PigTab

A Whisker client

---

by Rudolf Cardinal

www.whiskercontrol.com

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

PigTab

Printed: March 2024 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 PigTab

1.1 About PigTab

Purpose

Cognitive test battery for minipigs.
Written for Sidse Arnfred (Copenhagen, Denmark).
The battery includes:

- Reinforcement familiarization
- Touchscreen training
- Visual discriminations (simple and compound) with reversal and intradimensional and extradimensional set-shifting
- Delayed matching and non-matching to sample
- Spatial working memory
- Three-choice serial reaction time test
- Delayed matching to location
- Progressive ratio schedule

Software requirements

Requires Whisker v2.0 or greater.

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 and Shibley Rahman for helpful discussions on the ID/ED and PAL tasks!

Copyright

Revision history

- 5-Sep-2002. Started.
- 28-Sep-2002. First version completed, with documentation and object definition system.

1.2 Required devices

The program requires to claim devices in groups named **box0, box1, box2...** with device names as listed below in bold:

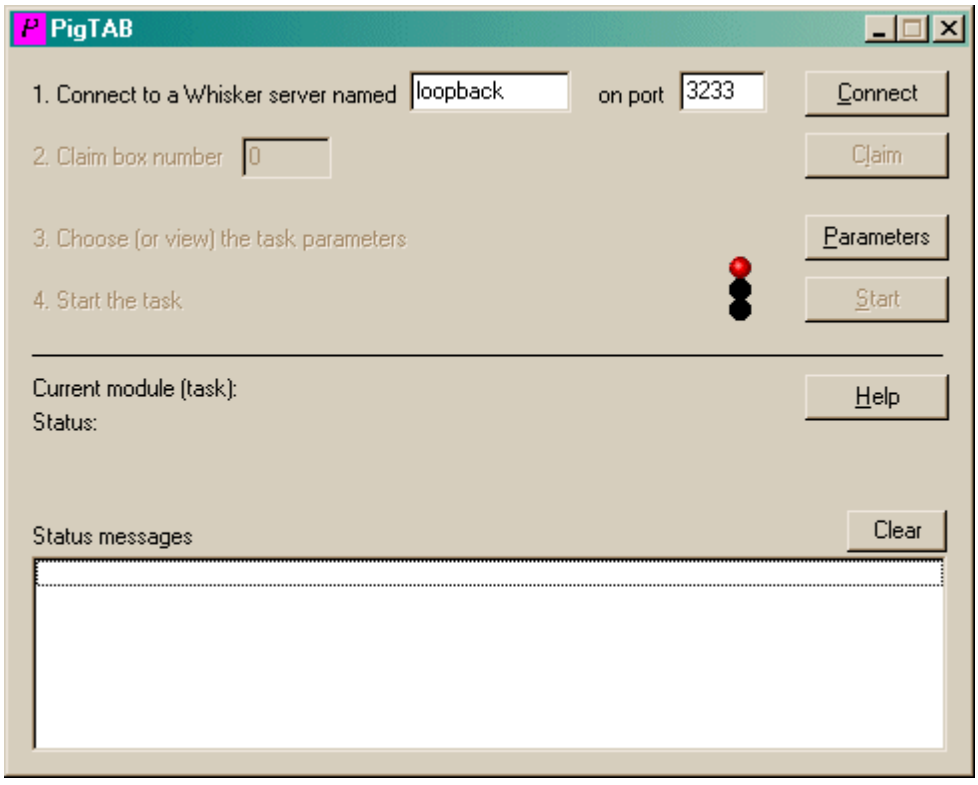
```
# ---------------- Box 0 definition
# INPUTS

# OUTPUTS
```

```
line    24      box0    HOUSELIGHT
line    27      box0    PELLET
line    30      box0    PIGPUSHER

# DISPLAY
display  0      box0    SCREEN

# AUDIO
audio    0      box0    SOUND

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

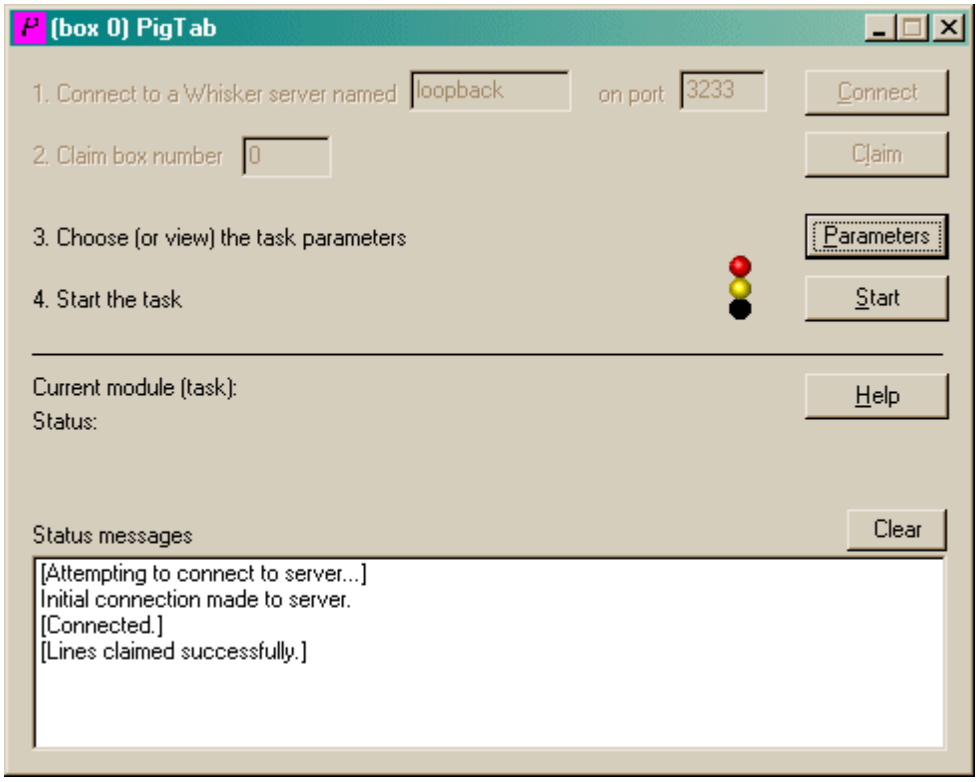
Please ensure that these devices are available and listed in the device definition file in use by the server. (The snippet above shows an extract from a typical definition file.)

1.3 Using PigTab

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



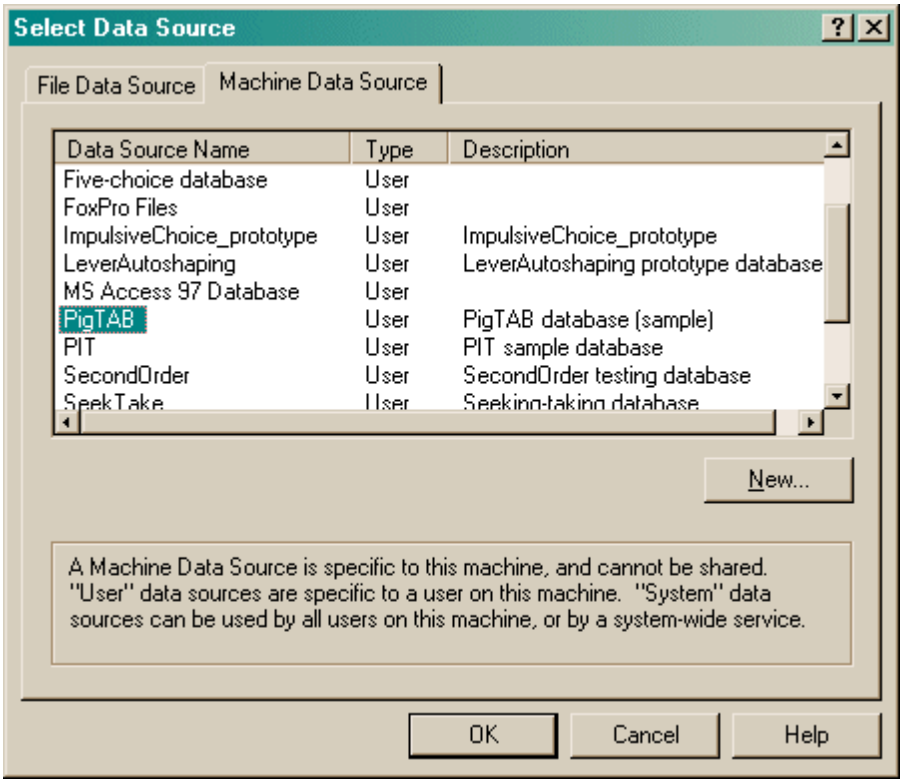
You must connect to a Whisker server, claim an operant chamber (box), and set up the parameters for your tasks. Configuration is explained on a separate page. Once that's done, the traffic lights will turn amber.



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

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

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.

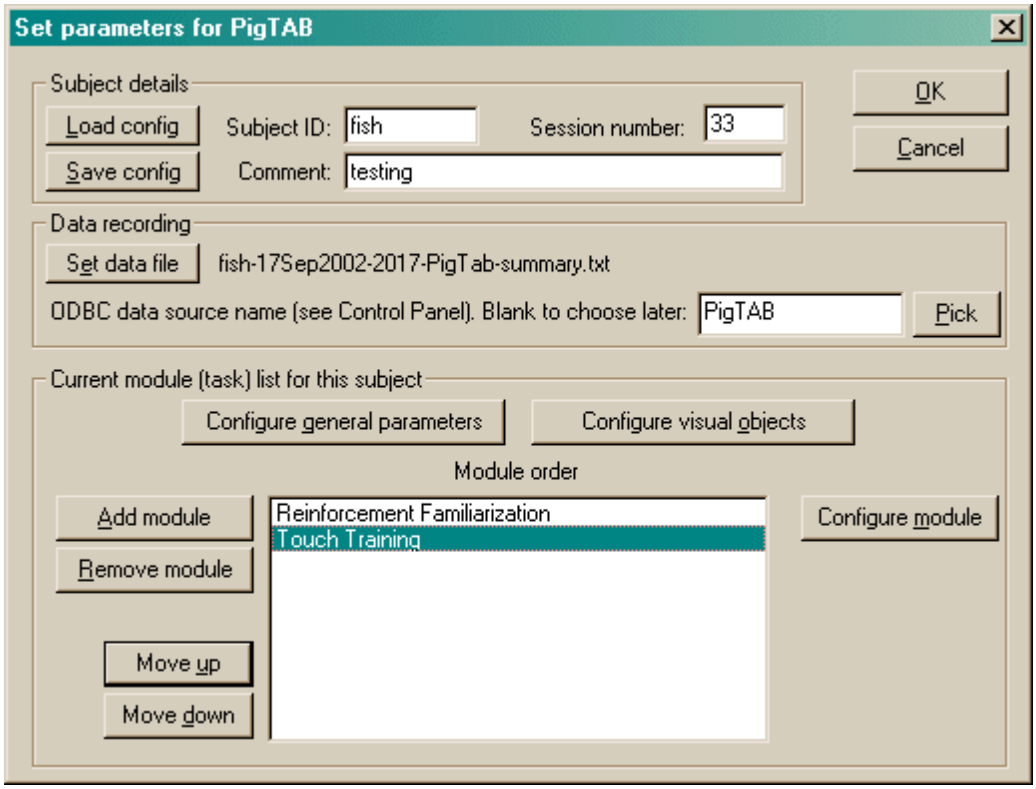


Your data will be saved and PigTab has then finished.

1.4 Configuring PigTab (general settings)

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

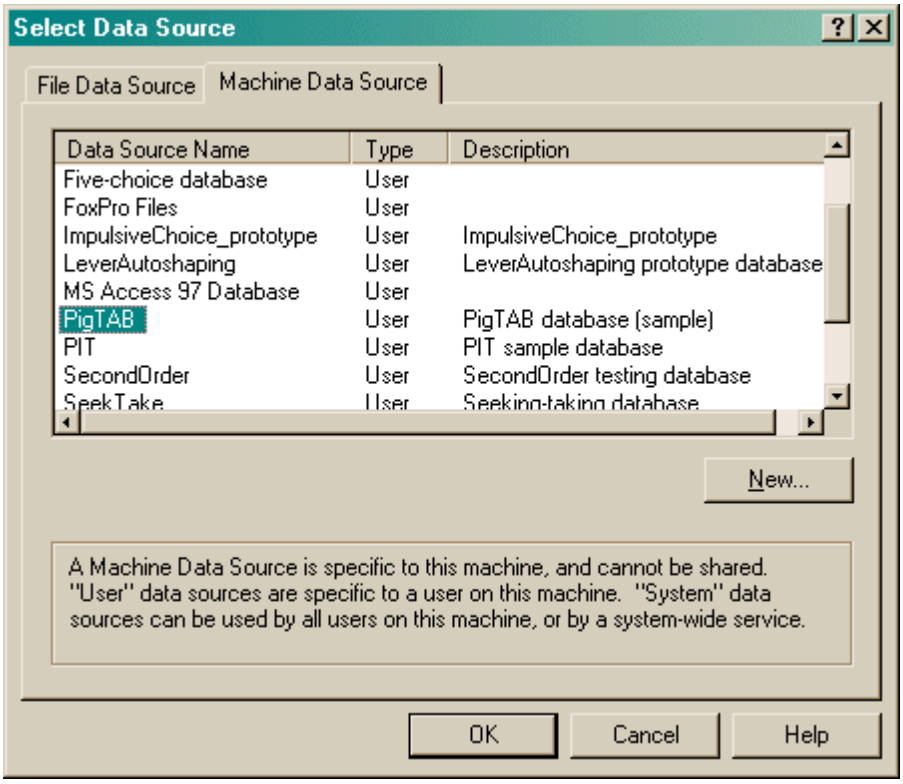
PigTab allows you to run a selection of tasks. You must therefore tell PigTab which tasks you want to run, and in what order.



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.) 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. 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.

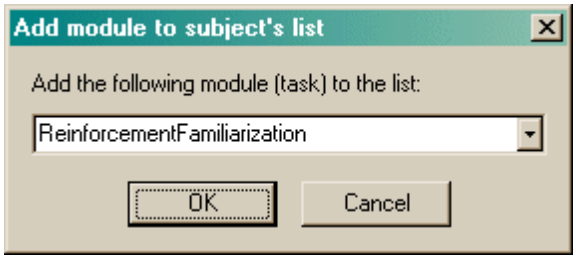
A sample configuration is supplied with PigTab. It's called "PigTab_TestConfig_And_SampleObjectLibrary.xml" and it lives in the directory you installed PigTab into (typically C:\Program Files\PigTab).

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).

The bottom half of the screen contains the module (task) list for the subject. You may specify any number of tasks from those that PigTab 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.

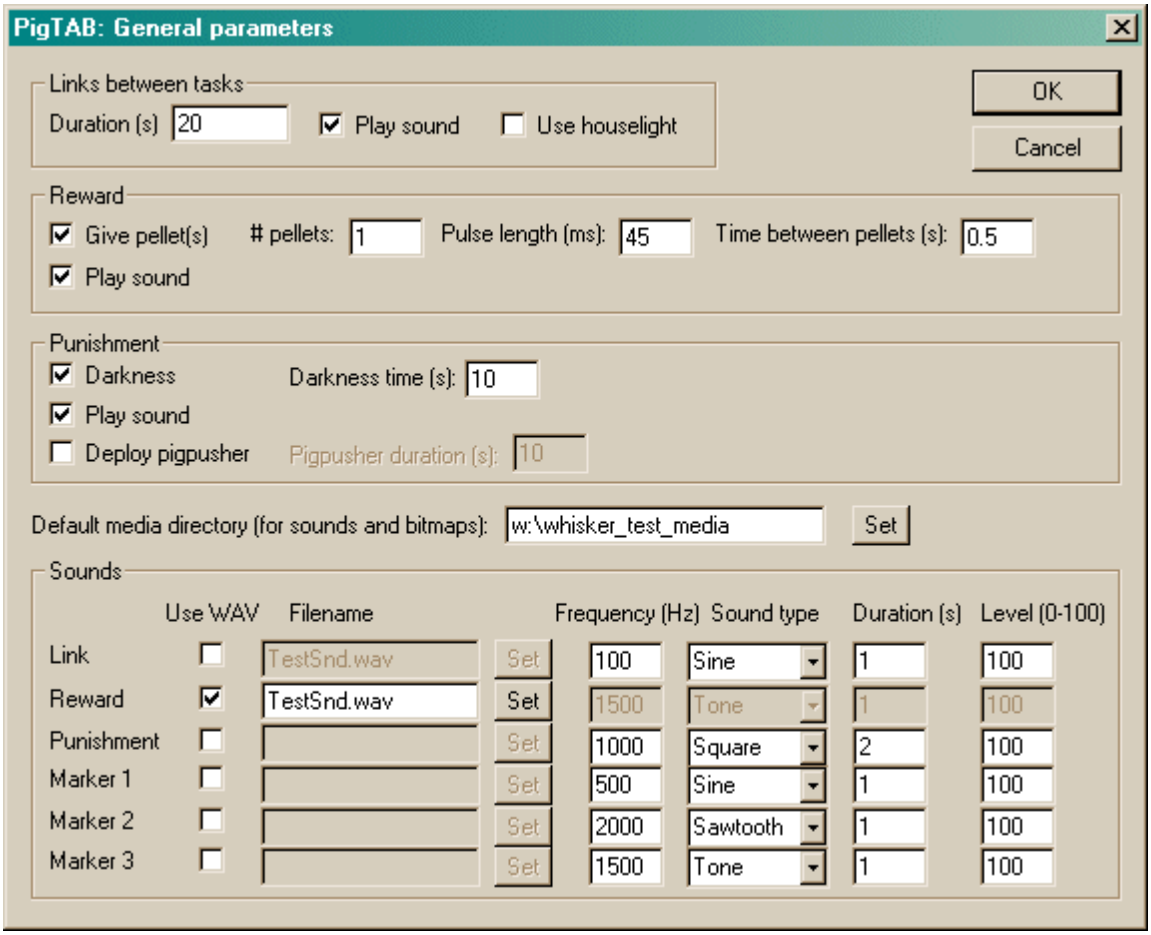
Click **Configure visual objects** to set up objects used by the tasks.

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 (with their corresponding abbreviated forms, which you will sometimes see) are:

- Reinforcement Familiarization (*ReinforcementFamiliarization*)
- Touch Training (*TouchTraining*)
- Visual Discrimination and Set Shifting (*VisualDiscriminationSetShifting*)

- Delayed Matching/Non-matching To Sample (*DelayedMatchingToSample*)
- Spatial Working Memory (*SpatialWorkingMemory*)
- Three-Choice Serial Reaction Time (*ThreeChoice*)
- Delayed Matching To Location (*DelayedMatchingToLocation*)
- Progressive Ratio Schedule (*ProgressiveRatio*)

1.4.1 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.

Reward

- **Give pellet?** If you select this option, pellets will be delivered when the subject is rewarded (M&M® sweets, for example, in the sphere of Danish pig testing).
- **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, 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.
- **Play sound?** If you select this option, the Reward sound will be played when the subject is rewarded (see *Sounds* below).

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

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).
- **Play punishment sound?** Chooses whether or not to play the Punishment sound as part of the punishment.
- **Deploy pigpusher?** Chooses whether or not to activate the Pig Pusher, a fabled device for shoving pigs bodily away from the touchscreen. No M&Ms for you, little pig. **(NOTE:** not having seen a pig pusher, we have not set up a safety timer to prevent it being activated for too long; we presume that it is not a dangerous device.)
- **Pigpusher duration (s).** Sets the length of time for which the pigpusher is switched on.
- Shocks are not explicitly implemented as a punishment option (for ethical reasons: this seems pretty severe). Contact rudolf@pobox.com if this causes problems.

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 PigTab, 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:

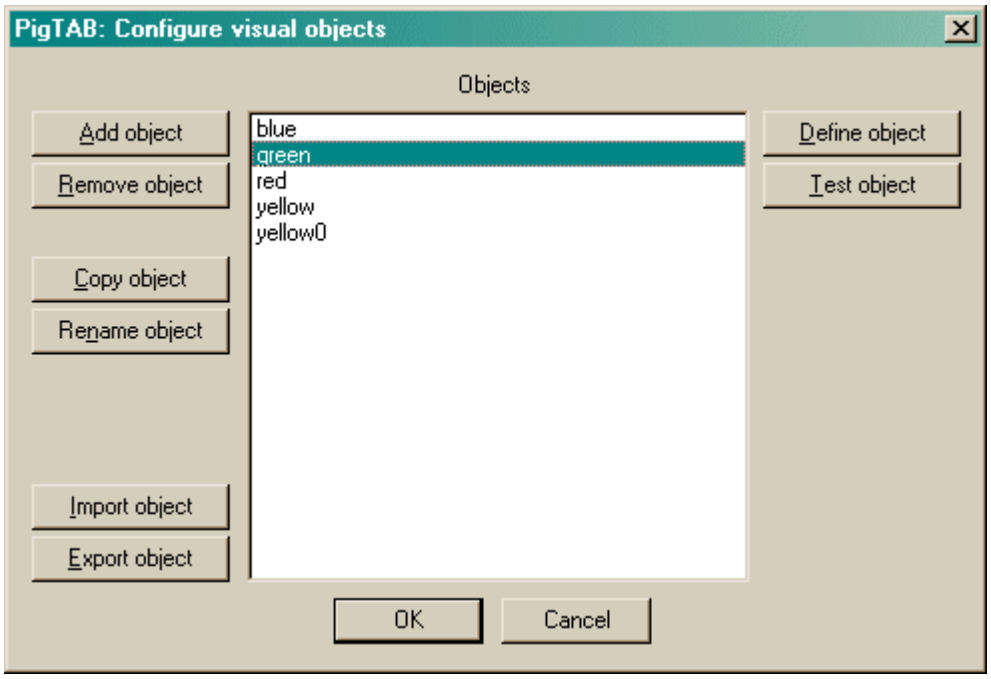
- **Use WAV file.** Sounds may either be played as simple tones or as WAV files.
- **Filename.** To specify WAV files. Click **Set** to browse for the file. (Only applicable to WAV sounds.)
- **Frequency (Hz).** Specifies the sound's frequency in Hertz. (Only applicable to non-WAV sounds.)
- **Sound 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). (Only applicable to non-WAV sounds.)

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.

1.5 The visual object library

Editing the library

Every visual object (picture) used by PigTab 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 **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 **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.

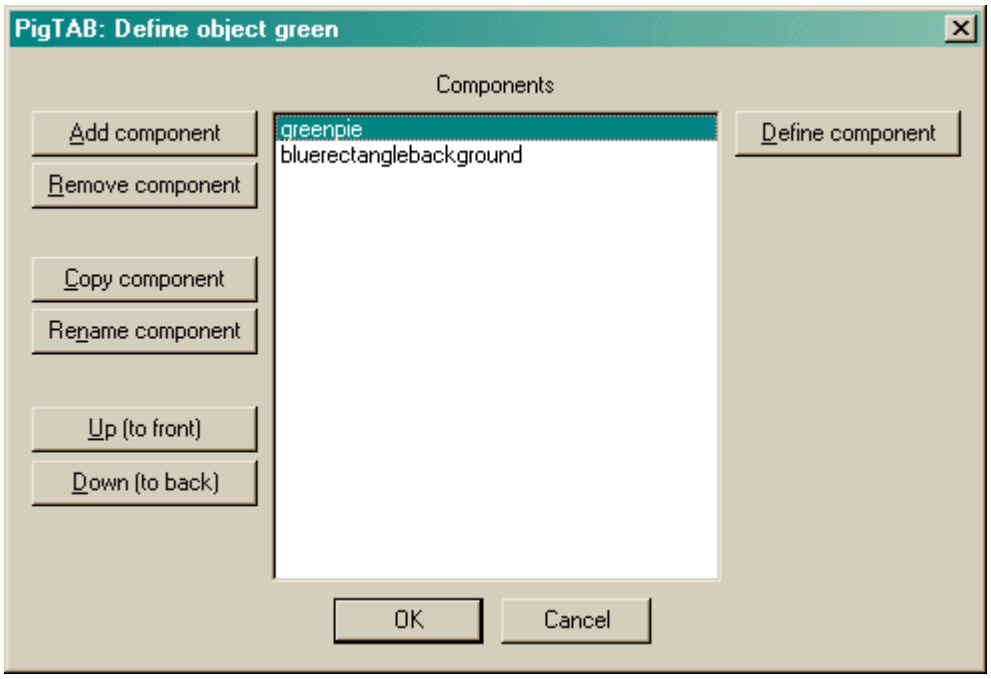
A sample object library is supplied with PigTab. It's called **PigTab_TestConfig_And_SampleObjectLibrary.xml.**

Editing individual objects

When you use your new object in a PigTab 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.

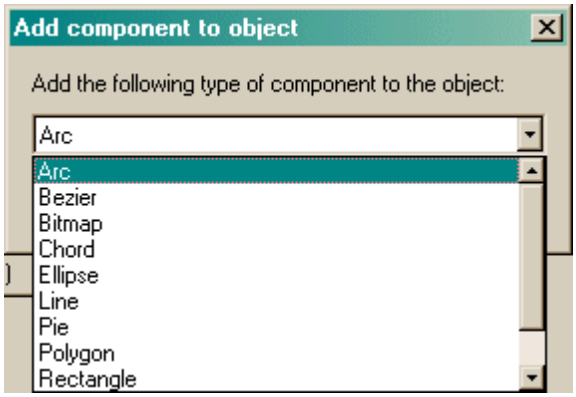
Editing components of individual objects

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.)



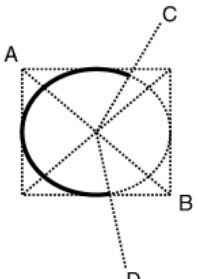
- Click **Copy** or **Rename** to copy or rename a component. (Note that two components can't have the same name.)
- Click **Up** or **Down** to move components up or down the stack.
- Click **Define** to specify a component's details.

1.5.1 Defining components of objects

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

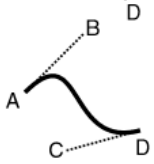
Aside from bitmaps and text, all component types are illustrated in the picture below.
Your objects should begin at the top-left point (0,0). Increasing x/y coordinates move to the right and down.

Arc



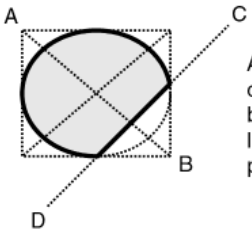
The arc is part of the ellipse bounded by the rectangle from A to B. Imaginary lines are drawn from the centre of the rectangle to C, and to D. The arc begins where these lines intersect the ellipse. It is drawn anticlockwise (in other words, if C and D were reversed, the opposite part of the ellipse would be drawn; see "Chord" for a drawn example).

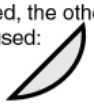
Bezier



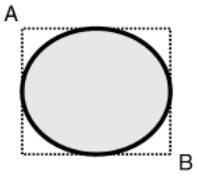
The Bezier spline is drawn from A to D.
Points B and C are "control points" that pull the curve towards them.

Chord



A chord is a solid figure created by the intersection of an ellipse and a straight line. The ellipse is bounded by the rectangle between A and B. C and D specify the line. (If C and D were reversed, the other part of the ellipse would be used:  .)

Ellipse

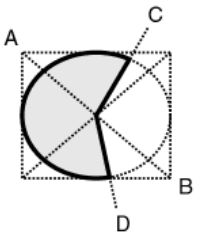


The ellipse is drawn within the rectangle bounded by A and B. The centre of the ellipse is at the centre of the rectangle.

Line

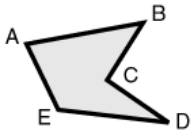
Not too complicated.

Pie



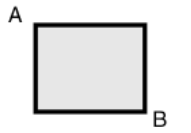
A pie is exactly like an arc but is a solid figure. (Again, it is drawn anticlockwise, and reversing C and D would cause the rest of the pie to be drawn instead.)

Polygon



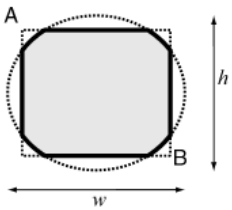
A polygon joins all the specified points, in order, completing the shape if necessary.
The fill mode is complicated. **Alternate:** the system fills the area between odd-numbered and even-numbered polygon sides on each scan line. That is, the system fills the area between the first and second side, between 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 PigTab 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.

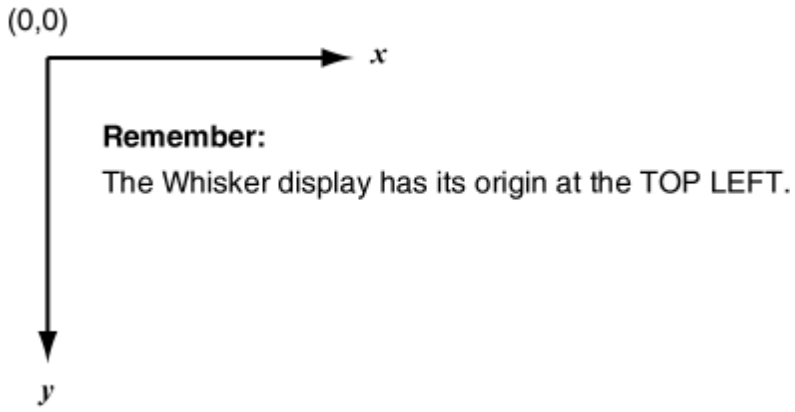
Defining the components

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
- Chord
- Ellipse
- Line
- Pie
- Polygon
- Rectangle
- Rounded rectangle (roundrect)
- Text

1.5.2 Size and coordinates

What coordinate system does PigTab use?



How do the tasks position objects on the screen?

All PigTab 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 they are using, assuming that the stimulus begins at (0,0) in its internal coordinate system (the system with which you define objects).

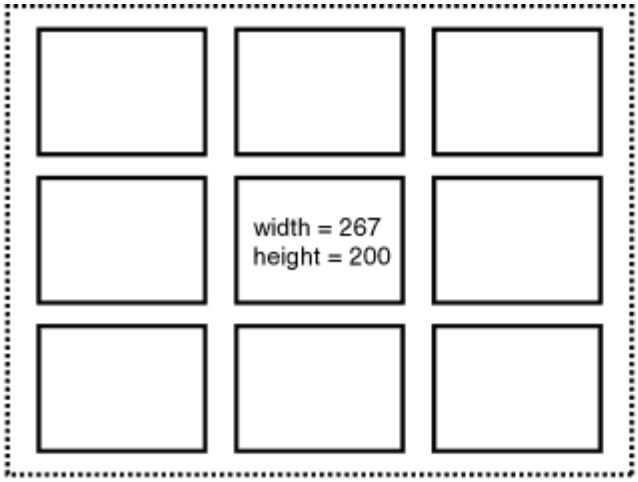
The tasks will have problems determining the size of text, and of bitmaps if you don't force the bitmap to a specified size. You should therefore test these stimuli before using them.

- Reinforcement Familiarization doesn't display any stimuli.
- Touch Training uses the nine-way grid. When one stimulus is displayed, it is shown in the

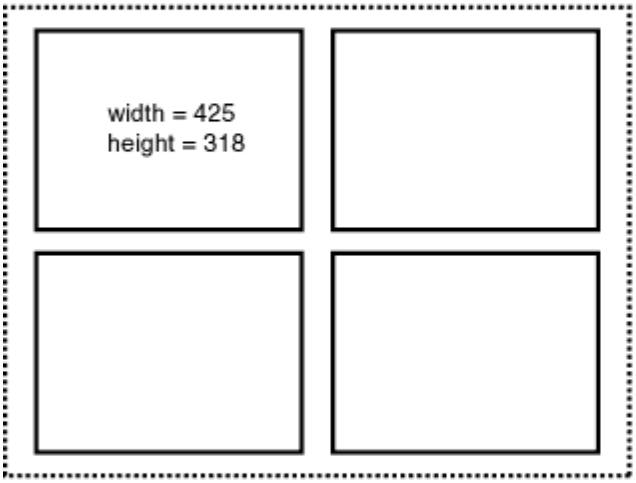
centre; when three are shown, they are position at the top centre (stimulus 1), bottom left (stimulus 2), and bottom right (stimulus 3).

- Visual Discrimination and Set Shifting uses the two-way grid.
- Delayed Matching/Non-matching To Sample uses centre of the nine-way grid for the presentation phase, and the four-way grid for the choice phase.
- Spatial Working Memory uses the nine-way grid.
- Three-Choice Serial Reaction Time uses the nine-way grid.
- Delayed Matching To Location uses the nine-way grid.
- Progressive Ratio Schedule uses the centre of the nine-way grid.

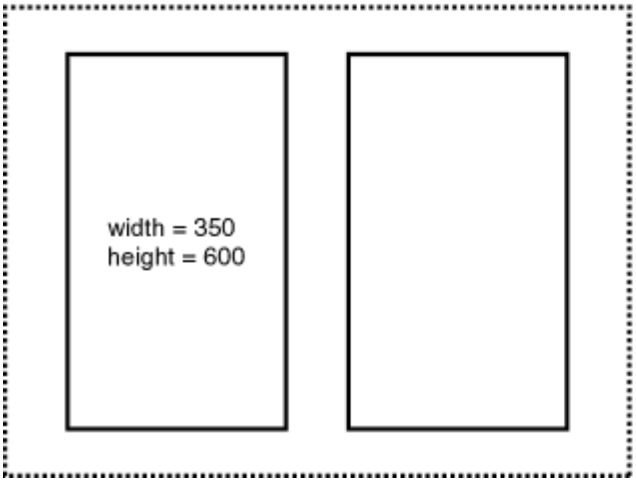
The sizes of these grids are shown below. For example, stimuli designed for use with the DMTL task should fit in a 267x200 rectangle.



Nine-way grid



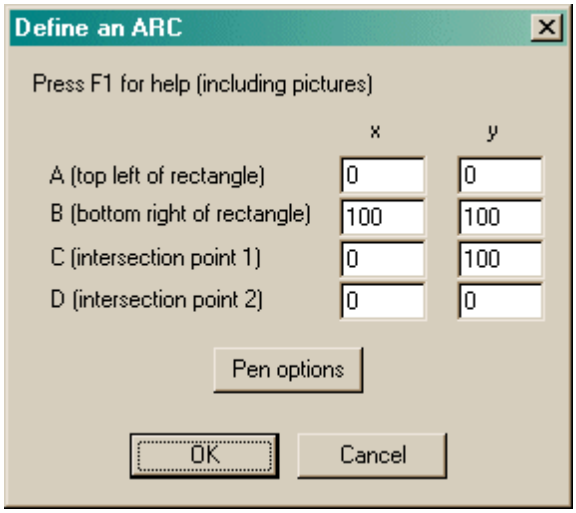
Four-way grid



whole screen
width = 1000
height = 750

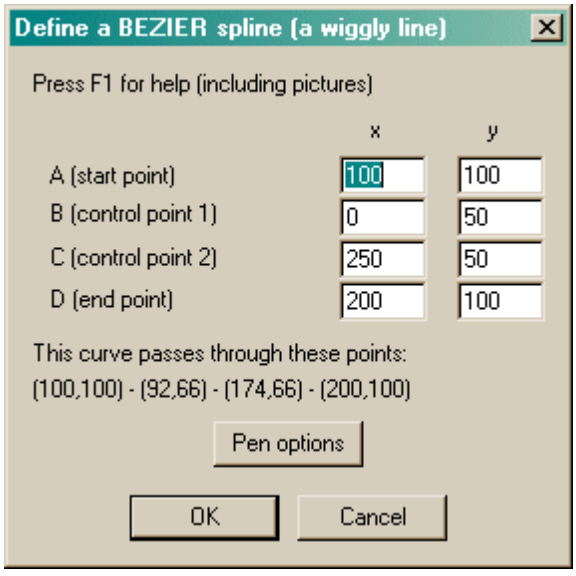
Two-way grid

1.5.3 Arc



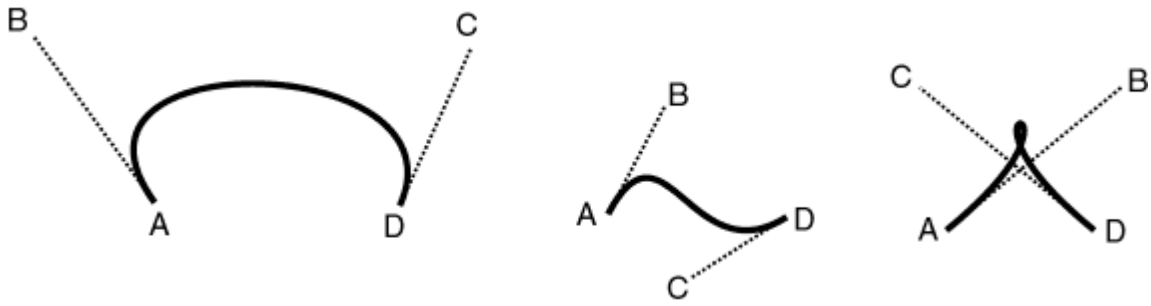
- The meaning of the points is explained in the figure above.
- Click Pen options to determine how the edge of the object is painted.

1.5.4 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

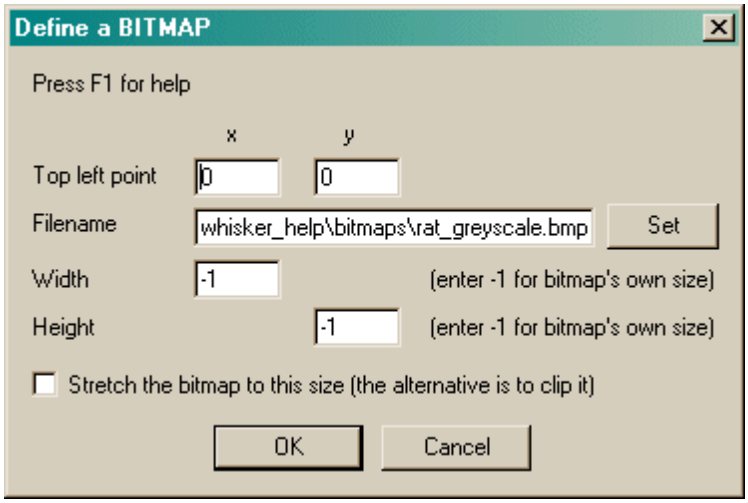
$$A = P$$
$$B = {}^{1}/_{6}(-5P + 18Q - 9R + 2S)$$
$$C = {}^{1}/_{6}(2P - 9Q + 18R - 5S)$$
$$D = S$$

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

Some properties of Bezier curves

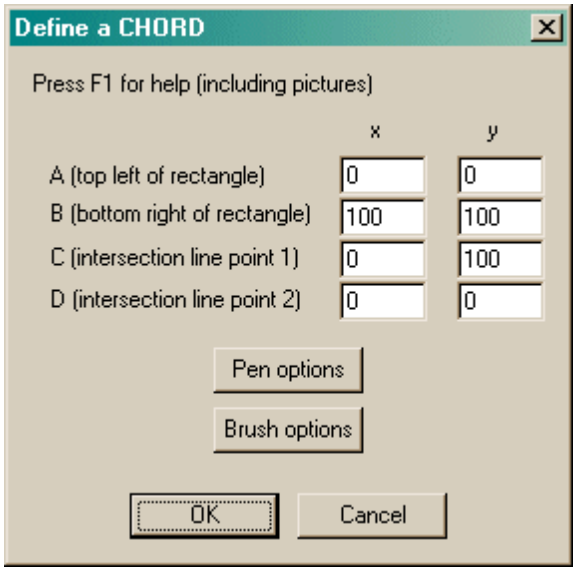
- Bezier curves always pass through their first and last points (A and D here), but not necessarily through their other control points (B and C).
- A Bezier curve always lies fully within the convex hull defined by its control points.
- The line AB has the same tangent at the curve at A, and the line 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).

1.5.5 Bitmap



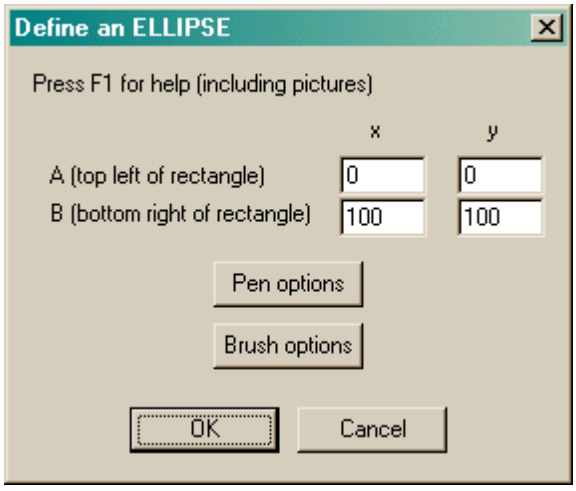
- Choose the **(x, y) coordinates of the top left point.** This is normally (0,0).
- Choose the **filename** of the bitmap. Click **Set** to browse for the file. If you are running PigTab on a different computer to WhiskerServer, remember that the filename must be accessible by the server, not the client.
- Choose a width and height to force the bitmap to, or leave the values at -1 to use the bitmap's intrinsic height.
- 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.

1.5.6 Chord



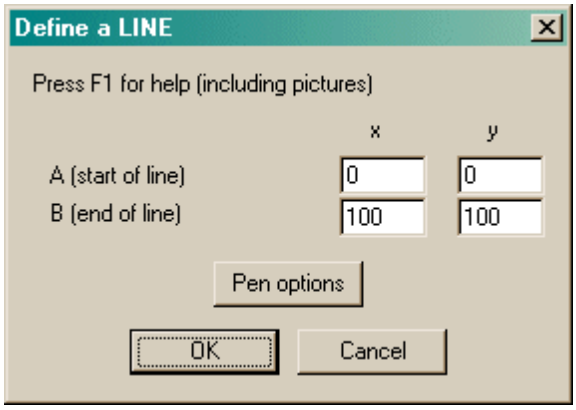
- The meaning of the points is explained in the figure above.
- Click Pen options to determine how the edge of the object is painted.
- Click Brush options to determine how the inside of the object is filled.

1.5.7 Ellipse



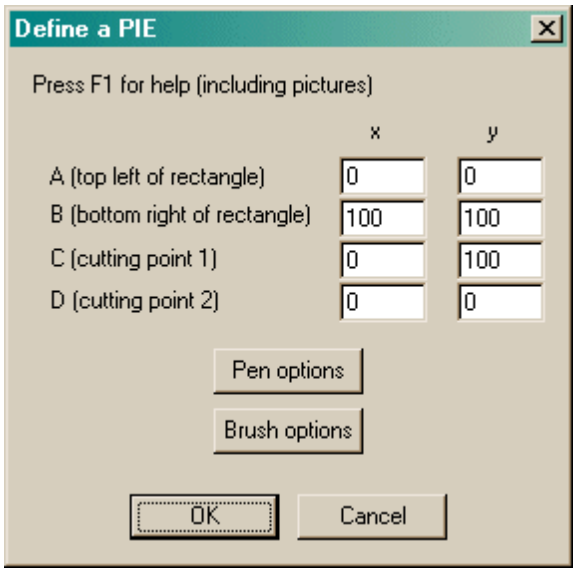
- The meaning of the points is explained in the figure above.
- Click Pen options to determine how the edge of the object is painted.
- Click Brush options to determine how the inside of the object is filled.

1.5.8 Line



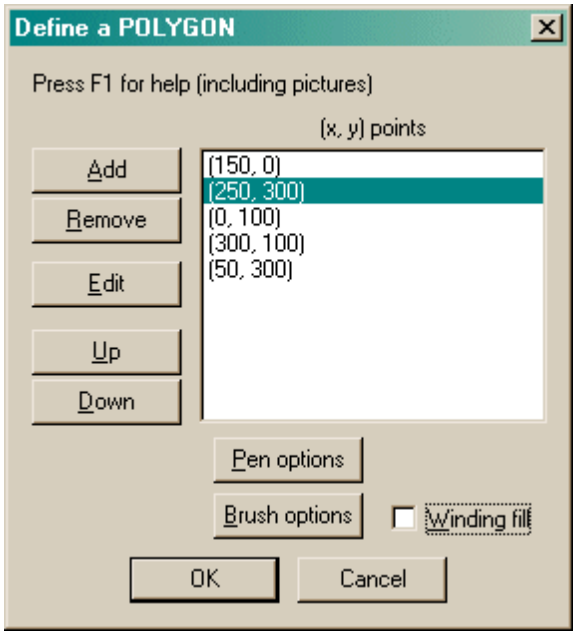
- The meaning of the points is explained in the figure above.
- Click Pen options to determine how the edge of the object is painted.

1.5.9 Pie



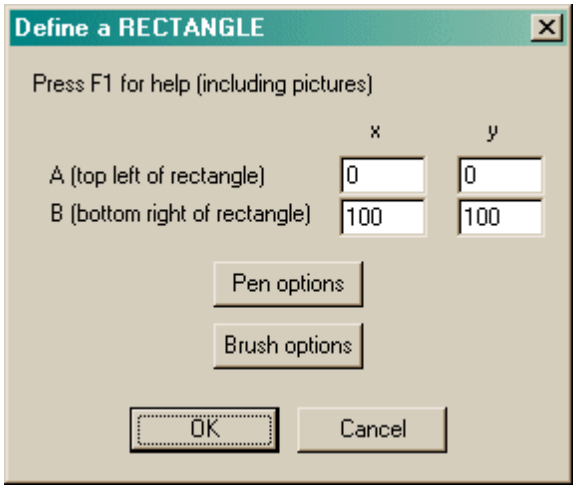
- The meaning of the points is explained in the figure above.
- Click Pen options to determine how the edge of the object is painted.

1.5.10 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.

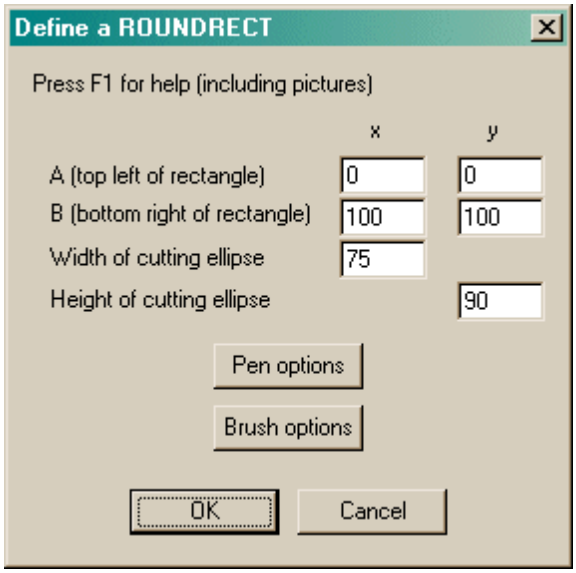
1.5.11 Rectangle



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

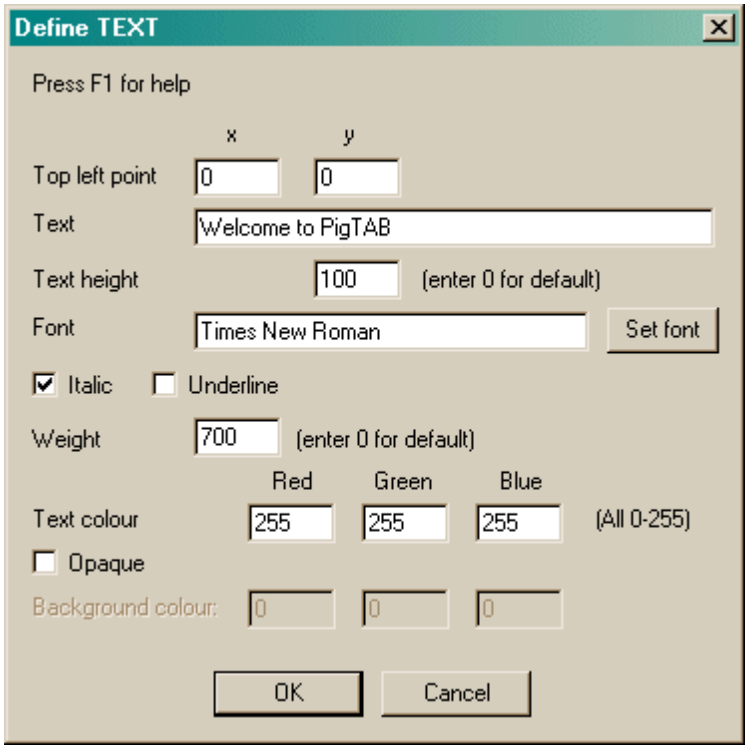
- Click Brush options to determine how the inside of the object is filled.

1.5.12 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.

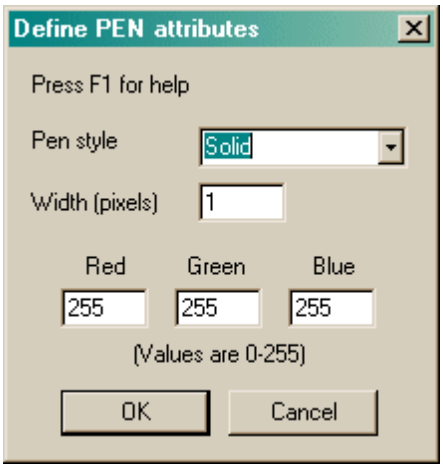
1.5.13 Text



- 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.

1.5.14 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.

1.5.15 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.

1.6 Configuring individual tasks

Choose a task to see how to configure it:

- Reinforcement Familiarization
- Touch Training
- Visual Discriminations and Set Shifting
- Delayed Matching/Non-matching to Sample
- Spatial Working Memory
- Three-Choice Serial Reaction Time
- Delayed Matching to Location
- Progressive Ratio Schedule

1.6.1 Reinforcement Familiarization

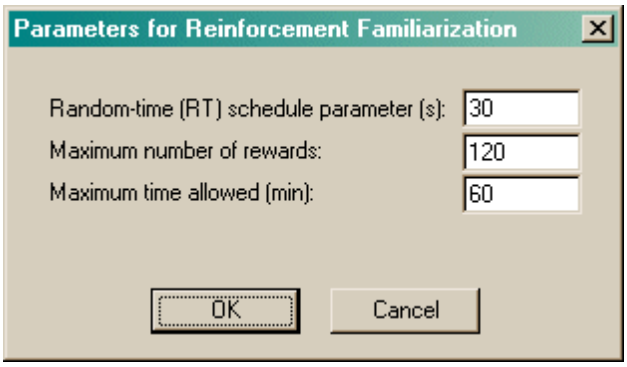
About the task

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.



- **Parameter of RT 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. (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.) 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.
- **Maximum number of rewards.** Once this number of rewards has been delivered, the task stops. (Specify 0 for no limit.)

- **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.

1.6.2 Touch Training

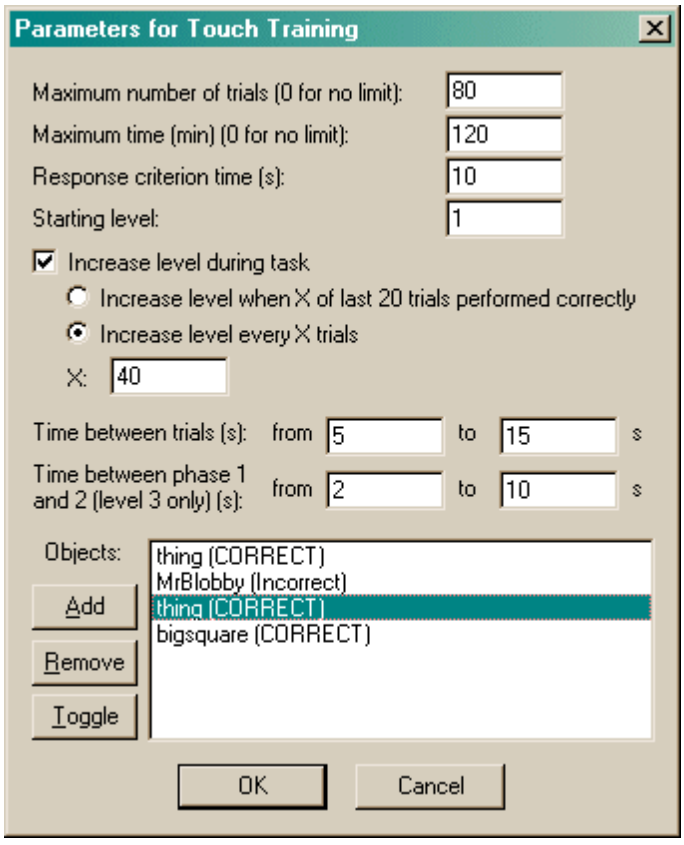
About the task

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. If the subject responds incorrectly or fails to respond, punish it.

- Level 1: correct response is touching any object
- Level 2: correct response is touching an object that is designated Correct
- Level 3 (sequential touching): trials have two phases. In phase 1, a single object is presented in the centre of the screen. If the subject touches it, we proceed to phase 2; otherwise, we punish. In phase 2, we play marker sound 2, and present *three other* objects; if the subject touches any one, it's rewarded (otherwise, punished).

Configuring the task



- **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.
- **Starting level.** The task will start at this level.
- **Increase level during task.** If enabled, the level will increase as the task goes on. You can either increase the level 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.
- **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.
- **Time between phase 1 and 2.** Only applies to level 3 (sequential touching). This sets the time between the subject responding to the phase 1 stimulus and the stimuli for phase 2 being presented.
- The **set of objects to use** is shown in the list at the bottom right. You may **add** objects from the library, **remove** them from the list, and **toggle** their status as being Correct or Incorrect (for Level 2, see above). You may add objects more than once. Objects are selected with equal probability from the list you see (so if you enter an object twice, you double its chance of being used; you can also have two copies of the same object where one is designated Correct and one is Incorrect, should you choose).

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

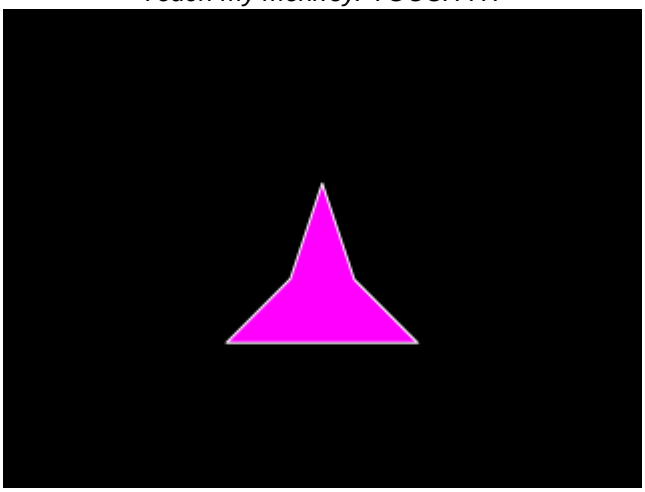
More on displaying objects

The task works with an internal scaling system based on a display that is 1000 units (pixels) wide and 750 high. Each object 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").) When one stimulus is displayed, it is shown in the centre; when three are shown, they are position at the top centre (stimulus 1), bottom left (stimulus 2), and bottom right (stimulus 3).

Screenshot from the task

Touch my monkey. TOUCH IT!



1.6.3 Visual Discriminations and 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, 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 "black 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(-)

Dimensional shifting in pigs

In PigTab, Sidse wants the possible dimensions to be **colour** (which incorporates a dimension of luminance, or light intensity) and **location** (four possibilities in two pairs: left/right, up/down).

Therefore, a simple discrimination would be between two colours, or two spatial locations. But remember that for simple discriminations, dimension B must be undefined. Let's assume A() = colour() and B() = location(). Therefore, we cannot present two objects of different colours simultaneously, because then dimension B would be defined and irrelevant, not undefined. It would be a compound discrimination. Therefore, to use location as a dimension means that we have to present stimuli *sequentially*, not simultaneously. And that brings a problem: if the subject has to choose which of two sequential stimuli is correct, how is it to succeed? We can't just present one, then the other, because it might decide that the first one was correct only when it's seen the second. (Note that if you were to adopt a variant of this technique, you'd have to randomize presentation order as another potential dimension.) We can't present them both, then both simultaneously (because then location is present as a dimension on test!). We could alternate them until it makes a response, but this would bring in problems of interpretation if subjects differed in response latency/speed (either at baseline, or as the result of an experimental intervention). A test using location as a dimension and/or alternating stimuli would also be rendered incomparable to previous human, primate, and rat testing methods, none of which have used location as a dimension.

My conclusion: using location as a dimension in the test either (1) makes the distinction between simple and compound discrimination impossible, or (2) introduces a variety of practical and interpretative problems in designing the test environment. Therefore, I shall implement PigTab using a different dimension. Location shall be randomized.

What's best as the two dimensions? Could we use shape and colour? We'd have to define "absence of colour" - plausibly a black fill colour.

As to the sequence, let's adopt the Birrell & Brown (2000) sequence, adding a simple reversal and removing the ID reversal and the ED reversal as requested by Sidse (or making them optional).

The sequence used by PigTab

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.

A technical issue: manufacturing compound stimuli

Compound stimuli can be designed by the user ("here's a shape; it has dimension attributes A3 and B4"), or assembled by the program (in some way, the user says "here are stimuli A1-A6 and B1-B6, please combine them as needed"). Now some attributes can only be created by the program (location would be one, but we're not going to use that). Which method is best? For shape/line combinations, it would seem more sensible for the program to assemble them. In the sequence shown above, the user would need to specify 12 stimuli (A1-6, B1-6) if the program does the combination, or 14 stimuli if the user does the combination (in total, 14 stimuli are shown in the sequence above).

That's not much of a difference, really. What other advantages do these two methods bring?

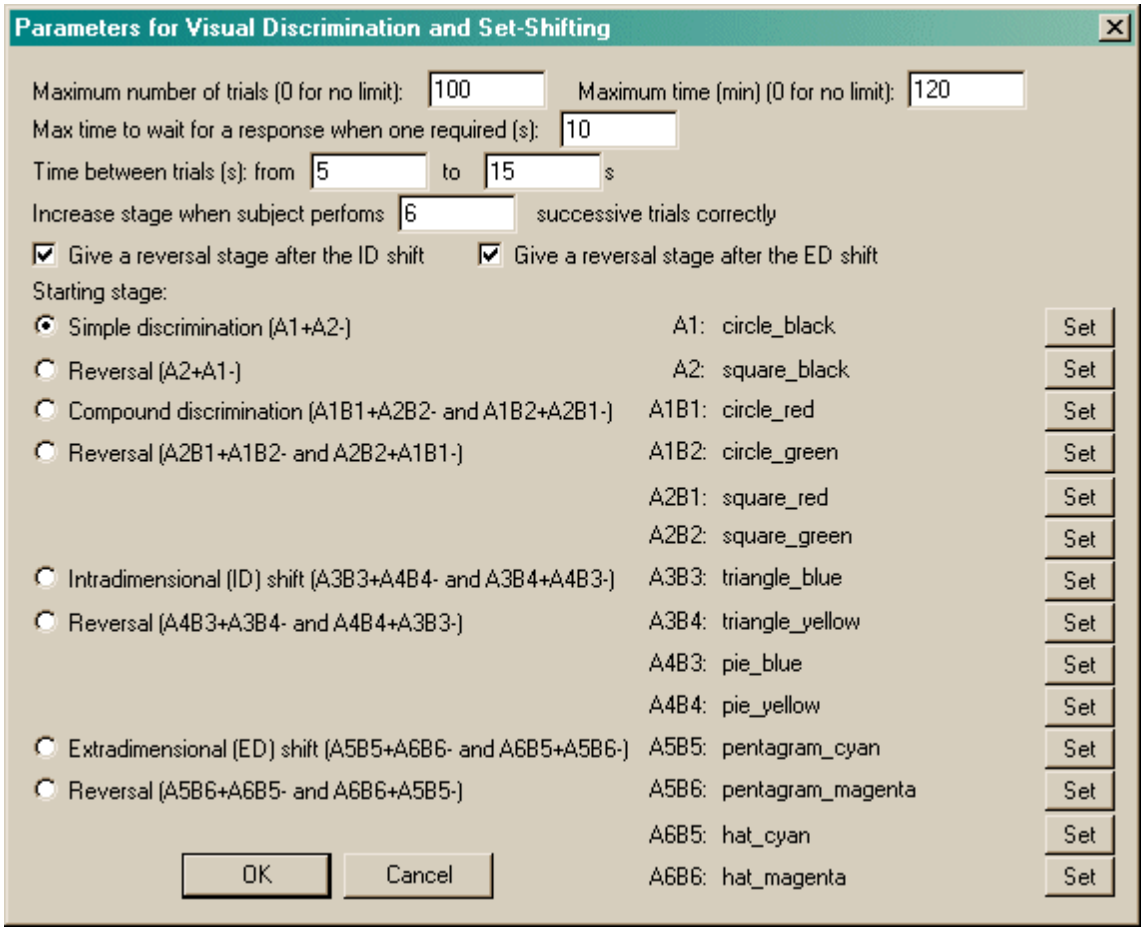
User generation means that the program could be used for any dimension conceived by the user - dimensions such as "which actor's in this photo" and "what colour house he's standing in front of", for example, which the program couldn't create.

Program generation means that the user has to generate fewer stimuli; this advantage becomes greater as the number of exemplars of each dimension increases. It means that dimensions are limited to things the program can generate. Superimposing visual stimuli is easy (and the basis of the program used by Dias et al., 1997). Changing the colour of a stimulus would be fairly easy (within PigTab's object generation system, this would mean the program iterating through all components of an object and forcing their brush fill settings to the chosen colour).

I think we should adopt user generation, as it's more flexible. For example, it's very easy to replace the stimulus set and perform a new dimensional shift assessment.

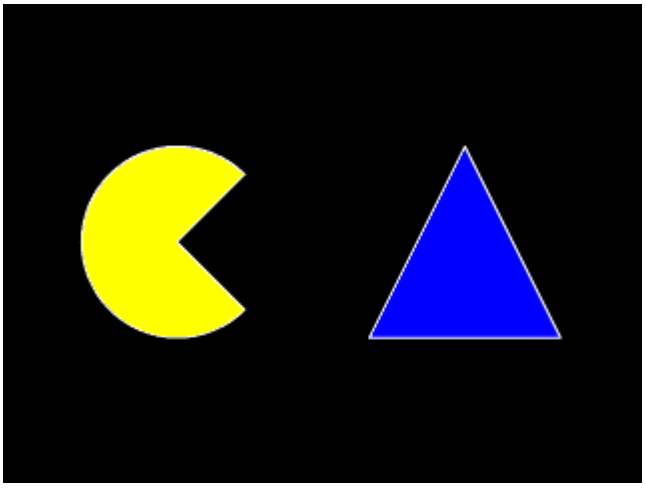
Configuring the task

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:



- **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.
- **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.
- **Increase stage when subject performs X successive trials correctly.** Fairly obvious, I hope. Set the value of X in the box.
- **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.
- **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.

Screenshots of the task



1.6.4 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 another object (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 until the subject gets them right. **RNC: I've implemented a one-off correction phase as used in DMTL (Sidse's PAL request); ask if she wants consistency across tasks or to repeat entire trials again and again.**

Choice of objects

- In primate testing, unique objects are normally used for each trial, so a D(N)MTS task requires a large library of pictures.
- Pigs are poorer at visual discrimination; therefore, Sidse wants to ensure that the test only uses objects known to be discriminable by pigs. She suggests (1) that the objects are combinations of one of 6 colours and one of 6 patterns; (2) that the target object is selected pseudorandomly, such that it is never an object that has been presented in the last 5 trials, unless the correction procedure is on; (3) that the distractor object is similarly selected, such that the distractor has not been presented in the last 5 trials, either as a target object or as a distractor.
- This constraint is implemented - but note that if you don't have enough stimuli for this purpose, this constraint will be violated (what else could the task do, after all?).

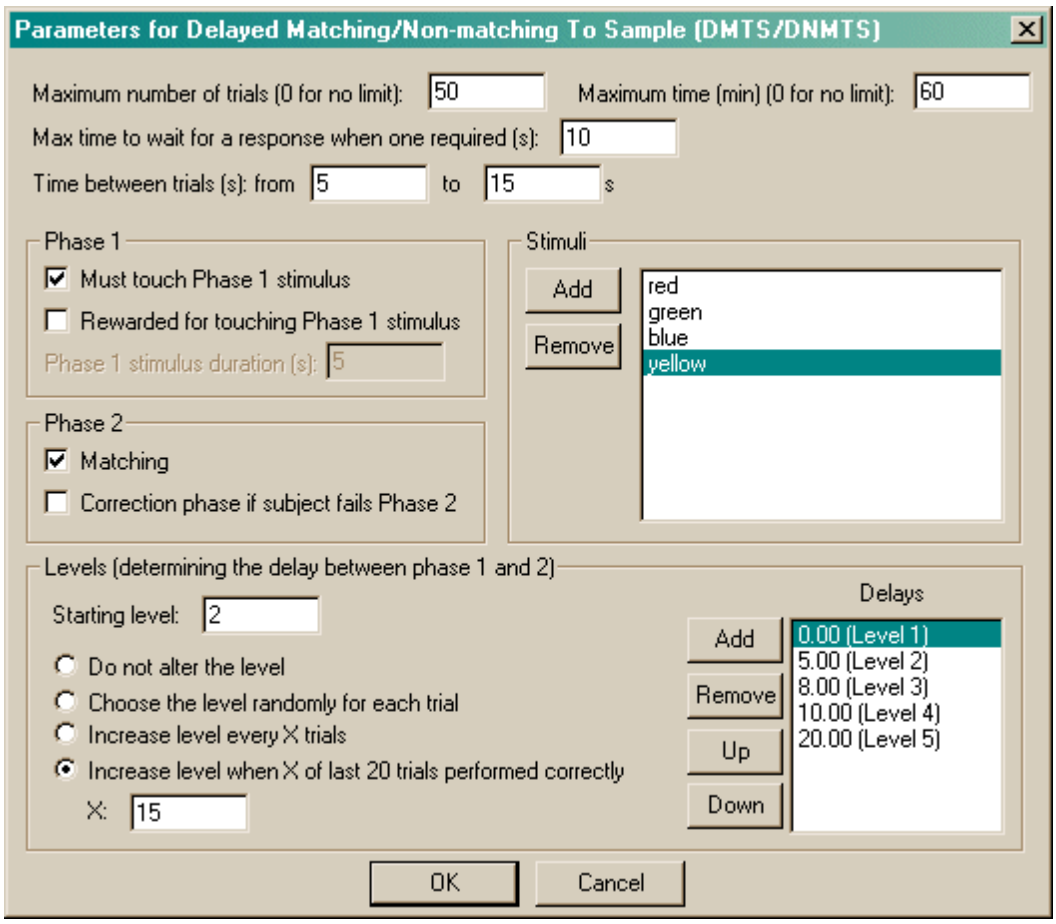
Levels

Levels are defined by the delay (option to enter these; typically, level 1 has a zero delay). The task can be run either:

- with the level chosen at random for each trial (the possible levels to be used are specified in a

list);
- with a fixed level;
- with a level specified at the start, but that increases when a criterion is met (the criterion being that X of the last 20 trials were performed correctly).

Configuring the task



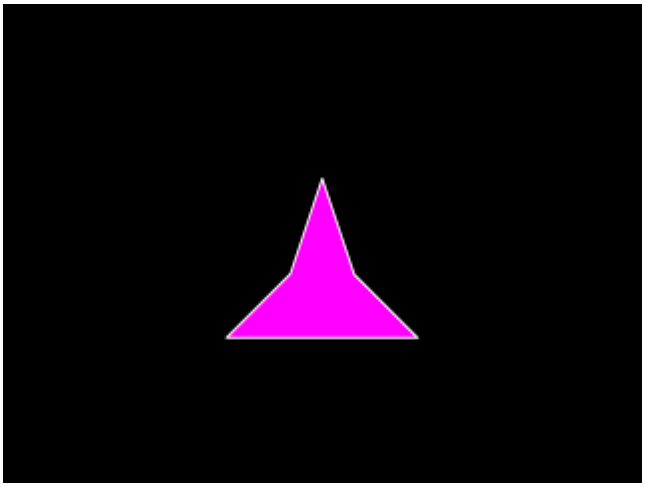
- **Maximum number of trials.** When the subject has performed this number of trials, the task ends. (You may specify 0 for no limit, though you must specify a limit on the number of trials, the time, or both.)
- **Maximum time.** When this time elapses, the task is terminated as soon as the current 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. (This time limit applies to Phase 1, if you require your subjects to touch the Phase 1 stimulus, and Phase 2, and the correction procedure if one is used.)
- **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.
- **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.** If you do not want your subject to have to touch the stimulus, you must specify the **Phase 1 stimulus duration** instead.
- **Matching.** If this is ticked, the task is delayed matching to sample. Otherwise, it's delayed non-matching to sample.
- **Correction phase if subject fails Phase 2.** Optionally, phase 2 can be repeated immediately (once) if the subject fails it the first time.

- **Stimuli.** This shows the list of available stimuli. You cannot put a stimulus into the list more than once. Click **Add** and **Remove** to add/remove stimuli.
- **Levels.** 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, 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.)

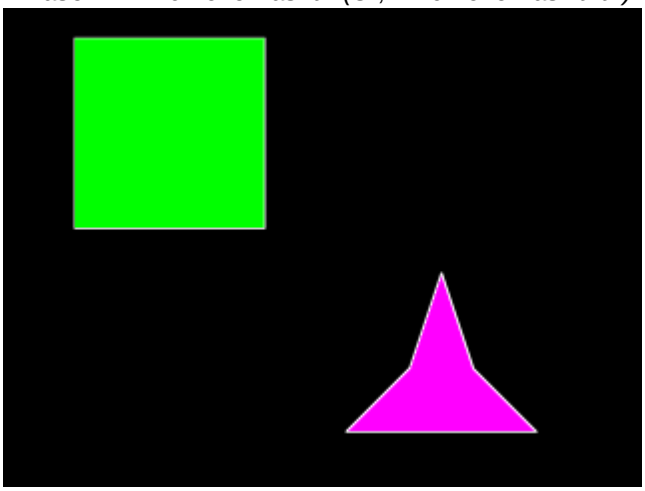
Options for reward and punishment are set in the General Parameters section; visual objects are defined in the Visual Object Library.

Screenshots from the task

Phase 1



Phase 2. Which one was it? (Or, which one wasn't it?)



1.6.5 Spatial Working Memory

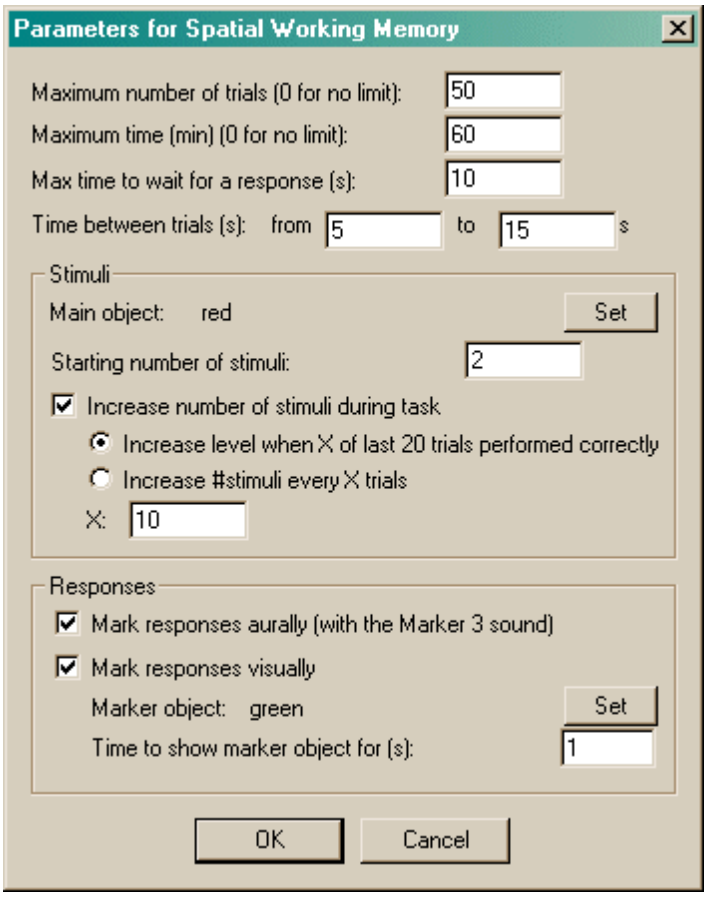
About the task

A number of identical objects are presented in different locations. (Four? Nine?) 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 ("response") object is presented for a defined duration. (In typical human testing, the primary objects are boxes and the response 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).

Configuring the task

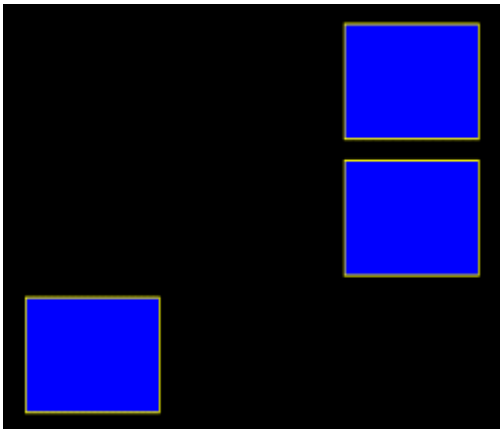


- **Maximum number of trials.** When the subject has performed this number of trials, the task ends. (You may specify 0 for no limit, though you must specify a limit on the number of trials, the time, or both.)
- **Maximum time.** When this time elapses, the task is terminated as soon as the current 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.
- **Main object.** This shows the object that will be used as the stimulus. Click **Set** to choose the stimulus.

- **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.
- **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.

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

Screenshot from the task



1.6.6 Three-Choice Serial Reaction Time

About the task

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

A Centre Object is presented in the centre of the screen, and the subject must touch it, and keep touching for a specified duration. (Optionally, it is rewarded for this.)

There is then a delay.

Then a different object is presented in one of three locations for a certain duration (typically brief) **[RNC: unclear whether SIdse wanted a choice from three objects, or just three locations; the latter is most akin to the rat 5CSRTT so I've gone with that]**. The subject must respond (within a timeout), and is rewarded for doing so. It's punished for missing or getting it wrong.

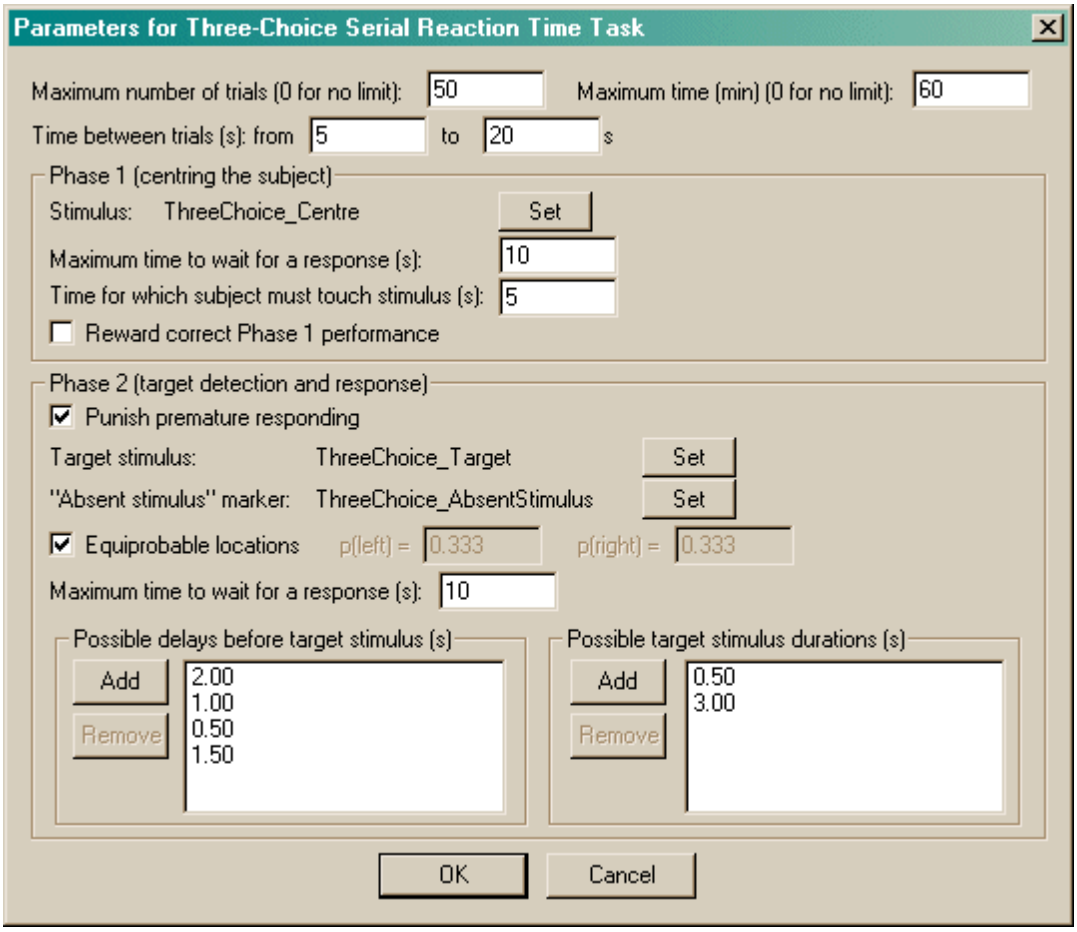
Sidse also wants to be able to choose whether the targets are randomly distributed among the three locations (33% - 33% - 33%) in the test phase, or are distributed with a different pre-specified distribution. **I'm not sure why, but it's in there. Think twice before using it, though!**

The requested level system was a bit complex. In any case, no matter what the level, the duration of the phase 1 object didn't vary. The two variables were (1) delay preceding the

target; (2) target duration. It's important in the 5CSRTT that the delay preceding the target can vary randomly; our rat version allows random variation of this delay but requires you to specify stimulus duration in advance. So I think we should add one level of flexibility (whilst not implementing Sidse's request exactly) and allow the user to specify a list of delays and a list of target durations, and then have the program choose from each.

Note that as the delay can be zero, the location of the centre object mustn't be the same as one of the targets (otherwise the pig would be touching the centre target as soon as it was displayed). So the centre target is placed below centre (with the intention that the target locations will be visible while the pig touches the centre marker).

Configuring the task



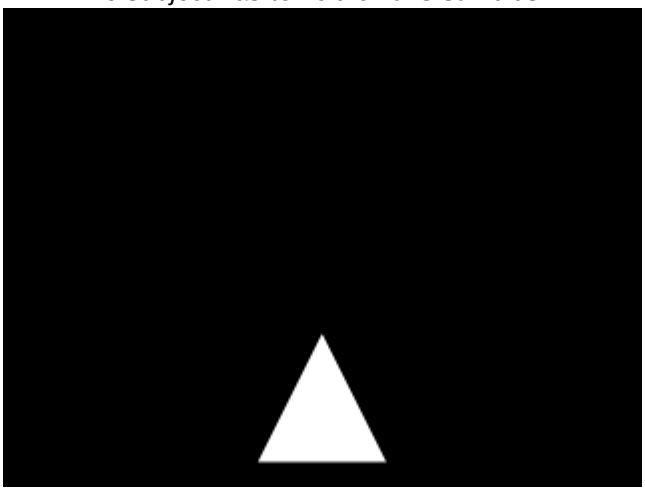
- **Maximum number of trials.** When the subject has performed this number of trials, the task ends. (You may specify 0 for no limit, though you must specify a limit on the number of trials, the time, or both.)
- **Maximum time.** When this time elapses, the task is terminated as soon as the current 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.
- **Phase 1 - Stimulus.** Choose the centring stimulus (the one that the subject has to keep touching for a while before the targets appear). Click **Set** to choose the stimulus.
- **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 touch stimulus.** Choose the length of time for which you want your subject to keep its nose on the centring stimulus.

- **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!**
- **Phase 2 - Punish premature responding.** If the subject responds in one of the three 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).
- **Phase 2 - Target stimulus.** Choose the target stimulus by clicking **Set.**
- **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 - Equiprobable locations?** If ticked, p(target on the left) = p(target in the middle) = p (target on the right) = 1/3. If you wish, you can force the location probabilities to something else by removing this tick and filling in the **p(left)** and **p(right)** boxes. Obviously, p(middle) = 1 - p (left) - p(right) in these circumstances.
- **Phase 2 - Maximum time to wait for a response.** This time is measured from the *start* of the target stimulus. If this time limit expires and the subject hasn't responded, it fails the trial.
- **Possible delays before target stimulus.** Use the **Add** and **Remove** buttons to create a list of possible delay values. Every trial, a delay is chosen at random from this list.
- **Possible target stimulus durations.** Use the **Add** and **Remove** buttons to create a list of possible stimulus duration values. Every trial, a stimulus duration is chosen at random from this list.

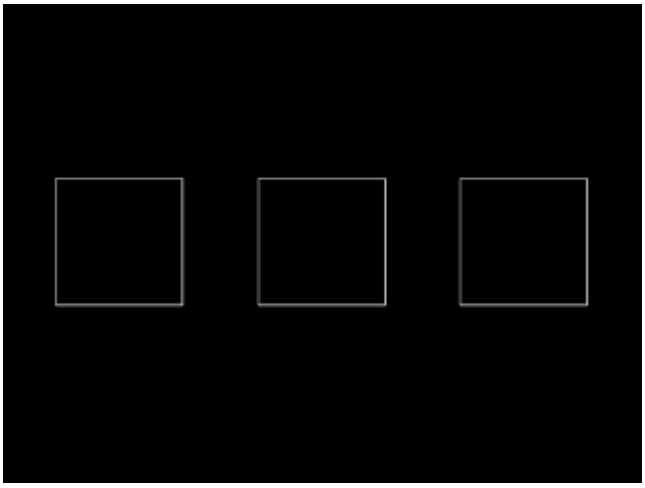
Options for reward and punishment are set in the General Parameters section; visual objects are defined in the Visual Object Library.

Screenshots from the task

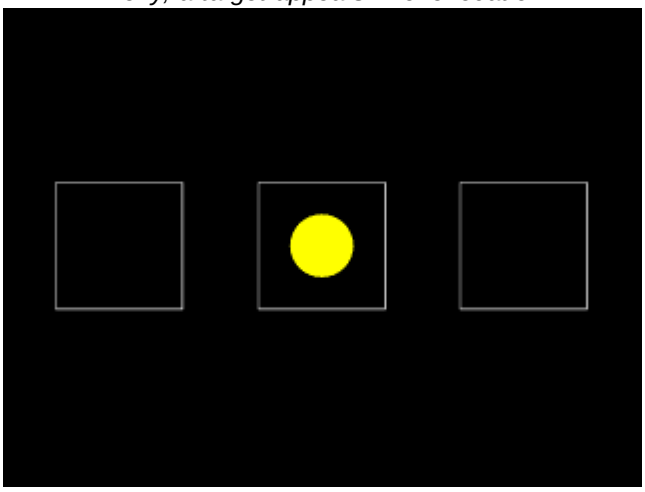
The subject has to hold on this stimulus...



... after which it monitors three locations.



Briefly, a target appears in one location.



1.6.7 Delayed Matching to Location

About the task

RNC comment: This was referred to by Sidse as "paired associates learning". I can see no reason to call it that. It doesn't seem to resemble the human CANTAB PAL test (though I haven't got a good description of that). I've called it "delayed matching to location (DMTL)".

I'm also not sure why we need to have a whole library of stimuli for this task. [Excerpt adapted from Sidse's wish-list: In Monkey-CANTAB, a unique object is used for each trial, so the task requires a large library of images. Pigs are poorer at visual discrimination; therefore, Sidse wants to ensure that the test only uses objects known to be discriminable by pigs. She suggests (1) that the objects are combinations of one of 6 colours and one of 6 patterns; (2) that the target object is selected pseudorandomly, such that it is never an object that has been presented in the last 10 trials.] **Object identity has no bearing on the task, so why not dedicate one object to this task (perhaps even as a discriminative stimulus indicating that "now we're doing DMTL") and just use that one? I can't think of an obvious reason why changing the task stimulus every task is important for DMTL. There are only two tasks in which the subject sees a whole array of identical stimuli - this and Spatial Working Memory, which is a one-stage task and different in several other ways. Probably not wise to use the same object for SpatialWM and DMTL, and other than that I can't see any problem.**

So we'll use one object for the moment. Simpler.

I seem to have a lot of queries about this task, so maybe I've misinterpreted the whole thing... in which case we'll have to do PAL, whatever that might be - and consider this a bonus task! :-)

Further information on human paired-associates learning (PAL) from Shibley Rahman: at the highest level, eight boxes are displayed around the edge of the screen. A number of stimuli (up to eight) are shown sequentially in different locations. In the test phase, a stimulus is shown in the centre and the subject must indicate which location it was in previously. It's therefore a delayed matching-sample-to-location test (the association under test is between stimulus and location).

This would fit with Sidse's comments re having lots of stimuli, but she says "... Showing an object in one OBL [meaning location] and then after a delay presenting this object again in all locations and the correct location has to be touched." That's not the same as human PAL as described by Shibley, who should know.

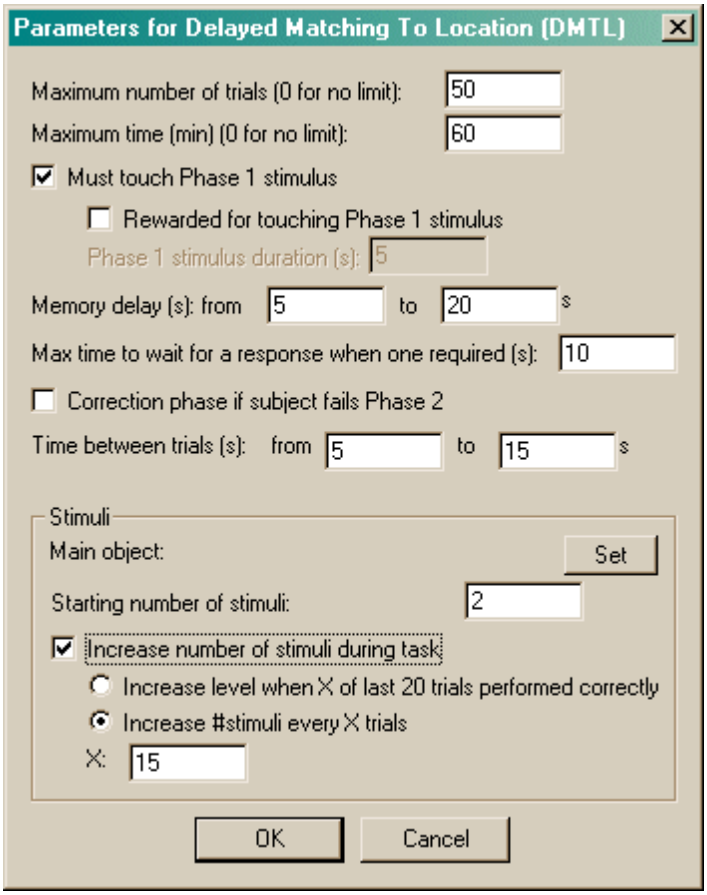
Mike agrees that DMTL only becomes PAL when you are presented >1 stimulus/location pair in the first phase, and then have to match them in a subsequent phase.

So we need to ask the Danes about this.

- Marker 1 signals the beginning of a trial.
- One object is shown, somewhere. (Optionally, the subject must touch this, in which case it is punished and the trial ends if it fails to respond. As a further option, it may be rewarded for touching it.)
- Then there's a "memory" delay. (Configurable.)
- Marker 2 signals end of delay.
- Several (1-4) copies of the original object are presented in random locations. Only one is in the same location as the original object. This one must be touched (rewarded).
- Timeouts/incorrect responses punished.

Levels: same as number of objects shown in second phase (1-4).
Option to repeat the second stage of the trial (once) if the subject gets it wrong. (We"ll present this repeat of phase 2 straight after it's been punished for getting it wrong in phase 2.) (Sidse requested an unspecified "correction procedure" in addition to this, but this is a correction procedure, so we'll leave it at this unless she requests something else.)

Configuring the task



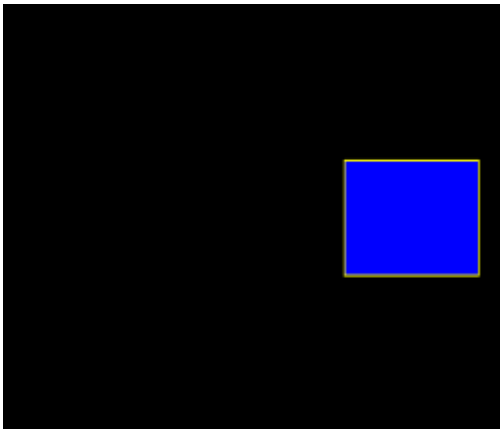
- **Maximum number of trials.** When the subject has performed this number of trials, the task ends. (You may specify 0 for no limit, though you must specify a limit on the number of trials, the time, or both.)
- **Maximum time.** When this time elapses, the task is terminated as soon as the current 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.)
- **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.** If you do not want your subject to have to touch the stimulus, you must specify the **Phase 1 stimulus duration** instead.
- **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 Phase 1, if you require your subjects to touch the Phase 1 stimulus, and Phase 2, and the correction procedure if one is used.)
- **Correction phase if subject fails Phase 2.** Optionally, phase 2 can be repeated immediately (once) if the subject fails it the first time.
- **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.
- **Main object.** This shows the object that will be used as the stimulus. Click **Set** to choose the stimulus.
- **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.

Options for reward and punishment are set in the General Parameters section; visual objects are

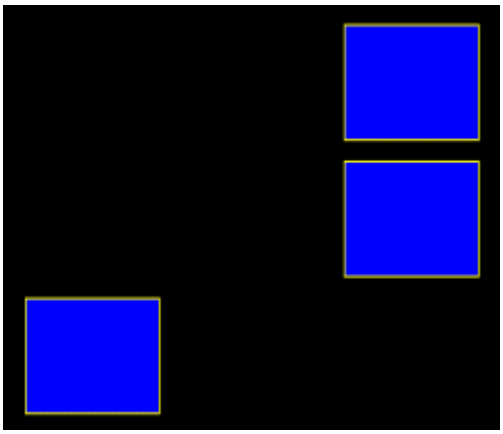
defined in the Visual Object Library.

Screenshots from the task

Phase 1



Phase 2



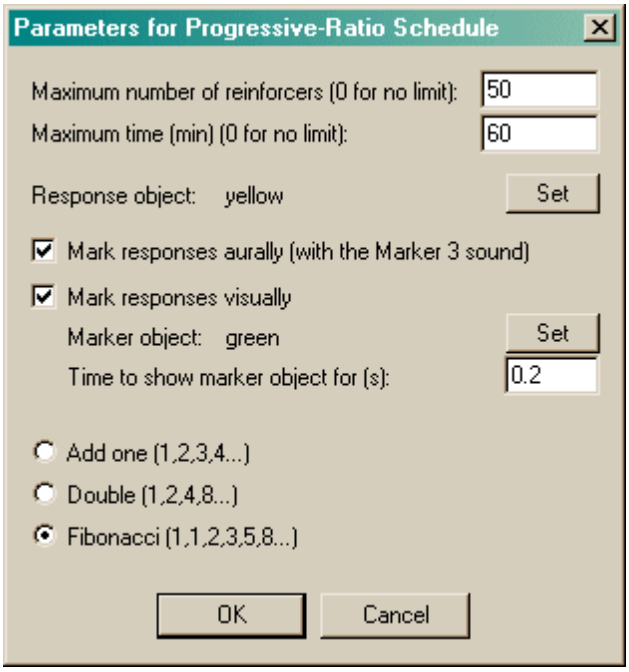
1.6.8 Progressive Ratio Schedule

About the task

Presents a single object. Reward touches under a progressive ratio schedule. Initial requirement 1 touch.

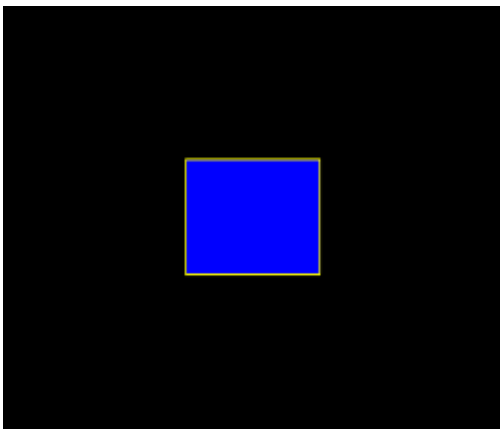
Configuring the task

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.
- **Progressive ratio requirement series.** Choose your series. A selection of techniques for incrementing the ratio requirement is shown.

Screenshot from the task



DEVELOPMENT NOTES

"Algorithm for progression to be entered: 1.25, 1.5, 1.75, 2"... suggest using a proper one. There's a power series technique that's quite popular and should be added.

Hodos, W (1961). Progressive ratio as a measure of reward strength. Science 134: 943-944. [Original progressive ratio ref?]

1.7 Before you start the task

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

1.8 Results

PigTab 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.)

1.8.1 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.

```
PIGTAB -- SUMMARY FILE
______________________

IDENTIFICATION

Subject:               fish
Session:               110
Date/time code:        28-Sep-2002 (19:19)
Comment:               testing
Summary file name:     fish-28Sep2002-1919-PigTab-summary.txt
Default ODBC database: PigTab

Box:                   0
Client computer name:  WILLOW
Server computer name:  needle

GENERAL PARAMETERS

Default media directory:                w:\whisker_test_media
Link - Duration (s):                    20
Link - Play sound during link?          Y
```

```
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 - Play sound?                      Y
Punishment - Darkness?                    Y
Punishment - Darkness time (s):           10
Punishment - Play sound?                  Y
Punishment - Deploy pigpusher?            N
Punishment - Pigpusher duration (s):      10
Sounds - Link sound is WAV?               N
Sounds - Link sound filename:             TestSnd.wav
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?             Y
Sounds - Reward sound filename:           TestSnd.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:
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:
Sounds - Marker1 sound frequency (Hz):    500
Sounds - Marker1 sound type:              Sine
Sounds - Marker1 sound duration (s):      1
Sounds - Marker1 sound level (0-100):     100
Sounds - Marker2 sound is WAV?            N
Sounds - Marker2 sound filename:
Sounds - Marker2 sound frequency (Hz):    2000
Sounds - Marker2 sound type:              Sawtooth
Sounds - Marker2 sound duration (s):      1
Sounds - Marker2 sound level (0-100):     100
Sounds - Marker3 sound is WAV?            N
Sounds - Marker3 sound filename:
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: yellow

polygon,polygon 3 50 0 0 100 100 100 -winding -penstyle solid -penwidth 1 -
pencolour 255 255 255 -brushsolid 255 255 255
text,text 150 0 "Welcome to PigTab" -height 20 -textcolour 255 255 255 -backcolour
0 0 0 -weight 700 -italic -font "Times New Roman"
line,line 0 0 100 100 -penstyle solid -penwidth 3 -pencolour 255 0 0
arc,arc 0 0 100 100 0 100 0 0 -penstyle solid -penwidth 3 -pencolour 255 0 0
bitmap,bitmap 0 0 "w:\whisker_help\bitmaps\rat_greyscale.bmp" -height 100 -width
100 -stretch
rect,rectangle 0 0 100 100 -penstyle solid -penwidth 1 -pencolour 255 255 255 -
brushsolid 255 255 255
roundrect,roundrect 0 0 100 100 75 90 -penstyle solid -penwidth 1 -pencolour 255
255 255 -brushsolid 255 255 255
```

```
bezier,bezier 100 100 0 50 250 50 200 100 -penstyle solid -penwidth 1 -pencolour
255 255 255

-- Object 1: red

roundrect,roundrect 0 0 100 100 75 90 -penstyle solid -penwidth 1 -pencolour 255
255 255 -brushsolid 255 0 0

-- Object 2: green

pie,pie 0 0 100 100 0 100 0 0 -penstyle solid -penwidth 1 -pencolour 255 255 255 -
brushsolid 0 255 0

-- Object 3: blue

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

-- Object 4: yellow0

polygon,polygon 3 50 0 0 100 100 100 -winding -penstyle solid -penwidth 1 -
pencolour 255 255 255 -brushsolid 255 255 255
text,text 150 0 "Welcome to PigTab" -height 20 -textcolour 255 255 255 -backcolour
0 0 0 -weight 700 -italic -font "Times New Roman"
line,line 0 0 100 100 -penstyle solid -penwidth 3 -pencolour 255 0 0
arc,arc 0 0 100 100 0 100 0 0 -penstyle solid -penwidth 3 -pencolour 255 0 0
bitmap,bitmap 0 0 "w:\whisker_help\bitmaps\rat_greyscale.bmp" -height 100 -width
100 -stretch
rect,rectangle 0 0 100 100 -penstyle solid -penwidth 1 -pencolour 255 255 255 -
brushsolid 255 255 255
roundrect,roundrect 0 0 100 100 75 90 -penstyle solid -penwidth 1 -pencolour 255
255 255 -brushsolid 255 255 255
bezier,bezier 100 100 0 50 250 50 200 100 -penstyle solid -penwidth 1 -pencolour
255 255 255

-- Object 5: circle_black

circle_black,ellipse 0 0 300 300 -penstyle solid -penwidth 3 -pencolour 255 255
255 -brushsolid 0 0 0
background_rectangle,rectangle 0 0 300 300 -penstyle null -penwidth 1 -pencolour
255 255 255 -brushsolid 0 0 0

-- Object 6: square_black

square_black,rectangle 0 0 300 300 -penstyle solid -penwidth 3 -pencolour 255 255
255 -brushsolid 0 0 0

-- Object 7: circle_red

circle_red,ellipse 0 0 300 300 -penstyle solid -penwidth 3 -pencolour 255 255 255
-brushsolid 255 0 0
background_rectangle,rectangle 0 0 300 300 -penstyle null -penwidth 1 -pencolour
255 255 255 -brushsolid 0 0 0

-- Object 8: circle_green

circle_green,ellipse 0 0 300 300 -penstyle solid -penwidth 3 -pencolour 255 255
255 -brushsolid 0 255 0
background_rectangle,rectangle 0 0 300 300 -penstyle null -penwidth 1 -pencolour
255 255 255 -brushsolid 0 0 0

-- Object 9: square_red
```

```
square_red,rectangle 0 0 300 300 -penstyle solid -penwidth 3 -pencolour 255 255
255 -brushsolid 255 0 0

-- Object 10: square_green

square_green,rectangle 0 0 300 300 -penstyle solid -penwidth 3 -pencolour 255 255
255 -brushsolid 0 255 0

-- Object 11: triangle_blue

triangle_blue,polygon 3 150 0 0 300 300 300 -winding -penstyle solid -penwidth 3 -
pencolour 255 255 255 -brushsolid 0 0 255
background_rectangle,rectangle 0 0 300 300 -penstyle null -penwidth 1 -pencolour
255 255 255 -brushsolid 0 0 0

-- Object 12: triangle_yellow

triangle_yellow,polygon 3 150 0 0 300 300 300 -winding -penstyle solid -penwidth 3
-pencolour 255 255 255 -brushsolid 255 255 0
background_rectangle,rectangle 0 0 300 300 -penstyle null -penwidth 1 -pencolour
255 255 255 -brushsolid 0 0 0

-- Object 13: pie_blue

pie_blue,pie 0 0 300 300 300 0 300 300 -penstyle solid -penwidth 3 -pencolour 255
255 255 -brushsolid 0 0 255
background_rectangle,rectangle 0 0 300 300 -penstyle null -penwidth 1 -pencolour
255 255 255 -brushsolid 0 0 0

-- Object 14: pie_yellow

pie_yellow,pie 0 0 300 300 300 0 300 300 -penstyle solid -penwidth 3 -pencolour
255 255 255 -brushsolid 255 255 0
background_rectangle,rectangle 0 0 300 300 -penstyle null -penwidth 1 -pencolour
255 255 255 -brushsolid 0 0 0

-- Object 15: pentagram_cyan

pentagram_cyan,polygon 5 150 0 250 300 0 100 300 100 50 300 -winding -penstyle
solid -penwidth 3 -pencolour 255 255 255 -brushsolid 0 255 255
background_rectangle,rectangle 0 0 300 300 -penstyle null -penwidth 1 -pencolour
255 255 255 -brushsolid 0 0 0

-- Object 16: pentagram_magenta

pentagram_magenta,polygon 5 150 0 250 300 0 100 300 100 50 300 -winding -penstyle
solid -penwidth 3 -pencolour 255 255 255 -brushsolid 255 0 255
background_rectangle,rectangle 0 0 300 300 -penstyle null -penwidth 1 -pencolour
255 255 255 -brushsolid 0 0 0

-- Object 17: hat_cyan

hat_cyan,polygon 5 150 50 200 200 300 300 0 300 100 200 -winding -penstyle solid -
penwidth 3 -pencolour 255 255 255 -brushsolid 0 255 255
background_rectangle,rectangle 0 0 300 300 -penstyle null -penwidth 1 -pencolour
255 255 255 -brushsolid 0 0 0

-- Object 18: hat_magenta

hat_magenta,polygon 5 150 50 200 200 300 300 0 300 100 200 -winding -penstyle
solid -penwidth 3 -pencolour 255 255 255 -brushsolid 255 0 255
```

```
background_rectangle,rectangle 0 0 300 300 -penstyle null -penwidth 1 -pencolour
255 255 255 -brushsolid 0 0 0

-- Object 19: ThreeChoice_Centre

white_triangle,polygon 3 100 0 200 200 0 200 -winding -penstyle solid -penwidth 1
-pencolour 255 255 255 -brushsolid 255 255 255
background_rectangle,rectangle 0 0 200 200 -penstyle null -penwidth 1 -pencolour
255 255 255 -brushsolid 0 0 0

-- Object 20: ThreeChoice_AbsentStimulus

rectangle,rectangle 0 0 200 200 -penstyle solid -penwidth 3 -pencolour 255 255 255
-brushsolid 0 0 0

-- Object 21: ThreeChoice_Target

target_circle,ellipse 50 50 150 150 -penstyle null -penwidth 1 -pencolour 255 255
255 -brushsolid 255 255 0
rectangle,rectangle 0 0 200 200 -penstyle solid -penwidth 3 -pencolour 255 255 255
-brushsolid 0 0 0

CONFIGURATIONS

-- Task 0: ThreeChoice

Maximum no. trials (0 for no limit):                10
Max. time (min) (0 for no limit):                   4
Phase 1 stimulus (centring object):                 ThreeChoice_Centre
Max. time for subject to respond in phase 1:        10
Hold time (time that subject must stay on stim.):   5
Reward subject for phase 1?                         N
Punish premature responding?                        Y
Phase 2 stimulus (target):                          ThreeChoice_Target
Phase 2 location marker (absence of target):        ThreeChoice_AbsentStimulus
All three target locations equiprobable?            Y
... if not, p(left):                                0.333
... if not, p(right):                               0.333
Max. time for subject to respond in phase 2:        10
Minimum intertrial time (s):                        5
Maximum intertrial time (s):                        5
Possible delays before target appearance:
2.000000;1.000000;0.500000;1.500000
Possible target durations:                          0.500000;3.000000

-- Task 1: DMTS

Matching? (Alternative is non-matching.)            N
Maximum no. trials (0 for no limit):                10
Max. time (min) (0 for no limit):                   4
Must touch phase 1 stimulus:                        Y
Maximum time for responding (s) (for phases 1 + 2): 10
Phase 1 stimulus duration if no touch required:     5
Rewarded for touching phase 1 stimulus:             Y
Correction phase if subject fails phase 2:          Y
Starting level:                                     1
Method of changing level:                           Level_random
... criterion (X), if applicable:                   15
Minimum intertrial time (s):                        5
Maximum intertrial time (s):                        5
Number of stimuli available:                        2
Stimuli available:                                  square_green;hat_magenta
Number of levels (values of the memory delay):      5
```

```
Level delay values:
0.000000;1.000000;2.000000;3.000000;4.000000

-- Task 2: VisualDiscrim

Maximum no. trials (0 for no limit):                    100
Max. time (min) (0 for no limit):                       120
Stimulus A1:                                            circle_black
Stimulus A2:                                            square_black
Stimulus A1B1:                                          circle_red
Stimulus A1B2:                                          circle_green
Stimulus A2B1:                                          square_red
Stimulus A2B2:                                          square_green
Stimulus A3B3:                                          triangle_blue
Stimulus A3B4:                                          triangle_yellow
Stimulus A4B3:                                          pie_blue
Stimulus A4B4:                                          pie_yellow
Stimulus A5B5:                                          pentagram_cyan
Stimulus A5B6:                                          pentagram_magenta
Stimulus A6B5:                                          hat_cyan
Stimulus A6B6:                                          hat_magenta
Maximum time for responding (s):                        10
Starting stage:                                         Intradimensional (ID)
shift
Increase stage after how many sequential correct trials: 6
Give ID reversal stage?                                 Y
Give ED reversal stage?                                 Y
Minimum intertrial time (s):                            5
Maximum intertrial time (s):                            15

-- Task 3: DMTL

Maximum no. trials (0 for no limit):                    10
Max. time (min) (0 for no limit):                       3
Main object:                                            yellow
Must touch phase 1 stimulus:                            Y
Maximum time for responding (s) (for phases 1 + 2):     10
Phase 1 stimulus duration if no touch required:         5
Rewarded for touching phase 1 stimulus:                 N
Minimum memory delay (s):                               2
Maximum memory delay (s):                               6
Correction phase if subject fails phase 2:              Y
Starting number of stimuli:                             2
Increase number of stimuli?                             Y
... regardless of performance?                          Y
... criterion (every X trials or if X/last 20 correct): 3
Minimum intertrial time (s):                            5
Maximum intertrial time (s):                            15

-- Task 4: SpatialWM

Maximum no. trials (0 for no limit):                    10
Max. time (min) (0 for no limit):                       3
Main object:                                            red
Mark responses aurally?                                 Y
Mark responses visually?                                Y
Visual marker object:                                   green
Visual marker time (s):                                 0.1
Starting number of stimuli:                             2
Increase number of stimuli?                             Y
... regardless of performance?                          N
... criterion (every X trials or if X/last 20 correct): 2
Maximum time for responding (s):                        5
Minimum intertrial time (s):                            2
```

```
Maximum intertrial time (s):                              3

-- Task 5: ProgressiveRatio

Maximum no. reinforcers: 4
Maximum time (min):      5
Manipulandum object:     yellow
Mark responses aurally?  Y
Mark responses visually? Y
Visual marker object:    green
Visual marking time (s): 0.2
Ratio progression type:  Fibonacci

-- Task 6: TouchTraining

Response criterion time (s):                              10
Starting level:                                           2
Increase level?                                           Y
... regardless of performance?                            Y
... criterion (every X trials or if X/last 20 correct): 3
Maximum no. trials:                                       10
Maximum time (min):                                       5
Minimum intertrial time (s):                              5
Maximum intertrial time (s):                              10
Min inter-phase pause (sequential level):                 2
Max inter-phase pause (sequential level):                 5
Number of objects used:                                   4
...  Object 0 (name, correct?): yellow, Y
...  Object 1 (name, correct?): green, N
...  Object 2 (name, correct?): yellow, Y
...  Object 3 (name, correct?): red, Y

-- Task 7: ReinforcementFamiliarization

Parameter of RT schedule (s): 2
Maximum number of rewards:    5
Maximum time (min):           2

RESULTS

*** TASK ABORTED (cancelled by user or contact with server lost)
Emergency save of outstanding results:

-- Results for Task 0: Three-Choice Serial Reaction Time

Trial,TrialStartTimeMs,Phase1Responded,Phase1ResponseLatencyMs,Phase1Succeeded,
Phase1TimeHeldForMs,Phase1Rewarded,Phase1Punished,DelayStartTimeMs,DelayTimeMs,
Premature,FirstPrematureLocation,FirstPrematureResponseTimeMs,PrematurePunished,
Phase2Given,Phase2StartTimeMs,Phase2StimulusDurationMs,Phase2OfferedLocation,
Phase2ChosenLocation,Phase2Responded,Phase2RespondedCorrectly,
Phase2ResponseLatencyMs,Phase2Rewarded,Phase2Punished,ITITimeMs
0,24040979,Y,5808,N,701,N,Y,0,0,N,-1,0,N,N,0,0,-1,-1,N,N,0,N,N,5000
1,24062509,Y,2865,N,3194,N,Y,0,0,N,-1,0,N,N,0,0,-1,-1,N,N,0,N,N,5000
2,24083600,Y,621,Y,5007,N,N,24089238,1000,N,-1,0,N,Y,24090249,3000,2,2,Y,Y,1442,Y,
N,5000
3,24097730,N,0,N,0,N,N,0,0,N,-1,0,N,N,0,0,-1,-1,N,N,0,N,N,0

PIGTAB FINISHED

Started at:        28-Sep-2002 (19:20)
Finished at:       28-Sep-2002 (19:21)

Successfully wrote to database: ODBC;DSN=PigTab;DBQ=D:\Whisker\CODE\clients\RNC -
```

```
Copenhagen\PigTab\PigTab database (sample).mdb;DriverId=281;FIL=MS Access;
MaxBufferSize=2048;PageTimeout=5;
```

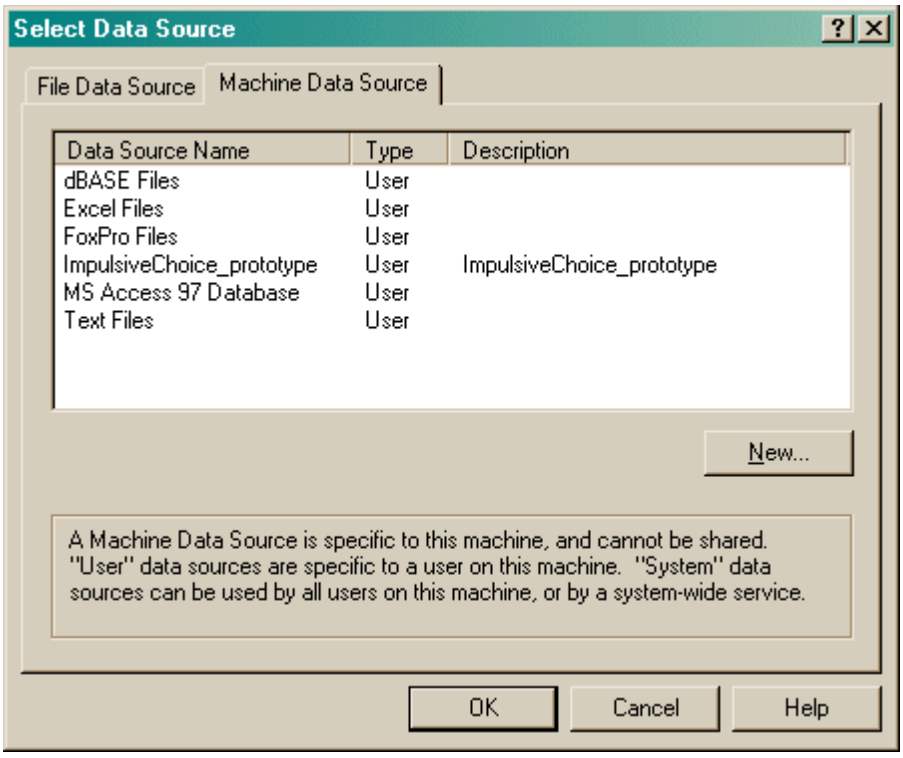
1.8.2 Creating a new ODBC source

What happens if you're using PigTab for the first time, and need to set up an ODBC (Open Database Connectivity) source for PigTab? You should configure it via **Control Panel** → **ODBC** [in Windows 2000, **Control Panel** → **Adminstrative Tools** → **Data Sources (ODBC)**]. Alternatively, you can set one up "on the fly", as explained here.

The example below is for a PIT (Pavlovian-instrumental transfer) database, as I couldn't be bothered to redo all the screenshots for a PigTab database. However, every step is identical! Just choose your copy of the PigTab database instead.

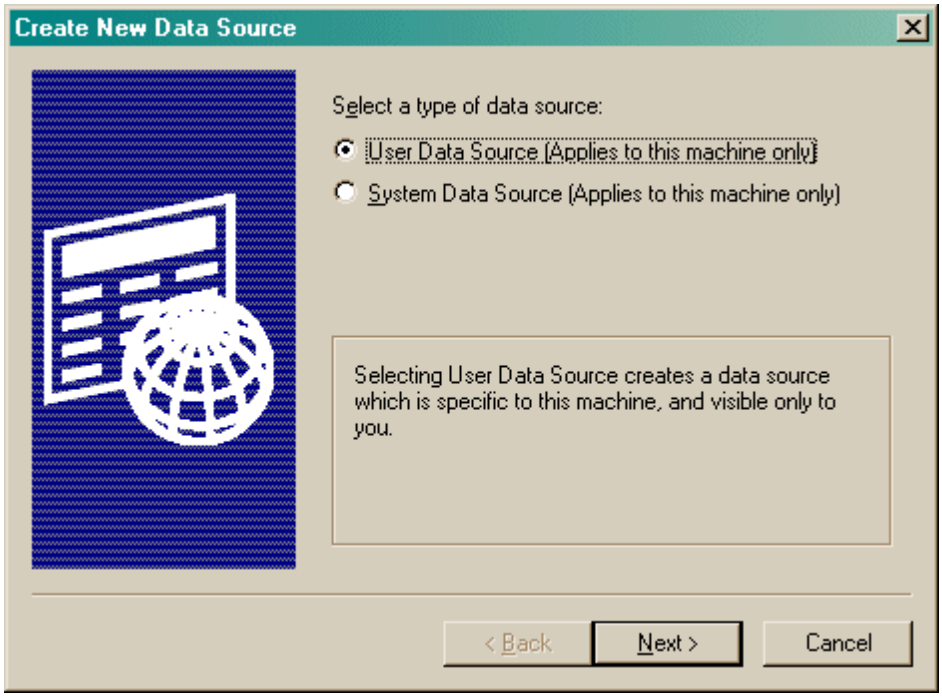
Remember: you shouldn't use the supplied database without making a copy for yourself. (It will work, but if you ever uninstalled or reinstalled PigTab, this file might be replaced or lost. It is much safer to make your own copy and set up ODBC to use your copy.)

Suppose you're looking for a PIT database. But there isn't one...

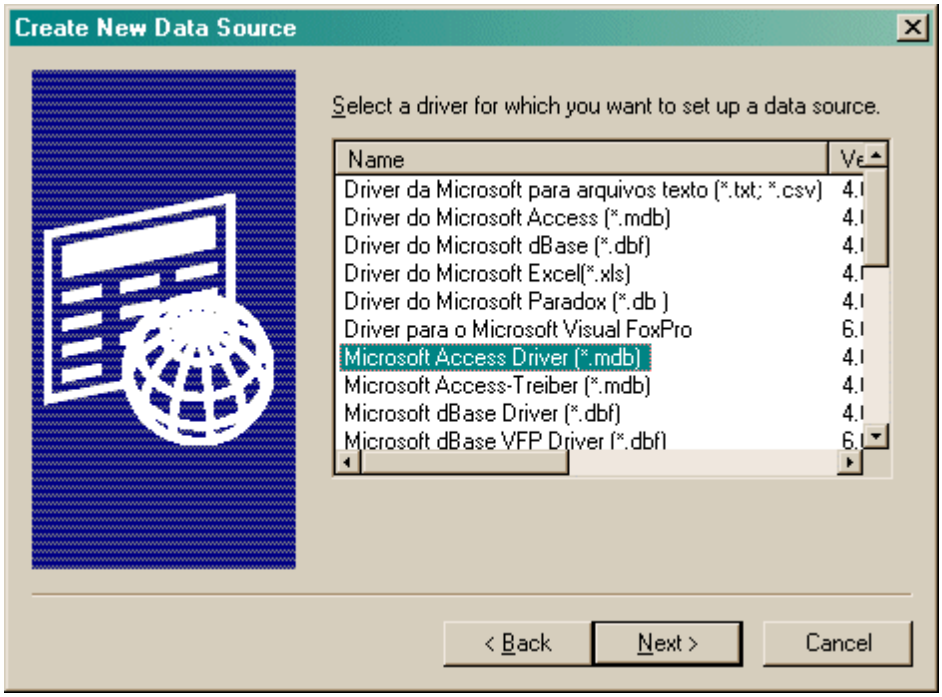


Let's assume that you have **already made a working copy of the prototype database supplied with the task.** How do we go about setting this up as an ODBC data source?

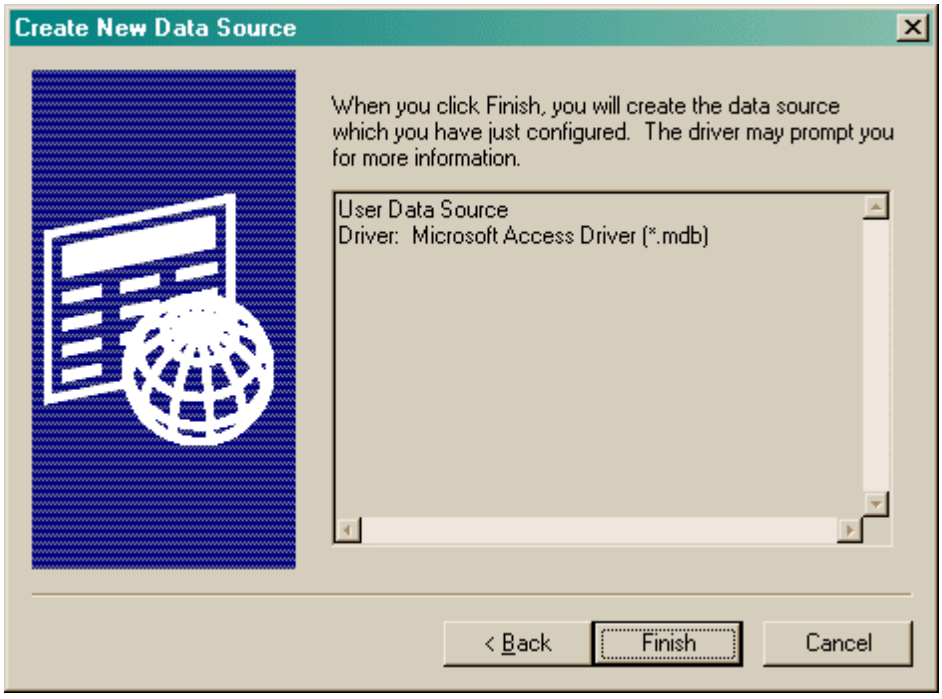
Click New.



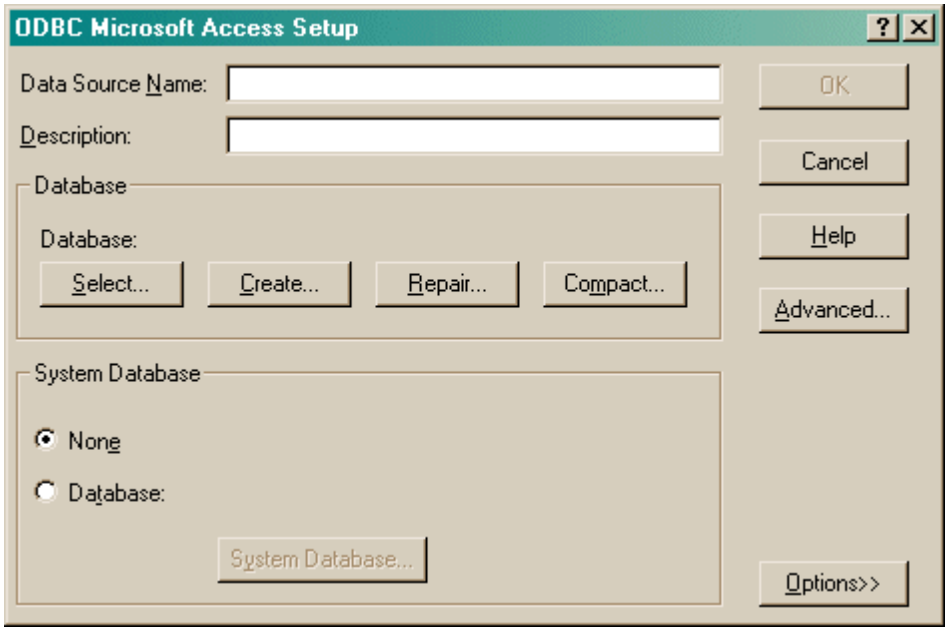
Choose a User or System data source. **User** is probably more sensible. Click Next.



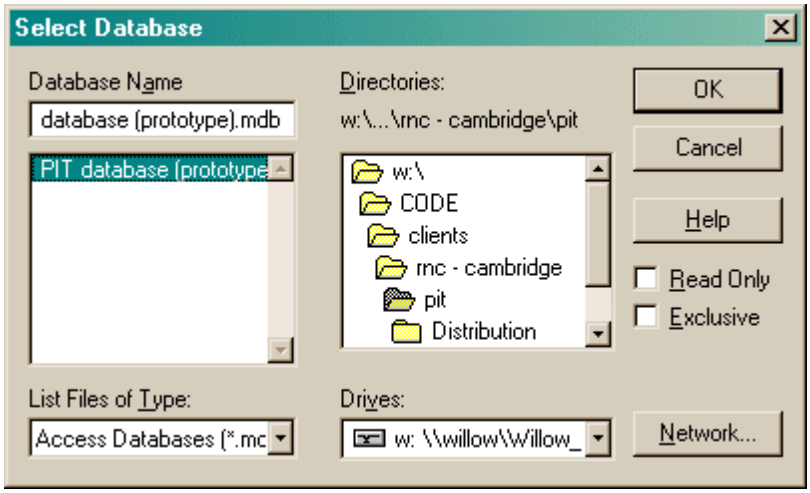
Choose your database driver. Click Next.



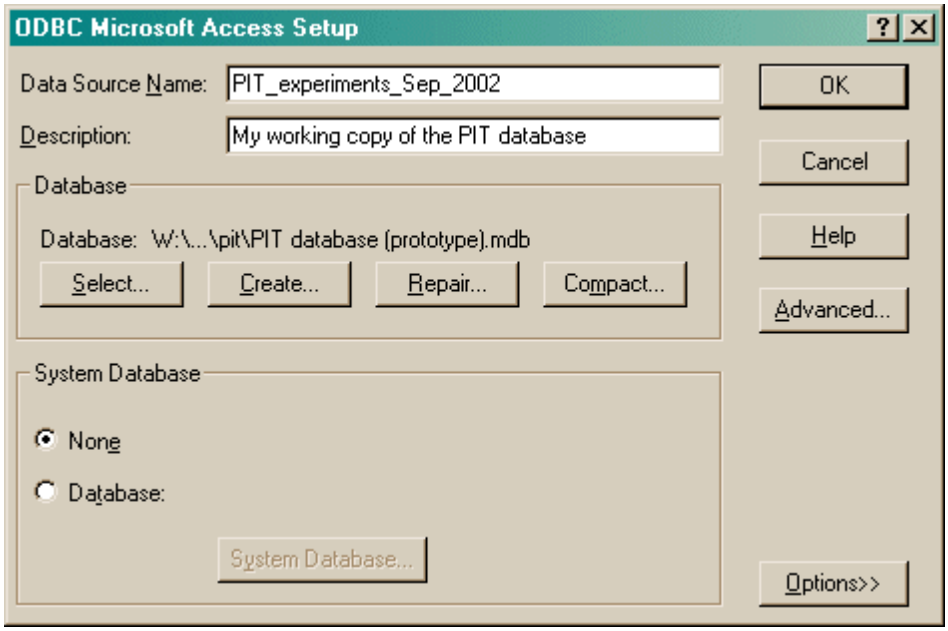
Click Finish.



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.

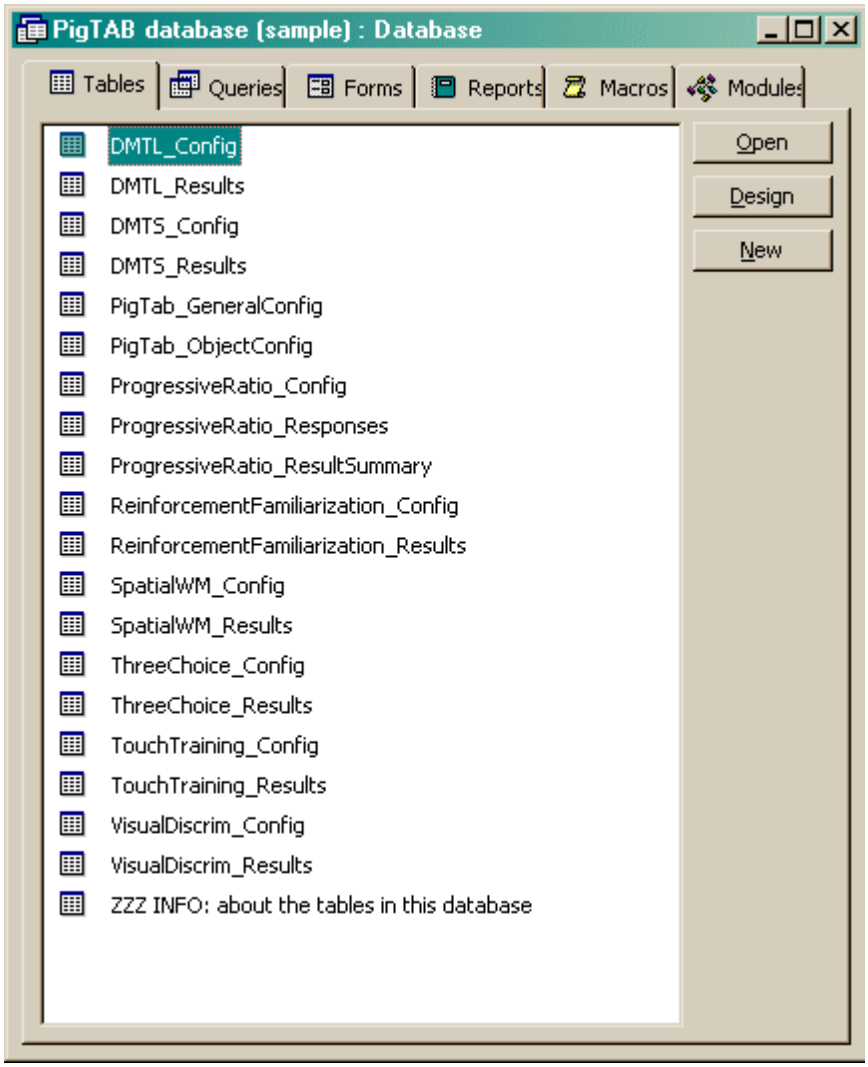
1.8.3 Using the Microsoft Access database for PigTAB

Remember: you shouldn't use the supplied database without making a copy for yourself. (It will work, but if you ever uninstalled or reinstalled PigTab, 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 **"PigTab 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:



PigTab 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 DMTL_Results table (which stores results information for the Delayed Matching To Location task):

DMTL_Results : Table

Field Name	Data Type	
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)
NumStimuli	Number	Number of stimuli used in Phase 2
StimulusLocations	Text	Locations of the stimuli in Phase 2. First location is the target; others are dist
Phase1Responded	Yes/No	Did the subject respond in Phase 1?
Phase1ResponseLatencyMs	Number	Phase 1 response latency (ms)
Phase1Succeeded	Yes/No	Did Phase 1 succeed? (If so, either the subject was required to respond and
Phase1Rewarded	Yes/No	Was the subject rewarded for responding correctly in Phase 1?
Phase1Punished	Yes/No	Was the subject punished for failing phase 1?
MemoryDelayMs	Number	Actual memory delay given
Phase2Given	Yes/No	Was Phase 2 given?
Phase2StartTimeMs	Number	Phase 2 start time by the system clock (ms)
Phase2Responded	Yes/No	Did the subject respond in Phase 2?
Phase2RespondedCorrectly	Yes/No	Did the subject respond to the correct stimulus in Phase 2?
Phase2ResponseLatencyMs	Number	Phase 2 response latency (ms)
Phase2Rewarded	Yes/No	Was the subject rewarded for responding correctly in Phase 2?
Phase2Punished	Yes/No	Was the subject punished for failing phase 2?
CorrectionGiven	Yes/No	Was a correction procedure given?
CorrectionStartTimeMs	Number	Correction phase start time by the system clock (ms)
CorrectionResponded	Yes/No	Did the subject respond in the correction phase?
CorrectionRespondedCorrect	Yes/No	Did the subject respond to the correct stimulus in the correction phase?
CorrectionResponseLatencyM	Number	Correction phase response latency (ms)
CorrectionRewarded	Yes/No	Was the subject rewarded for responding correctly in the correction phase?
CorrectionPunished	Yes/No	Was the subject punished for failing the correction phase?
ITITimeMs	Number	Length of the ITI before the next trial (ms)

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 DMTL_Results table (holding results for the Delayed Matching To Location task). I entered some sample results into this table.

DMTL_Results : Table

_Reco	DateTimeCod	Subject	Box	Modu	Trial	TrialStartTi	Num	StimulusL	Phase1Res	Phase1Res	Phase1Succe
1	9/2002 15:00:07	fish	0	0	0	97818644	2	4	☑	2945	☑
2	9/2002 15:07:57	fish	0	0	0	98215005	2	6	☑	5708	☑
3	9/2002 15:07:57	fish	0	0	1	98247402	2	8	☑	6950	☑
4	9/2002 15:07:57	fish	0	0	2	98277846	2	8	☐	0	☐
5	9/2002 15:07:57	fish	0	0	3	98308460	3	6:7	☑	8522	☑
6	9/2002 15:07:57	fish	0	0	4	98328518	3	4:5	☑	4006	☑
7	9/2002 15:07:57	fish	0	0	5	98388425	3	4:8	☐	0	☐
8	9/2002 19:23:11	fish	0	0	0	113541673	2	0:6	☑	3255	☑
9	9/2002 19:23:11	fish	0	0	1	113557716	2	4:8	☑	7561	☑
10	9/2002 19:23:11	fish	0	0	2	113616080	2	3:0	☐	0	☐

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.

1.8.4 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	**N**umberOfSubjects	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.

Index

- P -