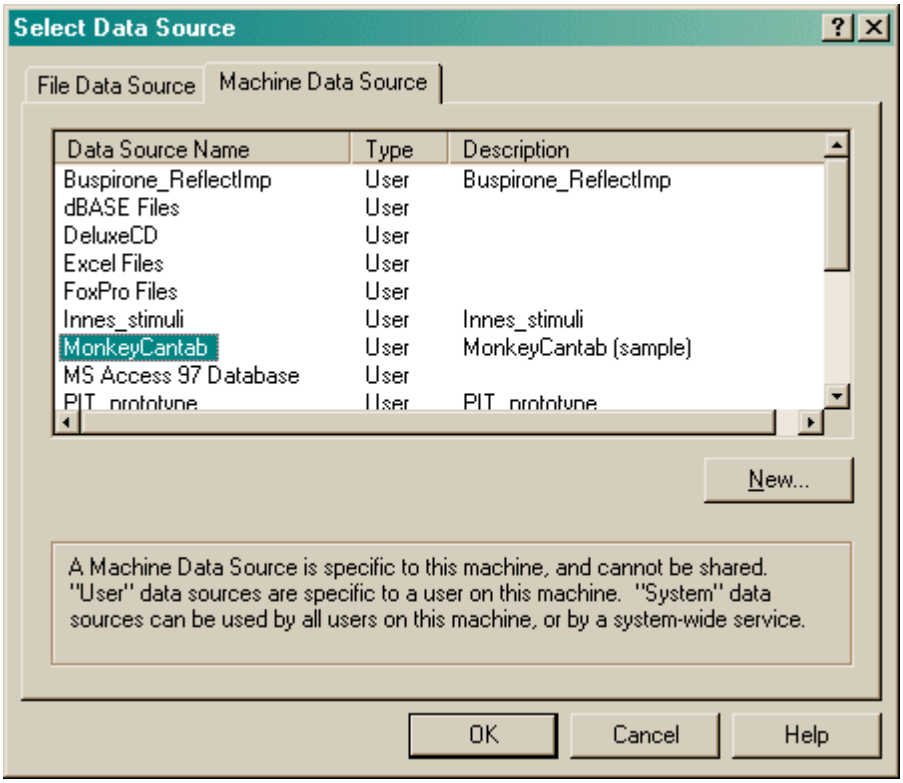


You may **load** a configuration file or **save** it again. (If you load a configuration file, alter the settings, and click OK, your changes will be saved automatically for next time.) Loading may take a few seconds if the configuration files contain many stimuli, such as the supplied ID/ED task stimuli.

You may set any of the subject details **(ID, session number, comment)** by typing in the relevant boxes. You may **set a data file** if you choose; if you load a configuration file, the program will choose a default data file for you. (Similarly, if you type in a subject name from scratch, and no data file name exists, the program will guess one for you; if you wish to regenerate a new filename based on the current subject name and date/time, click **Autoset**.) This data file contains a textual summary of your results. The full result set is saved in a database via the ODBC (Open Database Connectivity) protocol. You may select the database at this point, or when the program finishes running.

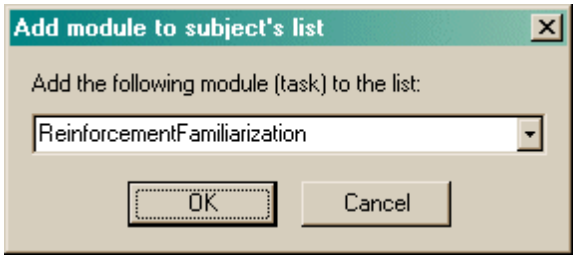
A sample configuration is supplied with MonkeyCantab. It's called "MonkeyCantab_TestConfig_And_SampleObjectLibrary.xml" and it lives in the directory you installed MonkeyCantab into (typically C:\Program Files\MonkeyCantab).

To pick an ODBC database *in advance* of finishing, click **Pick** and you will be offered the ODBC Data Source picker (below). Your choice will be recorded and will apply to this subject from now on (or until you specify a different source).



If you don't specify an ODBC data source now, or you delete the value in the **ODBC data source name** box, you'll be asked to choose when the task ends (and that choice will only apply to the session in progress). To find out how to **create a new ODBC source, click here.**

The bottom half of the screen contains the module (task) list for the subject. You may specify any number of tasks from those that MonkeyCantab provides. They may be executed in any order. Click **Add Module** to add a new module to the end of the list. Click a module in the list to select it, and click **Remove Module** to or remove that module from the list. Click **Move Up** or **Move Down** to move a module up/down the list.



Click **Configure general parameters** to set up parameters that apply to all tasks.

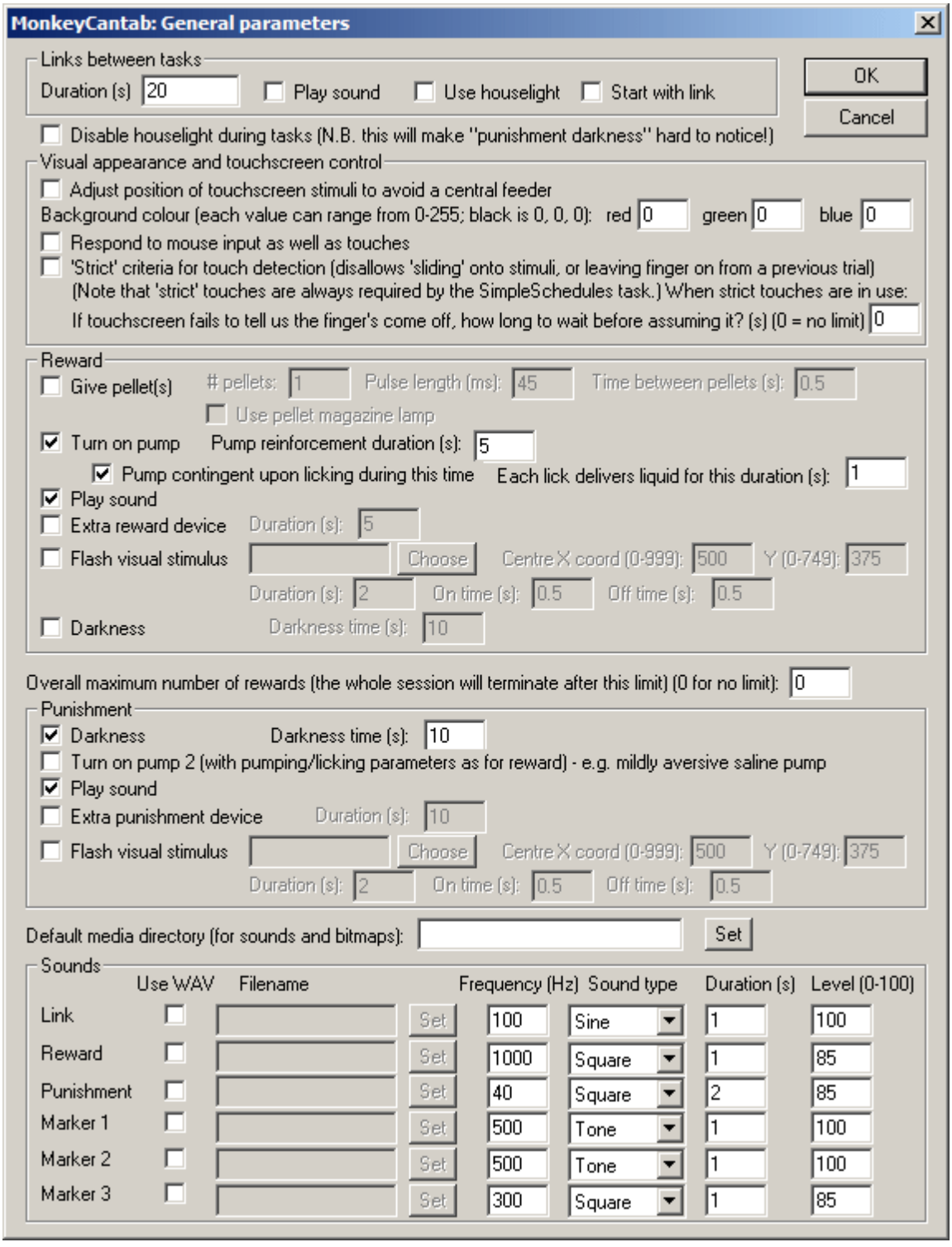
Click **Configure visual objects** to set up objects used by the tasks.

Select a module from the list and click **Configure module** to set up parameters for the chosen module. Parameters that are specific to each task are explained with each task's description (click on the links below). The module names are:

- Reinforcement Familiarization
- Touch Training
- Visual Discriminations and Set-Shifting (Predefined Style)
- Visual Discriminations and Set-Shifting (Superimposed Style)

- Reversal Learning
- Delayed Matching/Non-matching To Sample
- Spatial Working Memory, the casual name for the Self-Ordered Search Task
- Multiple-Choice Serial Reaction Time
- Paired-Associates Learning
- Simple Schedules of Reinforcement

1.6.1 General parameters



Links between tasks

- **Duration (s).** The duration of the link between tasks.
- **Play sound?** If you select this option, the Link sound will be played at the start of the link (see *Sounds* below).
- **Houselight on?** By default, the houselight is on during tasks but switched off during linking periods. Choose this option to keep it on during the links.
- **Start with link?** If you choose this option, the session will start with a link before any tasks begin. This allows you to insert a pause before the start of the first task.

Disable houselight during tasks?

If ticked, the houselight will be off during actual tasks. Note that if you don't have the houselight on, the "punishment darkness" will be harder to notice! :-) (Although the notional extra darkness will still contribute to the punishment time, so specifying a "darkness time" with no houselights allows you just to specify a timeout as a punishment.)

Visual appearance and touchscreen control

- **Adjust position of touchscreen stimuli to avoid a central feeder.** If you select this option, the grid used to display the stimuli (see Size, coordinate, and grids) are adjusted in an attempt to make sure that no stimuli are displayed in the centre of the screen. You may wish to select this option if you have a feeder or lick tube positioned in the centre of the testing chamber directly in front of the touchscreen.
- **Background colour.** Choose the colour of the background for use during tasks, by specifying a red/green/blue combination - by default black (red = 0, green = 0, blue = 0). Each of the three numbers can range from 0 (none) to 255 (full). So black is R=0, G=0, B=0; white is R=255, G=255, B=255; bright red is R=255, G=0, B=0; bright yellow is R=255, G=255, B=255... and so on. When no task is running, the display will be black.
- **Respond to mouse input as well as touches.** By default, the program responds to touches and ignores direct mouse input. This allows mouse input to be used in addition to touchscreen input. (Note that WhiskerServer can be used to make mouse input to the *server's* "observation" copy of a display mimic touchscreen input to the "real" copy of a display - the one the subject sees. This option allows mouse input to the "real" copy to be used.)
- **'Strict' touches.** If ticked, the subject is not allowed to touch the background and then "slide" its finger onto the stimulus, or to leave the finger on the screen from a previous trial. It must remove its finger and touch the stimulus squarely. Note that the SimpleSchedules task requires, and always uses, strict touches, regardless of this setting (and it always uses the "touchscreen timeout" value described next).
  - **If touchscreen fails to tell us the finger's come off, how long to wait before assuming it?** The use of "strict" touches requires that the touchscreen send "finger off" messages correctly. Some touchscreens fail to do this, occasionally. If this happens, then MonkeyCantab might think that a finger is on the screen when actually it has been removed. So that the program does not wait for ever, you can specify a timeout here: if this time elapses, the equivalent of an automatic "finger off" message is generated so the task can progress. Specify 0 for no limit, if you're confident in your touchscreen!

Reward

- **Give pellet?** If you select this option, pellets will be delivered when the subject is rewarded.
  - **Pellets per reinforcement.** Choose the number of pellets per reinforcement. (This option is only applicable if you choose to give pellets in the first place.)
  - **Pellet pulse length (ms).** Select the length, in milliseconds, of the electrical pulse that will successfully activate your pellet dispenser. (For typical Med Associates 45-mg pellet dispensers, 45 ms works quite well, and for Cambridge Cognition pellet dispensers, 100 ms

is typical, but you will have to experiment to find the best value for your device.)
    - **Interpellet gap (s).** Only applicable if you are giving multiple pellets per reinforcement. This determines the length of time, in seconds, that the program will wait between giving each pellet in a multi-pellet reward. Choose a value that is long enough to let your pellet dispenser recover from the previous delivery - very short interpellet gaps can cause pellet dispensers to jam.
    - **Use magazine lamp?** If selected, then the pellet magazine lamp will be turned on when a pellet reward is delivered, and turned off when the subject next collects reward (i.e. responds at the pellet magazine door).
- **Turn on pump?** Select this to use pump reinforcement.
    - **Pump reinforcement duration.** How long should the pump run?
    - **Pump contingent upon licking during this time?** If this option is *not* ticked, then the pump simply runs for the time specified. If you tick this option, the pump is *available* for this duration, but is only actually activated when the subject licks.
        - **Each lick delivers liquid for this duration (s):** If the pump is made contingent upon licking, then each lick activates the pump for a certain time. Specify that time here.
- **Play sound?** If you select this option, the Reward sound will be played when the subject is rewarded (see *Sounds* below).
- **Extra reward device?** To have another device activated when the subject is rewarded (e.g. an external tone generator), tick this option.
    - **Duration (s).** How long should this extra reward device be activated for?
- **Flash visual stimulus?** Optionally, an arbitrary visual stimulus can be displayed during reward, and flashed if need be. The stimulus so chosen will not respond to being touched in any way (moreover, it will be "transparent" to touches, so anything being displayed by the MonkeyCantab task in use can still be responded to - so take care not to position your reward stimulus where it might get in the way of a task!).
    - Click **Choose** to select a stimulus from the available visual stimuli.
    - **X and Y coordinates.** Specify an X coordinate and a Y coordinate for the *centre* of the stimulus (see Size, coordinates, and grids).
    - **Overall duration (s).** Choose the duration for which the stimulus should be displayed/ flashed for.
    - **On time and off time (s).** Choose the "on time" and "off time" for the flashing stimulus. To explain on time, off time, and overall duration, suppose the overall duration is 10 s, with the on time at 2 s and the off time at 1 s. Then the stimulus will be on for 2 s, off for 1 s, on for 2 s, off for 1 s, on for 2 s, off for 1 s, on for 1 s - and then a total of 10 s has elapsed and the stimulus is turned off and left off.
- **Darkness?** If you select this option, the houselight will be switched off as part of the reward. (This option is off by default and is here to enable symmetry with punishment if desired.)
    - **Darkness time (s).** This sets the length of time the houselight will be off (only applicable if you selected the previous option).

Note that you could give rewards with no sounds, or sounds with no rewards, or both.

Overall number of rewards. Optionally, MonkeyCantab can terminate an entire session when a certain number of rewards have been delivered (where "one reward" is defined as above). This applies across all tasks. Specify 0 for no limit. For example, if you specify a limit of 100 rewards here, and your subject starts with a Reversals task in which it gains 60 rewards, and then proceeds to a DMTS task, then when 40 rewards have been gained in the DMTS task, the whole session will end, even if the DMTS task's trial limit has not yet been reached. *Note that this facility does not "interrupt" individual tasks at odd points. If a task allows the delivery of two or more rewards in one trial (e.g. in DMTS if both Phase 1 and Phase 2 are rewarded, or in Spatial Working Memory if every touch is being rewarded), then the DMTS task gets to choose when it actually ends (in this case, after a full trial has been completed). So this limit is an "advisory" rather than an "absolute" limit: tasks are allowed to complete trials that they're in the middle of, even if they hit this limit. In practice, this limit will not be exceeded by very many.*

Punishment

- **Darkness?** If you select this option, the houselight will be switched off as part of the punishment.
  - **Darkness time (s).** This sets the length of time the houselight will be off (only applicable if you selected the previous option).
- **Turn on pump 2?** Optionally, you can use the PUMP2 device to deliver mildly aversive substances (e.g. saline), instead of the rewarding substance that you normally deliver as reward (via the PUMP device). Since this technique only works if the manner of collecting reward and punishment is identical (so the subject can't tell in advance what it's going to get), the other parameters for pumping (e.g. lick contingency, duration) are exactly the same as those for the reward pump.
- **Play punishment sound?** Chooses whether or not to play the Punishment sound as part of the punishment.
- **Extra punishment device?** To have another device activated when the subject is punished (e. g. an external tone generator), tick this option.
  - **Duration (s).** How long should this extra punishment device be activated for?
- Shocks are not explicitly implemented as a punishment option (for ethical reasons: this seems pretty severe). Contact rudolf@pobox.com if this causes problems.
- You can also specify a **visual stimulus** to be displayed during punishment (flashing, if desired); the parameters correspond to those for reward stimuli (see above).

Default media directory

If the server needs WAV files or bitmaps (.BMP) and cannot find them, it looks in this directory. If you have a collection of multimedia files (.WAV, .BMP) that you are using with MonkeyCantab, we suggest you select that directory here. Click **Set** to browse for the directory.

Sounds

For the predefined sounds (Link, Reward, Punishment, Marker1, Marker2, Marker3), you may set the following options:

- **Use WAV file.** Sounds may either be played as simple tones or as WAV files.
- **Filename.** To specify WAV files. Click **Set** to browse for the file. (Only applicable to WAV sounds.)
- **Frequency (Hz).** Specifies the sound's frequency in Hertz. (Only applicable to non-WAV sounds.)
- **Sound type.** Choose the waveform of your sound. "Tone" is similar to "Sine" but contains more energy. (Only applicable to non-WAV sounds.)
- **Duration (s).** The duration of the sound, in seconds. (Only applicable to non-WAV sounds.)
- **Level (0-100).** The volume of the sound. Maximum volume is 100; minimum volume is 0. More specifically, this number is 100 minus the sound attenuation in decibels (dB).

To disable a sound, choose a non-WAV sound and set its duration and/or volume to zero.

Marker 1 is typically used to indicate the start of a trial.
Marker 2 is typically used to indicate the start of a second phase of a trial.
Marker 3 is typically used to provide response feedback.

1.6.2 Mimicking Monkey CANTAB for DOS

Cambridge Cognition Ltd used to sell Monkey CANTAB for DOS.

To mimic this, several settings are relevant.

Sounds

Reward sound: **1000 Hz,** sound type **square,** volume 85
Feedback sound (marker 3?): **300 Hz,** sound type **square,** volume 85
Punishment sound: **40 Hz,** sound type **square,** volume 85

[Frequencies confirmed with Cambridge Cognition, e-mail of 25 Feb 2004: PC_TONES file reads "correct=1000 feedback=300 incorrect=40".]
[Square-wave choice is preferred by Spencer Tye of Merck, Sharpe, Dohme, Feb 2004.]

1.6.3 Use with dogs

 Woof.

For use with dogs, in apparatus that has an omnidirectional lever (an on/off device triggered whenever the dog pushes a stick in any direction).

The lever can be used to initiate trials in a variety of tasks. This is configured within each task, not in the General Parameters.

Details:

- Reinforcement Familiarization: the lever is ignored, as this is a task for non-contingent reward delivery.
- Touch Training: lever activation can be required to initiate trials.
- Simple Schedules of Reinforcement: the lever is ignored, as the schedule is for free-operant touchscreen responding (adding a lever requirement would make the schedules into complex chain schedules instead).
- Spatial WM (SOSS): lever activation can be required to initiate trials.
- PAL: lever activation can be required to initiate trials.
- DMTS: lever activation can be required to initiate trials.
- MCSRT: lever activation can be required to initiate trials.
- Visual discriminations and set-shifting: lever activation can be required to initiate trials.
- Reversal learning: lever activation can be required to initiate trials.
- Impulsive choice: lever activation can be required to initiate trials.

From the program's point of view, the dog lever is indistinguishable from the monkey lever (i.e. it's simply an on/off lever as far as the program is concerned, and is referred to as LEVER in the Required Devices).

In general, when a lever-press is required to start a trial:

- the houselight is on throughout a task and is not switched off during the ITI (unless the previous trial was failed and the punishment includes darkness)
- lever activations are required to start each trial, rather than to start the task as a whole
- the Marker 1 sound (where used), which previously indicated the start of the trial, will indicate the opportunity to press a lever to start the trial

- when the lever is activated, the trial begins immediately
- if the lever is already being held "on" when a lever response is required, the program waits until it's released and /activated again.
- if the session time limit elapses while the program is waiting for a lever-press, the program will wait for the current trial to be initiated and completed, and will then finish.

1.7 Visual stimuli

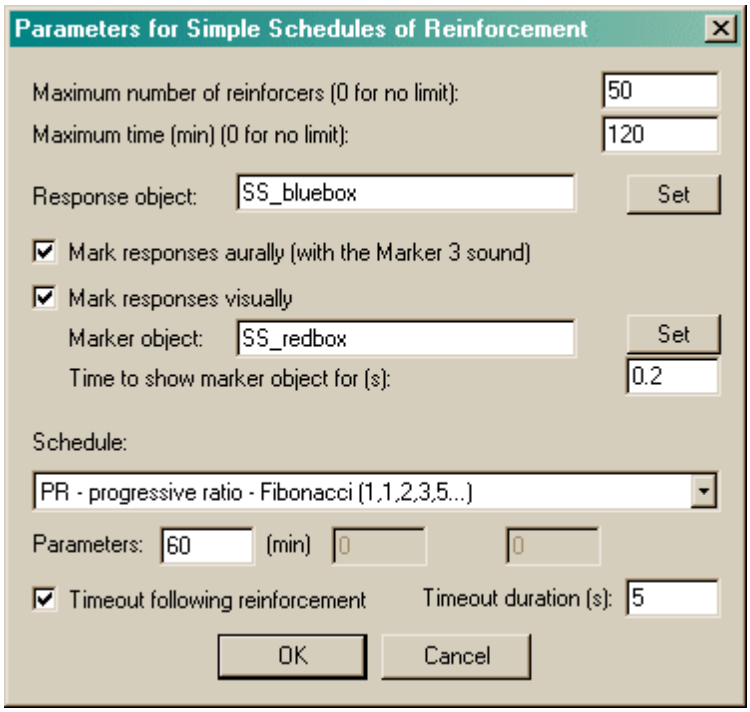
Stimuli used by MonkeyCantab come from one of two places: the **visual object library,** which contains stimuli you have defined, and a group of **predefined stimuli.**

For further details, see:

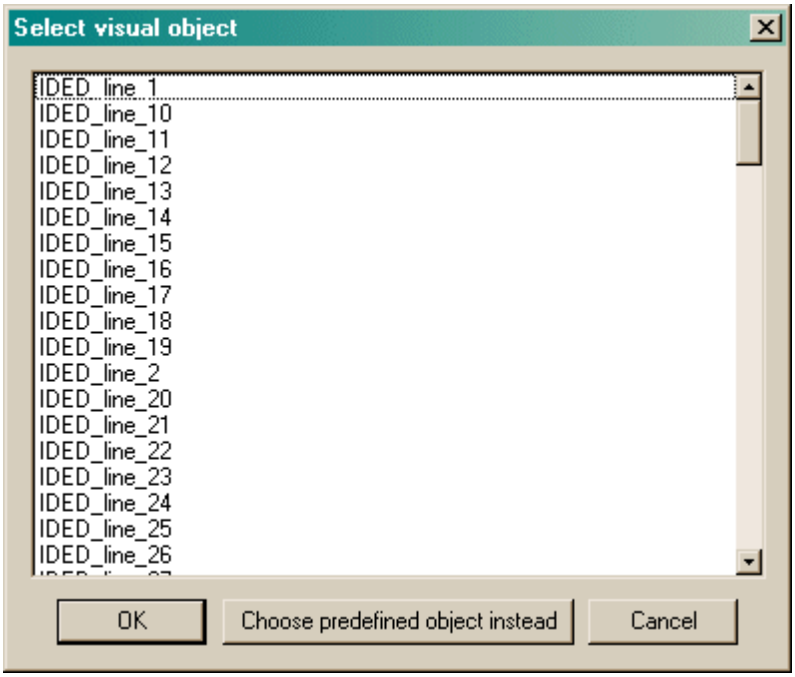
- How to choose stimuli for use with a task
- How big should a stimulus be? The coordinate and grid systems used by MonkeyCantab
- Editing your own stimuli
- Predefined stimuli

1.7.1 Choosing stimuli for a task

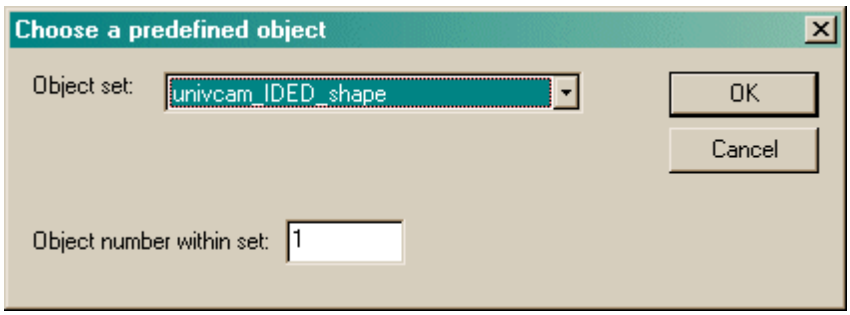
Many tasks ask you to pick stimuli to use. For example, here's the Simple Schedules task:



It would like to use two stimuli (the **response object** and the **marker object**). At the moment, the stimuli "SS_bluebox" and "SS_redbox" are being used. But you could change that. If you know the name of one of your stimuli in the visual object library, or the name of a predefined stimulus, you can type that name directly. Otherwise, press the **Set** button next to the object. You'll get a choice like this:



This is the list of all the objects currently defined in your Visual Object Library. If there aren't any, or there aren't enough, you need to edit the visual object library to add some more. If you see one that you want, highlight it (by clicking on it) and then click OK. If you want to use a predefined object, click **Choose predefined object instead,** and you'll see this:

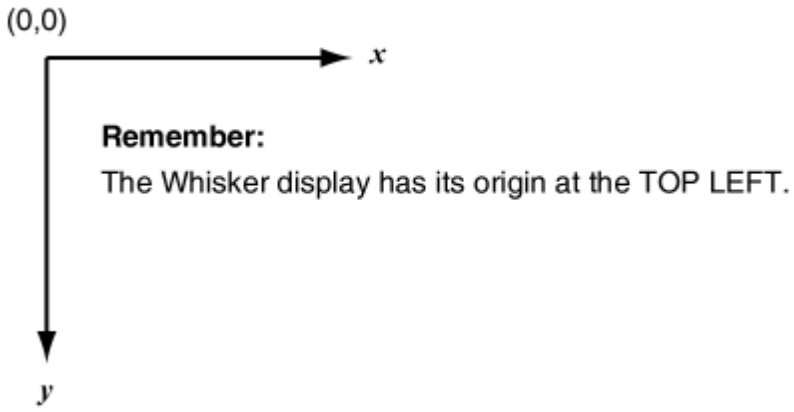


Choose a stimulus set from the predefined stimuli available; then choose an object within that set, and press OK.

1.7.2 Size, coordinates, and grids

If you're having problems positioning your own stimuli, see My stimuli are mis-positioned.

What coordinate system does MonkeyCantab use?



How do the tasks position objects on the screen?

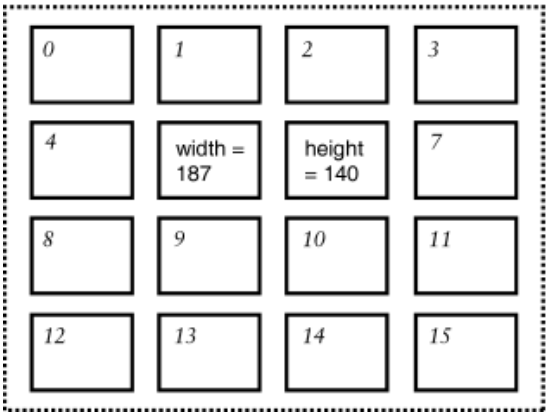
All MonkeyCantab tasks treat the screen as if it were a 1000x750 grid. They divide the screen up into rectangles, shown below. When called upon to display a stimulus, they attempt to work out the stimulus size and then they try to position the stimulus in the centre of the grid square tha they are using, assuming that the stimulus begins at (0,0) in its internal coordinate system (the system with which you define objects).

The tasks will have problems determining the size of text, and of bitmaps if you don't force the bitmap to a specified size. You should therefore test these stimuli before using them.

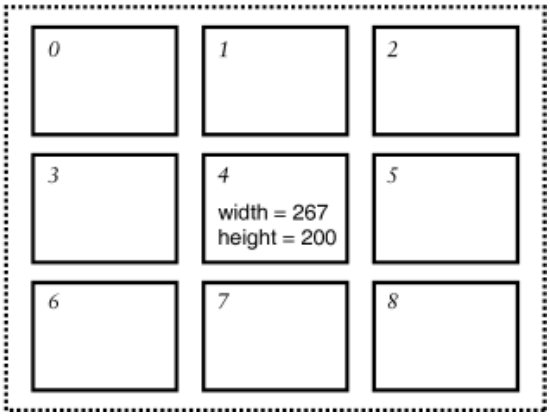
- Reinforcement Familiarization doesn't display any stimuli.
- Touch Training uses the nine-way grid. For levels 1 and 2, the middle row of the non-way grid is used; for level 3, when three stimuli are shown, they are positioned at the top centre (stimulus 1), bottom left (stimulus 2), and bottom right (stimulus 3).
- Visual Discrimination and Set Shifting uses the two-way grid.
- Reversal Learning uses the two-way grid for two-stimulus tasks, and the middle row of the nine-way grid for three-stimulus tasks.
- Delayed Matching/Non-matching To Sample uses the nine-way grid (typically location 4 for the presentation phase, and locations 0, 2, 6, and 8 for the choice phase).
- Spatial Working Memory uses the sixteen-way grid, the sixteen-way scattered pattern, and the eight-way scattered pattern.
- Multiple-Choice Serial Reaction Time uses the location 7 of the nine-way grid for the touchscreen centring response (Phase 1) stimulus. In the choice phase, it uses the whole screen for the one-choice task, locations 6 and 8 of the nine-way grid for the two-choice task, the middle row of the nine-way grid for the three-choice task, and the five-choice pattern for the five-choice task. If these locations are not to your liking, you can configure alternative locations manually in the task parameters.
- Paired-Associates Learning uses the nine-way grid or the nine-way (extreme) grid.
- Simple Schedules of Reinforcement uses the centre of the nine-way grid.
- Impulsive Choice uses the nine-way grid.
- Rapid Visual Information Processing uses the nine-way grid.

The sizes of these grids are shown below. In the 16-, 9-, and 4-way grids, the space between the grid squares and between the grid and the edge of the screen is 5% of the screen's extent. Refer to the width/height of individual cells; for example, stimuli designed for use with tasks that use the nine-way grid should fit in a 267x200 rectangle. In the grids used by the Spatial Working Memory task, the grid squares are numbered and these numbers are used to position stimuli.

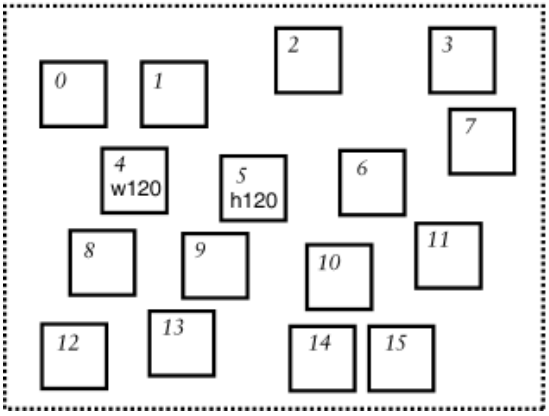
Sixteen-way grid



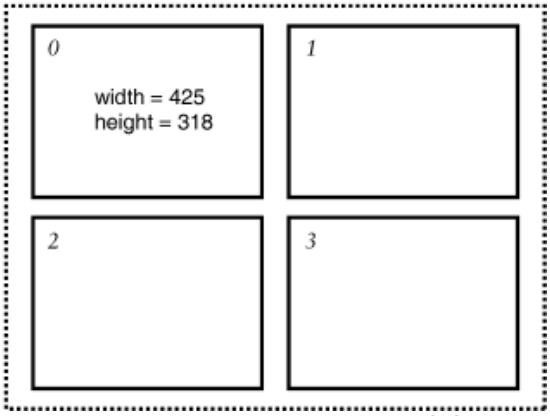
Nine-way grid



Sixteen-way scattered (approx.)

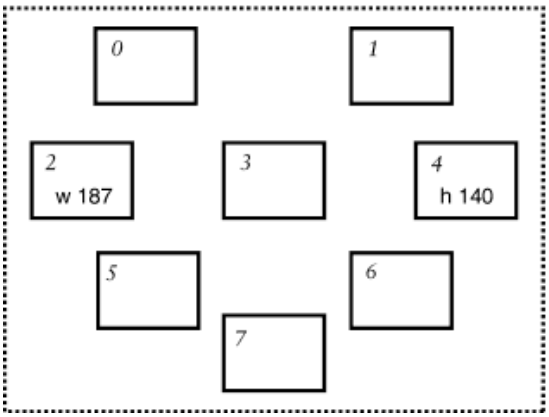


Four-way grid

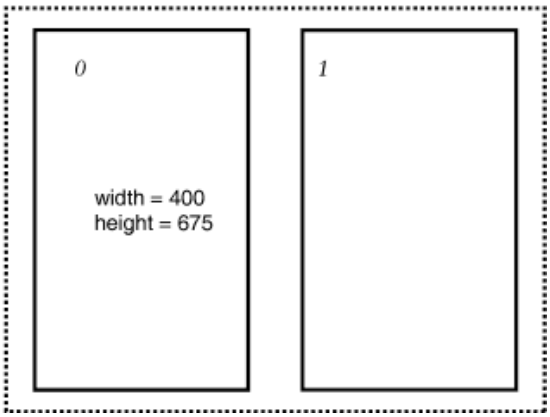


whole screen
width = 1000
height = 750

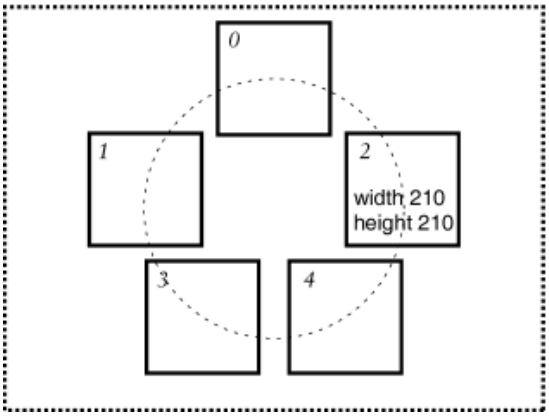
Eight-way scattered



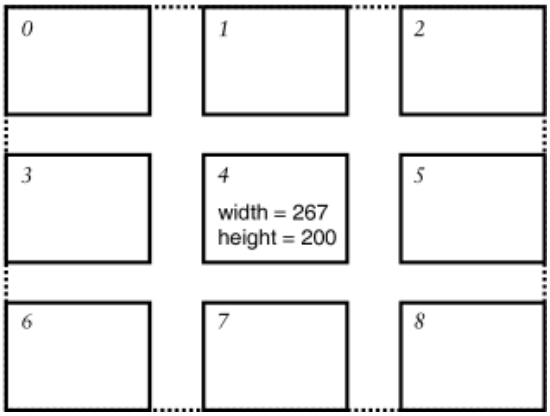
Two-way grid



Five-choice pattern



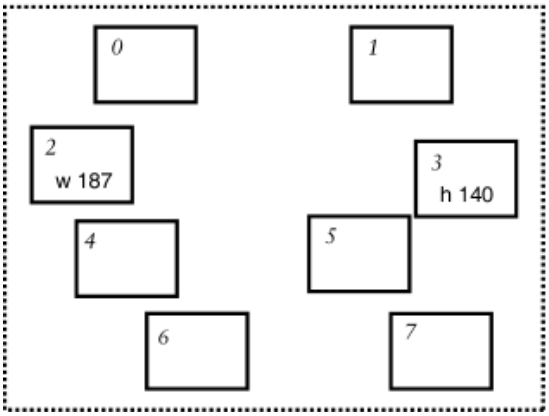
Nine-way (extreme) grid



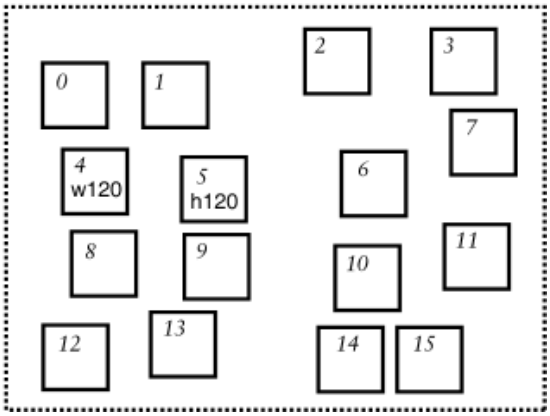
Avoiding a central feeder

If the program is avoiding a central feeder, which is just under 5% of the screen's width, the two-, four-, and sixteen-way grids are used unaltered. The sixteen- and eight-way scattered grids, and the five- and three-choice patterns, are modified as follows:

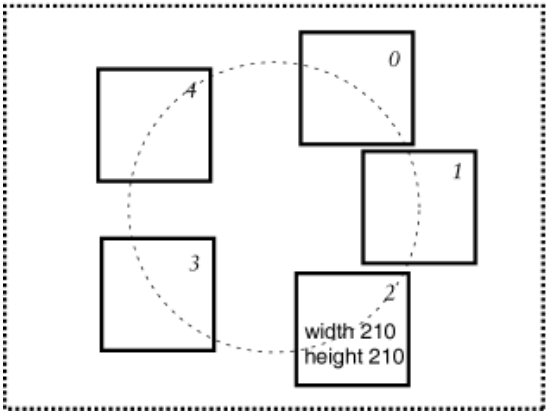
Eight-way scattered (modified)



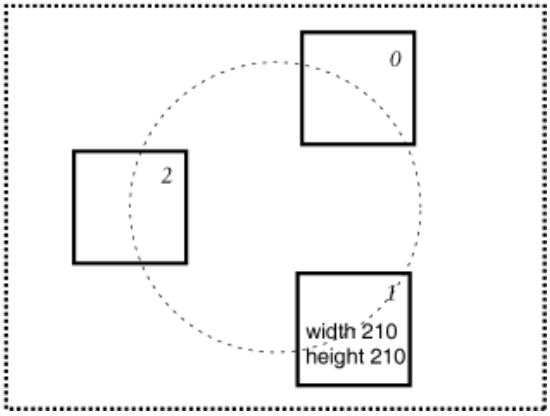
Sixteen-way scattered (modified)



Five-choice pattern (modified)



Three-choice pattern (modified)



This leaves only the nine-way grid, which can't easily be modified to avoid the centre! The tasks that use it behave as follows:

- The **touch training** program ignores the problem; choose stimuli that are large enough to be

touched from either side of the feeder.
- The **multiple-choice** task continues to use location 4 or 7 of the nine-way grid for its centring stimulus, if touchscreen centring is selected. You should choose a stimulus wide enough to be touched from both sides of the licker. The same applies to the target stimulus used in the one-choice version of the task.
- The **PAL** task should be used with the **corners** grid, rather than the **edges** grid. This alleviates the problem.
- The **schedules of reinforcement** task ignores the problem; choose a stimulus that's large enough to be touched from either side of the feeder.
- The **reversals** task (in three-stimulus mode) uses, at random, locations 0/5/6 or locations 2/3/8 of the nine-way grid, and records which one it used.
- The **DMTS** task allows you to specify exactly the locations in which stimuli are presented. Choose locations that aren't in the centre.
- The **ImpulsiveChoice** task only uses the centre (obligatorily) for the initiation stimulus, if one is used; choose a stimulus large enough to be touched from either side of the feeder.
- The **RVIP** program ignores the problem; choose stimuli that are large enough to be touched from either side of the feeder.

1.7.3 My stimuli are mis-positioned!

What do you do if your stimuli are mis-positioned? There are several things you can do.

(1) Ensure that your touchscreen is calibrated correctly.

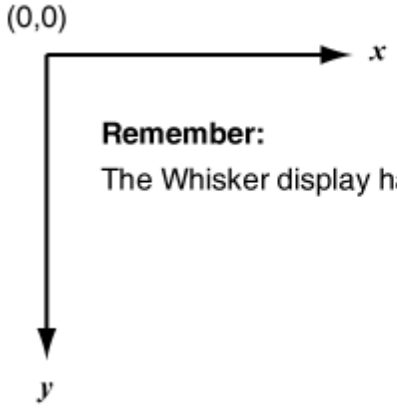
If the stimuli look OK but touches aren't being detected at quite the right position, your touchscreen needs calibration. WhiskerServer helps you calibrate. When you're not running MonkeyCantab but have WhiskerServer running, choose "Display → Show test pattern on all displays" from the WhiskerServer menus. Now touch your touchscreen; you should see a crosshair appear and change colour. If the crosshair isn't exactly where your finger is, recalibrate your touchscreen (for example, using the UPDD control panel - e.g. Start → Control Panel → Pointer Devices → Calibrate).

(2) Alter the positioning by redefining the stimuli.

How does the program position stimuli?

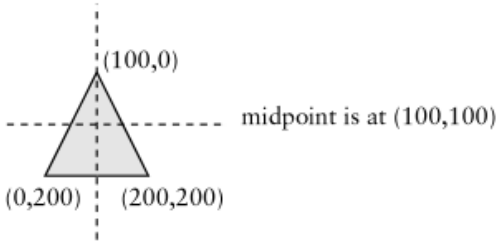
- When MonkeyCantab displays a stimulus, it does so in a grid location. The grid pattern depends on the task; for example, the D(N)MTS task uses a 3x3 (9-way) grid, while the Visual Discriminations task uses a two-way grid.
- Next, it calculates the **total extent** (X extent and Y extent) of your stimulus.
- It works out the **midpoint** of your stimulus, which it assumes to be the point that's halfway between the leftmost point and the rightmost point, and halfway between the topmost point and the bottommost point.
- It displays your stimulus so that the midpoint of your stimulus is at the midpoint of the grid location.

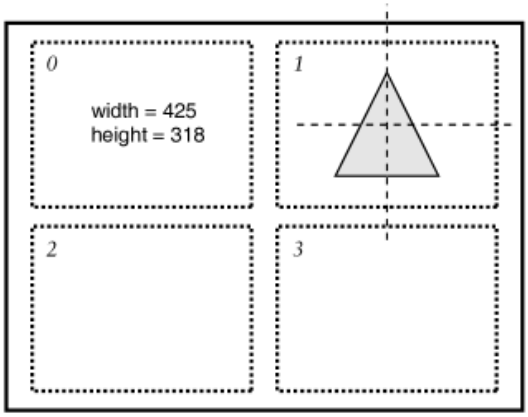
Here are some examples of how you can use this:



Remember:
The Whisker display has its origin at the TOP LEFT.

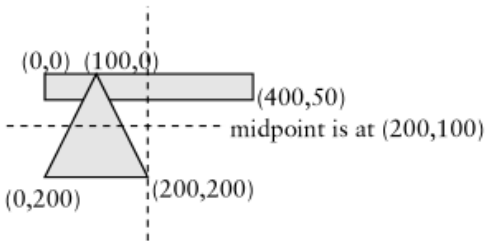
1 Suppose your stimulus is to be shown in the top-right location of the 4-way grid.
It's a triangle, with points at (100,0), (0,200), (200,200).
The leftmost coordinate is 0; rightmost 200; topmost 0; bottommost 200.
The program calculates its midpoint as (100,100).
It displays the stimulus so this midpoint is in the centre of the grid location.

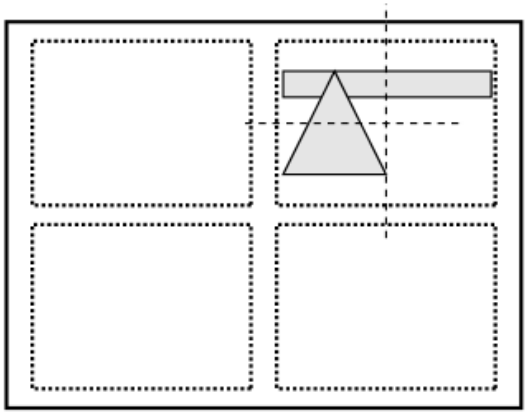




Screen with four-way grid shown

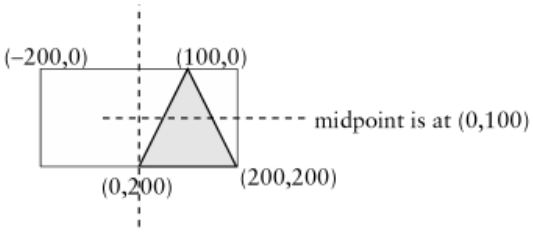
2 Now you add a rectangle to your stimulus: top left (0,0) bottom right (400,50).
Overall, the leftmost coordinate is 0; rightmost 400; topmost 0; bottommost 200.
The program calculates its midpoint as (200,100).
It displays the stimulus so this midpoint is in the centre of the grid location.

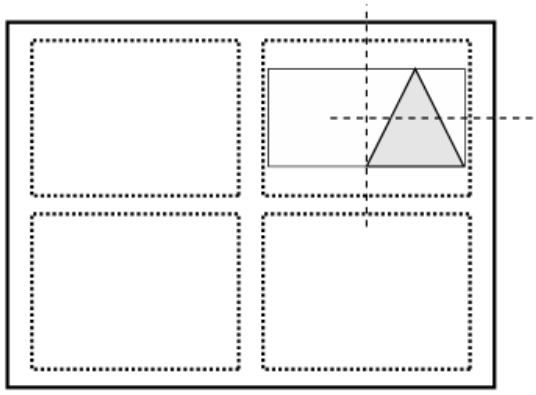




Screen with four-way grid shown

3 If you wanted to have your triangle to be shifted slightly to the right, you could add an invisible rectangle. Suppose you take your triangle and add a black rectangle (with a black border) in the background: top left (–200,0) bottom right (200,200). Overall, the leftmost coordinate is –200; rightmost 200; topmost 0; bottommost 200. The program calculates its midpoint as (0,100).
It displays the stimulus so this midpoint is in the centre of the grid location.





Screen with four-way grid shown

Note that for BITMAPS, you may need to specify a desired width/height, or your stimuli may be mis-positioned. See bitmaps.

(3) Move the whole display left/right or up/down using the controls on your monitor.

(4) Shrink or enlarge the whole display using the controls on your monitor.

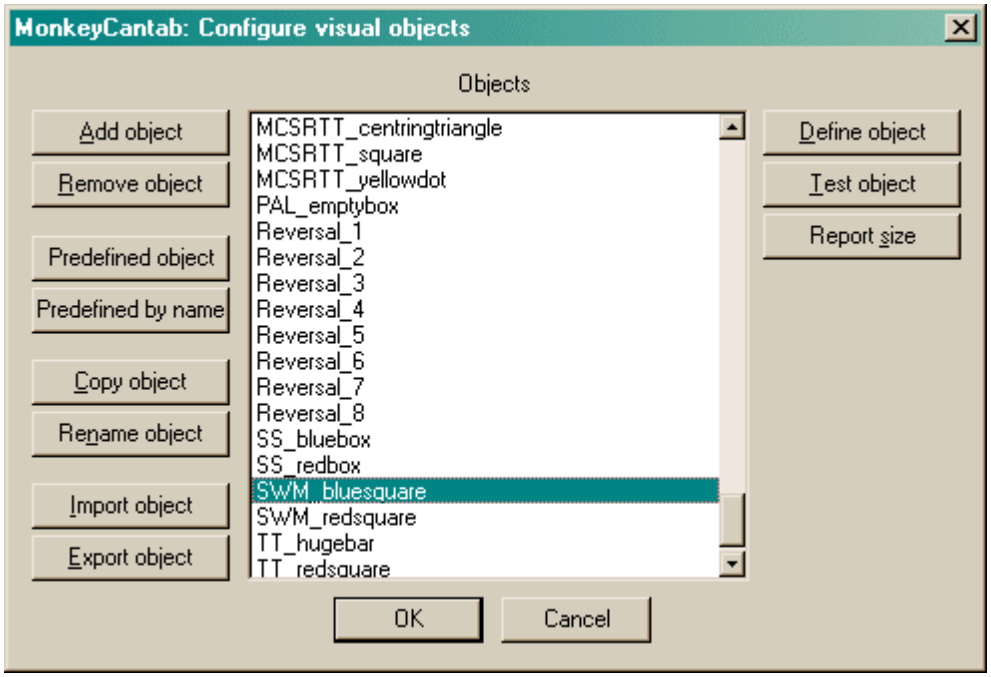
(5) Shrink or enlarge the whole display by altering the size of the screen's border, using WhiskerServer.

If stimuli are too far left when they're on the left of the screen, too far right when they're on the right of the screen, too far up when they're at the top, and so on, you can simply shrink the active area of the screen (within WhiskerServer, choose Server → Configure hardware → Display devices, and alter the "border" setting).

1.7.4 Editing stimuli

Editing the visual object library

Every visual object (picture) used by MonkeyCantab lives in the **visual object library** and has a **unique name** associated with it.



- You can **add** and **remove** objects from the list - though you cannot remove an object that is being used by one of your tasks. You can make a **copy** of an object and you can **rename** objects. If you rename an object, all references to it by other tasks in your task list will be amended accordingly.
- Click **Define object** to configure the object itself.
- If you are already connected to a Whisker server, you may click **Test object,** and the object will be displayed in a new window (a "virtual device window") on the server's desktop for you to inspect it.
- Click **Report size** to show the overall size in units (compared to an active screen area that is 1000 notional units wide and 750 units high).
- Click **OK** to accept your changes, or **Cancel** to abort.
- Sometimes (as shown above) when you select an object, a message appears saying " **WARNING: This object has no touchable components."** In this case, the object will not respond to being touched, so you are probably best avoiding it in all your tasks until you have added a component to it that is touchable (see below)!
- You may add **predefined objects,** either by selecting them from a list (click **Predefined object**) or by specifying their name directly (**Predefined by name**).
- You may **import and export objects from other configuration files.** You may find it convenient to keep one configuration file as a master object library (for example, you could give it a dummy subject name and not use it to run tasks). If you design a handy object for one subject, you could export it to the library. When you create a new configuration file, you can either load one that's quite similar and save it under a new name, or start from scratch and import objects from your library. Click **Import** and **Export** to import/export objects. You will be asked to choose the configuration file for import/export, and then to choose the objects to copy across. When importing, you can select multiple objects (click on several objects). If objects with the same name exist in the file you are importing into (or exporting to), the objects will be renamed upon arrival.

A sample object library is supplied with MonkeyCantab. It's called **MonkeyCantab_TestConfig_And_SampleObjectLibrary.xml.**

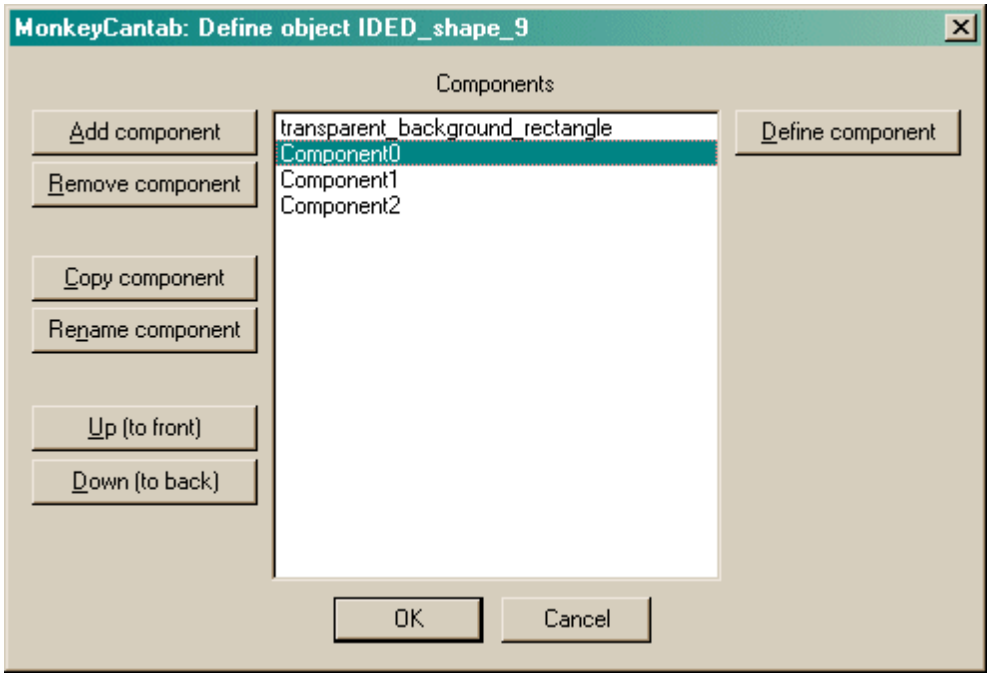
Editing individual objects

When you use your new object in a MonkeyCantab task, the task may place your object anywhere

on the screen. Each task defines certain locations that it uses (for example, the ThreeChoice task displays them in one of three locations in a horizontal line in the middle of the screen). The task's locations are defined as rectangles. The tasks automatically try to centre your objects in these rectangles. (Some objects, like text, give them more difficulty!) This centring system relies on you basing the top-left corner of your objects at the point (0,0). If you don't, your objects will be offset in the tasks.

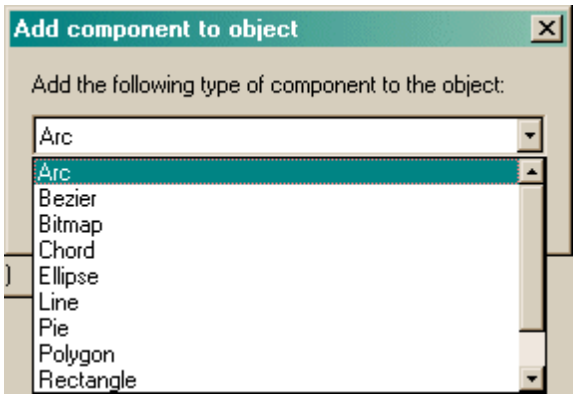
Editing components of individual objects

When you clicked *Define* in the dialogue box above, you can define components of one particular object in the dialogue box shown below.



Here we are defining the object called "green", and at present it has two components, named "greenpie" and "bluerectanglebackground". The objects are in a stack: "greenpie" is at the top of the stack, so it will be the object in front of all the others. "Bluerectangle..." is at the bottom of the pile.

- Click **Add** or **Remove** to add/remove components from the list. When you click Add, you will be offered a choice of the various types of component that are available, and then be asked to give your new component a name. (Note that two components can't have the same name.)



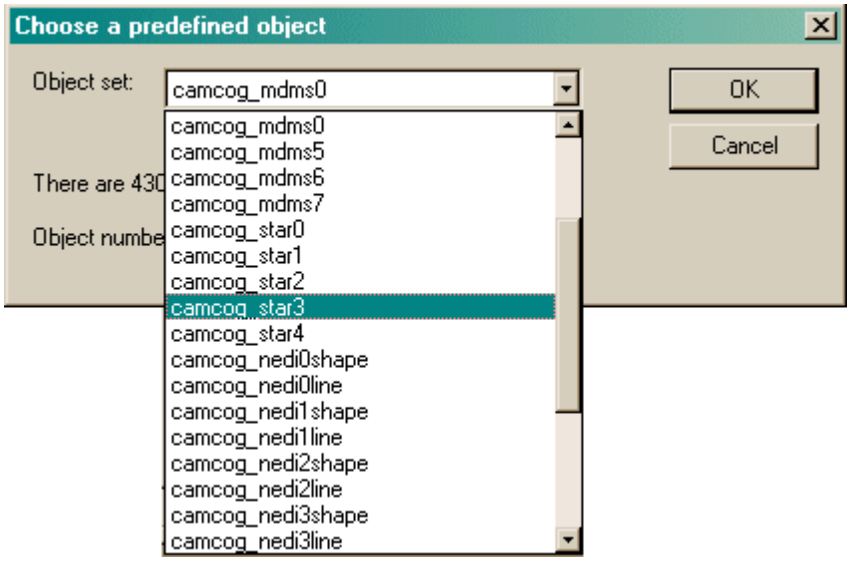
- Click **Copy** or **Rename** to copy or rename a component. (Note that two components can't have

the same name.)
- Click **Up** or **Down** to move components up or down the stack.
- Click **Define** to specify a component's details.

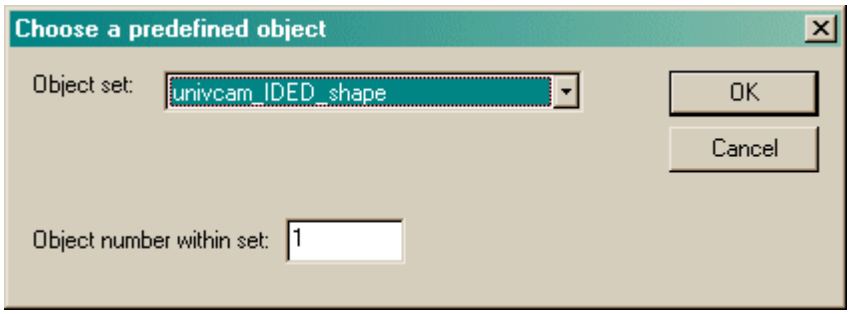
Adding copies of predefined objects to your library of stimuli

MonkeyCantab is supplied with a set of predefined stimuli. You can use them in your tasks **without** having to add them to your Visual Object Library. However, you can also make copies of predefined stimuli in your library; you can then edit them to make new stimuli.

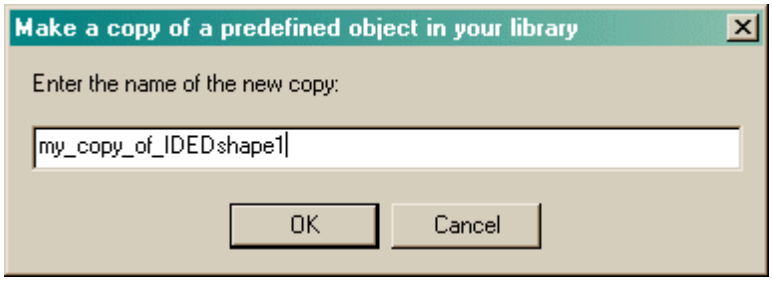
When you click on **Predefined object,** you are offered a choice of predefined stimuli. First, pick the stimulus set:



Next, choose the object number within that set:

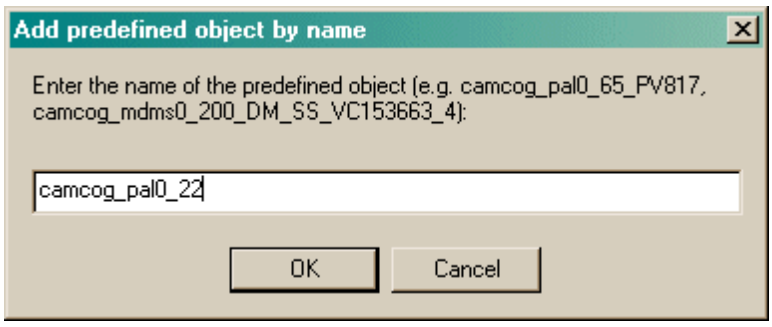


Finally, give your copy a name:



When you click on **Predefined by name,** the same procedure applies except that steps 1 and 2

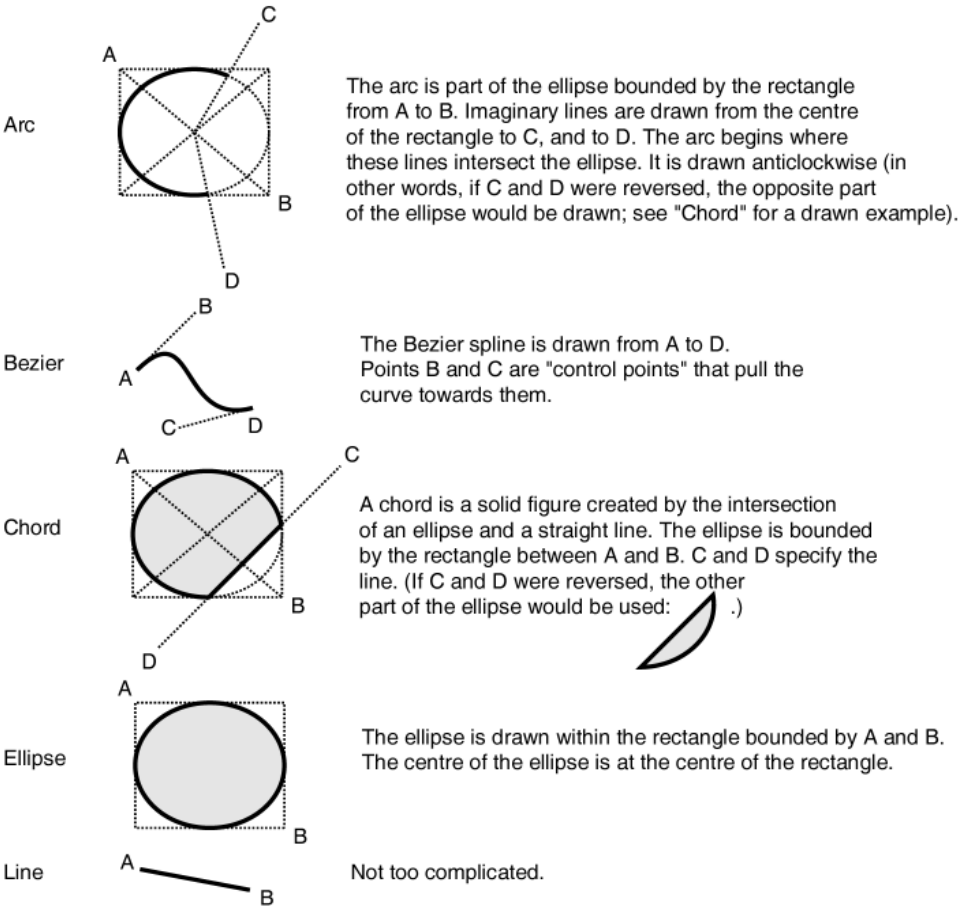
are replaced by you typing in the name of a predefined stimulus or a **variant** thereof (see Predefined stimuli), like this:



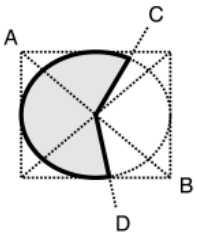
1.7.4.1 Defining components of objects

What do the components look like, and how do we define them?

Aside from bitmaps and text, all component types are illustrated in the picture below.
Your objects should begin at the top-left point (0,0). Increasing x/y coordinates move to the right and down.

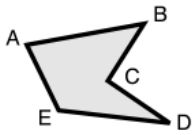


Pie



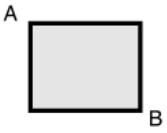
A pie is exactly like an arc but is a solid figure. (Again, it is drawn anticlockwise, and reversing C and D would cause the rest of the pie to be drawn instead.)

Polygon



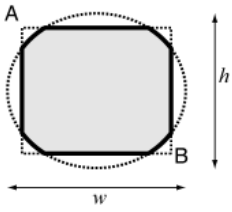
A polygon joins all the specified points, in order, completing the shape if necessary. The fill mode is complicated. **Alternate:** the system fills the area between odd-numbered and even-numbered polygon sides on each scan line. That is, the system fills the area between the first and second side, between the third and fourth side, and so on. This mode is the default. **Winding:** the system uses the direction in which a figure was drawn to determine whether to fill an area. Each line segment in a polygon is drawn in either a clockwise or an anticlockwise direction. Whenever an imaginary line drawn from an enclosed area to the outside of a figure passes through a clockwise line segment, a count is incremented. When the line passes through an anticlockwise line segment, the count is decremented. The area is filled if the count is nonzero when the line reaches the outside of the figure.

Rectangle



Not too complicated.

Roundrect



A rounded rectangle is drawn. The rectangle from A to B is drawn with corners that are part of the ellipse whose width is w and whose height is h. (The ellipse and the rectangle share their centre.)

Which components are touchable?

Note that **arcs, bezier splines, lines, and text CANNOT support mouse or touchscreen events.** Everything else (bitmaps, chords, ellipses, pies, polygons, rectangle, rounded rectangles) can.

What part of the object is touchable?

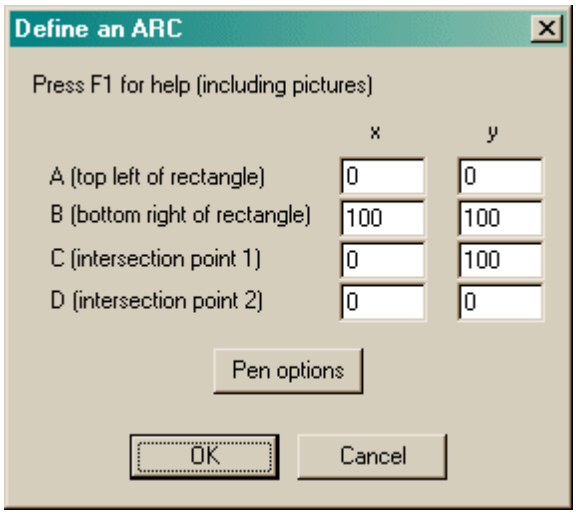
What you see is what you can touch. If you want a larger area to be touchable, define a black rectangle of the desired size (all MonkeyCantab tasks use a black background), giving it a black or null pen, and place it at the bottom of your object's stack of components.

Defining the components

When you click **Define** in the Component Definition dialogue, you can set the options for a particular component. Here are the possible components:

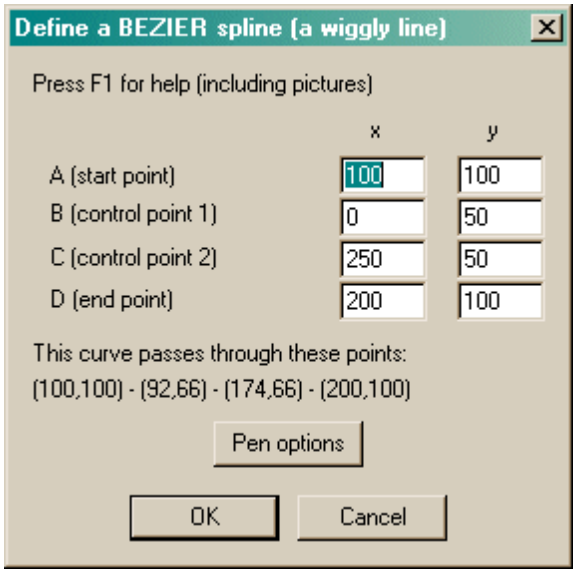
- Arc
- Bezier spline
- Bitmap
- **CamcogQuadPattern** - a special type of shape used to implement older stimulus types
- Chord
- Ellipse
- Line
- Pie
- Polygon
- Rectangle
- Rounded rectangle (roundrect)
- Text

1.7.4.2 Arc



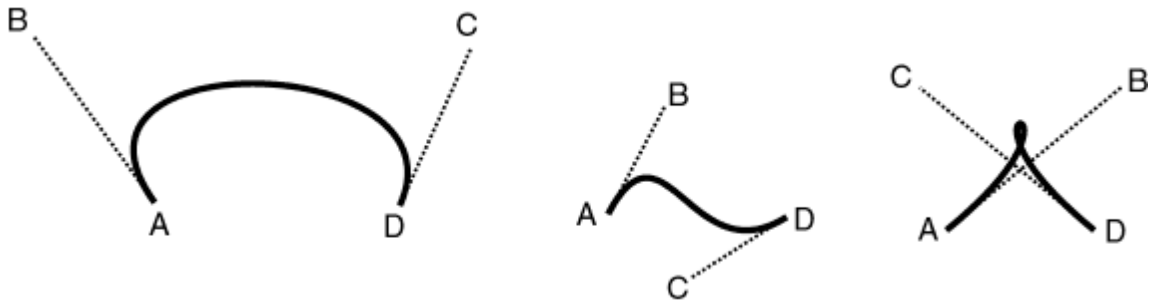
- The meaning of the points is explained in the figure above.
- Click Pen options to determine how the edge of the object is painted.

1.7.4.3 Bezier spline



- The meaning of the points is explained in the figure above and in the examples and mathematical description below.
- Click Pen options to determine how the edge of the object is painted.

Some more examples:



Definition of a Bezier curve (de Casteljau, 1959; Bezier, 1962)

Let's start simple. Imagine a line that begins at A and ends at D. Let t be a variable from 0 to 1. We can define a point $P_{AD}(t)$ on the line segment AD as

$$P_{AD}(t) = (1-t)A + tD$$

If we add another point, B, into the picture, we can define $P_{AB}(t)$ as a point between A and B, and $P_{BD}(t)$ as a point between B and D. If we apply the same method to define $P_{AB\text{-}BD}(t)$ as a point between $P_{AB}(t)$ and $P_{BD}(t)$, we get

$$P_{AB}(t) = (1-t)A + tB$$
$$P_{BD}(t) = (1-t)B + tD$$
$$P_{AB\text{-}BD}(t) = (1-t)\{ (1-t)A + tB \} + t\{ (1-t)B + tD) \} = (1-t)^2A + 2t(1-t)B + t^2D$$

This quadratic equation in t defines a quadratic Bezier curve - a parabola. For computer graphics purposes, cubic Bezier curves are more often used. As you might expect, these are defined by four points A, B, C, and D. If the cubic Bezier function is defined as a point between $P_{AB\text{-}BC}(t)$ and $P_{BC\text{-}CD}(t)$ in the same manner as we've been doing so far...

$$P_{AB}(t) = (1-t)A + tB$$
$$P_{BC}(t) = (1-t)B + tC$$
$$P_{CD}(t) = (1-t)C + tD$$
$$P_{AB\text{-}BC}(t) = (1-t)\{ (1-t)A + tB \} + t\{ (1-t)B + tC) \} = (1-t)^2A + 2t(1-t)B + t^2C$$
$$P_{BC\text{-}CD}(t) = (1-t)\{ (1-t)B + tC \} + t\{ (1-t)C + tD) \} = (1-t)^2B + 2t(1-t)C + t^2D$$
$$\text{CubicBezier}(t) = (1-t)P_{AB\text{-}BC}(t) + tP_{BC\text{-}CD}(t) = \ldots = (1-t)^3A + 3(1-t)^2tB + 3(1-t)t^2C + t^3D$$

Summary: Where we end up

Cubic Bezier splines are usually defined with endpoints A and D and control points B and C that are not on the curve, as above. The equation for a point on this curve is given by

$$\text{CubicBezier}(t) = (1-t)^3A + 3(1-t)^2tB + 3(1-t)t^2C + t^3D$$

where t is the curve's parameter and ranges from 0 to 1. This curve can be expressed in a different way: as a curve passing *through* four points, PQRS, where P=CubicBezier(0), Q=CubicBezier($^1/_3$), R=CubicBezier($^2/_3$), and S=CubicBezier(1). From the formula above,

$$P = A$$
$$Q = {}^1/_{27}(8A + 12B + 6C + D)$$
$$R = {}^1/_{27}(A + 6B + 12C + 8D)$$
$$S = D$$

and therefore

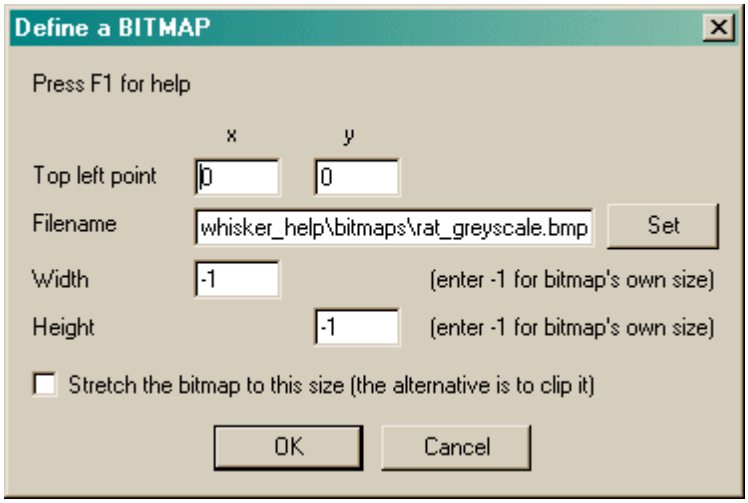
$$A = P$$
$$B = {}^{1}/_{6}(-5P + 18Q - 9R + 2S)$$
$$C = {}^{1}/_{6}(2P - 9Q + 18R - 5S)$$
$$D = S$$

In MonkeyCantab's Bezier dialogue box (above), you enter the points ABCD and MonkeyCantab shows you the points PQRS ("The curve passes through these points: ...").

Some properties of Bezier curves

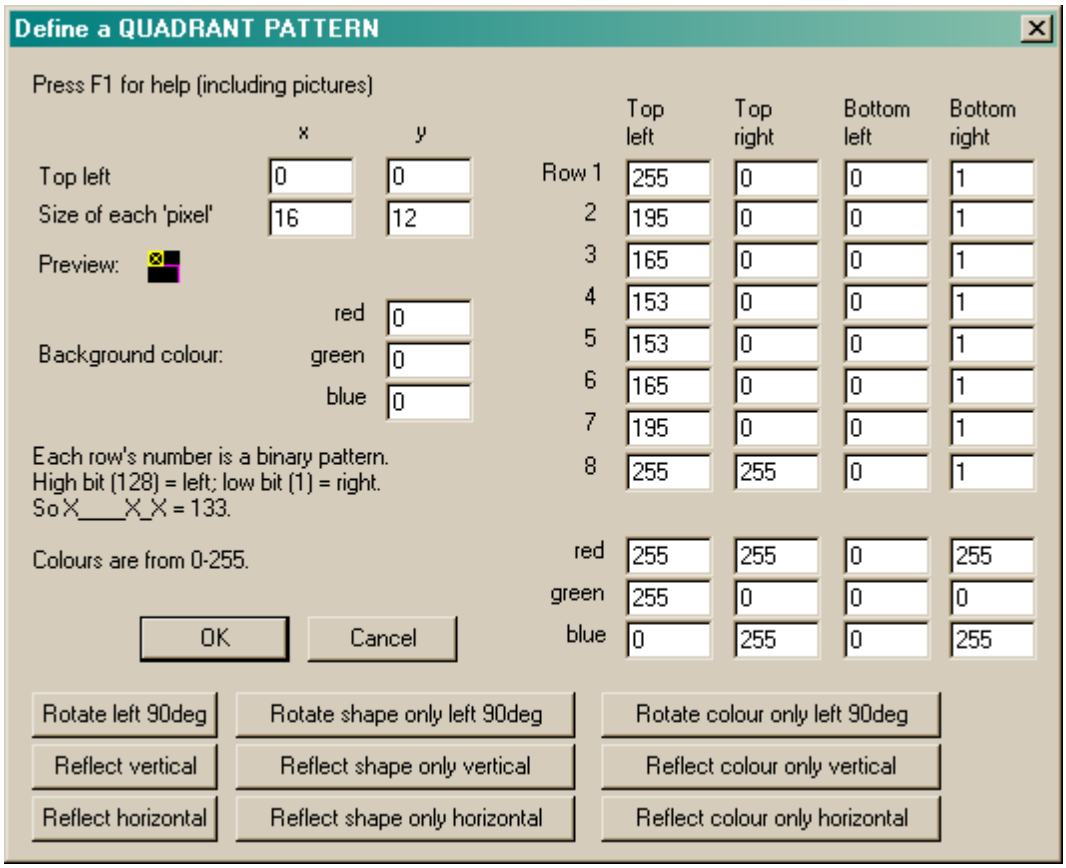
- Bezier curves always pass through their first and last points (A and D here), but not necessarily through their other control points (B and C).
- A Bezier curve always lies fully within the convex hull defined by its control points.
- The line AB has the same tangent at the curve at A, and the line CD has the same tangent as the curve at D.
- Bezier curves are always divisible into two Bezier curves (in a manner which makes them easy to draw iteratively). See Yuan, F. (2001), *Windows Graphics Programming*, Hewlett-Packard/ Prentice Hall, New Jersey (p481 onwards).

1.7.4.4 Bitmap



- Choose the **(x, y) coordinates of the top left point.** This is normally (0,0).
- Choose the **filename** of the bitmap. Click **Set** to browse for the file. If you are running MonkeyCantab on a different computer to WhiskerServer, remember that the filename must be accessible by the server, not the client.
- Choose a width and height to force the bitmap to, or leave the values at -1 to use the bitmap's intrinsic height. **Note that using the bitmap's intrinsic size may lead MonkeyCantab to mis-position the stimulus (because MonkeyCantab doesn't then know how big the bitmap is, and it may mis-calculate its position).**
- Choose whether to **stretch** or **clip** the bitmap (leave the tickbox unticked for clipping). Stretching means that the bitmap is deformed to fit your specified width/height. Clipping means that the bitmap's size isn't changed, but that the right/bottom edges may be cut off if the width/ height you specify are smaller than the bitmap's intrinsic size.

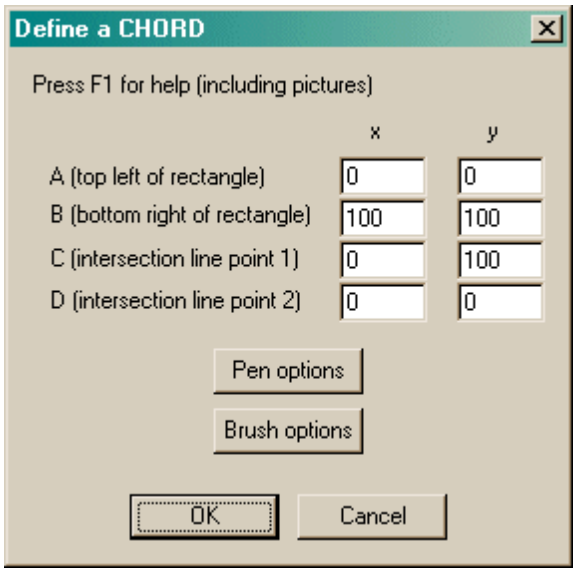
1.7.4.5 CamcogQuadPattern



This allows you to edit 16 x 16 coloured patterns, as used by many of the predefined stimuli. You can see a small preview of the final stimulus as you edit it (though its actual, final size may be altered by changing the size of each 'pixel', or dot, making up the pattern).

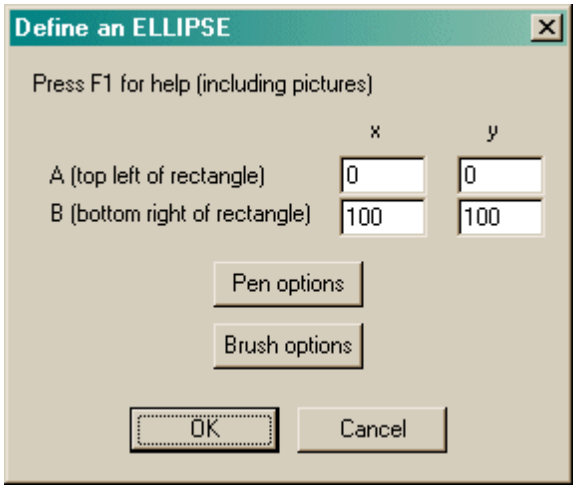
- Choose the **(x, y) coordinates of the top left point.** This is normally (0,0).
- Choose the size of each 'pixel'.
- Specify the background colour, and the detail and colour for each quadrant.
- Colours are specified from 0-255. Here are some examples with different the red/green/blue (RGB) components:
  - 0/0/0 gives black;
  - 255/0/0/0 is red;
  - 0/255/0 is green;
  - 0/0/255 is blue;
  - 255/255/0 is yellow;
  - 255/0/255 is magenta;
  - 0/255/255 is cyan;
  - 255/255/255 is white;
  - 100/100/100 is a dark grey;
  - 255/150/0 is an orangey colour; and so on.

1.7.4.6 Chord



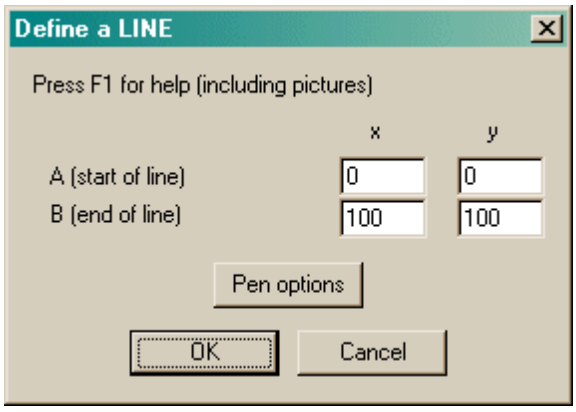
- The meaning of the points is explained in the figure above.
- Click Pen options to determine how the edge of the object is painted.
- Click Brush options to determine how the inside of the object is filled.

1.7.4.7 Ellipse



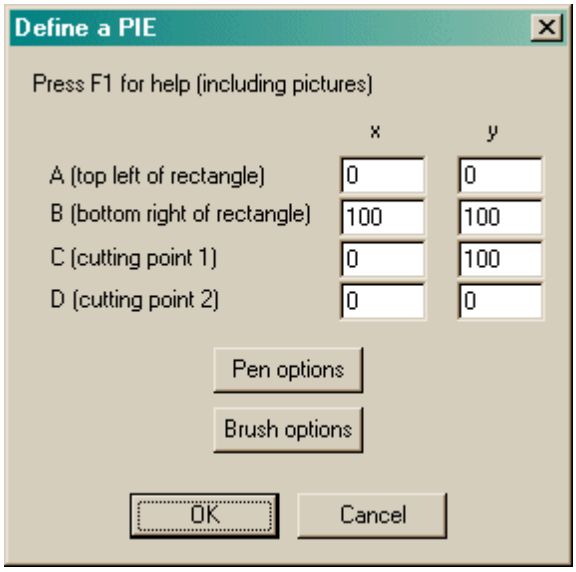
- The meaning of the points is explained in the figure above.
- Click Pen options to determine how the edge of the object is painted.
- Click Brush options to determine how the inside of the object is filled.

1.7.4.8 Line



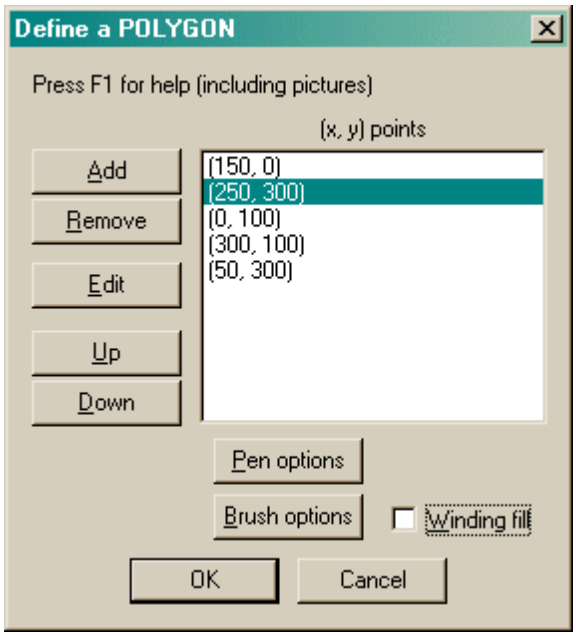
- The meaning of the points is explained in the figure above.
- Click Pen options to determine how the edge of the object is painted.

1.7.4.9 Pie



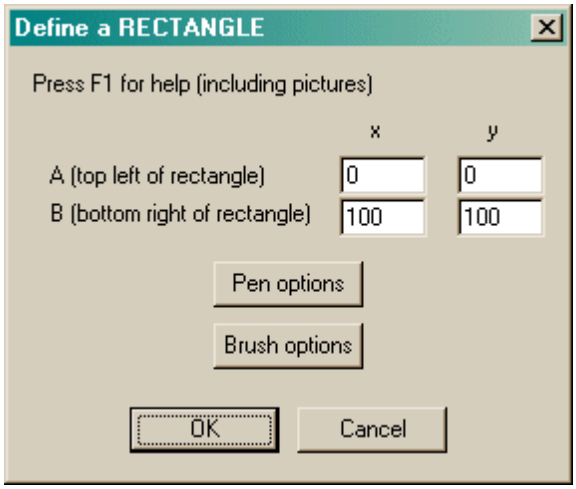
- The meaning of the points is explained in the figure above.
- Click Pen options to determine how the edge of the object is painted.

1.7.4.10 Polygon



- The meaning of the points is explained in the figure above. You need at least three points for a polygon component.
- Click **Add** or **Remove** to add points to the polygon or remove them.
- Click **Edit** to alter a point.
- Click **Up** or **Down** to re-order the points.
- The meaning of the rather complicated **winding/alternate fill setting** is also explained in the figure above.
- Click Pen options to determine how the edge of the object is painted.
- Click Brush options to determine how the inside of the object is filled.

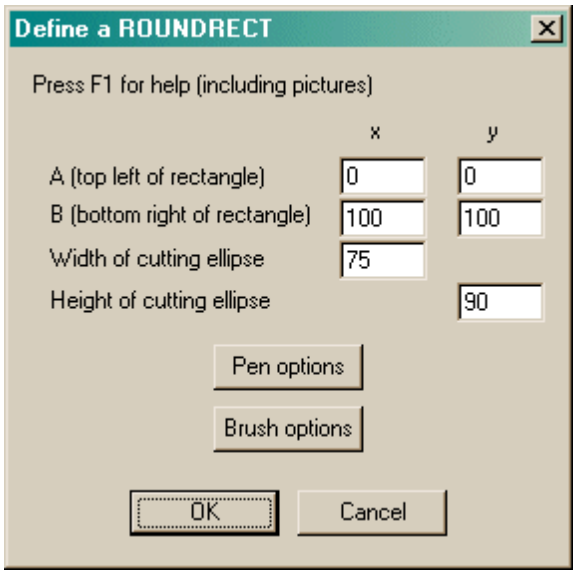
1.7.4.11 Rectangle



- The meaning of the points is explained in the figure above.
- Click Pen options to determine how the edge of the object is painted.

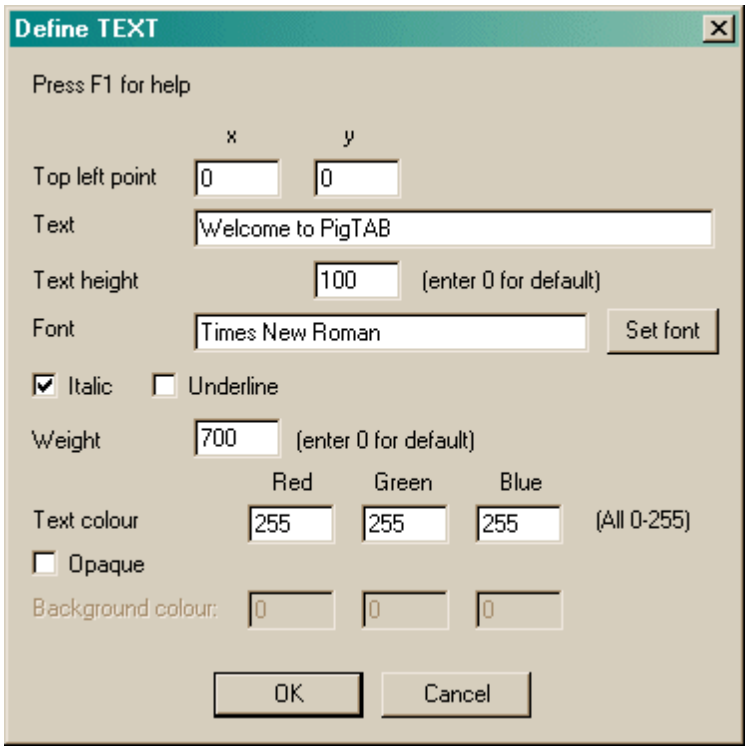
- Click Brush options to determine how the inside of the object is filled.

1.7.4.12 Rounded rectangle



- The meaning of the points is explained in the figure above.
- Click Pen options to determine how the edge of the object is painted.
- Click Brush options to determine how the inside of the object is filled.

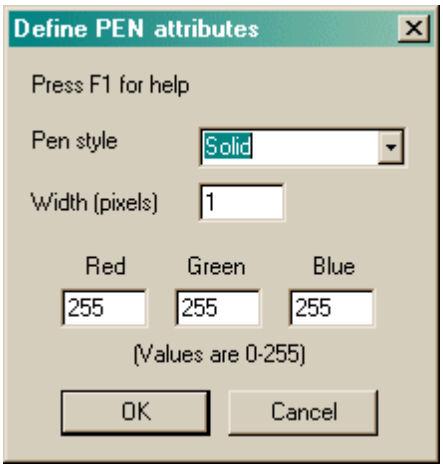
1.7.4.13 Text



- Choose the **(x, y) coordinates of the top left point.** This is normally (0,0).
- Set the **text** itself.
- Set the **text height** (in pixels, not points), or use 0 for a default setting.
- Choose the **font name** to be used by the server.
- Choose whether the font should be **italic** or **underlined.**
- Choose a font **weight** (equivalent to "boldness", and ranging from 1-1000), or use 0 for a default weight.
- Choose the **text colour.** Bear in mind that picking a black font (as would be common for wordprocessing and the like) will make the font invisible on the default black background of Whisker screens. Alter the background colour, place the text on another object, or change the font colour.
- Many of the settings listed above can be set by clicking the **Set font** button, which lists the fonts available on your system.
- Choose whether the font should be **opaque** or not. If it is opaque, the gaps between the letters are filled in with a **background colour,** in which case you may choose this too.

1.7.4.14 Pen options

Pens draw round the edges of components.



- Choose a **pen style.** The options *solid, dash, dot, dashdot,* and *dashdotdot* should be fairly self-explanatory. *Null* gives an invisible pen. *Insideframe* is relevant when the pen is thick; for example, if you draw a circle of diameter 100 units with a pen of width 20 units, the circle will normally end up having an external diameter of 120 units and an internal diameter of 80 units (as the pen overlaps by 10 units on the inside and the outside of the circle). If you specify insideframe, the circle's outside diameter is 100 units in this situation.
- Choose a **pen width** in pixels.
- Choose a **pen colour** by specifying values from 0-255 for the red, green, and blue components of the colour.

1.7.4.15 Brush options

Brushes are used to fill the insides of solid components with colours or patterns.

- Choose a **brush style.** This may be *hollow* (invisible), *solid*, or *hatched* (in which case you can specify the hatching style and colour).
- For solid and hatched brushes, choose the **hatching colour.**
- If you select a hatched brush, choose the **hatching style.** The hatching styles are *back diagonal* (lines at 45° anticlockwise from the horizontal axis), *cross* (horizontal and vertical lines); *diagcross* (lines at 45° clockwise and anticlockwise from the horizontal); *fdiagonal* (lines at 45° clockwise to the horizontal); *horizontal* (horizontal lines); *vertical* (vertical lines).
- A hatched brush may either be **opaque or transparent.** If it is transparent, you can see through the hatching to whatever is beneath. If it is opaque, you may set the **background colour** used to fill in the gaps in the hatching.

1.7.5 Predefined stimuli

MonkeyCantab is supplied with a set of built-in stimulus sets. They are:

	Stimulus name	where ***X*** is from 1 to...
• University of Cambridge ID/ED stimuli: shape set,	univcam_IDED_shape_***X***	54
line set.	univcam_IDED_line_***X***	44
• Cambridge Cognition DNMTS stimuli: sets 0, 5, 6,	camcog_mdms0_***X***	430
7.	camcog_mdms5_***X***	469
	camcog_mdms6_***X***	16
	camcog_mdms7_***X***	10
• Cambridge Cognition PAL stimuli: sets 0 to 5.	camcog_pal0_***X***	67
	camcog_pal1_***X***	17
	camcog_pal2_***X***	17
	camcog_pal3_***X***	17
	camcog_pal4_***X***	17
	camcog_pal5_***X***	18
• Cambridge Cognition 'STAR' stimuli: sets 0 to 4.	camcog_star0_***X***	48
	camcog_star1_***X***	48
	camcog_star2_***X***	48

- Cambridge Cognition ID/ED stimuli: sets 0 to 8, both of which have a shape subset and a line subset.

camcog_star3_***X***	48
camcog_star4_***X***	48
camcog_nedi0shape_***X***	6
camcog_nedi0line_***X***	6
camcog_nedi1shape_***X***	6
camcog_nedi1line_***X***	6
camcog_nedi2shape_***X***	6
camcog_nedi2line_***X***	6
camcog_nedi3shape_***X***	6
camcog_nedi3line_***X***	6
camcog_nedi4shape_***X***	6
camcog_nedi4line_***X***	6
camcog_nedi5shape_***X***	6
camcog_nedi5line_***X***	6
camcog_nedi6shape_***X***	6
camcog_nedi6line_***X***	6
camcog_nedi7shape_***X***	6
camcog_nedi7line_***X***	6
camcog_nedi8shape_***X***	3
camcog_nedi8line_***X***	2

These stimulus sets can be used in a number of ways.

- You can make your own copy of them by importing them into your configuration file (giving them a name of your choice), and then edit them for your own needs. See The Visual Object Library.
- You can refer to them directly, whereever you are asked to supply a stimulus name, using the names shown above. For example, the first stimulus in the University of Cambridge ID/ED shape set is called **univcam_IDED_shape_1.** (Please note that if you create your own stimulus with the same name as a predefined stimulus, MonkeyCantab will choose your version.)
- You can use whole sets of stimuli in tasks that require large numbers of stimuli, such as the D (N)MTS and PAL tasks. These tasks can **manipulate** the stimuli, to generate a much larger set of potential stimuli (for example, by altering the colours of each quadrant).

Technical details of the stimulus-generating techniques used

- The PAL task varies the stimuli by altering the colour of each quadrant; since there are 4 quadrants per stimulus and 7 possible colours (red, green, yellow, blue, magenta, cyan, white), there are 7^4 = 2401 variants on each stimulus (giving, for the PAL0 stimulus set, 160,867 possible stimuli in total). You can refer to them directly; for example, variant 817 of stimulus 65 of the PAL 0 set is called **camcog_pal0_65_PV817. Variants are numbered from 0.**

- The DNMTS task varies stimuli in several ways; a single varant number is interpreted according to the variation system required. Stimuli are created in groups of four (even if not all four are to be used on a given trial) using the specified stimulus, plus the three that follow it in the set (wrapping round to the start if need be).
  - **(1)** If shape quadrants are to be shuffled, then the quadrants are exchanged between stimuli, keeping them in the same place (e.g. top right), in a manner that gives 64 ways of generating 4 stimuli from the 4 stimuli that we started with.
  - **(2)** Colour reassignment is applied. This can be:
    - **unmodified (UN)** (don't change a thing). *If shapes are also not shuffled, this is intended to correspond to the old DOS Monkey CANTAB scheme of "STIMULUS DIRECT".*
    - **vary colours (VC)** (alters the colours of each quadrant of each of the 4 stimuli in the same way; as there are 7 usable colours, this gives 7*7*7*7 = 2401 further variations).
    - **monochrome, shape-only discrimination (MS)** (all stimuli are made the same colour; there are 7 possible colours) [remember, monochrome means one colour, not black-and-white]. The colour is that of the top-left quadrant of the first shape in the group (shifted by

the value of the variant). *This is intended to correspond to the old DOS Monkey CANTAB scheme of "MONOCHROME / SHAPE".*

- **monochrome, shape-only discrimination, fixed colour (MF)** (all stimuli are made the same colour; the user specifies this colour) [remember, monochrome means one colour, not black-and-white]. *This option is new as of Jan 2007.* The colours are as follows (with red/green/blue components shown on a scale of 0-255):

  - colour 0 = black (R 0, G 0, B 0)
  - colour 1 = red (R 255, G 0, B 0)
  - colour 2 = green (R 0, G 255, B 0)
  - colour 3 = yellow (R 255, G 255, B 0)
  - colour 4 = blue (R 0, G 0, B 255)
  - colour 5 = magenta (R 255, G 0, B 255)
  - colour 6 = cyan (R 0, G 255, B 255)
  - colour 7 = white (R 255, G 255, B 255)
  - colour 8 = orange (R 255, G 165, B 0)
  - colour 9 = indigo (R 75, G 0, B 130)
  - colour 10 = violet (R 238, G 130, B 238)
  - colour 11 = moccasin (R 255, G 228, B 181)
  - colour 12 = light slate grey (R 119, G 136, B 153)
  - colour 13 = saddle brown (R 139, G 69, B 19)

  There is only one colour variant, because only one colour can be used at one time (but the user can specify which colour this is).

- **monochrome, colour-only discrimination (MC)** (all stimuli are made the same shape; all stimuli are monochrome [meaning one colour, not black-and-white] but the colour of each stimulus in the set of four is different; there is only 1 possible variation with this theme before colours are repeated, which can't be done as it would make at least one stimulus non-unique in the full set). *This is intended to correspond to the old DOS Monkey CANTAB scheme of "MONOCHROME / COLOUR".*
- **monochrome, mixed (MM)** (one distractor has a different shape and a different colour to the target; one has the same shape but a different colour; one has the same colour but a different shape. There are seven available colours, so 3 variations of two colours each). *This is intended to correspond to the old DOS Monkey CANTAB scheme of "MONOCHROME / MIXED".*
- **seven colours per trial (SC)** (the target uses four colours; the incorrect stimulus uses the three remaining unused colours and one colour that overlaps with the target stimulus, though the overlapping colour is never in the same quadrant in the incorrect stimulus as it is in the target). The target stimulus can have up to 7*6*5*4 = 840 colour variations, so that limits the number of trial-unique stimuli; the further variations in the distractor (namely 4 possible choices of overlapping colour, 3 possible choices of quadrant to place that overlapping colour, and 3*2*1 possible arrangements of the unique colours in the distractor) do not contribute to trial uniqueness and are chosen in a way that is deterministic but tends to vary from trial to trial (specifically, the variant number is re-used to determine these, so these parameters covary with the colour scheme of the target; this is not obvious to the user!). *18-Oct-2004: this scheme, when used with one target and only the first incorrect stimulus, is intended to correspond to the old DOS Monkey CANTAB scheme of "PAIRED STIMULI", as used by Weed et al. (1999) Cognitive Brain Research 8: 185. With the MDMS5 stimulus set (469 stimuli), this gives 840*469/2 = 196,980 unique trials with two stimuli per trial. It can also be used with four stimuli per trial; a second pair of stimuli, related in the same way as the target/first incorrect stimulus, are added to the set. These extra stimuli will not necessarily share exactly one colour with the target.* ***Please note that this scheme does not work well with 3 stimuli per trial (one target plus two incorrect stimuli),*** *as the target is picked at random from the first three stimuli - so there is no guarantee that it will share exactly one colour with the incorrect stimuli.*

- Therefore, since DMTS stimulus set 0 has 430 stimuli in it, the most variable stimulus set you can generate (using quadrant shuffling plus full colour variation) is 430 x 64 x 2401 = 66,075,520 stimuli. If four stimuli are used on each trial, this gives you 16,518,880 possible unique trials. I feel that should be enough.
- DMTS-shuffled stimuli can be referred to by appending _**DM,** then _**SS** for shape quadrant shuffling or _**NS** for no shuffling, then _UN, _VC, _MS, _MC, _MM, or _SC for one of the colour variation schemes, and then a variation number, then "_", and then which of the four generated stimuli you want. So the fourth stimulus within the set generated by the last unique shape-shuffled, colour-varying variation on stimulus 200 of DMTS set 0 might be **camcog_mdms0_200_DM_SS_VC153663_3.** *An exception: the _MF scheme takes an additional parameter: the number immediately after "MF" is the colour, and then an underscore follows, and then the variant number and so on (e.g. camcog_mdms0_200_DM_SS_MF7_0_3, where 7 is the colour and 0 is the variant).* **Please note** that as the stimuli are generated in sets of 4, you may end up with *non-unique* stimuli if you generate further sets that overlap. For example, if you generate a set of 4 stimuli based on stimulus 1 (which involves looking at stimuli 2, 3, and 4), and you show all the stimuli to the subject, then to guarantee unique stimuli you should request the next four starting with stimulus 5, not stimulus 2. **Variants, and stimuli within a set of four, are numbered from 0.**

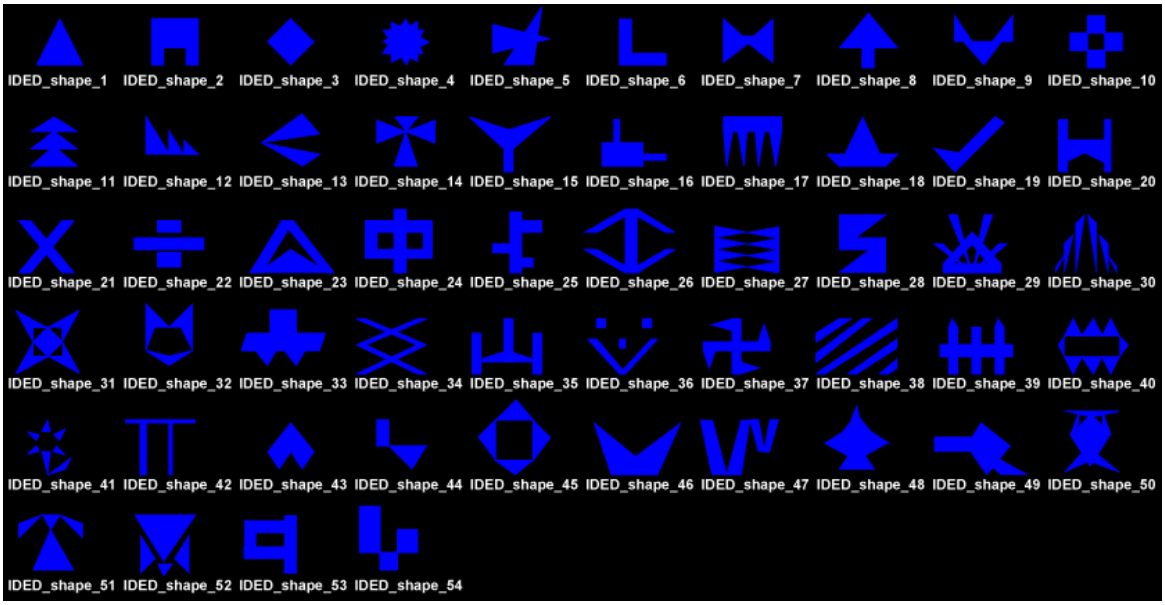
Stimulus provenance and copyright

- The University of Cambridge ID/ED stimuli were developed by A.C. Roberts and colleagues at the University of Cambridge (e.g. Dias R, Robbins TW, Roberts AC, 1996, *Nature* **380:** 69; Dias R, Robbins TW, Roberts AC, 1997, *J. Neurosci.* **17:** 9285). The versions presented here are the stimuli in use at the University of Cambridge in 2003. To the extent that these stimuli are copyrightable (one, for example, is a single horizontal line, which probably cannot be the subject of copyright), **copyright** to some of these stimuli may have been transferred from the creators to Cambridge Cognition Ltd (see below). However, shapes 22, 51, 52, 53, and 54, and lines 1, 22, 28, 41, 42, 43, and 44 are not part of Cambridge Cognition's stimulus set, and copyright to these stimuli probably lies with the creators.
- The Cambridge Cognition ID/ED stimuli are a subset of the University of Cambridge ID/ED stimuli. Cambridge Cognition used these stimuli under licence from A.C. Roberts, T.W. Robbins, and colleagues, and probably hold the **copyright** to these stimuli.
- **Copyright** to the Cambridge Cognition PAL, D(N)MTS, and STAR stimuli is held by Cambridge Cognition (to the extent that such stimuli are copyrightable, as always).

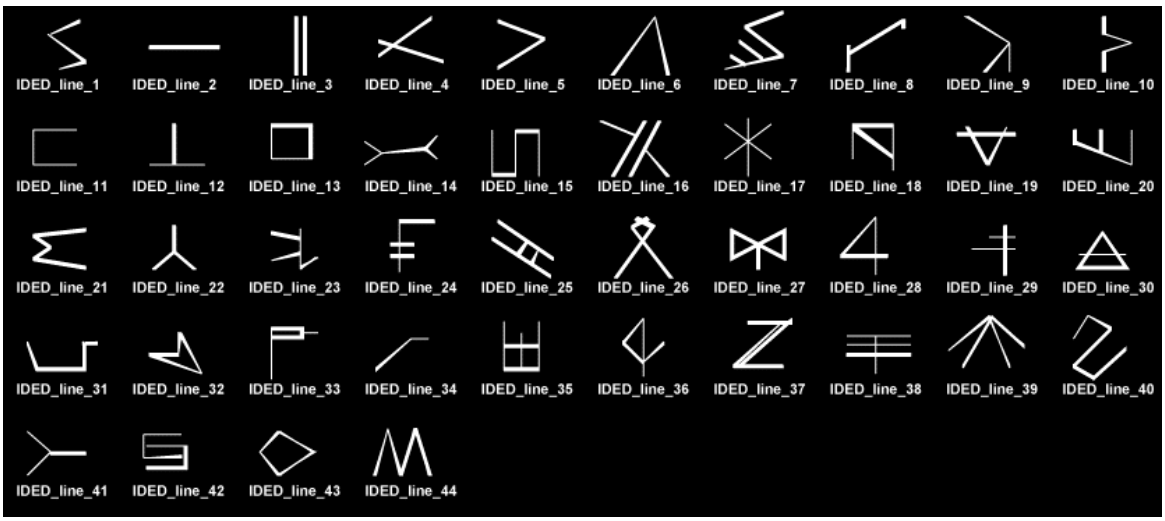
1.7.5.1 University of Cambridge ID/ED stimuli

Note: the stimuli shown here are at a lower resolution than those seen on the screen during the task. All stimuli should be prefixed by univcam_ to refer to them directly in MonkeyCantab.

Shapes:



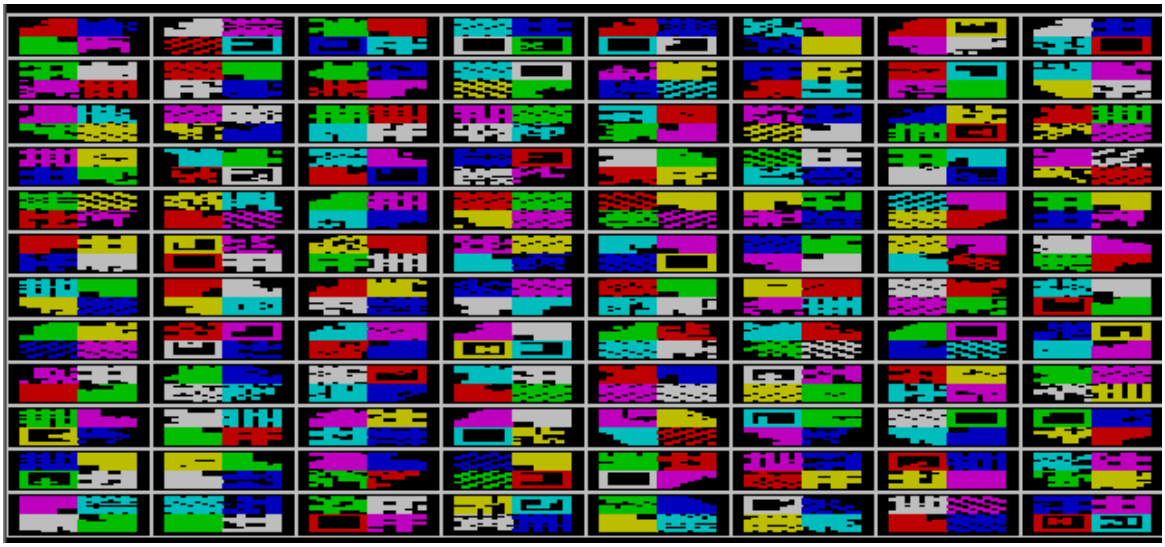
Lines:



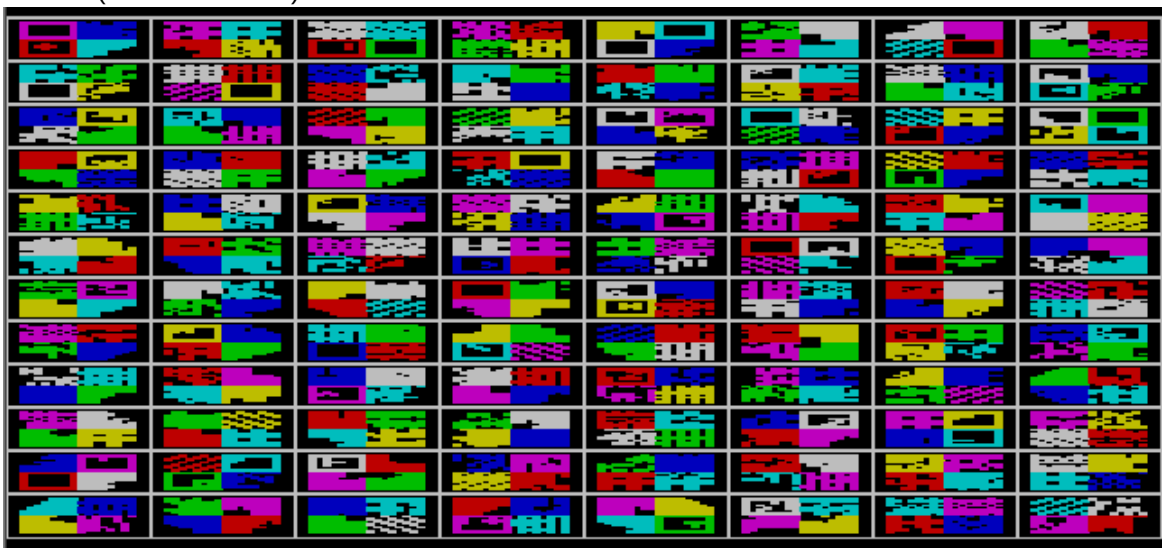
1.7.5.2 Camcog D(N)MTS stimuli

Note: the stimuli shown here are at a lower resolution and colour contrast than those seen on the screen during the task.

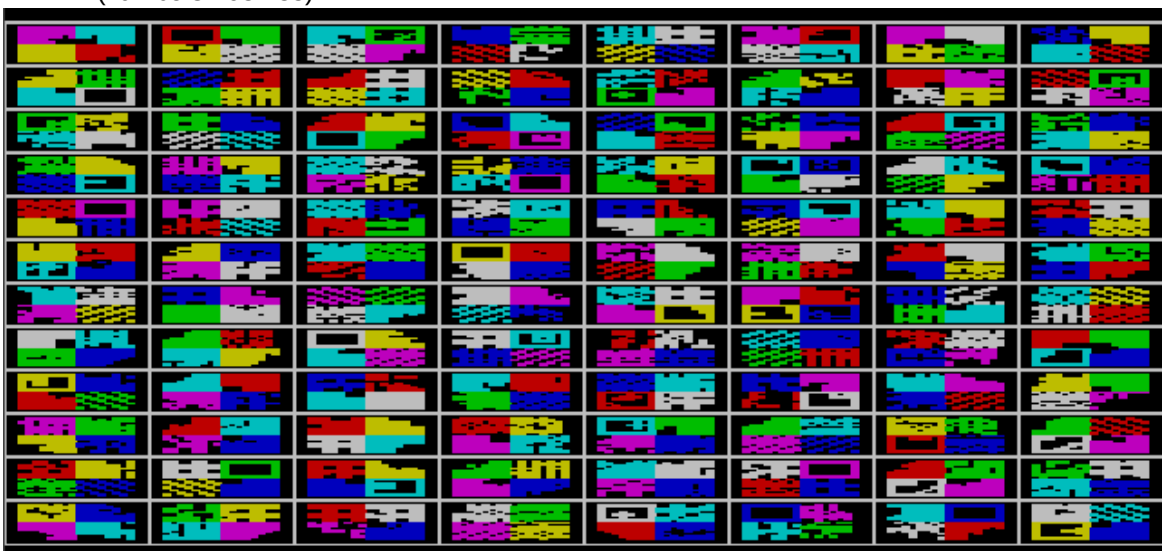
D(N)MTS stimulus set 0:
(numbers 1-96)



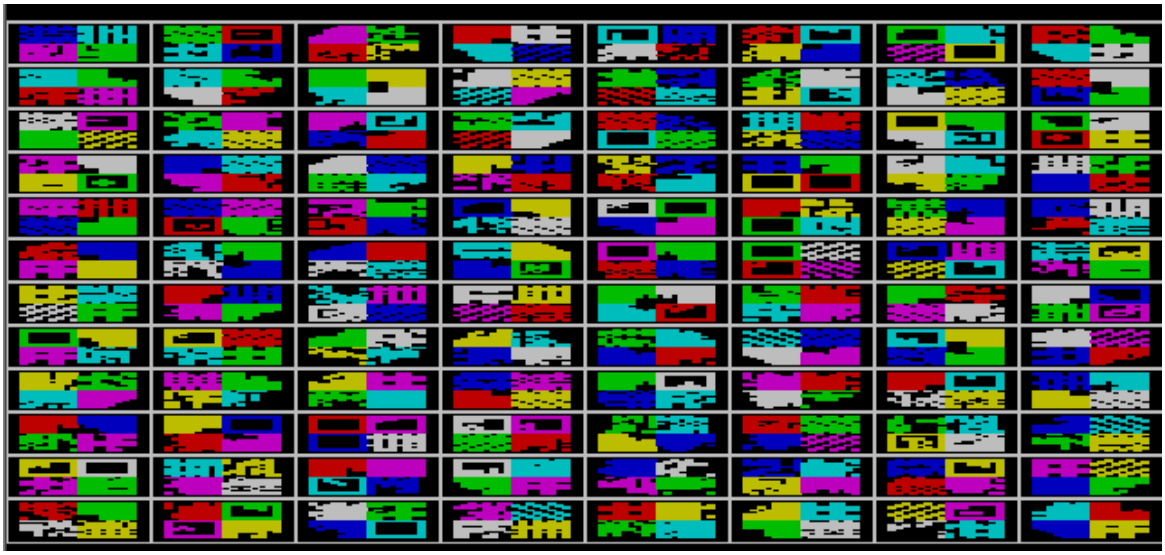
(numbers 97-192)



(numbers 193-288)



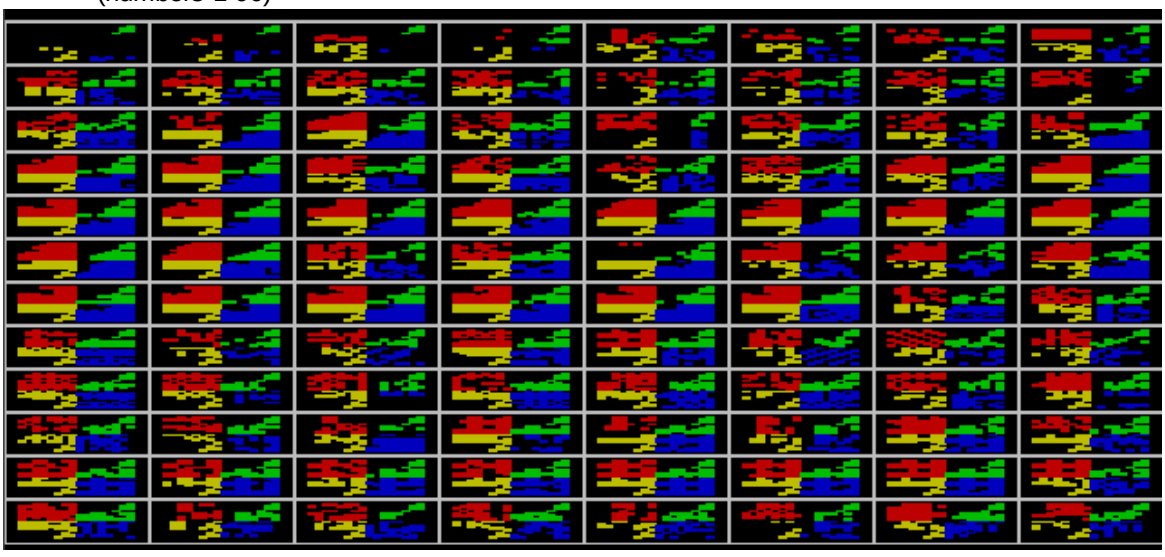
(numbers 289-384)



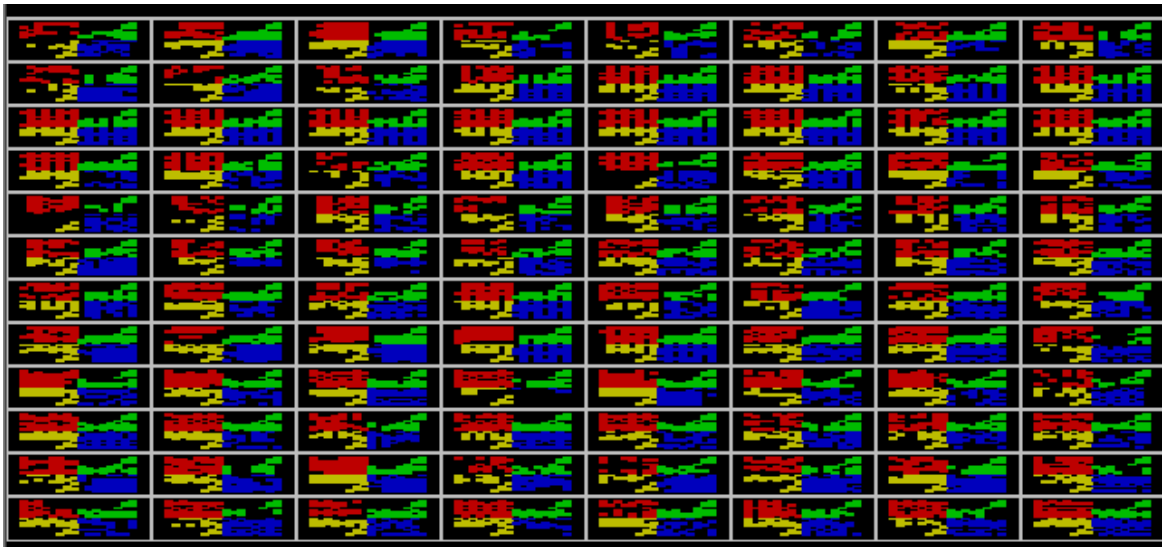
(numbers 385-430)

D(N)MTS stimulus set 5:

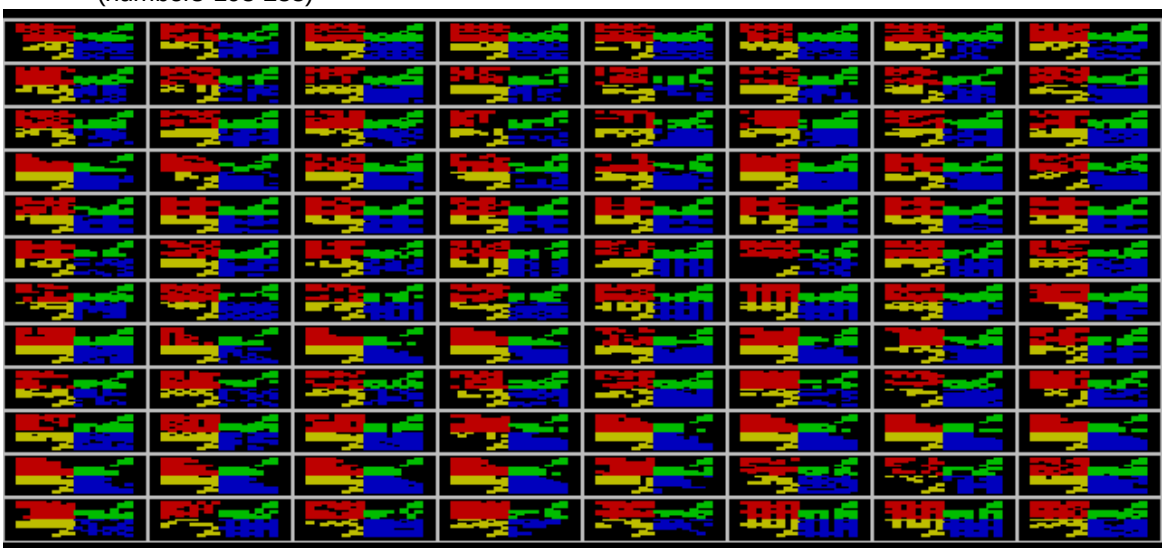
(numbers 1-96)



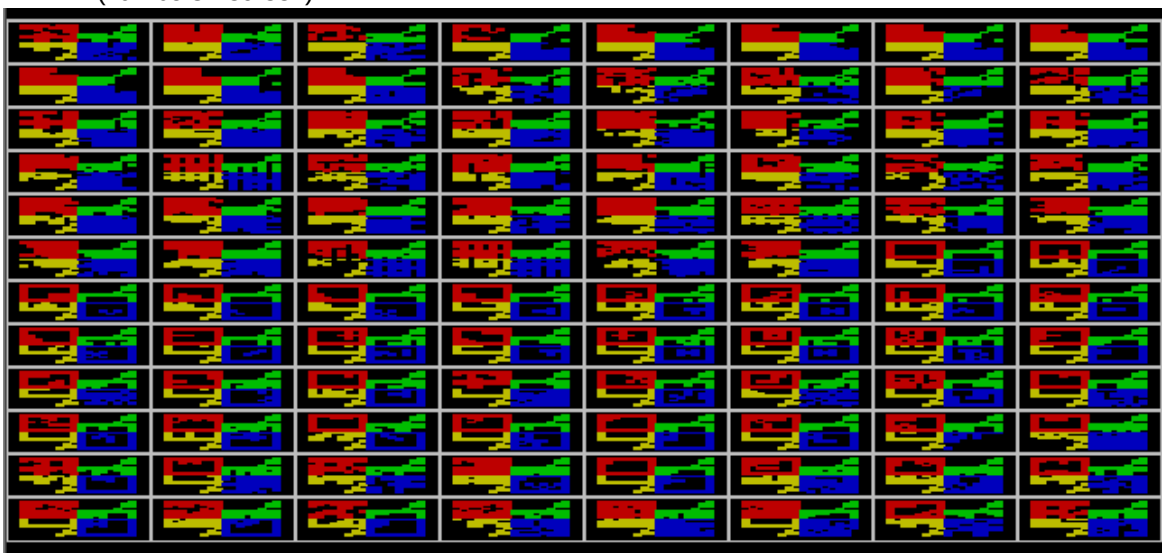
(numbers 97-192)



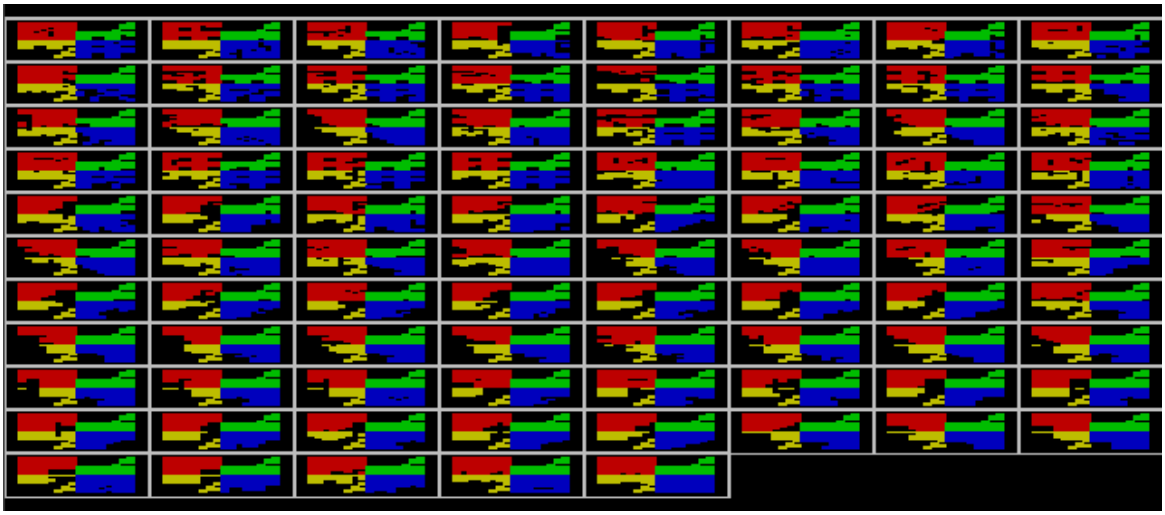
(numbers 193-288)



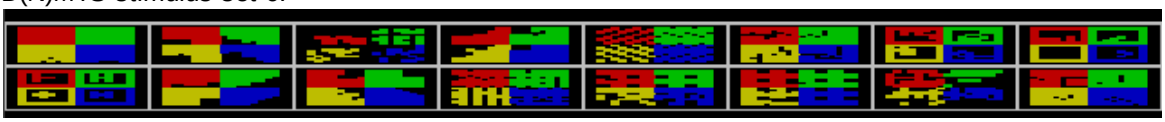
(numbers 289-384)



(numbers 385-469)



D(N)MTS stimulus set 6:

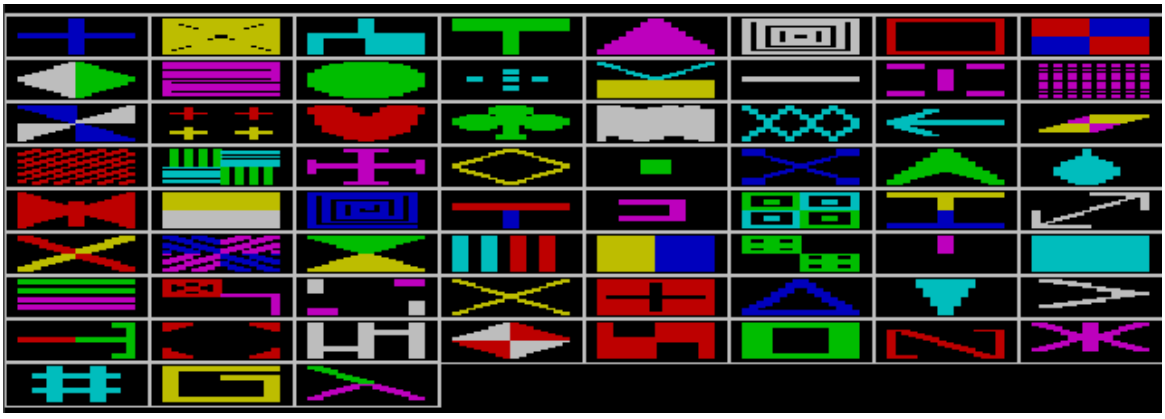


D(N)MTS stimulus set 7:

1.7.5.3 Camcog PAL stimuli

Note: the stimuli shown here are at a lower resolution and colour contrast than those seen on the screen during the task.

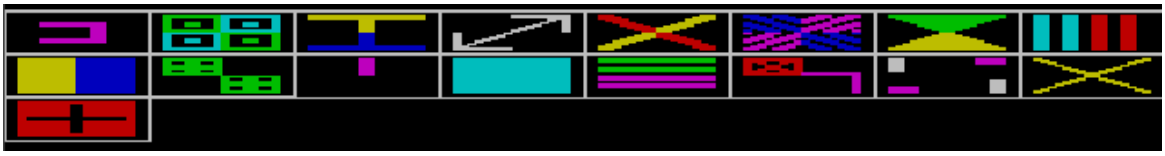
PAL stimulus set 0:



PAL stimulus set 1:

PAL stimulus set 2:

PAL stimulus set 3:



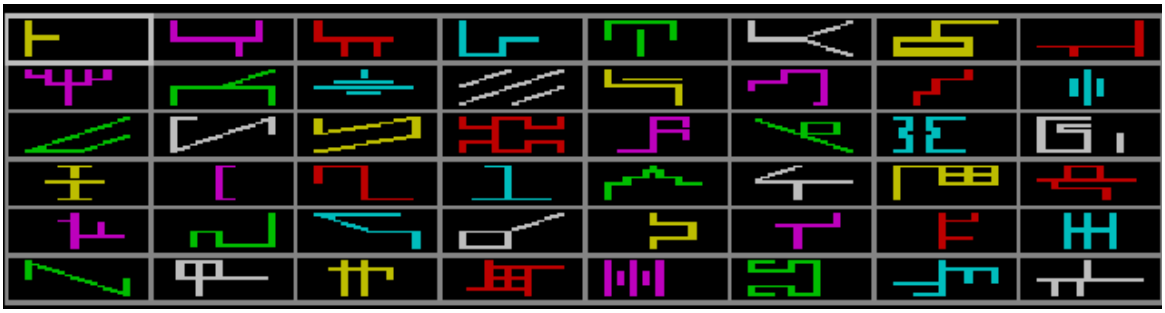
PAL stimulus set 4:

PAL stimulus set 5:

1.7.5.4 Camcog STAR stimuli

Note: the stimuli shown here are at a lower resolution and colour contrast than those seen on the screen during the task.

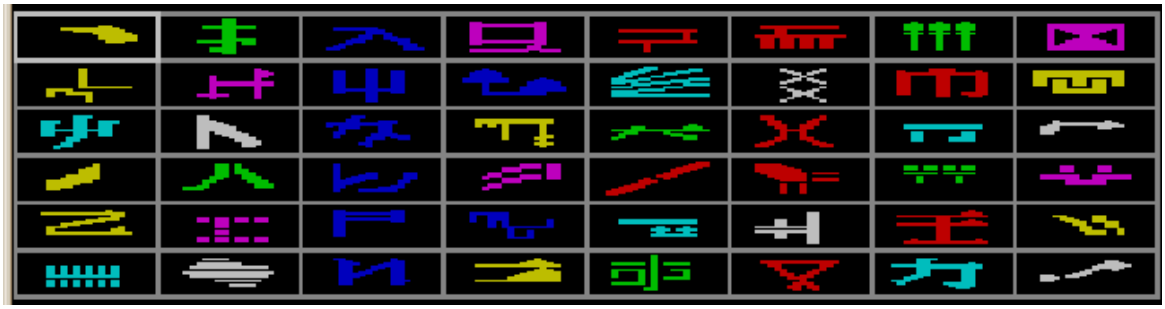
STAR stimulus set 0:



STAR stimulus set 1:

STAR stimulus set 2:

STAR stimulus set 3:

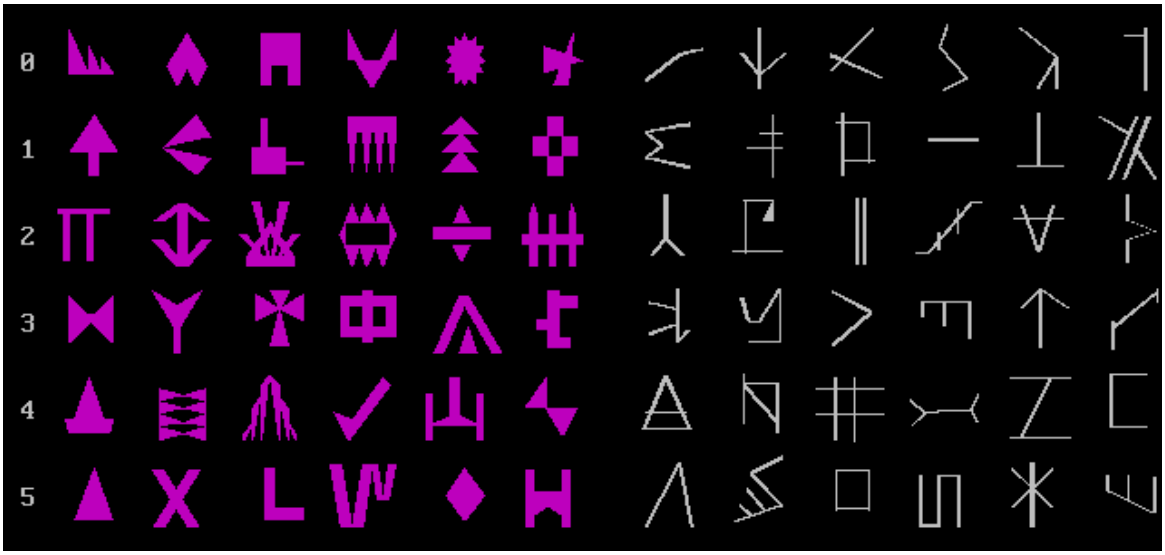


STAR stimulus set 4:

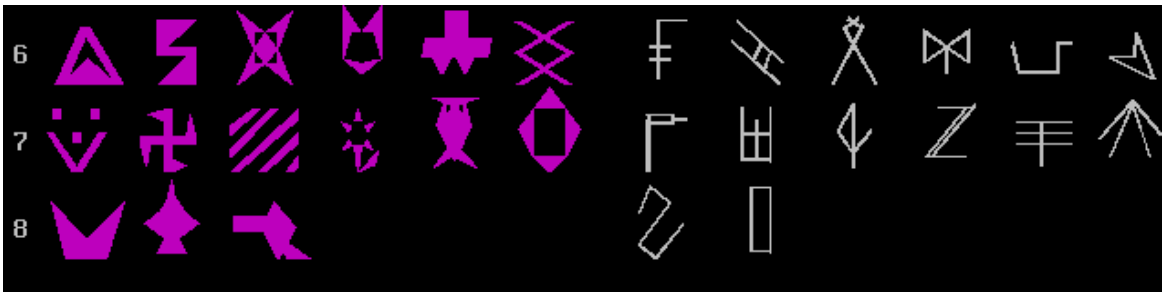
1.7.5.5 Camcog ID/ED stimuli

Note: the stimuli shown here are at a lower resolution and colour contrast than those seen on the screen during the task.

ID/ED stimulus sets 0 to 5:



ID/ED stimulus sets 6 to 8:



1.8 Configuring individual tasks

Choose a task to see how to configure it:

- Reinforcement Familiarization
- Touch Training
- Visual Discriminations and Set Shifting (ID/ED task)
- Reversal Learning
- Delayed Matching/Non-matching to Sample (D[N]MTS)
- List-based Delayed Matching/Non-matching to Sample (ListDMS)
- Self-Ordered Search (SOS) (a.k.a. Spatial Working Memory, SWM)
- Five-Choice Serial Reaction Time (5CSRTT)
- Paired-Associates Learning (PAL)
- Simple Schedules of Reinforcement
- Impulsive Choice

1.8.1 Reinforcement Familiarization

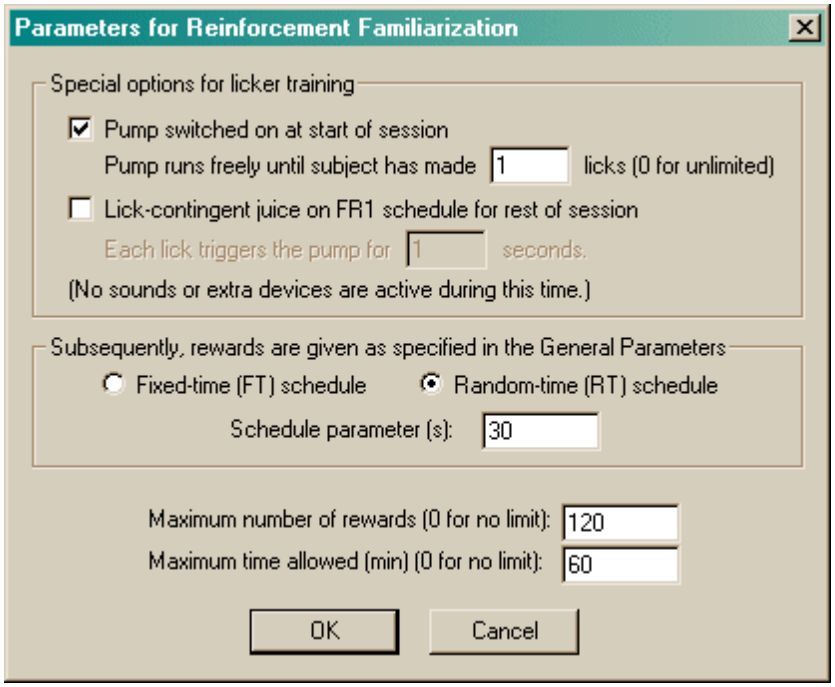
About the task

Purpose: to train the subject to take rewards, and associate them with a sound.

The task presents a sound and reward (in the Copenhagen lab, typically an M&M® chocolate pellet) together, at random intervals

Configuring the task

What constitutes "a reinforcer" is defined in the General Parameters. This allows you to set the reward sound and define what sound is associated with the presentation of a reward.



- **Special options for licker training.** These options are only applicable if you have a pump delivering liquid reinforcement and a lick sensor. If you wish, you can proceed through two initial stages before the main schedule starts. These extra options do NOT use the reward parameters specified in the General Parameters - for example, no sounds are played. If you select one or more of these options, they are triggered in the order they are listed in the parameters dialogue box: (1) pump switched on... (2) lick-contingent juice on FR1 schedule... (3) FT/RT schedule.
  - **Pump switched on at start of session.** If selected, the pump is activated without a pause at the start of the session, until the subject has made the number of licks specified in the **licks** box. At this point, the pump is switched off. If you specify zero licks here, the pump will run for the *entire* session and the program will never proceed to another phase.
  - **Lick-contingent juice on FR1 schedule for rest of session.** This option allows you to reward licking for the rest of the session. Each lick activates the pumps for a fixed time (specified in the **Each lick triggers...** box).

- **Schedule parameter (s).** Now begins the proper schedule; all rewards delivered here follow the settings specified in the General Parameters.
  - **Fixed-time schedule.** Reward is delivered at regular intervals. Reward is delivered; the program waits for the end of reinforcement; this parameter specified how long it will *then* wait before delivering the next reward. (Note that this is not quite the standard parameter of an FT schedule: the standard way is to specify the time between the start of reward 1 and the start of reward 2; here, this parameter specifies the time between the end of reward 1 and the start of reward 2. This is an easy way to prevent the second reinforcer being scheduled while the first is still being delivered! You can think of this parameter as the inter-reinforcement interval.)
  - **Random-time schedule.** An imaginary clock ticks once a second. Every clock tick, the program flips a coin that will turn up heads with a probability of 1/*parameter*. If the coin turns up heads, a reinforcer is delivered. Thus, if the parameter is 30, reinforcement is delivered with a probability of 1/30 every second, meaning that on average reinforcement is delivered once every 30 seconds. (However, for as long as it takes to deliver the reinforcer

or the reinforcer-associated sound, the clock stops ticking in order to prevent another reinforcer being delivered at the same time - if you need a precise RT schedule, you must take this time into account too.)

- **Maximum number of rewards.** Once this number of rewards has been delivered, the task stops. (Specify 0 for no limit.) This number includes rewards earned on the FR1 licking part of the task, but not on the "free juice" part.
- **Maximum time permitted during task.** Once this time limit has been reached, the task stops. (Specify 0 for no limit.)

You must specify either a maximum number of rewards, or a maximum time, or both.

The houselight is on during the task.

Mimicking previous Arachnid programs

Notes for the University of Cambridge:

LICKER

Old program: Juice drips until the monkey first licks. Then either
(a) juice is delivered for 1s whenever the monkey licks - if it licks during that 1s, juice is prolonged - or
(b) juice is delivered continuously ("free juice") for the whole session.

To mimic these:
(a) pump switched on at start of session; pump runs freely until the monkey has made 0 licks (i.e. unlimited);
(b) pump switched on at start of session; pump runs freely until the monkey has made 1 lick; lick-contingent juice on FR1 schedule for rest of session (each lick triggers the pump for 1s).

TONE

Old program: Juice-tone pairings. Fixed-time schedule. Turn juice on, wait for 16 licks, then turn the juice off. Subsequently present tone/juice pairings on FT schedule. When tone is on, either (a) juice is delived when monkey licks or (b) juice drips all the time.

To mimic this: *Pump switched on at start of session until subject has made 16 licks; subsequently, FT schedule of reinforcement. Under the General Parameters, choose either (a) pump contingent on licking, or (b) not.*

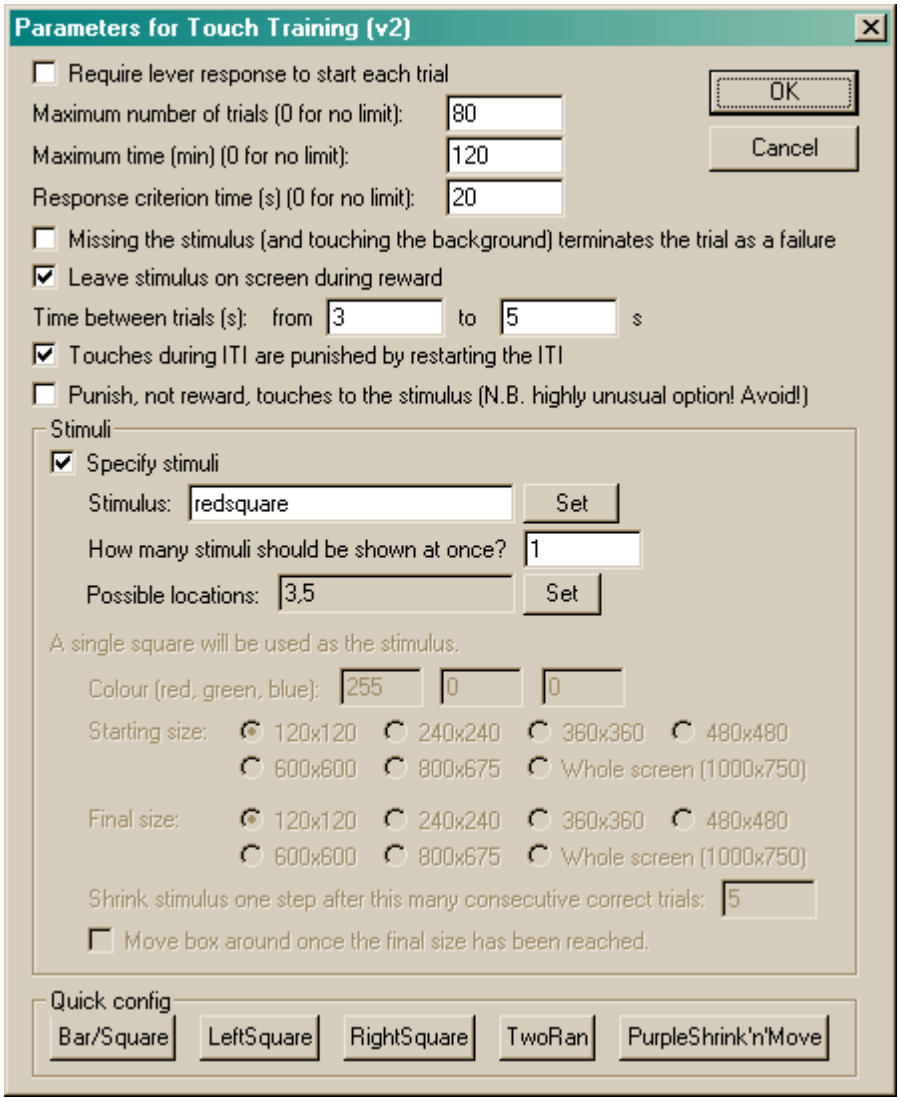
1.8.2 Touch Training

About the task

Purpose: to train the subject to touch stimuli.

A trial begins and marker sound 1 is played. Object(s) are presented on the screen. If the subject responds correctly within a criterion time, reward it. Punish the subject if it fails to respond in time (if you've set a time limit), or if it misses the stimulus and touches the background instead (if you've chosen to punish this).

Configuring the task



- **Require lever response to start each trial.** Requires that the subject make a lever response to initiate each trial. See Use with Dogs.
- **Maximum number of trials.** When the subject has performed this number of trials, the task ends. (You may specify 0 for no limit, though you must specify a limit on the number of trials, the time, or both.)
- **Maximum time.** When this time elapses, the task is terminated as soon as the current trial has finished. (You may specify 0 for no limit, though you must specify a limit on the number of trials, the time, or both.)
- **Response criterion time.** If the subject fails to respond to a stimulus within this time, an omission is scored.
- **Missing the stimulus terminates the trial as a failure.** If ticked, the trial is terminated if the subject touches the background. Otherwise, the subject can touch the background first and later touch the stimulus, or (if Strict Touches are not selected in the General Parameters) can 'slide' onto the stimulus.
- **Leave stimulus on screen during reward.** Pretty self-explanatory, I hope...
- **Time between trials.** Specify a minimum and a maximum intertrial time (they may be the same). The actual time is chosen with a rectangular probability distribution within these values. The time between trials starts **after** any reward or punishment from the previous trial has finished.
- **Touches during ITI are punished by restarting the ITI.** Fairly obvious.
- **Punish, not reward, touches to the stimulus.** If ticked, touching the stimulus delivers

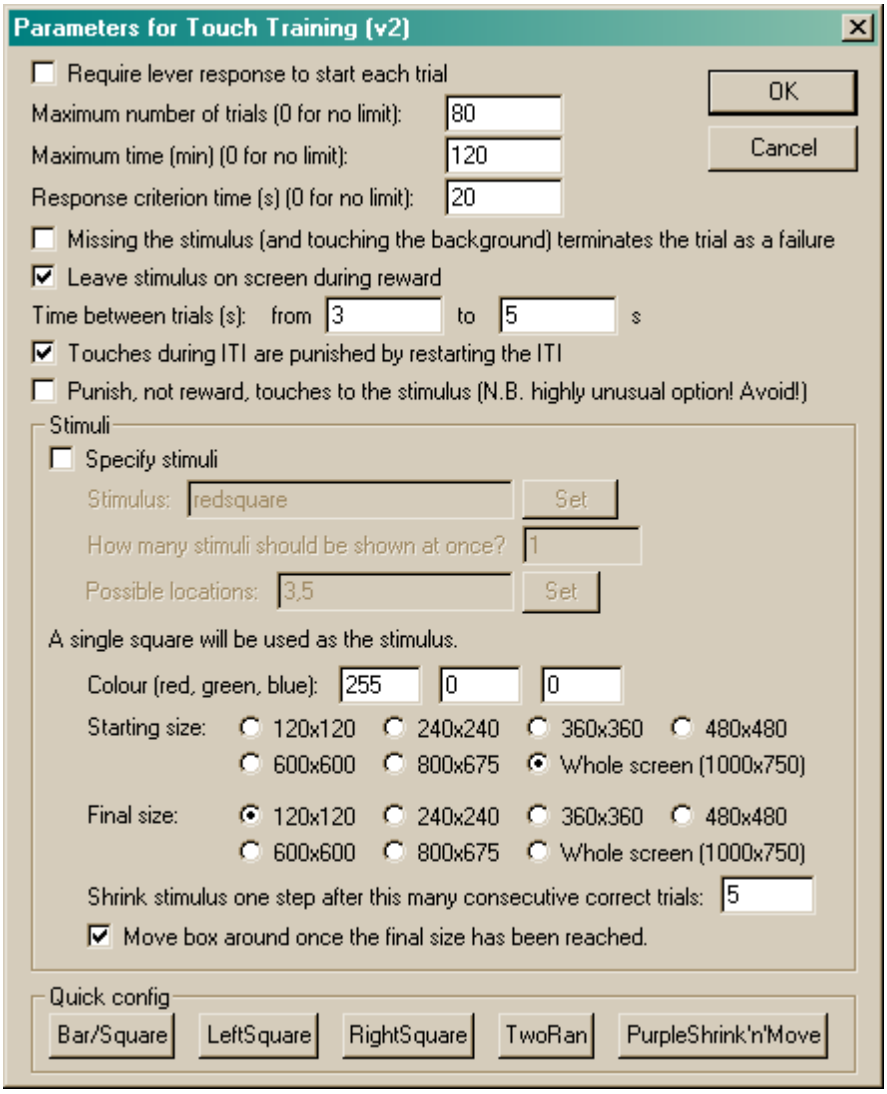
punishment (defined in the General Parameters) rather than reward, even though the response is marked as the "correct thing to do" in the database. *Highly unusual - avoid! Implemented for a very specific extinction-like experiment, **not** for general touch training.*

Stimuli

- **Specify stimuli.** If ticked, you specify the stimuli exactly. If not, a square is used (and it can shrink as time goes by).

  Options available if "Specify stimulus" is ticked:
  - **Stimulus.** Choose the stimulus (click **Set** to select from the list of available stimuli).
  - **How many stimuli should be shown at once?** Self-explanatory.
  - **Possible locations.** Click **Set** to choose the set of locations at which the stimulus (or stimuli) should be presented. Locations 0-8 represent nine equally-sized regions of the screen; see the map in Size and coordinates (nine-way grid) or the map shown when you click **Set.**

  Options available if "Specify stimulus" is not ticked:
  - **Colour (red, green, blue).** Specify the square's colour. Each value (red, green, blue) can be from 0 to 255.
  - **Starting size.** Specify the size that the stimulus starts at.
  - **Final size.** Specify the size that the stimulus finishes at.
  - **Shrink stimulus... after this many consecutive correct trials.** When the subject performs this many trials correct in a row, the stimulus shrinks one step (until it reaches the final size).
  - **Move box around once the final size has been reached.** If this is ticked, then once the stimulus has reached the final size and the subject performs *n* correct trials in a row (where *n* was defined in *Shrink stimulus...*) then the stimulus starts to move around randomly on each trial.



Quick configuration; mimicking previous Arachnid and CamCog programs

Click **Bar/Square, LeftSquare, RightSquare, TwoRan**[dom], or **PurpleShrink'n'Move** to choose a predefined scheme. The first four mimic the University of Cambridge training programs TSCR-BAR, SQUARE, LEFTSQUARE, RIGHTSQUARE, and TWORAN, though you still need to choose the stimulus to use. For exact compatibility with these programs, use a wide bar stimulus for Bar/Square and a square for the others. The last (PurpleShrink'n'Move) mimics Cambridge Cognition's previous edition of MonkeyCANTAB.

What constitutes "reward" and "punishment"?

Options for reward and punishment are set in the General Parameters section; visual objects are defined in the Visual Object Library.

More on displaying objects

The task works with an internal scaling system based on a display that is 1000 units (pixels) wide and 750 high. If you specify the stimulus, it should fit into a rectangle that is 267 (w) x 200 (h). Imagine that the screen is divided into nine. (See this diagram ("Size and coordinates").) The possible locations are these nine boxes.

Screenshot from the task

Touch, my monkey. TOUCH IT!

1.8.3 Visual Discriminations and Set-Shifting

This task allows you to give visual discrimination tasks of the following kinds:

- simple discrimination (and reversal)
- compound discrimination (and reversal)
- intradimensional set shift (and reversal)
- extradimensional set shift (and reversal)

To learn what these mean, see About set-shifting.

Because there are at least two ways of manufacturing compound stimuli for use in this task, there are two "styles" in which you can run this task. They are:

- Predefined stimuli style - in which you, the experimenter, create the compound stimuli yourself;
- Superimposed stimuli style - in which you specify simple stimuli and the program superimposes them to create compound stimuli.

Choose one of these, and configure its parameters.

Tip: The "standard" shapes/lines discrimination typically used in monkey testing uses the Superimposed stimuli style.

1.8.3.1 About set-shifting

About the task

Presents two objects. One is correct and one is incorrect. If the subject touches the correct one, it is rewarded; if it touches the wrong one, it is punished. The discrimination varies in difficulty. The subject learns over several trials which stimuli are correct and which are incorrect; then we confuse it by changing the rules.

Stimuli can be *simple* (when they vary along only one dimension, such as shape, colour, intensity, or location), or *compound* (when they vary along more than one dimension).

Dimensional shifting in humans

In human testing, typical dimensions might be colour, shape, and number. A famous example of this kind of test is the Wisconsin Card Sorting Test (Grant & Berg, 1948, J Exp Psychol 38: 404), in which subjects must sort cards with a variable number of coloured shapes. They must discover the sorting rule solely by reinforcement from the experimenter. The rule is then changed: for example, they may be required to switch from "red correct, blue wrong" to "blue correct, red wrong" - a reversal. Alternatively, they may be given a new set of colours to learn - an intradimensional shift. Again, they may also be required to switch from "blue correct, red wrong" to "circles correct, squares wrong" - an extradimensional shift.

Dimensional shifting in non-human primates

In primate testing, two typical dimensions are "blue shapes" and "white squiggly lines". [See Dias R, Robbins TW, Roberts AC (1997). Dissociable forms of inhibitory control within prefrontal cortex with an analog of the Wisconsin Card Sort Test: restriction to novel situations and independence from "on-line" processing. *Journal of Neuroscience* 17: 9285-9297.] They use a large number of exemplars in each dimension (shapes and lines).

Basic test sequence in the general case

Assume there are two dimensions, A and B. There is a large catalogue of objects. Each object is assigned a value for each dimension (or is stated not to possess that dimension). For example, if the dimensions are colour and shape, then each object would be assigned a value for colour (e.g. "red") and shape (e.g. "triangle"). If there were a third dimension, of "superimposed wiggly line", a red triangle stimulus might have a value of "nonexistent" for this third dimension.

OK. For the sake of the following general prototype, keep in mind the primate tests in which the dimensions are typically "background shape" (values e.g. square, circle, triangle...) and "superimposed wiggly line" (similarly, with a variety of values).

- **1. Simple discrimination.** Displayed objects possess dimension A but not dimension B. Within dimension A, value 1 is designated correct and value 2 is designated incorrect. In other words, we reward subjects if A()=1 and punish them if A()=2. (We can run this discrimination for as long as we can find target objects such that A(object)=1 and distractor objects such that A (object)=2, as long as in both cases B(object)=undefined.)
- **2. Simple reversal.** We carry on except that we now designate A()=1 as incorrect and A()=2 as correct. The subject has to reverse its responses to a previously-learned discrimination.
- **3. Compound discrimination.** We now introduce a second dimension, B, but ignore its value. Now objects are selected for presentation on the basis that ((A(object)=1 or A(object)=2) and B (object) != undefined). The symbol "!=" means "does not equal". The subject is rewarded if it selects an object such that A(object)=2 and are punished if they select an object such that A (object)=1. This continues the "A" discrimination from the previous, reversal stage.
- **4. Reversal of compound discrimination.** We reverse the discrimination along dimension A and continue to ignore B. The trials are just the same as the previous stage, but A()=1 is correct and A()=2 is incorrect once more.
- **5. Intradimensional shift.** We introduce new values of dimension A; dimension B is still ignored. Objects are selected such that ((A(object)=3 or A(object)=4) and B(object) != undefined). Subjects are rewarded if A(object)=3 and punished if A(object)=4.
- **6. Extradimensional shift.** We switch our attention to dimension B and ignore A. Objects are selected such that ((B(object)=1 or B(object)=2) and A(object) != undefined). Subjects are rewarded if B(response)=1 and punished if B(response)=2.

What's ignored? By whom? A caveat. When a dimension B (or anything else) is present and being ignored, we would be wise to ensure that p(response is correct) is not dependent on the value of this dimension. Otherwise, our subject may well learn to discriminate on the dimension we (and it) are supposed to be ignoring. (Other dimensions like location or presentation order are important to

consider here.) A dimension can be made irrelevant by holding the value of the dimension constant (e.g. always presenting stimuli in a single location, always making the background shape purple, etc.), or by randomizing it (e.g. always randomizing the locations of pairs of objects presented). If a dimension is randomized rather than held constant, the subject may *attempt* to learn the discrimination based on this dimension but cannot succeed.

Therefore, we need a library of objects with at least one object in each of the following categories:

A=1; B=undefined
A=2; B=undefined
A=1; B=anything except "undefined", constant or randomized
A=2; B=anything except "undefined", constant or randomized
A=3; B=anything except "undefined", constant or randomized
A=4; B=anything except "undefined", constant or randomized
A=anything except "undefined", constant or randomized; B=1
A=anything except "undefined", constant or randomized; B=2

Having lots of objects in each of these categories may be a good thing behaviourally, as it may lead the subject to extract the general features of that dimension/value (for example, if A(1) is equivalent to colour(red), then having lots of red objects may lead the subject to learn that "red" is correct, and not "red triangle"). This is the approach taken by the primate tasks, but not by tasks used for rats (or, now, pigs).

Dimensional shifting in rats

Birrell & Brown (2000) implemented a set-shifting task in rats (Birrell JM & Brown VJ, 2000. Medial frontal cortex mediates perceptual attentional set shifting in the rat. *Journal of Neuroscience* 20: 4320-4). Rats dug in two bowls for food. The bowls has dimensions of (A) odour; (B) filling medium; (C) surface texture. They adopted a policy of changing all stimuli at times of ID or ED shifting (a "total change design", p4321, which is required for accurate interpretation of the difference between reversal learning and ED shifts; see p4323). Their test sequence was as follows (+ indicates correct stimuli, - incorrect, bold indicates the correct part of the stimulus):

Simple discrimination	**A1**(+), A2(-)
Compound discrimination	**A1**/B1(+), A2/B2(-)
	A1/B2(+), A2/B1(-)
Reversal	**A2**/B1(+), A1/B2(-)
	A2/B2(+), A1/B1(-)
ID shift	**A3**/B3(+), A4/B4(-)
	A3/B4(+), A4/B3(-)
Reversal	**A4**/B3(+), A3/B4(-)
	A4/B4(+), A3/B3(-)
ED shift	**B5**/A5(+), B6/A6(-)
	B5/A6(+), B6/A5(-)
Reversal	**B6**/A5(+), B5/A6(-)
	B6/A6(+), B5/A5(-)

This illustrates the general test sequence nicely.

The sequence used by MonkeyCantab

Much the same:

Simple discrimination	**A1**(+), A2(-)
Reversal	**A2**(+), A1(-)

Compound discrimination	**A1**/B1(+), A2/B2(-)
	A1/B2(+), A2/B1(-)
Reversal	**A2**/B1(+), A1/B2(-)
	A2/B2(+), A1/B1(-)
ID shift	**A3**/B3(+), A4/B4(-)
	A3/B4(+), A4/B3(-)
Reversal (optional)	**A4**/B3(+), A3/B4(-)
	A4/B4(+), A3/B3(-)
ED shift	**B5**/A5(+), B6/A6(-)
	B5/A6(+), B6/A5(-)
Reversal (optional)	**B6**/A5(+), B5/A6(-)
	B6/A6(+), B5/A5(-)

How long shall we test for? The usual measure on this task is trials (or errors) to criterion. Birrell & Bowman (2000) used a criterion of 6 consecutive correct correct responses. That seems reasonable (though make the number configurable).

Left/right position should be chosen randomly for each trial. Order of presentation of the two alternative pairs (e.g. which to present of A1/B1-A2/B2 or A1/B2-A2/B1) should be randomized in pairs of trials.

1.8.3.2 Predefined Stimuli Style

In this version of the task, it is your job as the experimenter to create appropriate compound stimuli, using combinations of dimensions that you are interested in.

Why use this version of the task?

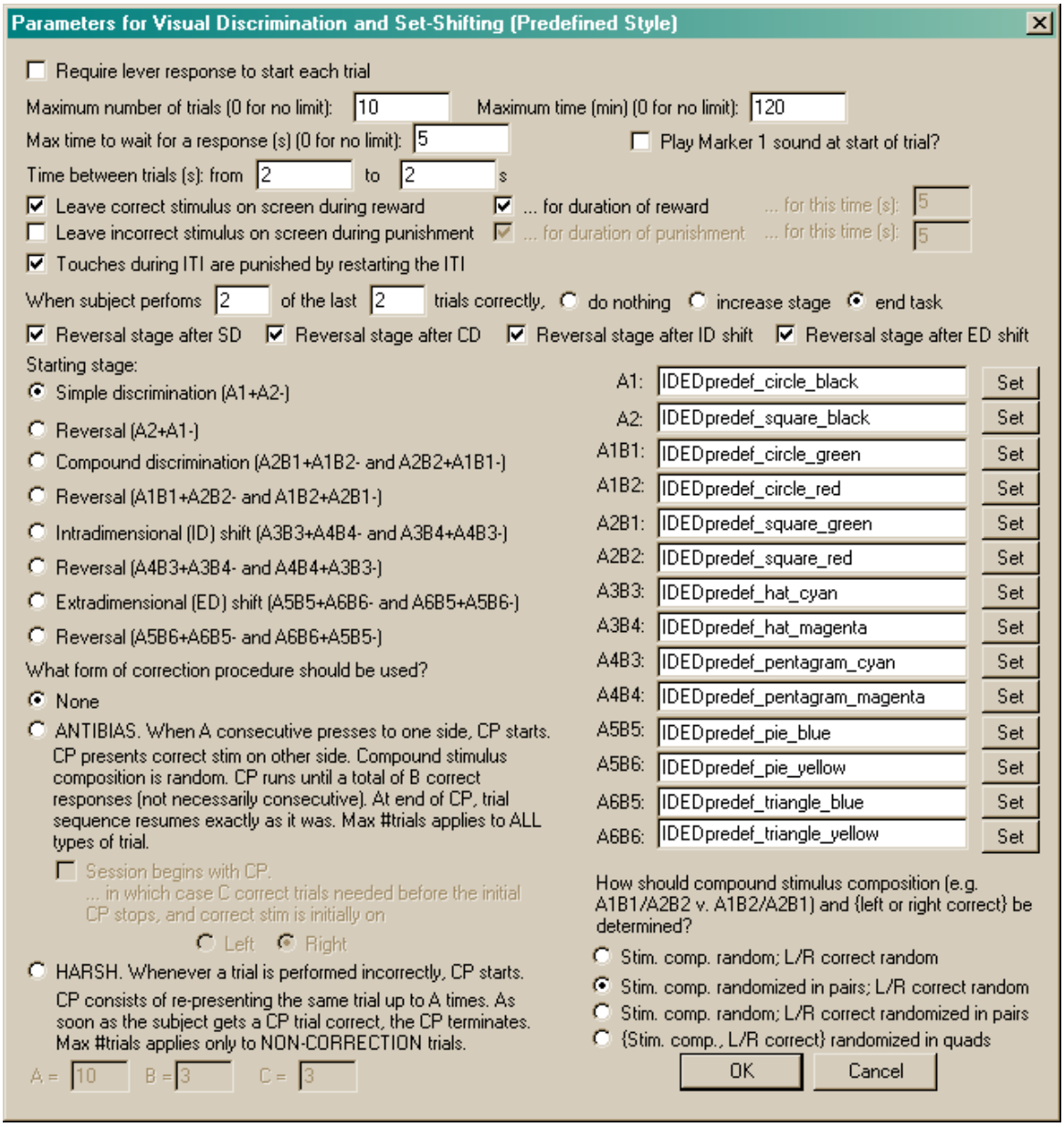
You can use this version to present compound stimuli containing *arbitrary* dimensions - colour/number, colour/shape, shape/number, shape/line, vehicle/person... For example, you could use bitmaps taken from films to run a {this photograph contains a tank or a car}/{this photograph contains Harrison Ford or David Niven} vehicle/person compound discrimination! The dimensions can be arbitrary because you create compound stimuli yourself.

Why not use this version of the task?

If you want to be able to generate a large number of potential compound stimuli from a library of simple stimuli, you could use this version... but it would be a lot of work to create all the compound stimuli. If the compound stimuli can be generated by superimposing the simple stimuli, you can save yourself some work by using the Superimposed Stimuli Style.

Configuring the task

Given that the user generates the stimuli, configuring the task is just a matter of plugging in the stimuli to the pattern shown above. That is, we need one object to correspond to each of **A1, A2, A1/B1, A1/B2, A2/B1, A2/B2, A3/B3, A3/B4, A4/B3, A4/B4, A5/B5, A5/B6, A6/B5, A6/B6.** Here's the configuration dialogue box, with the example of a **colour/shape** paradigm:

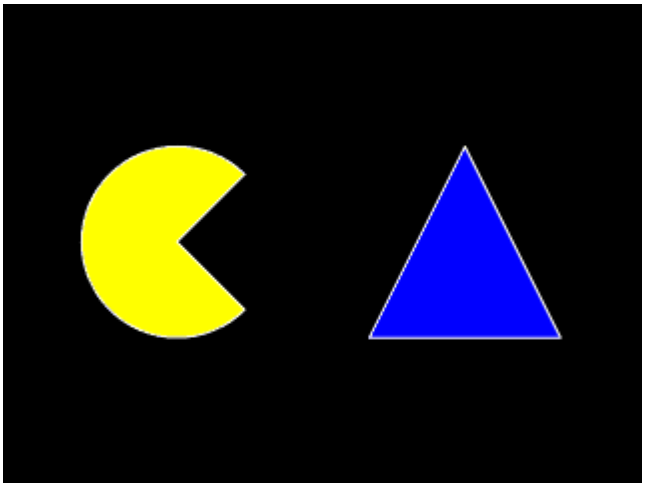


- **Require lever response to start each trial.** Requires that the subject make a lever response to initiate each trial. See Use with Dogs.
- **Maximum number of trials.** When the subject has performed this number of trials, the task ends. (You may specify 0 for no limit, though you must specify a limit on the number of trials, the time, or both.)
- **Maximum time.** When this time elapses, the task is terminated as soon as the current trial has finished. (You may specify 0 for no limit, though you must specify a limit on the number of trials, the time, or both.)
- **Maximum time to wait for a response.** If the subject fails to make a response within this time, the subject fails the trial.
- **Play Marker 1 sound at start of trial?** Fairly obvious.
- **Time between trials.** Specify a minimum and a maximum intertrial time (they may be the same). The actual time is chosen with a rectangular probability distribution within these values. The time between trials starts **after** the reward or punishment from the previous trial has finished.
- **Leave correct stimulus on during reward? (etc.)** When the subject responds, the correct stimulus can be left on the screen during reward, and/or the incorrect stimulus can be left on

during punishment. These stimuli can either be left on for the duration of the reward/punishment (as specified in the General Parameters), or you can specify how long to leave them on the screen for.

- **When subject performs X of the last Y trials correctly... do nothing/increase stage/end task.** Fairly obvious, I hope. Set the value of X and Y in the boxes.
- **Give a reversal stage after the simple discrimination.** Enables/disables the "SD reversal" phase.
- **Give a reversal stage after the compound discrimination.** Enables/disables the "CD reversal" phase.
- **Give a reversal stage after the ID shift.** Enables/disables the "ID reversal" phase.
- **Give a reversal stage after the ED shift.** Enables/disables the "ED reversal" phase.
- **Starting stage.** Choose the stage to start at for this session.
- **Correction procedure.** Choose the type of correction procedure you wish to use. The meaning of the types of correction procedure is explained carefully in the dialogue box. Note that if your subject shifts up a stage, any ongoing correction procedure is cancelled, and all correction procedure counts are reset.
- **Stimuli.** Choose the stimuli required by the task (A1, A2, A1B1, etc.). An example is shown above, using shape and colour as dimensions A and B respectively.
- **Randomization.** Choose the randomization technique used for the task. For compound stimuli, in any given trial you can present either A1B1 / A2B2 compounds, or A1B2 / A2B1 compounds. Furthermore, the correct stimulus can be on the right or on the left. The stimulus composition and the "left/right correct" assignment can be fully random (with a very small chance that you get ten on the left consecutively correct, because that's what random means). Or, the "left/right correct" assignment can be randomized in pairs (meaning that in pair of trials you get one "left correct" trial and one "right correct" trial, e.g. L,R - R,L - R,L - L,R - R,L - R,L - R,L - ...), with the stimulus composition totally random. Or the stimulus composition can be randomized in pairs (e.g. A1B1[/A2B2], A1B2[/A2B1] - A1B2[/A2B1], A1B1[/A2B2] - A1B2[/A2B1], A1B1[/A2B2] - ...) with the L/R assignment fully random. Or the four possible combinations (L versus R correct and A1B1/A2B2 versus A1B2/A2B1) can be randomized in groups of four trials (quads).

Screenshots of the task



1.8.3.3 Superimposed Stimuli Style

In this version of the task, you supply *simple* stimuli to the program, and it superimposes two of them to create appropriate *compound* stimuli.

Why use this version of the task?

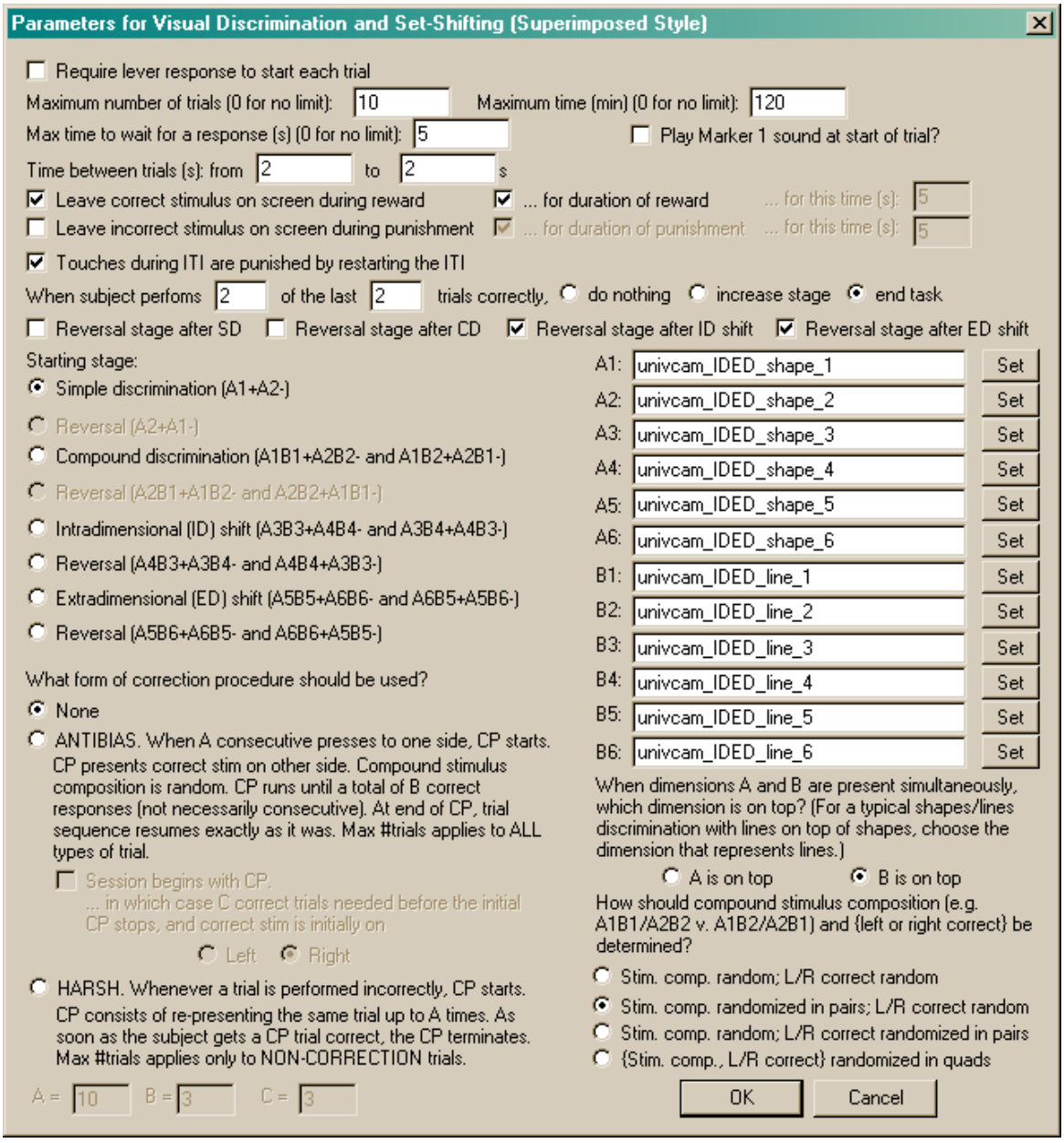
It means you can use a large library of simple shapes, and the program can put them together to

create new compound stimuli without any extra effort on your part. For example, if you have a library of blue shapes and a library of white lines, you can have the program superimpose the lines on the shapes to create shape/line compound stimuli.

Why not use this version of the task?

The dimensions must be superimposable. You couldn't easily use if for a colour/number compound discrimination. Nor could you use if for a {this photograph contains a tank or a car}/{this photograph contains Harrison Ford or David Niven} vehicle/person compound discrimination! If you want to run this kind of test, you need to use the Predefined Stimuli Style.

Configuring the task

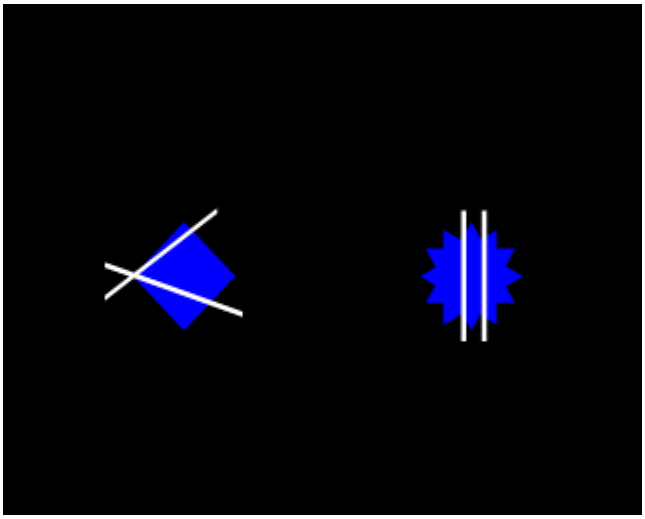


- **Require lever response to start each trial.** Requires that the subject make a lever response to initiate each trial. See Use with Dogs.
- **Maximum number of trials.** When the subject has performed this number of trials, the task

ends. (You may specify 0 for no limit, though you must specify a limit on the number of trials, the time, or both.)

- **Maximum time.** When this time elapses, the task is terminated as soon as the current trial has finished. (You may specify 0 for no limit, though you must specify a limit on the number of trials, the time, or both.)
- **Maximum time to wait for a response.** If the subject fails to make a response within this time, the subject fails the trial.
- **Play Marker 1 sound at start of trial?** Fairly obvious.
- **Time between trials.** Specify a minimum and a maximum intertrial time (they may be the same). The actual time is chosen with a rectangular probability distribution within these values. The time between trials starts **after** the reward or punishment from the previous trial has finished.
- **Leave correct stimulus on during reward? (etc.)** When the subject responds, the correct stimulus can be left on the screen during reward, and/or the incorrect stimulus can be left on during punishment. These stimuli can either be left on for the duration of the reward/punishment (as specified in the General Parameters), or you can specify how long to leave them on the screen for.
- **When subject performs X of the last Y trials correctly... do nothing/increase stage/end task.** Fairly obvious, I hope. Set the value of X and Y in the boxes.
- **Give a reversal stage after the simple discrimination.** Enables/disables the "SD reversal" phase.
- **Give a reversal stage after the compound discrimination.** Enables/disables the "CD reversal" phase.
- **Give a reversal stage after the ID shift.** Enables/disables the "ID reversal" phase.
- **Give a reversal stage after the ED shift.** Enables/disables the "ED reversal" phase.
- **Starting stage.** Choose the stage to start at for this session.
- **Correction procedure.** Choose the type of correction procedure you wish to use. The meaning of the types of correction procedure is explained carefully in the dialogue box. Note that if your subject shifts up a stage, any ongoing correction procedure is cancelled, and all correction procedure counts are reset.
- **Stimuli.** Choose the stimuli required by the task (A1, A2, A1B1, etc.). An example is shown above, using shape and colour as dimensions A and B respectively.
- **Dimension on top.** Your compound stimuli are made up of two dimensions, A and B. Simple discriminations are performed with dimension A alone. When you create compound stimuli by superimposing simple stimuli, would you like A or B to be on top? If you are using a typical shapes/lines discrimination, for example, you probably want lines on top.
- **Randomization.** Choose the randomization technique used for the task. For compound stimuli, in any given trial you can present either A1B1 / A2B2 compounds, or A1B2 / A2B1 compounds. Furthermore, the correct stimulus can be on the right or on the left. The stimulus composition and the "left/right correct" assignment can be fully random (with a very small chance that you get ten on the left consecutively correct, because that's what random means). Or, the "left/right correct" assignment can be randomized in pairs (meaning that in pair of trials you get one "left correct" trial and one "right correct" trial, e.g. L,R - R,L - R,L - L,R - R,L - R,L - R,L - ...), with the stimulus composition totally random. Or the stimulus composition can be randomized in pairs (e.g. A1B1[/A2B2], A1B2[/A2B1] - A1B2[/A2B1], A1B1[/A2B2] - A1B2[/A2B1], A1B1[/A2B2] - ...) with the L/R assignment fully random. Or the four possible combinations (L versus R correct and A1B1/A2B2 versus A1B2/A2B1) can be randomized in groups of four trials (quads).

Screenshots of the task



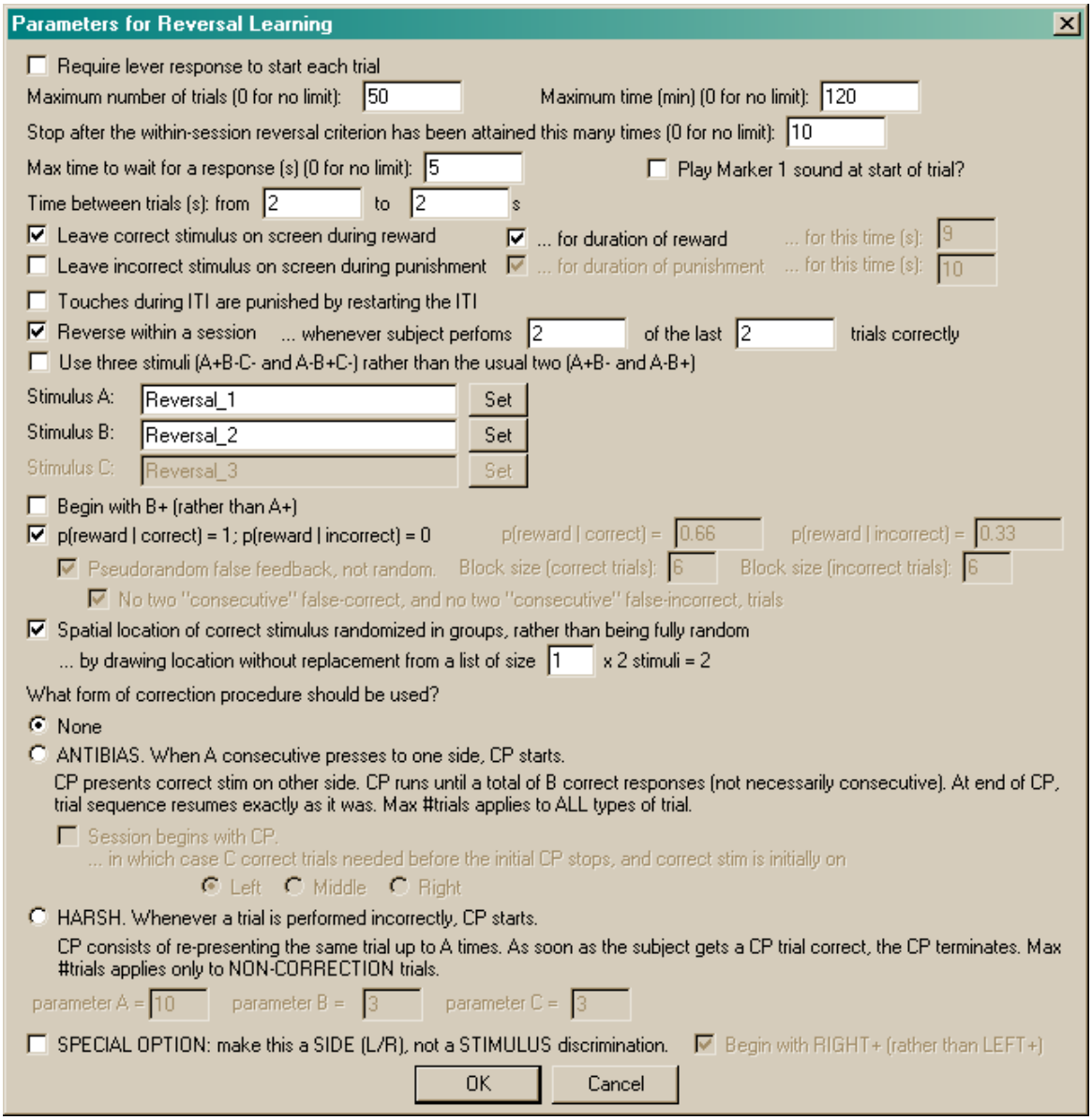
1.8.4 Reversal Learning

About the task

Provides facilities for simple or serial reversal learning, with either two stimuli (A+B- → A-B+) or three (A+B-C- → A-B+C-). You could use one of the Visual Discrimination tasks to accomplish basic between-session reversal learning; this task provides more sophisticated options.

There is an option to use three objects. In this task, a subject is trained with A+B-C- and then reversed to A-B+C-; perseveration can then be measured directly as the degree to which subjects respond to A more than to C. For examples of this task in the recent neurobiological literature, see Arnsten et al. (1997; Neurobiology of Aging 18: 21-28) or Jentsch et al. (2002; Neuropsychopharmacology 26: 183-190). I'm sure this form of the task has a much longer history, but I don't have my copy of Mackintosh (1974; "The Psychology of Animal Learning") to hand!

Configuring the task



- **Require lever response to start each trial.** Requires that the subject make a lever response to initiate each trial. See Use with Dogs.
- **Maximum number of trials.** When the subject has performed this number of trials, the task ends. (You may specify 0 for no limit, though you must specify a limit on the number of trials, the time, or both.)
- **Maximum time.** When this time elapses, the task is terminated as soon as the current trial has finished. (You may specify 0 for no limit, though you must specify a limit on the number of trials, the time, or both.)
- **Stop after the within-session reversal criterion has been attained this many times.** It's easiest to give an example. If you enable within-session reversals (see below) and enter "2" here, then the subject will be started on one configuration (e.g. A+B-), allowed to proceed until it has passed its first reversal criterion, tested on the new task (A-B+), allowed to proceed until it has passed the second reversal criterion, and then stopped. (Within-session reversal criteria are explained below. You may specify 0 for no such limit. If you do not enable within-session reversals, this option will have no effect.)
- **Maximum time to wait for a response.** If the subject fails to make a response within this time, the subject fails the trial. (You may specify 0 for no limit.)
- **Play Marker 1 sound at start of trial?** Fairly obvious.

- **Time between trials.** Specify a minimum and a maximum intertrial time (they may be the same). The actual time is chosen with a rectangular probability distribution within these values. The time between trials starts **after** any reward or punishment from the previous trial has finished.
- **Leave correct stimulus on during reward? (etc.)** When the subject responds, the correct stimulus can be left on the screen during reward, and/or the incorrect stimulus can be left on during punishment. These stimuli can either be left on for the duration of the reward/punishment (as specified in the General Parameters), or you can specify how long to leave them on the screen for.
- **Touches during ITI are punished by restarting the ITI.** Fairly obvious.
- **Reverse within a session... when subject performs X of the last Y trials correctly.** Fairly obvious, I hope. Set the value of X and Y in the boxes.
- **Use three stimuli rather than two?** Choose whether you want a two-stimulus task or a three-stimulus task.
- **Stimuli.** Choose the stimuli required by the task (A, B, +/- C). In a three-stimulus task, stimulus C is never correct.
- **Begin with B+ (rather than A+)?** If you want stimulus B to be correct initially, tick this box. Otherwise, A will be correct.
- **Probability of reward given a correct/incorrect response.** A conventional reversal procedure has p(reward | correct) = 1 and p(reward | incorrect) = 0. However, if you would like a fully probabilistic reversal task, untick this box. You may then specify p(reward | correct) and p (reward | incorrect) directly. For example, if you specify 0.8 and 0.2, then correct responses would be rewarded 80% of the time, while incorrect responses would be rewarded 20% of the time.
  - If you reward some "incorrect" responses, and you have chosen the option "Leave correct stimulus on during reward", the program will leave the **chosen** stimulus on (i.e. one that is notionally "incorrect", but is being rewarded on this trial). This seems the only consistent thing to do. Essentially, a probabilistic task blurs the definition of "correct" and "incorrect", so the option is best described as "Leave chosen stimulus on if it's rewarded"!
  - **Pseudorandom false feedback, not random.** By default, the false feedback system gives feedback on a random basis (i.e. it flips a biased coin with the bias you specifed above on each trial). You may want false feedback to be *pseudorandom* instead, e.g. to guarantee that if you specify 20% false feedback, two out of every 10 trials have false feedback (whereas in a random system with p = 0.2 for false feedback, you are not guaranteed two out of every 10). This pseudorandom option allows you to specify blocks of X correct trials and Y incorrect trials, such that in every consecutive X correct trials and Y incorrect trials, a certain proportion (specified above) give true or false feedback. As always, for more explanation of this topic, see randomness, pseudorandomness, and drawing without replacement.
    - If you use the pseudorandom option, specify the **block size** for correct and for incorrect trials. This is the number of consecutive trials (correct or incorrect - they are calculated separately) over which the system calculates. **Be careful how you specify this; errors are possible.** If you specify p(reward | correct) = 0.8, then you're saying that you want 80% of your correct trials to be rewarded - so you might do well to pick a number that is a multiple of 5 here (e.g. 10 for 8/10 correct trials to be rewarded, or 20 for 16/20 correct trials to be rewarded). If you pick 0.8 and then specify a sequence of 9 trials, the program will not behave as you want (it'll calculate 0.8 * 9 = 7.2, then add 0.5 to implement correct rounding, giving 7.7, and truncate this to 7 for the number of "truthful" trials, then take 9 - 7 = 2 for the "untruthful" trials, so the probability of truth will not be exactly what you requested).
    - You can also enforce that **no two consecutive correct or incorrect trials (taken separately) give false feedback.** For example, if your subject responds correctly but receives false feedback (punishment), then responds incorrectly on the next two trials, and then responds correctly again, this option would ensure that this last correct trial is not punished (because it's the second of two "consecutive" correct trials). **Note that this option, with small block sizes, can lead to predictable trial sequences**

(because the constrains leave the program little or no choice).

- **Spatial location of correct stimulus randomized in pairs...**
  - If this box is not ticked, the location of the correct stimulus is chosen at random for each trial (and, in the three-stimulus task, the location of the "incorrect but correct in the past" and the "never correct" stimuli are similarly chosen at random).
  - If this box is ticked, then the locations are randomized in groups where the group size is ***n*** times the number of stimuli. You specify this value of ***n*** in the box labelled "**... by drawing without replacement from a list of size *n*** x the number of stimuli...".
  - Suppose that you specify **n = 1;** then the locations will be randomized in pairs (for the two-stimulus task), meaning that in every pair of trials, the correct stimulus is on the left on one trial and on the right in the other, but the order of those two trials within the pair is random. For the three-stimulus task, there are six possible spatial combinations (ABC, ACB, BAC, BCA, CAB, CBA) and in every six trials one of these combinations will be used, with the order within the group of six being random.
  - Put another way, then if *n* = 1, for the two-stimulus task, each block of two trials will contain one "left correct" (L) trial and one "right correct" (R) trial. (Therefore, in this case, it's impossible to get more than two trials in a row with the same side correct.)
  - If you specify *n* = 2, then for the two-stimulus task, each block of four trials will contain two L trials and two R trials. (In this example, it's impossible to see more than four trials in a row with the same side correct.)
  - For more explanation of this topic, see randomness, pseudorandomness, and drawing without replacement.

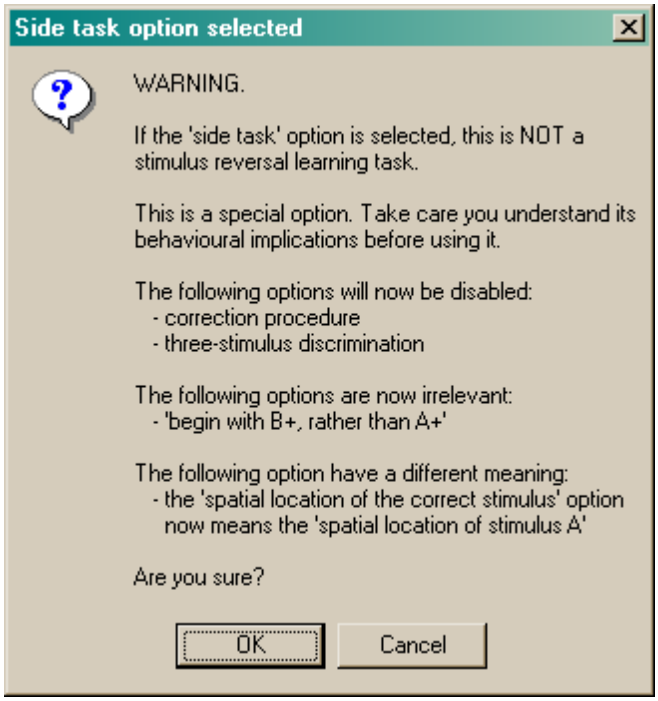
- **Correction procedure.** Choose the type of correction procedure (CP) you wish to use. Correction procedures are used to try to prevent the subject responding to one side (spatial location), rather than one stimulus. For example, if two stimuli are presented and one is correct and the other is wrong, but the correct stimulus is randomly presented on the left or right, the subject could win on 50% of trials simply by responding to the left. Now, a good analysis of the data (the best being an analysis based on the principles of signal detection theory) will immediately show that the subject is not discriminating the two stimuli. However, some experimenters wish to discourage the use of a spatial strategy further. A common way of doing so is a correction procedure. For example, if the animal keeps responding to one side, the correction procedure could present the correct stimulus on the other side until the subject breaks its positional habit and responds to the other side. The meaning of the types of correction procedure available in this task is explained in the dialogue box, and as follows.
  - **None.** There is no correction procedure. Trials are simply presented according to the usual options we have discussed above.
  - **Antibias.** When a certain number of consecutive responses have been made to one side (left or right), the correction procedure begins. The number of trials this takes is known as **parameter A**. Having begun, the correction procedure presents the correct stimulus on the other side, overriding the usual mechanism (discussed above) for deciding which side the correct stimulus is shown on. The correction procedure continues until the subject has made a certain number of correct responses - this number is called **parameter B**. The correct responses do not have to be in sequence (so if B is 5, then the subject might get a series of correction trials *correct - wrong - wrong - correct - correct - wrong - correct - wrong - wrong - correct*, and then the correction procedure would stop). Once this target number of successful "correction" trials has been achieved, the correction procedure stops, and the usual sequence of trials resumes afterwards. (If a maximum number of trials has been set for the session, both "standard" and "correction" trials count towards this limit.)
    - When using the Antibias correction system, it is also possible to **begin the session with the correction procedure.** This allows the experimenter to "force" the correct stimulus to one side until the subject gets enough correct trials to terminate the correction procedure. You might use it if, for example, your subject began a correction procedure at the end of its previous session, and then ran out of time/trials, so you would like it to resume. Suppose you like your correction procedures to run until the subject has got 10 trials right (B = 10), but in the last session your subject got to 4

and then ran out of time. You'd like to finish off the correction procedure from last time (requiring the subject to get 6 more "correction" trials right at the start of the session before normal business resumes) but then have your usual B = 10 if the subject requires another correction procedure. No problem: specify **parameter C** = 6 (and B = 10 as usual) to achieve this.

- In a two-stimulus task, stimuli can either be on the left or the right, and our correction procedure is simple. What happens if we use a three-stimulus task, in which the stimuli can be on the left, on the right, or in the middle? Well, if the "antibias" correction procedure is employed with a *three-stimulus* task and the subject perseverates in the middle, then the correct stimulus is randomly assigned to the left or the right location for the correction procedure. If it perseverates on the left, then the correct stimulus is assigned to the right-hand side; if it perseverates on the right, the correct stimulus is assigned to the left-hand side. In all cases, once the correction procedure has determined where the correct stimulus is to be, it chooses the location of the "incorrect but once correct" stimulus and the "never correct" stimulus (stimulus C, as in A+B-C-/A-B+C-) at random.

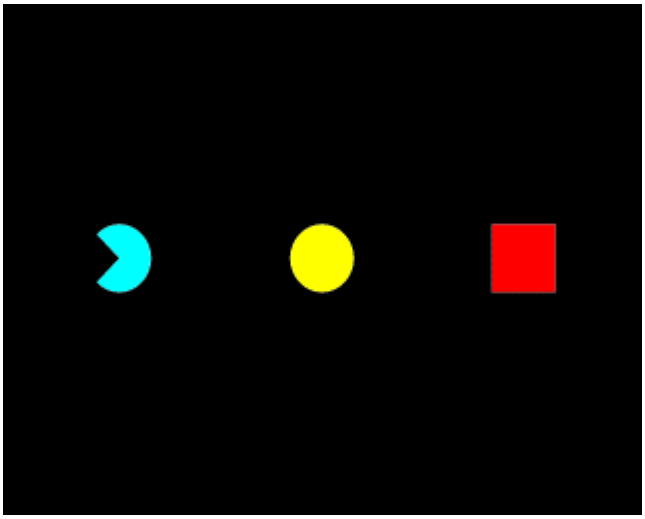
- **Harsh.** In this system, *whenever* the subject gets a trial wrong, the correction procedure starts. This type of correction procedure simply repeats the trial that the subject got wrong, until it gets it right, or until a limiting number of trials (**parameter A**) is reached. When either of these conditions is met, the correction procedure stops. (If a maximum number of trials has been set for the session, only "standard" [non-correction] trials count towards this limit when using the Harsh system.)
- Whatever the type of correction procedure, note that if you allow within-session reversals to happen and your subject achieves the criteria for reversing, the reversal takes priority over the correction procedure: any ongoing correction procedure is cancelled, and all correction procedure counts are reset (i.e. the whole correction system starts again from scratch).

- **SPECIAL OPTION: make this a side (LEFT/RIGHT) rather than a stimulus discrimination.** This is a special option that stops the program running a stimulus discrimination/reversal task, and makes it a SIDE or LOCATION discrimination. That is, either "left" or "right" is correct (and the program can, if you wish, reverse between these), but the A and B stimuli are displayed at random, so their visual appearance is irrelevant. For example, for one trial the A stimulus might be on the left and the B stimulus on the right (with the left-hand stimulus being correct) and for the next trial B might be on the left and A on the right (with the left-hand stimulus again being correct).
  - Begin with RIGHT+ (rather than L+)? If you tick this box, you will begin with LEFT-RIGHT+; if you leave the box blank, you will get LEFT+RIGHT-.
  - If the special option is ticked, a few other options become irrelevant, and the meaning of the "spatial randomization" option changes (rather than "how should the L/R location of the correct stimulus be picked?", it now means "how should the A/B stimulus to be displayed at the correct location be picked?") and a warning message to this effect pops up:



Screenshots of the task

The moment of choice, with three stimuli.



In this case, the subject responded correctly and correct stimuli are being left up during reward:

1.8.5 Delayed Matching/Non-matching to Sample

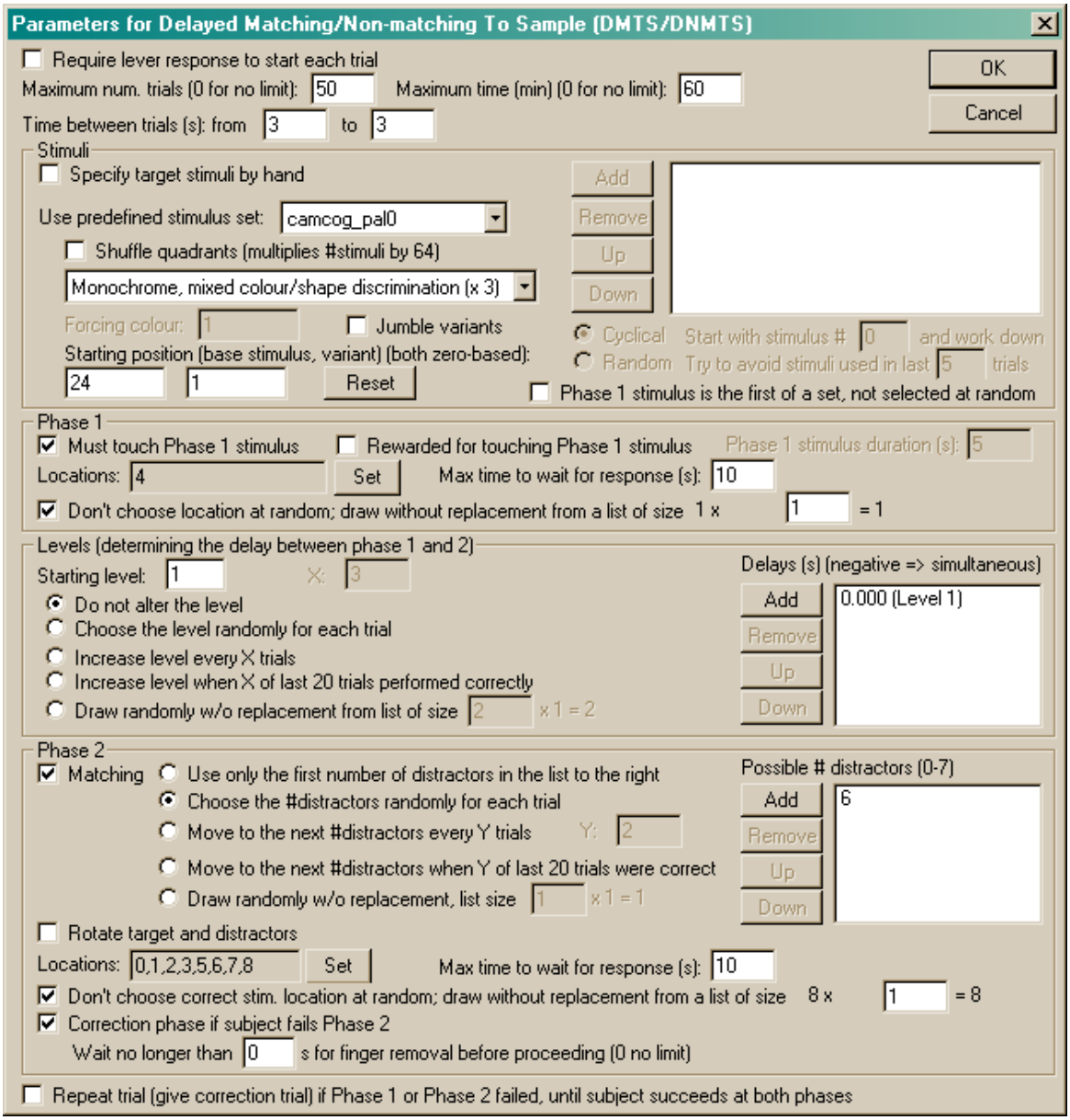
About the task

The Marker 1 sound is played to signal the start of a trial. An object is shown in the centre of the screen (Phase 1). (Optionally, the subject has to touch this object; optionally, it can be rewarded for doing so.) The object vanishes, and a delay ensues. After this delay, the object is re-presented together with one or more other objects (Phase 2), heralded by the Marker 2 sound.

In **delayed matching to sample (DMTS),** the subject must touch the object that was shown first. In **delayed non-matching to sample (DNMTS),** it must touch the new object. (Both test the subject's ability to remember information about the first object during the delay; the matching/ nonmatching option is typically used to account for or overcome a subject's species-specific natural tendency to select either familiar or novel stimuli.)

Correct responses are rewarded; incorrect responses are punished. Optional correction procedure: if correction is switched on, failed trials are repeated once, or until the subject gets it right (see below).

Configuring the task



- **Require lever response to start each trial.** Requires that the subject make a lever response to initiate each trial. See Use with Dogs.
- **Maximum number of trials.** When the subject has performed this number of trials, the task ends. (You may specify 0 for no limit, though you must specify a limit on the number of trials, the time, or both.)
- **Maximum time.** When this time elapses, the task is terminated as soon as the current trial has finished. (You may specify 0 for no limit, though you must specify a limit on the number of trials, the time, or both.)
- **Time between trials.** Specify a minimum and a maximum intertrial time (they may be the same). The actual time is chosen with a rectangular probability distribution within these values. The time between trials starts **after** any reward or punishment from the previous trial has finished.

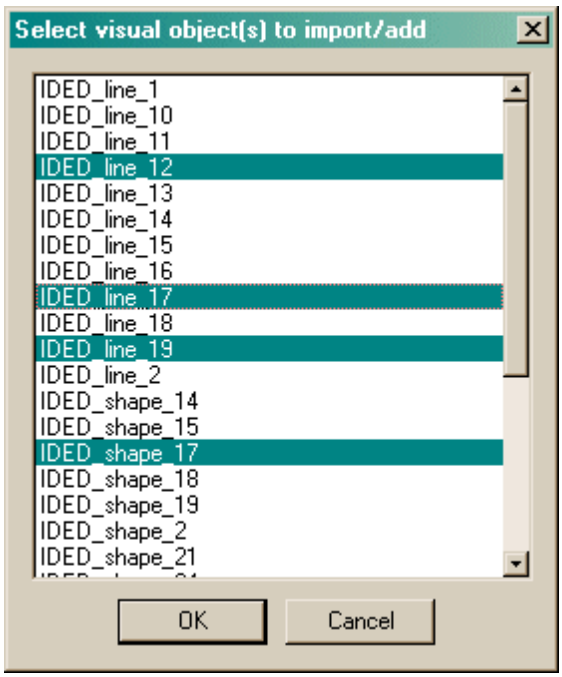
Stimuli

- **Specify stimuli by hand.** If you choose to specify stimuli by hand, the list shows the available stimuli. You cannot put a stimulus into the list more than once. Click **Add** and **Remove** to add/remove stimuli. You can also choose one of two methods for choosing the stimuli for each trial:

- **Cyclical.** The program begins with a specified **stimulus number** and selects stimuli for each trial by working down the list, resuming at the start of the list if/when it runs out of stimuli to use at the bottom of the list.
- **Random.** The program picks a set of stimuli to use at random on each trial. You will need to fill in **"Try to avoid stimuli used in last X trials";** the program will try not to choose any stimuli that have appeared (or were scheduled to appear, in the case of failure at phase 1) in the last X trials.

It's your responsibility to ensure that enough stimuli are in the list! The program will complain if you try to start it and there aren't enough stimuli to set up a trial. It won't complain if it needs to re-use stimuli from trial to trial.

Incidentally, when you **Add** stimuli, you can choose several at once by holding down the Shift key as you click on stimuli:



- **Predefined stimuli.** You may also use one of the predefined stimulus sets. If you choose this, you have several further options:

  - You can **shuffle the quadrants** of the stimuli, which multiplies the number of available stimuli by 64.
  - You can **vary the colours** of the predefined stimuli in one of several ways:

    Colours unmodified
    Vary colours (multiplies #stim x 2401)
    Monochrome, shape-only discrimination (x 7)
    Monochrome, colour-only discrimination (x 1)
    Monochrome, mixed colour/shape discrimination (x 3)
    Seven colours per trial (x 840)
    Monochrome, shape-only discrimination, fixed colour (x 1, although you can run this task with up to 13 different colours)

    For full explanations of these options, see "Technical details of the stimulus-generating techniques used".

  - Optionally, you can **jumble the variants,** which is a deterministic process (i.e. it's not

random), but it does tend to make adjacent trials much less similar, and is therefore to be recommended.
When you use a predefined stimulus set, you specify the **starting position** - the starting stimulus and colour variant. Both are zero-based (i.e. "0" means the first stimulus or the first variant, and "1" the second...) The program will store the final position at the end of the task, so that next time you can reload the subject's configuration file and just carry on. The position is stored within the configuration file (with the intention that you have one configuration file per subject). The program cycles through all the stimuli with the first colour variant, then starts again at the start of the set and uses all stimuli with the next colour variant, and so on, until all colour variants are exhausted. More likely, your subjects will be exhausted first, since there can be many millions of possible stimuli! *(As of 2-Apr-2005, the old option to set "stimulus within subset" has been removed, and this parameter always starts at zero, since it confused people.)*

- For the "monochrome, shape-only discrimination, fixed colour" option, you can also specify the **forcing colour** (the colour to force all the stimuli to be). See "Technical details of the stimulus-generating techniques used" for details.

- **Phase 1 stimulus is the first of a set, not selected at random.** Whichever stimulus selection method is used, the program must create a list of stimuli for use on each trial. This list must contain the phase 1 stimulus (call it the S+) and one or more alternative stimuli to present along with the S+ in phase 2. By default, the program picks the S+ at random from this list. If you tick this option, the *first* stimulus in the list is assigned as the S+. The upshot: tick this option if you would like complete predictability of which stimulus in a set will be the S+.

Phase 1

- **Must touch phase 1 stimulus.** If this is selected, then the subject must respond to the Phase 1 stimulus in order to proceed to phase 2. If you choose this option, you may also choose whether or not the subject should be **rewarded for touching the Phase 1 stimulus** (but if you do that, prior to 2/7/9 the memory delay began at the END OF THE REWARD; after 2/7/9 [MonkeyCantab 5.6] it begins at the start of the reward and the memory delays must all exceed the maximum reward time set in the General Parameters). If you do not want your subject to have to touch the stimulus, you must specify the **Phase 1 stimulus duration** instead.
- **Locations.** Here you can specify (by clicking **Set**) the location(s) that your Phase 1 stimulus may be displayed in. The possible locations range from 0 (top left) to 8 (bottom right), as illustrated in the *9-way grid* picture shown in the Size, coordinates, and grids section. Typically, location 4 (the centre of the screen) is used.
- **Maximum time to wait for a response.** If the subject fails to make a response within this time, the subject fails the trial. (This time limit applies to Phase 1, if you require your subjects to touch the Phase 1 stimulus.)
- **Choose location at random, or draw without replacement?** By default, on each trial, the Phase 1 location is chosen at random from the possibilities listed in the Locations box (see above). If you tick "Don't choose at random; draw without replacement", then the program behaves as follows. It makes a list containing N copies each of all the possible locations, where N is the number you can type into the box (labelled "... from a list of size [number of possible locations] x **N** = [total list size]."). Suppose you want locations 3, 4, and 5 as your possible phase 1 locations - then there are three possible locations (call this L=3). If you enter N=2, the program will enter the locations {3,3,4,4,5,5} (N=2 copies of each possible location) into its list, for a total list size of L x N = 6. For each trial, it will then choose a location at random from this list, removing it from the list at the same time ("drawing [out from the list] without replacement"). Suppose the first trial chooses location 4; then the list will be {3,3,4,5,5} for the next trial. When the list is emptied, it is repopulated with L x N entries as before. The idea behind this is to allow you to prevent the program from choosing the location completely at random, instead ensuring an exactly equal distribution of locations across trials, nevertheless with some random element from trial to trial. (Obviously, if you specify N=1, then if you have L possible locations, each possible location will be used once in every L trials.) The larger N is, the closer the system is to

true random sampling (i.e. an infinite N is equivalent to unticking this option). Be aware, however, that fully random choice means that your subject cannot predict anything on the basis of past locations (even if the distribution of locations across trials is not exactly even as a consequence of random sampling), but with a draw-without-replacement system, location *does* become informative (e.g. if you've had locations 4,3,4,5,3 for the first five trials in our example, the subject could in principle be certain that location 5 will be the one to be used next).

- For more explanation of this topic, see randomness, pseudorandomness, and drawing without replacement.

Levels (determining the delay between Phase 1 and Phase 2)

- On the right-hand side of the screen is the list of memory delays (delays between phase 1 and phase 2) that correspond to levels in the task. You may **add, remove, or re-order** the levels with the buttons next to the list. You may then choose the **starting level,** and the **method** by which the task chooses a level for each trial.

  You can have the level fixed, or chosen randomly for each trial - in which case you can't enter a starting level - or you can increase the level by one every X trials, or you can increase the level by one when X of the last 20 trials have been performed correctly. Set your chosen value of X in the box.

  "Random" means truly random, not pseudorandom. If you want to have a certain proportion of trials with one delay and a certain proportion with another, you could do it like this... Suppose you want *roughly* 40 trials with a 2-s delay and 20 trials with a 4-s delay, mixed at random. You could specify level 1 = 2 s, level 2 = 2 s, level 3 = 4 s, and choose levels at random for a total of 60 trials. They would be chosen at random (so you're not guaranteed exactly 40/20 trials, but on average they would be in a 2:1 ratio and the mean values would be 40 and 20).

  (If, instead, you wanted 40 trials with a 2-s delay *followed by* 20 trials with a 4-s delay, then you'd specify level 1 = 2 s, level 2 = 4 s, and increase levels every 40 trials, for a total of 60 trials.)

  "Draw without replacement" gives a pseudorandom system. It works just like the draw-without-replacement system used for the location (see above). For more explanation of this topic, see randomness, pseudorandomness, and drawing without replacement.

- **Simultaneous presentation** of phase 1 and phase 2 stimuli can also be achieved. In this variant of the task (Weed et al. 1999 *Cog Brain Res* **8:** 185; Robbins et al. 1997 *Psychopharm* **134:** 95), phase 1 proceeds as normal. On completion of phase 1, phase 2 is then immediately presented, but the phase 1 stimulus persists in its original location; touches to this stimulus are ignored, and the subject must touch the *other* stimulus that is visually identical to it but in a different location. Simultaneous presentation is triggered by specifying a **negative delay** in the delay list.

Phase 2

- **Matching.** If this is ticked, the task is delayed matching to sample. Otherwise, it's delayed non-matching to sample.
  - For DNMTS, the original object is shown together with one novel object.
  - For DMTS, the original object is shown together with **zero** (for training only), **one, two, or three** novel objects ('distractors'), or from 17-Nov-2004, up to **seven** distractors. Choose how many novel objects you want to present in Phase 2. (But note: since the automatic stimulus-varying procedures available to DMTS generate stimuli in groups of 4, the stimuli generated by these procedures with other total numbers of stimuli per trial are not guaranteed to follow the same rules as with 4 total stimuli per trial.)
    - **Specify a list of the possible number of distractors,** just as you specified a list of

delays (see above).

- **Choose how you would like the program to pick the number of distractors from your list.** There are options to use only the first number in the list; to choose the number of distractors randomly from the list for each trial; to progress steadily through the list until the last one is reached (whereupon the program will stay on the last value); to progress through the list when the subject gets Y of the last 20 trials correct (you specify Y; when the program reaches the end of the list, it stays there); and to draw without replacement from your list (multiplied by some multiplier that you specify, just as for the the other draw-without-replacement systems discussed above). For more explanation of this topic, see randomness, pseudorandomness, and drawing without replacement.
- **Note** that if you use an option that varies the number of distractors within the task, and you are using stimuli generated by manipulating a predefined stimulus set, as discussed above, the exact stimuli used may vary across sessions, particularly if the number of distractors depende on the subject's performance (e.g. "progress to a new number of distractors when the subject gets Y of the last 20 trials correct").

- **Rotate target and distractors.** If ticked, all phase 2 stimuli will be rotated by 90 degrees, left or right at random. Currently this option only works for stimuli of the type CamcogQuadPattern, which includes all predefined stimuli of the Camcog D(N)MTS, PAL, STAR, and ID/ED sets.
- **Correction phase if subject fails Phase 2.** Optionally, phase 2 can be repeated immediately (once) if the subject fails it the first time.
  - The correction phase is identical to Phase 2, and should begin immediately after any punishment delivered for Phase 2 is complete. Since the correction phase is identical to Phase 2, the stimuli are shown at the same locations. Therefore, the correction phase cannot sensibly be started until the subject has removed its finger from the screen following phase 2. The option "**Wait no longer than X seconds for finger removal before proceeding (0 no limit)**" controls this behaviour. If set to the default of zero, meaning "no limit", the program waits until the finger is removed. However (as of Apr 2006) there have been problems with this feature that have not yet been identified; they may be hardware faults in some particular touchscreens. The result is that the "finger removal" message appears not to get through to MonkeyCantab, so it waits indefinitely to start the correction phase. This option allows you to specify a limit to this time. If you specify 60 s, for example, then MonkeyCantab will proceed with the correction phase as soon as the finger is removed following phase 2, or when 60 s have elapsed following phase 2, whichever comes first.
- **Locations.** Here you can specify (by clicking **Set**) the location(s) that your Phase 2 stimuli may be displayed in. The possible locations range from 0 (top left) to 8 (bottom right), as illustrated in the *9-way grid* picture shown in the Size, coordinates, and grids section. Typically, locations 0, 2, 6, and 8 (the top left, top right, bottom left, and bottom right) are used.
- **Maximum time to wait for a response.** If the subject fails to make a response within this time, the subject fails the trial. (This time limit applies to Phase 2, and the correction procedure if one is used.)
- **Choose location at random, or draw without replacement?** By default, on each trial, the Phase 2 TARGET stimulus location is chosen at random from the possibilities listed in the Locations box (see above). If you tick "Don't choose at random; draw without replacement", then the program uses the same draw-without-replacement system as described above for the Phase 1 locations. For more explanation of this topic, see randomness, pseudorandomness, and drawing without replacement. Note that the location of the target stimulus is drawn according to this system; the locations of the distractor stimuli are always chosen at random from the other possible locations. Note also the caveats above about drawing without replacement, i.e. that it makes location *informative* to the subject, where it previously wasn't. (My personal view is that random distributions are preferable to drawing without replacement for this reason.) If a subject never reaches Phase 2, the location that *would* have been used for the Phase 2 Target is used for the next trial.

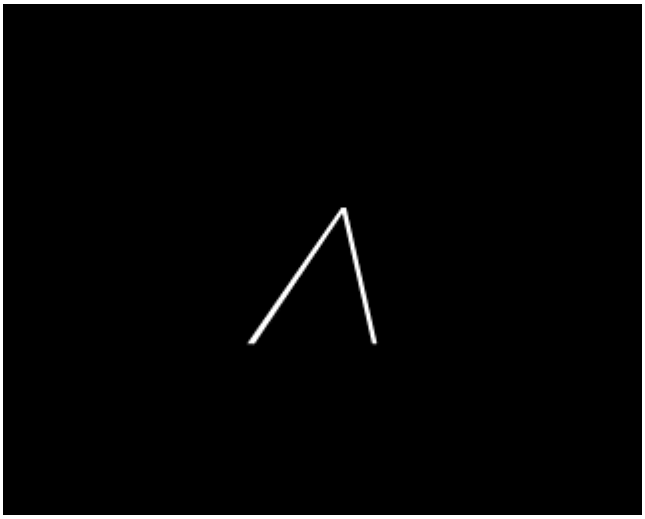
- **Repeat trial (give correction trial) if Phase 1 or Phase 2 failed, until subject succeeds at both phases.** If ticked, then failure at either Phase 1 or Phase 2 triggers an exact repeat of the

whole trial (total trial count permitting) until the subject succeeds at both Phase 1 and Phase 2.

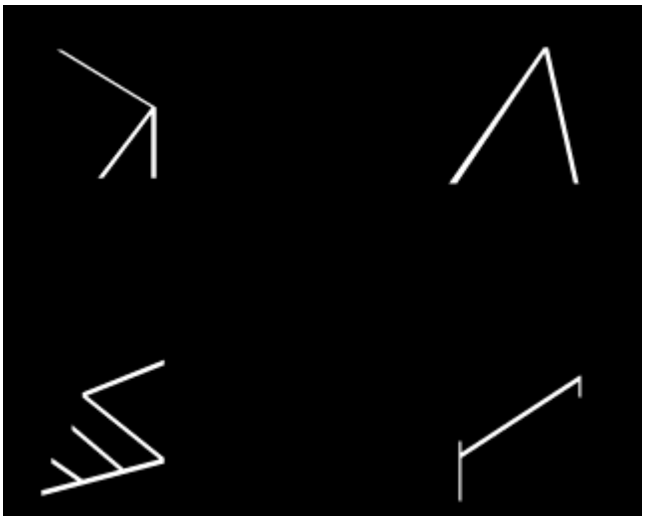
Options for reward and punishment are set in the General Parameters section; visual objects are defined in the Visual Object Library.

Screenshots from the task

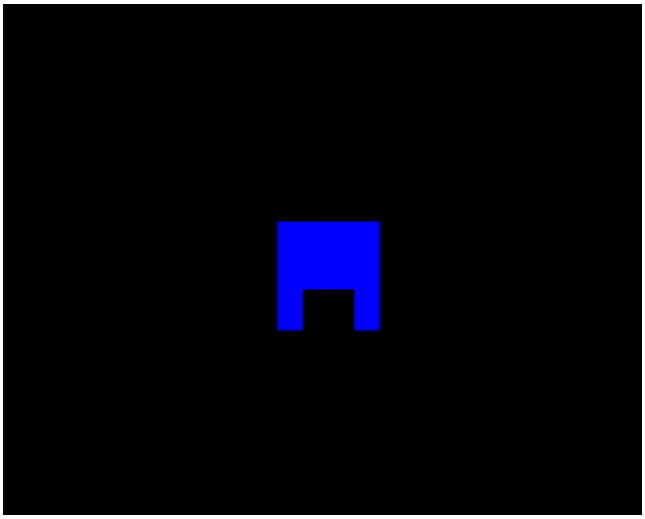
Phase 1



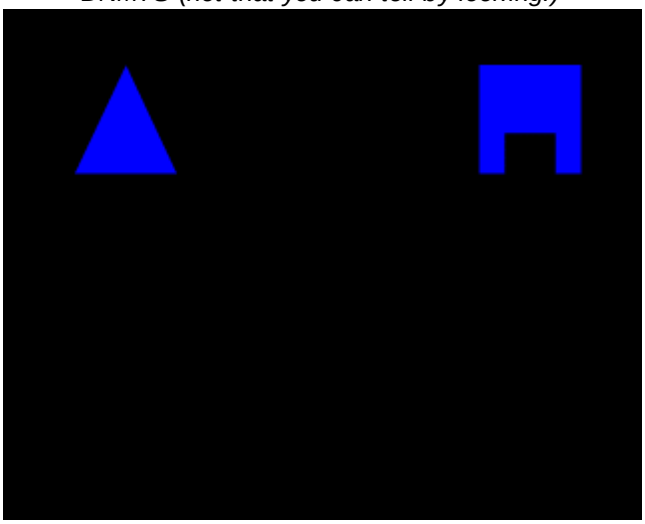
Phase 2: DMTS with three incorrect stimuli



Phase 1: a different stimulus...



DNMTS (not that you can tell by looking!)



1.8.6 List-based Delayed Matching/Non-Matching to Sample

About the task

This task is a variant of the normal D(N)MTS task. Suppose you want to test many long delays (e.g. 5 minutes, 10 minutes, and 15 minutes). In the conventional task, in which SAMPLE and CHOICE phases are paired, this would be slow:

SAMPLE 1 ... CHOICE 1 (5 minutes plus response times)
SAMPLE 2 ... CHOICE 2 (10 minutes plus response times)
SAMPLE 3 ... CHOICE 3 (15 minutes plus response times)
... total 30 minutes plus response times

But the task could potentially be run in a more time-efficient manner, also requiring the subject to memorize several samples at once:

SAMPLE 3 ...
SAMPLE 2 ...
SAMPLE 3...
CHOICE 3

CHOICE 2
CHOICE 3
... total 15 minutes plus response times

The ListDMS task implements a generic algorithm that takes a large set of "dumbbell"-shaped temporal objects:
SAMPLE-gap-CHOICE
or XXXXX----------XXXXXX
and schedules them to minimize the overall total time, without any of the "XXX" parts overlapping.

Several scheduling methods may be used, including
"stack"
XXXXXXX--------XXXXXXXXXX
XXXXX-----XXXXXX

"nest"
XXXXXXXX-------------------XXXXXXXXXX
XXXXX------XXXXXX

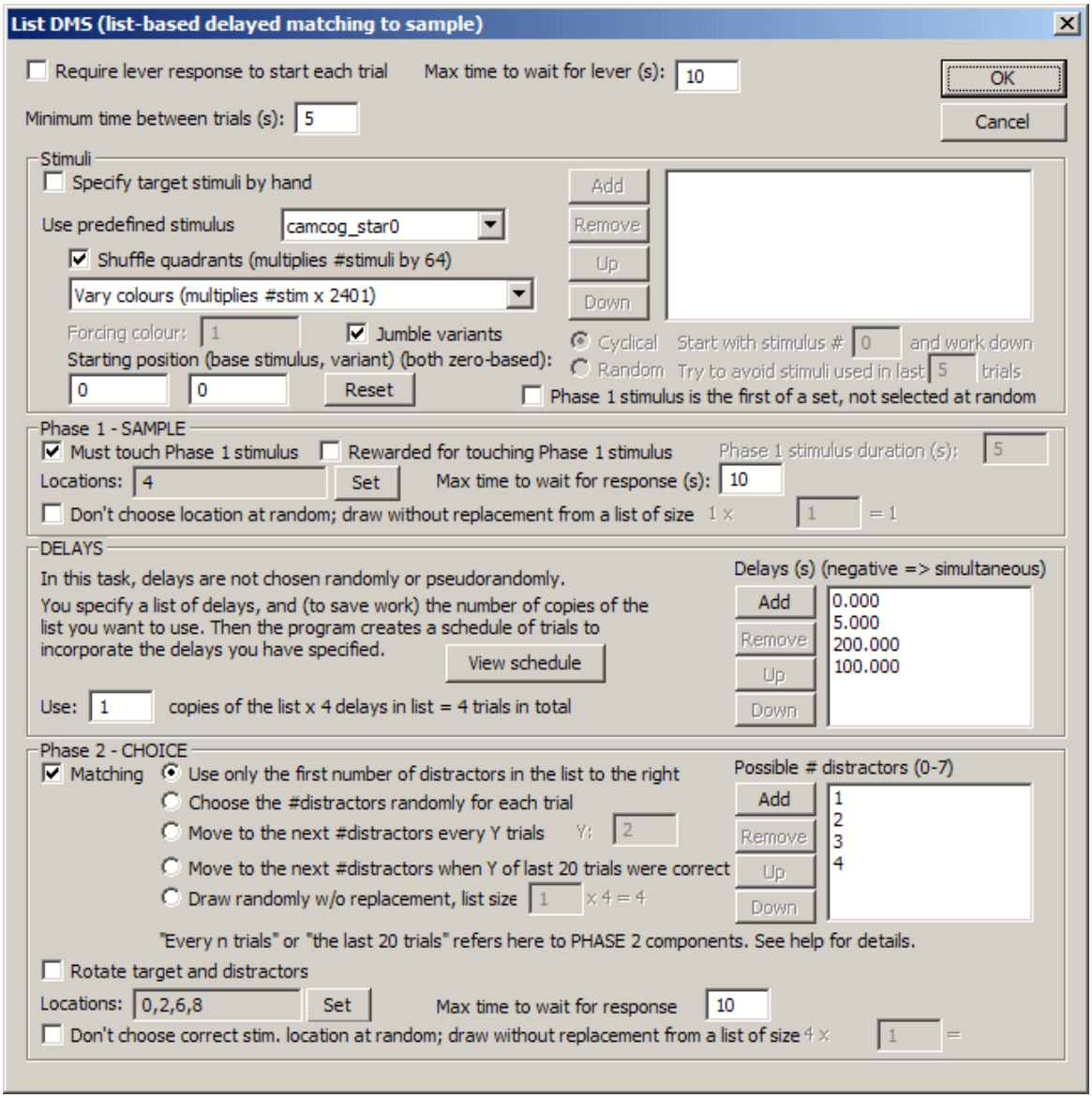
"overlap"
XXXXXXXX----------------XXXXXXXX
XXXXXXX----------------------XXXXXXXX

If you're not interested in how the scheduler works, skip this paragraph. The schedule is determined **completely in advance.** Therefore, we need to know how long each part of each trial's XXX-------XXXX structure is. The **memory delay** begins at the moment that the sample phase (Phase 1) ends - because that's when the stimulus vanishes, and the subject must begin to rely on its memory. Therefore, the scheduled "dumbbell", with its two "elements" to be scheduled, is made up as follows:

- *first element length* = maximum Phase 1 (sample phase) time *(the scheduler takes into account whether lever initiation is being used, and whether the subject has to respond to Phase 1, and any reward time if Phase 1 is being rewarded, and so on)*
- *gap length* = memory delay MINUS first element length *(because the memory delay could start almost immediately after Phase 1 begins, if the subject responds quickly, or right at the end, if it responds late)*.
- *second element length* = maximum Phase 2 (choice phase) time PLUS maximum Phase 1 time *(for the same reason as above) (the scheduler also takes into account the minimum intertrial time, and the maximum reward/punishment time)*.

A few parts of the conventional D(N)MTS task are incompatible with this process (such as specifying the session length - this now becomes a consequence purely of the trials scheduled). The "correction" phase is removed. The maximum time between trials is now a consequence of the trial schedule and cannot be specified manually. The sequence of trial delays is now determined by the scheduler, not by the user. There are slight modifications to the way that the number of distractors can progress (but only to the extent of making the task logically consistent). Otherwise, the ListDMS task implements all the options of the D(N)MTS task.

Configuring the task



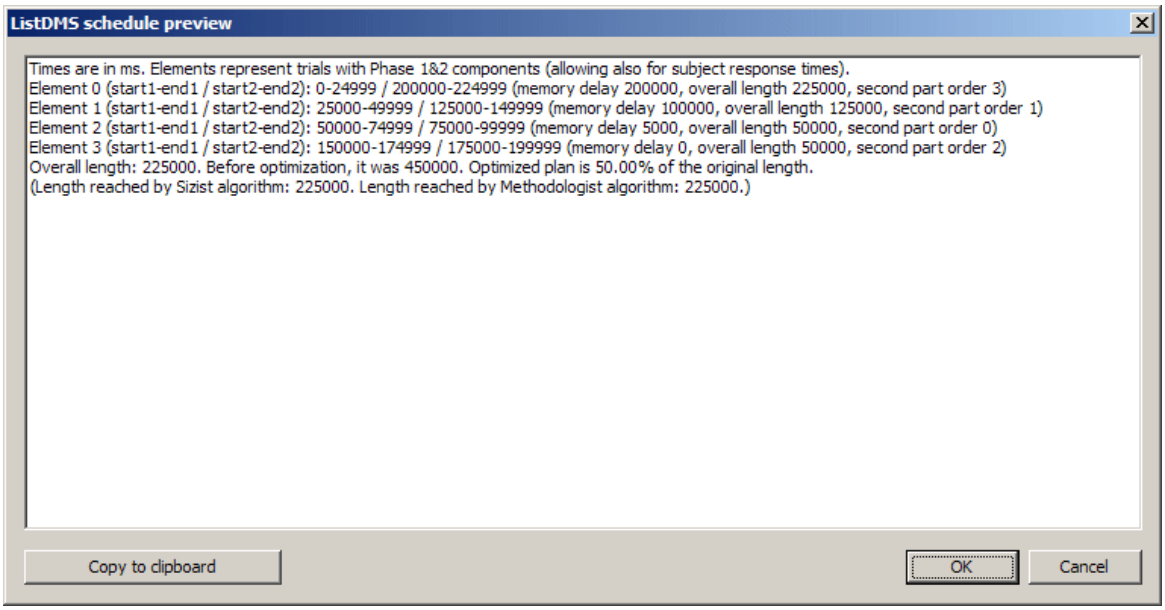
- **All the parameters are the same as the D(N)MTS task, with the exception of the following.**
- *Maximum number of trials: REMOVED.*
- *Maximum time: REMOVED.*
- *"Try to avoid stimuli used in the last N trials":* Stimuli for both phase 1 (sample) and phase 2 (choice) are assigned at the moment that Phase 1 begins. Therefore, "used in the last N trials" may include some trials for which Phase 1 has been presented, but Phase 2 is yet to be presented. Other than this quirk, the functionality is identical to the D(N)MTS task.
- *Time between trials: can specify the **minimum** only, not the maximum.* Try to make this as short as possible, to keep the overall session time down and to allow more space to squeeze in trials interleaved among each other.
- **Delays** are specified in a simple list, with a list multiplier (how many copies of the list are to be used?). This determines the number of trials, and the schedule. For example, if you have 4 delays in the list, and use 4 copies of the list, you will get 16 trials. You can enter the same delay into the list several times, if you wish. Delays can be zero. As in the D(N)MTS task, setting a delay as negative means a zero delay with the sample stimulus shown alongside the choice stimuli in Phase 2 ("simultaneous" variant).
- Once you've specified the delays and the multiplier, you can have a look at the schedule that'll be used: click **View Schedule**. More detail is given below.
- The options for the **number of distractors** are the same as the D(N)MTS task, except for a

caveat about two options: "Move to the next #distractors every Y trials" and "Move to the next #distractors when Y of last 20 trials were correct". The need for a change is obvious: since sample and choice phases may now occur in novel sequences. Since the number of distractors is *assigned* internally when Phase 1 begins, these options are changed as follows. (1) The **"Move to the next #distractors every Y trials"** option works as usual, except that the sequence of distractors is an orderly progression from the point of view of the Phase 2 elements (and therefore not necessarily of the Phase 1 elements) *even if that Phase 2 is never presented (because the subject fails Phase 1)*. (2) The **"Move to the next #distractors when Y of last 20 trials were correct"** option refers to the last 20 choice phases (Phase 2s), *at the time that the trial's Phase 1 begins*; this clearly isn't quite as sensible as in the D(N)MTS task and it is probably best not used.

- *The option to repeat all or parts of a trial (give a correction trial) is removed, as this unpredictability would mess up the scheduling.*

More on the schedules

A sample schedule looks like this:



This particular example is for **1** copy of a list with delays **0, 5, 100, 200** sec (with response time limits, reward times, etc., set in the rest of the task parameters, as described above, and in the General Parameters). The scheduler has done this:

```
0      25     50     75     100    125    150    175    200
SAMPLE0.................................................CHOICE0
(this will be TRIAL 0)
.......SAMPLE1.....................CHOICE1
(this will be TRIAL 1)
..............SAMPLE2CHOICE2
(this will be TRIAL 2)
.........................................SAMPLE3CHOICE3
(this will be TRIAL 3)
```

Note also that extra time is allowed for response times, reward, etc.; therefore, trials with short memory delays appear to have no gap scheduled (all this means is that the potential variability in subjects' responses means that there is no gap into which another trial, or component of a trial, could potentially be scheduled).

There appears to be no gap between, for example, SAMPLE0 and SAMPLE1 - but the scheduler has already incorporated the "minimum time between trials" into the end of each segment, so this gap will be present (from the scheduler's point of view, SAMPLE0 ends when the sample is over, and then any time that the sample *could* have taken if the subject were a bit slower to respond [if you're making your subject respond to sample phases], plus the minimum time between trials, and any reward time, etc.).

You can take a copy of the schedule for the clipboard if you wish (though, of course, it is also saved in the results textfile, and equivalent information is saved to the results database).

In the "Chronological Order" section of the results (e.g. the ListDMS_ChronologicalOrder table in the database), this structure will appear as follows, if the subject responds to all trials:

SegmentNumber,SegmentStartTimeMs,Trial,Phase
0,...,0,1
1,...,1,1
2,...,2,1
3,...,2,2
4,...,1,2
5,...,3,1
6,...,3,2
7,...,0,2

In the trial-based results (e.g. the ListDMS_Results table in the database), **all** trials appear, even if they were not given; look for the **Phase1Given** and **Phase2Given** fields to see if they were actually delivered. Look at the **OrderInPhase2Sequence** field to quickly determine the sequence of Phase 2 components (this is the number shown as "second part order" in the schedule description shown above). Phase 1 components are given in the order that the trials appear in the results.

1.8.7 Self-Ordered Search (a.k.a. Spatial Working Memory)

About the task

A number of identical objects are presented in up to sixteen different locations. The subject must touch each one, but must not touch an object twice.

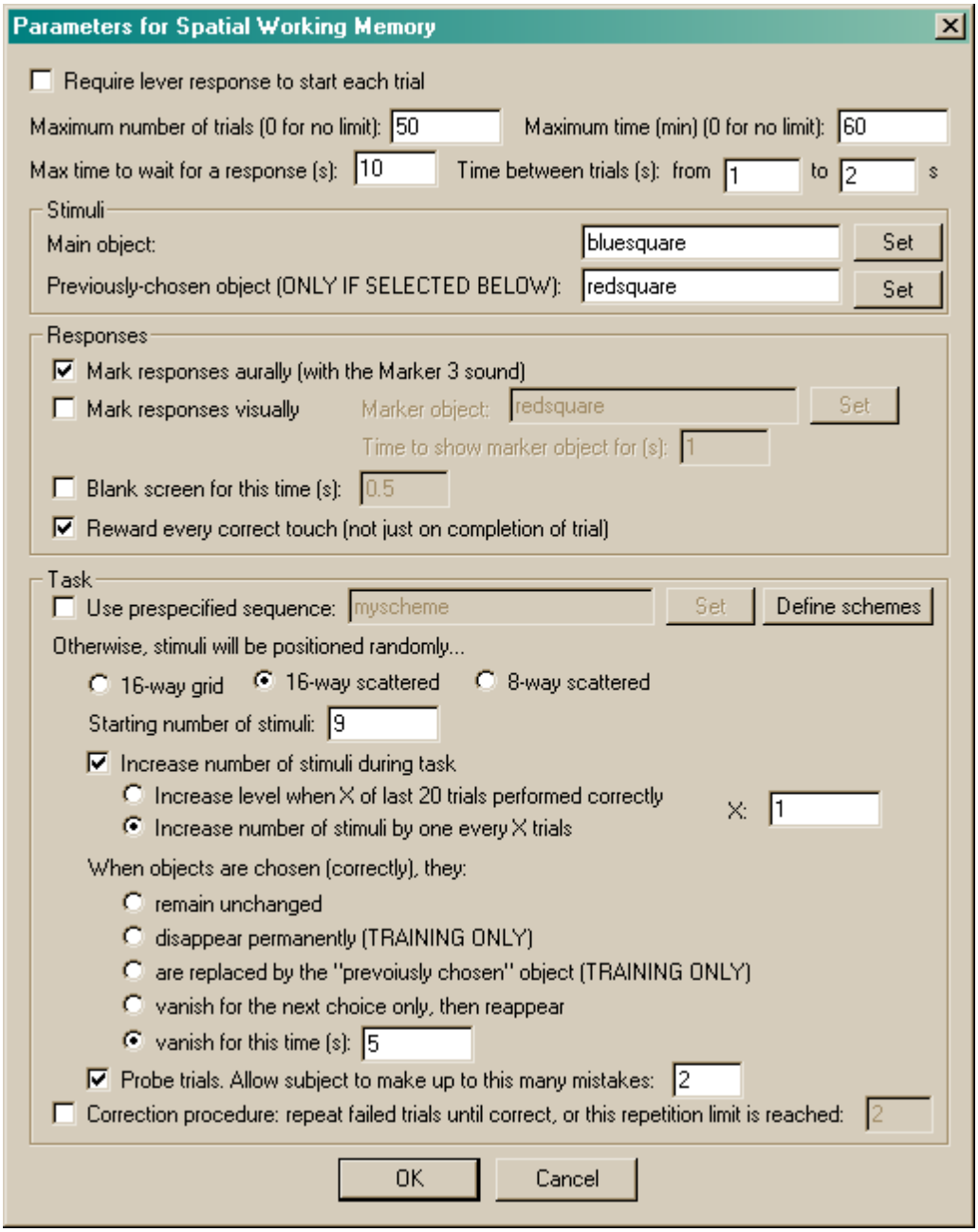
- A marker tone (Marker 1) indicates the trial onset.
- When the subject touches an object, a different ("marker") object can be presented for a defined duration. (In typical human testing, the primary objects are boxes and the marker object is envisaged as "what's in the box".)
- The subject is rewarded after a correct sequence of responses.
- It's punished for repeating a response.
- It's punished if a response isn't made within a criterion time of the last response

Difficulty is set by the number of stimuli (typically beginning with 2).
Optionally, the level increases after a criterion (of X correct responses out of the last 20 trials).

This task is based on Collins P, Roberts AC, Dias R, Everitt BJ & Robbins TW (1998). Perseveration and strategy in a novel spatial self-ordered sequencing task for non-human primates: effects of excitotoxic lesions and dopamine depletions of the prefrontal cortex. Journal of Cognitive Neuroscience 10: 332-354.

The commonly-used term "spatial working memory" may not be wholly accurate; "self-ordered [spatial] search" is (as it makes no claims about the psychological basis of the task).

Configuring the task



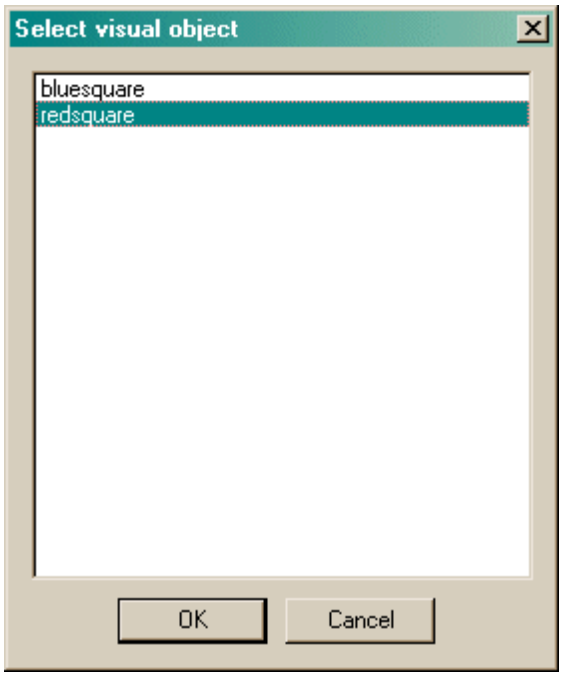
General settings

- **Require lever response to start each trial.** Requires that the subject make a lever response to initiate each trial. See Use with Dogs.
- **Maximum number of trials.** When the subject has performed this number of trials, the task ends. (You may specify 0 for no limit, though you must specify a limit on the number of trials, the time, or both.)
- **Maximum time.** When this time elapses, the task is terminated as soon as the current trial has finished. (You may specify 0 for no limit, though you must specify a limit on the number of trials, the time, or both.)
- **Maximum time to wait for a response.** If the subject fails make a response within this time, the subject fails the trial, which is terminated. (If there are 3 stimuli, and this limit is 10 s, then the subject has 10 s in which to make the first response, and then 10 s in which to make the second, and so on.)

- **Time between trials.** Specify a minimum and a maximum intertrial time (they may be the same). The actual time is chosen with a rectangular probability distribution within these values. The time between trials starts **after** any reward or punishment from the previous trial has finished.

Stimuli

- **Main object.** This shows the object that will be used as the stimulus. Click **Set** to choose the stimulus.
- **Previously chosen object.** When your subject chooses a stimulus, a number of things can happen to it for subsequent choices - it can stay the same, vanish temporarily or permanently, or be replaced by another object. If you want it to be replaced by another object, the replacement object is called the Previously Chosen object (because it marks locations that your subject has chosen previously). Therefore, to use this option (see below), or to use prespecified sequences (schemes), which may also use this option, you must set a Previously Chosen object. Click **Set** to choose this stimulus. If you won't use either schemes or the "replace with the previously chosen object" option, you can ignore this setting.



Responses

- **Mark responses aurally.** If this is selected, the Marker 3 sound will be played to inform the subject that it has touched the picture successfully. (The schedule pauses while this sound is played.)
- **Mark responses visually.** If this is selected, you may replace the response object with another picture for a brief period of time, to indicate visually that the subject has made a successful response. The marker object is shown here; click **Set** to choose one from the visual object library and choose the **Time to show marker object for (s).** The schedule is paused while the marker object is being shown.
- **Blank screen.** If this is selected, then the whole screen goes blank for a predefined time after your subject has chosen. This option can be used to reduce reliance on visual fixation to support spatial memory. If you tick this option, set the **time** for which the screen should be blanked (in seconds).

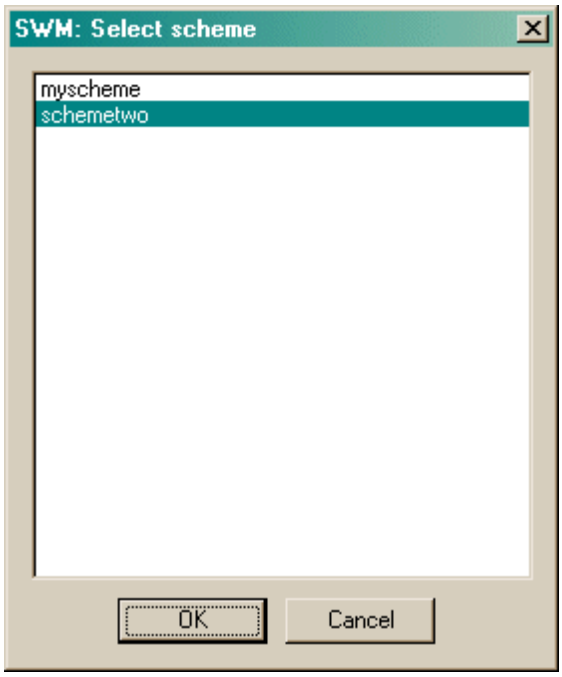
Note the order in which MonkeyCantab executes these options if you choose them all: at the moment of response, the marker object is shown and the marker sound is played. Once soon as

the marker object has been displayed for its specified time, the screen is blanked.

- **Reward every correct touch (not just on completion of trial).** After Weed et al. (1999) *Cog. Brain Res.* **8:**185, this option allows every correct touch to be rewarded, rather than delivering reward only on completion of the whole sequence. Normally, this is disabled (i.e. whole sequences must be completed to obtain reward).
  - *Bugfix modification 11 March 2008: this option allows the delivery of rapid sequences of rewards, if the subject responded rapidly, with a reward being delivered before the previous one finished. This was made most plain when visual flashing stimuli accompanied reward. To prevent this, some mechanism must prevent a second reward occurring until the first has finished. The most flexible way of making this change is simply to* ***disable rewards*** *until the previous reward has finished. I have implemented this by* ***ignoring responses*** *during ongoing reward. You may well wish the stimuli not to be visible while responding is disabled - in which case I suggest you tick the "blank screen" option, blanking the screen for as long as the reward takes (defined in* General Parameters*).*

Task

- **Use prespecified sequence.** If you tick this option, you can click **Set** to choose a predefined SWM trial sequence. (If there are none available, or you want to edit a scheme, click **Define schemes** instead; we will look at scheme definition in a moment.)



If you are not using prespecified sequences (schemes), the following options are available:

- **Grid type.** This may be a 16-way aligned grid, a 16-way scattered pattern, or an 8-way scattered pattern. See Size and coordinates for illustrations; these illustrations also show the numbering of the available positions in the grid.
- **Starting number of stimuli.** Obvious!
- **Increase number of stimuli during task.** If enabled, the number of stimuli will increase as the task goes on. You can either increase the number of stimuli after a fixed number of trials, or when the subject reaches a criterion of a certain number of "correct" trials within the last 20 trials. Fill in this number (either the number of trials, or the number to get right out of 20) in the box labelled X.
- **When objects are chosen correctly...**
  - *Remain unchanged.* The object is not altered. (If you are using visual markers and/or

screen blanking, then this option causes the stimulus to reappear as it was once the marker/blank screen have been displayed.)

- *Disappear permanently.* The object never returns.
- *Are replaced by the "Previously Chosen" object.* The object is replaced by a different object, known as the Previously Chosen object. (You saw above how to define the Previously Chosen object.) One way this feature might be used is to set the Previously Chosen object to a black (or otherwise invisible) rectangle. This differs from the *Disappear Permanently* option: if the object disappears, touches to its former location aren't punished. If the objects is replaced by an invisible but touchable object, touches to this location are punished.
- *Vanish for the next choice only.* The object vanishes for the next choice, but then reappears. This prevents perseveration on the previously-chosen object.
- *Vanish for a specified time.* Chosen stimuli vanish for a certain time (measured from the moment they are touched), and then reappear after a certain time. You'll need to specify this **time,** in seconds. If you are using visual markers, this time must be greater than the visual marker time (so that the object doesn't try to reappear while the marker is still being shown).

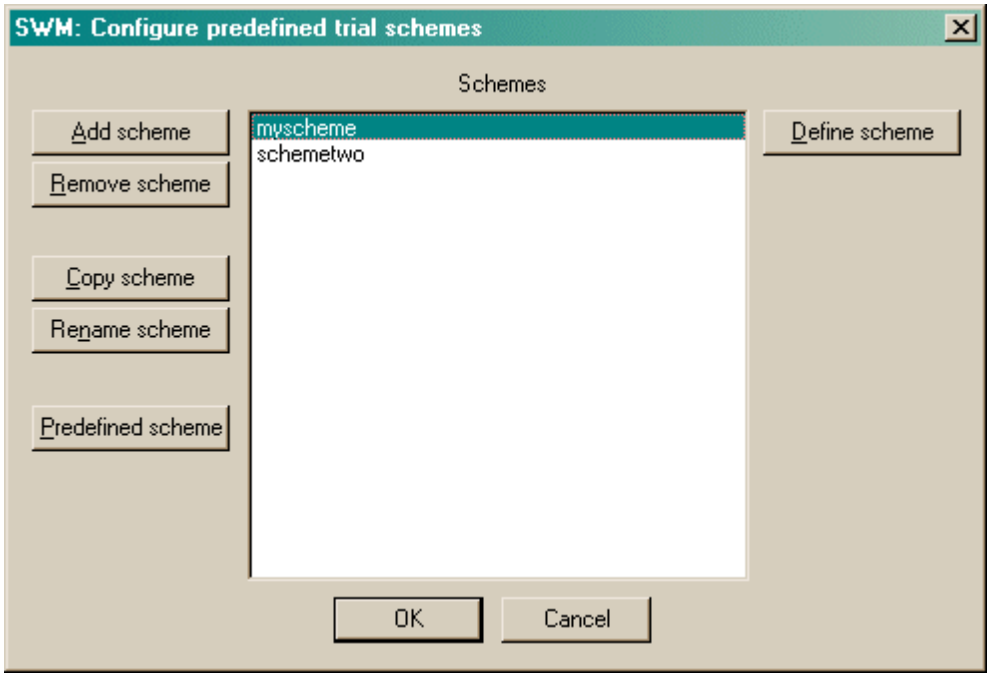
If you are using prespecified sequences (schemes), you will still need to specify the "vanish time" (exactly as above) for trials in your scheme during which you use the "vanish for a specified time" method.

- **Probe trials.** This allows a whole session to use the probe trial facility, without using a scheme. (If you are using a scheme, you can choose whether each trial is a probe trial or not; see below). In a probe trial, subjects are allowed to make a certain number of repetition errors without penalty. Specify the maximum number of errors. (For example, if the maximum number of errors is 3, then the trial will terminate on the 3rd error.)

Options for reward and punishment are set in the General Parameters section; visual objects are defined in the Visual Object Library.

Schemes (predefined trial sequences)

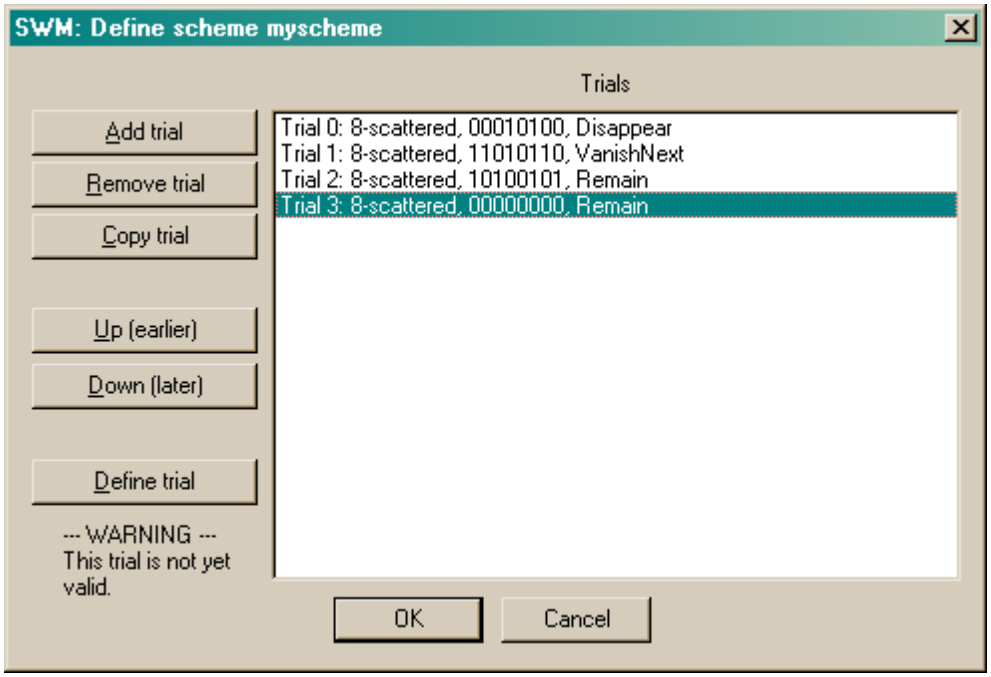
To create or edit schemes, click **Define schemes.** This brings up the following dialogue box listing currently-defined schemes:



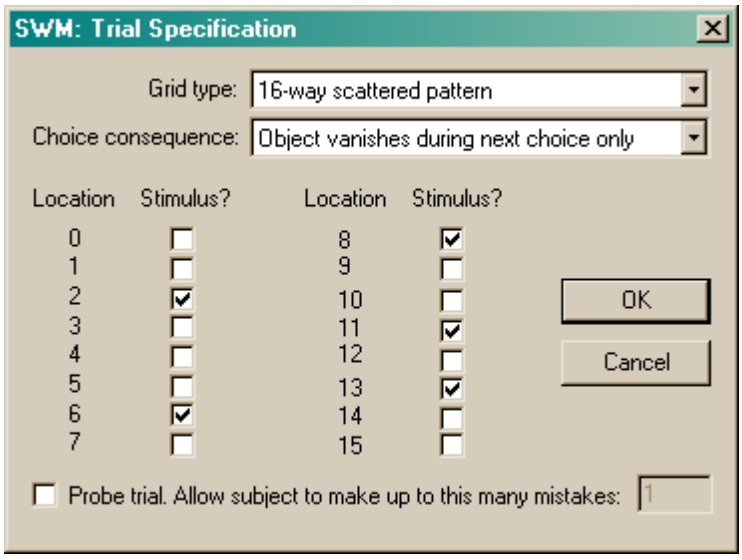
You may add, remove, copy, and rename schemes. Click a scheme and click **Define scheme** to edit a particular scheme. Each scheme is a sequence of trials. If a scheme contains invalid trials (trials with no stimuli), when you click the scheme you will see a warning to that effect.

One scheme is a sequence of trials. When you've clicked **Define scheme,** you can see that sequence of trials. In the example below, there are only two trials:

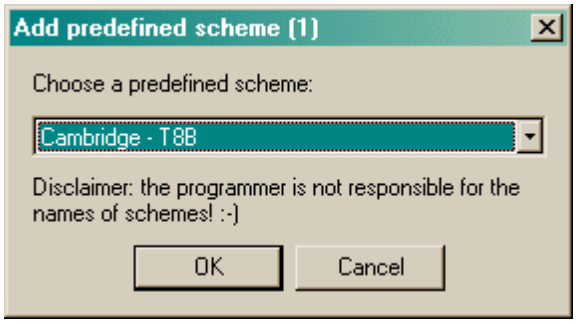
- The first trial, trial number 0, uses an 8-way scattered pattern to display its stimuli. No stimuli are defined at the moment (so the trial's definition includes eight zeroes - "00000000" - and there's a warning being displayed to that effect). Any stimuli that were to be defined would remain on the screen after they'd been chosen.
- The second trial, trial number 1, uses a 16-way scattered pattern to display its stimuli. Five stimuli will be presented, in location numbers 2, 6, 8, 11, and 13, so as locations are numbered from 0-15, the stimulus code reads "0010001010010100". Stimuli that are chosen vanish for the next trial, and then reappear.



You may add, remove, copy, re-order, and define trials. When you click **Define trial,** another dialogue box appears to let you edit the trial. Here's the trial specification for trial 1:



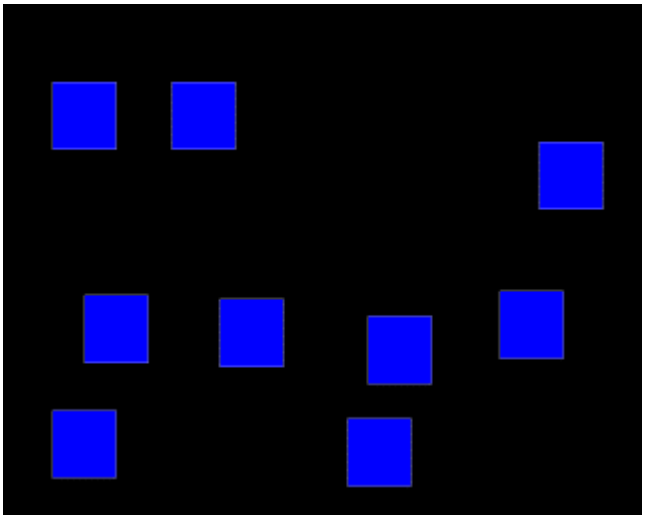
At the main scheme configuration dialogue box, you can also click **Predefined scheme** and choose from a list of built-in schemes:



You will be asked to give the new scheme a name.

Screenshot from the task

Here's an example where ten stimuli are being shown in various locations from the 16-way scattered pattern:



1.8.8 Multiple-Choice Serial Reaction Time

About the task

A marker tone (Marker 1) signals the onset of a trial.

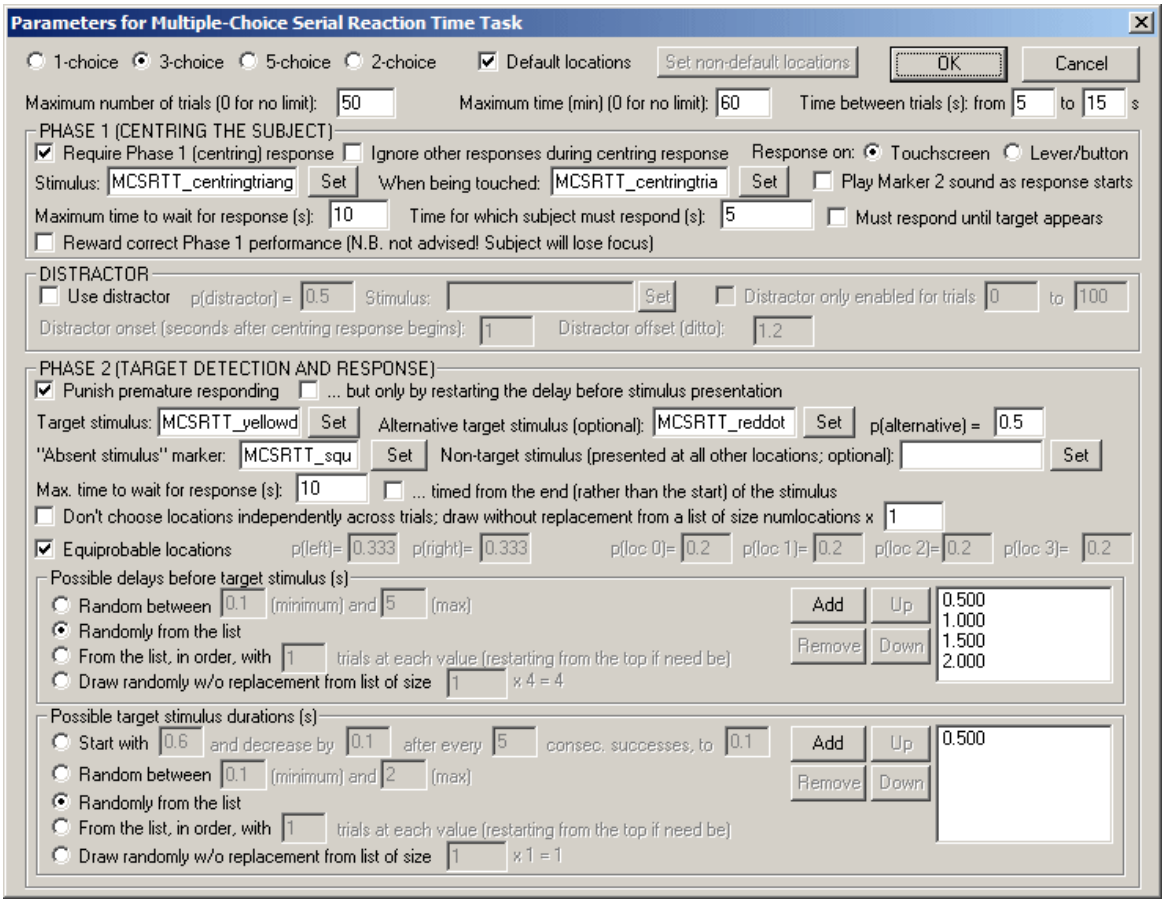
A Centring Object is presented below the centre of the screen, and the subject must touch it, and keep touching for a specified duration. (Optionally, it is rewarded for this.) As an alternative, the subject must respond on a non-touchscreen device (a button or lever) and hold it for the specified duration.

There is then a delay.

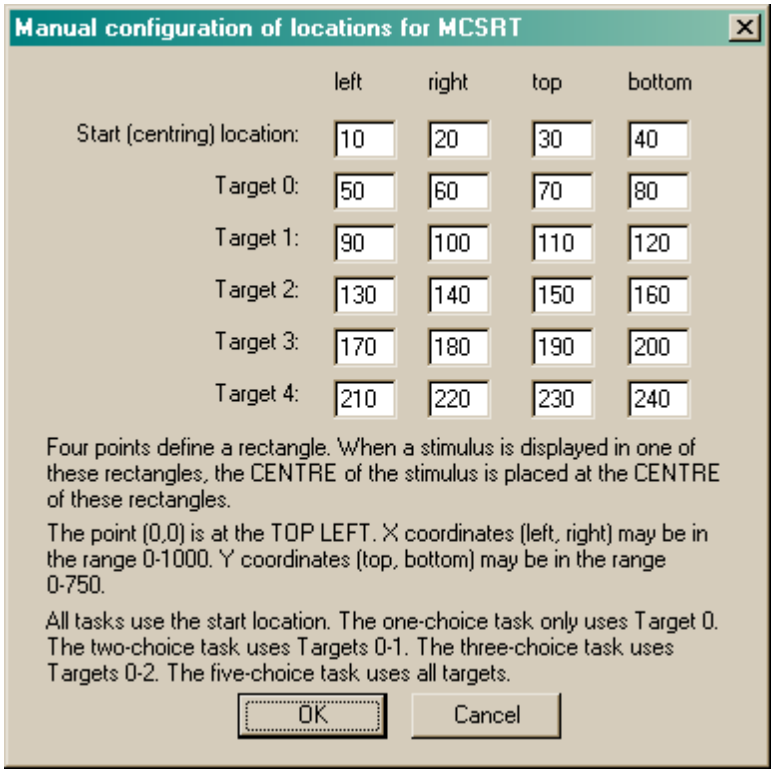
Then a different object is presented in one of 1, 3, or 5 locations for a certain duration (typically brief). The subject must respond (within a timeout), and is rewarded for doing so. It's punished for missing or getting it wrong.

Optionally, you can choose whether the targets are randomly distributed among the locations (e.g. 33% - 33% - 33% with three locations) in the test phase, or are distributed with a different pre-specified distribution. There are probably not very many good reasons for using an uneven distribution, so think twice before using it!

Configuring the task



- **One-, two-, three-, or five-choice task.** See the Size and coordinates section for a description of the arrangement of stimuli. (Note that these options are out of the logical order for back-compatibility with previous configuration files! :-)
- **Default locations.** If ticked, the usual patterns of stimuli are offered - a single central target for the one-choice task, two targets (near the bottom left and bottom right of the screen) for the two-choice task, three targets in a row for the three-choice task, and a ring of five targets for the five-choice task. If a Phase 1 stimulus is used, by default this is below the targets and centred, except for the five-choice task, in which case it is in the centre of the ring. The default locations are described more precisely in the Size and coordinates section. If you untick the "default locations" option, you can click a button to **set non-default locations,** using a separate dialogue box, as shown below.



- **Maximum number of trials.** When the subject has performed this number of trials, the task ends. (You may specify 0 for no limit, though you must specify a limit on the number of trials, the time, or both.)
- **Maximum time.** When this time elapses, the task is terminated as soon as the current trial has finished. (You may specify 0 for no limit, though you must specify a limit on the number of trials, the time, or both.)
- **Time between trials.** Specify a minimum and a maximum intertrial time (they may be the same). The actual time is chosen with a rectangular probability distribution within these values. The time between trials starts **after** any reward or punishment from the previous trial has finished.

PHASE 1 (CENTRING THE SUBJECT)

- **Require a Phase 1 (centring) response.** If ticked, the subject must make a response to centre it. If not, the task proceeds straight to Phase 2, and any distractors are timed from then as well.
- **Phase 1 - Ignore other responses during centring response.** If ticked, then any responses to the target locations during the centring response are ignored. Responses to the background (non-target areas) of the touchscreen are also ignored, unless the centring responses is made on the touchscreen, in which case a response to a non-target area is treated as the subject having released the centring response.
- **Phase 1 - Response device.** Would you like to have the subject be centred by holding on the touchscreen, or on a digital device (lever/button), before Phase 2 will begin?
  - **Phase 1 - Stimulus (to be touched).** Applicable if you are using a touchscreen response in Phase 1. Choose the centring stimulus (the one that the subject must touch for a while before the targets appear). Click **Set** to choose the stimulus.
  - **Phase 1 - Stimulus (shown while touch is held).** Applicable if you are using a touchscreen response in Phase 1. Choose the stimulus that is shown *while* the subject is touching. Click **Set** to choose the stimulus.
- **Phase 1 - Play Marker 2 sound as response starts.** If ticked, the Marker 2 sound (as defined in the General Parameters) is played as the subjects begins to make the Phase 1 response.

- **Phase 1 - Maximum time to wait for a response.** If the subject fails to respond within this time, the subject fails the trial.
- **Phase 1 - Time for which subject must make response.** Choose the length of time for which you want your subject to keep its nose/finger on the centring stimulus.
- **Phase 1 - Must respond until target appears.** If ticked, the subject must hold until the target itself is presented. In this case, Phase 2 is initiated as soon as the centring response is made, and any distractors are timed from this point as well.
- **Phase 1 - Reward correct performance.** Optionally, you can reward your subject for passing Phase 1 - *but I [RNC] don't recommend it, as your subject wouldn't then be facing the right way to watch Phase 2!* If the subject must respond until the Phase 2 target appears, then the program will not allow you to reward Phase 1 (as reward would then arrive at the same time as the target).

DISTRACTOR

- **Use distractor?** Optionally, you can present a distractor stimulus in a non-target location (chosen at random) before the target is shown.
  - **p(distractor).** Choose the probability of a distractor stimulus on any given trial.
  - **Distractor stimulus.** Set the stimulus you'd like to use as a distractor.
  - **Distractor onset.** Set the distractor onset time, relative to the *start of the Phase 1 holding response.* If you want the target and non-target locations to be indicated before the distractor appears (which is sensible), the distractor onset time will need to be greater than the time for which the subject must make the Phase 1 response.
  - **Distractor offset.** Set the distractor offset time, also relative to the start of the Phase 1 holding response. The distractor offset time must be greater than the distractor onset time. Since the distractor mustn't be present when the target is shown, the distractor offset time must be less than (phase 1 holding response time + minimum delay from start of phase 2 to target stimulus). [These delays are set below; see *Possible delays before target stimulus*.]
  - **Distractor only enabled for certain trials.** If ticked, you can specify the first and last trials (numbered from 0 onwards, i.e. the first trial of the session is trial 0) on which the distractor is enabled. The probability of distraction still applies - so if you want to have no distractor except for trials 29-59, when a distractor should always be given, then you should tick this box, fill in "29" as the first trial (the thirtieth) and "59" as the last trial (the sixtieth) and ensure that *p(distractor)* is set to 1.

  Note: if the non-target stimulus is being displayed when the distractor comes on (i.e. if the distractor onset time exceeds the centring response hold time), the distractor stimulus is presented *on top of* the non-target stimulus (not instead of it).

PHASE 2 (TARGET DETECTION AND RESPONSE)

- **Phase 2 - Punish premature responding.** If the subject responds in one of the possible target locations before the target is presented, this is termed a premature response. You can punish these (in which case the trial will be terminated).
  - **... but only by restarting the delay before stimulus presentation.** This option allows premature responses to be punished but with no overt stimulus changes. If ticked, then a premature response will restart the delay preceding the target, but will not terminate the current trial. No further distractors will be scheduled for the current trial, however.
- **Phase 2 - Target stimulus.** Choose the target stimulus by clicking **Set.**
  - Optionally, you can have an **alternative stimulus** replace the usual target stimulus, and specify the **probability** that the alternative, rather than the "main", target will be used on any given trial. (Apart from their visual appearance, the main and alternative target stimuli behave identically. This option is to allow you to have visually different versions of the target, intermixed randomly.)
- **Phase 2 - "Absent stimulus" marker.** During the delay, while the target is being presented,

and afterwards, any locations that aren't showing the target show this marker stimulus. Choose the stimulus by clicking **Set.** Typically, this stimulus would be an empty box.

- **Phase 2 - Non-target stimulus.** Optionally, during the presentation of the target stimulus, non-target stimuli (rather than absent-stimulus markers) can be displayed at all other locations, to make the task harder. Specify the non-target stimulus here, or leave it blank if you don't want to use this feature.
- **Choose locations independently across trials, or draw without replacement?** By default, on each trial, the Phase 2 target location is chosen at random. If you tick "Don't choose locations independently across trials; draw without replacement...", then the program behaves as follows. It makes a list containing N copies each of all the possible locations, where N is the number you can type into the box (labelled "... from a list of size numlocations x **N**"). For each trial, the target location is drawn at random, without replacement, from this list. When the list is emptied, it is repopulated with numlocations x N entries as before. The idea behind this is to allow you to prevent the program from choosing the location completely at random, instead ensuring an exactly equal distribution of locations across trials, nevertheless with some random element from trial to trial. (Obviously, if you specify N=1, then if you have L possible locations, each possible location will be used once in every L trials.) The larger N is, the closer the system is to true random sampling (i.e. an infinite N is equivalent to unticking this option). Be aware, however, that true random choice (choice independent for each trial) means that your subject cannot predict anything on the basis of past locations (even if the distribution of locations across trials is not exactly even as a consequence of random sampling), but with a draw-without-replacement system, location *does* become informative. For more explanation of this topic, see randomness, pseudorandomness, and drawing without replacement.
- **Phase 2 - Equiprobable locations? (Only applicable if you are not using the draw-without-replacement system.)** If ticked, p(target on the left) = p(target in the middle) = p (target on the right) = 1/3, for the three-choice task; similarly, p(each location) = 0.2 in the five-choice task, and p(left) = p(right) = 0.5 for the two-choice task. If you wish, you can force the location probabilities to something else by removing this tick and filling in the probability information. For the three-choice task, specify **p(left)** and **p(right)** boxes. Obviously, p(middle) = 1 - p(left) - p(right) in these circumstances. For the five-choice task, fill in **p(location 0)** through **p(location 3)** and p(location 4) will be calculated for you. For the two-choice task, specify p(left), and p(right) will be calculated as 1 - p(left) when the task runs.

  Note that this does not apply to the distractor stimulus, which is always equiprobably located in one of the non-target locations.

- **Phase 2 - Maximum time to wait for a response.** This time is measured from the *start* of the target stimulus, unless otherwise specified (see next option). If this time limit expires and the subject hasn't responded, it fails the trial.
- **Phase 2 - ... timed from the end (rather than the start) of the stimulus.** If ticked, the maximum time to wait for a response (see previous option) is measured from the *end* of the target stimulus, rather than from the start.

- **Possible delays before target stimulus.** There are several ways to specify the possible delays (in seconds):
  - **Random between...** Every trial, a delay is chosen at random between the two specified (minimum and maximum) values. The probability density function is rectangular within that range.
  - **Randomly from the list.** Every trial, a delay is chosen at random from the list shown to the right.
  - **From the list, in order.** Every trial, a delay is chosen from the list, starting with the top value in the list. You may specify the number of trials for which each list value is applied (e.g. if you specify 5, then the subject will get 5 trials with the first delay, then 5 with the next, then 5 with the third... until the list is exhausted, at which point the program starts again from the top of the list).
  - **Draw without replacement.** This is a pseudorandom selection technique. A list is

populated with the possible alternatives. In fact, it's populated with *n* copies of each of the possible alternatives, where *n* is the *multiplier* that you can type into the box. When a value is required, one is drawn at random from the list, without replacing it back in the list. If the list ever runs out of values, it is repopulated as before. If the multiplier is 1, then each of the *x* alternatives is used once in every *x* trials. If the multiplier is *n*, then each of the *x* alternatives is used *n* times in every *nx* trials. If the multiplier is very large, this system approximates true random selection. For more explanation of this topic, see randomness, pseudorandomness, and drawing without replacement.

Use the **Add, Remove, Up,** and **Down** buttons to edit the list of possible delay values.

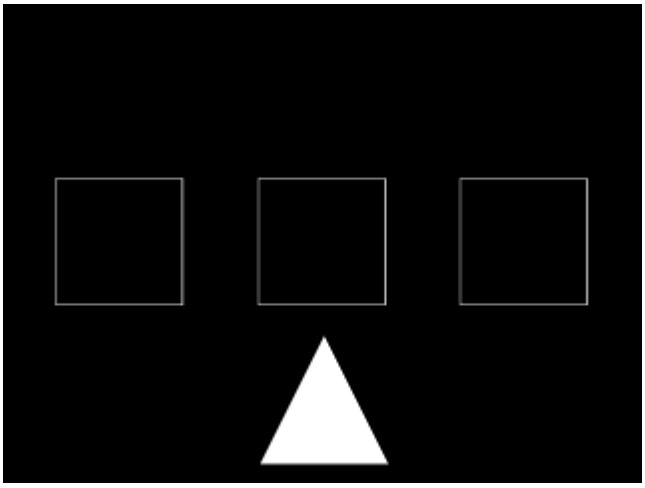
- **Possible target stimulus durations.** There are several ways to specify the possible stimulus durations (in seconds):
  - **Start with...** The program starts with the specified stimulus duration. When the subject gets a certain number of trials correct consecutively, the stimulus duration is decreased by a certain amount, down to a specified minimum value.
  - **Random between...** Every trial, a stimulus duration is chosen at random between the two specified (minimum and maximum) values. The probability density function is rectangular within that range.
  - **Randomly from the list.** Every trial, a stimulus duration is chosen at random from the list shown to the right.
  - **From the list, in order.** Every trial, a stimulus duration is chosen from the list, starting with the top value in the list. You may specify the number of trials for which each list value is applied (e.g. if you specify 5, then the subject will get 5 trials with the first stimulus duration, then 5 with the next, then 5 with the third... until the list is exhausted, at which point the program starts again from the top of the list).
  - **Draw without replacement.** This is a pseudorandom selection technique. A list is populated with the possible alternatives. In fact, it's populated with *n* copies of each of the possible alternatives, where *n* is the *multiplier* that you can type into the box. When a value is required, one is drawn at random from the list, without replacing it back in the list. If the list ever runs out of values, it is repopulated as before. If the multiplier is 1, then each of the *x* alternatives is used once in every *x* trials. If the multiplier is *n*, then each of the *x* alternatives is used *n* times in every *nx* trials. If the multiplier is very large, this system approximates true random selection. For more explanation of this topic, see randomness, pseudorandomness, and drawing without replacement.

  Use the **Add, Remove, Up,** and **Down** buttons to edit the list of possible stimulus duration values.

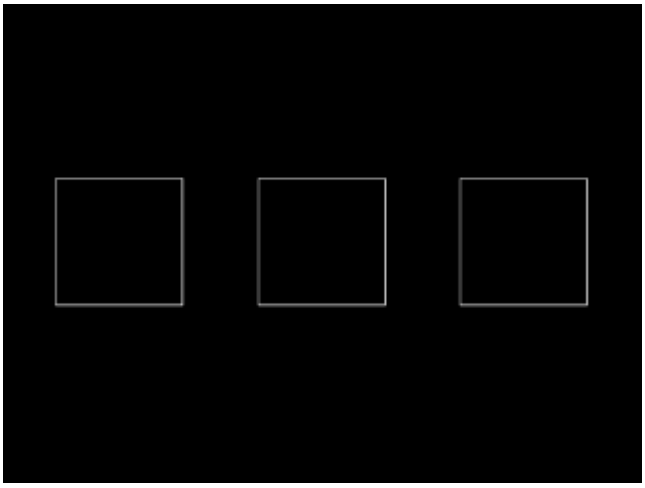
Options for reward and punishment are set in the General Parameters section; visual objects are defined in the Visual Object Library.

Screenshots from the task

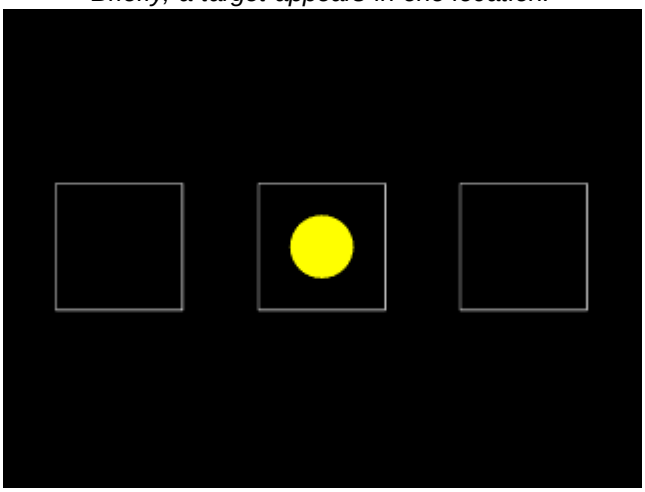
Here's an illustration of a 3-choice task, using a touchscreen centring response. The subject has to hold on to the triangular stimulus...



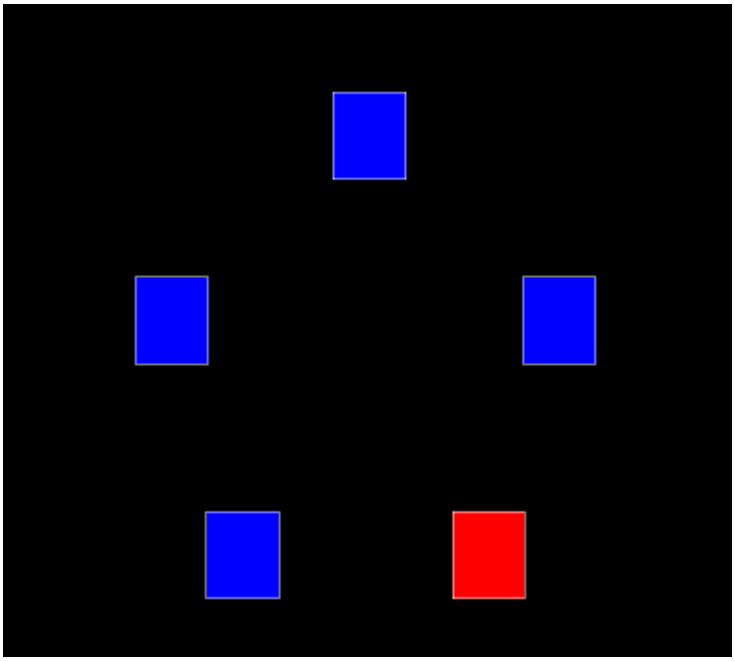
... after which it monitors the three locations.



Briefly, a target appears in one location.



Here's a screenshot of a five-choice task (with different target stimuli).



1.8.9 Paired-Associates Learning

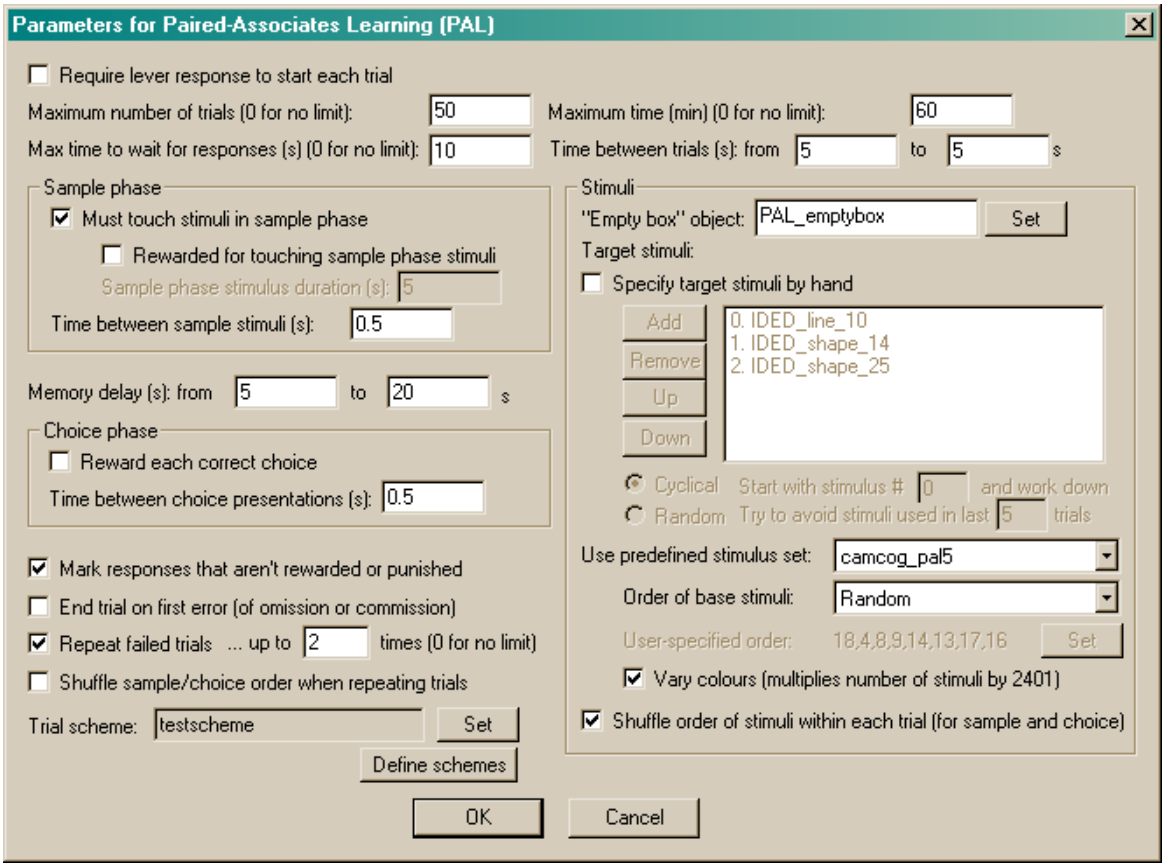
About the task

The basic principle of a paired-associates learning task is to associate multiple pairs of stimuli, and test recall. An example from human games is the Klimt Memo game; you have a deck of 72 cards with 36 images from paintings by Gustav Klimt. You scatter the cards face-down on a table. Each player turns over cards two at a time, trying to remember where each image is; if you turn over two identical cards, you remove the pair from the table; otherwise, you replace them in the face-down position. The objective is to turn over as many matching pairs as possible. The basic association being tested is a {stimulus, location} pair, and you must remember many of these to do well in the game.

Both the human and monkey versions of CANTAB use a stimulus-location pair. One might call the test "delayed matching-sample-to-location". In human CANTAB, subjects are first shown up to 8 stimuli in different locations around the edge of the screen. In the test phase, they are then shown a series of single stimuli in the centre of the screen and asked to indicate the location in which each stimulus was presented in the first phase. In monkey CANTAB, only 4 locations are used and the exemplar is not in the centre of the screen; instead, the exemplar is shown in every possible location and the monkey must choose the one location in which that stimulus was previously shown. For the task to be described as PAL, >1 stimulus must be used (otherwise it's just delayed matching to location).

The sample stage begins with the Marker 1 sound; thereafter, individual stimuli are presented. The subject may be required to touch them. After all the stimuli have been sampled, there is a "memory delay". The choice phase begins with the Marker 2 sound. The subject will be offered multiple choices, one for each stimulus that was seen in the sample phase. Each time, the subject must touch the location in which that stimulus appeared in the sample phase.

Configuring the task



- **Require lever response to start each trial.** Requires that the subject make a lever response to initiate each trial. See Use with Dogs.
- **Maximum number of trials.** When the subject has performed this number of trials, the task ends. (You may specify 0 for no limit, though you must specify a limit on the number of trials, the time, or both.)
- **Maximum time.** When this time elapses, the task is terminated as soon as the current trial has finished. (You may specify 0 for no limit, though you must specify a limit on the number of trials, the time, or both.)
- **Maximum time to wait for a response.** If the subject fails make a response within this time, the subject fails the trial. (This time limit applies to the sample phase only if you require your subjects to touch the sample stimulus; it always applies to the choice phase.)
- **Time between trials.** Specify a minimum and a maximum intertrial time (they may be the same). The actual time is chosen with a rectangular probability distribution within these values. The time between trials starts **after** any reward or punishment from the previous trial has finished.
- **Must touch stimuli in sample phase.** If this is selected, then the subject must respond to each sample stimulus. If you choose this option, you may also choose whether or not the subject should be **rewarded for touching sample stimuli.** If you do not want your subject to have to touch the stimulus, you must specify the **Sample stimulus duration** instead.
- **Time between sample stimuli.** This is the time between consecutive sample stimuli.
- **Memory delay.** This is the time between the end of the last sample stimulus *and any associated reward (e.g. the time it takes to operate the pellet dispenser and play a "reward" sound)* and the first of the choices. Specify a minimum and a maximum time (they may be the same). The actual time is chosen with a rectangular probability distribution within these values.
- **Reward each correct choice.** If this is selected, every time the subject makes a correct choice (in the choice phase, obvoiusly), it gets reward. If this option is not selected, subjects only get rewarded at the end of the trial if they have not made any choice errors. Regardless of this option, any time the subject makes an error, it is punished (as specified in the General Parameters).

- **Time between choice presentations.** This is the time between consecutive choice presentations *(not including any time taken to deliver reward, if you have chosen to do this).*
- **Mark responses that aren't rewarded or punished.** If this is chosen, then any responses that are neither rewarded nor punished are marked with the Marker 3 sound (as specified in the General Parameters).
- **End trial on first error.** If this is selected, then whenever the subject makes a mistake, the trial ends.
- **Repeat failed trials.** If this is selected, then if the subject fails to get all the choice presentations correct (or if you have "End trial on first error" selected and it gets something wrong in the sample phase) then the trial will be repeated. Set the **number of times a trial may be repeated,** too. Optionally, the order in which samples and choices are presented can be shuffled for repeated trials (while holding the stimulus-location pairs constant); to do this, tick **Shuffle sample/choice order when repeating trials.**
- **Trial scheme.** Trial schemes define the number and type of trials you will use. Click **Set** to choose a trial scheme. Click **Define** to define trial schemes. See below for details.

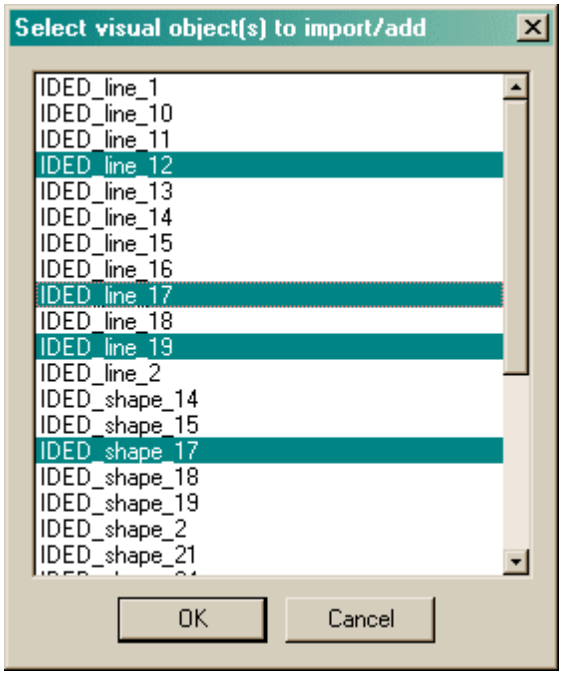
The task ends if EITHER (a) the total maximum number of trials is reached, or (b) the trial scheme has been completed and has run out of further trials to offer.

Stimuli

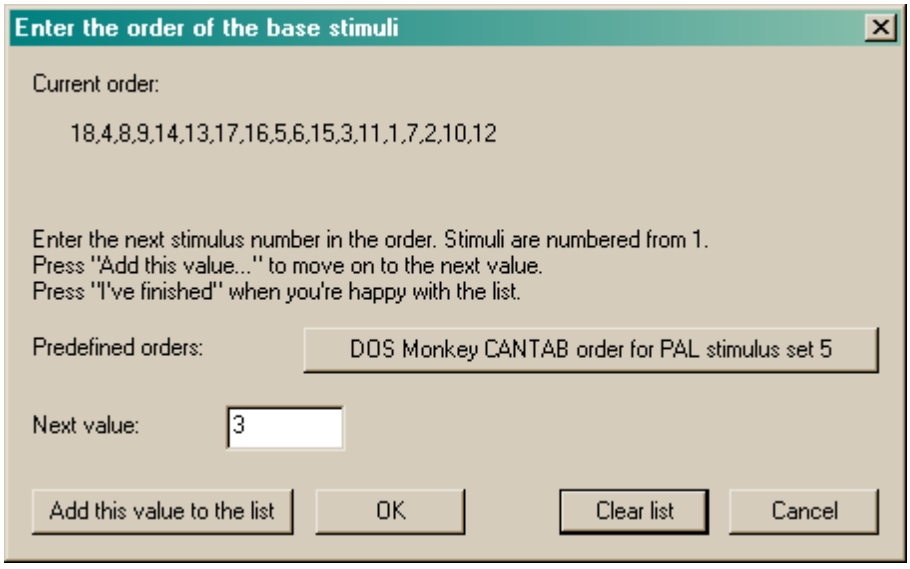
- **"Empty box" stimulus.** Choose the stimulus to be used as an "empty box" marker (for more details, see "Block Specification", below).
- **Specify stimuli by hand.** If you choose to specify stimuli by hand, the list shows the available stimuli. You cannot put a stimulus into the list more than once. Click **Add** and **Remove** to add/remove stimuli. You can also choose one of two methods for choosing the stimuli for each trial:
  - **Cyclical.** The program begins with a specified **stimulus number** and selects stimuli for each trial by working down the list, resuming at the start of the list if/when it runs out of stimuli to use at the bottom of the list.
  - **Random.** The program picks a set of stimuli to use at random on each trial. You will need to fill in **"Try to avoid stimuli used in last X trials";** the program will try not to choose any stimuli that have appeared (or were scheduled to appear, in the case of failure at phase 1) in the last X trials.

  It's your responsibility to ensure that enough stimuli are in the list! The program will complain if you try to start it and there aren't enough stimuli to provide a unique stimulus in each location for every trial in your scheme. It won't complain if it needs to re-use stimuli from trial to trial.

  Incidentally, when you **Add** stimuli, you can choose several at once by holding down the Shift key as you click on stimuli:



- **Predefined stimuli.** You may also use one of the predefined stimulus sets.
  - You may choose the **order of the base stimuli within the set** - either random, ascending, descending, or user-specified. If you choose the "user-specified" order, you may click the **Set** button to specify the order:



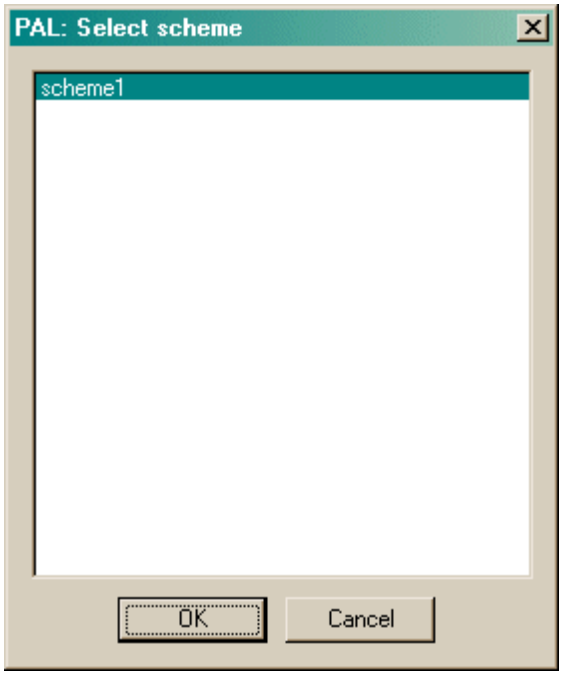
  - If you choose a predefined stimulus set, you also have the option of **varying the colours** of the predefined stimuli.

The program runs through all stimuli (in the order you chose), beginning with the base stimulus set (no variation - "variant 0"). If you choose to use colour variants, the program then runs through all the stimuli in the same order as the first run, with variant 1, and then again with variant 2, and so on. The program does *not* store the subject's position for next time. (I believe this is how Cambridge Cognition's DOS version of MonkeyCantab worked.) "Variants" are modifications of the base stimulus by altering the colours of each quadrant. There are 7^4 = 2401 variants of each stimulus.

- **Shuffle order of stimuli within each trial.** Having picked the stimuli for a given trial from your list of stimuli, or from the predefined set, you then have the option of shuffling the order of presentation of these, in the sample and choice phases. If you want an absolutely pre-specified sequence of stimuli, you would not want this ticked.

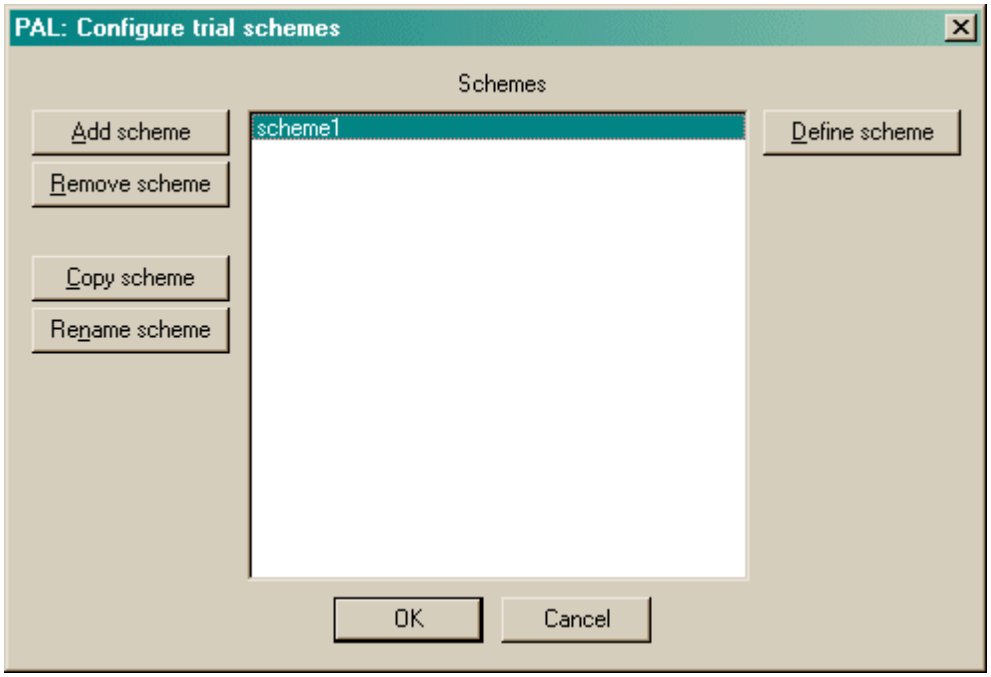
Set trial scheme

When you click this button, you can choose from a list of defined trial schemes.



Trial schemes

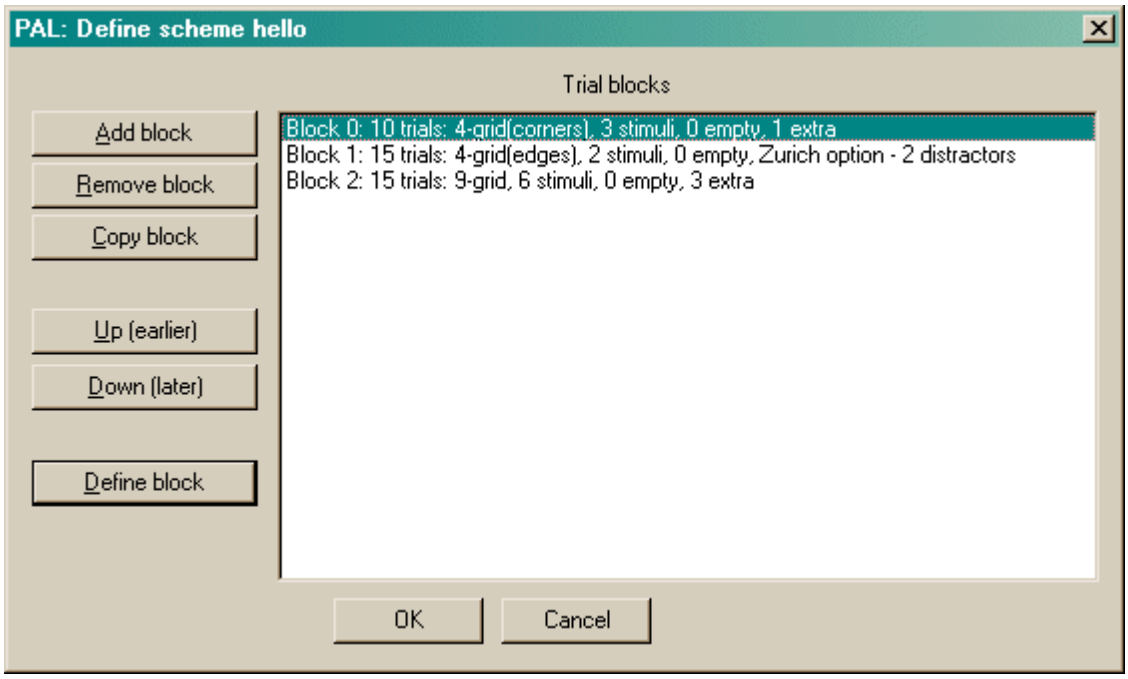
When you click **Define schemes** from the main parameters dialogue box, you can see the schemes. You may add, remove, copy, and rename schemes. Click a scheme and then click **Define scheme** to edit one particular scheme.



If a scheme is mis-configured, a warning message will appear when you click on the scheme.

Define scheme

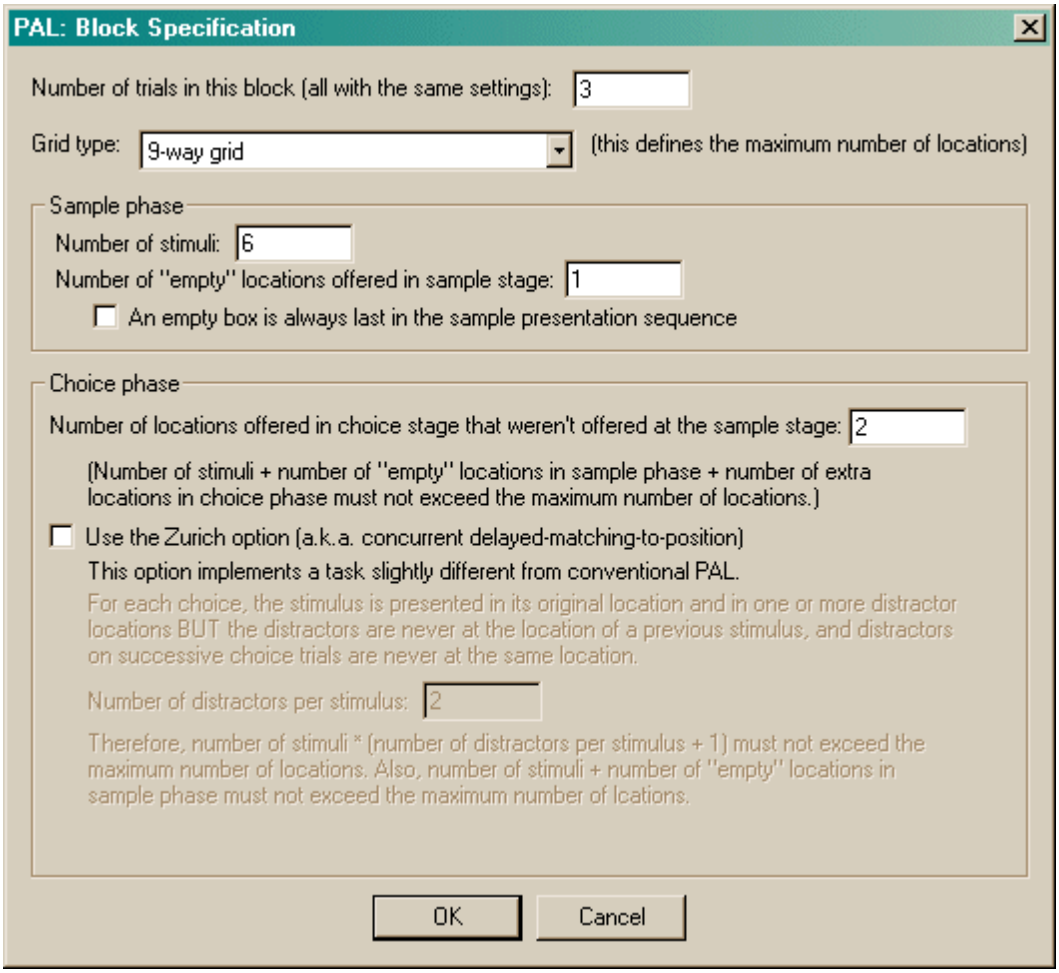
When you click this button, you can edit an individual scheme. Schemes consist of **blocks** of trials. Here, there are two blocks. You may add, remove, copy, and re-order blocks. Click **Define block** to edit one particular block.



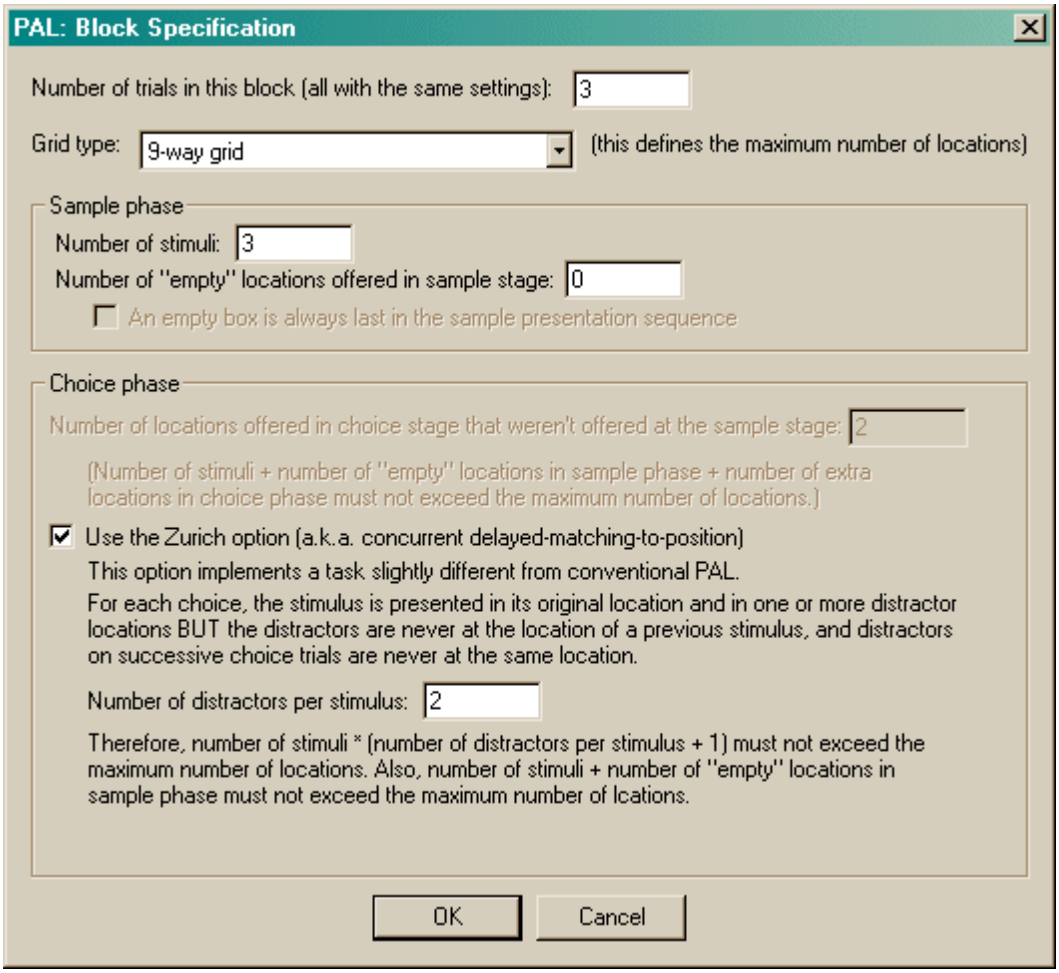
If a block is mis-configured, a warning message will appear when you click on the block.

Block specification

Here's where you can edit the block.



If you tick the "Zurich option", the options change slightly:



- **Number of trials.** Set the number of trials in this block. All these trials will run with the same settings (though individual stimuli and locations will be chosen at random for each trial within the block).
- **Grid type.** Choose from a variety of grid patterns; this setting determines the number of available locations (e.g. 4 or 9). The three numbers that follow must not add up to more than the number of available locations!

For the sample phase:

- **Number of stimuli.** The number of stimuli presented in the sample and choice phases.
- **Number of "empty" locations offered in sample stage.** You may show the "empty box" object at a number of locations in the sample phase, if you wish. This can be useful to enforce responding to particular areas of the screen. Set this number here.
  - **An empty box is always last in the sample presentation sequence.** This option allows you to enforce the rule that one of the empty boxes will always be shown last in the sample phase.

For the choice phase:

- **Number of locations offered in the choice stage that weren't offered at the sample stage.** In the choice phase, the default task will always use all the locations in which sample stimuli were presented (that's obvious), and it'll also use the locations at which you displayed the "empty box" in the sample phase, if any. You can also offer locations in the choice phase that were never offered in the sample phase. Choose that number of extra locations here.
- **Zurich option.** If you select this option, designed for a research group in the Swiss Federal Institute of Technology in Zurich, a rather different task is used. In the choice phase, a sample stimulus is shown in its correct location (as always), but instead of distractors appearing in all

locations where other samples were presented, distractors only appear in locations where no stimulus was presented in the sample phase. Furthermore, the distractors for one stimulus don't overlap with the distractors for a different stimulus. Imagine you are using the four-way grid, and have 2 stimuli shown in the sample phase. Suppose stimulus 1 appears in location A, and stimulus 2 appears in location B. Then the distractor location for stimulus 1 can only be C or D. If it is C, then the distractor location for stimulus 2 can only be D. That is, there is no overlap of sample or distractor locations for stimuli 1 and 2. So the only thing you need to choose is the **number of distractors that accompany each stimulus.**

For the conventional task,

number of sample stimuli + number of "empty" locations in sample phase + number of "extra" locations in choice phase

must not exceed the total number of locations in the grid. For the Zurich task, neither

number of sample stimuli + number of "empty" locations in sample phase

nor

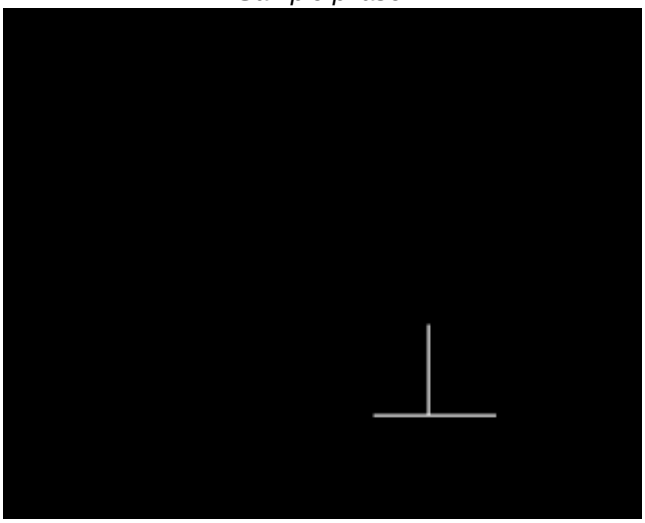
*number of sample stimuli * (number of distractors per stimulus + 1)*

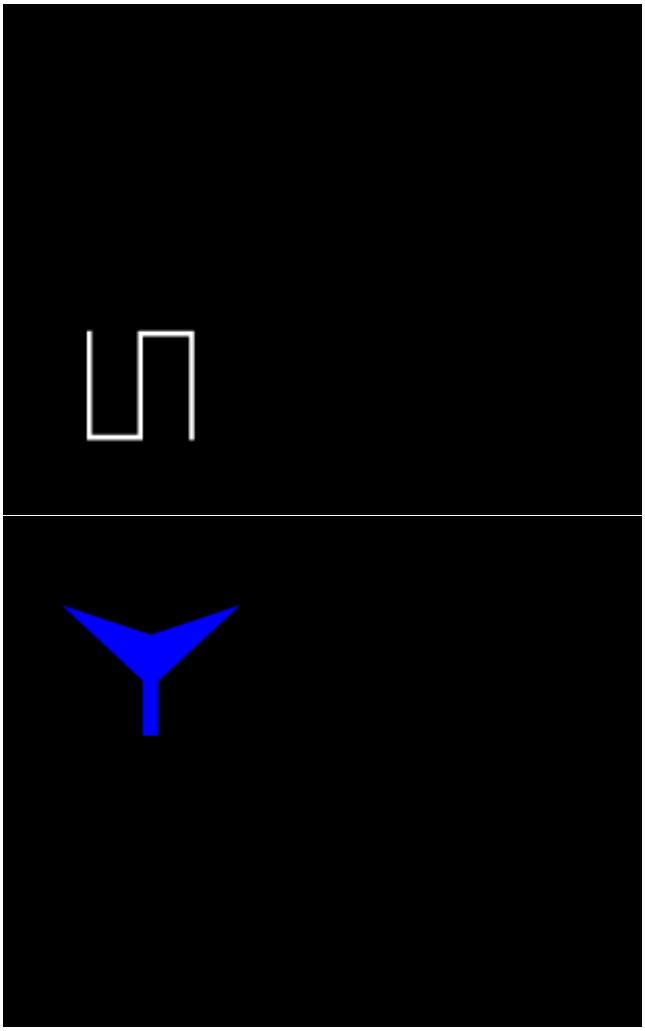
can exceed the total number of locations in the grid. Otherwise, a warning will appear showing that your scheme/block is mis-configured.

Screenshots from the task

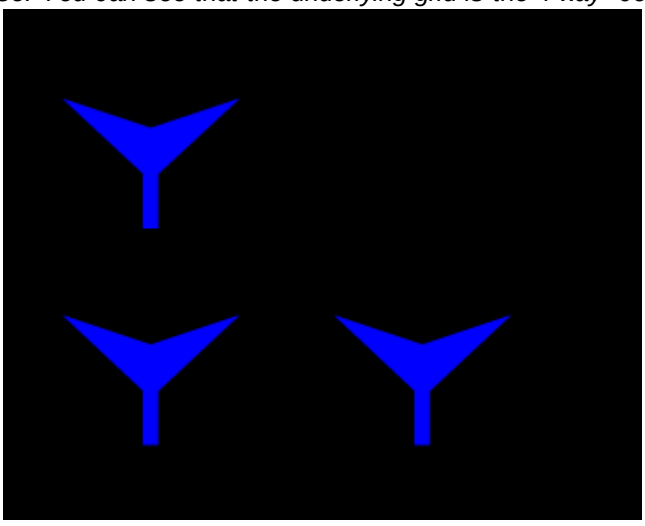
This is a very simple example, with the conventional PAL task, three stimuli, no "empty boxes" shown in the sample phase, and no extra locations offered in the choice phase.

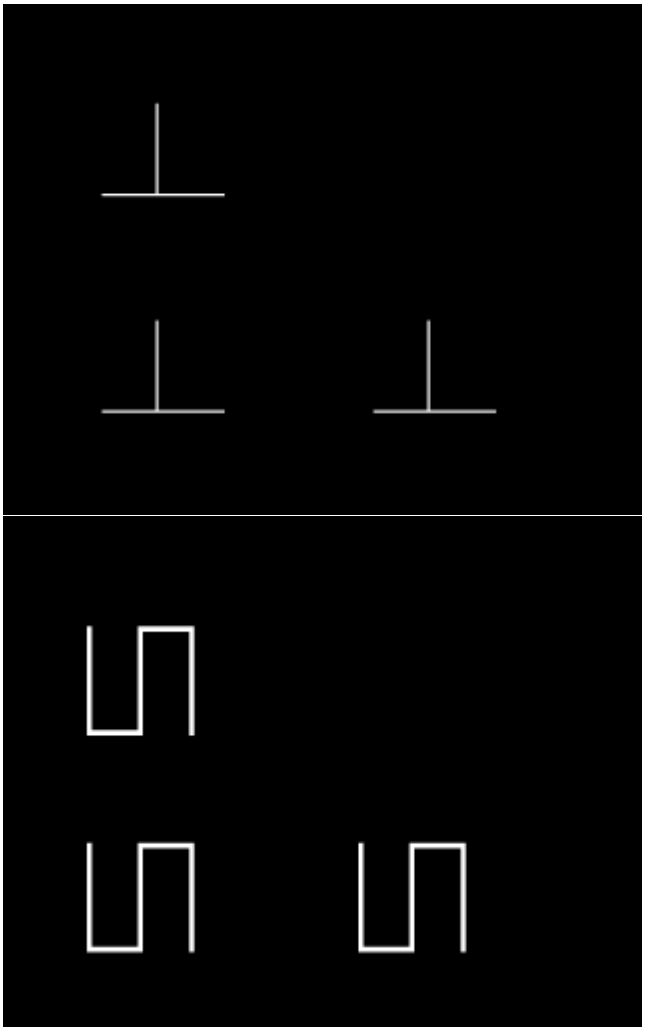
Sample phase



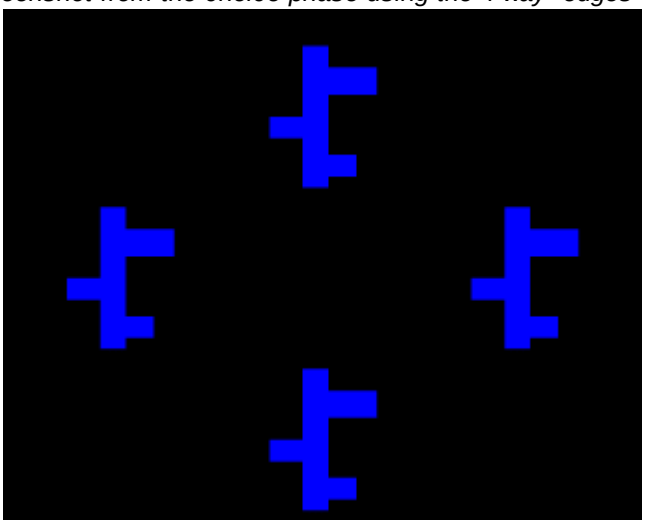


Choice phase. You can see that the underlying grid is the 4-way "corners" grid.

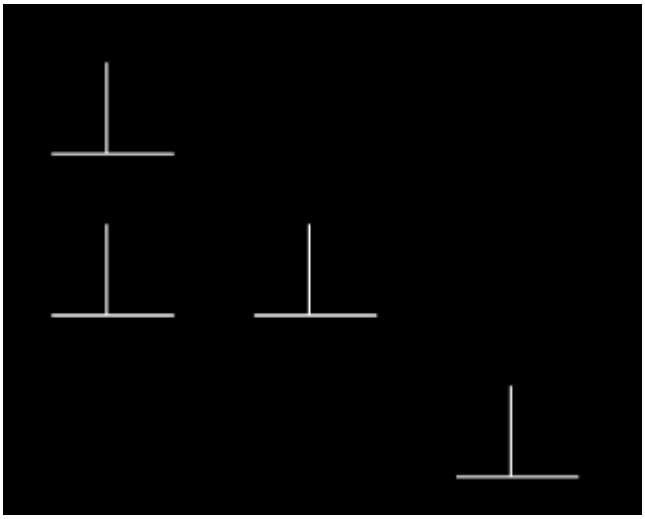




A screenshot from the choice phase using the 4-way "edges" grid.



A screenshot from the choice phase using four locations of the 9-way grid.



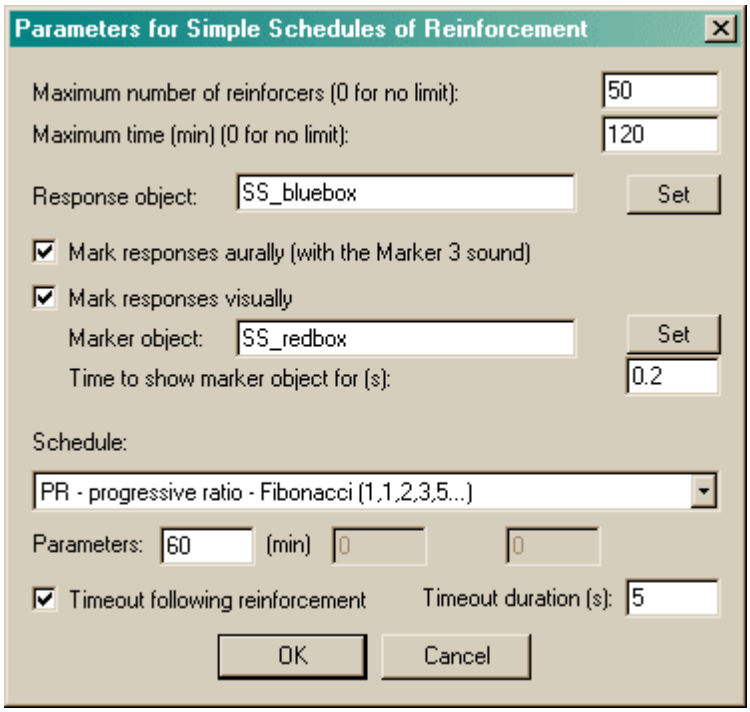
1.8.10 Simple Schedules of Reinforcement

About the task

Schedules of reinforcement using a touchscreen manipulandum.

Configuring the task

What constitutes "a reinforcer" is defined in the General Parameters. This allows you to set the reward sound and define what sound is associated with the presentation of a reward.



- **Maximum number of reinforcers.** Once this number of rewards has been delivered, the task stops. (Specify 0 for no limit.) You must specify either a maximum number of rewards, or a maximum time, or both.
- **Maximum time (min).** Once this time limit has been reached, the task stops. (Specify 0 for no limit.) You must specify either a maximum number of rewards, or a maximum time, or both.

- **Response object.** The picture that the subject has to touch. Click **Set** to choose an object from the visual object library.
- **Mark resposes aurally.** If this is selected, the Marker 3 sound will be played to inform the subject that it has touched the picture successfully. (The schedule pauses while this sound is played.)
- **Mark responses visually.** If this is selected, you may replace the response object with another picture for a brief period of time, to indicate visually that the subject has made a successful response. The marker object is shown here; click **Set** to choose one from the visual object library and choose the **Time to show marker object for (s).** The schedule is paused while the marker object is being shown. The marker object may also be maintained for as long as the subject's finger remains on the screen (i.e. they have to let go to get the manipulandum back).

- **Schedule.** Choose the schedule, and up to three **parameters** associated with the schedule. The schedules are:

  - **CRF - continuous reinforcement (FR-1).** One reinforcer per response.
  - **EXT - extinction.** No reinforcers.
  - **FR *x* - fixed ratio.** One reinforcer per *x* responses.
  - **VR *x* to *y* - variable ratio (specifying min, max).** After a variable number of responses (randomly chosen from *min* to *max* inclusive), one reinforcer is delivered.
  - **RR *x* - random ratio.** P(reinforcer | response) = 1/*x*.
  - **PROB *p* - probabilistic.** P(reinforcer | response) = *p*.
  - **FI *x* - fixed interval.** The first response after *x* seconds is reinforced. The *first response* of the schedule is also reinforced.
  - **RI *x* - random interval.** Reinforcement is set up on a random-time schedule (see below); after reinforcement has been set up, the next response is reinforced.
  - **VI *x* to *y* - variable interval (specifying min, max).** After a variable time (from *min* to *max* seconds), the next response is reinforced.
  - **FT *x* - fixed time (NONCONTINGENT).** No lever is present. Reinforcement is delivered every *x* seconds.
  - **VT *x* to *y* - variable time (specifying min, max) (NONCONTINGENT).** No lever is present. The schedule waits for between *min* and *max* seconds, then delivers a reinforcer, then repeats.
  - **RT *x* - random time (NONCONTINGENT).** Every second, p(reinforcer delivered this second) = 1/*x*. Thus, on average, reinforcement is delivered once every *x* seconds, but the subject cannot predict the likelihood of reinforcement based on how long it has waited (unlike a typical VT schedule).
  - **PR - progressive ratio - add one (1,2,3,4...)** - progressive ratio schedule, adding one to the ratio requirement at each step. The schedule termination is determined by the parameter; if *parameter* is >0, then when *parameter* minutes have elapsed since the last reinforcer (or response - see below), the schedule stops. We suggest 60 as a sensible value.
  - **PR - progressive ratio - double (1,2,4,8...) -** progressive ratio schedule, doubling the ratio requirement at each step. The schedule termination is determined by the parameter; if *parameter* is >0, then when *parameter* minutes have elapsed since the last reinforcer (or response - see below), the schedule stops. We suggest 60 as a sensible value.
  - **PR - progressive ratio - Fibonacci (1,1,2,3,5...) -** progressive ratio schedule with a Fibonacci progression. The schedule termination is determined by the parameter; if *parameter* is >0, then when *parameter* minutes have elapsed since the last reinforcer (or response - see below), the schedule stops. We suggest 60 as a sensible value.
  - **PR - progressive ratio - Roberts exponential (A * exp(reinfnum * B) - A) -** progressive ratio schedule with an exponential progression, based on Roberts DCS & Richardson NR (1992), Self-administration of psychomotor stimulants using progressive ratio schedules of reinforcement, *Neuromethods* 24: 233-269 (eds Boulton A, Baker G, Wu PH; Humana Press). The ratio requirement is (A * exp(reinforcer number * B)) - A, rounded to the nearest integer. Typically, A is 5. A typical schedule might have B=0.2; these values yield ratio

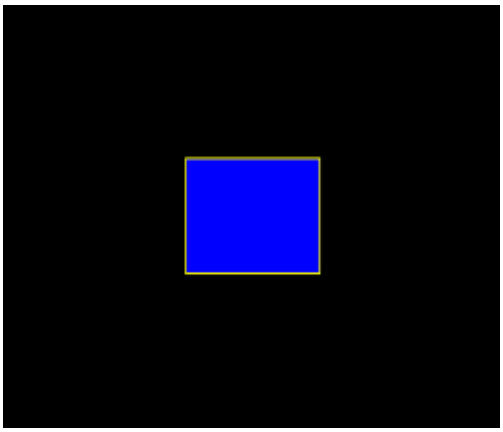
requirements {1, 2, 4, 6, 9, 12, 15, 20, 25, 32, 40, 50, 62, 77, 95, 118, 145, 178, 219, 268, 328, 402, 492, 603, 737, 901, 1102, 1347, ...}. A steeper PR schedule is obtained with B=0.25, giving {1, 3, 6, 9, 12, 17, 24, 32, 42, 56, 73, 95, 124, 161, 208, 268, 346, 445, 573, 737, 948, 1218, 1566, 2012, 2585, 3321, 4265, 5478, ...} The schedule termination is determined by the other parameter (on the left, labelled (min)); if this parameter is >0, then when this many minutes have elapsed since the last reinforcer (or response - see below), the schedule stops. We suggest 60 as a sensible value.

- **DELAYFR1 - FR1 with delayed reinforcement.** This is an FR1 schedule, but there is a delay between responding and reinforcement. This delay is the sole parameter (specified in seconds).
- **PR - progressive ratio - double increment every A reinforcers.** The increment starts at 1, and doubles every A reinforcers. If A is 8, then the ratio requirements are 1, 2, 3, 4, 5, 6, 7, 8, 10, 12, 14, 16, 18, 20, 22, 24, 28, 32, 36... The schedule termination is determined by the parameter; if *parameter* is >0, then when *parameter* minutes have elapsed since the last reinforcer (or response - see below), the schedule stops. We suggest 60 as a sensible value.

Special case: the first response on contingent interval schedules (FI, RI, VI) is always reinforced. For noncontingent schedules, nothing is shown on the display.
See also Notes on reinforcement timing.

For **progressive ratio schedules,** you may choose whether the timeout that eventually terminates the schedule, if selected, is calculated from the last response or the last reinforcer.

Screenshot from the task



1.8.10.1 Notes on reinforcement timing

It is possible to cause reinforcement conflicts in this task. Most particularly, if you use a delayed-reinforcement schedule, it is possible that reinforcement is scheduled while the reinforcing device is busy. **SimpleSchedules *ignores* requests for reinforcement for devices that are busy.** It records in the event log (text-based and ODBC) whether a scheduled reinforcement was *actually* given.

What happens when you request a timeout on a delayed-reinforcement schedule? Should the timeout occur at the time of the response, or at the time of the reinforcement? Well, a timeout is to stop you responding, so it should occur at the time you respond. This is what SimpleSchedules does.

What happens when your schedule wants to reinforce, and to give you a timeout, but your reinforcement device is busy? SimpleSchedules will not reinforce (because the device is busy

reinforcing you anyway) but it *will* implement your timeout.

As of 4 May 2005, the manipulandum is not visible unless it could be reinforced (i.e. during timeouts and when the reinforcing device is busy, the manipulandum is not shown - except for delayed reinforcement schedules, where the current state of the reinforcing device is immaterial).

1.8.11 Impulsive Choice

Purpose

Choice with delayed and/or probabilistic reinforcement (discrete-trial task).

Originally based on Evenden JL, Ryan CN (1996). The pharmacology of impulsive behaviour in rats: the effects of drugs on response choice with varying delays of reinforcement. Psychopharmacology 128: 161–170.

Sample publications using this form of task (PMID refers to PubMed ID at http://www.pubmed.com)

- Cardinal RN, Robbins TW, Everitt BJ (2000). The effects of d-amphetamine, chlordiazepoxide, alpha-flupenthixol and behavioural manipulations on choice of signalled and unsignalled delayed reinforcement in rats. Psychopharmacology 152: 362–375. PMID 11140328.
- Cardinal RN, Pennicott DR, Sugathapala CL, Robbins TW, Everitt BJ (2001). Impulsive choice induced in rats by lesions of the nucleus accumbens core. Science 292: 2499–2501. PMID 11375482.
- Cardinal RN, Cheung THC (2005). Nucleus accumbens core lesions retard instrumental learning and performance with delayed reinforcement in the rat. BMC Neuroscience 6: 9. PMID 15691387.
- Cheung THC, Cardinal RN (2005). Hippocampal lesions facilitate instrumental learning with delayed reinforcement but induce impulsive choice in rats. BMC Neuroscience 6: 36. PMID 15892889.
- Cardinal RN, Howes NJ (2005). Effects of lesions of the nucleus accumbens core on choice between small certain rewards and large uncertain rewards in rats. BMC Neuroscience 6: 37. PMID 15921529.

The following articles illustrate a quantitative methodology (to be commended) to establish aspects of reinforcer delay/magnitude/probability sensitivity; this paradigm can also be accomplished with the present task.

- Kheramin S, Body S, Mobini S, Ho MY, Velázquez-Martinez DN, Bradshaw CM, Szabadi E, Deakin JF, Anderson IM (2004). Effects of quinolinic acid-induced lesions of the orbital prefrontal cortex on inter-temporal choice: a quantitative analysis. Psychopharmacology 165: 9-17. PMID 12474113.
- Kheramin S, Body S, Ho M, Velazquez-Martinez DN, Bradshaw CM, Szabadi E, Deakin JF, Anderson IM (2003). Role of the orbital prefrontal cortex in choice between delayed and uncertain reinforcers: a quantitative analysis. Behavioural Processes 64: 239-250. PMID 14580695.
- Kheramin S, Body S, Ho MY, Velazquez-Martinez DN, Bradshaw CM, Szabadi E, Deakin JF, Anderson IM (2004). Effects of orbital prefrontal cortex dopamine depletion on inter-temporal choice: a quantitative analysis. Psychopharmacology 175: 206-14. PMID 14991223.
- Bezzina G, Cheung TH, Asgari K, Hampson CL, Body S, Bradshaw CM, Szabadi E, Deakin JF, Anderson IM (2007). Effects of quinolinic acid-induced lesions of the nucleus accumbens core on inter-temporal choice: a quantitative analysis. Psychopharmacology 195: 71-84. PMID 17659381.

Touchscreen and operant chamber versions of this task

This program (part of MonkeyCantab) is a touchscreen-based task. A separate task (known simply as ImpulsiveChoice) exists to run this task in conventional lever-based operant chambers; see http://www.whiskercontrol.com.

Typical trial structure

A number of types of task are possible with this program. The trial structure of a typical delayed-reinforcement task, albeit using operant chambers rather than touchscreens, is shown below (taken from Cardinal et al. 2001):

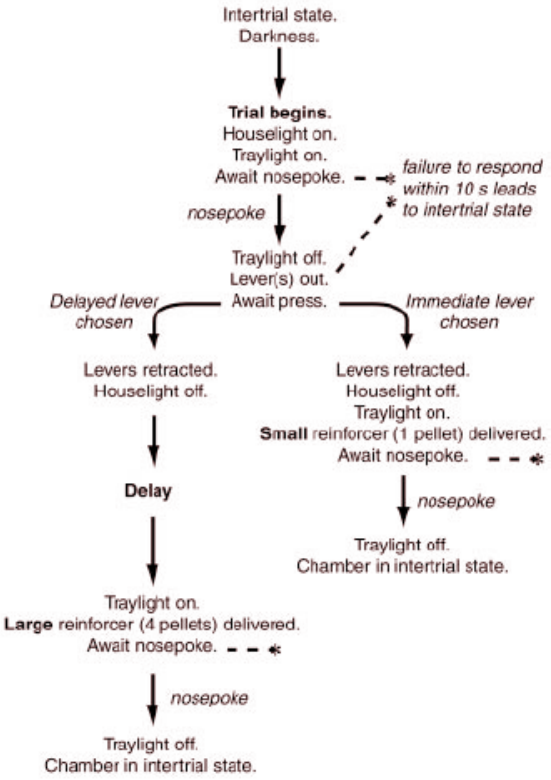
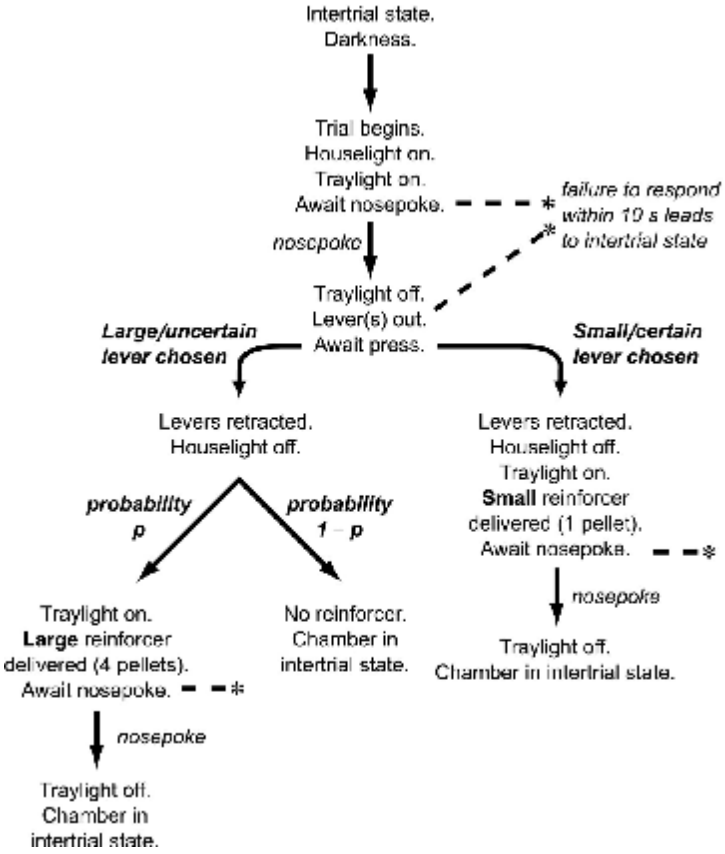


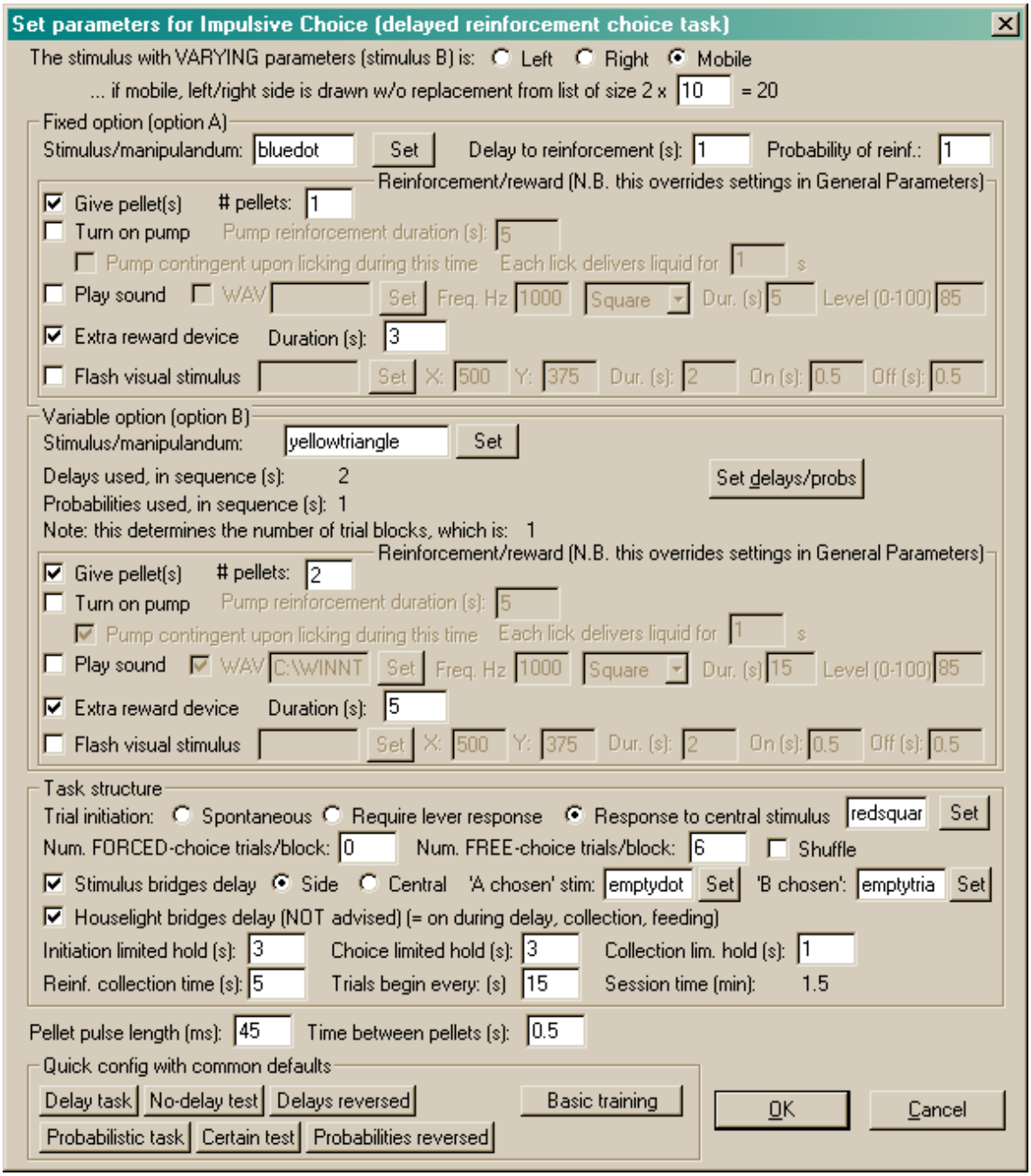
Fig. 1. Delayed reinforcement choice task. The format of a single trial is shown; trials began at 100-s intervals. A session lasted 100 min and consisted of five blocks, each comprising two trials in which only one lever was presented (one trial for each lever, in randomized order) followed by 10 choice trials. The delay to the large reinforcer was varied systematically across the session. Delays for each block were 0, 10, 20, 40, and 60 s, respectively.

Likewise, the trial structure of a typical probability-based task is shown below (taken from Cardinal & Howes 2005):



Parameters

The parameters dialogue box looks like this:



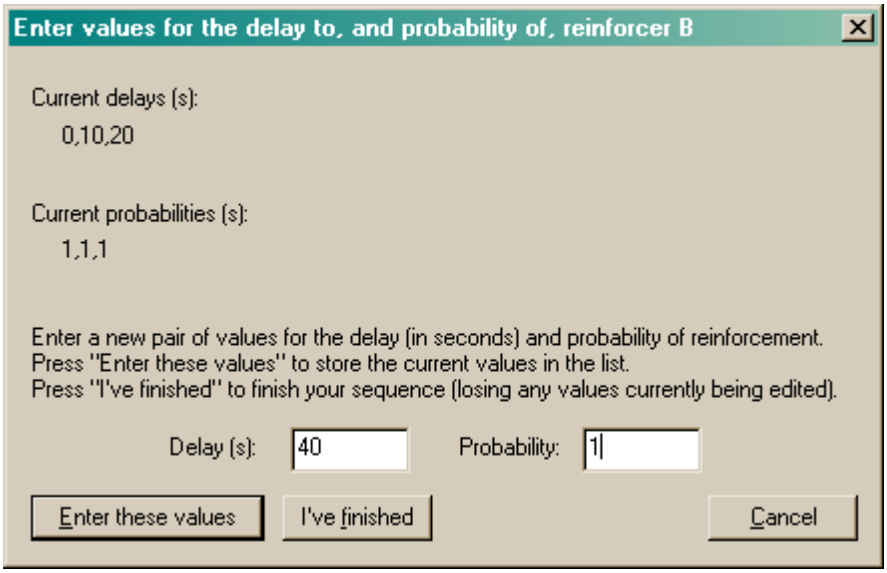
The options are as follows:

- **The stimulus with varying parameters (stimulus B) is left/right/mobile.** Whereas the usual operant chamber version of this task associates a lever's *side* (e.g. left/right) with a particular reinforcement outcome, this touchscreen task associates a visual *stimulus* with a particular outcome. Suppose a red circle is your chosen stimulus for option A (which might be a small, immediate reward) and a green square is your chosen stimulus for option B (which might be a large, variably delayed reward). Should the option B stimulus always be on the left, always on the right, or variably on the left and right?
  - **... if mobile, left/right side is drawn without replacement from list of size 2 x N.** If you choose the "mobile" option, then specify this option. You choose a number; we will call it N. When you run the task, N copies of "left" are put into a hat, and N copies of "right" are also put in. The contents of the hat are shuffled. For each trial in turn, an option is then drawn out at random (and not replaced); this option determines the side of option B. When the hat is empty,

the procedure is repeated. So if you specify N = 1, then each consecutive trial pair contains one "A on the right / B on the left" trial and one "A on the left / B on the right" trial. If you specify a very large N, the procedure becomes close to drawing truly at random. See Drawing Without Replacement for more details.
    - As a further complexity, forced-choice trials are, by default, arranged in A/B pairs (see below). The "B side" is kept constant across that pair. A new "B side" is picked for each *pair* as described above. In free-choice trials, a new B side is picked for each *trial* as described above.
- **Fixed option (option A).** This section specifies the parameters for option A, whose parameters do not vary across the session.
    - **Stimulus/manipulandum.** What stimulus will be shown for the subject to touch to obtain option A? **Set** the stimulus (see Choosing Stimuli For A Task).
    - **Delay to reinforcement (s).** Specify the delay to reinforcement for option A.
    - **Probability of reinforcement.** Specify the probability that reinforcement will be delivered, given that option A is chosen (from 0 to 1).
    - **Reinforcement/reward parameters.** These set the actual parameters of reinforcement when it is delivered. These parameters override the General Parameters, but the meaning of each setting is the same.
        - **Give pellets.** If selected, set the **number of pellets.**
        - **Turn on pump.** If selected, select the **pump reinforcement duration.** You can also make the **pump contingent upon licking** during the time the pump is "scheduled" to be on, in which case **each lick delivers liquid** (activates the pump) for **N** seconds; you specify N.
        - **Play sound.** If selected, a sound is played along with the reward; you can choose whether that sound comes from a **.WAV** file (in which case you can **set** the filename) or is of a specified **waveform/duration;** in either case, you can set the duration and sound level. See the General Parameters for more detail of sound parameters.
        - **Extra reward device.** If selected, you can also activate the Extra Reward Device (see Required Devices and General Parameters) for a specified **duration.**
        - **Visual reward stimulus.** If selected, you can display and flash a reward stimulus. Specify the stimulus **name** (by clicking the **Set** button), its **X** and **Y** coordinates, its overall **duration**, and its **on-flashing time** and **off-flashing time**, just as in the General Parameters.
- **Variable option (option B).**This section specifies the parameters for option B, whose parameters vary across the session.
    - **Stimulus/manipulandum.** As for option A.
    - **Delays and probabilities for option B.** Although you specify delays and probabilities as for option A, this time you specify a list of {delay, probability} pairs. Click the **Set delays/probs** button to do so. See below for details of the data entry process. ***One pair of values is used for each trial block.***
    - **Reinforcement/reward parameters.** As for option A.
- **Task structure.**
    - **Trial initiation: spontaneous/lever/stimulus.** Should trials initiate themselves spontaneously? Should a lever response be required to start the trial? Or should the subject have to touch a central stimulus (which you can **set**) to initiate each trial?
    - **Number of forced-choice trials per block.** Trials are grouped into blocks. Blocks typically start with a certain number of forced-choice trials; by this I mean that only one option (A or B) is offered in each trial. Specify an even number here; "A" trials and "B" trials are (by default) delivered in pairs, with the A/B order within each pair being random.
    - **Number of free-choice trials per block.** After the forced-choice trials, there are free-choice trials, where A and B are offered. Specify the number of free-choice trials in each block. ***Remember that the delay/probability of option B stays constant within a block, but varies across blocks.***
    - **Shuffle trials within a block?** Optionally, instead of having the forced-choice trials followed by the free-choice trials, you can shuffle (randomize) the order of all trials within a block.
    - **Stimulus bridges delay.** If selected, a stimulus will be displayed during the delay to the A and B reinforcers (assuming that the response is rewarded). Choose the stimuli to be displayed when A and B have been selected, and choose whether this stimulus should be displayed on

the side of the manipulandum (i.e. the A stimulus will be displayed on the A side, etc.) or in the centre of the screen.
  - **Houselight bridges delay.** Ordinarily in this task, the houselight is on when waiting for trial initiation, and when waiting for a choice, and off at all other times. If this is ticked, the houselight is left on during any delay, and during reinforcement delivery/collection/feeding.
  - **Initiation limited hold.** If the subject does not initiate the trial within this time, the trial is abandoned. (Not applicable for the "spontaneous" method of trial initiation; see above.)
  - **Choice limited hold.** If the subject does not choose A or B within this time, the trial is abandoned.
  - **Collection limited hold.** If the subject does not collect the reinforcer within this time, the task returns to the intertrial state. ***(Note that typical MonkeyCantab equipment cannot detect pellet collection, only licks at the pump; however, behaviourally it is unlikely to matter if the task returns to the intertrial state whilst pellets are being collected/eaten. Obviously, collection latency data will only be available for pump licking.)***
  - **Reinforcer collection time.** Once the reinforcer has been collected, the lights stay on for this length of time before the task returns to the intertrial state.
  - **Trials begin every...** (known as the trial *period*, in the physics sense). Other things being equal, trials should begin at fixed intervals in delay-of-reinforcement tasks so as to avoid the confound of a subject choosing a short-delay reinforcer often and thus doing better than a subject choosing a long-delay reinforcer. This task fixes the trial *period* (the interval at which trials begin) so as avoid this problem. You must specify a time that exceeds the maximum reinforcer delay, plus initiation/choice/etc. limited hold times.
  - The session time is shown, calculated from the information you have provided.
- **Pellet pulse length.** This option should be set to match your pellet dispenser; see the General Parameters.
- **Time between pellets.** This option should be set to match your pellet dispenser (and to make the arrival of multiple pellets distinctive to the subject); see the General Parameters.
- **Quick configuration options.** These buttons provide commonly used default settings for the assessment of delay or probability sensitivity; see the publications cited above for further discussion.

To specify a list of delays/probabilities for reinforcer B, click the "Set delays/probs" button:



Every time you click "Enter another pair", the current pair of values is added to the list (shown near the top) and you can enter another.

1.8.12 Rapid Visual Information Processing

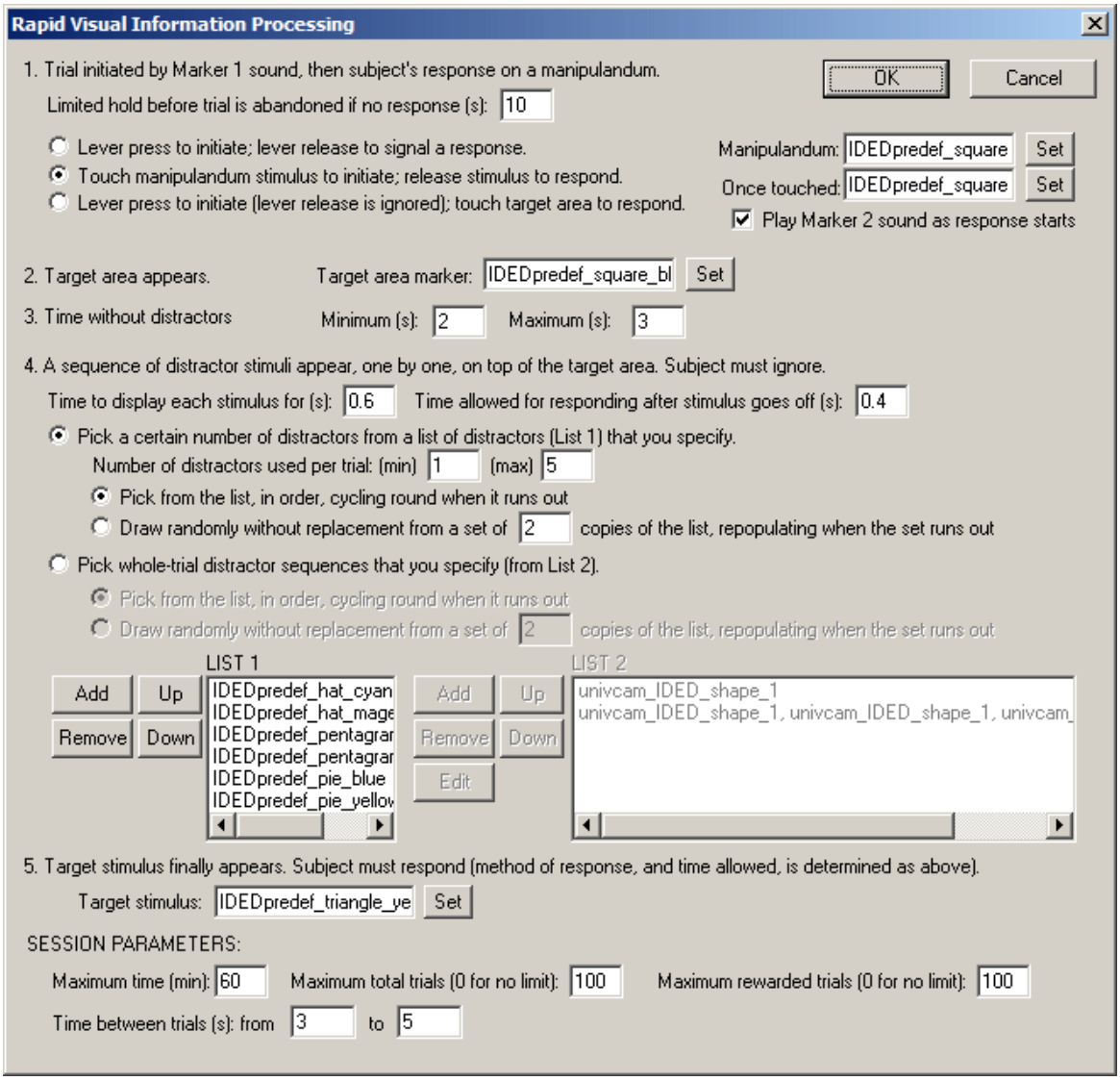
About the task

- Each trial begins with a Marker 1 sound.
- The subject may initiate a trial (and subsequently respond) by pressing (and then releasing) a lever, or by touching (and later releasing) a stimulus on the screen, or by pressing a lever to initiate the trial and then touching the target to respond.
- Following initiation, the target area appears on the screen.
- After a configurable delay, stimuli start to appear in the target area. The subject must watch for the target stimulus. Before it appears, there is usually a series of distractor stimuli that must be ignored.
- Success occurs when the subject responds to the target stimulus. Failure occurs when the subject fails to initiate the trial, fails to respond to the target, or responds early to one of the distractors.

Visual appearance

- The visual manipulandum, if used, appears in the centre bottom position of the nine-way grid.
- The target marker, distractors, and target appear in the centre top position of the nine-way grid.

Configuring the task



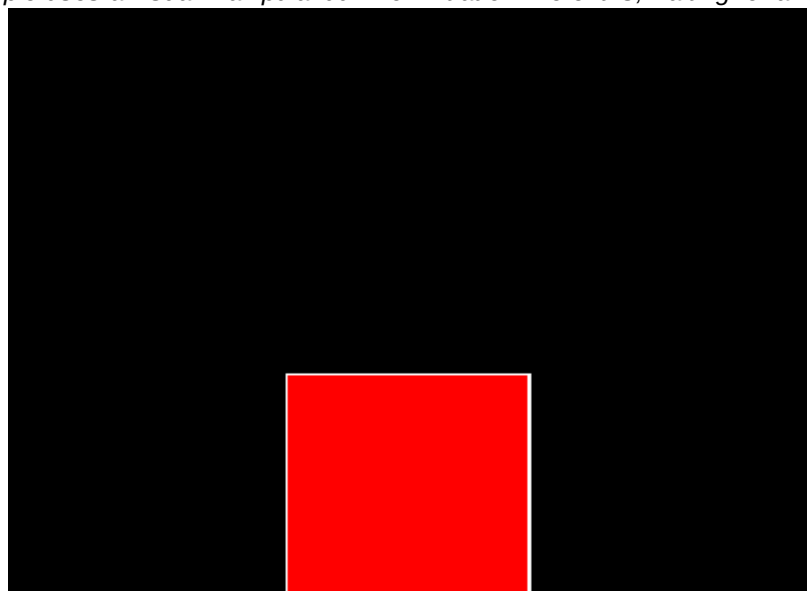
- **Limited hold before trial is abandoned if no response (s).** Specify the time from the trial start before the trial is abandoned if the subject does not initiate.
- **Method of trial initiation and response.** Options here are listed next. Additionally, the Marker 2 sound can be played when the subject initiates.
  - INITIATE with a lever press; RESPOND by releasing the lever. This option extends the lever (retracting it at the end of the trial) and does not involve a visual manipulandum on the touchscreen.
  - INITIATE by touching a visual manipulandum; RESPOND by releasing the same manipulandum. This option does not use the lever, but requires a visual manipulandum. Click the **Set** buttons to choose a visual stimulus for this (see Choosing Stimuli). The visual manipulandum changes once it is touched to the "manipulandum once touched" option (just supply the same stimulus in both boxes for nothing to change).
  - INITIATE by pressing a lever; RESPOND by touching the target area. This option does not require the visual manipulandum. The lever stays out for the whole trial (though responses on it after initiation are ignored).
- **Target area marker.** Once the trial is initiated, the target area marker appears, and remains for the rest of the trial.
- **Time without distractors.** After initiation, there is a pause before distractors start to appear. Specify the minimum and maximum pause. The program picks a pause length at random within that range.

- **Time to display each stimulus.** Each stimulus is displayed for a certain time, then goes off for a certain time (during which responses are still counted). Specify these times.
- **Sequence of distractors.** There are two options.
  - LIST 1. In this method, you provide a list (List 1) of individual distractor stimuli, and tell the program (as a range) how many distractors to use per trial. The program either uses them in sequence or pseudorandomly as it needs distractors.
  - LIST 2. In this method, you specify a list (List 2) of SEQUENCES of distractor stimuli, one for each trial. In this method, the program may use your *sequences* sequentially or pseudorandomly, but it does not interfere with the sequence you specify within a given trial. The number of distractors on each trial is determined by the length of your sequence. Difficult tasks may be created this way (e.g. if your target is a yellow triangle, you could specify a list of distractors of green circle, green circle, green circle, green circle, yellow circle; this last distractor may cause premature responding as it shares its colour with the target).
  - Whichever method you use, you can have the program pick stimuli (List 1) or trials (List 2) from your list sequentially, cycling round to the start when the list runs out. Alternatively, you can put *N* copies of the relevant list into a big hat, shuffle everything in the hat, and then pick stimuli (List 1) or trials (List 2) from the hat until the hat is empty, before repeating the whole process. This process is called drawing without replacement, and the number *N* is something you can specify in the dialogue box ("... a set of *N* copies of the list...").
- **Target stimulus.** Specify the eventual target stimulus. It is displayed for the same time as the other stimuli (see above).
- **Maximum time / maximum total trials / maximum rewarded trials.** I hope these are obvious. There are further limits available on rewards in the General Parameters.
- **Time between trials.** Specify the minimum and maximum intertrial time.

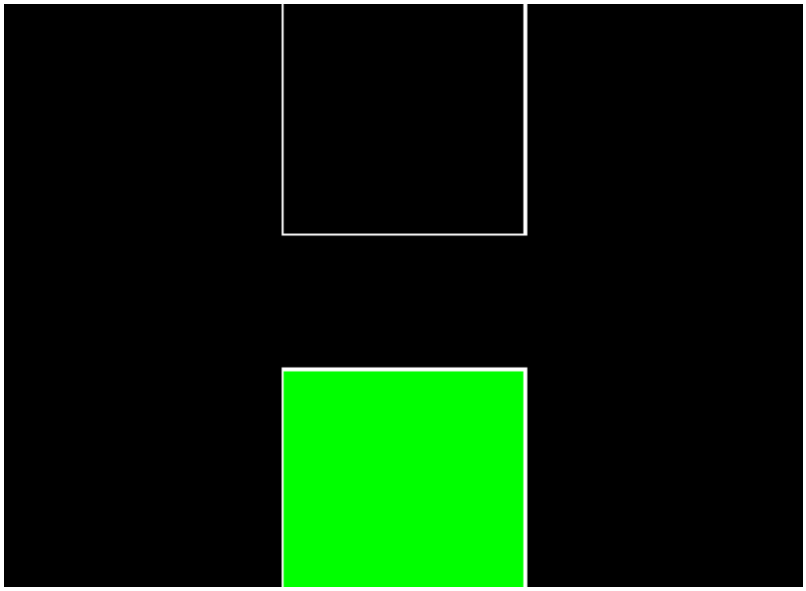
Screenshots

These screenshots don't use ideal stimuli! Anyway, the gives a rough idea.

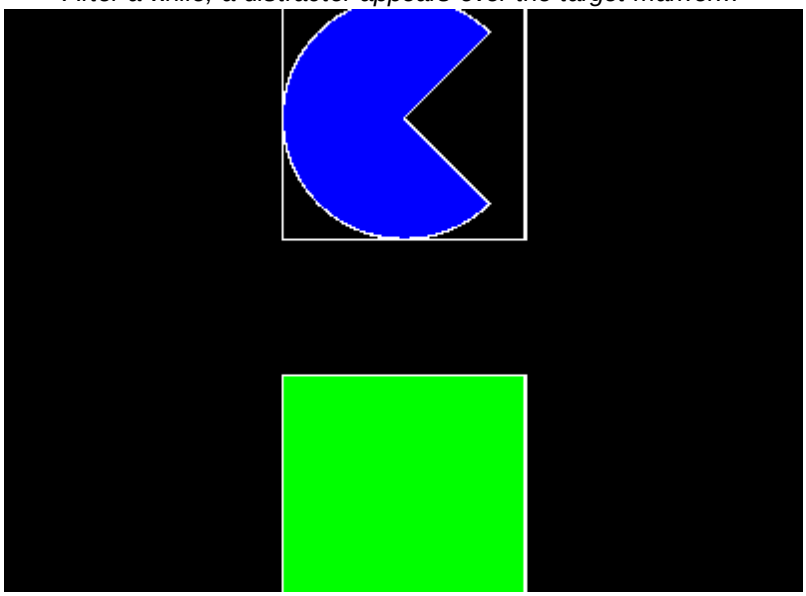
This example uses a visual manipulandum for initiation. Here it is, waiting for a response...



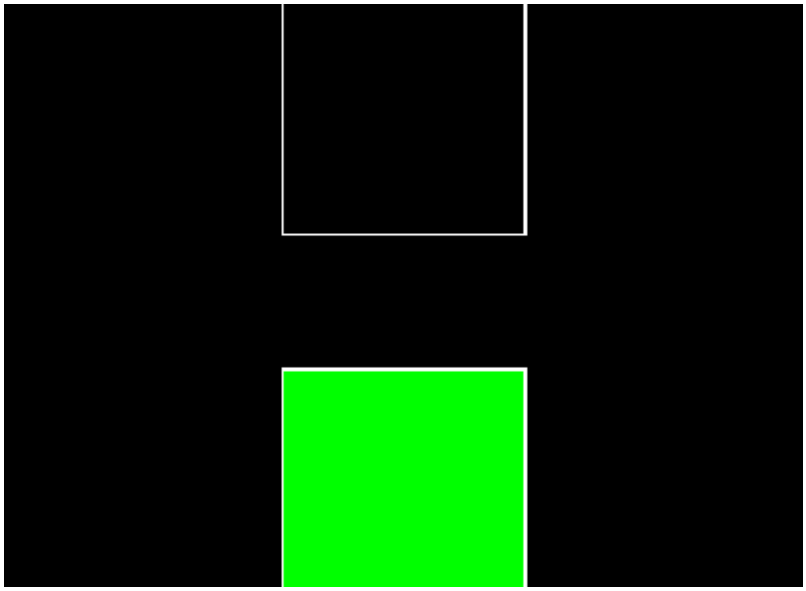
Now the subject has responded (and continues to touch the manipulandum, which has now turned green). A target marker has appeared at the top of the screen.



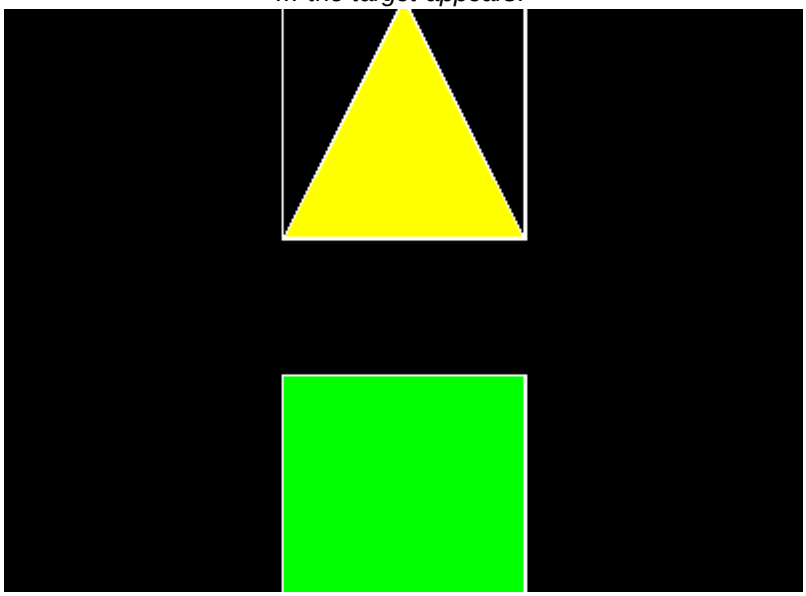
After a while, a distractor appears over the target marker...



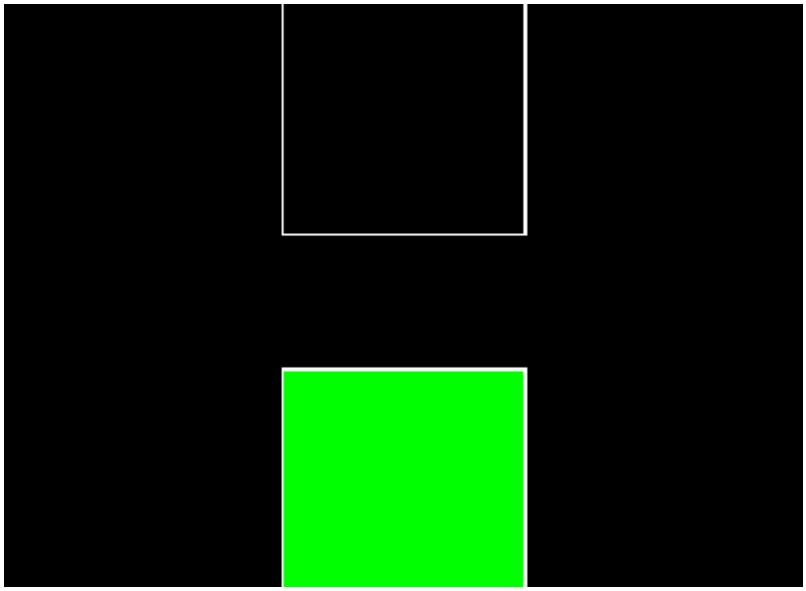
... and vanishes again. This process might be repeated several times, until...



… the target appears.



If the subject doesn't respond to the target, it still has a little bit of time after the target disappears to respond.



1.9 Before you start the task

Have you remembered to make your own copy of the supplied database and set it up under ODBC?

1.10 Randomness, pseudorandomness, drawing without replacement

Suppose I wish to pick a series of numbers from 1 to 6.

I could pick them **at random**. I could do this by rolling a true die. Every time I rolled the die, I would obtain a number from 1-6 with equal probability. The probability of obtaining a "1" would be 1/6 (approximately 0.17); the probability of obtainin a "2" would be 1/6, and so on.

If I rolled the die 100 times, I would expect to get roughly 17 ones, roughly 17 twos, roughly 17 threes, roughly 17 fours, roughly 17 fives, and roughly 17 sixes. It is *possible* that I would get 100 sixes—but very unlikely (the probability of this is 1.53×10^{-78}, or one in six hundred thousand quintillion quintillion quintillion quintillion). But it is certain that I would not get exactly the same number of ones, twos, threes, fours, fives, and sixes (because you can't have six equal whole numbers adding up to 100). If I rolled the die 60 times, you'd expect about 10 of each number—but it's also pretty unlikely that I'd get *exactly* 10 of each number.

If I rolled the die like this, there is absolutely no way to predict the next number that will come up on each roll.

So much for randomness.

If I want a series of 60 numbers and I want to ensure that I end up with equal numbers of ones, twos, threes, fours, fives, and sixes, I could simply take them in a **predictable order**: 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 5, 5, 5, 5, 5, 5, 5, 5, 5, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6. This gives me the distribution I want (ten of each number), but the sequence is anything but random, and is highly predictable. If you know the last number that came up, you have an extremely good idea of what the next number would be.

So much for total predictability.

To obtain a reasonable combination of unpredictability and obtaining an overall equal distribution of numbers, we could use a **pseudorandom** technique called **drawing without replacement**, or **draw-without-replacement**, sometimes shortened to **draw-w/o-replacement** or just **DWOR**. Imagine we would like a sequence of 60 numbers, each in the range 1-6, more or less randomly, but ensuring that we get 10 of each number. We could write the number "1" on ten pieces of paper, the number "2" on ten more pieces of paper, and so on. We could then put all 60 pieces of paper in a hat, shuffle them, and draw them out in sequence, *without replacing them*. We'd then be guaranteed ten of each type, but the local order would be fairly random. It would not be totally random—if we've drawn out 10 ones, 10 twos, 9 threes, 10 fours, 10 fives, and 10 sixes, then we can be absolutely certain that the last number out of the hat will be a 3. However, we'd have to remember the numbers that had come out so far.

So it's impossible to have a completely random order and to guarantee a certain distribution—but drawing without replacement is a good way to approximate randomness while guaranteeing a certain distribution.

There are several ways to draw without replacement. I've just described a situation in which 10 copies of each number (1-6) are put into the hat, shuffled, and drawn individually for 60 trials. It's possible (but again extremely unlikely!) that the first ten trials would all be ones, the next ten all twos, and so on. If we wanted to guarantee that *in every six consecutive trials* we will see each possible digit (1-6) once, we should do something different. We should write the number "1" on one piece of paper, the number "2" on a second piece of paper, and so on, up to six pieces of paper. We should then put the six into the hat, shuffle them, and draw them out (without replacement) for the first six trials. We should then put the six pieces of paper back in the hat, shuffle, and repeat the process for the next six trials, and so on. This process gives *less randomness* (if you know just the first five trials in a set of six, then in principle, you can have perfect knowledge of the sixth number to be drawn) but *better control over the local distribution* of numbers.

We've just seen two examples in which a **list** of six numbers (1, 2, 3, 4, 5, 6) are put into a hat and drawn for 60 trials. In the first, we put ten copies of each item in the list into the hat (giving 60 pieces of paper in total) and drew them. I call this a **draw-without-replacement (DWOR) multiplier** of 10. In the second, we put one copy of each item in the list into the hat, drew them until we'd run out of numbers (after six trials), then put them all back into the hat. I call this a DWOR multiplier of 1.

We've just seen the concept of a *list* of possibilities, which is multiplied by a *DWOR multiplier* to give a set of options that are then *drawn at random without replacement* from a hat; when the hat is empty, we restart the process.

The bigger the DWOR multiplier, the closer the DWOR technique comes to total randomness (if the DWOR multiplier were infinitely large, they are exactly the same process). The smaller the DWOR multiplier, the closer the technique is to total predictability.

This program offers the DWOR technique as a way of selecting possible values for various parameters (such as stimulus durations).

1.11 Results

MonkeyCantab always stores results in two places. One is a human-readable text file. The other is a database. (You choose the name of this file in the main parameters dialogue box, and you can choose the database here as well.)

1.11.1 Text-based results file

A sample results file is shown below. The configuration information is shown first; the results follow. (There aren't very many results, because I got bored creating the file.) The results section is shown in bold.

Tasks generally produce results in a comma-delimited format with a header line giving the field names. (This format is itself suitable for importing into a relational database.) Some tasks produce other human-readable summary information.

I encourage you to think of this file as a backup. The database contains all this information and can be used to retrieve both simple and highly detailed information about a subject's performance.

```
MonkeyCantab -- SUMMARY FILE
____________________________

IDENTIFICATION

Subject:               test
Session:               23
Date/time code:        08-Apr-2003 (15:20)
Comment:               test
Summary file name:     test-08Apr2003-1520-MonkeyCantab-summary.txt
Default ODBC database: MonkeyCantab_prototype

Box:                   0
Client computer name:  EGRET
Server computer name:  loopback

GENERAL PARAMETERS

Default media directory:
Link - Duration (s):                      5
Link - Play sound during link?            N
Link - Houselight on during link?         N
Reward - Give pellet?                     Y
Reward - Pellets per reinforcement:       1
Reward - Pellet pulse length (ms):        45
Reward - Interpellet gap (s):             0.5
Reward - Give pump?                       Y
Reward - Pump duration (s):               5
Reward - Pump contingent upon licking?    Y
Reward - If cont., lick pump time (s):    1
Reward - Play sound?                      Y
Reward - Activate extra reward device?    Y
Reward - Extra reward device time (s):    5
Punishment - Darkness?                    Y
Punishment - Darkness time (s):           10
Punishment - Play sound?                  Y
Punishment - Extra punishment device?     Y
Punishment - Extra device duration (s):   10
Sounds - Link sound is WAV?               N
Sounds - Link sound filename:
Sounds - Link sound frequency (Hz):       100
Sounds - Link sound type:                 Sine
Sounds - Link sound duration (s):         1
Sounds - Link sound level (0-100):        100
Sounds - Reward sound is WAV?             N
```

```
Sounds - Reward sound filename:           C:\WINNT\MEDIA\Utopia Exclamation.WAV
Sounds - Reward sound frequency (Hz):     1500
Sounds - Reward sound type:               Tone
Sounds - Reward sound duration (s):       1
Sounds - Reward sound level (0-100):      100
Sounds - Punishment sound is WAV?         N
Sounds - Punishment sound filename:       C:\WINNT\MEDIA\Windows Logoff Sound.wav
Sounds - Punishment sound frequency (Hz): 1000
Sounds - Punishment sound type:           Square
Sounds - Punishment sound duration (s):   2
Sounds - Punishment sound level (0-100):  100
Sounds - Marker1 sound is WAV?            N
Sounds - Marker1 sound filename:          C:\WINNT\MEDIA\RINGIN.WAV
Sounds - Marker1 sound frequency (Hz):    500
Sounds - Marker1 sound type:              Tone
Sounds - Marker1 sound duration (s):      1
Sounds - Marker1 sound level (0-100):     100
Sounds - Marker2 sound is WAV?            N
Sounds - Marker2 sound filename:          C:\WINNT\MEDIA\RINGOUT.WAV
Sounds - Marker2 sound frequency (Hz):    500
Sounds - Marker2 sound type:              Tone
Sounds - Marker2 sound duration (s):      1
Sounds - Marker2 sound level (0-100):     100
Sounds - Marker3 sound is WAV?            N
Sounds - Marker3 sound filename:          C:\WINNT\MEDIA\RECYCLE.WAV
Sounds - Marker3 sound frequency (Hz):    500
Sounds - Marker3 sound type:              Tone
Sounds - Marker3 sound duration (s):      1
Sounds - Marker3 sound level (0-100):     100

VISUAL OBJECTS

-- Object 0: redsquare

redsq,rectangle 0 0 100 100 -penstyle solid -penwidth 1 -pencolour 255 255 255 -
brushsolid 255 0 0

-- Object 1: bluesquare

bluesq,rectangle 0 0 100 100 -penstyle solid -penwidth 1 -pencolour 255 255 255 -
brushsolid 0 0 255

CONFIGURATIONS

-- Task 0: SpatialWM

Maximum no. trials (0 for no limit):                  50
Max. time (min) (0 for no limit):                     60
Minimum intertrial time (s):                          5
Maximum intertrial time (s):                          15
Maximum time for responding (s):                      10
Main object:                                          bluesquare
Previously-chosen object (not always relevant):       redsquare
Mark responses aurally?                               Y
Mark responses visually?                              Y
Visual marker object:                                 redsquare
Visual marker time (s):                               1
Blank screen to mark each response?                   N
Time to blank screen (s):                             0.5
Use prespecified trial sequence?                      Y
Prespecified trial sequence to use:                   myscheme
Grid type:                                            16-way scattered
Starting number of stimuli:                           9
```

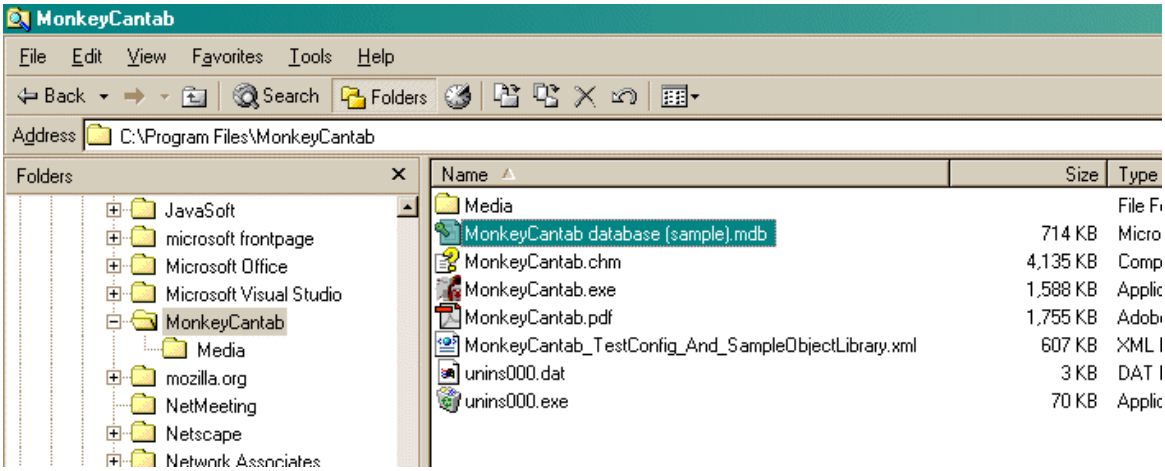
```
Increase number of stimuli?                              Y
... regardless of performance?                           Y
... criterion (every X trials or if X/last 20 correct): 1
When correct choice made, chosen stimulus...:            Replace with Previously
Chosen object
Time chosen stimulus vanishes for (if relevant) (s):     10

RESULTS

-- Results for Task 0: Spatial Working Memory

Trial,TrialStartTimeMs,GridType,NumStimuli,ChoiceConsequence,StimulusLocations,
Responses,ResponseTimesRelTrialStartMs,Correct,IncorrectRepeated,IncorrectTimeUp,
Rewarded,Punished
0,1116985724,16-way scattered,16,Vanish for fixed
time,0:1:2:3:4:5:6:7:8:9:10:11:12:13:14:15,0:1:2:7:6:5:4:8:9:10:15:11:14:13:12:3,8
049:10433:12121:14032:15512:16888:18559:21759:23135:24415:25751:27150:28454:29846:
32189:34414,Y,N,N,Y,N
1,1117036681,8-way scattered,1,Remain unchanged,5,0,3006,Y,N,N,Y,N
2,1117055808,16-way scattered,5,Vanish for next
choice,2:6:8:11:13,0:1:3:2:4,1475:3491:6539:8594:10402,Y,N,N,Y,N
3,1117077946,16-way grid,8,Disappear
permanently,0:2:4:6:9:11:13:15,0:1:3:2:4:5:7:6,1574:3262:4550:6053:7557:8917:10173
:11460,Y,N,N,Y,N
4,1117107698,8-way scattered,8,Replace with Previously Chosen
object,0:1:2:3:4:5:6:7,0:2:5:7:6:4:1:2,2075:3806:7012:8412:10244:12036:17049:19887
,N,Y,N,N,Y

MonkeyCantab FINISHED

Started at:        08-Apr-2003 (15:21)
Finished at:       08-Apr-2003 (15:24)

Successfully wrote to database: ODBC;DSN=MonkeyCantab_prototype;DBQ=D:
\Whisker\CODE\clients\rnc - cambridge\MonkeyCANTAB\MonkeyCantab database (sample).
mdb;DriverId=281;FIL=MS Access;MaxBufferSize=2048;PageTimeout=5;
```

1.11.2 Creating a new ODBC source

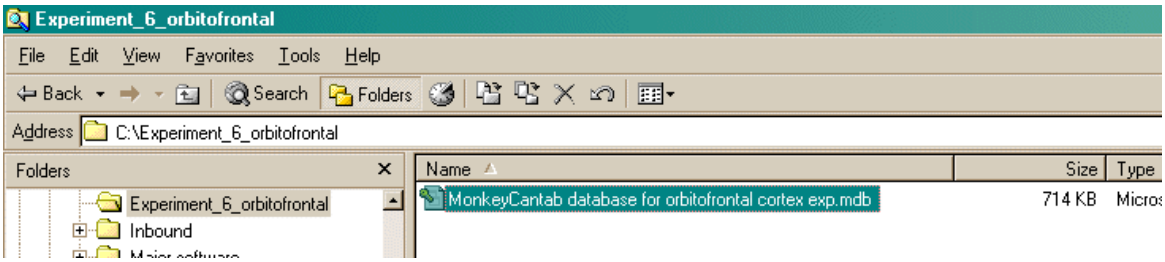
What happens if you're using MonkeyCantab for the first time, or you're starting a new experiment, and you need to set up a new ODBC (Open Database Connectivity) source for MonkeyCantab? You should configure it via **Control Panel** → **ODBC** [in Windows 2000, **Control Panel** → **Adminstrative Tools** → **Data Sources (ODBC)**]. This section shows you various ways to achieve this.

Remember: you shouldn't use the supplied database without making a copy for yourself. (It will work, but if you ever uninstalled or reinstalled MonkeyCantab, this file might be replaced or lost. It is much safer to make your own copy and set up ODBC to use your copy.)

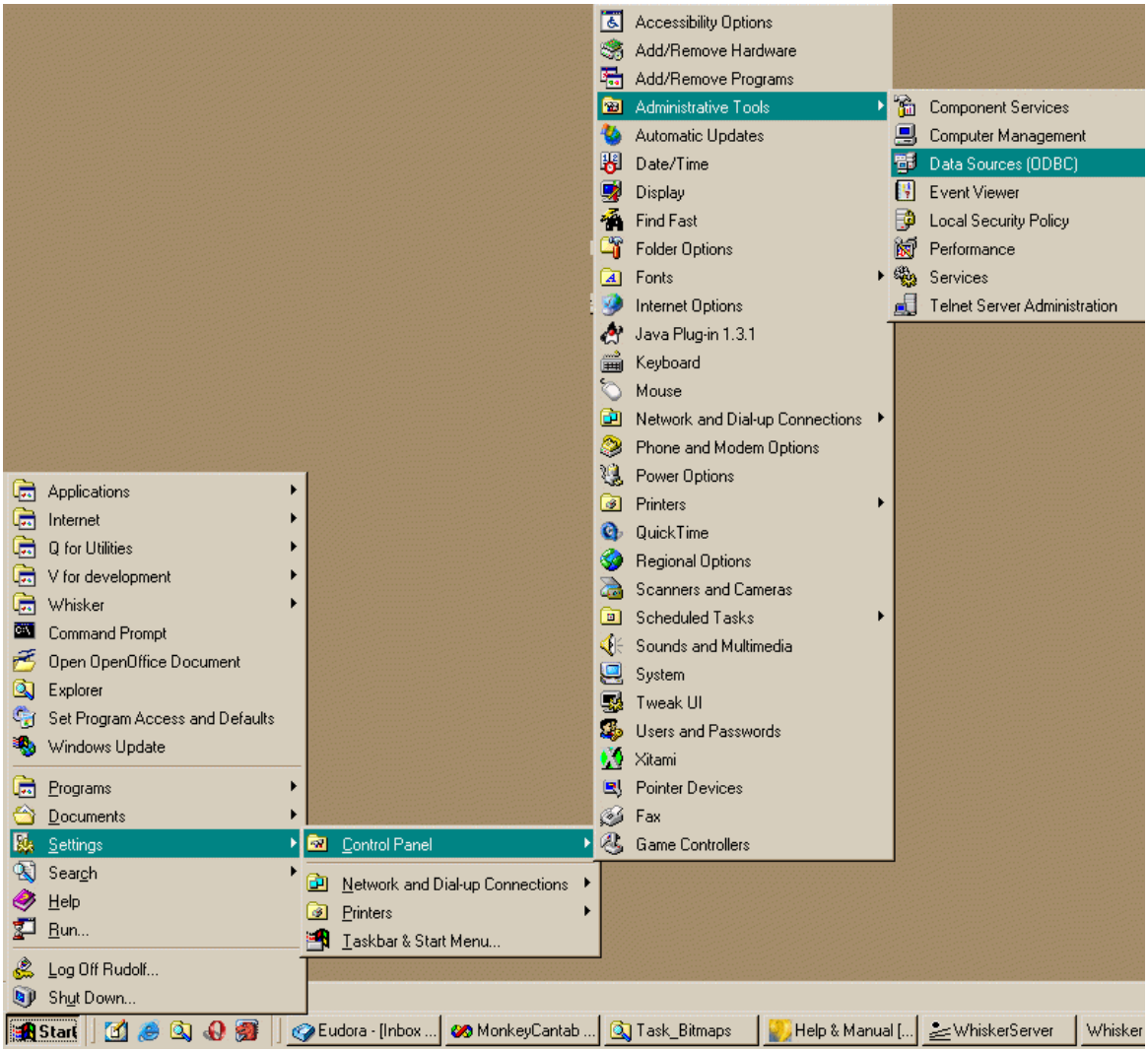
First, make a copy of the supplied database to store *your* data in. Copy the supplied database:



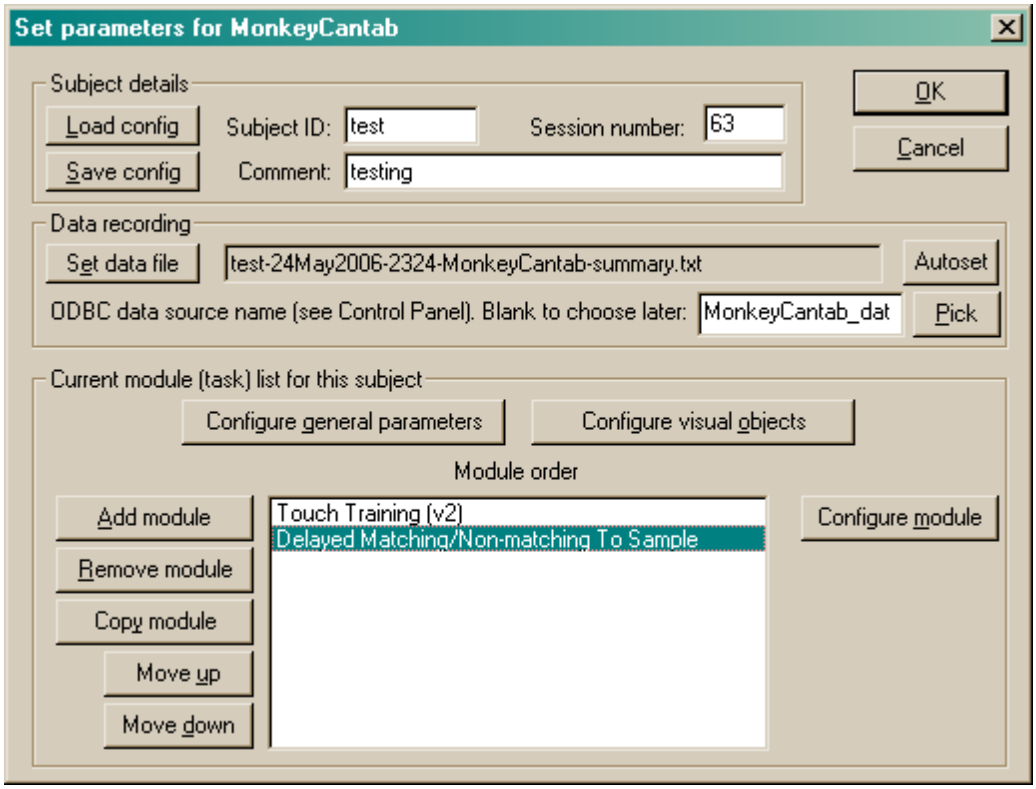
to your own:



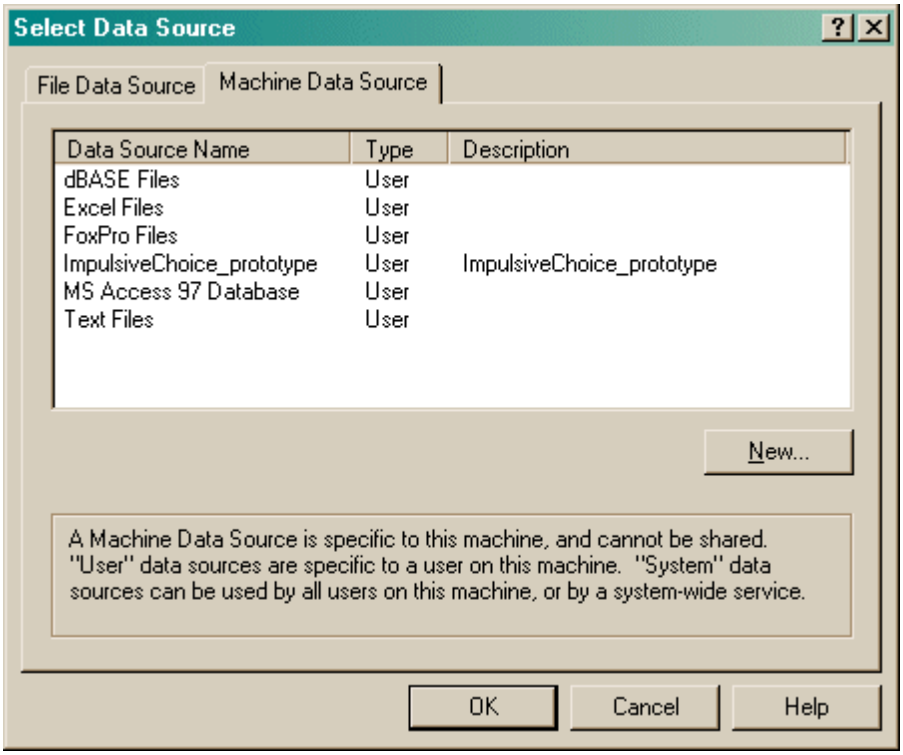
Now set *your* copy up as an **ODBC source** as follows. Choose **Control Panel** → **ODBC** [in Windows 2000, **Control Panel** → **Adminstrative Tools** → **Data Sources (ODBC)**, or the equivalent for your version of Windows:



Alternatively, you can get to the same point from MonkeyCantab itself. Click **Pick** from the Parameters dialogue:

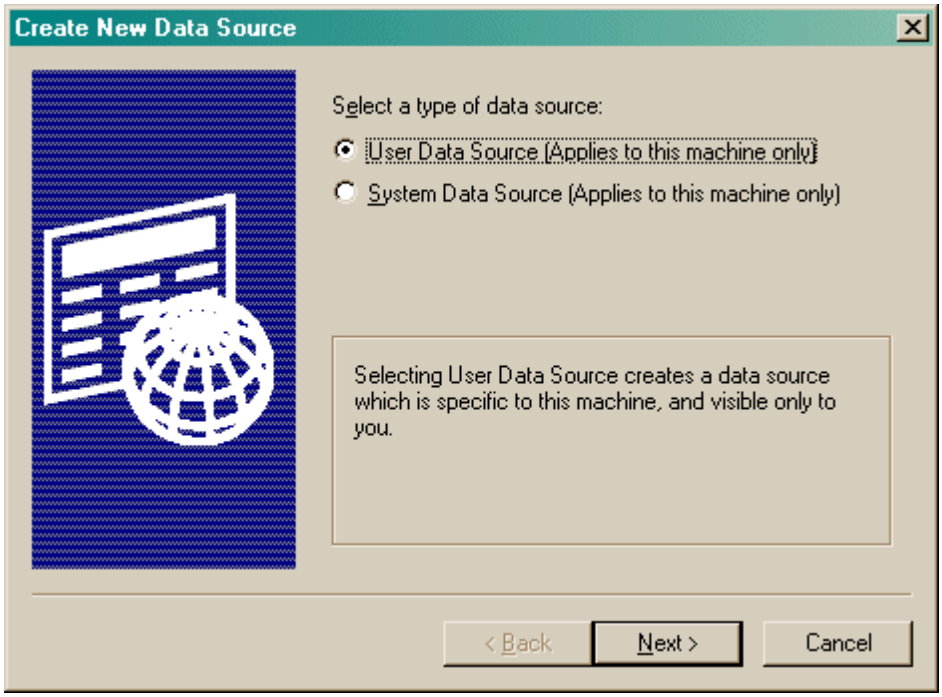


Either way, you get to something like this:

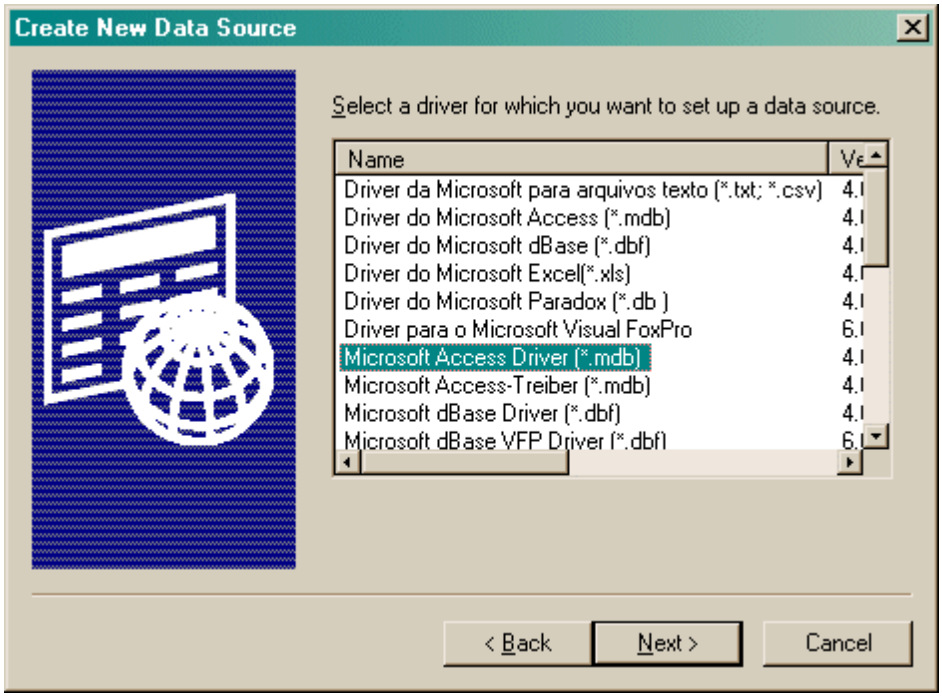


Let's assume that you have **already made a working copy of the prototype database supplied with the task,** as described above. How do we go about setting this up as an ODBC data source?

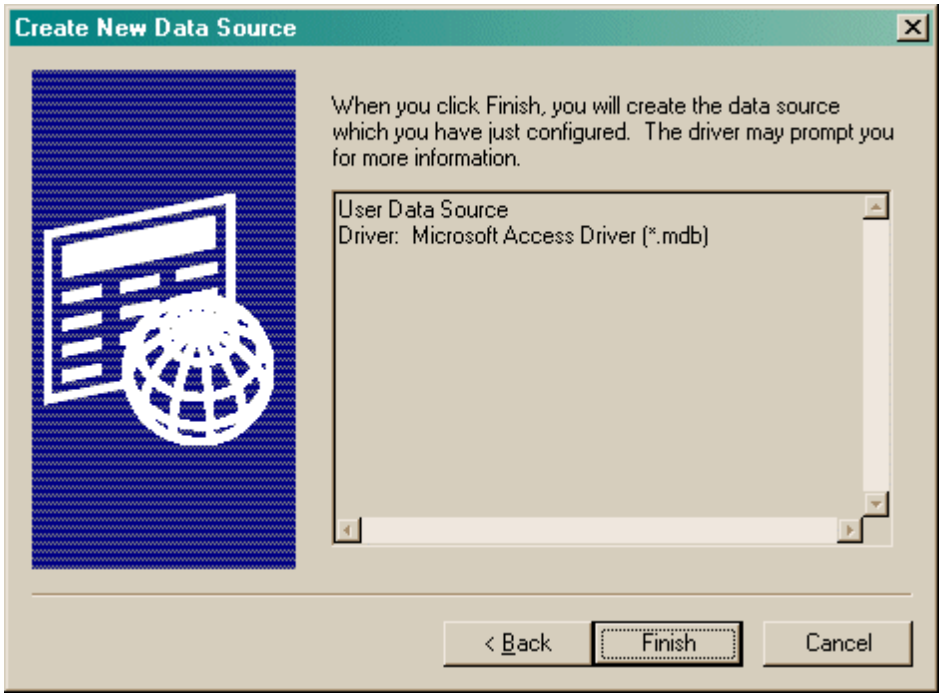
Click New.



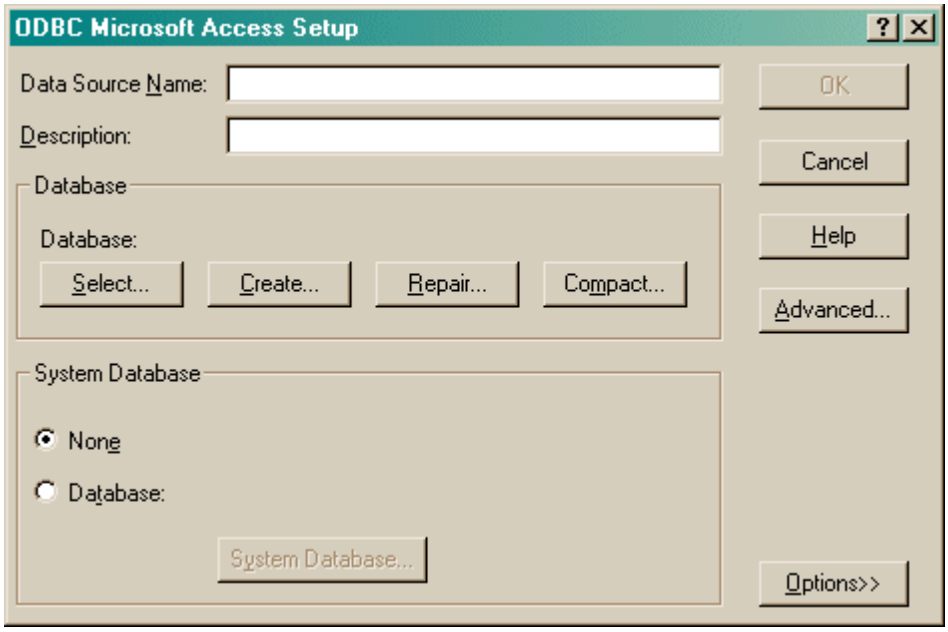
Choose a User or System data source. **User** is probably more sensible. Click Next.



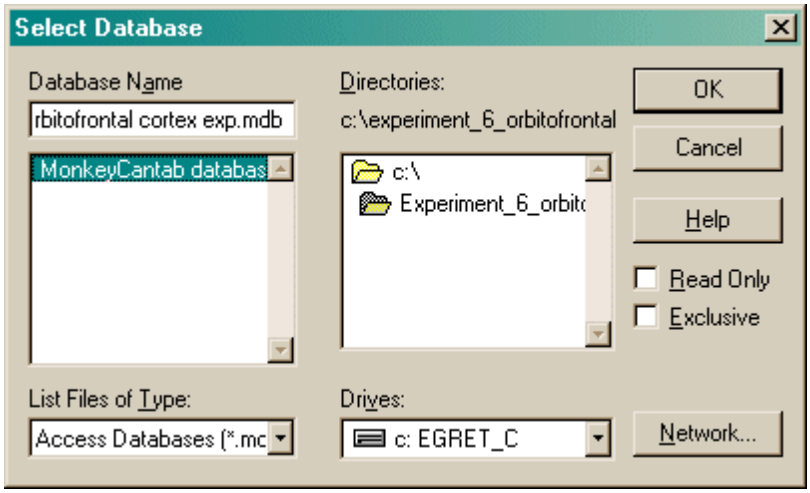
Choose your database driver. You probably want one that's in your language, and the supplied database is in Microsoft Access format (although MonkeyCantab itself will store data in any suitable ODBC-compatible database that has the right table and field names). Click Next.



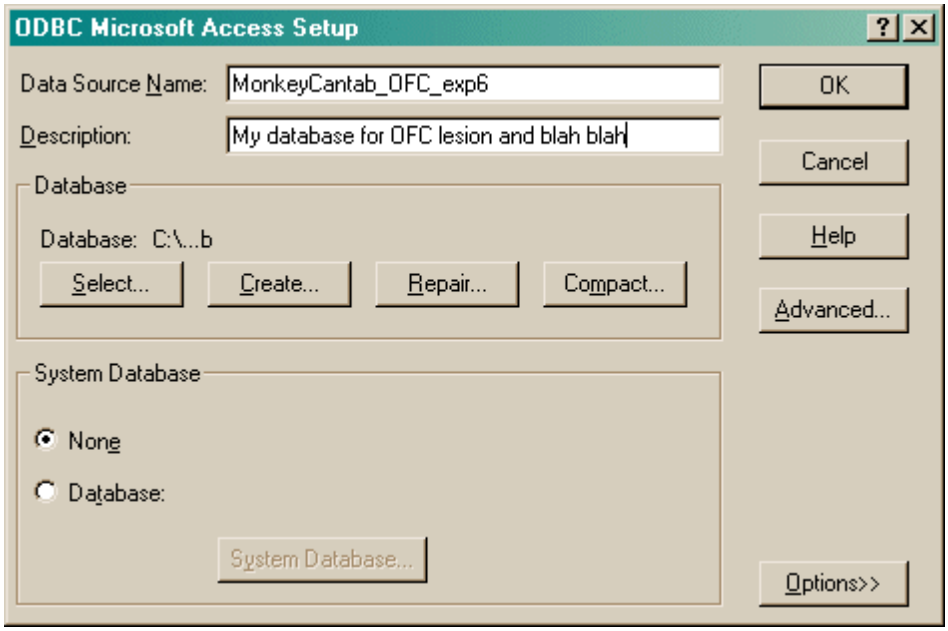
Click Finish.



You should fill in the **Data Source Name (no spaces)** and the **description,** and **Select** a database. When you click Select, this dialogue box appears:



Choose your database here and click OK. Your ODBC data source fields should now all be set up:



Click OK. You will be returned to the ODBC selection screen with your new data source now available.

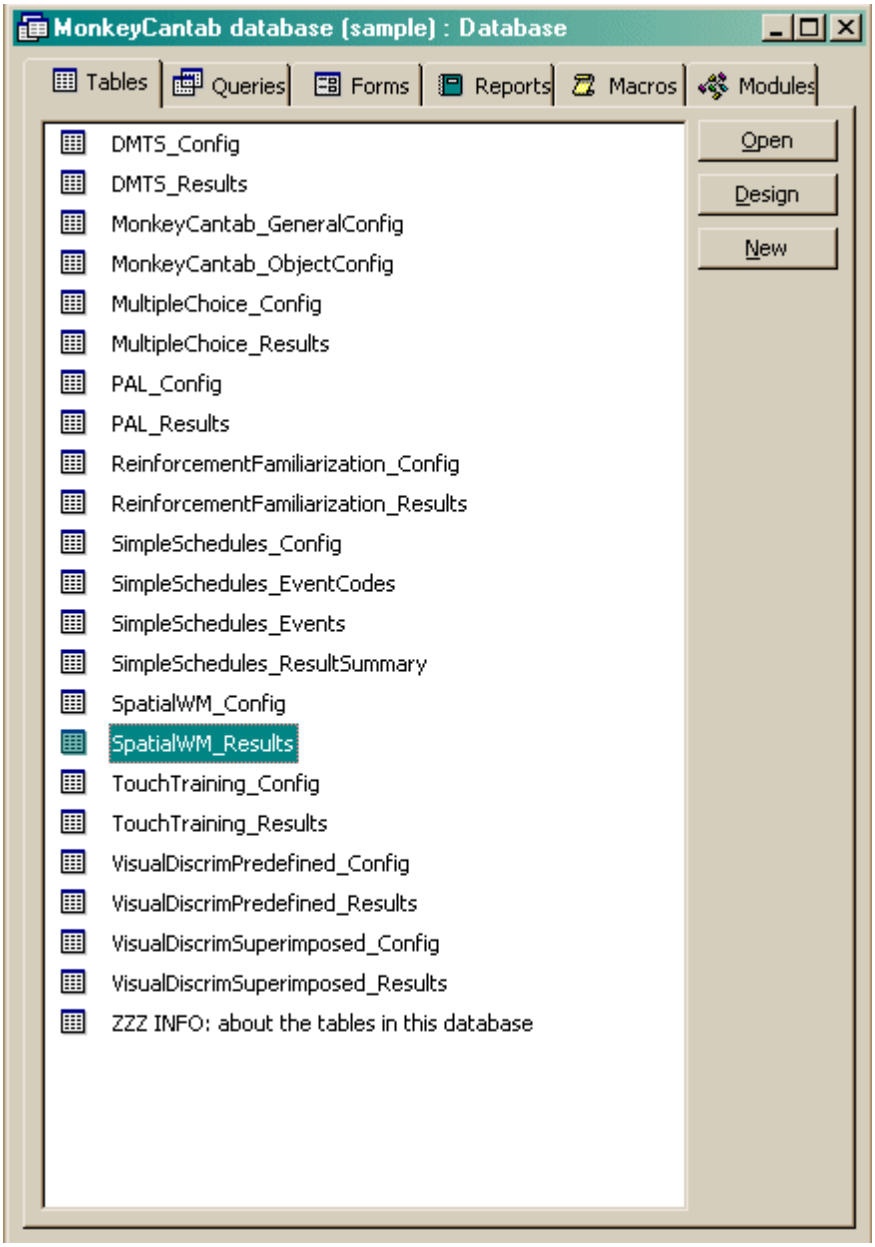
1.11.3 Using the Microsoft Access database for MonkeyCantab

Remember: you shouldn't use the supplied database without making a copy for yourself. (It will work, but if you ever uninstalled or reinstalled MonkeyCantab, this file might be replaced or lost. It is much safer to make your own copy and set up ODBC to use your copy.)

When supplied, the database is called **"MonkeyCantab database (sample).mdb"**. Make a copy before using it!

You need Microsoft Access (97 or higher) to use this database. Sorry about that.

When you open the database, it looks like this:



MonkeyCantab will store its results here. The table **"ZZZ INFO: about the tables in this database"** summarizes the information held in each table. Click a table and click **Design** to view a list of all the fields. Here, for example, is the design view for the SpatialWM_Results table (which stores results information for the Spatial Working Memory task):

SpatialWM_Results : Table

Field Name	Data Type	Desc
RecordNum	AutoNumber	(Automatically generated.)
DateTimeCode	Date/Time	Date/time the program was started.
Subject	Text	Subject
Box	Number	Box number
ModuleNumber	Number	Order of this task in the master task list
Trial	Number	Trial number
TrialStartTimeMs	Number	Trial start time by the system clock (ms)
GridType	Text	Description of the array of possible locations
NumStimuli	Number	Number of stimuli used
ChoiceConsequence	Text	What happens when an object is chosen (correctly)?
StimulusLocations	Text	Locations of the stimuli. Example: "3:1:4:7" - four stimuli used; positions 1-9 represent a nine-wa
Responses	Text	Responses, in order. Numbers are stimulus numbers, not locations. Example: "3:4:2:2" means th
ResponseTimesRelTrialStartM	Text	Response times relative to the trial start time, in ms. Example: "3147:8192:13892:19002"
Correct	Yes/No	Did the subject perform correctly?
IncorrectRepeated	Yes/No	Did the subject fail because it chose a stimulus twice?
IncorrectTimeUp	Yes/No	Did the subject fail becase it didn't respond within the criterion time?
Rewarded	Yes/No	Was the subject rewarded?
Punished	Yes/No	Was the subject punished?

Don't modify anything in Design view unless you know what you're doing!

If you close the Design view and click **Open** instead, you see the *contents* of this table. Here is the contents of the SpatialWM_Results table (holding results for the Spatial Working Memory task). I entered some sample results into this table.

SpatialWM_Results : Table

_RecordNu	DateTimeCod	Subject	Box	Module	Trial	TrialStartTime	GridType	NumStimuli	ChoiceConseq	Stimulusl
1	1/2003 16:22:39	test	0	0	0	255503893	16-way scattere	16	Vanish for fixed	0:1:2:3:4:5
2	1/2003 16:22:39	test	0	0	1	255553774	8-way scattered	1	Remain unchan	5
3	1/2003 16:22:39	test	0	0	2	255575380	16-way scattere	5	Vanish for next	2:6:8:11:1
4	1/2003 16:22:39	test	0	0	3	255603318	16-way grid	8	Disappear perm	0:2:4:6:9:1
5	1/2003 16:22:39	test	0	0	4	255629134	8-way scattered	8	Replace with Pi	0:1:2:3:4:5
(AutoNumber)										

Feel free to explore the tables.

When you want to extract data for analysis, you may want to create **queries** to do so. (Queries are listed in the "Queries" section of the main database screen.) Queries can be created using Access's visual query design system, or using the language SQL (Structured Query Language). A little on relational database principles and SQL follows.

1.11.4 Relational databases in general

I have found the most useful way to store data is in a **relational database,** often called a relational database management system (RDBMS). A relational database stores data in **tables,** which are made up of *fields* and *records:*

A table:			*five fields:*		
	Date	Rat	NumResponses	NumStimuli	NumReinforcements
one record:	17/2/00 12:29:00	M4	56	5	1
another:	17/2/00 14:37:06	M5	437	43	8
… and so on	17/2/00 12:54:00	M4	263	26	5

The driving principle behind a relational database is this: **never duplicate data.** Let's say our rats came from two groups, Sham and Lesion. If we wanted to record this in the database, so we could analyse data by group, we could store it like this:

Table *BigData*

Date	Rat	Group	NumResponses	NumStimuli	NumReinforcements

17/2/00 12:29:00	M4	sham	56	5	1
17/2/00 14:37:06	M5	lesion	437	43	8
17/2/00 12:54:00	M4	sham	263	26	5

However, this introduces two problems. Firstly, it generates very large tables. Secondly, and more importantly, it is unclear what to do if the data is inconsistent – let's say the underlined 'sham' was changed to 'lesion' by mistake. The database would then not know whether rat M4 was in the Sham or Lesion group – there would be entries for both. The solution to both problems is to create two tables, *linked* on the smallest possible unit of information (in this example, the rat name):

Table *Responses*

Date	**Rat**	NumResponses	NumStimuli	NumReinforcements
17/2/00 12:29:00	**M4**	56	5	1
17/2/00 14:37:06	**M5**	437	43	8
17/2/00 12:54:00	**M4**	263	26	5

Table *Groups*

Rat	Group
M4	sham
M5	lesion

By using the rat name as a **key** (also known as a *foreign* key), the database can link the two tables together whenever we want to know how many responses the two groups made on average.

When we want to find out that sort of information, we **query** the database, specifying how we want to see the data. We could, for example, obtain the following (ignoring a glaring scientific error!):

Query *AverageByGroups*

Group	NumberOfSubjects	MeanNumResponses	MeanNumStimuli	MeanNumReinforcements
sham	2	159.5	15.5	3
lesion	1	437	43	8

Summary of database principles

So relational databases split up the data (which should be entered in well-designed tables without any duplication of information) from queries that look at the data in an infinite variety of ways.

A concrete example: Microsoft Access 97

Microsoft Access 97 is a commonly-used relational database for PCs. It isn't perfect, by a long shot, but I've found it good enough. It supports **structured query language (SQL)** for designing queries; this is a powerful quasi-English language. For example, the query shown above would be written in SQL like this:

```
SELECT group,
    count(*) as NumberOfRats,
    avg(NumResponses) as MeanNumResponses,
    avg(NumStimuli) as MeanNumStimuli,
    avg(NumReinforcements) as MeanNumReinforcements
FROM responses, groups
```

```
WHERE responses.rat = groups.rat
GROUP BY group
;
```

If you find all this a bit cryptic, Access also provides a graphical interface for designing queries.

Getting data out of a database

Given a well-designed database, you should be able to get the data out in any conceivable way. The size of this manual doesn't permit a detailed look at relational database design or queries, but there are abundant sources. If you use Microsoft Access, there's the help system, but I also recommend Viescas JL (1997), *Running Microsoft Access 97*, Microsoft Press. Beyond that there is a whole field of database design.

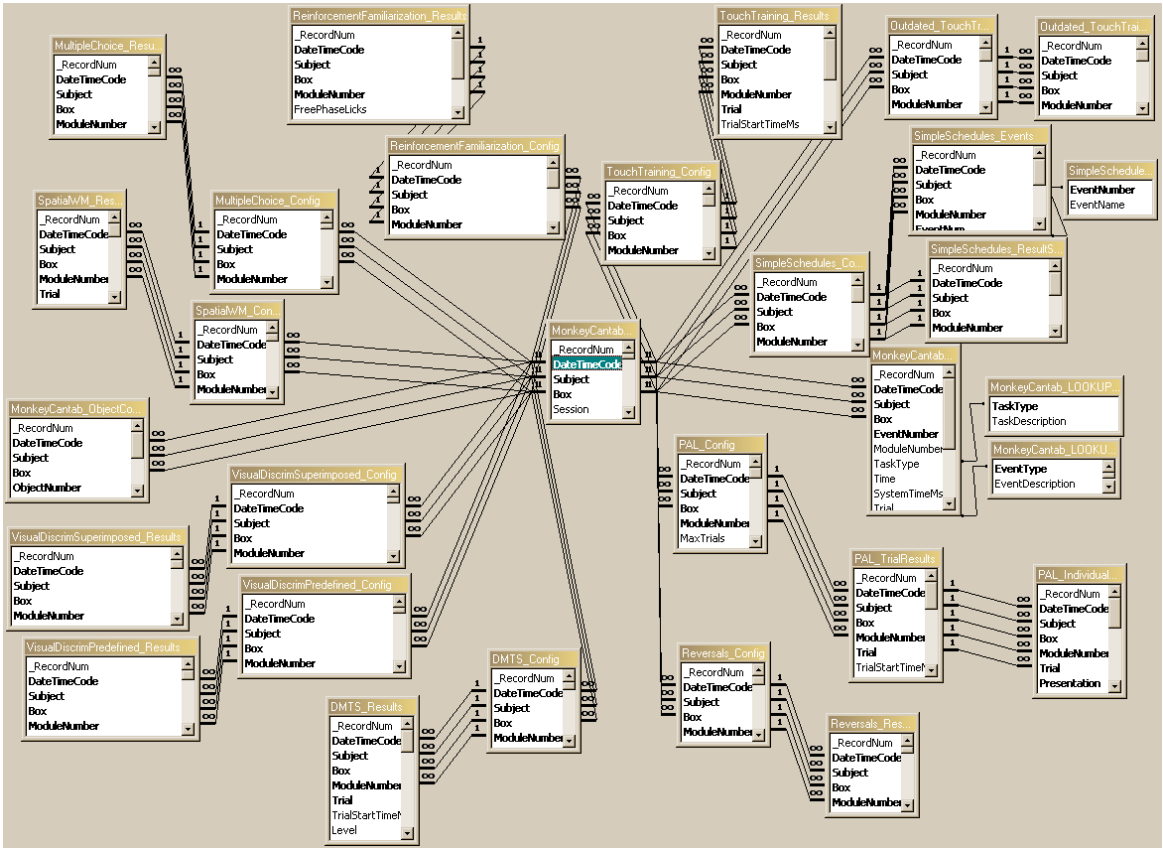
Tip

I operate on the principle that any view of the data is achievable. If the graphical query design can't do it, you can use SQL. If SQL can't do it alone, you can use Visual Basic to augment it. If all that fails (and it hasn't failed me yet) you can always re-export the data and use a general-purpose programming language to analyse it. If the data's there, you can get at it.

One thing is worth noting: modern statistical packages (e.g. SPSS, *http://www.spss.com/*) are starting to support the ODBC standard for exchanging information with databases. You can set up database queries to create views of the data that your stats packages can use, then set up sequences of ODBC capture, analysis and graphical presentation in your stats package. Then whenever you import new data, you can run the entire analysis in a matter of seconds. If you handle large volumes of data, it easily repays the initial effort.

1.11.5 Database structure

This is the structure of the MonkeyCantab database. Not shown are the ImpulsiveChoice_Config and ImpulsiveChoice_Results tables, which follow the same key structure as the other tasks.



1.12 TROUBLESHOOTING

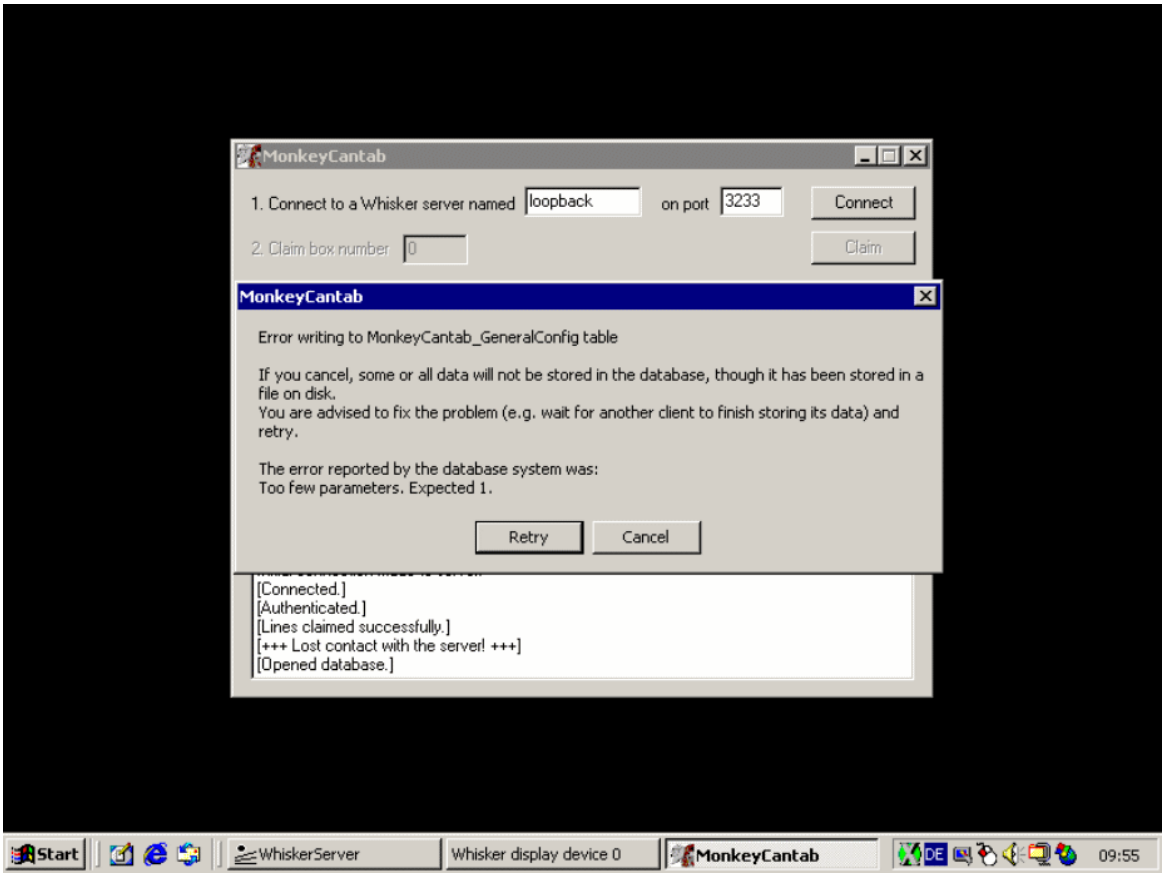
Here's a collection of problems commonly encountered.

- When my task is saving data, it generates a database error. Why? What do I do?

1.12.1 Troubleshooting - database errors

When MonkeyCantab finishes, it tries to save the results into a database. Here are some things that can go wrong.

ODBC "Too few parameters. Expected 1" error



This cryptic error, generated by the Windows ODBC (database) system, indicates that **a field used by MonkeyCantab does not exist in the database.** In this example, a field is missing from the "MonkeyCantab_GeneralConfig" table.

Usually, this means that you are trying to use a **newer version of MonkeyCantab than the one your database was designed for.**

First, **don't panic.** MonkeyCantab writes data to a text-based results file before it attempts to write to a database - so your data should already have been saved to disk.

Next, you can do one or more of these things.

1. Make a copy of the database supplied with your current (newer) version of MonkeyCantab, create an ODBC link to it, and click "Retry". This will store your data. You may later need to merge it with data in your previous (older) database, if the two sets of data must be integrated, but that is a problem for later.
2. You can also alter the design of your current (older) database, so that it has all the fields in the newer version of the database. This is a more skilled procedure.

Index

- M -