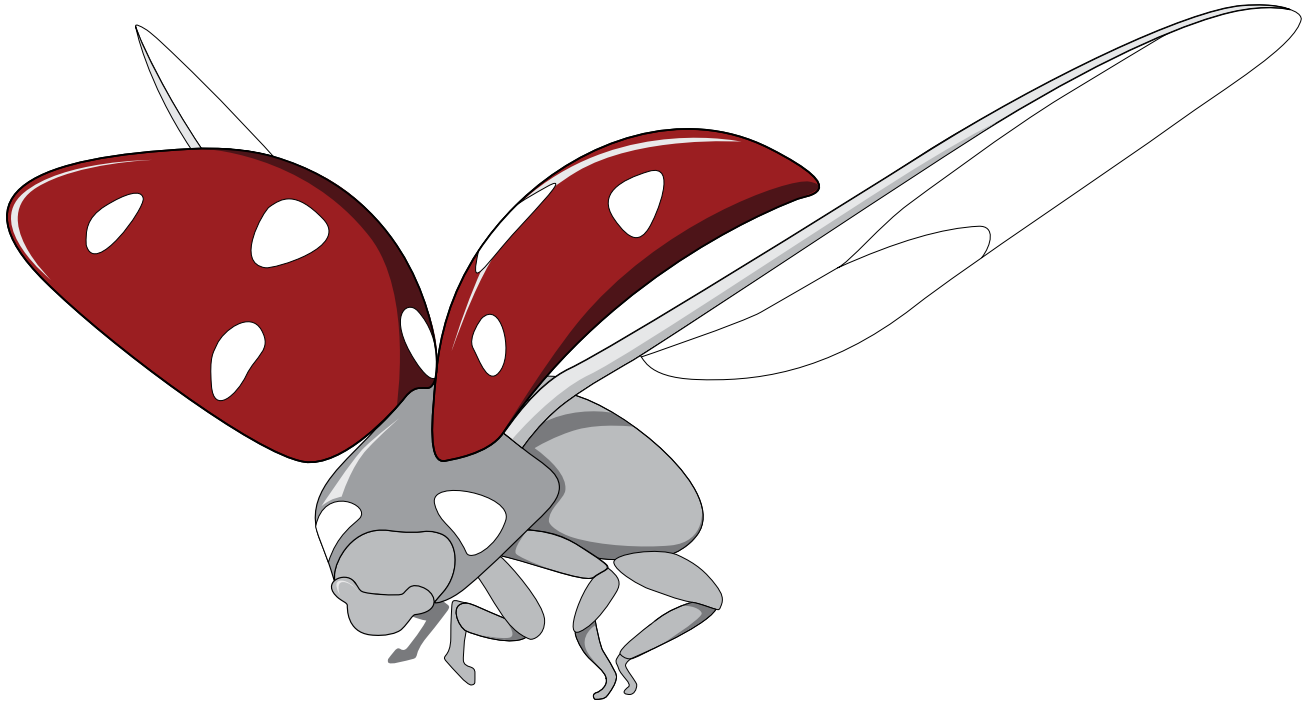


The **B**eetle **B**locks Primer

About

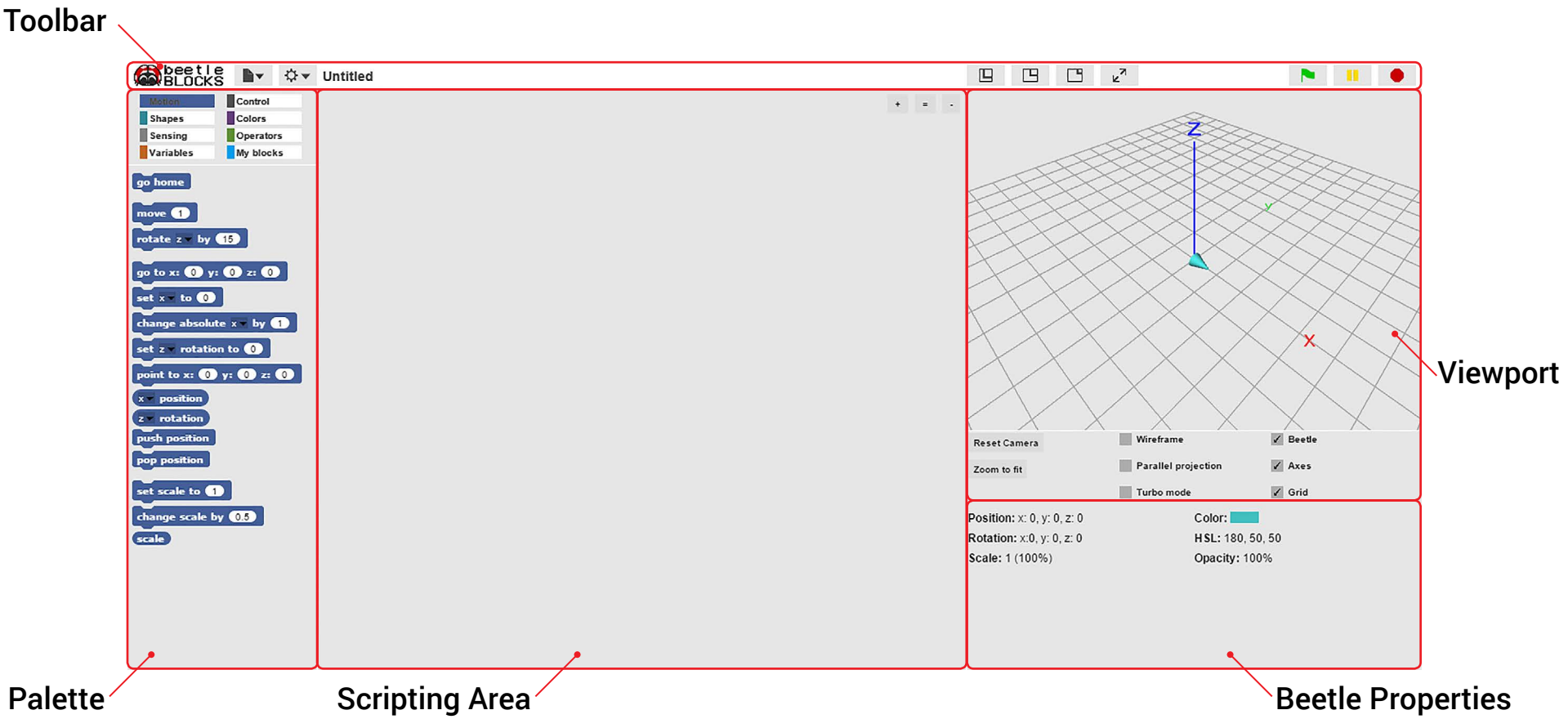
Beetle Blocks is a graphical blocks-based programming environment for 3D design and fabrication.
Use code to control a beetle that can place 3D shapes and extrude its path as a tube. Then make a 3D print!

By Eric Rosenbaum and Duks Koschitz, with software development by Bernat Romagosa and Jens Moenig.
Beetle Blocks is based on Scratch and Snap!

Contents

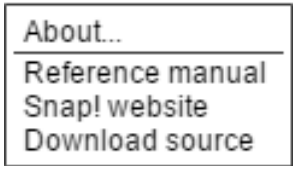
User Interface

In this section we describe in detail the various buttons, menus, and other clickable elements of the Beetle Blocks user interface.
Here is the map of the Beetle Blocks window:



Toolbar

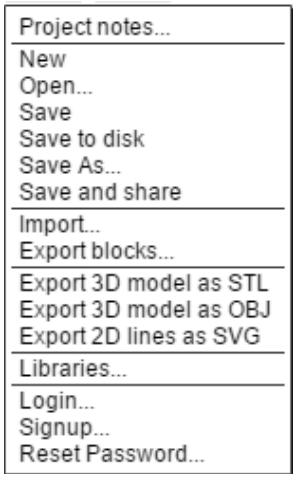
The Beetle Blocks logo menu



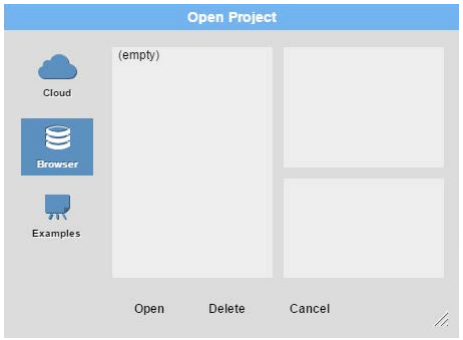
The Beetle Blocks logo at the left end of the tool bar is clickable.

The About option displays information about the Bittle Blocks itself, including version numbers for the source modules, the implementors, and the license (AGPL: you can do anything with it except create proprietary versions, basically).
The Reference manual option downloads a copy of the latest revision of this manual in PDF format.
The Snap! website option opens a browser window pointing to snap.berkeley.edu, the web site for Snap!.
The Download source option downloads, to your browser's usual download directory, a zip file containing the Javascript source files for the Beetle Blocks. You can read the code to learn how the Beetle Blocks is implemented, host a copy on your own computer (this is one way to keep working while on an airplane), or make a modified version with customized features. (However, access to cloud accounts is limited to the official version hosted at Berkeley.)

The file menu



The Project notes option opens a window in which you can type notes about the project: How to use it, what it does, whose project you modified to create it, if any, what other sources of ideas you used, or any other information about the project. This text is saved with the project, and is useful if you share it with other users.
The New option starts a new, empty project. Any project you were working on before disappears, so you are asked to confirm that this is really what you want (you should save the current project, if you want to keep it, before using New).
The Open… option shows a project open dialog box in which you can choose a project to open:



In this dialog, the three large buttons at the left select a source of projects: **Cloud** means your Beetle Blocks account's cloud storage; **Browser** means your browser's local store (data accessible only in that browser, on that computer, by that user); **Examples** means a collection of sample projects we provide.
The text box to the right of those buttons is an alphabetical listing of projects from that source; selecting a project by clicking shows its thumbnail (a picture of the stage when it was saved) and its project notes at the right.
The three buttons at the bottom select an action to perform on the selected project. The first button, **Open**, loads the project into Beetle Blocks and closes the dialog box. The second button is **Delete**, and if clicked it deletes the selected project. The **Cancel** button closes the dialog box without opening a project. (It does not undo any sharing, unsharing, or deletion you've done.)

The Save menu option saves the project to the same source and same name that was used when opening the project. (If you opened another user's shared project or an example project, the project will be saved to your own cloud account, if logged in, or browser localstore.)
The Save to disk menu
The Save as… menu option opens a dialog box in which you can specify where to save the project:
This is much like the Open dialog, except for the horizontal text box at the top, into which you type a name for the project. You can also share, unshare, and delete projects from here. If you are logged in, the dialog starts with Cloud selected; if not, Browser will be selected.
The Save and share menu
The Import… menu option is for bringing some external resource into the current project, rather than loading an entirely separate project. You can import block libraries (XML format, previously exported from Beetle Blocks itself). Using the Import option is equivalent to dragging the file from your desktop onto the Beetle Blocks window.
The Export blocks… option is used to create a block library. It presents a list of all the global blocks in your project, and lets you select which to export. It then opens a browser tab with those blocks in XML format, as with the Export project option. Block libraries can be imported with the Import option or by dragging the file onto the Beetle Blocks window.
The Export 3D model as STL menu
The Export 3D model as OBJ menu
The Export 2D lines as SVG menu
The Libraries… option presents a menu of useful, optional block libraries:

The libraries and their contents may change, but as of this writing **the Iteration, composition** library blocks are:
The cascade blocks take an initial value and call a function repeatedly on that value, f(f(f(f…(x)))).
The compose block takes two functions and reports the function f(g(x)).
The first three repeat blocks are variants of the primitive repeat until block, giving all four combinations of whether the first test happens before or after the first repetition, and whether the condition must be true or false to continue repeating. The last repeat block is like the repeat primitive, but makes the number of repetitions so far available to the repeated script.

The List utilities librar.
The append block takes any number of list inputs and reports a list of all the items of all the input lists. Reverse reports a list with the items of the input list in reverse order. Remove duplicates from reports a list in which no two items are equal. The sort block takes a list and a two-input comparison predicate, such as <, and reports a list with the items sorted according to that comparison. The other four blocks are versions of the list tools that provide a # variable containing the position in the input list of the currently considered item. This version of map also allows multiple list inputs, in which case the mapping function must take as many inputs as there are lists; it will be called with all the first items, all the second items, and so on.

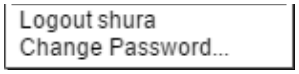
The Streams library.
Streams are a special kind of list whose items are not computed until they are needed. This makes certain computations more efficient, and also allows the creation of lists with infinitely many items, such as a list of all the positive integers. The first five blocks are stream versions of the list blocks in front of, item 1 of, all but first of, map, and keep. Show stream takes a stream and a number as inputs, and reports an ordinary list of the first n items of the stream. Finally, sieve is an example block that takes as input the stream of integers starting with 2 and reports the stream of all the prime numbers.

The Variadic reporters library.
These are versions of the associative operators +. ×, and, and or that take any number of inputs instead of exactly two inputs. As with any variadic input, you can also drop a list of values onto the arrowheads instead of providing the inputs one at a time.

The Words, sentences library.
These blocks make it more convenient to think of a text string either as a word made of letters or as a sentence made of words, rather than of individual characters. Each word of a sentence is a string of characters not including a space. That is, the sentence-related blocks split the string up at space characters. Any number of consecutive spaces counts as a single separator. The names should be self-explanatory except for the last block, which takes a text string as input and reports a list of one-character-long strings.

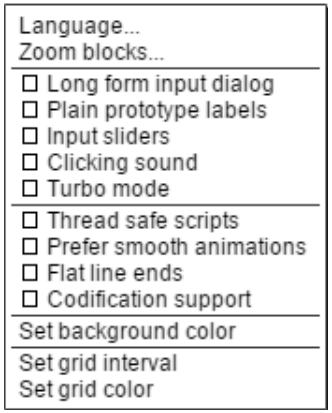
Choose **Login…** if you have a Beetle Blocks account and remember your password.
Choose **Signup…** if you don't have an account.
Choose **Reset Password…** if you've forgotten your password or just want to change it. You will then get an email, at the address you gave when you created your account, with a new temporary password. Use that password to log in, then you can choose your own password, as shown below.

If you are already logged in, you'll see this menu:



Logout is obvious, but has the additional benefit of showing you who's logged in.
Change password… will ask for your old password (the temporary one if you're resetting your password) and the new password you want, entered twice because it doesn't echo.

The settings menu



The Settings Menu shows a menu of Beetle Blocks options, either for the current project or for you permanently, depending on the option:

The Language… option lets you see the Beetle Blocks user interface (blocks and messages) in a language other than English. (Note: Translations have been provided by users. If your native language is missing, send us an email!)
The Zoom blocks... option lets you change the size of blocks, both in the palettes and in scripts. The standard size is 1.0 units. The main purpose of this option is to let you take very high-resolution pictures of scripts for use on posters. It can also be used to improve readability when projecting onto a screen while lecturing, but bear in mind that it doesn't make the palette or script areas any wider, so your computer's command-option-+ feature may be more practical. Note that a zoom of 2 is gigantic! Don't even try 10.

The remaining options let you turn various features on and off.

Long form input dialog, if checked, means that whenever a custom block input name is created or edited, you immediately see the version of the input name dialog that includes the type options, default value setting, etc., instead of the short form with just the name and the choice between input name and title text. The default (unchecked) setting is definitely best for beginners, but more experienced programmers may find it more convenient always to see the long form.

Plain prototype labels eliminates the plus signs between words in the Block Editor prototype block. This makes it harder to add an input to a custom block; you have to hover the mouse where the plus sign would have been, until a single plus sign appears temporarily for you to click on. It's intended for people making pictures of scripts in the block editor for use in documentation, such as this manual. You probably won't need it otherwise.

Input sliders provides an alternate way to put values in numeric input slots; if you click in such a slot, a slider appears that you can control with the mouse. The range of the slider will be from 25 less than the input's current value to 25 more than the current value. If you want to make a bigger change than that, you can slide the slider all the way to either end, then click on the input slot again, getting a new slider with a new center point. But you won't want to use this technique to change the input value from 10 to 1000, and it doesn't work at all for non-integer input ranges. This feature was implemented because software keyboard input on phones and tablets didn't work at all in the beginning, and still doesn't on Android devices, so sliders provide a workaround. It has since found another use in providing "lively" response to input changes; if Input sliders is checked, reopening the settings menu will show an additional option called Execute on slider change. If this option is also checked, then changing a slider in the scripting area automatically runs the script in which that input appears. The project live-tree in the Examples collection shows how this can be used; it features a fractal tree custom block with several inputs, and you can see how each input affects the picture by moving a slider. This option is per-project, not per-user.

Clicking sound causes a really annoying sound effect whenever one block snaps next to another in a script. Certain very young children, and our colleague Dan Garcia, like this, but if you are such a child you should bear in mind that driving your parents or teachers crazy will result in you not being allowed to use Snap!.

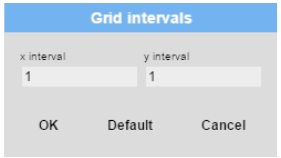
Turbo mode makes most projects run much faster, at the cost of not keeping the stage display up to date. (Beetle Blocks ordinarily spends most of its time drawing sprites and updating variable watchers, rather than actually carrying out the instructions in your scripts.) So turbo mode isn't a good idea for a project with glide blocks or one in which the user interacts with animated characters, but it's great for drawing a complicated fractal, or computing the first million digits of , so that you don't need to see anything until the final result. While in turbo mode, the button that normally shows a green flag instead shows a green lightning bolt. (But when clicked hat blocks still activate when the button is clicked.) Flat design changes the "skin" of the Beetle Blocks window to a really hideous design with white and pale-grey background, rectangular rather than rounded buttons, and monochrome blocks (rather than the shaded, somewhat 3D-looking normal blocks). The monochrome blocks are the reason for the "flat" in the name of this option. The only thing to be said for this option is that it may blend in better with the rest of a web page when a Snap! project is run in a frame in a larger page.

Thread safe scripts changes the way Beetle Blocks responds when an event (clicking the green flag, say) starts a script, and then, while the script is still running, the same event happens again. Ordinarily, the running process stops where it is, ignoring the remaining commands in the script, and the entire script starts again from the top. This behavior is inherited from Scratch, and some converted Scratch projects depend on it; that's why it's the default. It's also sometimes the right thing, especially in projects that play music in response to mouse clicks or keystrokes. If a note is still playing but you ask for another one, you want the new one to start right then, not later after the old process finishes. But if your script makes several changes to a database and is interrupted in the middle, the result may be that the database is inconsistent. When you select Thread safe scripts, the same event happening again in the middle of running a script is simply ignored. (This is arguably still not the right thing; the event should be remembered and the script run again as soon as it finishes. We'll probably get around to adding that choice eventually.)

Prefer smooth animations slows down frame rate (time between updates to the display) to a fixed 1/30 second, like Scratch. (Ordinarily Beetle Blocks updates the display as often as possible.) One reason you might want this, explaining the option's name, is to ensure that sprites move a constant distance from one redisplay to the next, so that motion seems smooth rather than jerky. Another reason is to ensure that your program runs at the same speed on different computers. A third reason, somewhat counterintuitive, is that a project with many scripts running at once (for example, one that clones sprites repeatedly) may run faster if more time is available for scripts because less time is spent on the display of the screen.

Flat line ends affects the drawing of thick lines (large pen width). Usually the ends are rounded, which looks best when turning corners. With this option selected, the ends are flat. It's useful for drawing a brick wall.

Codification support enables an experimental feature that can translate a Beetle Blocks project to a text-based (rather than block-based) programming language. The feature doesn't know about any particular other language; instead, you can provide a translation for each primitive block using these special blocks:
Using these primitive blocks, you can build a block library to translate into any programming language. Watch for such libraries to be added to our library collection (or contribute one). To see some examples, open the project "Codification" in the Examples project list. Edit the blocks map to Smalltalk, map to JavaScript, etc., to see examples of how to provide translations for blocks.



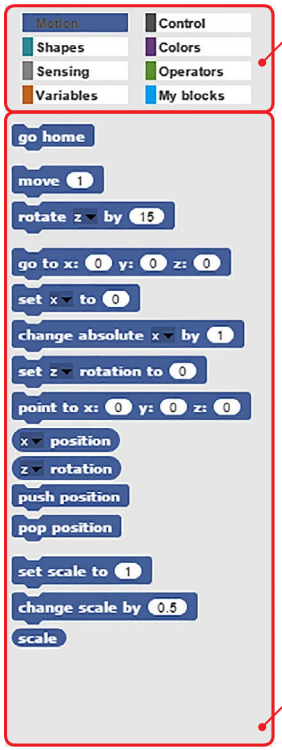
Set grid interval option gives you an opportunity to change the intervals of the grid in both X and Y axes.

Set background color and **Set grid color** options give you an opportunity to change the appearance of the Beetle Blocks Viewport.

Project control buttons

Above the right edge of the stage are three buttons that control the running of the project.
Technically, **the green flag** is no more a project control than anything else that can trigger a hat block: typing on the keyboard or clicking on a sprite. But it's a convention that clicking the flag should start the action of the project from the beginning. It's only a convention; some projects have no flag-controlled scripts at all, but respond to keyboard controls instead. Shift-clicking the button enters Turbo mode, and the button then looks like a lightning bolt: Shift-clicking again turns Turbo mode off.
Scripts can simulate clicking the green flag by broadcasting the special message __shout__go__ (two underscores in each of the three positions shown).
The pause button suspends running all scripts. If clicked while scripts are running, the button changes shape to become a play button: Clicking it while in this form resumes the suspended scripts. There is also a pause block in the Control palette that can be inserted in a script to suspend all scripts; this provides the essence of a breakpoint debugging capability.
The stop button stops all scripts, like the stop all block. It does not prevent a script from starting again in response to a click or keystroke; the user interface is always active.

The Blocks Palette

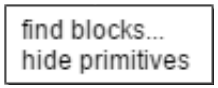


Blocks Categories

At the top of the palette area are the eight categories: Motion, Shapes, Sensing, Variables, Control, Colors, Operators, and My blocks. Each of these categories contain a blocks palette.

Context Menu for the Palette Background

Right-click/control-click on the grey background of the palette area shows this menu:

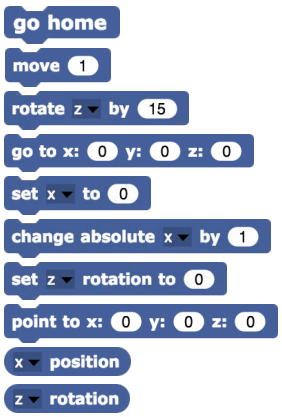


The find blocks... option.
The hide primitives option hides all of the primitives in that palette.

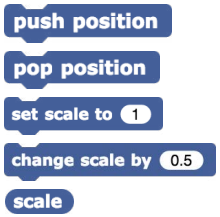
Blocks

Block categories

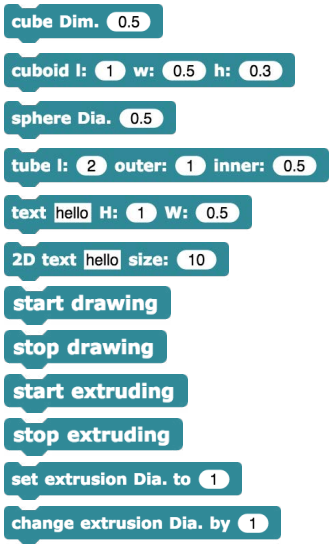
Motion - this category includes blocks that:



- send the Beetle to the origin without changing its color;
- move the Beetle to the set ammount of units;
- rotate the Beetle in chosen axis to the set angle;
- send the Beetle to certain point in space;



Shapes - Create a variety of shapes at the current location of the beetle.

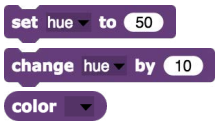


Sensing - Pull information from outside the software, implement user input.

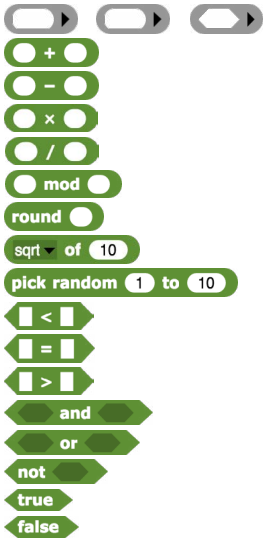
Variables - Define variables and create data lists.

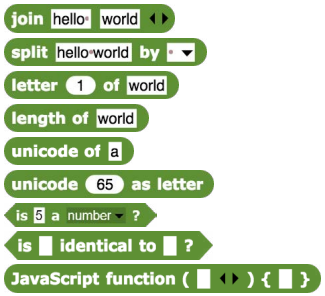
Control - Recursion tools - Create loops and conditional statements.

Colors - Change the colors of the pen, shapes, and extrusion



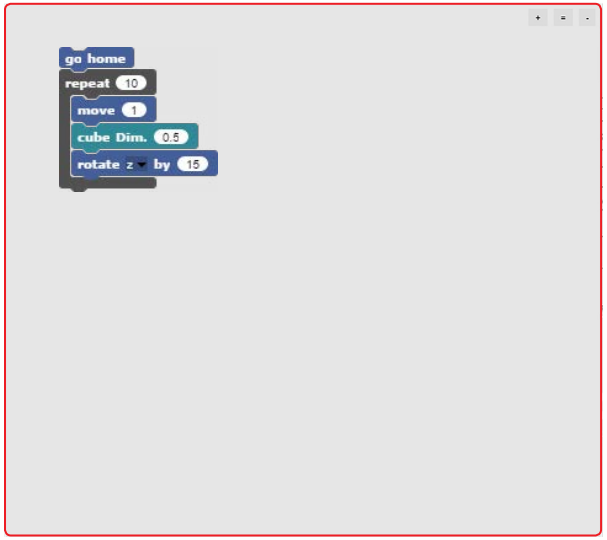
Operators - Use basic math and numeric functions





My Blocks - Create your own blocks using the previous categories

The Scripting Area



The scripting area is the middle region of the Beetle Blocks window, where you'll compile your blocks to make a script. If the script exceeds the scripting are, the panning option becomes available. In this case you can navigate the scripting area by pressing the middle button on your mouse.

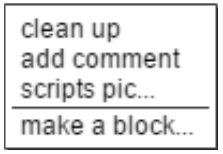
The three buttons in the top right corner allow you to zoom in or out to enlarge or shrink blocks in the scripting area.

You can drag and drop blocks from the library into this space.
When dragging blocks close to other blocks you'll be presented with the option to snap them together.
The order of the blocks matter - when executed, they run from top to bottom.

Clicking on a script runs it. The script will have a green "halo" around it while it's running. Clicking a script with such a halo stops the script. If a script is shown with a red halo, that means that an error was caught in that script, such as using a list where a number was needed, or vice versa. Clicking the script will turn off the halo

Scripting area background

Control-click/shift-click on the grey striped background of the scripting area gives this menu:



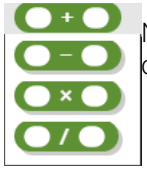
The clean up option rearranges the position of scripts so that they are in a single column, with the same left margin, and with uniform spacing between scripts. This is a good idea if you can't read your own project!
The add comment option puts a comment box like the picture below in the scripting area. It's attached to the mouse, as with duplicating scripts, so you position the mouse where you want the comment and click to release it. You can then edit the text in the comment as desired.

You can drag the bottom right corner of the comment box to resize it. Clicking the arrowhead at the top left changes the box to a single-line compact form, so that you can have a number of collapsed comments in the scripting area and just expand one of them when you want to read it in full. If you drag a comment over a block in a script, the comment will be attached to the block with a yellow line.
Comments have their own context menu, with obvious meanings:

The scripts pic… option opens a new browser tab with a picture of all scripts in the scripting area, just as they appear, but without the grey striped background. Note that "all scripts in the scripting area" means just the toplevel scripts of the current sprite, not other sprites' scripts or custom block definitions.
Finally, **the make a block…** option does the same thing as the "Make a block" button in the Variables palette. It's a shortcut so that you don't have to keep switching to Variables if you make a lot of blocks.

Control-click/right-clicking a primitive block within a script shows a menu like this one:

The **help…** option shows the help screen for the block, just as in the palette. The other options appear only when a block is right-clicked/control-clicked in the scripting area.
Not every primitive block has a **relabel…** option. When present, it allows the block to be replaced by another, similar block, keeping the input expressions in place. For example, here's what happens when you choose relabel… for an arithmetic operator:

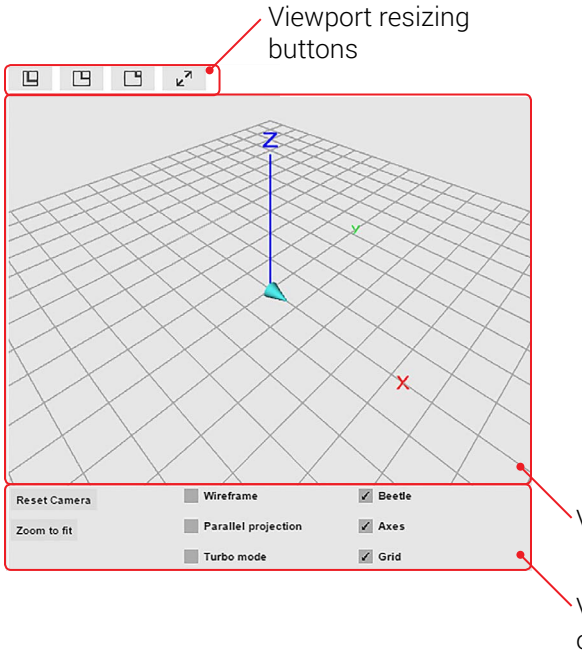


Note that the inputs to the existing – block are displayed in the menu of alternatives also. Click a block in the menu to choose it, or click outside the menu to keep the original block.

The duplicate option makes a copy of the entire script starting from the selected block. The copy is attached to the mouse, and you can drag it to another script (or even to another Block Editor window), even though you are no longer holding down the mouse button. Click the mouse to drop the script copy.
The block picture underneath the word duplicate is another duplication option, but it duplicates only the selected block, not everything under it in the script. Note that if the selected block is a C-shaped control block, the script inside its C-shaped slot is included. If the block is at the end of its script, this option does not appear.
The delete option deletes the selected block from the script.

The script pic… option opens a new browser tab containing a picture of the entire script, not just from the selected block to the end. You can use the browser's Save feature to put the picture in a file.
If the script does not start with a hat block, or you clicked on a reporter, then there's one more option: **ringify** or, if there is already a grey ring around the block or script, **unringify**. Ringify surrounds the block (reporter) or the entire script (command) with a grey ring, meaning that the block(s) inside the ring are themselves data, as an input to a higher order procedure, rather than something to be evaluated within the script.

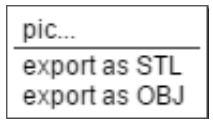
The Viewport Area



The viewport is your view into the beetle's 3D world.
You can navigate this space with your mouse
Drag across the 3D view to rotate it - Scroll to zoom in or out.
Hold down the shift key and drag to pan side to side or up and down.

In the Beetle's 3D world, there are 3 axes, shown as red, green, and blue lines pointing in different directions.
X is red - Y is green - Z is blue
The gray grid is drawn on the X and Y axes. The Z axis points up away from the grid. The Beetle starts out pointing along the X axis.

Control-clicking/right-clicking on the viewport background shows the stage's own context menu:

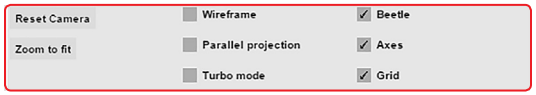


The pic… option opens a browser tab with a picture of everything on the stage. What you see is what you get. (If you want a picture of just the background, select the stage, open its costumes tab, control-click/right-click on a background, and export it.)
The stage's **export as STL** option
The **export as OBJ** option

Viewport resizing buttons

In the tool bar, but above the left edge of the stage, are four buttons that change the size of the viewport. Clicking any of the first three buttons changes the size of the viewport according to the icon it shows.
The next presentation mode button normally looks like this: . Clicking the button enlarges the viewport and eliminates the other user interface elements. When you open a shared project using a link someone has sent you, the project starts in presentation mode. While in presentation mode, the button looks like this: . Clicking it returns to normal (project development) mode.

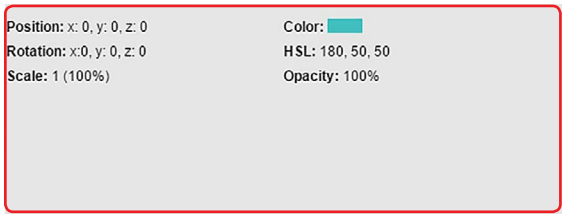
Viewport options



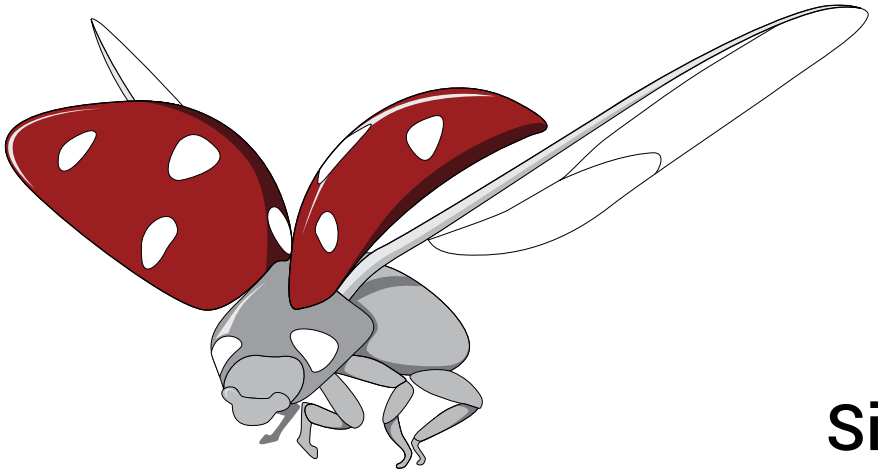
Under the viewprot area there is a **viewport control area** that allow you to change the appearance of the viewport.
Click **Reset Camera** to reset, or **Zoom to fit** to see all the shapes you have created.
You can also toggle a few viewport options:

Wireframe will change the viewport display mode to a wireframe
Parallel Projection will change the viewport to a 2d top view
Turbo mode executes the script faster, prioritizing speed over the live preview
Beetle, Axes, and **Grid** allow you to toggle the display of these options

Beetle Properties



State indicators are lables under the viewport area of the Beetle Blocks window, that give you all necessary information about the current position, color, scale and orientation of the Beetle in relation to the wolrd origin.



Simple geometric shapes

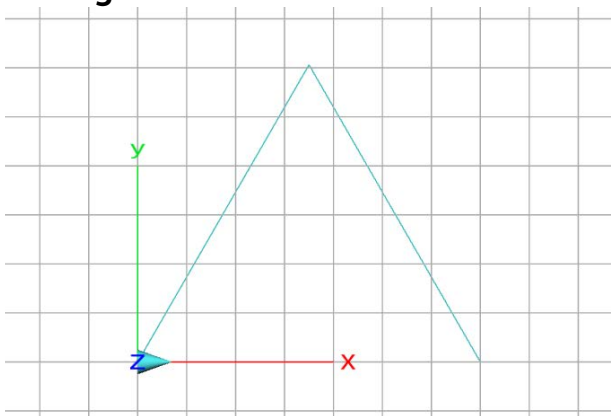
2D

Triangle
Square
Circle
Polygon

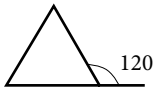
3D

Cubes
Drawing with cuboids
Pyramid
Octahedron
Sphere

Triangle

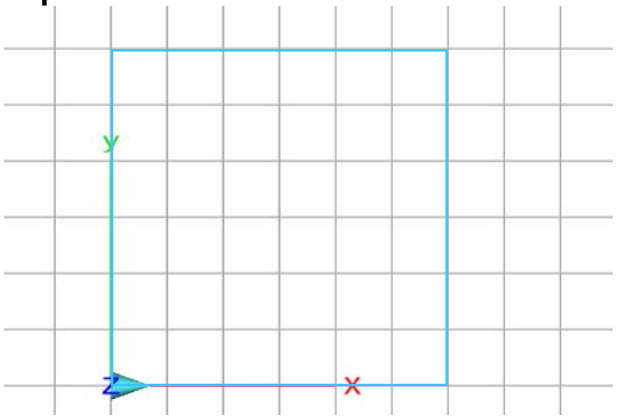


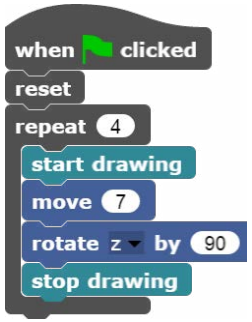
The equilateral triangle was created by moving the Beetle 7 units and rotating it by 120 degrees.



Commands are repeated 3 times to create 3 edges of the triangle.

Square

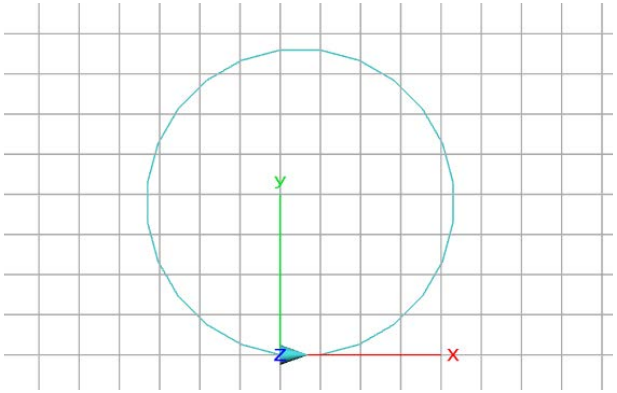


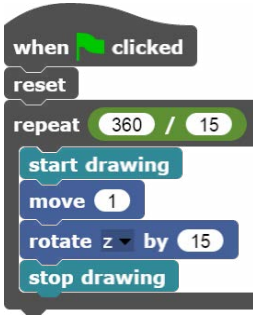


The square was created by moving the Beetle 7 units and rotating it by 90 degrees.

Commands are repeated 4 times to create 4 edges of the square.

Circle

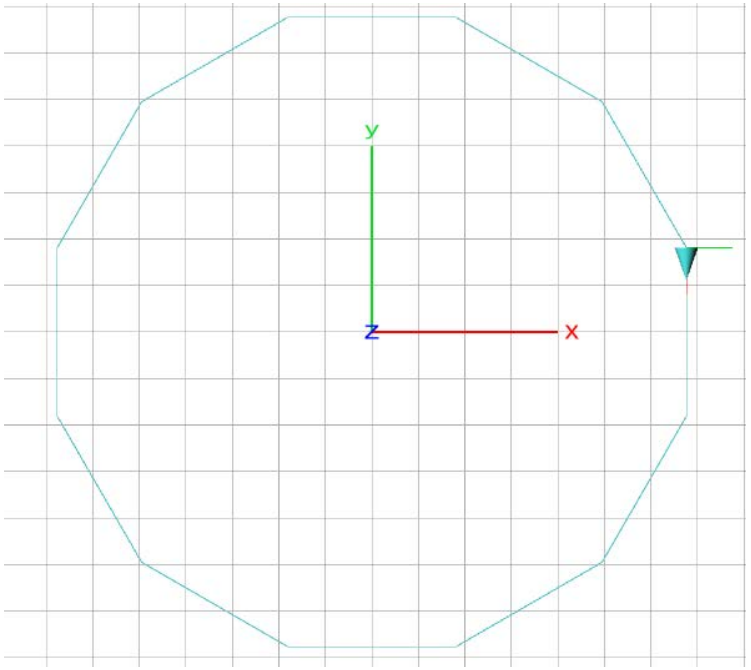


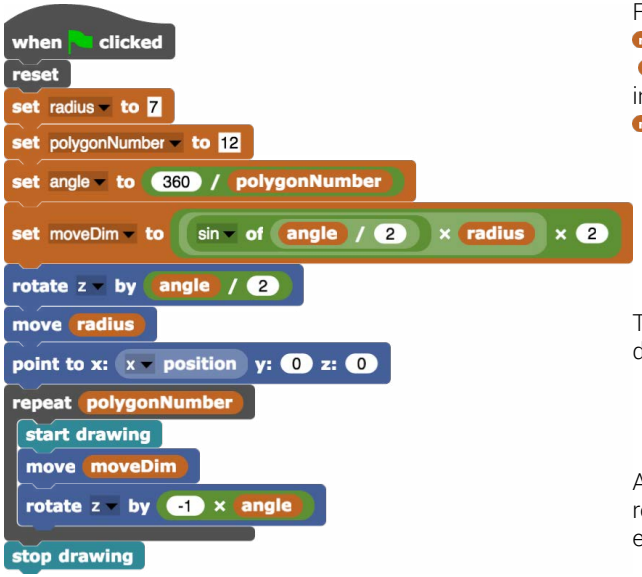


The circle was created by moving the Beetle 1 unit and rotating it by 15 degrees.

Commands are repeated 360 / 15 times, where 360 is the whole circle and 15 is the angle the Beetle was rotated.

Polygon



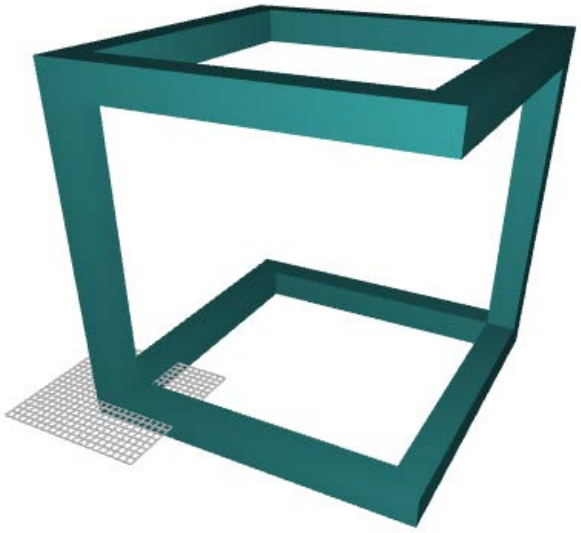


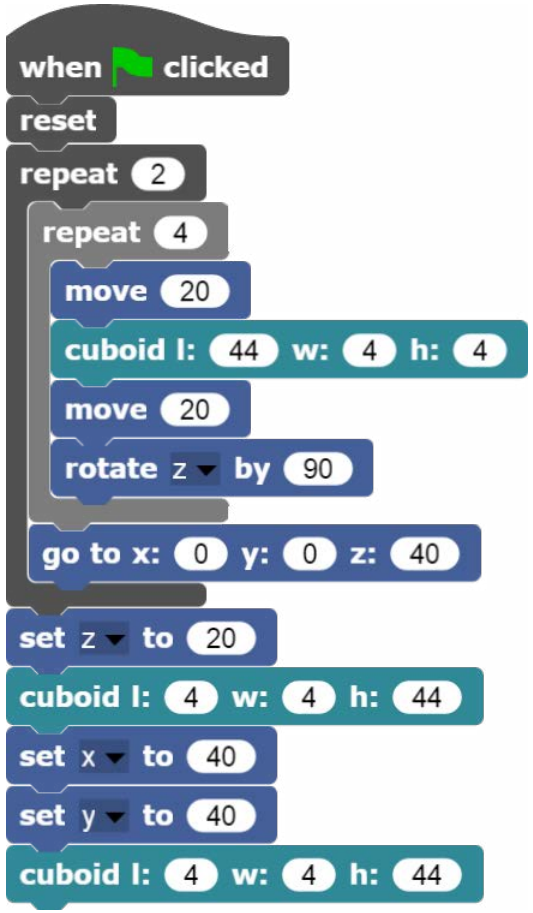
First, the variables: radius, polygonNumber, angle, moveDim are set, where
angle is the angle betwee two radiuses connecting the end points of one polygon
moveDim is the length of the polygon.

Then, the Beetle was placed in the corner of the dodecagon and pointed to X axis.

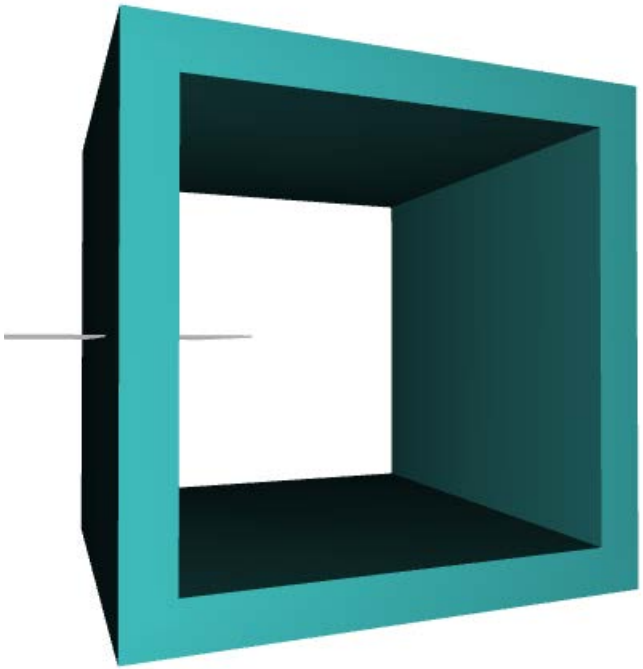
After the first polygon is drawn the Beetle was rotated by the -angle. The commands are repeated polygonNumber times to create a closed shape.

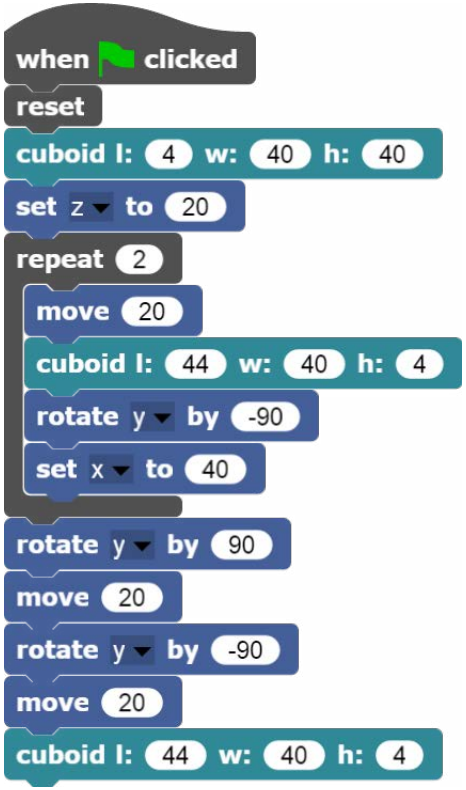
Cube as Line



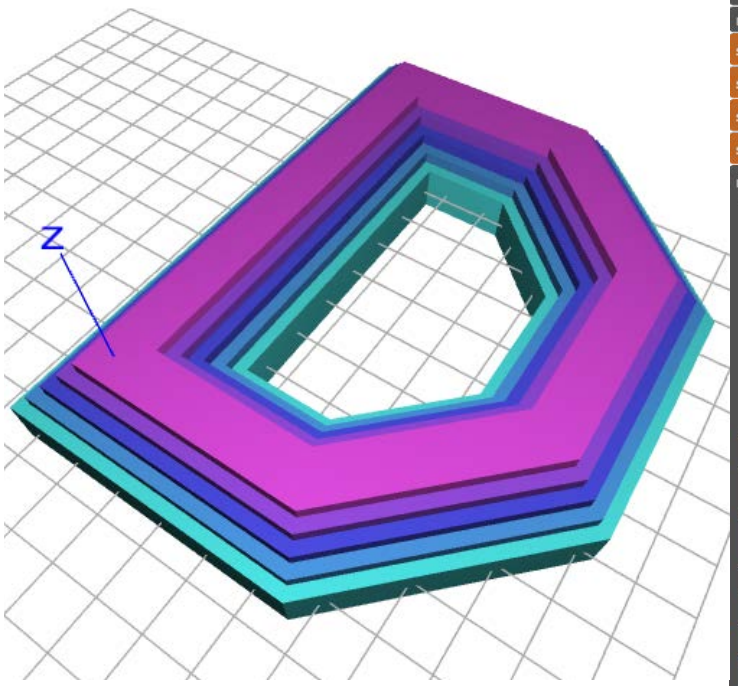

when clicked
reset
repeat 2
repeat 4
move 20
cuboid l: 44 w: 4 h: 4
move 20
rotate z by 90
go to x: 0 y: 0 z: 40
set z to 20
cuboid l: 4 w: 4 h: 44
set x to 40
set y to 40
cuboid l: 4 w: 4 h: 44

Cube as Surface





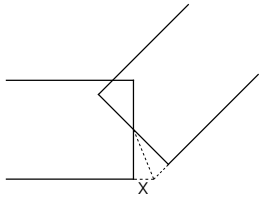
Drawing With Cuboids




```
when clicked
reset
set length to 4
set angle to 45
set width to 3
set height to 1
repeat 5
  repeat 5
    move length / 2
    cuboid l: length + tan of angle / 2 × width w: width h: height
    move length / 2
    rotate z by angle
  point to x: 0 y: 0 z: 0
  move y position / 2
  cuboid l: y position × 2 + width w: width h: height
  go home
  change hue by 30
  change width by -0.5
  change height by 0.5
```

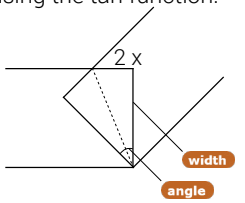
Then difficulty of this example is to align the corners of the shape.

Cuboids are drawn from the center, therefore the first step is to move the Beetle to the half of the length of the cuboid. If we placed the cuboid with length equals **length**, then the corner would look like this:



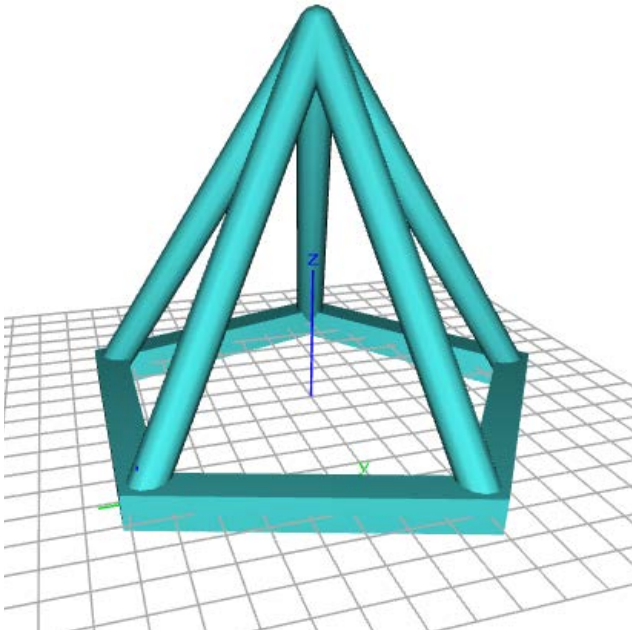
In order to align the corners we need to add X on the both sides of the cuboid. The length of the cuboid therefore equals **length** + 2X.

The 2x is a cathetus of the right triangle which we can define using the tan function:



where 2X equals `tan of angle / 2 × width`

Pyramid



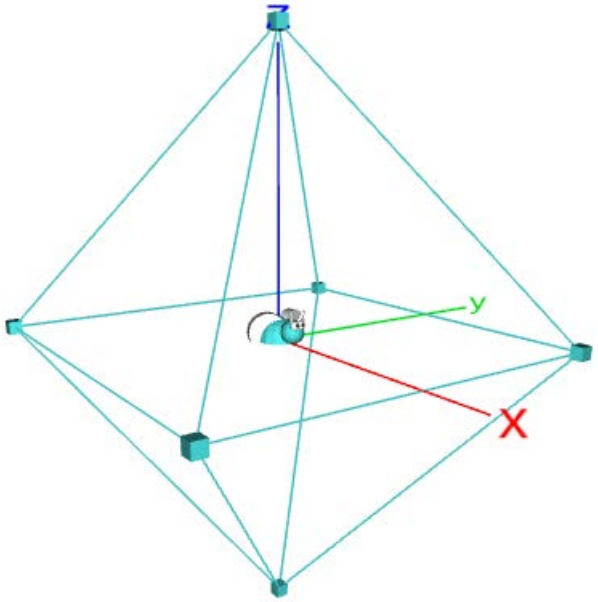
```
when clicked
reset
set width to 1
set height to 1
set radius to 7
set polygonNumber to 5
set angle to 360 / polygonNumber
set moveDim to sin of angle / 2 x radius x 2
rotate z by angle / 2
move radius
point to x: x position y: 0 z: 0
repeat polygonNumber
    move moveDim / 2
    cuboid l: moveDim + tan of angle / 2 x width w: width
    h: height
    move moveDim / 2
    push position
    point to x: 0 y: 0 z: 10
    start extruding
    move sqrt of 100 + radius x radius
    stop extruding
    pop position
    rotate z by angle x -1
```

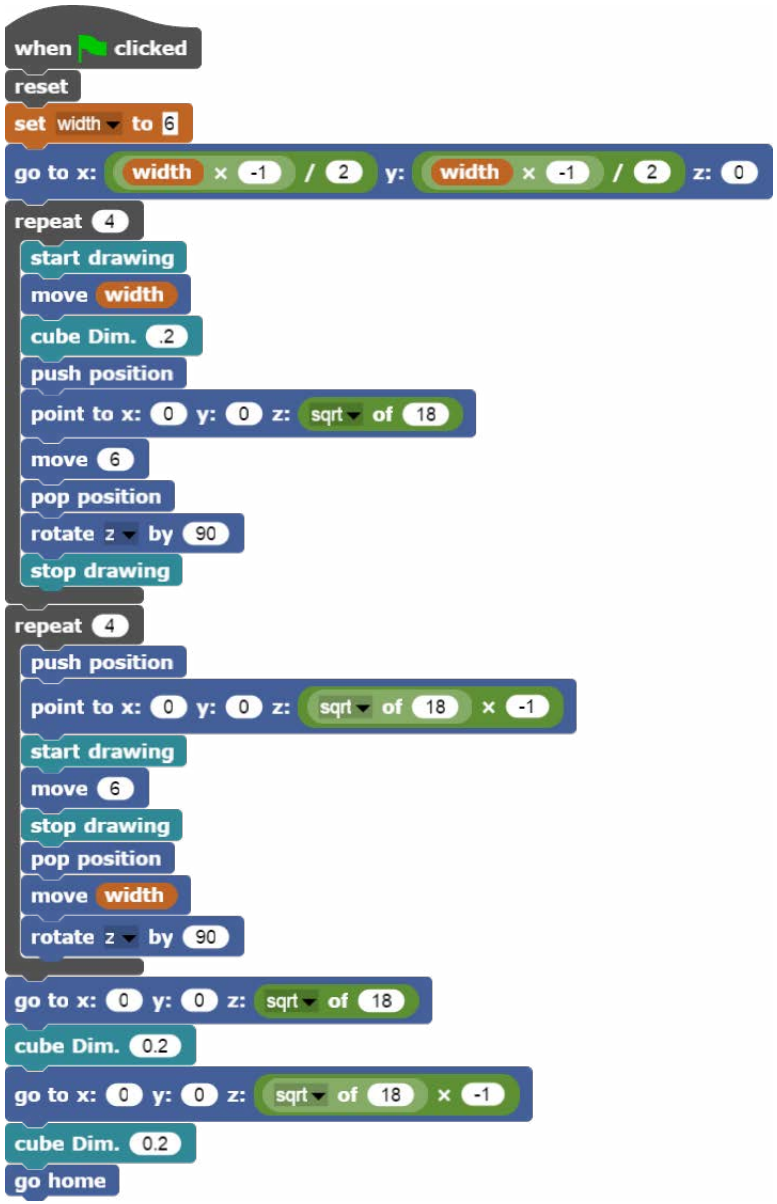
The base was created using the same set of commands that were used in Polygon example.=

Similar to the example of octahedron, we remember every point of the base by using `push position` block. We draw the verical edge of the pyramyd from a corner and and go back to the original point of it.

Repeat commands `polygonNumber` times to create closed shape.

Octahedron



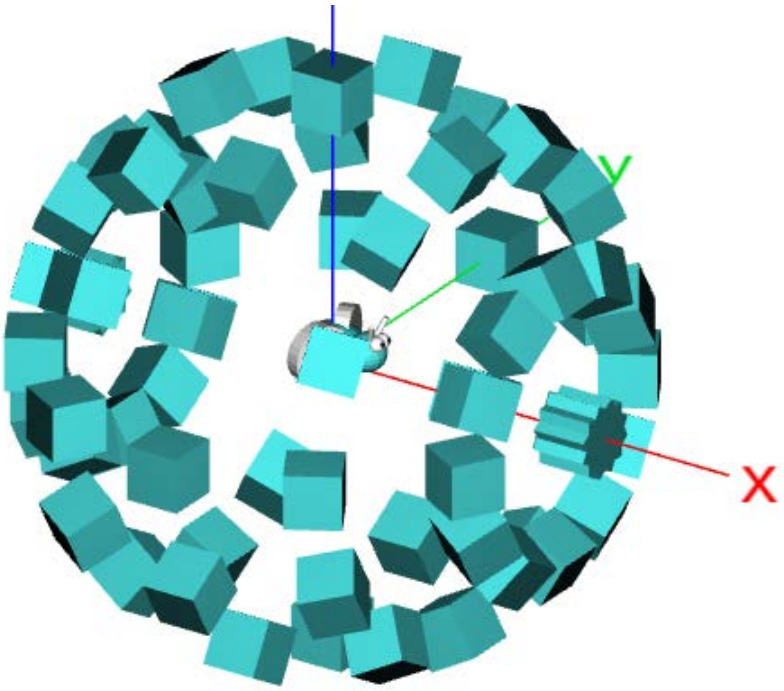


First, we set the length of the edge of the octahedron and place the Beetle in one of the corners of it.

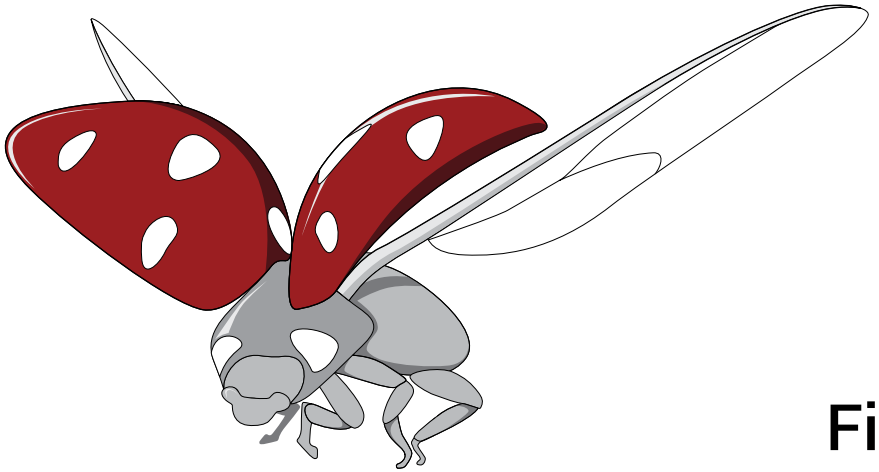
Then, we draw the horisontal edge of the shape, remember the end point of it by using push position block, draw an angled edge of the octahedron and come back to the point. Rotate the beetle by 90 degrees and repeat these commands 4 times to create the top half of the octahedron.

The same commmands were repeted in opposite direction to create the second half of the shape.

Sphere



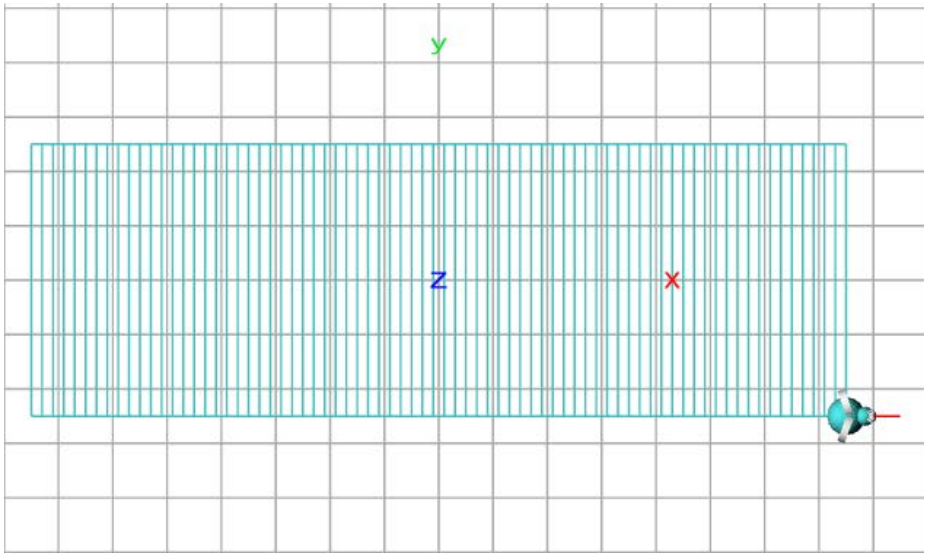

```
reset
sphere points lats: 12 longs: 12 diam: 5
    cube Dim. 0.5
    wait 0.05 secs

+sphere+points+lats:+ lats = 8 +longs:+ longs = 10 +diam:+ diam = 5
+ blocks λ +
push position
repeat lats / 2
    push position
    repeat longs
        rotate z by 360 / longs
        push position
        move diam / 2
        run blocks
        pop position
    pop position
    rotate x by 360 / lats
pop position
```

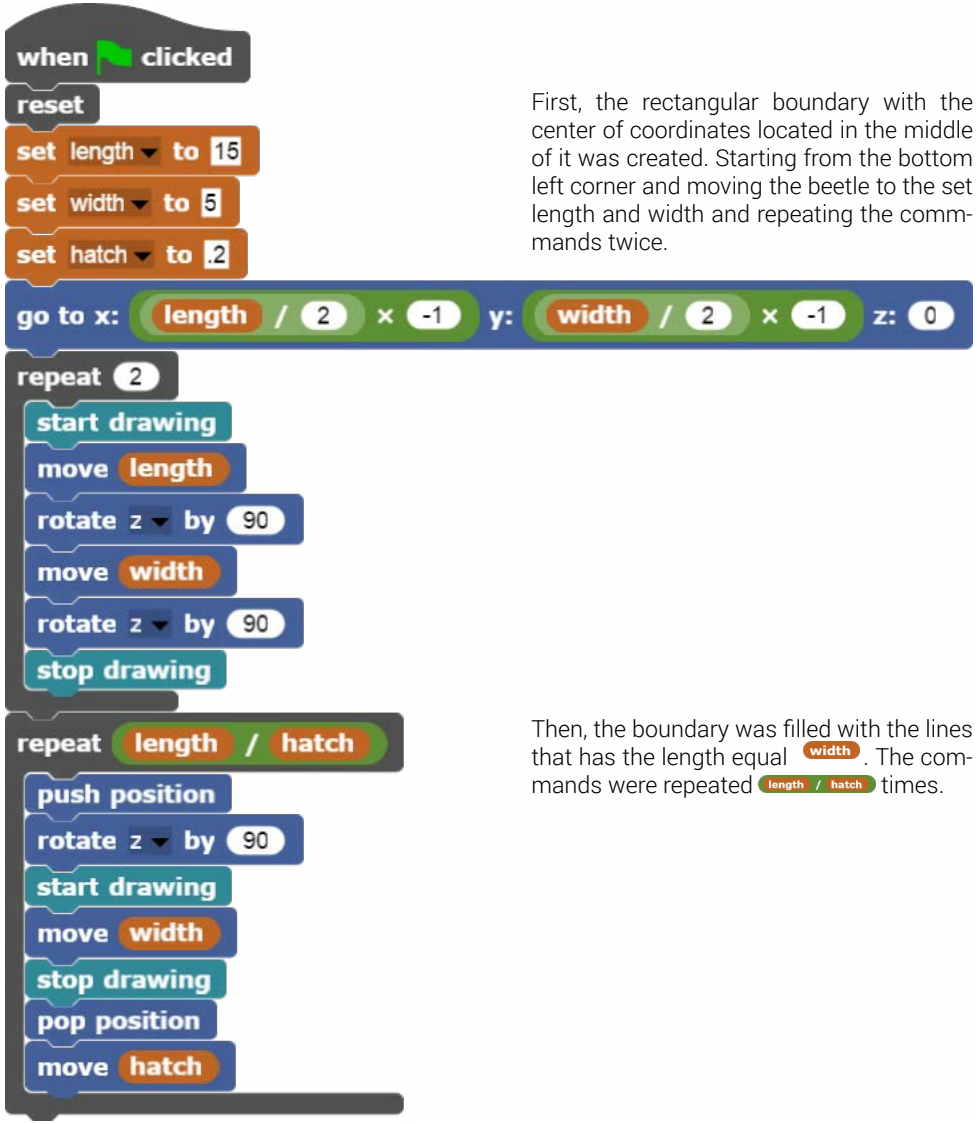


Filled shapes

Rectangle
Circle
Triangle
Drawing with hatches
Twisting hatch

Filled Rectangle

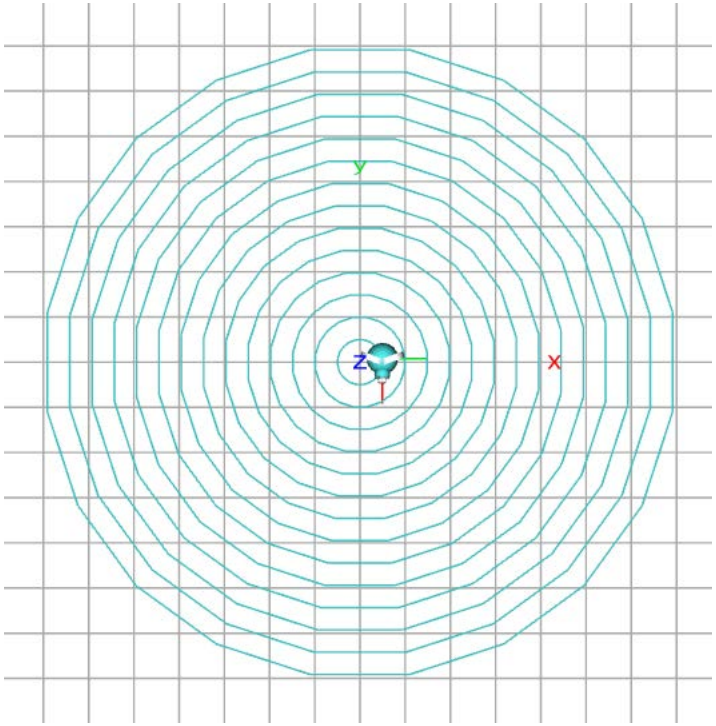




First, the rectangular boundary with the center of coordinates located in the middle of it was created. Starting from the bottom left corner and moving the beetle to the set length and width and repeating the commmands twice.

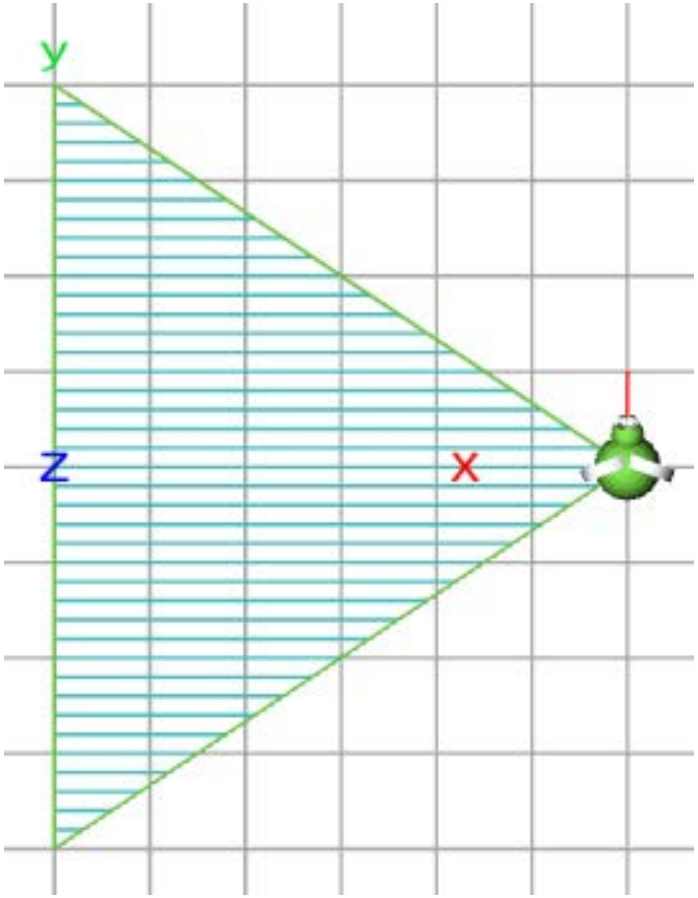
Then, the boundary was filled with the lines that has the length equal width. The commands were repeated length / hatch times.

Filled Circle



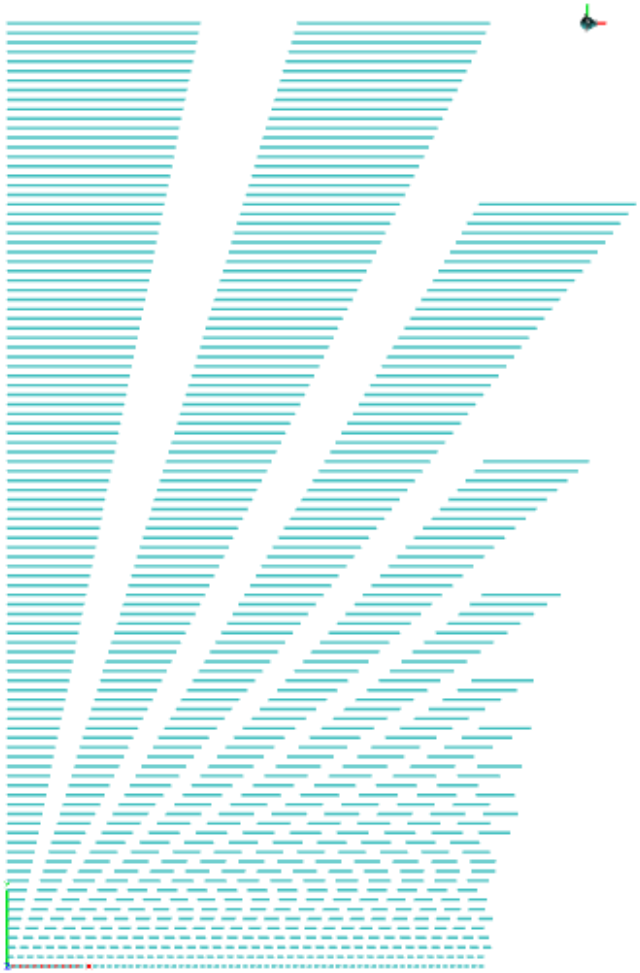

```
when clicked
reset
set radius to 7
set polygonNumber to 20
set angle to 360 / polygonNumber
set moveDim to sin of angle / 2 × radius × 2
repeat radius / .5
    go home
    rotate z by angle / 2
    move radius
    point to x: x position y: 0 z: 0
    repeat polygonNumber
        start drawing
        move moveDim
        rotate z by angle × -1
        stop drawing
    change radius by -.5
    set moveDim to sin of angle / 2 × radius × 2
```

Filled Triangle



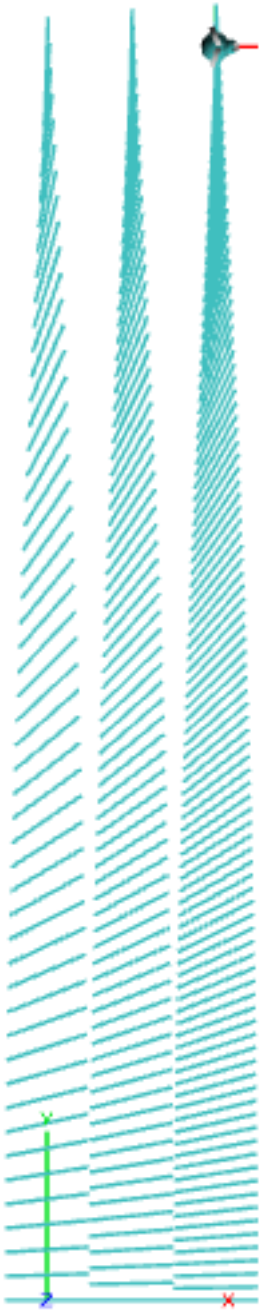

```
when clicked
reset
set width to 4
set length to 6
set hatch to .2
script variables a
set a to length
repeat width / hatch
    start drawing
    move a
    move a x -1
    rotate z by 90
    move hatch
    rotate z by -90
    change a by length x hatch / width x -1
    stop drawing
go to x: length y: 0 z: 0
set a to length
repeat width / hatch
    start drawing
    move a x -1
    rotate z by -90
    move hatch
    rotate z by 90
    change a by length x hatch / width x -1
    move a
    stop drawing
set hue to 100
push position
start drawing
go to x: length y: 0 z: 0
pop position
rotate z by 90
move width x 2
go to x: length y: 0 z: 0
```

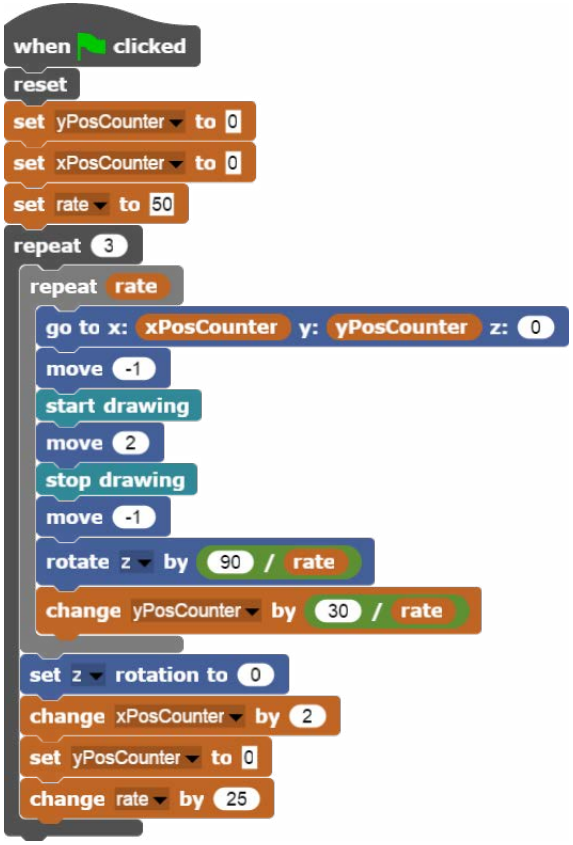
Drawing With Hatches

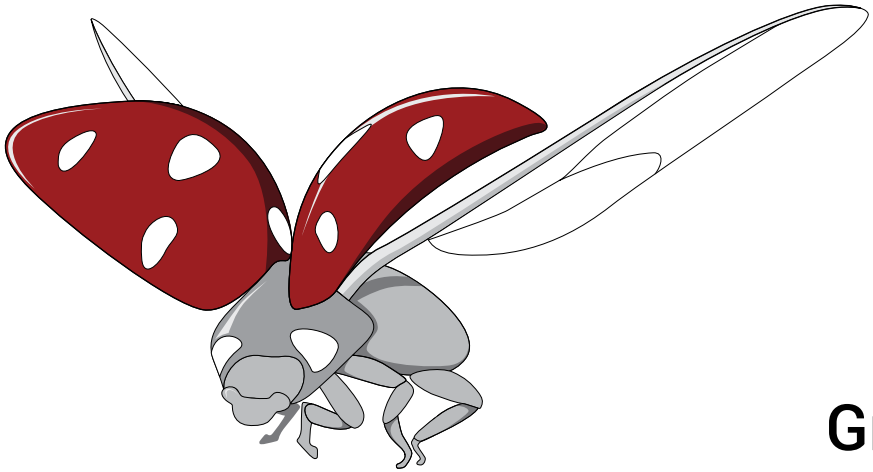


```
when clicked
reset
set columnCounter to 0
set lineLength to .25
repeat 100
  go to x: 0 y: columnCounter z: 0
  repeat until x position = 25 or x position > 25
    start drawing
    move lineLength
    stop drawing
    move lineLength / 2
  change columnCounter by .5
  change lineLength by .1
```

Twisting Hatch







Grids

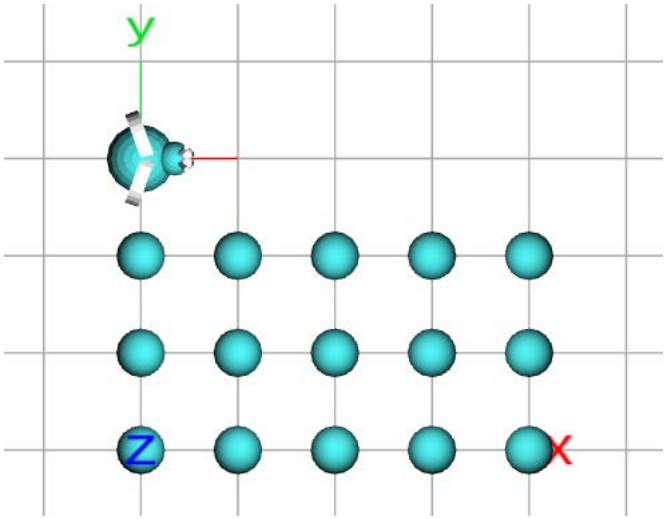
2D

Even grid
Staggered grid

3D

3d grid

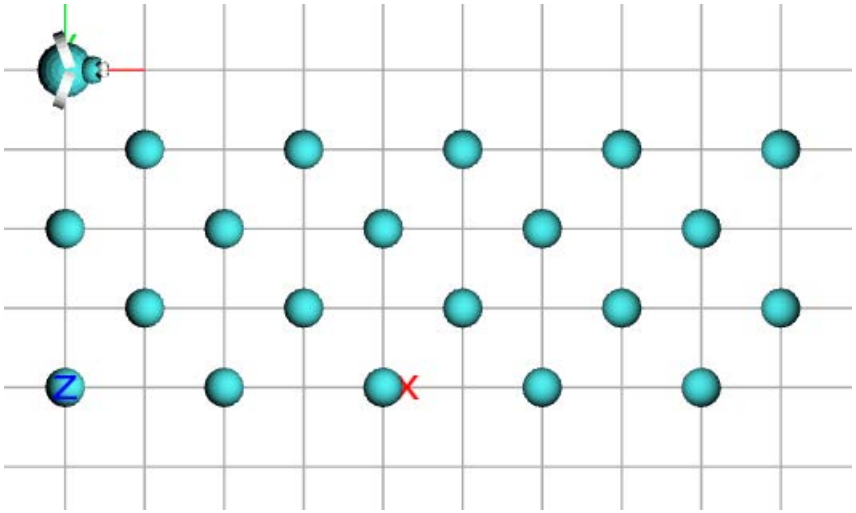
Even Grid

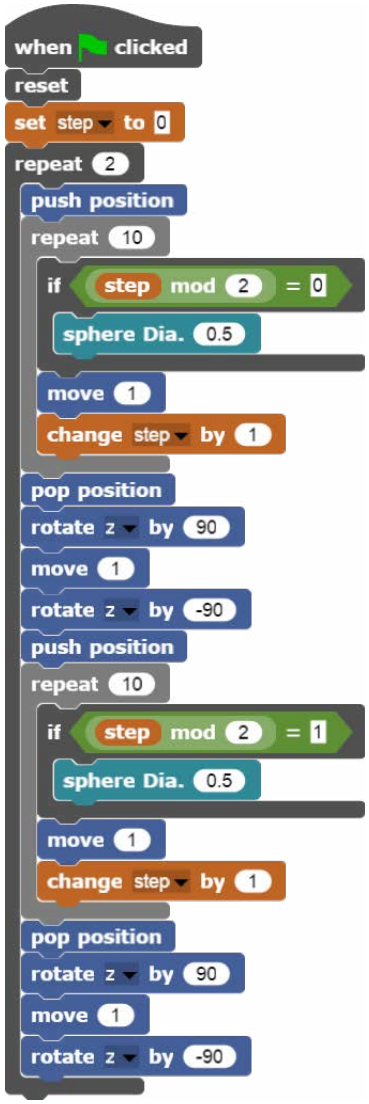


First, the length and the width of the boundary were set. The first line of spheres is placed horisontally in 1 unit distanse from each other.

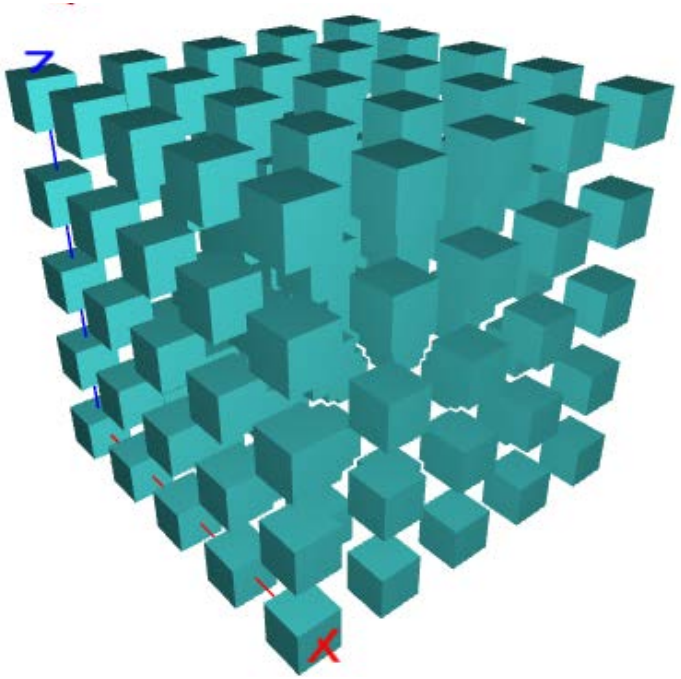
Then the Beetle was moved to the push position and moved 1 unit vertically to create the next line of circles.

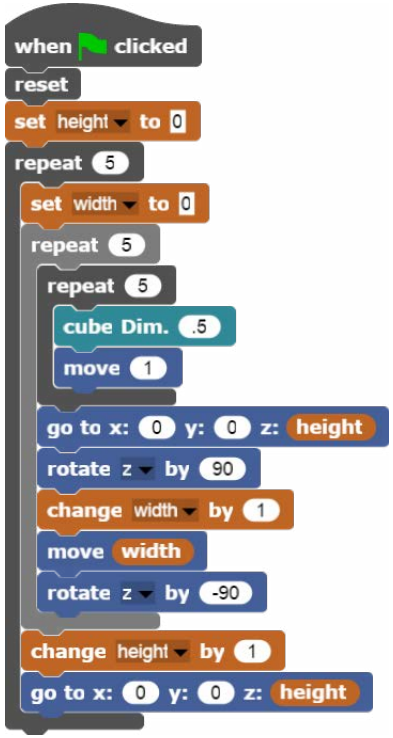
Staggered Grid

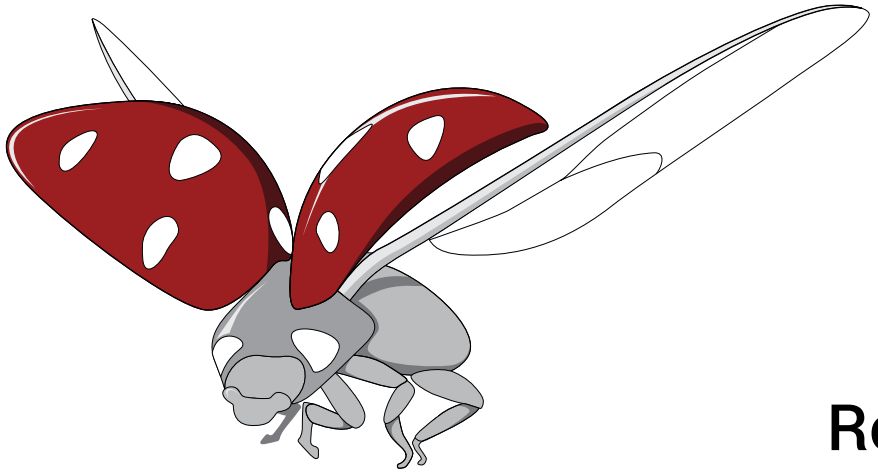




3D Grid




when clicked
reset
set height to 0
repeat 5
set width to 0
repeat 5
repeat 5
cube Dim. .5
move 1
go to x: 0 y: 0 z: height
rotate z by 90
change width by 1
move width
rotate z by -90
change height by 1
go to x: 0 y: 0 z: height



Rotational patterns and models

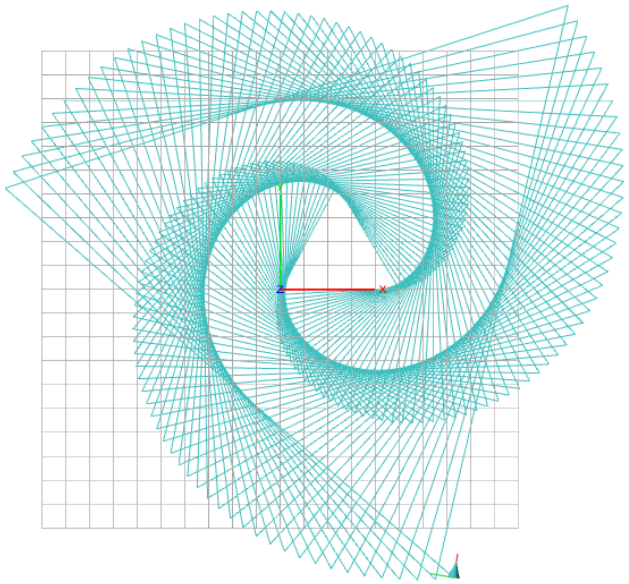
2D

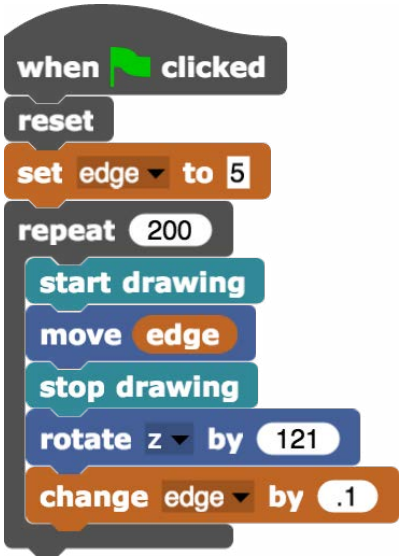
Rotated triangles
Spirals
Rotational pattern with rectangles

3D

Rotated loops
Spirals
Shells

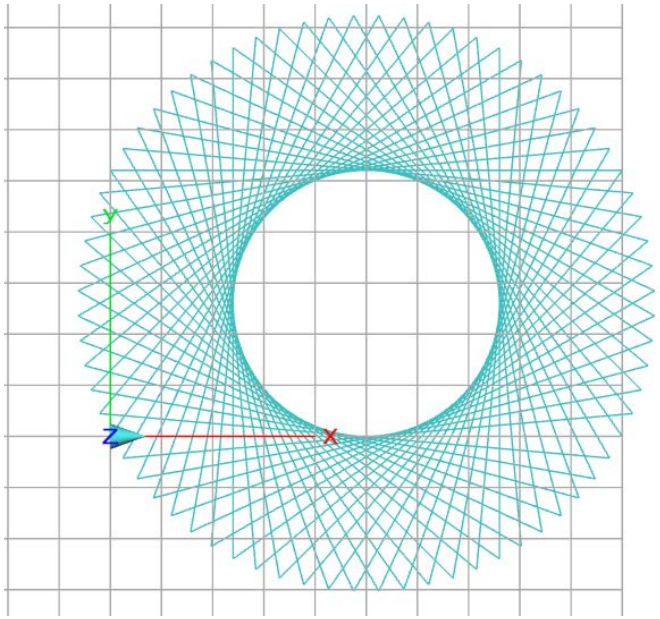
Rotated Triangles

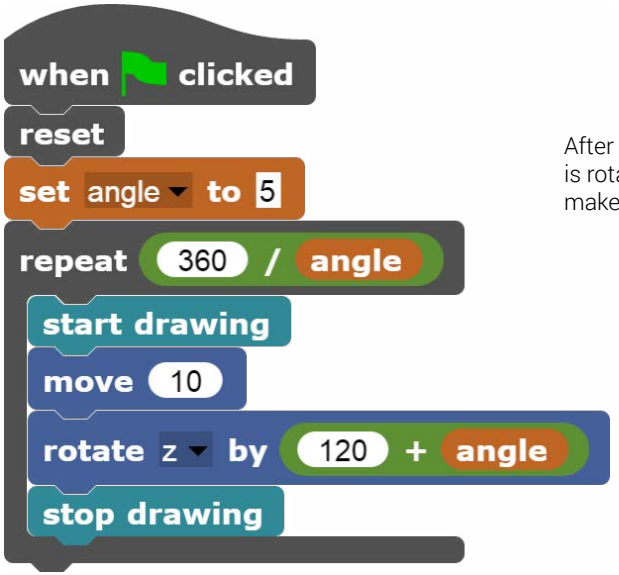




After drawing the first edge, the Beetle is rotated by 121 degrees (120 makes a triangle).

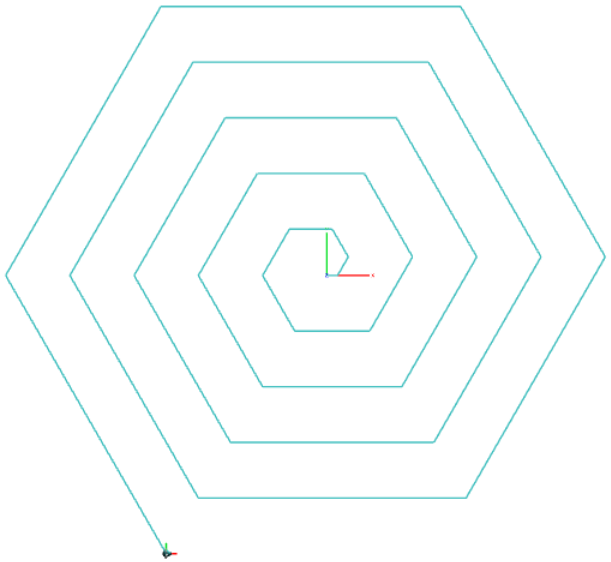
Creating a spiral by changing the size of every folloving edge by .1

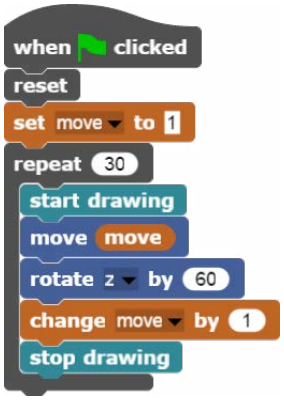


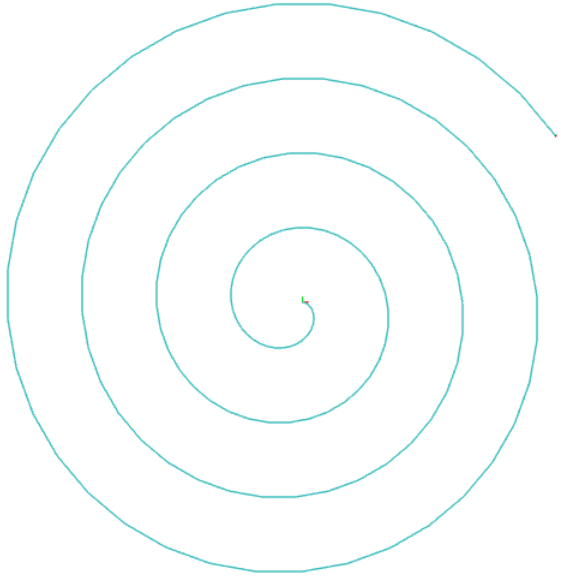


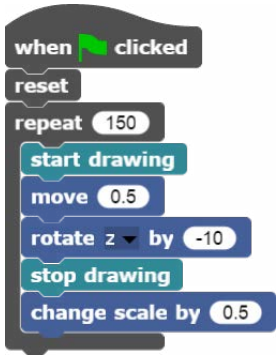
After drawing the first edge, the Beetle is rotated by 120 + angle degrees (120 makes a triangle).

Spirals

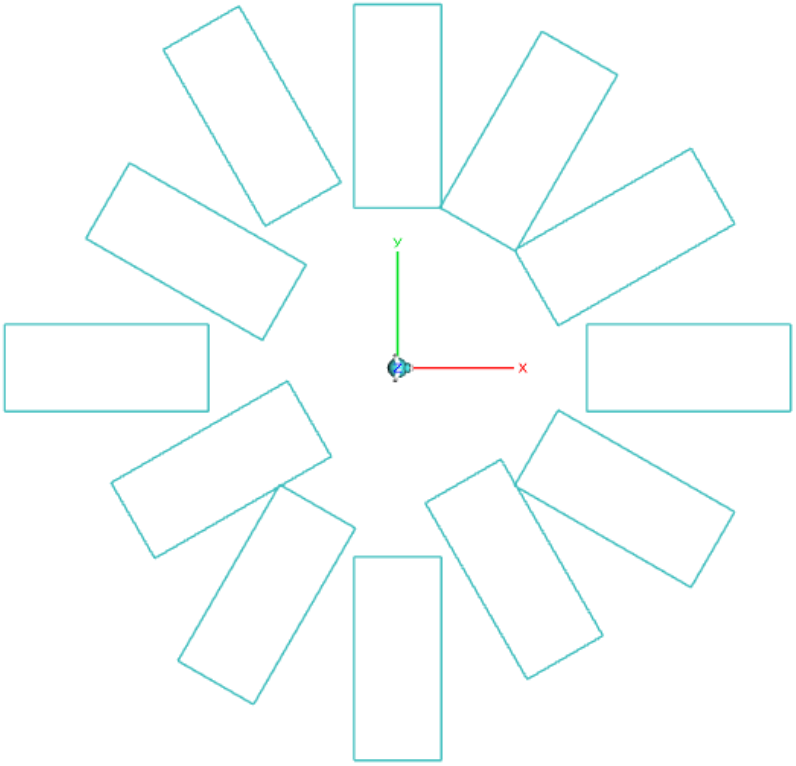


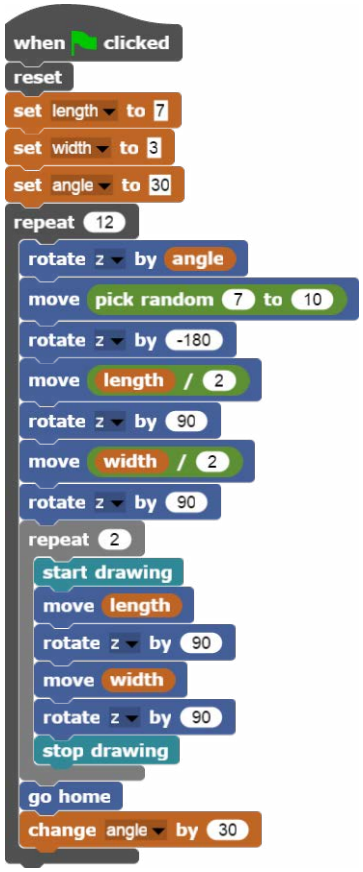

when clicked
reset
set move to 1
repeat 30
start drawing
move move
rotate z by 60
change move by 1
stop drawing



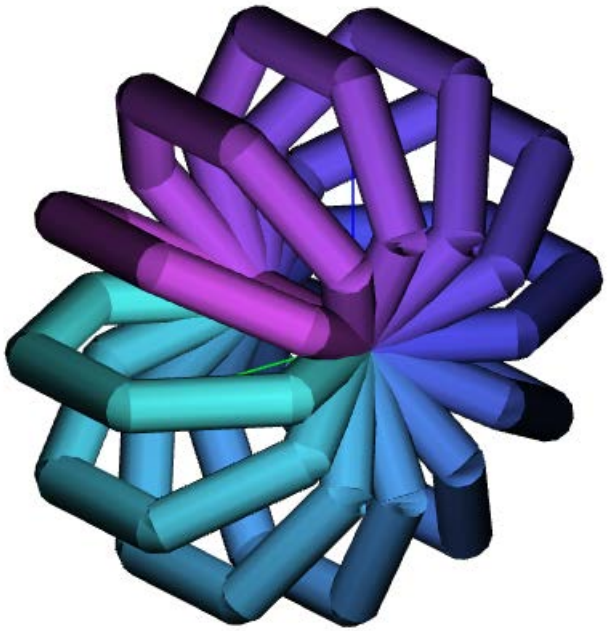

when clicked
reset
repeat 150
start drawing
move 0.5
rotate z by -10
stop drawing
change scale by 0.5

Rotational Pattern With Rectangles

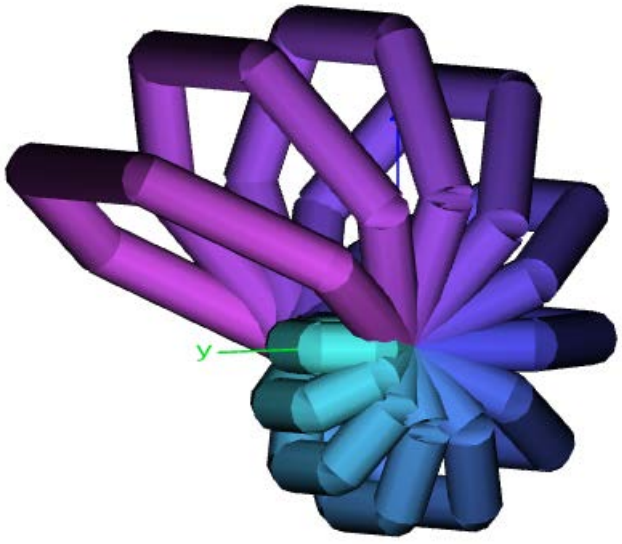




Rotated Loops

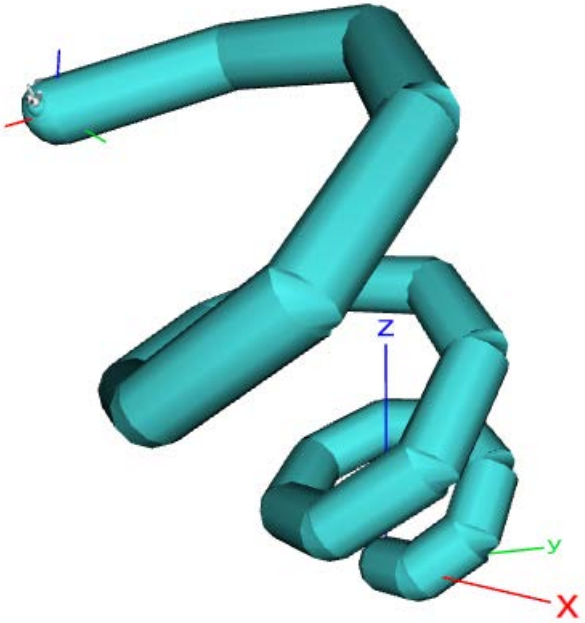


```
reset
repeat 12
  start extruding
  repeat 6
    move 3
    rotate z by 60
  stop extruding
  rotate x by 30
  change hue by 10
```

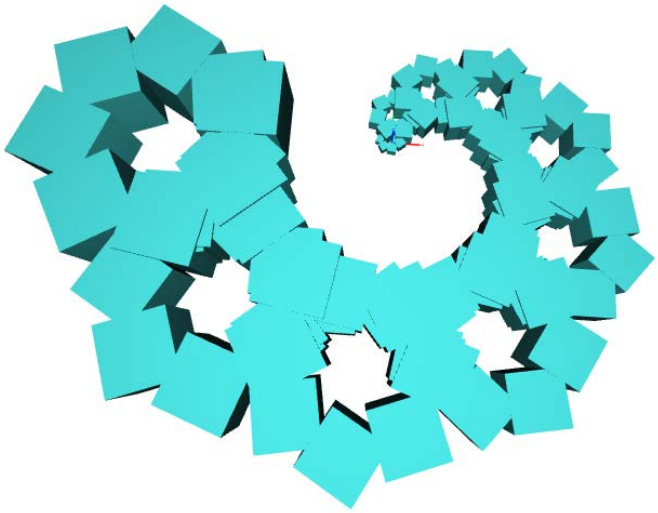



```
reset
set step to 1
repeat 12
  start extruding
  repeat 6
    move step
    rotate z by 60
  stop extruding
  rotate x by 30
  change step by 0.25
  change hue by 10
```

3D Spirals

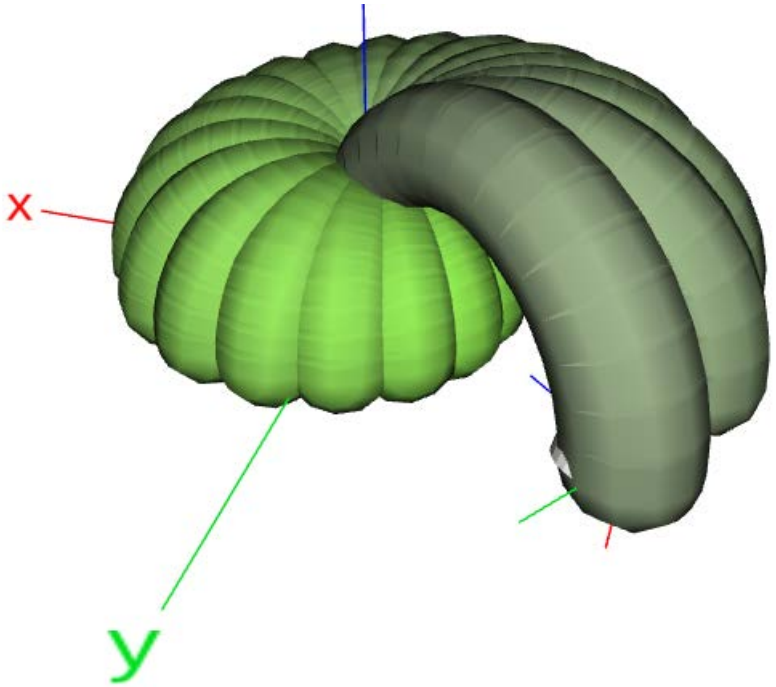



```
reset
set step to 1
start extruding
repeat 20
    move step
    rotate z by 36
    rotate y by 36
    change step by 0.1
stop extruding
```

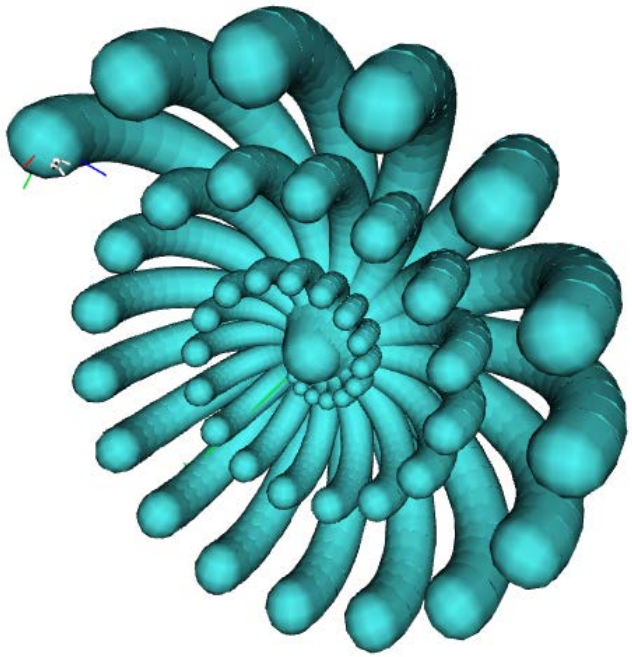


```
when clicked
reset
repeat 10
    repeat 10
        cube Dim. 0.5
        change scale by 0.2
        rotate z by 50
        move 0.5
    rotate z by 180
```

Shells

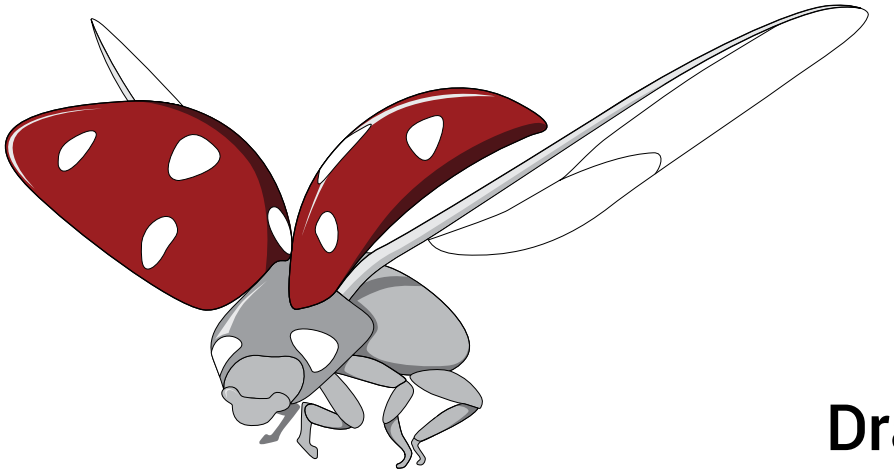



```
when clicked
script variables step
reset
set hue to 95
set step to .1
set y rotation to -90
repeat 20
  change step by 0.01
  rotate y by 180
  rotate x by 20
  go to x: 0 y: 0 z: 0
  start extruding
  repeat 18
    move step
    rotate y by 10
  stop extruding
  change saturation by -2
```



```
+Extruder+
rotate y by 180
rotate x by 20
go to x: 0 y: 0 z: 0
start extruding
repeat 18
  move step
  rotate x by -0.9
  rotate y by 10
stop extruding
```

```
when clicked
reset
set step to 0
set extrusion Dia. to 0.2
set y rotation to -90
repeat 50
  change step by 0.01
  Extruder
  change extrusion Dia. by 0.02
```



Drawing with boundaries

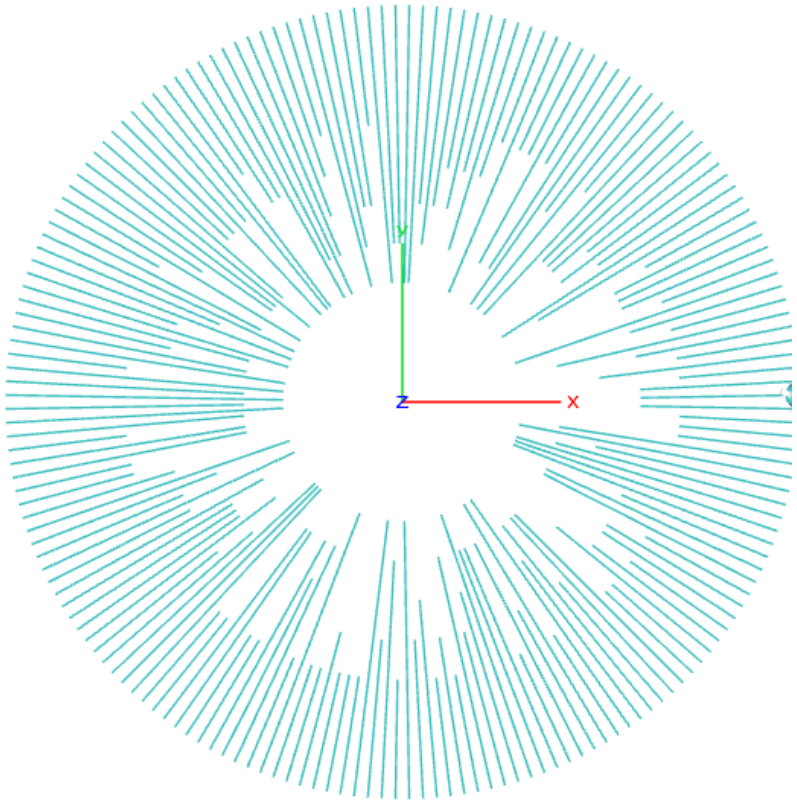
2D

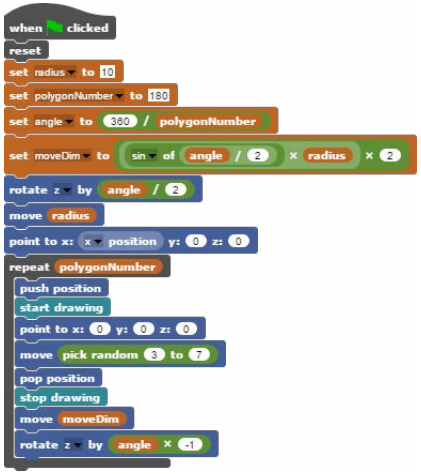
Circle boundary
Boundary with sin function
Bouncing arcs within boundary

3D

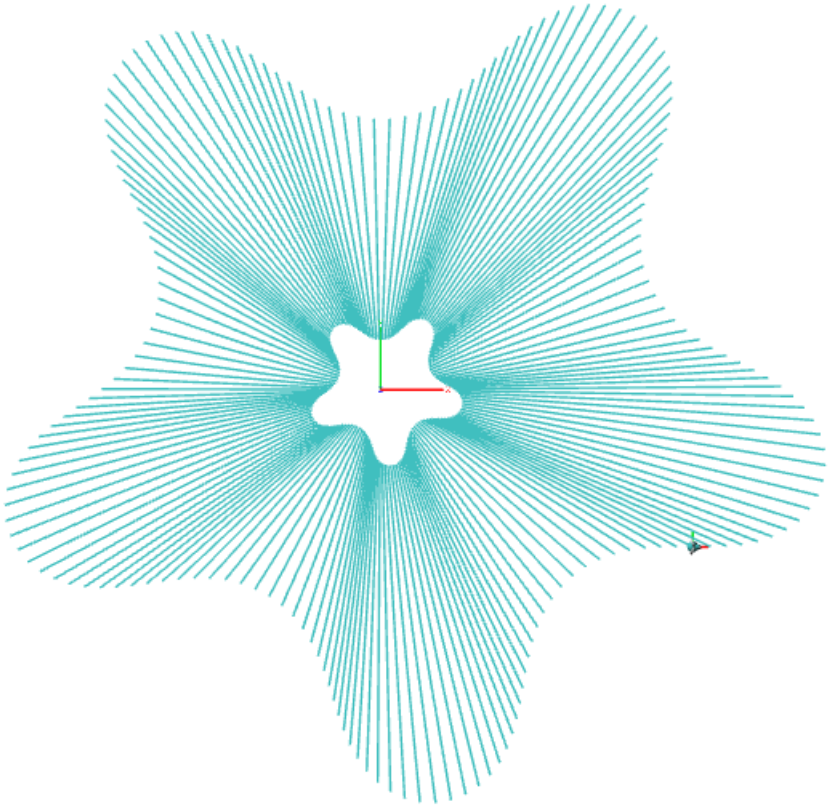
3d boundary

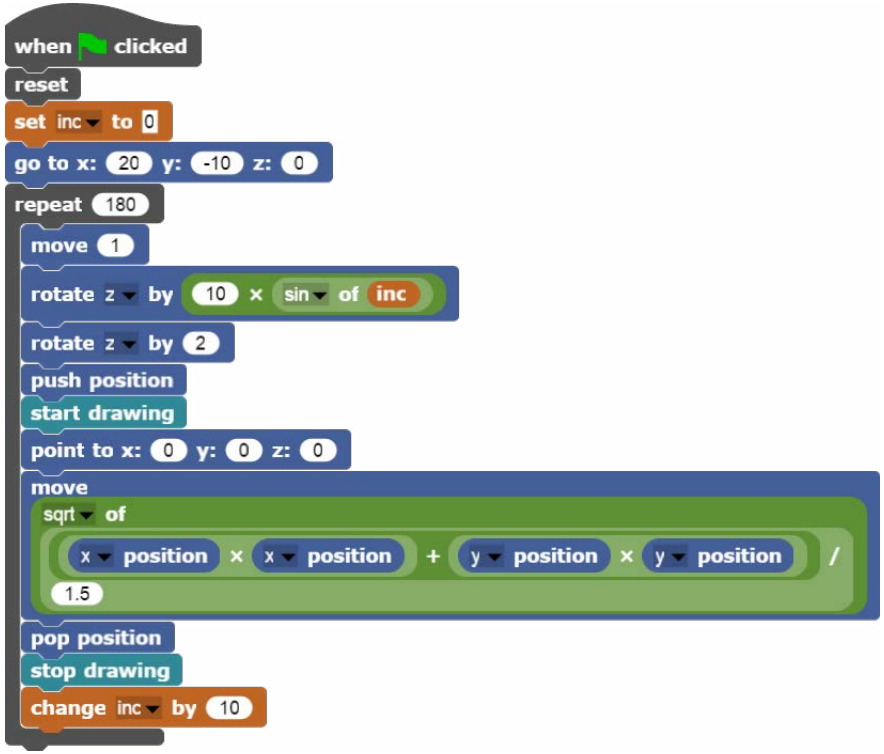
Circle Boundary



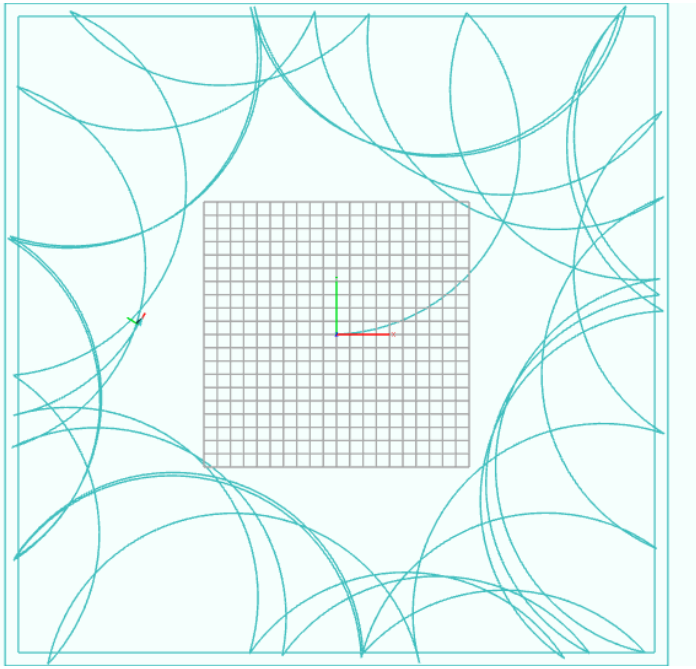


Boundary With Sin Function





Bouncing Arcs Within Boundary



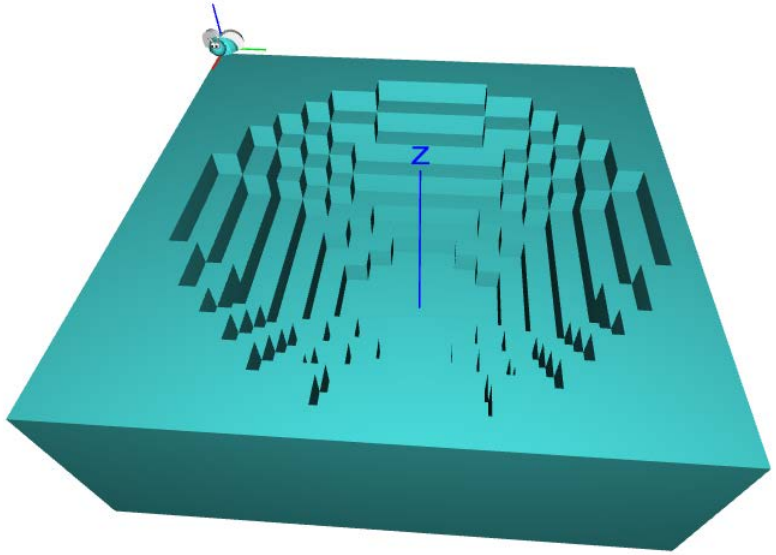
```
+squareBoundary+
rotate z by 45
move sqrt of 1250
rotate z by 45
repeat 4
    rotate z by 90
    start drawing
    move 50
    stop drawing
rotate z by 135
move sqrt of 2
rotate z by 45
repeat 4
    start drawing
    move 48
    stop drawing
    rotate z by -90
```

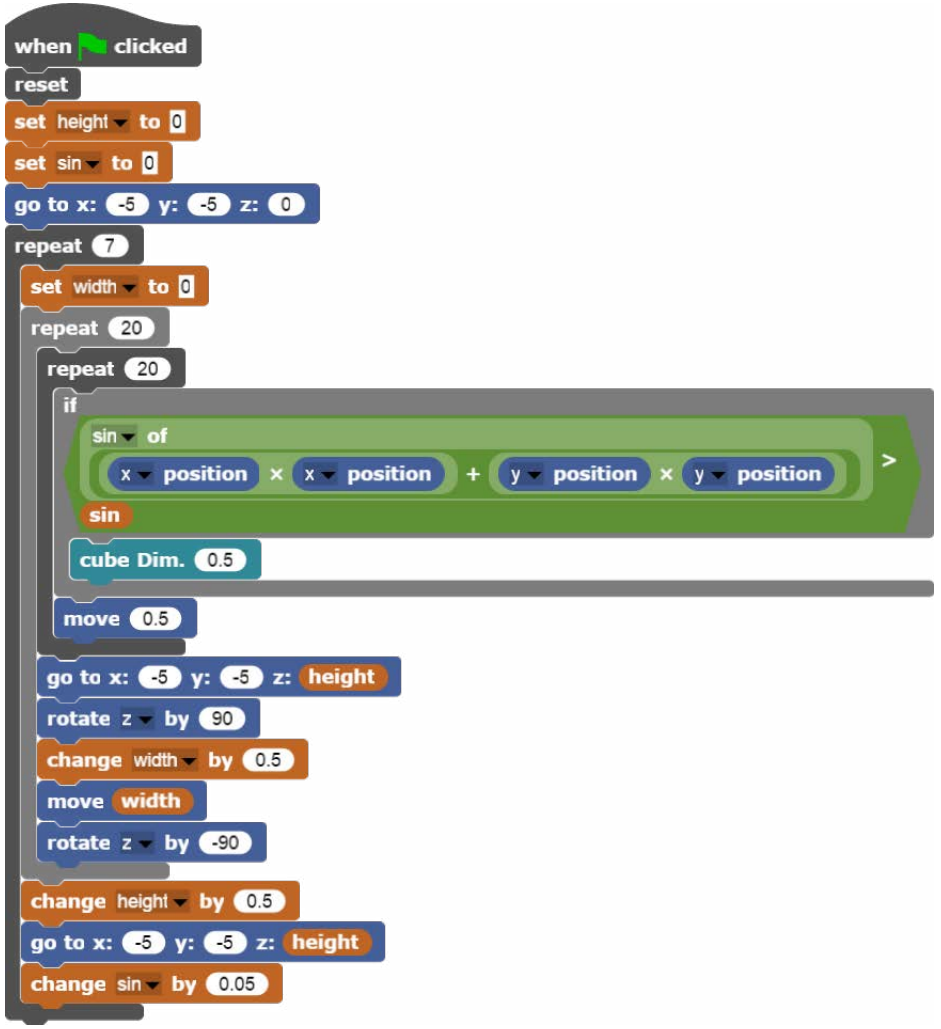
First, the block squareBoundary was created by placing the Beetle in the corner of the square with the edge = 50, and drawing the outside square of the boundary in a clockwise ditection. Then drawing the inside square with the edge = 48.

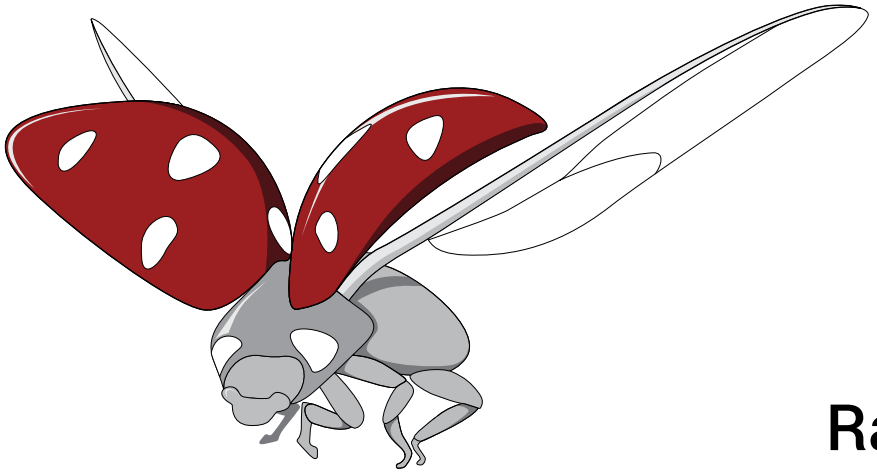
```
when clicked
reset
squareBoundary
go home
repeat 750
    rotate z by 4
    start drawing
    move 1
    if y position < -24 or y position > 24 or
       x position < -24 or x position > 24
        point to x: 0 y: 0 z: 0
        move 1
    stop drawing
```

Start drawing a curve till it hits the boundary. If the line hits or goes outside the boundary, the Beetle points to (0,0,0) and continues moving in different direction.

3D Boundary


Z


when clicked
reset
set height to 0
set sin to 0
go to x: -5 y: -5 z: 0
repeat 7
set width to 0
repeat 20
repeat 20
if
sin of
x position × x position + y position × y position
>
sin
cube Dim. 0.5
move 0.5
go to x: -5 y: -5 z: height
rotate z by 90
change width by 0.5
move width
rotate z by -90
change height by 0.5
go to x: -5 y: -5 z: height
change sin by 0.05



Randomness

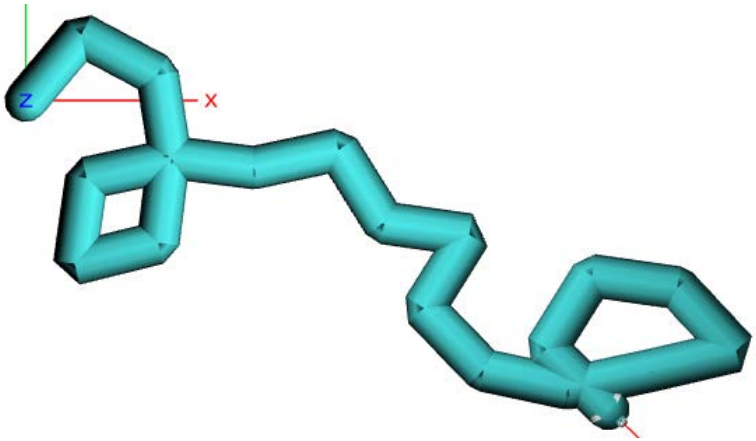
2D

Random walk
Random rotation
Random walk within boundary
Rotational pattern with random walk within boundary

3D

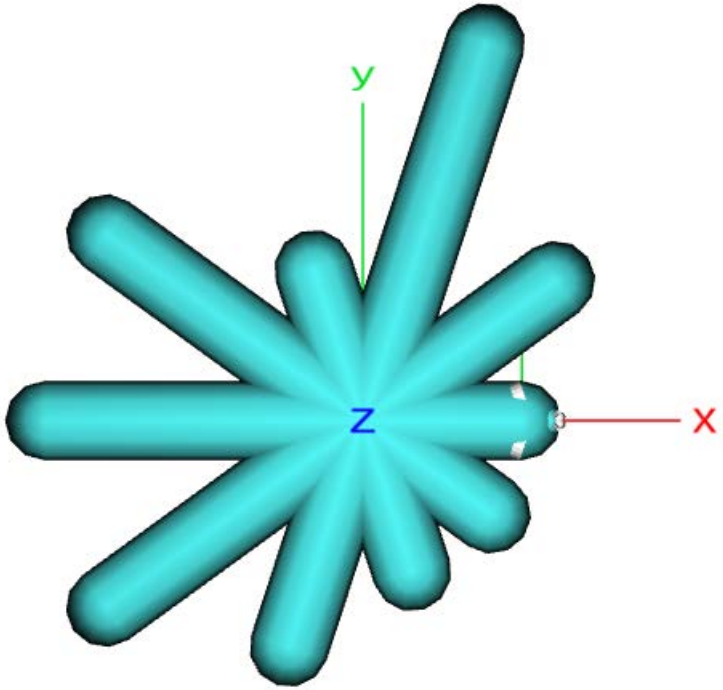
Random pattern
Spheres
Rotational pattern with random cubes

Random Walk



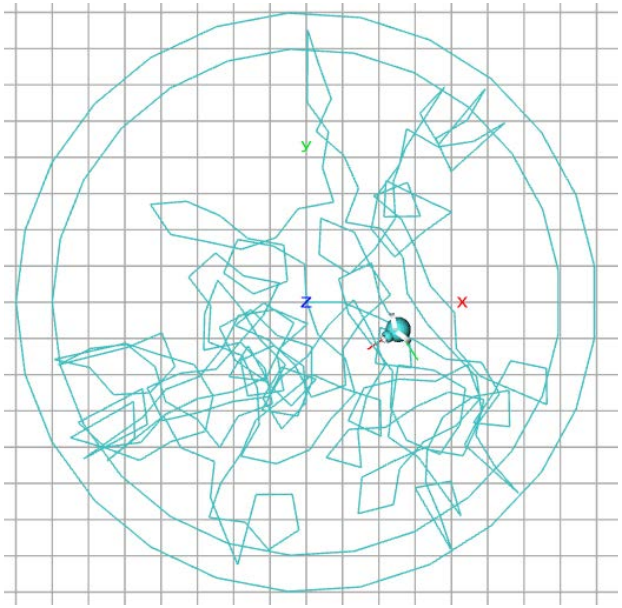

```
reset
start extruding
repeat 20
    rotate z by pick random -120 to 120
    move 2
stop extruding
```

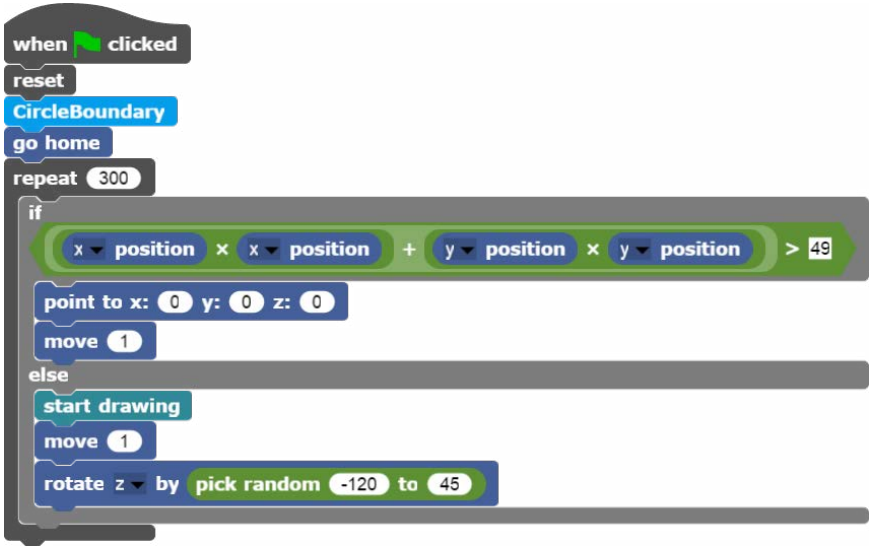
Random Rotation

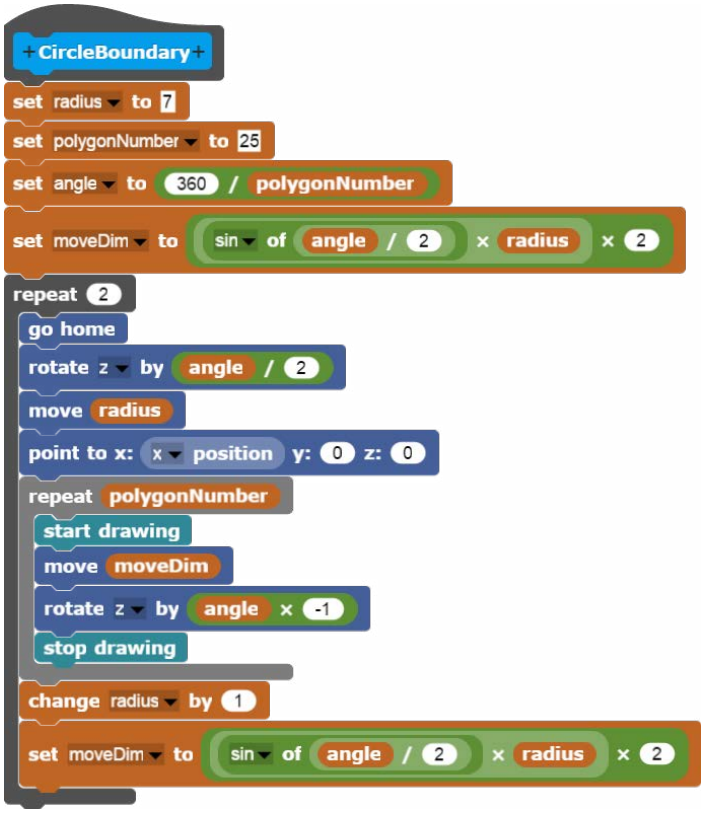



```
reset
repeat 10
    go to x: 0 y: 0 z: 0
    start extruding
    rotate z by 36
    move pick random 1 to 5
    stop extruding
```

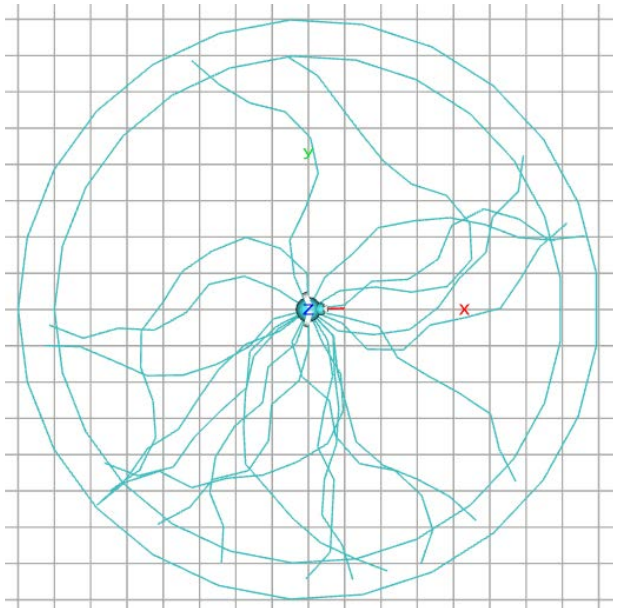
Random Walk Within Boundary

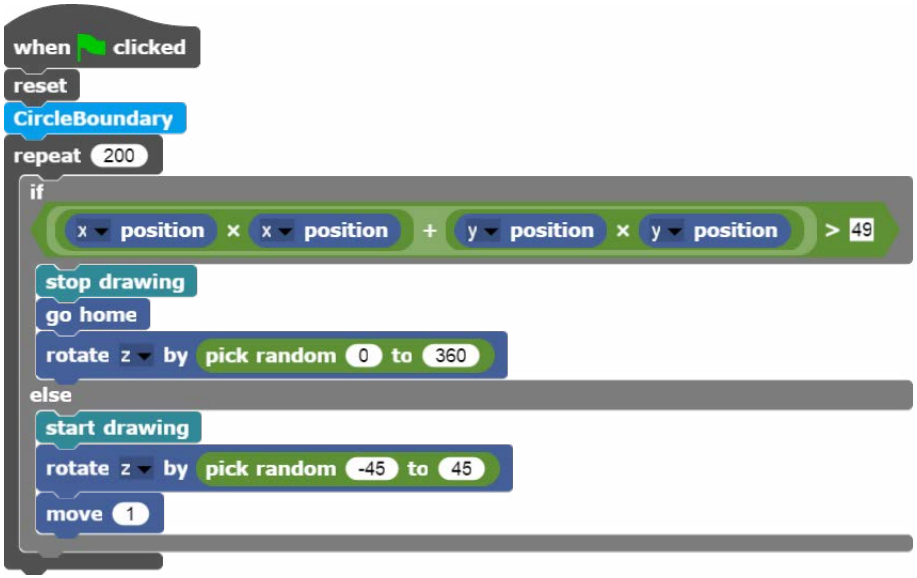




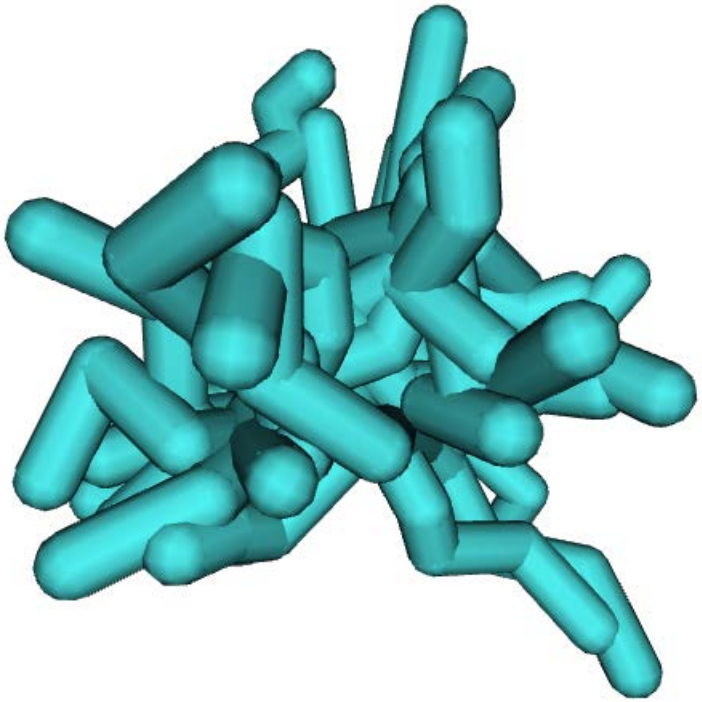


Rotational Pattern With Random Lines Within Boundary



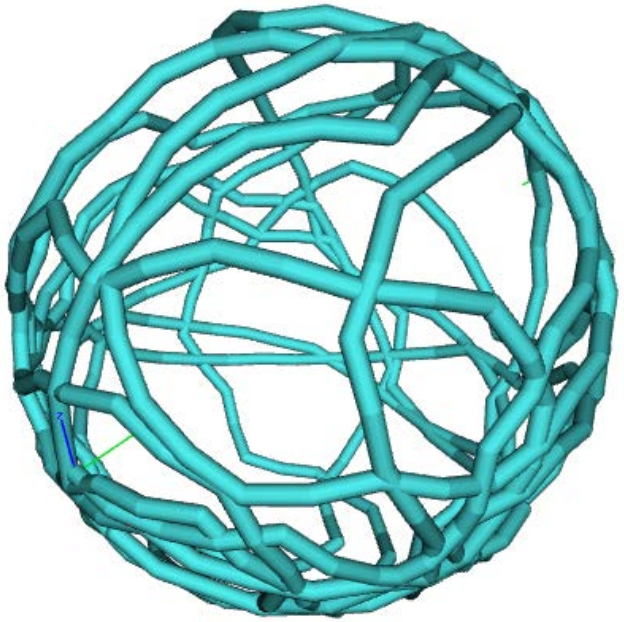


Random Pattern

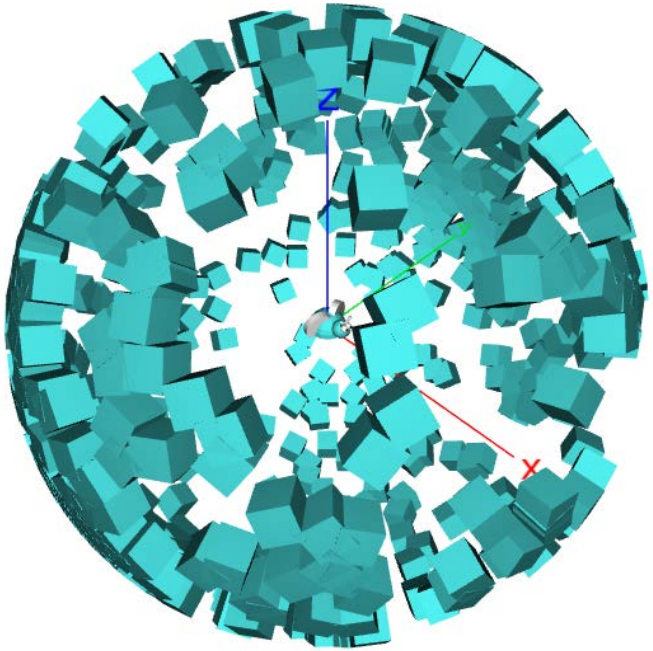


```
when clicked
reset
repeat 300
  push position
  if (x position < -5 or x position > 5 or y position < -5 or y position > 5 or z position < -5 or z position > 5)
    pop position
    point to x: 0 y: 0 z: 0
    move 1
  else
    start extruding
    move pick random 0 to 2
    rotate z by pick random -60 to 60
    rotate y by pick random -60 to 60
    rotate x by pick random -60 to 60
    stop extruding
```

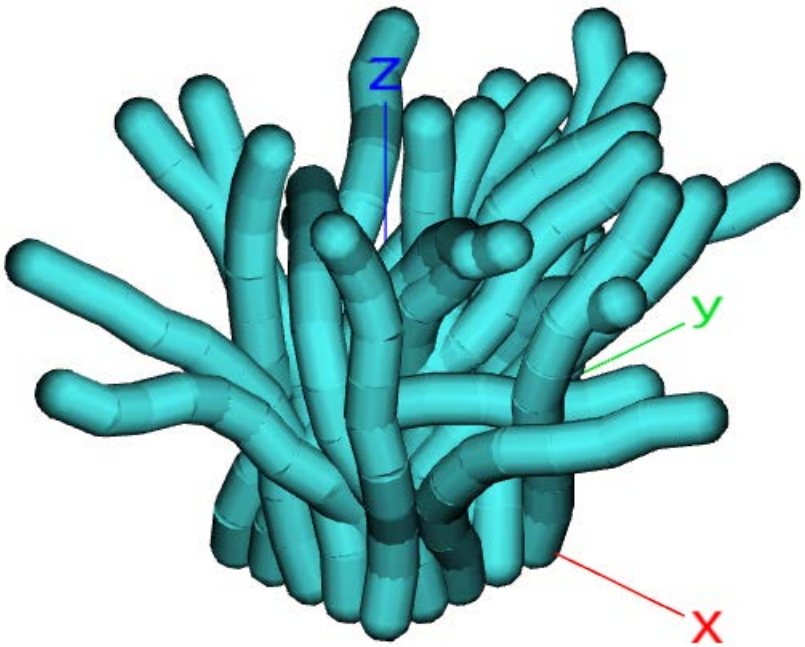
Spheres



```
when clicked
reset
repeat 200
  start extruding
  move 3
  stop extruding
  rotate z by 20
  start extruding
  move 3
  stop extruding
  rotate y by pick random -60 to 60
```

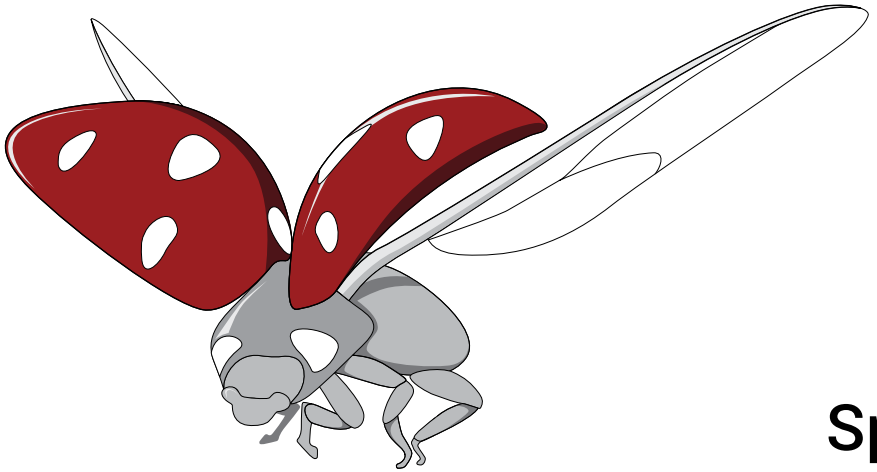



```
when clicked
reset
repeat 500
  rotate y by pick random 0 to 360
  rotate z by pick random 0 to 360
  rotate x by pick random 0 to 360
  move 5
  cube Dim. 0.5
  go home
```

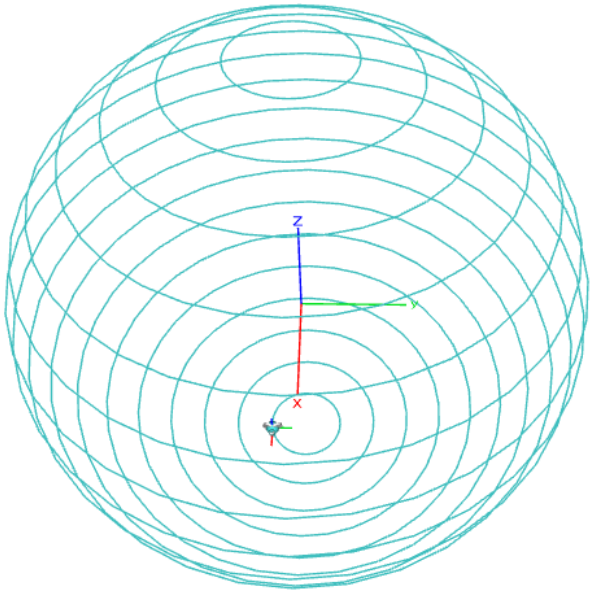


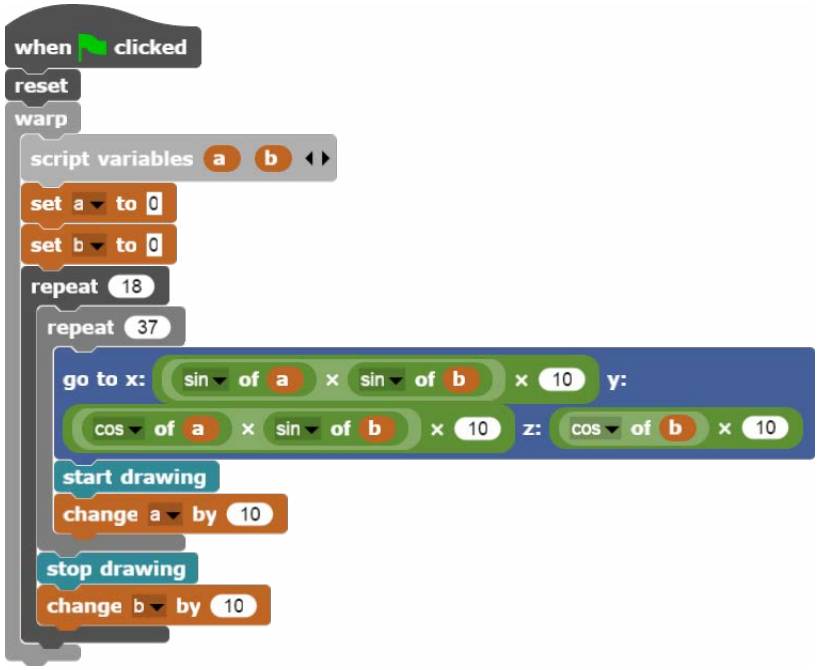

```
reset
script variables range
set range to 20
set extrusion Dia. to 0.5
grid 5 by 5 with step size 0.5
  rotate y by 90
  start extruding
  repeat 10
    rotate y by pick random -1 x range to range
    rotate z by pick random -1 x range to range
    move 0.5
  stop extruding
```

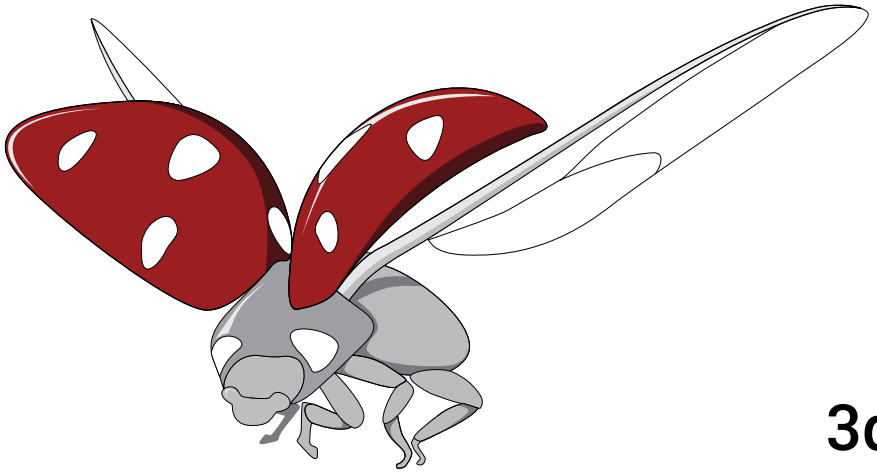
```
+grid+ x steps = 3 +by+ y steps = 4 +with+step+size+ step = 1 +
blocks λ +
push position
repeat y steps
  repeat x steps
    push position
    run blocks
    pop position
    move step
  move step x x steps x -1
  rotate z by -90
  move step
  rotate z by 90
pop position
```



Spheres calculus with sin and cos

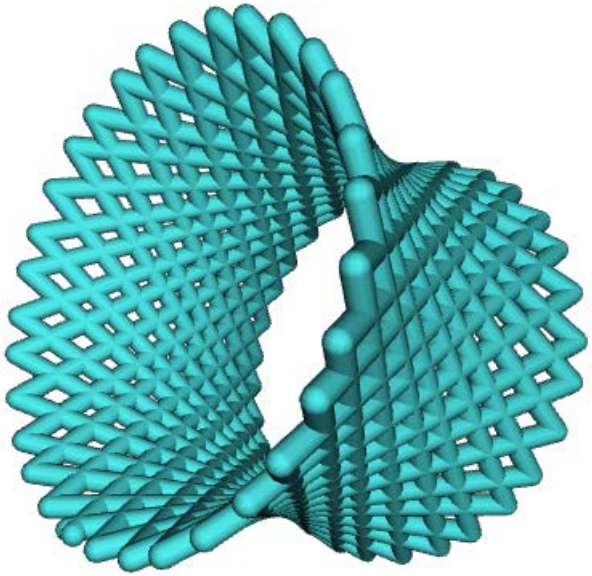

Z
Y
X

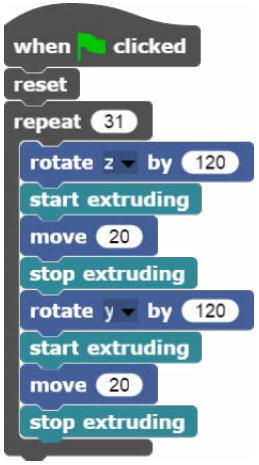

when clicked
reset
warp
script variables a b
set a to 0
set b to 0
repeat 18
repeat 37
go to x: sin of a x sin of b x 10 y:
cos of a x sin of b x 10 z: cos of b x 10
start drawing
change a by 10
stop drawing
change b by 10



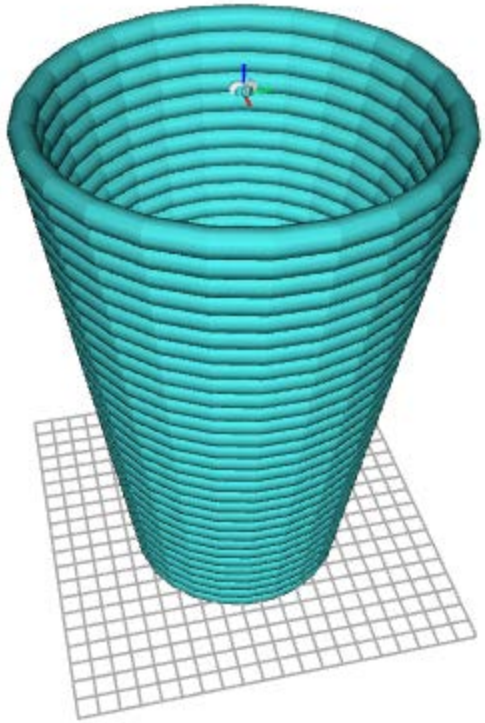
3d objects

Cup
Vase
Staircase
Chair
Disco Ball
Beetle



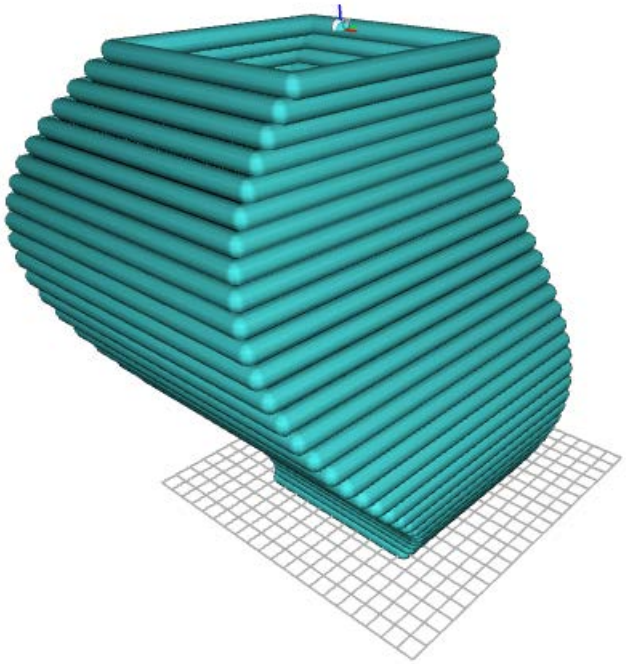

when clicked
reset
repeat 31
rotate z by 120
start extruding
move 20
stop extruding
rotate y by 120
start extruding
move 20
stop extruding

Cup



```
when clicked
reset
set radius to 5
set polygonNumber to 25
set angle to 360 / polygonNumber
set moveDim to sin of angle / 2 × radius × 2
script variables a
set a to 0
repeat 30
    push position
    rotate z by angle / 2
    move radius
    point to x: x position y: 0 z: a
    repeat polygonNumber
        start extruding
        move moveDim
        stop extruding
        rotate z by angle × -1
    pop position
    change a by 0.8
    set z to a
    change radius by 0.1
    set moveDim to sin of angle / 2 × radius × 2
```

Vase



```
when clicked
reset
set radius to 7
set polygonNumber to 4
set angle to 360 / polygonNumber
set moveDim to sin of angle / 2 × radius × 2
script variables a b
set a to 0
set b to 0
repeat 30
    push position
    rotate z by angle / 2
    move radius
    point to x: x position y: 0 z: a
    repeat polygonNumber
        start extruding
        move moveDim
        stop extruding
        rotate z by angle × -1
    pop position
    change a by 0.8
    change b by 10
    change moveDim by sin of b / 1.5
    go to x: sin of b × 3 y: 0 z: a
```

Staircase

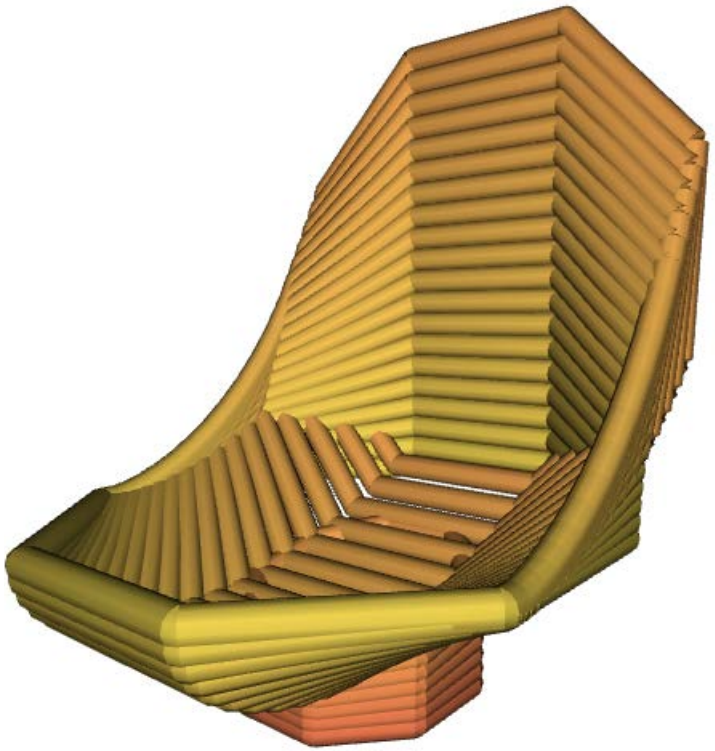
```
+RailSupport+
set height to 1
go to x: item height of RailX y: item height of RailY z:
item height of RailZ
repeat 23
  go to x: item height of RailX y: item height of RailY z:
  item height of RailZ – 3.25
  cuboid l: 0.25 w: 0.25 h: 6.5
  change height by 1
```

```
+RailExtrusion+
set height to 1
go to x: item height of RailX y: item height of RailY z:
item height of RailZ
repeat 23
  start extruding
  go to x: item height of RailX y: item height of RailY z:
  item height of RailZ
  stop extruding
  change height by 1
```

```
+RegisterPos+
go to x: 0 y: 0 z: height + 7
move length
insert x position at height of RailX
insert y position at height of RailY
insert z position at height of RailZ
```

```
when clicked
reset
set width to 5
set length to 15
set height to 1
set RailX to list
add height to RailX
set RailY to list
add height to RailY
set RailZ to list
add height to RailZ
repeat 24
  go to x: 0 y: 0 z: height – 0.5
  move width / 2
  rotate z by -90
  move length / 2
  cuboid l: length w: width h: 0.5
  RegisterPos
  change height by 1
  rotate z by 75
go to x: 0 y: 0 z: 12
cuboid l: 1 w: 1 h: 24
RailExtrusion
RailSupport
```

Chair



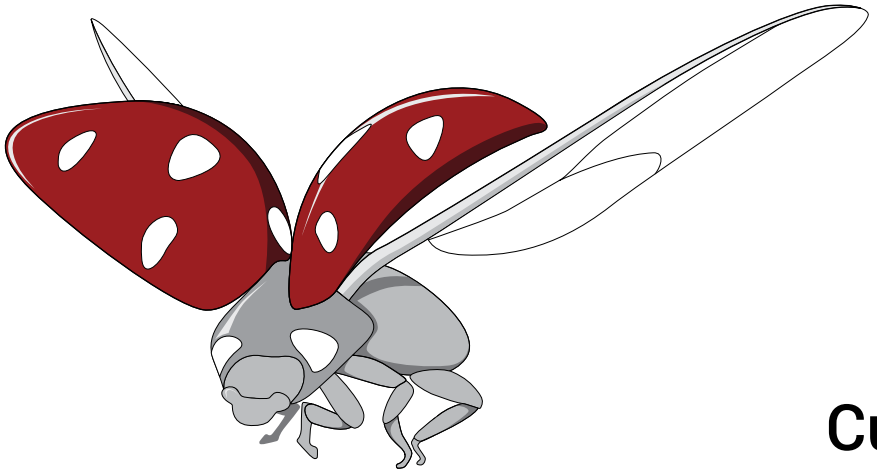
```
when clicked
reset
shell
leg
```

```
+shell+
set hue to 50
repeat 20
  warp
    move -3.5
    start extruding
    repeat 8
      move 7
      rotate z by 45
    stop extruding
    change hue by -1
    rotate x by 4.5
    set x to 0
    change absolute z by 0.9
```

```
+leg+
go home
set saturation to 50
set x rotation to 14
set z to -1.5
repeat 8
  warp
    set x to -1.5
    set y to 4
    start extruding
    repeat 8
      move 3
      rotate z by 45
    stop extruding
    change hue by -2
    rotate x by -2
    change absolute z by -0.6
```

Disco Ball

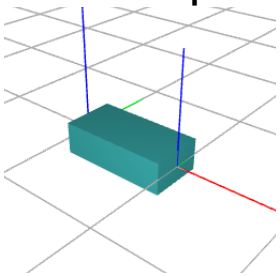
```
when clicked
reset
go home
go to x: 0 y: 0 z: 5.7
set opacity to 100
set hue to 80
set lightness to 50
set saturation to 100
sphere Dia. 11.5
set hue to 50
set lightness to 80
set saturation to 70
go to x: 0 y: 0 z: 0
repeat 36
    push position
    repeat 18
        move 1
        rotate y by 10
        set opacity to pick random 10 to 100
        cuboid l: 1 w: 1 h: 0.1
    pop position
    rotate z by 10
```



Custom blocks

Cuboid step
Tube step
Spiral
Go to 3d point
Point to 3d point
Draw with extrusion
Loop that makes a ring
Ring of loops
Grid

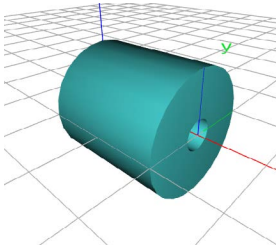
Cuboid Step



```
CuboidStep l: 1 w: 0.5 h: 0.3
```

```
+CuboidStep+l:+ length # = 1 +w:+ width # = 0.5 +h:+ height # = 0.3
+
move length / 2
cuboid l: length w: width h: height
move length / 2
```

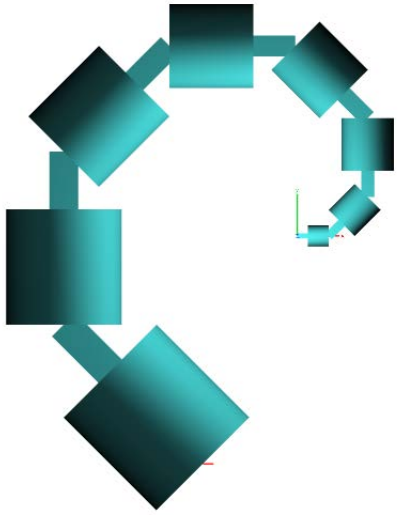
Tube Step



```
TubeStep l: 2 outer: 2 inner: 0.5
```

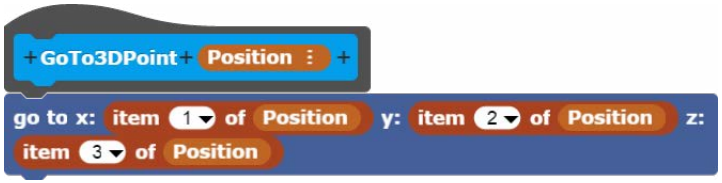
```
+TubeStep+l:+ length # = 2 +outer:+ outer # = 2 +inner:+
inner # = 0.5 +
move length / 2
tube l: length outer: outer inner: inner
move length / 2
```

Spiral

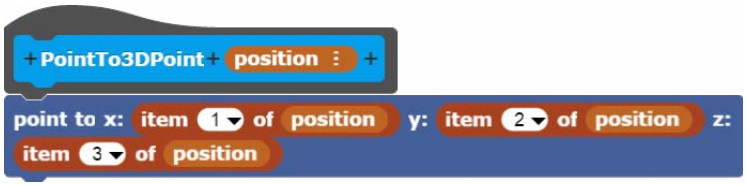


```
when clicked
reset
repeat 8
  CuboidStep l: 1 w: 0.5 h: 0.3
  TubeStep l: 2 outer: 2 inner: 0.5
  change scale by 0.75
  rotate z by 45
```

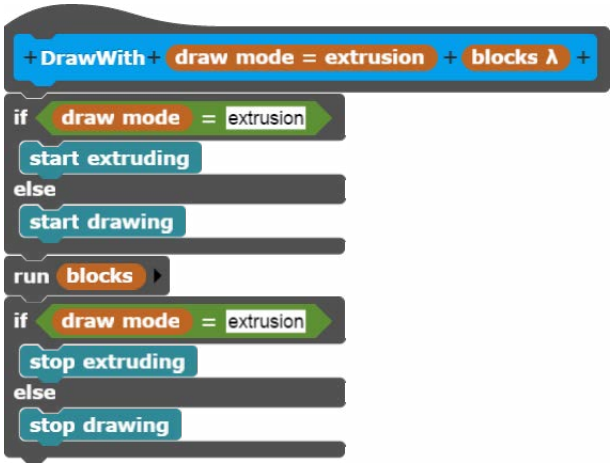
Go To 3D Point



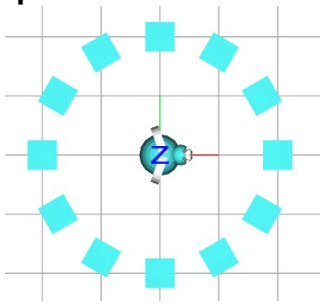
Point To 3D Point



Draw With Extrusion



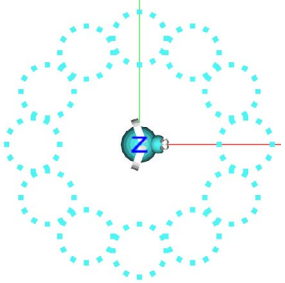
Loop That Makes a Ring



```
reset
circle points steps: 12 diam: 4
  cube Dim. 0.5
```

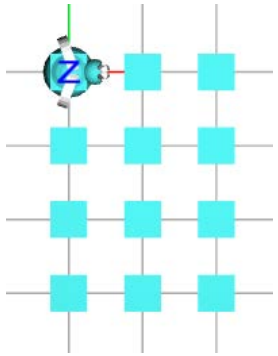
```
+circle+points+steps:+ steps = 12 +diam:+ diam = 4 + blocks λ +
push position
repeat steps
  push position
  move diam / 2
  run blocks
  pop position
  rotate z by 360 / steps
pop position
```

Ring of Rings



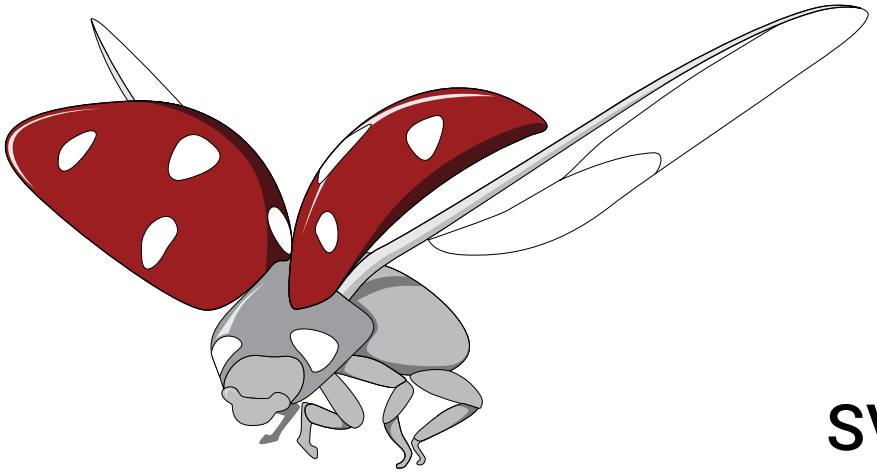
```
reset
circle points steps: 12 diam: 4
  circle points steps: 12 diam: 1
    cube Dim. 0.1
```

Grid

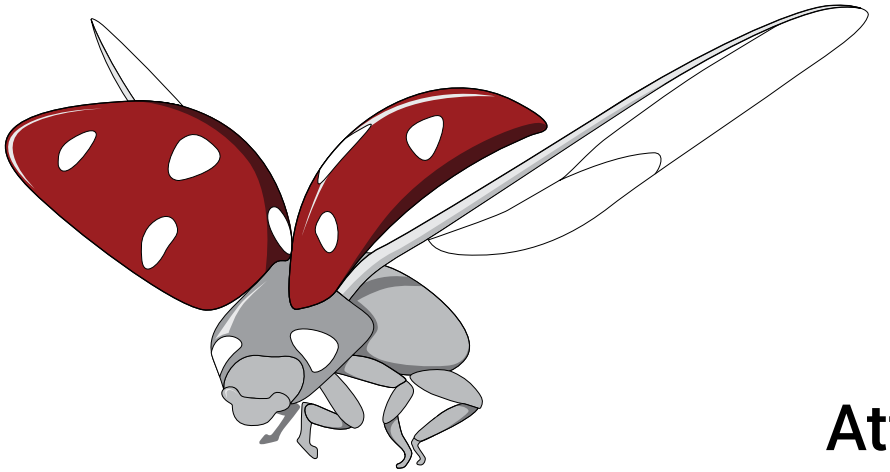


```
reset
grid 3 by 4 with step size 1
  cube Dim. 0.5
```

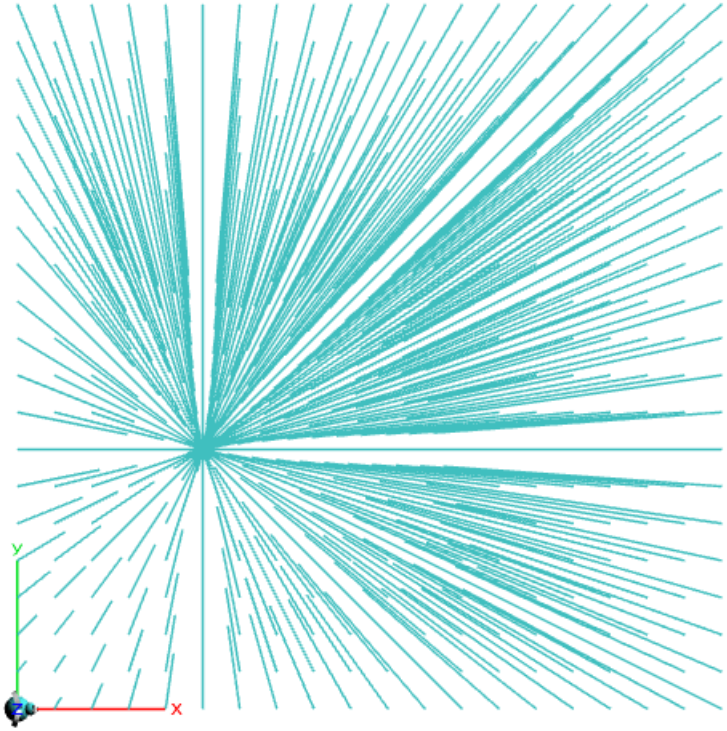
```
+grid+ x steps = 3 +by+ y steps = 4 +with+step+size+ step = 1 +
blocks λ +
push position
repeat y steps
  repeat x steps
    push position
    run blocks
    pop position
    move step
  move step × x steps × -1
  rotate z by -90
  move step
  rotate z by 90
pop position
```

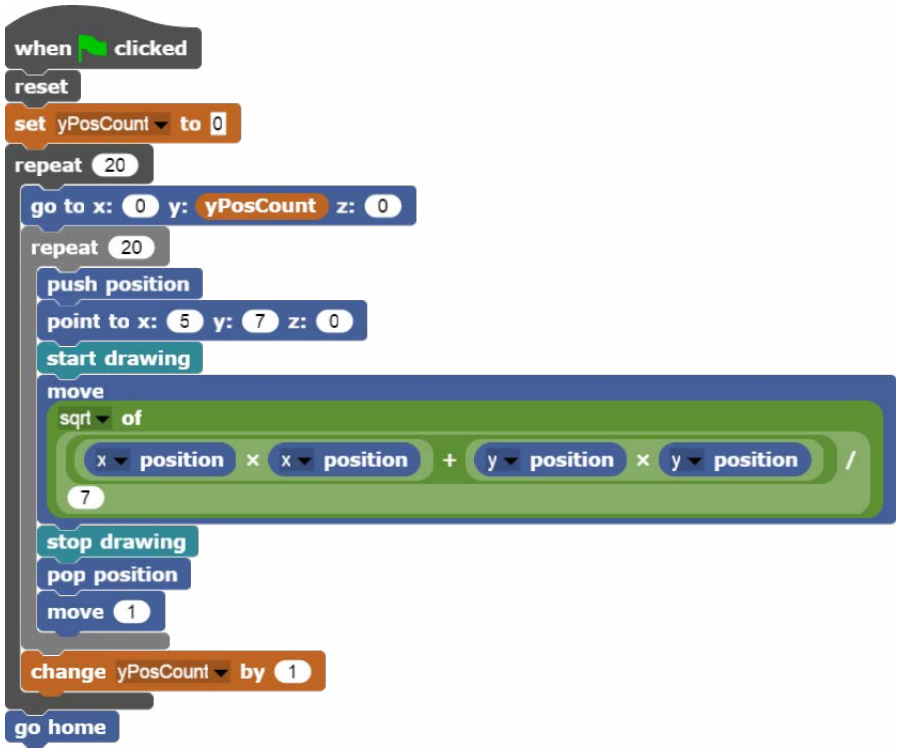


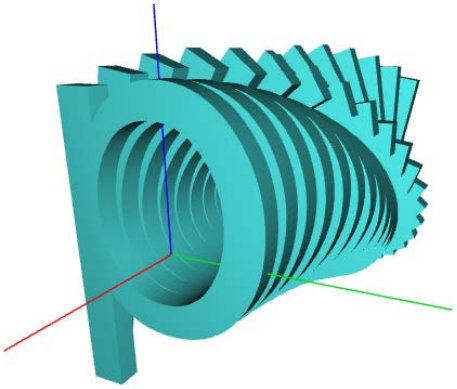
SVG file writer



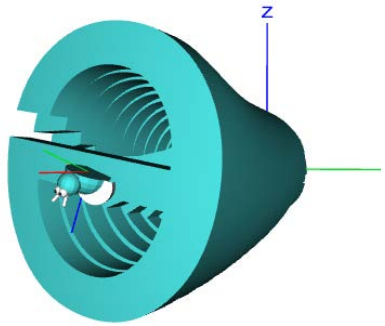
Attractors


y
x

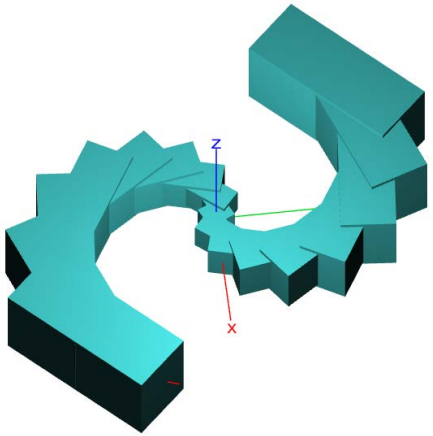

when clicked
reset
set yPosCount to 0
repeat 20
go to x: 0 y: yPosCount z: 0
repeat 20
push position
point to x: 5 y: 7 z: 0
start drawing
move
sqrt of
x position x x position + y position x y position /
7
stop drawing
pop position
move 1
change yPosCount by 1
go home



```
when clicked
reset
repeat 20
  rotate z by 90
  text b H: 3 W: 0.25
  rotate z by -90
  rotate x by 10
  move 0.25
```




```
when clicked
reset
repeat 20
  rotate z by 90
  text e H: 3 W: 0.25
  rotate z by -90
  rotate x by 10
  move 0.25
  change scale by 0.05
```




```
when clicked
reset
repeat 10
  cuboid l: 1 w: 0.4 h: 0.5
  move 0.4
  rotate z by 25
  change scale by 0.5
go home
set scale to 1
rotate z by 180
repeat 10
  cuboid l: 1 w: 0.4 h: 0.5
  move 0.4
  rotate z by 25
  change scale by 0.5
```