Introduction

To create or edit scripts requires the Advanced Tile Module

TileScript uses a text script to define the grid and how it replicates across the plan.

This system is really designed for more complex layouts or non-rectangular tiles. The script is quite straight forward and once you've understood the basics of how it works more complex grids can be entered and you'll have much more control than with the designer.

TileScript is a way of defining one or more tiles so that Callidus can quantify areas. It is an alternative to the very simple system of a just width and length of a tile and far more powerful than the Callidus tile designer.

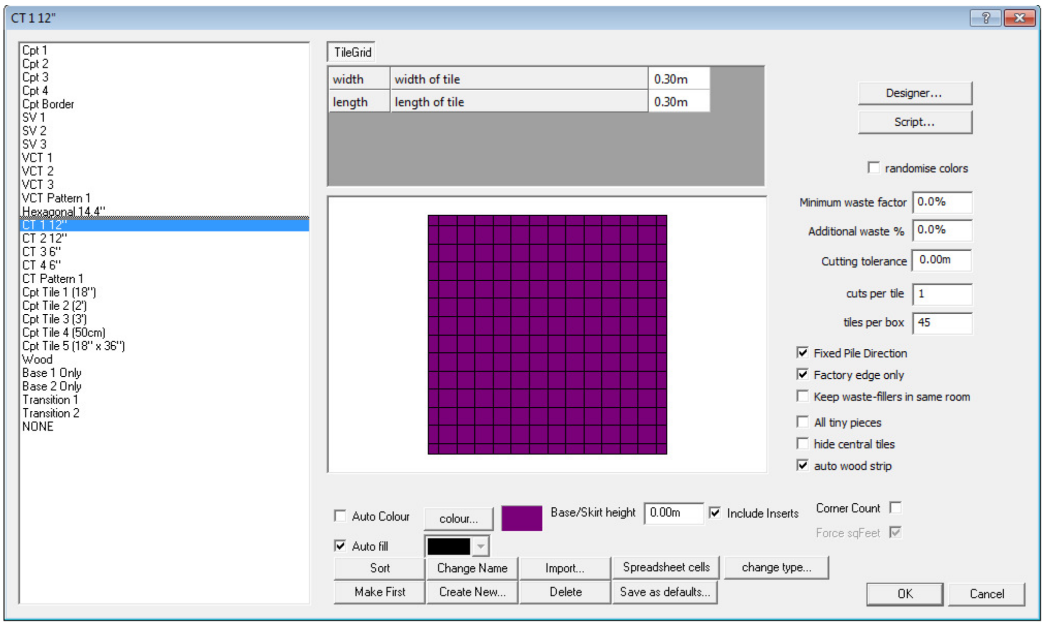
However, bear in mind that if you are entering a complex grid then some time may be needed to get it exactly right.

When dealing with complex tile grids it is VERY helpful to work from a good drawing - even draw it out to scale on graph paper if need be - so you can quickly find all the sizes and positions. Once you reach a certain level of complexity you can very quickly lose track of which bit goes where so a clear diagram is a real help.

The best way to become familiar with TileScript is to try a few simple examples to learn the general principals. Once you understand these you should be able to work with virtually any grid layout.

Do not use it if you do not need to

If you just need a single, rectangular tile then just work with the standard Material editor



CT 1 12"

Cpt 1
Cpt 2
Cpt 3
Cpt 4
Cpt Border
SV 1
SV 2
SV 3
VCT 1
VCT 2
VCT 3
VCT Pattern 1
Hexagonal 14.4"
CT 1 12"
CT 2 12"
CT 3 6"
CT 4 6"
CT Pattern 1
Cpt Tile 1 (18")
Cpt Tile 2 (2')
Cpt Tile 3 (3')
Cpt Tile 4 (50cm)
Cpt Tile 5 (18" x 36")
Wood
Base 1 Only
Base 2 Only
Transition 1
Transition 2
NONE

TileGrid

width	width of tile	0.30m
length	length of tile	0.30m

Designer... Script...

☐ randomise colors

Minimum waste factor: 0.0%
Additional waste %: 0.0%
Cutting tolerance: 0.00m
cuts per tile: 1
tiles per box: 45

☑ Fixed Pile Direction
☑ Factory edge only
☐ Keep waste-fillers in same room
☐ All tiny pieces
☐ hide central tiles
☑ auto wood strip

☐ Auto Colour colour...
☑ Auto fill
Base/Skirt height: 0.00m
☑ Include Inserts
Corner Count ☐
Force sqFeet ☑

Sort | Change Name | Import... | Spreadsheet cells | change type...
Make First | Create New... | Delete | Save as defaults...
OK | Cancel

if you frequently use the tile designer to create grids we suggest you switch over to TileScript - the effort involved will provide you with a great tool-set for your planning work

On screen painting

If you are preparing a design presentation and require photo realistic floor representations please note that such 'painting' of the final floor is achieved using the on-screen painting tools after the grid has been generated.

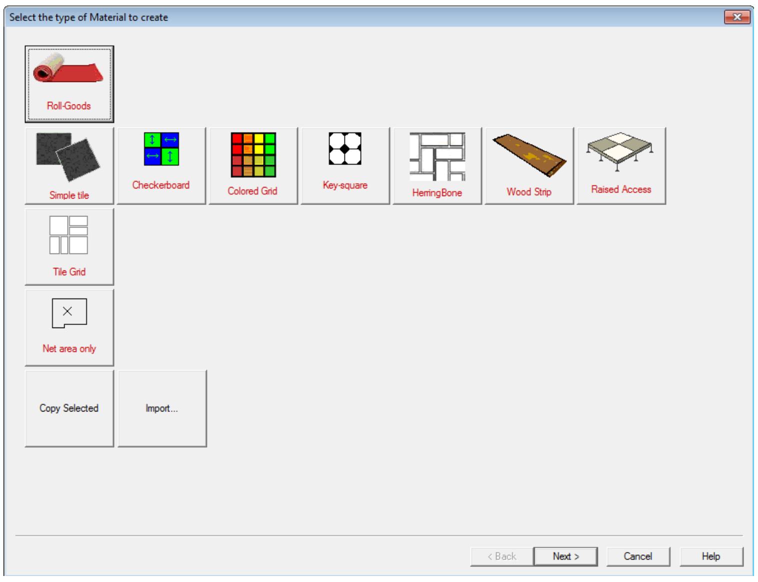
Apart from colour application to define different elements, Tile Script is primarily concerned with the physical layout of the tiles rather than their presentation.

Although you can specify images within TileScript we recommend that you just colour the tiles and then map images to colours after the floor has been covered in tiles.

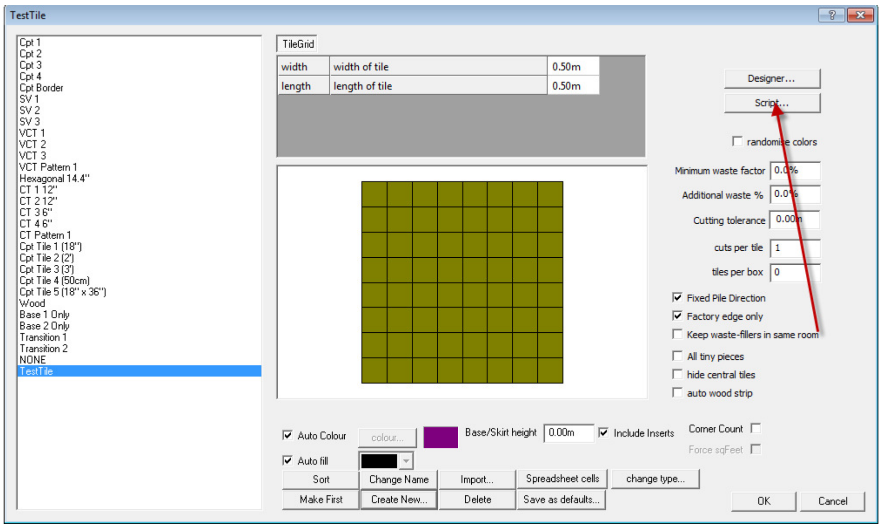
Creating a simple tile

although there is no real need to use TileScript for a simple tile it is worth doing so just to understand some basics

- Create a new tile



- pick Simple Tile
- run through the wizard until the new product appears
- then press the Script button



TestTile

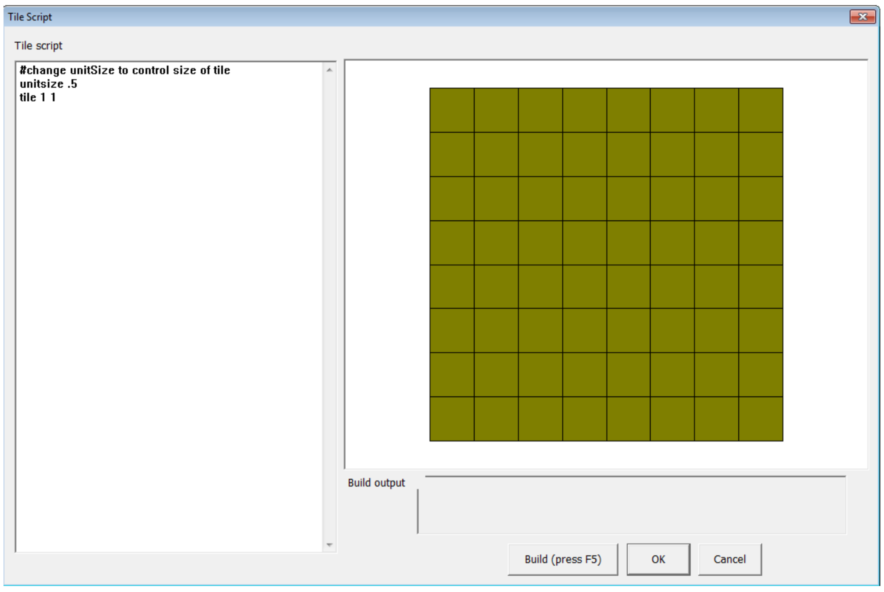
Cpt 1
Cpt 2
Cpt 3
Cpt 4
Cpt Border
SV 1
SV 2
SV 3
VCT 1
VCT 2
VCT 3
VCT Pattern 1
Hexagonal 14.4"
CT 1 12"
CT 2 12"
CT 3 6"
CT 4 6"
CT Pattern 1
Cpt Tile 1 (18")
Cpt Tile 2 (2')
Cpt Tile 3 (3')
Cpt Tile 4 (50cm)
Cpt Tile 5 (18" x 36")
Wood
Base 1 Only
Base 2 Only
Transition 1
Transition 2
NONE
TestTile

TileGrid

width	width of tile	0.50m
length	length of tile	0.50m

randomise colors: ☐
Minimum waste factor: 0.0%
Additional waste %: 0.0%
Cutting tolerance: 0.00m
cuts per tile: 1
tiles per box: 0
☑ Fixed Pile Direction
☑ Factory edge only
☐ Keep waste-fillers in same room
☐ All tiny pieces
☐ hide central tiles
☐ auto wood strip
☑ Auto Colour
☑ Auto fill
Base/Skirt height: 0.00m
☑ Include Inserts
Corner Count ☐
Force sqFeet ☐

any existing tile information is deleted and replaced with a default script for a single tile



Tile Script

Tile script

```
#change unitSize to control size of tile
unitsize .5
tile 1 1
```

Build output

You type the code directly into the 'Tile Script' edit box and the resulting grid is displayed to the right grid (press F5 to view the grid).

each TileScript line consists of a COMMAND and any number of parameters required by that command.

Even when you are building more complex grids remember that most of them can be built with just a few general commands

UNITSIZE

REP

TILE

Comments

If you are working on a complex grid you may want to leave a note as to what the script is doing . If the first character of a line is **#** then Callidus will just ignore it.

# this is just a comment and is ignored

Creating a Tile

A rectangular tile is entered with the command line

Tile 1 1 2 3

this means a *square tile measuring width=1 height=1 positioned at x=2 y=3*

When there are many arguments to a command it can assist with readability to use commas as a separator. e.g.

> *Tile .5,.5,,1 equivalent to*
> *Tile .5 .5 0 1*

Note that the double comma ,, before the '1' means that a default value of 0 will be used

Multiple tiles in one line

When you have the same size tile at multiple positions they can be entered in one line

TILE 1 1 0,0 1,0 2,0 3,0 4,0

equates to four 1x1 tiles place at x,y positions 0,0 1,0 2,0 3,0 4,0

Relative Units

Although you can enter the real dimensions for each tile it is often more convenient to use 'relative units'

Find the smallest dimension in the grid - i.e. the smallest dimension of all the tiles. In most cases every other dimension in the grid and the repeat will be some multiple of this smallest unit.

If the smallest tile dimension is 9" then add the first line of the script as

> ***unitsize 9"***

Now entering a tile

tile 4,1 2,3

would mean *a tile 4*9=36" wide by 9" high positioned x=18" y=27"*

The advantage being that if the same grid is to be used on another installation but with the smallest dimension 6" you just change the unitsize line to

unitsize 6"

and all the measurements automatically adjust themselves

UNITSIZE can be used to make the script more intuitive for you to enter. If you prefer to work in cm then set...

UNITSIZE 0.01

TILE 50,50

this creates a 50cm x 50cm tile whereas,

UNITSIZE 1"

TILE 18 ,18

creates 18" x 18" tile

Once UNITSIZE is set ALL sizes entered thereafter are multiplied by the UNITSIZE to get the final internal sizes for Callidus (which are metres)

Imperial units IMPORTANT

Due to the way the script is interpreted you cannot put imperial measurements directly into the tile sizes. For example

TILE 18" 18"

will be rejected (the inverted commas are not acceptable)

To work with imperial grids you must set the UNITSIZE to an imperial dimension and then work the tiles out from that.

Corners command

TILE allows for simple rectangular tiles but when you have irregular shaped tiles they can entered with the command

CORNERS 0,0 .5,1 1,1

CORNERS followed by a series of coordinate pairs representing each corner of the tile

This is generally used to create more complex designs that are then made available on our web library.

Repeating the grid

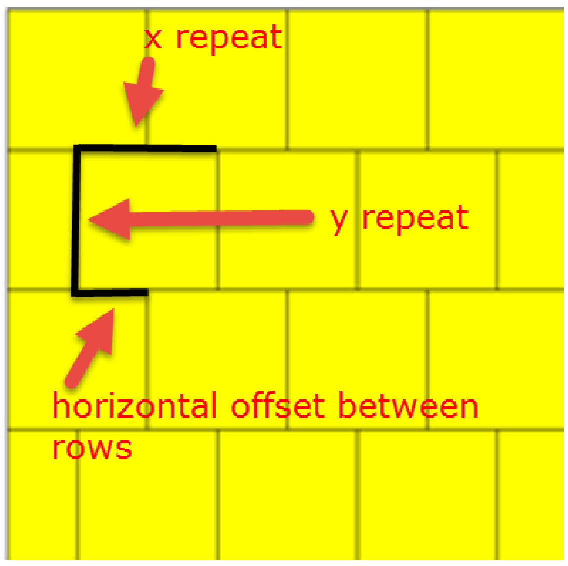
TileScript must define two essential parts:

As well as the grid block of tiles it also needs to know the repeat spacing so that this grid will properly fill the entire floor

add in the line

rep x y dx

for example



unitsize 1

rep 1 1 0.5

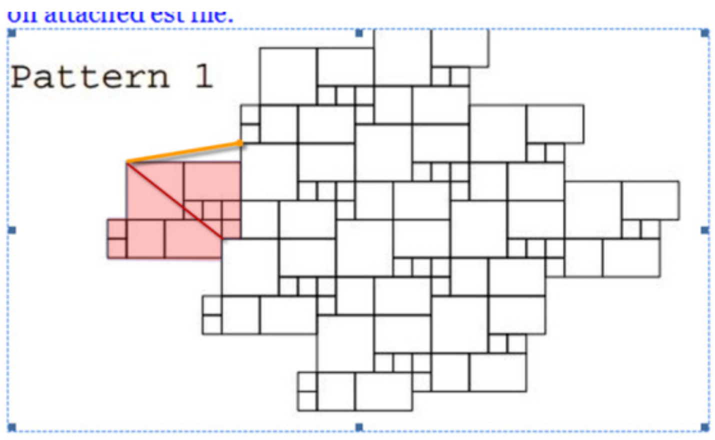
tile 1 1 0 0

the first tile will go down at 0,0. Then tiles will be placed horizontally at a space of 1 units (i.e. next to each other) while the next row will start 1 unit down and .5 to the right to give a classic brick bond effect

Repeat offset in both directions

In some cases the tile pattern does not repeat across the 'natural' horizontal.

In the case the Repeat needs to be



Horizontally: Across 6 **UP** 1 (orange line)

The next row starts 4 down and 5 across (red line)

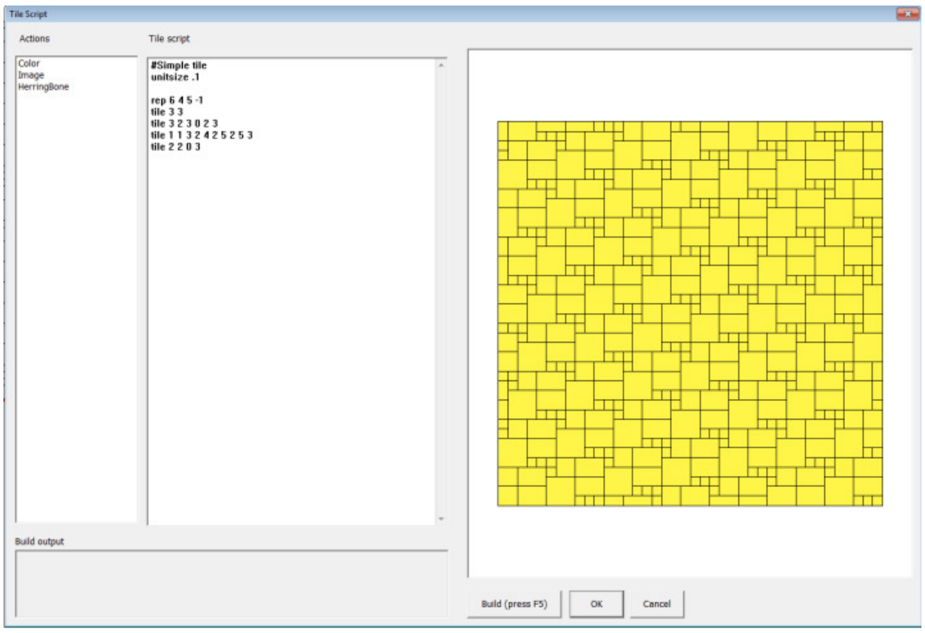
To work this in set a forth (dy) value to the REP line

REP 6 4 5 -1

The -1 at the end is the vertical offset for each horizontal grid block.

The way Callidus handles this is to adjust everything so the tile repeat runs along the ORANGE line rather than the horizontal axis.

IMPORTANT: If you add the vertical adjustment the "pile direction" within the room will be rotated slightly so that the tiles themselves still tend to run in the same orientation as in the diagram.



```
#Simple tile
unitsize .1

rep 6 4 5 -1
tile 3 3
tile 3 2 3 0 2 3
tile 1 1 3 2 4 2 5 2 5 3
tile 2 2 0 3
```

Automatically finding the repeat

Sometimes it can be difficult to see what the real repeat is and Callidus is able to determine this for you.

First off to help you build the grid itself set the repeat to a value large enough that a single grid can be seen.

REP 10 10

Once you are happy the grid is OK , change the line to

REP (remove all values)

Press F5 and if a repeat can be found the values will be set for you

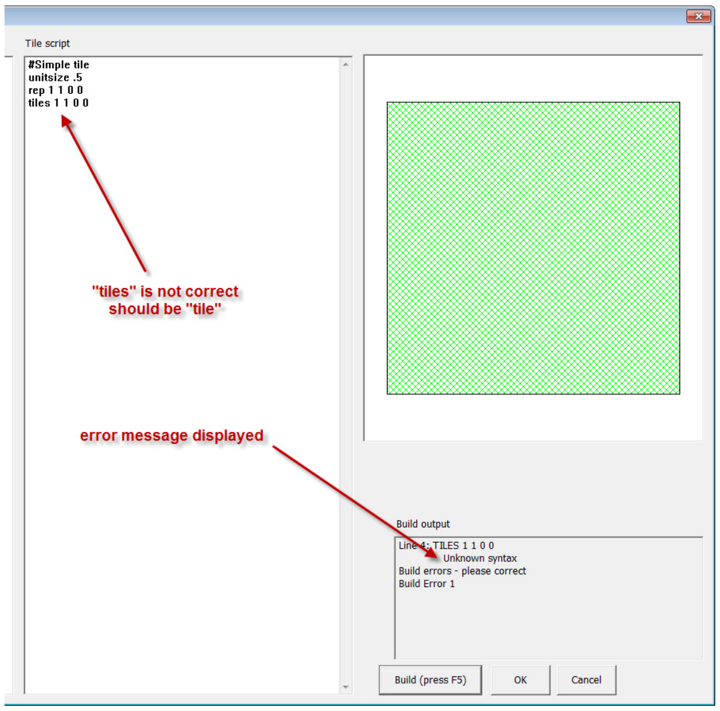
Validating the grid

Press F5 and the script is turned into a grid and then validated.

There are several reasons why a script may not work.

1/ Script errors

This simply means that the script has not been properly put together and a tile grid cannot be produced. You will see an error message



```
#Simple tile
unitsize .5
rep 1 1 0 0
tiles 1 1 0 0
```

```
Line 4: TILES 1 1 0 0
        Unknown syntax
Build errors - please correct
Build Error 1
```

In the case the you must correct the error and Press F5 again until the script can compile.

2/ no two tiles within the grid can overlap each other - on the floor the tiles are always laid next to each other, not on top of one another

3/ once the grid is formed its repeat must ensure that the total floor area is totally covered , without any overlapping tiles and gaps

Once these 3 conditions are met you should have a properly formed grid with a repeat that fills the floor. The script can now be used within Callidus.

Adding a colour to a tile

any tile can be given any colour or bitmap image to create a realistic appearance on the floor with the command

col name

Colours can either use the HTML format

col #26EB0A

or any of the following standard names

```
azure
aliceblue
lavender
lightcyan
powderblue
lightsteelblue
paleturquoise
lightblue
blueviolet
lightskyblue
skyblue
mediumslateblue
slateblue
cornflowerblue
cadetblue
indigo
mediumturquoise
darkslateblue
steelblue
royalblue
turquoise
dodgerblue
midnightblue
aqua
cyan
darkturquoise
deepskyblue
darkcyan
blue
mediumblue
darkblue
navy
mintcream
honeydew
greenyellow
yellowgreen
palegreen
lightgreen
darkseagreen
olive
aquamarine
chartreuse
lawngreen
olivedrab
mediumaquamarine
darkolivegreen
mediumseagreen
limegreen
seagreen
forestgreen
lightseagreen
springgreen
lime
mediumspringgreen
teal
green
darkgreen
lavenderblush
mistyrose
pink
lightpink
orange
lightsalmon
darkorange
coral
hotpink
tomato
orangered
deeppink
fuchsia
magenta
red
salmon
lightcoral
violet
darksalmon
plum
crimson
palevioletred
orchid
thistle
indianred
mediumvioletred
mediumorchid
firebrick
darkorchid
darkviolet
mediumpurple
darkmagenta
darkred
purple
maroon
lightgoldenrodyellow
ivory
lightyellow
yellow
floralwhite
lemonchiffon
cornsilk
gold
khaki
darkkhaki
snow
seashell
papayawhite
blanchedalmond
bisque
moccasin
navajowhite
peachpuff
oldlace
linen
antiquewhite
beige
wheat
sandybrown
palegoldenrod
burlywood
goldenrod
tan
chocolate
peru
rosybrown
darkgoldenrod
brown
sienna
saddlebrown
white
ghostwhite
whitesmoke
gainsboro
lightgray
silver
darkgray
gray
lightslategray
slategray
dimgray
darkslategray
blac
```

for example ***col fuchia*** means that the tile immediately above this line is given the colour 'fuchcia'

Colouring multiple tiles

You can enter multiple colours on the same line

COL c1,c2,c3,c4...

these multiple colours are applied **backwards** to tiles previously entered

c1 applied to the last tile
c2 applied to the last but one
c3 applied to the last but two ...

or if the same colour is needed on multiples

COL RED 15

means "Apply RED to the last 15 tiles".

Adding a bitmap image on a tile

To have a tile filled with a bitmap image use the command

IMAGE fileName

the file can be the entire path to the file such as

IMAGE C:\CALLIDUS\MYIMAGES\Tile.jpg

However, if you are painting multiple tiles from the same folder you can set this with the command

FILES C:\CALLIDUS\MYIMAGES

Now the IMAGE line is simplified to

IMAGE Tile.jpg

and Callidus will pick the file from the FILES path

as for the COL command, multiple images are applied in reverse order to the previous tiles

Image and Colour at the same time

SURFACE colour imageFile

this will apply a colour and an image at the same time

> ***This command is included to allow the Tile designer to generate usable TileScript and we do not recommend its use in pure TileScript***

Colour replacement with images

When you have multiple tiles with the same image it can be time consuming to keep adding the image file name to each tile.

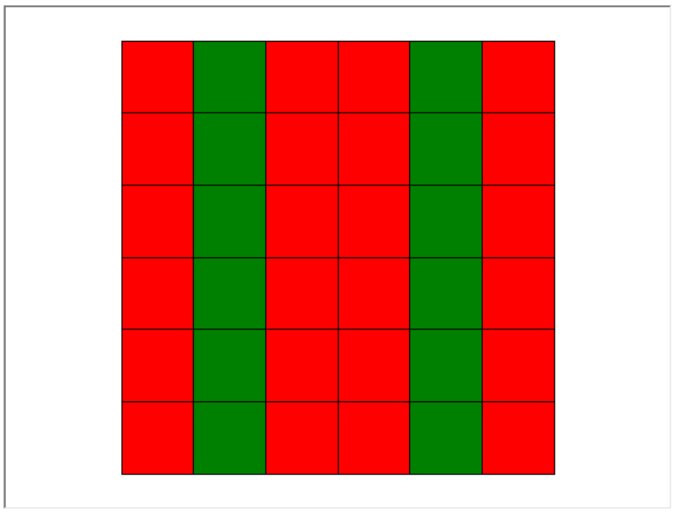
In this case you can separate the tiles using different colours and then with one command add a specific Image to all tiles with a specific colour

rep 5 1

tile 1 1 0,0 1,0 2,0 3,0 4,0

col red green red green red green

creates this grid



adding in the paint line as below will add calImage1.jpg to all tiles with the colour RED

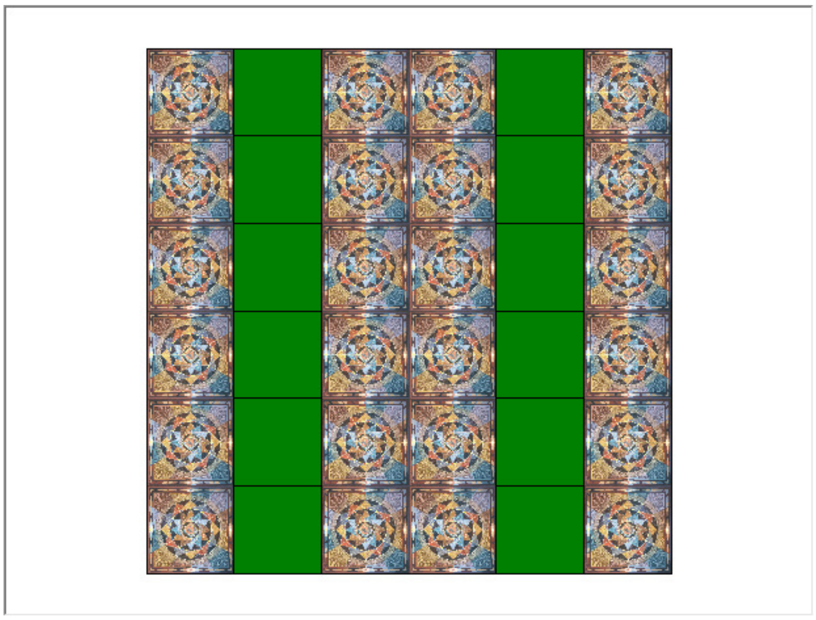
rep 5 1

tile 1 1 0,0 1,0 2,0 3,0 4,0

col red green red green red green

paint red c:\calImage1.jpg

this script generates



the file c:\calImage1.jpg would have to exist on your machine for this effect

Tile Rotation

In the same way that Tiles can be back-filled with colours, you can apply angles too with the TURN command

TURN ang1, ang2, ang3, ang4, ...

this example script creates a 4 tiles grid and rotates each by a quarter turn

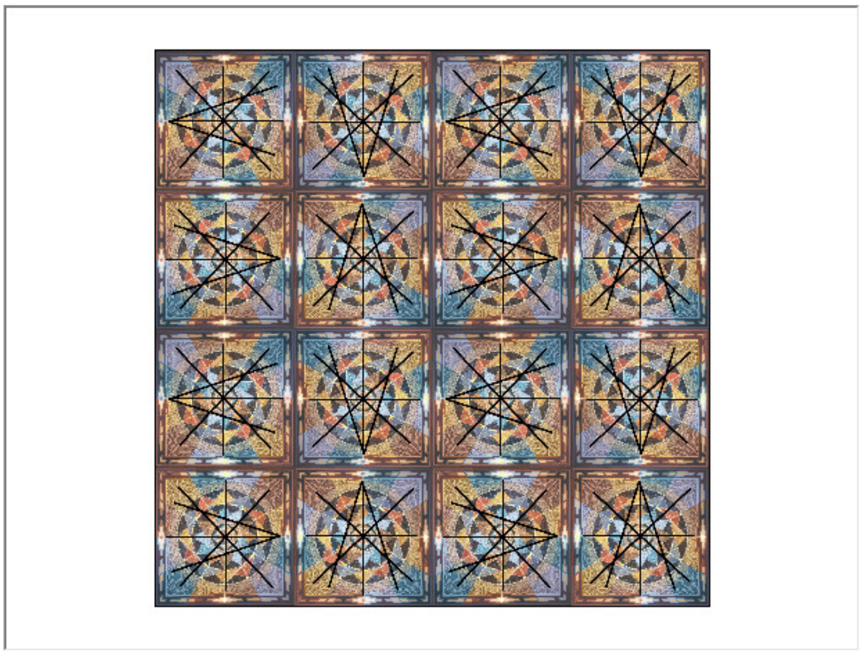
rep 2 2

tile 1 1 0,0 1,0 0,1 1,1

col red 4

paint red c:\calimage3.jpg

turn 0 90 180 270



they are first coloured RED then have an image applied to them

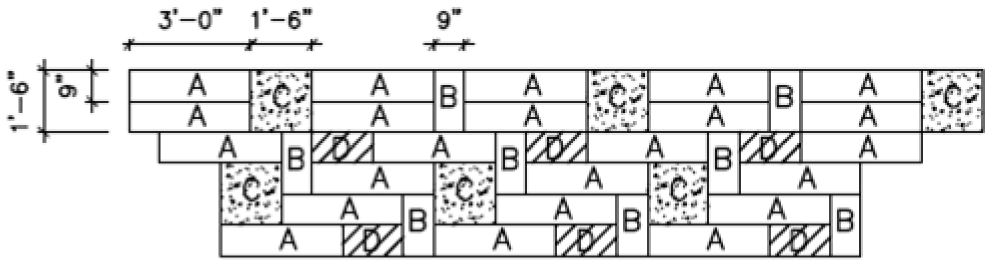
Finally the tiles are turned in a specific order to create a quarter turn repeating system.

This is callmage3.jpg tile image with a directional arrow added. When applied to the grid in this script you can see it being quarter turned across the entire grid

changing the repeat within the grid

In most cases you have a single grid of tiles all using the same repeat. But design do occur where rows of tiles repeat at different intervals. In these cases you just add another REP command at the point where the repeat changes.

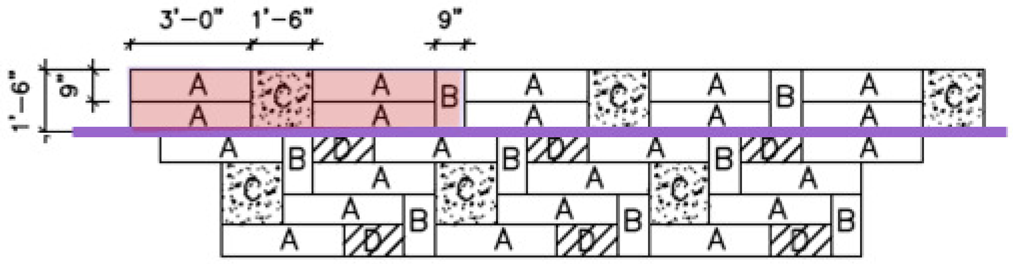
Although this design looks very complex the grid actually breaks down into 2 quite simple sections. But because they repeat differently across the floor the it may be difficult to see this straightaway.



First off, make a list of all the different sizes within the grid. Notice that by picking a UNITSIZE of 9 each tile would be the following sizes

Tile	Physical size	Relative size
A	3' x 9	4 x 1
B	9 x 1'6	1 x 2
C	1'6 x 1'6	2 x 2
D	1'6 x 9	2 x 1

If you look at the diagram notice that it is split into 2 blocks of tiles



The higher section consists of two A tiles plus 1 C tile another two A tiles and a B tile

this group repeats across horizontally 3'+1'6+3'+9 or a total of 8'3

This translates to 11 relative units (11x9) .

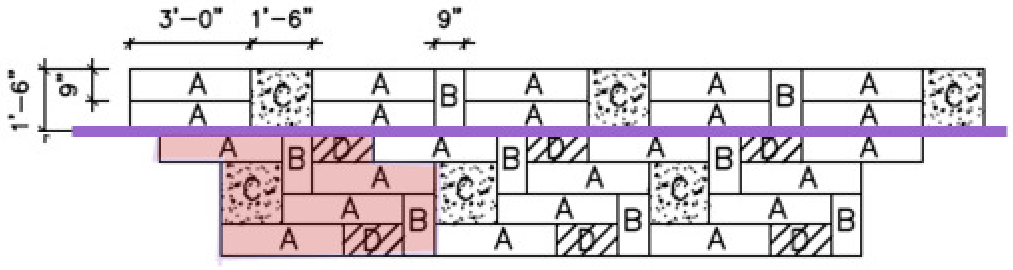
Assuming that the next row is laid directly under the last row in the diagram. The vertical drop between rows of this block would be 6 units with no offset

So the script would start something like this

UNITSIZE 9"
REP 11 6 0
TILE   4 1 0 0
TILE   4 1 0 1

TILE   2 2 4 0
TILE   4 1 6 0
TILE   4 1 6 1
TILE   1 2 10 0

The lower section consists of a different block **with a different repeat so we have to reset the repeat value for the tiles in that section**

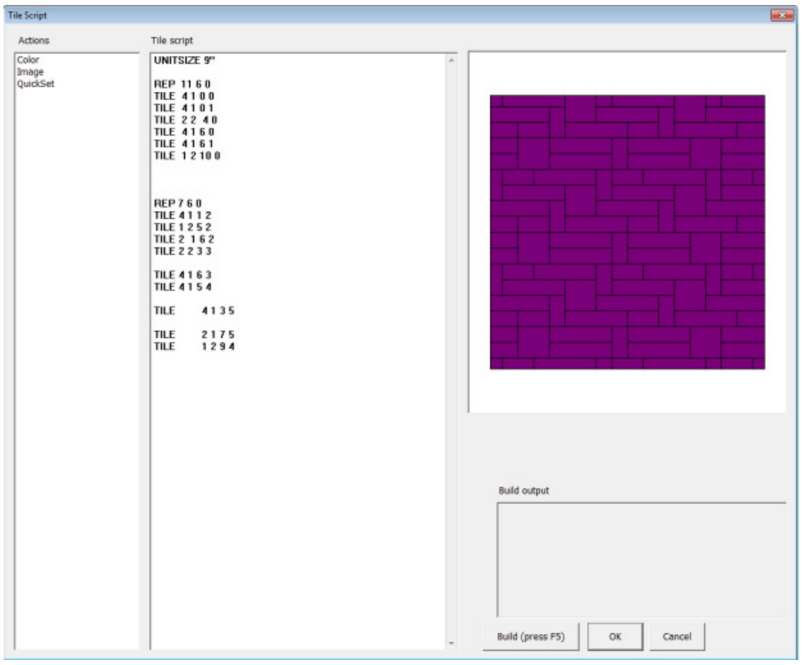


So the script for this part would be

REP 7 6 0

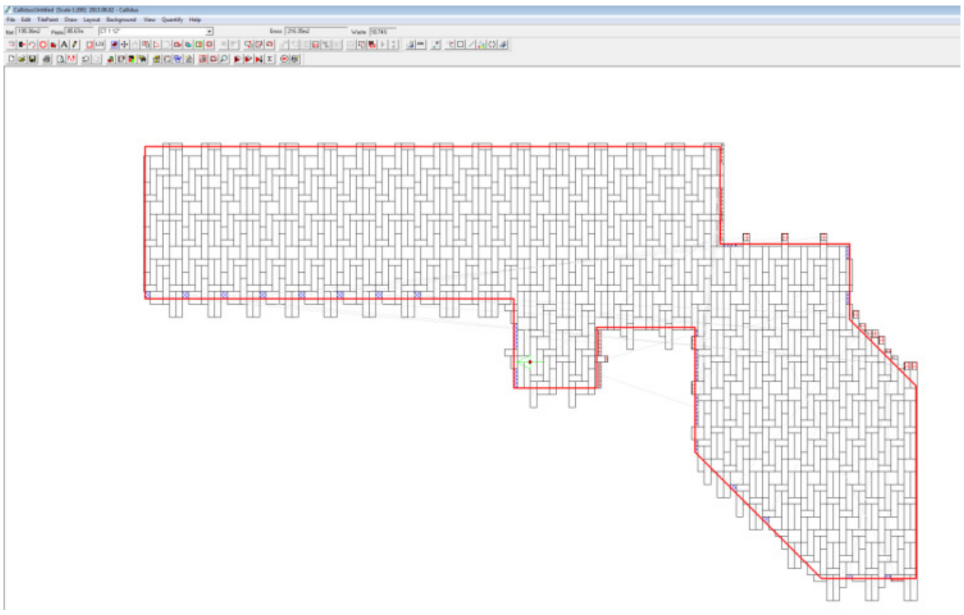
TILE 4 1 1 2
TILE 1 2 5 2
TILE 2 1 6 2
TILE 2 2 3 3
TILE 4 1 6 3
TILE 4 1 5 4
TILE 4 1 3 5
TILE 2 1 7 5
TILE 1 2 9 4

Here's the grid in the TileScript editor



```
UNITSIZE 9"

REP 11 6 0
TILE 4 1 0 0
TILE 4 1 0 1
TILE 2 2 4 0
TILE 4 1 6 0
TILE 4 1 6 1
TILE 1 2 10 0


REP 7 6 0
TILE 4 1 1 2
TILE 1 2 5 2
TILE 2 1 6 2
TILE 2 2 3 3

TILE 4 1 6 3
TILE 4 1 5 4

TILE      4 1 3 5

TILE      2 1 7 5
TILE      1 2 9 4
```

and used within a floor area



you can probably see that the hardest part of this process is to understand how the designer's tile grid is constructed and made to flow over the floor. In some cases this may take a while to finally figure it out but once that is fully understood, the script itself is relatively straightforward.

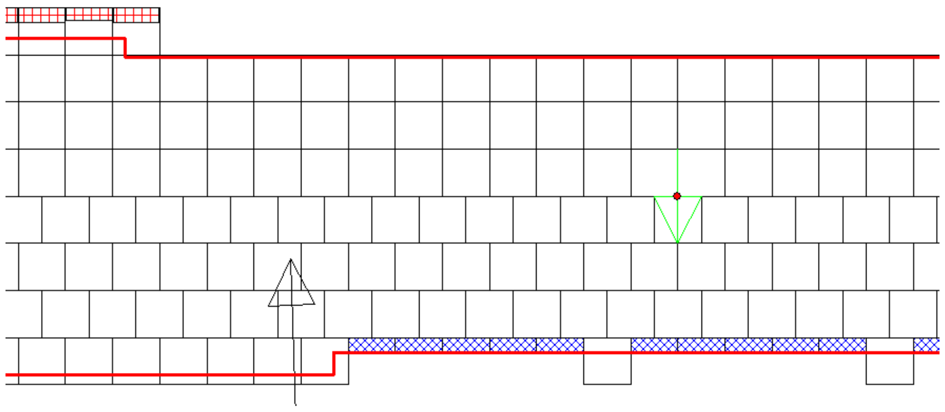
Setting the grid origin

Normally the grid is laid out such that 0,0 is mapped to the start point of the room. But you can change this using the command

ORIGIN x y

This can be useful where you have a grid that changes type and you want to control the position of that change

in this example the customer wants Monolithic tiles run down to about the mid-point of the corridor with an ashlar (brick bond) design filling out the lower part.



You can design the grid normally and then set the origin to be just under the last row of Monolithic tiles.

If you don't reset the Origin of the script then you may have to spend quite some time manually moving the start point to the position correct.

Often the command ORIGIN 0,y2 is used as this means keep the origin at 0 on the x but the grid will be positioned just under the last tile entered

Calculations within TileScript

Although users are free to use this feature, it is really to help with very complex grids which are normally pre-prepared and available from the TileScript Library webpage.

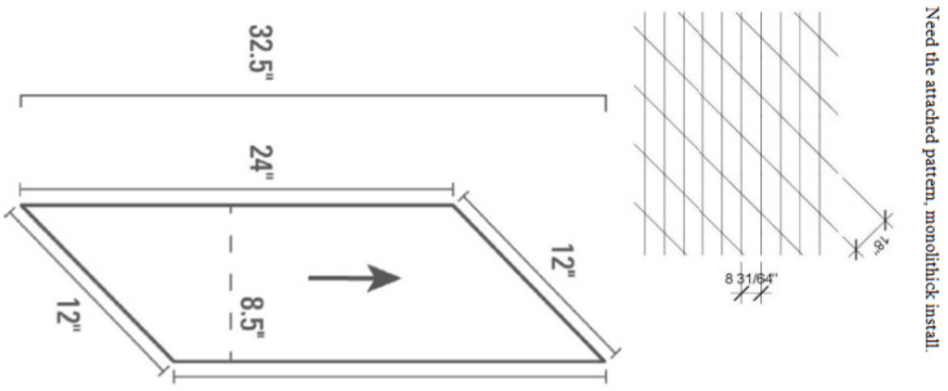
Calculated values can be introduced into TileScript and used within the actual tile definition. This is not something that is normally required but it can save a great deal of work when you encounter specific designs

The command to create a variable and set a value to it is

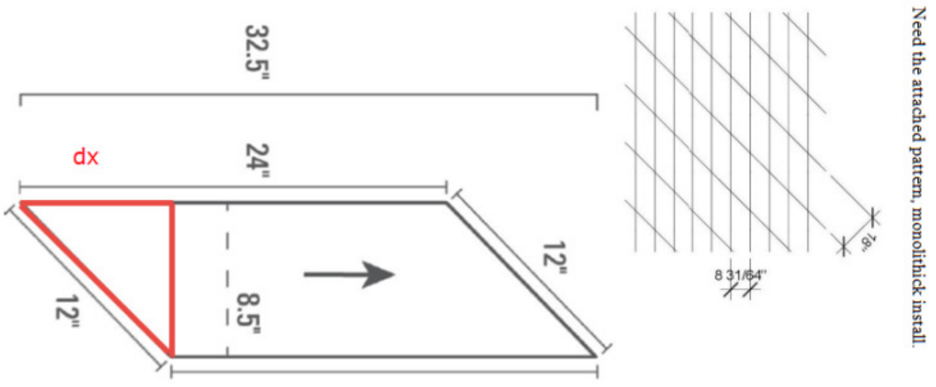
LET x=calculation.

(Note: TileScript is not a fully functioning programming language so in general try to keep things simple)

Take the following example



to break this up look at the right angled triangle at each end



using Pythagoras dx= sqrt(12^2-8.5^2)

so we can build the TileScript as follows

1. ***#set for inches***
2. ***unitsize 1"***
3. ***#resolve the triangle at the end of the tile***
4. ***let dx=sqrt(144-8.5*8.5)***
5. ***# set the repeat***
6. ***rep 24 8.5 -dx***
7. ***# build the trapezoid tile***
8. ***corners 0,0 24,0 24-dx,8.5 -dx,8.5***

line 4 is the calculation for dx

line 6 sets the repeat 24" across, 8.5" down , step back by dx

line 8 build the tile with the corners command

If you take out the comments this is just 4 lines of code to create the final tile.

1. ***unitsize 1"***
2. ***let dx=sqrt(144-8.5*8.5)***

```
rep 24 8.5 -dx
corners 0,0 24,0 24-dx,8.5 -dx,8.5
```

You can do the same within calculations but you would have to manually work out where all the corners were so it is far easier, quicker and safer to have Callidus do the work for you.

For even greater flexibility you can take the fixed numbers out of the maths and set them as variables

```
unitsize 1"
let long=24
let short=12
let ht=8.5

let dx=sqrt(short*short-ht*ht)
rep long ht  -dx
corners 0,0 long,0 long-dx,ht -dx,ht
```

Although it may take a little longer to set up you can now use the same script for any size tile just by changing one of the values in the first 4 lines. The rest of the script will take care of all the maths

Last tile position

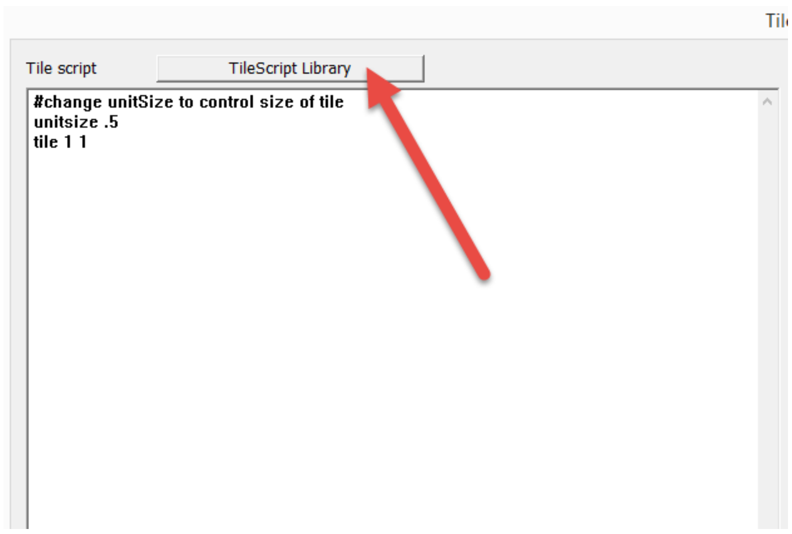
Each time a tile is added to the grid 4 variables are updated and can be used in calculations for the next file

x1,y1,x2,y2 represent the left, top, right, bottom positions of the last tile

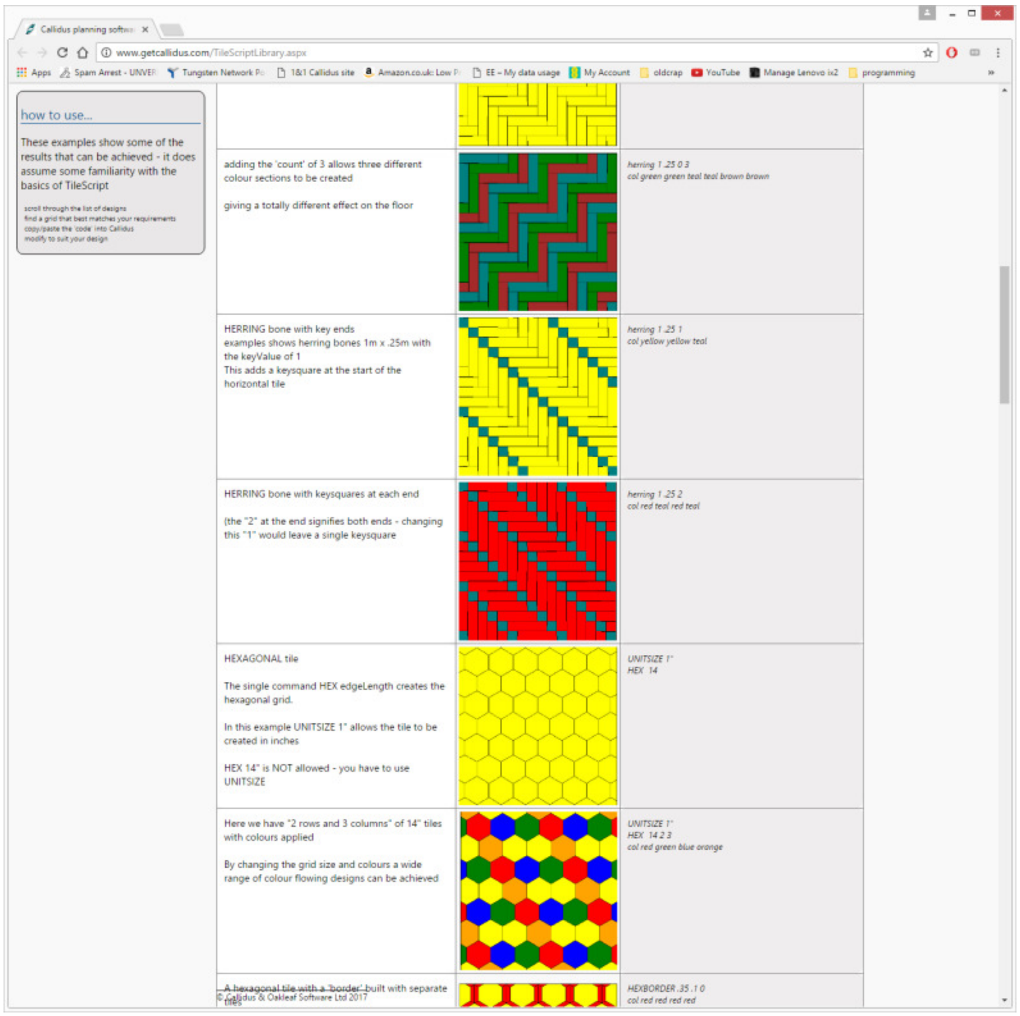
One consequence of this is that you cannot use x1,y1,x2 or y2 for your own variables.

Preassembled grids

On the TileScript editor there is a button labelled "TileScript Library". Press this and a web page will open showing a range of TileScript samples.



```
#change unitSize to control size of tile
unitsize .5
tile 1 1
```



how to use...

These examples show some of the results that can be achieved - it does assume some familiarity with the basics of TileScript

scroll through the list of designs
find a grid that best matches your requirements
copy/paste the 'code' into Callidus
modify to suit your design

adding the 'count' of 3 allows three different colour sections to be created giving a totally different effect on the floor		*herring 1 .25 0 3* *col green green teal teal brown brown*
HERRING bone with key ends examples shows herring bones 1m x .25m with the keyValue of 1 This adds a keysquare at the start of the horizontal tile		*herring 1 .25 1* *col yellow yellow teal*
HERRING bone with keysquares at each end (the "2" at the end signifies both ends - changing this "1" would leave a single keysquare		*herring 1 .25 2* *col red teal red teal*
HEXAGONAL tile The single command HEX edgeLength creates the hexagonal grid. In this example UNITSIZE 1" allows the tile to be created in inches HEX 14" is NOT allowed - you have to use UNITSIZE		*UNITSIZE 1"* *HEX 14*
Here we have "2 rows and 3 columns" of 14" tiles with colours applied By changing the grid size and colours a wide range of colour flowing designs can be achieved		*UNITSIZE 1"* *HEX 14 2 3* *col red green blue orange*
A hexagonal tile with a 'border' built with separate tiles		*HEXBORDER .35 .1 0* *col red red red red*

© Callidus & Oakleaf Software Ltd 2017

Scroll through the list until you find a grid similar to your requirements, then copy (select all and Ctrl+C) the text in the right hand cell - this contains the associated TileScript.

Return to Callidus, clear the existing script and paste(Ctrl+V) in the code from the website. Press F5 and you should see exactly the same grid now ready for us in Callidus.

Make any changes to UNITSIZE or other parameters to suit.

If you have a requirement that is not available please contact us and we will endeavour to provide a solution

Depreciated commands

over the development of TileScript some commands were added as experiments and have now been removed.

The following commands are no longer available

- ***BCOPY***
- ***COPY***
- ***LOCK***

If you have created previous scripts with these commands you may have to restructure them