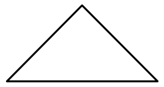
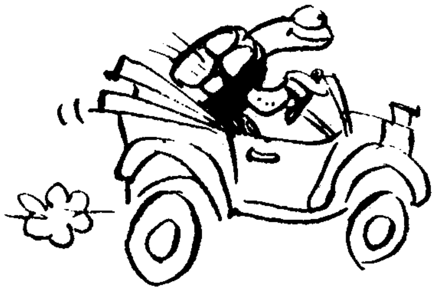
Welcome to the Turtle World of Logo

Type **CS** and press **Enter** ¿ to show your turtle.

Its **HOME** is in the middle of the screen.

Where is its head?
Where is its tail?





Your turtle is lazy. It will not move without a **C.A.R.**

Command	What to do!
Argument	How much to do!
Return (**Enter** ¿)	Do it!

Here are four movement commands your turtle understands

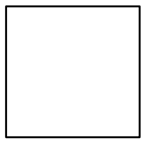
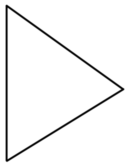
FD (FORWARD)	**RT** (RIGHT)
BK (BACK)	**LT** (LEFT)

Try a **C.A.R.** with each one in turn to see what they do.

E.g. **FD** $_{\text{space}}$ **100** Return (**Enter** ¿)

Use **CS** (Clear Screen) between each **C.A.R.** to return HOME with a clear screen.

Now write a sequence of **C.A.R.**s to draw a regular square and a regular triangle.
Regular means: identical 'sides' and identical 'corners'

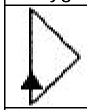
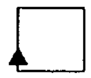
Keep trying until you succeed. Can you see a pattern in your results?

REPEAT is a new command to help you make regular polygons easier and quicker to draw. Here is how to use it:

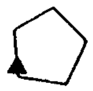
REPEAT$_{\text{space}}$**n [Com1**$_{\text{space}}$**Arg1**$_{\text{space}}$**Com2**$_{\text{space}}$**Arg2**$_{\text{space}}$**etc. ...]**

The **REPEAT** command **must have square brackets**. The brackets contain as many 'command argument' pairs as it is necessary to repeat.

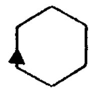
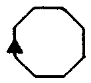
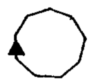
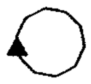
Try the new command on the **triangle** and the **square**.
Write those results in the table below and continue for the other 'regular' polygons.
Note: You may accidentally discover the 'angle of turn' for another polygon.
Be sure to write it in before you forget it.

Polygon	Name	Sides	Angle	Instructions to the Turtle
	Triangle	3		REPEAT 3 [FD RT]
	Square			REPEAT

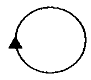
Here are two new commands to help you see whether your shape joins up **exactly**:
HT - Hide Turtle
ST - Show Turtle
... and, remember, **CS** - Clear Screen - will allow you to start anew.
Ask your Mentor for another command for a more **exact** method.

	Pentagon			

Ask your Mentor how to '**copy and edit'** previous tests in the Commander Window.

	Hexagon			
	Octagon			
	Nonagon			
	Decagon			

Now try a circle. Take note of the 'sides-angle' pattern above.

How fast can you make your circle draw?

Look at your completed chart on the previous page.

- Can you see a pattern in the **Sides** and the **Angle** columns?
- For each polygon, is there a relationship between the two numbers?
- Can you write a theory that says how many degrees need to be turned to complete the polygon so that your turtle returns to its original heading?

Total-trip Theorem: my hypothesis is....

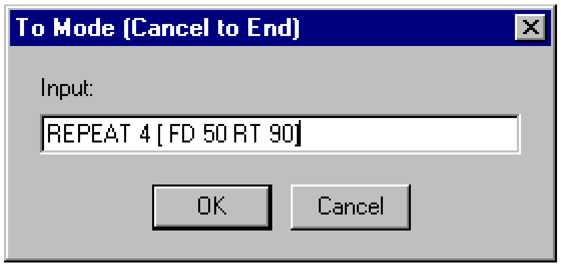
The PROCEDURE Mode: Teaching your turtle new tricks

Up to now you have worked in the **'direct command'** mode.
Now you will learn how to teach your turtle PROCEDURES that it can remember.

This is how to teach your turtle to draw a **SQUARE** (or **SQU**).

You will need two new commands: **TO <new proc>** to start the teaching mode, and **END** to end the teaching mode.

1. type **TO SQUARE** and press **Enter** ¿
2. type your SQUARE commands e.g. **REPEAT 4 [FD 50 RT 90]** and press **Enter** ¿
3. now type **END** to finish your teaching. The 'Commander' window will show 'SQUARE defined'.



REPEAT 4 [FD 50 RT 90]

4. finally type **SQUARE** and press **Enter** ¿ to get the turtle to draw your square.

Good Style

Concentrate on developing 'good style' when naming your procedures.

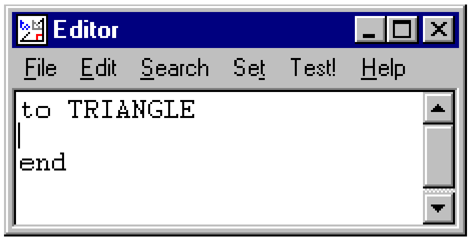
Instead of **SQUARE** you could have called the square procedure **S**, **SQ**, or **SQU**, or **BOX**, or perhaps **ROBERT**. It is best to keep your PROC names short but still meaningful. **S** would really be too short and could stand for **SUN** or **SHADOW** whilst **ROBERT** would not be a meaningful name for square.

Creating a new procedure with EDIT

You have already taught a SQUARE procedure using TO SQUARE.

Now you will teach a TRIANGLE procedure using **ED "TRIANGLE**.

- Type **ED "TRIANGLE** in the Commander Window 'Input Box' and press **Enter** ¿
- Click the cursor after TRIANGLE in the Edit Window and press **Enter** ¿ to create an empty line with the cursor at its beginning.



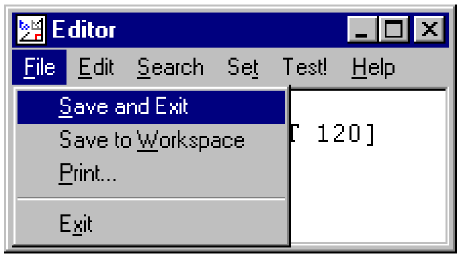
```
to TRIANGLE

end
```

- Now type your TRIANGLE commands **REPEAT 3 [FD 50 RT 120]** in the empty line.

You are now ready to **save** your new procedure.

Saving the Edit Window.

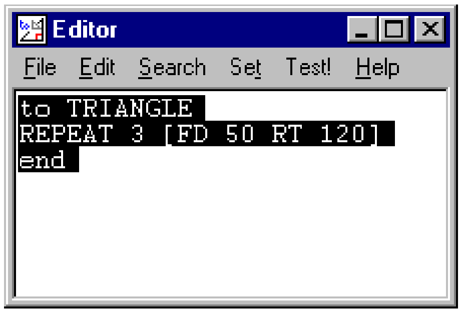
- Select **File>Save and Edit** and click your mouse.
- Type **TRIANGLE** in the Commander Window 'Input Box' and press **Enter** ¿ to test your new procedure.



Turn to the next page to learn how to **clone** procedures.

Cloning a new procedure with EDIT.

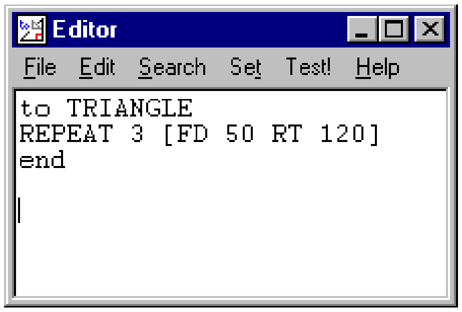
- Type **ED "TRIANGLE** in the Commander Window 'Input Box' and press **Enter** ¿
- Highlight **ALL** of the procedure to cover both **TO and END**
- Copy the highlighted area by **Edit>Copy** or by **holding [Ctrl] and pressing [C] (^C)**



```
to TRIANGLE
REPEAT 3 [FD 50 RT 120]
end
```

You are now ready to **clone** your procedure.

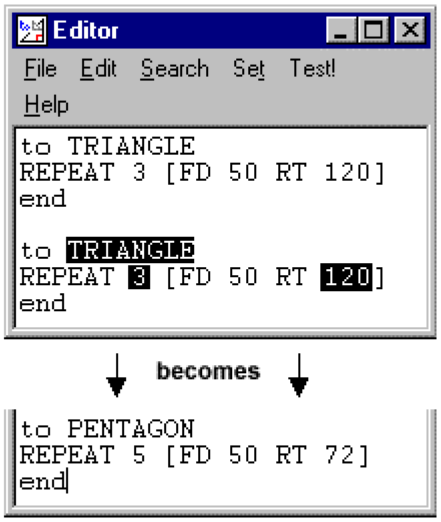
- Click after **END** and press **Enter** ¿ **twice** to create a gap between the existing procedure and the copy to be pasted.
- Paste the copy by **Edit>Paste** or by **holding [Ctrl] and pressing [V] (^V)**



```
to TRIANGLE
REPEAT 3 [FD 50 RT 120]
end
```

You are now ready to **edit** your **cloned** procedure.

- Highlight the pasted procedure name i.e. TRIANGLE and type the clone procedure name i.e. **PENTAGON**
- then highlight the 'number of turns' after REPEAT and type **5** to delete the 3
- and then highlight the 'angle of turn' after RT and type **72** to delete the 120
- now save your cloned procedure with **File>Save and Exit**.



```
to TRIANGLE
REPEAT 3 [FD 50 RT 120]
end

to TRIANGLE
REPEAT 3 [FD 50 RT 120]
end
```

```
to PENTAGON
REPEAT 5 [FD 50 RT 72]
end
```

Be sure to test your new procedure in the Commander Window.

- now repeat the above process for all your shapes.

Saving files of procedures

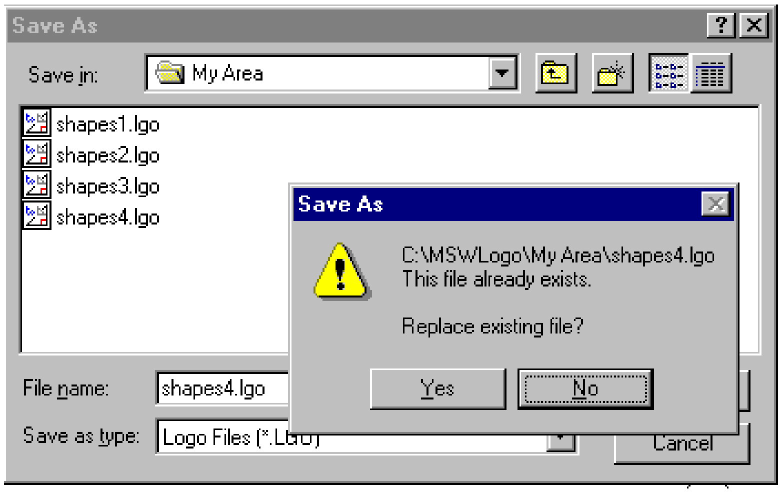
Now you have taught the turtle many things, you need a quick way to re-teach those things next time you switch the computer on. Here is how to **save** a file of procedures:

- First select <B>**File>Save**</B> on the MSWLogo Screen Menu Bar and select the Disk or Directory where your files are to be saved.
- Then type **SAVE "MyFile3.LGO** in the Commander Window 'Input Box' (or MyFile4, MyFile5, etc.- whatever is next in the sequence) and press **Enter** ¿ . ***Good style*** *requires that you use a* ***file name*** *that describes the procedures it contains.*

You need to be very careful with your choice of filename because if it is the same name as an existing file it will over-write it and the original file will be lost forever.

A **safer** method is to select **File>Save As** which will:

- open the file window to show the existing file names,
- allow you to type a new name, and
- if you choose an existing file name it will warn you.



Save As

C:\MSWLogo\My Area\shapes4.lgo
This file already exists.

Replace existing file?

Saving a file of selected procedures

It is good practice to save only the procedures that match the descriptive file name you have used.
To save a selected list of procedures you can use the 'savelist' command, e.g.
SAVEL [FLOWERS GRASS BUSH TREES ... etc.] "GARDEN
This would leave out any files not needed. Instead you could erase all the procedures not needed but you may prefer not to do that.

Loading files of procedures

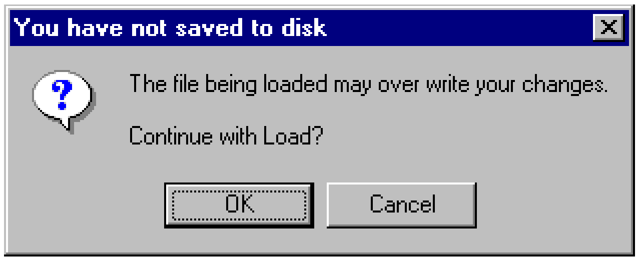
Here is how to **load** a file of procedures:

- Select **File>Load** on the MSWLogo Screen Menu Bar and select the Disk or Directory from which your file is to be loaded.
- Type **LOAD "MyFile.LGO** in the Commander Window 'Input Box' and **Enter** ¿ .

Several files may be loaded at the beginning but be **very careful** as procedures in later files will over-write any earlier procedures that have the **same name**.

After you have edited, or added procedures to, the workspace you will get this warning.

Think carefully about which choice you will make.



You have not saved to disk

The file being loaded may over write your changes.

Continue with Load?

Changing the size of your polygons

Use the Editor to change the size of your triangle: **ED "TRIANGLE** to create a triangle size 267 i.e. **FD 267**. Try another size.

To create two different-sized triangles at the same time you should now **clone TRIANGLE** to produce **TRI-50** and **TRI-267**. To have ten different sizes simultaneously you would need ten clones. This would be a rather cumbersome method.

Logo has a better way. You can give TRIANGLE an 'argument': a variable input size, e.g. TRIANGLE 100, TRIANGLE 250, etc. just like FD 100.

ED "TRIANGLE and change the procedure so that it reads:

Tells the computer to label a store with the name "**SIZE** and put the number it finds here in that store.

```
TO TRIANGLE  :SIZE
     REPEAT 3 [ FD  :SIZE  RT  120 ]
END
```

Tells the turtle to move forward the amount it will find in the store named "**SIZE**.

Now save the Editor and test your edited TRIANGLE. If you get the message "not enough inputs to TRIANGLE" you have forgotten to attach the input size, e.g. TRIANGLE **123**.

Now type **TRIANGLE 50 TRIANGLE 100 TRIANGLE 150 TRIANGLE 200 ... etc.** in the Commander Window 'Input Box' as a **single line input**.

Now include an **input variable** on all your shapes as follows:

- EDALL
- Highlight the $_{\text{space}}$**:SIZE** after **TO TRIANGLE** and copy by doing **[Ctrl] [C]** i.e. **^C**
- Click after each **TO shape** in turn and paste $_{\text{space}}$**:SIZE** by doing **[Ctrl] [V]** i.e. **^V**
- Delete the value after each **FD instruction** and **paste it again** by doing **^V**
- Save the Editor Window and test each shape procedure.

Experiment with your variable shapes. Try this pattern:

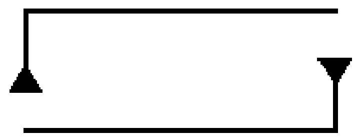
```
TO PATTERN
    REPEAT 3 [ TRIANGLE 100  RT 120 ]
END
```

See what other interesting patterns you can create using TRIANGLE, SQUARE, HEX, OCT etc. with **different REPEAT values and 'angle of turn' values**.

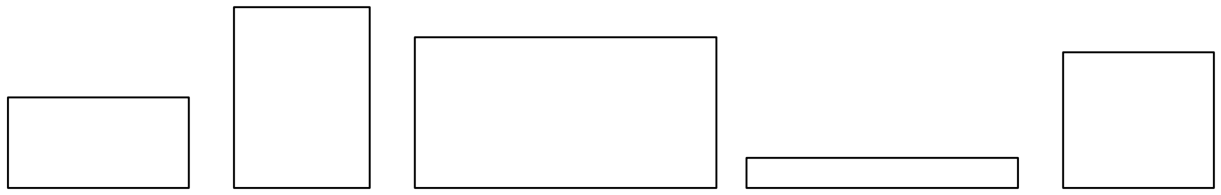
Working with more than one variable

Before we start putting shapes together we should teach our turtle two very useful procedures: the **rectangle** and the **arc**.

Teach your turtle to draw a rectangle with sides of 300 and 100.
    Can you use REPEAT?
    How many times does the pattern repeat?

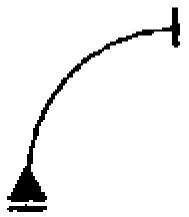
	TO RECTANGLE END

Now your rectangle works, how are you going to control the shape to produce different rectangles of different size and proportions?



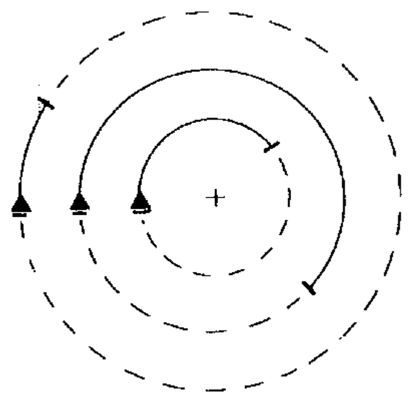
You can do this with variables!
    How many **numbers** will change?
    How many **variables** will that need?
    Each variable will need a different name. What will you call them?

Now do the same for ARC.
Clone CIRCLE; i.e. ED "CIRCLE, copy it and use the copy as a model for ARC.

	TO ARC END

How are you going to control

- how much the arc goes around the circle?
- how far 'out' it goes?



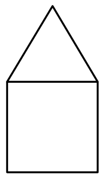
How many variables will you need?

What are going to call them?

Building a 'house'

Good Logo technique is to build more complex procedures from a set of simple procedures and then even more complex procedures using your new procedures.

To build a HOUSE (or HUT) you need to place a TRIANGLE brick on top of a SQUARE brick using a 'link' or 'cement'. This 'link' will use some of the primitive commands (FD, BK, RT, & LT). **'Play turtle'** before you write your commands.

	TO HOUSE END

Good Logo style is to arrange your procedure so that it is easy to read the procedure and easy to see patterns. Arrange your HOUSE procedure like this:

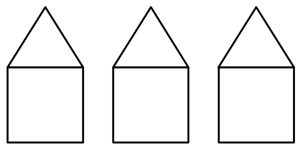
```
TO HOUSE
        ............................................................ ; Procedure 1 - the walls
        ............................................................ ; 'link' or 'cement' to roof
        ............................................................ ; Procedure 2 -the roof

END
```

Your house probably finishes with the turtle at the top of the wall. Add another 'link' or 'cement' after **Procedure 2** to bring the turtle back to the beginning of HOUSE, to exactly the same position from which it started.

Now teach your turtle to **VILLAGE** using your HOUSE procedure. You could teach your turtle to **SPACE**: a procedure containing the commands it needs to move from HOUSE to HOUSE without drawing a line.

You will need two new commands, **PU** and **PD**, to help create **SPACE** between houses.

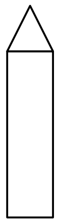
- **PU** means 'pen up' so your turtle does not draw as it moves
- **PD** means 'pen down' so your turtle continues drawing.

	TO SPACE END TO VILLAGE END

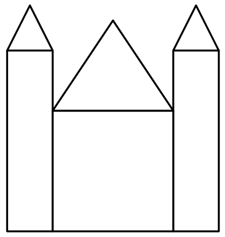
Could you use REPEAT for the VILLAGE procedure?

Linking simple 'shapes'

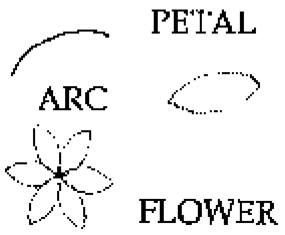
Start by designing a **TOWER**. You will need the RECTANGLE and TRIANGLE procedures. **Clone HOUSE** and use it as a model using the same pattern of procedures and 'links'.

	TO TOWER END

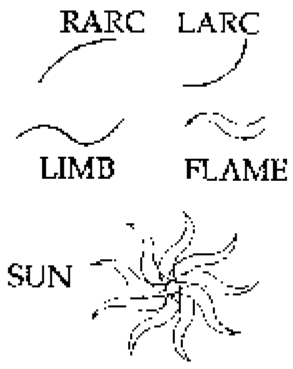
Now try to assemble a CHURCH.

	TO CHURCH END

Here are two more designs for you to try.

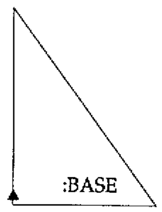
	

For this second design **clone ARC** to create the RARC (right-arc) and the LARC (left-arc) procedures.

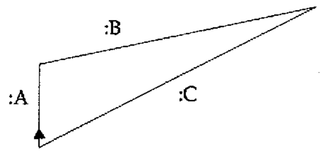
	

Linking more 'shapes'

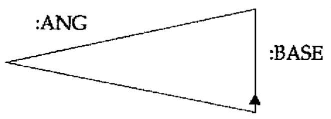
Ask your Mentor for these **'extra shapes'** consisting of various types of triangles, left and right parallelograms and a trapezium. Try them out!



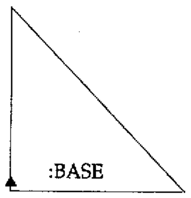
TRI30 :BASE



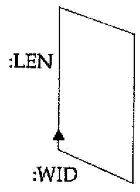
SCALENE :A :B :C



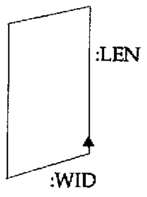
ISOTRI :ANG :BASE



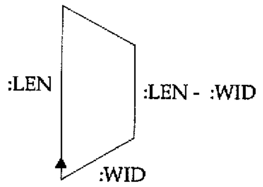
TRI45 :BASE



R.PAR :LEN :WID



L.PAR :LEN :WID



TRAP :LEN :WID

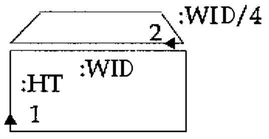
Playing 'turtle'

Work with a partner on the following pairs of shapes and pretend to be a turtle and do exactly what your partner says **except when the request is not a legal Lego command then you may reply "I don't know how to ....<whatever>".**

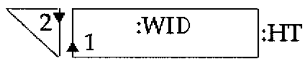
Write down each step that seems correct and then test them.

Note*: it may help to rotate the 'drawing' as you 'talk' your way along the path.*

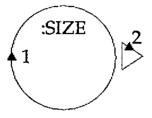
```
           RECT :HT :WID            ;Shape 1
turtle talk ..................................  ;Link
           TRAP :WID :WID/4         ;Shape 2
```



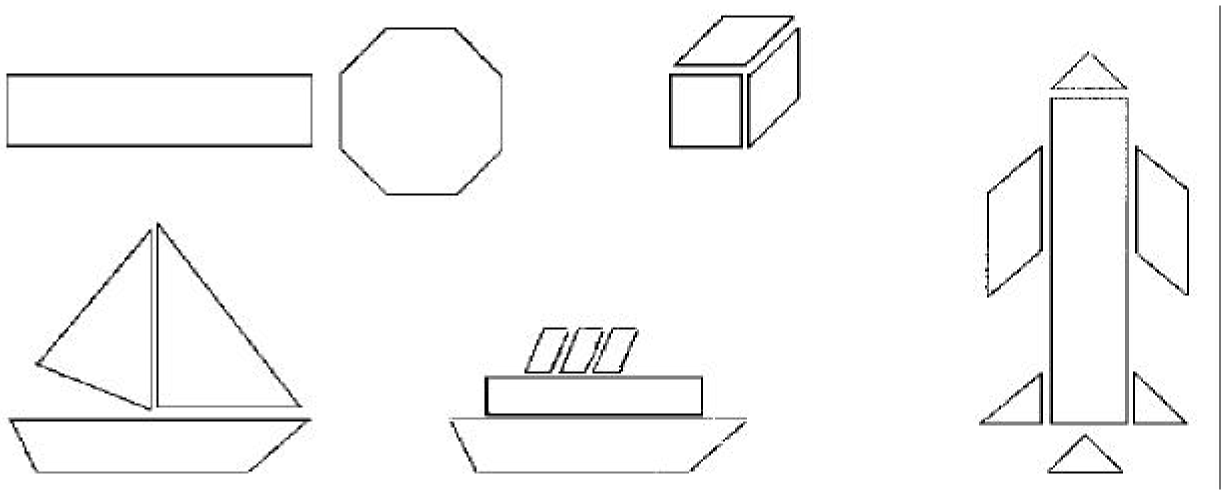
```
            RECT :HT :WID           ;Shape 1
turtle talk  .................................  ;Link
           TRI45 :HT                ;Shape 2
```



```
            CIR :SIZE               ;Shape 1
turtle talk  ..................................  ;Link
            TRI :SIZE               ;Shape 2
```



Now try linking each of these five groups of shapes.



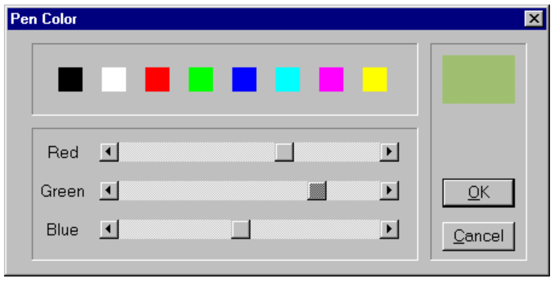
Proceed as follows:

- First **mark the turtle starting point** on each shape in a group and **number** them in the order you will draw them.
- Then 'turtle talk' your way around the group to move the turtle from the first starting point to the second and subsequent starting points.
- Finally test your 'turtle talk' remembering to 'talk' your way back to the first point: the Reverse-path Principle.

Now create some groups of your own using the available shapes.

'Turtle talk' your way around them and then test by writing procedures.

Choosing colours (colors)



You can choose pen, flood-fill, and screen colours by using the MSWLogo Screen **Menu Bar** e.g. **Set>PenColor** etc. and select one of the 'index' colours displayed there or use the Red, Green, Blue scroll-bars to create your own colour.

Mostly you will need to control colours with procedure commands: **SETPC pencolor**, **SETFC floodcolor**, and **SETSCREENCOLOR screencolor**.

For that you will need the colour values given in the **Colour Table** below, or if you have created your own colour then type **SHOW PENCOLOR**, or whatever, in the 'Input Box' and record the colour values displayed for future use.

Create a library of colour procedures if you have difficulty in remembering the colour values for each colour you use.
The one for 'red' follows:

```
TO red
    OP 4        ; or OP [255 0 0]
END
```

then you will be able to type **SETPC red** rather than have to recall the colour values.

Color Table

Name	Index	RGB values	Name	Index	RGB values
Black	0	[0 0 0]	Brown	8	[155 96 59]
Blue	1	[0 0 255]	Light brown	9	[197 136 18]
Green	2	[0 255 0]	Mid-green	10	[100 162 64]
Cyan	3	[0 255 255]	Blue-green	11	[120 187 187]
Red	4	[255 0 0]	Salmon	12	[255 149 119]
Magenta	5	[255 0 255]	Blue-ish	13	[144 113 208]
Yellow	6	[255 255 0]	Orange	14	[255 163 0]
White	7	[255 255 255]	Silver	15	[183 183 183]

Drawing coloured lines

When all lines you draw are black there is no need to lift the pen between shapes that touch and you only need **PU** and **PD** to jump to new locations without drawing a line. **When each shape is a different colour you must lift the pen between shapes** to avoid over-writing the different colour already drawn.

Alter your HOUSE procedure and put the pencolor required before each shape then put **PU at the beginning, and PD at the end, of each link**, as follows:

```
TO HOUSE :SIZE
    SETPC 6  SQU :SIZE            ; draw yellow walls
    PU  FD :SIZE RT 30  PD        ; link to roof
    SETPC 4  TRI :SIZE            ; draw red roof
    PU  LT 30 BK :SIZE PD         ; link to ground (R.P.P)
END
```

Note that the PU and PD are not components of the Reverse-path Principle.

You can also use pencolor to colour text and characters that you **LABEL** on the screen.
Ask your Mentor about Supplementary Topic: Using 'Characters' as graphics.

Filling an enclosed space with colour

The following basic sequence of events is **essential**:

1. choose the colours,
2. draw the shape,
3. lift the pen and move into the shape,
4. fill with colour,
5. return to the edge of the space and put the pen down,

as follows, **each line separately entered in the Input Box**:

- SETPC 6 SETFC 6 ; set pen and flood colours to yellow
- SQUARE 80 ; draw shape
- PU RT 45 FD 20 ; move **part-way** inside shape
- FILL ; fill with flood colour
- BK 20 LT 45 PD ; move back to edge (Reverse-path P.)

Note the need for **PU** in the third step to avoid drawing a line, reversed by **PD** in the fifth step to resume line drawing.

Now create a **COLOR.WALLS** procedure from the above lines and then clone **COLOR.ROOF** from COLOR.WALLS but due to the 'sharper' shape of the triangle you will need to reduce the 45° angle for the move inside the triangle.

Finally replace the SETPC 6 SQUARE :SIZE and the SETPC 4 TRIANGLE :SIZE lines in Step 2 above with your two new procedures to get a 'painted' house.

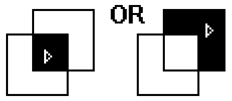
Your Mentor can offer a clever alternative method, e.g. **PAINT "SQU :SIZE yellow**, which doesn't need a new procedure for each coloured shape but before you use it you should have mastered the steps above.

Filling with colour when shapes overlap

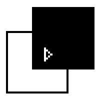
FILL has two modes of operation, which you need to understand if you are trying to flood a shape with colour when it overlaps another shape flooded with a different colour.

FILL, the default mode, or **(FILL "false)** will flood to the boundary of the colour currently under the pen.

Depending on how far the pen goes inside the last-drawn shape it could give either of the two adjacent results.



(FILL "true) will flood to the boundary of the shape just drawn (as shown to the immediate right) provided that the pencolor is a different colour than the shape overlapped.



There is potential here for creating some exiting patterns in flood-filled rotated shapes using combinations of **(FILL "true)** and **(FILL "false)**.

Designing Your Project

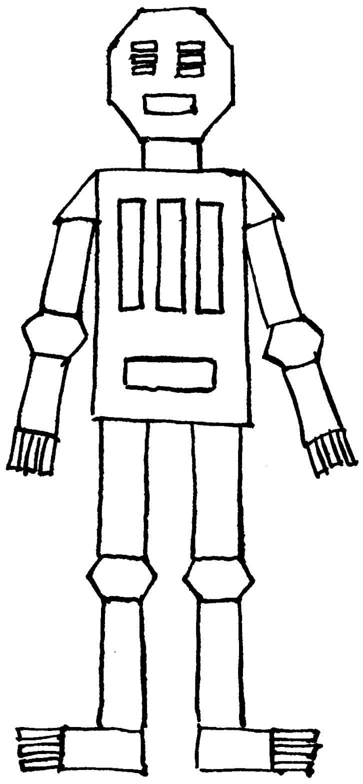
You now have a set of regular polygons:

TRIANGLE, SQUARE, ... CIRCLE, RECTANGLE and ARC (and maybe LARC and RARC) and 'Extra Shapes' which has given you some more simple shapes.

All of these shapes can be drawn in different sizes by the use of variables.

Now your task is to combine these simple shapes to build more complex units, then to combine these units to build an even more complex units and ultimately an elaborate design all over the screen.

Your project...



must use several 'simple' shapes (level 1)

must use variables for size control

must have good structure
- shapes make sub-units (level 2)
- sub-units make units (level 3)
- units make super-units (level 4)
- super-units make picture (level 5)
- must repeat the higher levels

must have good style
- procedures properly organized
  e.g. proc/cement/proc/cement etc.
- use 'reverse path principle' (to allow for later elaboration and use of RANDOM and IF)

may use RANDOM variables (ask your Mentor)

may use IF controls (ask your Mentor)

may use colour.

Here are several project ideas:

a city with buildings and vehicles; a forest with different trees and bushes; a seascape with boats, birds and fishes; a robot army; a garden; and a castle.

Ask your Mentor for a **'Screen Map'** and proceed as follows:

1. Sketch what you want your final screen to look like.
2. Convert your design to **simple shapes**
3. Alter your design so that some assemblies of simple shapes are repeated in different places for different purposes.
4. Name and sketch the level 2, level 3, level 4, level 5, etc. units and decide where each unit starts.
5. Teach procedures, test and debug each unit **starting at the lowest levels** before assembling higher level units and **before** final assembly.
6. Assemble the units to produce your final design.