Chapter 1

Dr. Geo Smalltalk sketch

The Dr. Geo *Smalltalk sketch* – (DSS) – are sketch entirely defined in Smalltalk language. So it is not about constructing a sketch with the Dr. Geo graphical interface but about describing a sketch with the Smalltalk language. We provide a nice programming interface for an easy and light syntax.

1.1 Smalltalk sketch by the example

Smalltalk itself is a high level language, carefully crafted iteratively for about 10 years at Xerox Parc Research Labs. When a sketch is defined with it, we can use all the power of the language to build recursively a sketch, or to position randomly some objects to get slightly different sketch at each execution of its Smalltalk code. Therefore, a Smalltalk sketch is freed from the constraints of the graphic user interface while reinforced by the Smalltalk language.

A Smalltalk sketch is a source code to execute in a **workspace**. It is a text editor where source code can be written and executed. To open it, use the shortcut Ctrl-k when no window is selected – or click on the background and in the menu select `Tools>Workspace` . It is possible to paste text into with the shortcut Ctrl-v. Read section **??**, p. **??** to learn more about this tool.

We will study several examples, each one will be written in a workspace and its source code will be selected with mouse, then executed with the shortcut Ctrl-d to *Do-it!*[1].

Let's start with a simple Smalltalk sketch:

```
DrGeoCanvas new
```

It is the smallest we can define. When executing it, it simply creates a new empty sketch. The Dr. Geo canvas displayed is simplified as there is no toolbar, only the menu bar.

A second example:

```
| c item |
c := DrGeoCanvas new.
item := c point: 1.2 @ -2.
item name: 'A'.
```

Here we define a sketch containing a free point A of coordinates $(1,2\ ;\ -2)$. An object is added to the construction by sending a message to the canvas, here `#point:` to create a free point given its coordinates. The result is a point object, we can also modify by sending message to it, here we rename it 'A'.

Let's continue with a third example:

[1] Alternatively, this is the entry to select in the contextual menu of the workspace.

```
| c triangle hasard m n p |

triangle := [:p1 :p2 :p3 |
c segment: p1 to: p2.
c segment: p2 to: p3.
c segment: p3 to: p1].

ourRandom := [5 - 10 atRandom].

c := DrGeoCanvas new.
m := c point: ourRandom value @ 0.
n := c point: 5 @ 0.
p := c point:  ourRandom value @ 3.
triangle value: m value: n value: p.
```

This example is interesting, it shows us three things:

1. The introduction of more elaborated construction, not initially implemented in Dr. Geo. Here we define a bloc of code `triangle` – between the square bracket – given three points, construct a triangle. We can compare this to macro-construction but with a different approach, programing oriented.
2. The definition of a bloc of code – `ourRandom` – to give us integer random number between -5 and 5. It is used for a random positioning of the points. Therefore each time the sketch is executed, it is slightly different.
3. Affecting the result of a construction – the result we got when sending a construction message to the canvas – is not mandatory, we do it when we need to keep a reference of the constructed object for later use. Here in the bloc of code `triangle`, we don't keep reference of the constructed segments. However concerning the defined points, we keep reference in temporary variables `m`, `n` and `p`, for later use as arguments when executing the bloc of code `triangle`.

To finish with this introduction by the examples, here is a last one:

```
| c a b d |

c := DrGeoCanvas new.
a := c point: 1@0.
b := c point: 5@0.
d := c line: a to: b.
a color: Color yellow;
   round;
   large.
b hide.
d dashed.
```

Two points and a line are constructed. Then messages are sent to them to modify the style or to hide some.

We have finished our small guided tour to the Dr. Geo *Smalltalk sketch*. In the following sections, we expose the command available.

1.2 Reference methods for the Dr. Geo Smalltalk sketch

To add an object in a construction, you send a message to the canvas. The resulting constructed object can be modified as well by sending message to it.

So before adding any object in a canvas, we need to create one with the command `DrGeoCanvas new`.

1.2.1 Various messages

`<canvas> DrGeoCanvas new`
⟶ Return a canvas and open it. The result is normally set in a variable for later use to add construction.
Example :

```
| canvas |
canvas := DrGeoCanvas new.
```

`canvas do: bloc`
`bloc` : Smalltalk bloc of code with instructions for construction and/or animation of the sketch.
Action : Execute the bloc of code in a specific background process. Use it when a construction need to be done step by step in front of the user or when the sketch is animated.
Example :

```
| canvas point |
canvas := DrGeoCanvas new.
point := canvas point: 0@0.
canvas do: [
   -5 to: 5 by: 0.1 do: [:x |
      point moveTo: x@(x cos * 3).
      (Delay forMilliseconds: 100) wait.
      canvas update]
]
```

`canvas update`
Action : Update the canvas after some objects were modified. Used mainly for animation.

`canvas fullscreen`
Action : The sketch is set full screen.

`canvas gridOn`
Action : Display the grid in the canvas.

`canvas centerTo: aPoint`
`aPoint` : Coordinates of a point.
Action : The canvas is displaced so the point given in argument is at the centre of canvas window.
Example :

```
canvas centerTo: 5@0.
```

`canvas scale: anInteger`
`anInteger` : Scale of the canvas. One unite is about 1 pixel.
Action : modify the scale of the canvas.
Example :

```
canvas scale: 10.
```

Point.

`<point> canvas point: aPoint`

`aPoint` : coordinates (x,y).
⟶ free point in the plane with coordinates `aPoint`.
Example :

```
canvas point: 5@2.
```

`<point> canvas pointX: v1 Y: v2`

`v1` : value object
`v2` : value object
⟶ point constrained by its coordinates
Example :

```
canvas pointX: (canvas freeValue: 2) hide Y: (canvas freeValue: 5) hide.
```

`<point> canvas pointOnCurve: curve at: abscissa`

`curve` : line (straight line, ray, segment, etc.)
`abscissa` : curvilinear abscissa of the free point, abscissa in interval [0 ; 1]
⟶ free point on a line
Example :

```
myPoint := canvas pointOnCurve: s1 at: 0.5.
```

`<point> canvas middleOf: p1 and: p2`

`p1` : point item or coordinates
`p2` : point item or coordinates
⟶ middle of two points
Example :

```
| a i |
a := canvas point: 1@1.
i := canvas middleOf: a and: 4@4.
```

`<point> canvas middleOf: s`

`s` : segment
⟶ middle of a segment
Example :

```
canvas middleOf: s.
```

`<point> canvas intersectionOf: l1 and: l2`

`l1` : line
`l2` : line
⟶ point of intersection of two lines
Example :

```
canvas intersectionOf: droite and: segment
```

`<point> canvas altIntersectionOf: l1 and: l2`

`l1` : line
`l2` : line
⟶ alternative point of intersection of two lines, when it exists

Example :
```
canvas altIntersectionOf:  droite and:  circle.
```

`<point> canvas point:  block parent:  item`

`block` : block of code returning coordinates
`item` : math item
⟶ point whose coordinates are calculated with the block of code with "item" as argument
Example :

```
| figure s mobile c block |
figure := DrGeoCanvas new.
s:=figure
   segment: (figure point: -5@0)
   to: (figure point: 5@0).
mobile := figure pointOnCurve: s at: 0.1.
block := [:mathItem | |x|
   x := mathItem point x.
   x @ (x * x * x / 25 - x)].
c := figure point: block parent: mobile.
figure locusOf: c when: mobile.
```

`<point> canvas point:  block parents:  itemCollection`

`block` : block of code returning coordinates
`itemCollection` : math item collection
⟶ point whose coordinates are calculated with the block of code with "itemCollection" as argument
Example :

```
| figure a b d m p |
figure:=DrGeoCanvas new.
a:=figure point: (-2)@1.
b:=figure point: 3@3.
d:=figure line: a to: b.
d color: Color blue.
m:=figure point: 1@(-1).
p:= figure
   point: [:parents | parents first closestPointTo: parents second point]
   parents: {d . m}.
```

Line.

`<line> canvas line:  p1 to:  p2`

`p1` : point or coordinates
`p2` : point or coordinates
⟶ line passing through these two points
Example :

```
| p1 |
p1 := canvas point:  0@0.
canvas line:  p1 to:  1@2.
```

`<line> canvas parallel: d at: p`

`p` : point or coordinates
`d` : direction (line, segment, vector,...)
⟶ line parallel to direction `d` and passing through point `p`
Example :

```
| a |
a := canvas point: 1@5.
canvas parallel: d at: a.
```

`<line> canvas perpendicular: d at: p`

`p` : point or coordinates
`d` : référence d'une direction (droite, segment, vecteur, ...)
⟶ line perpendicular to direction `d` and passing through point `p`
Example :

```
canvas perpendicular: d at: 1@5.
```

`<line> canvas perpendicularBisector: s`

`s` : segment
⟶ perpendicular bisector to `s`
Example :

```
canvas perpendicularBisector: (canvas segment: 0@0 to: 4@4)
```

`<line> canvas perpendicularBisector: a to: b`

`a` : point or coordinates
`b` : point or coordinates
⟶ perpendicular bisector to `s`
Example :

```
canvas perpendicularBisector: 0@0 to: 4@4
```

`<line> canvas angleBisector: a`

`a` : geometric angle defined by **three points**
⟶ angle bissector of the angle `a`
Example :

```
canvas angleBisector: angle
```

`<line> canvas angleBisectorSummit: a side1: b side2: c`

`a,b,c` : points defining a geometric angle $\widehat{bac}$
⟶ angle bisector of the angle $\widehat{bac}$
Example :

```
canvas angleBisectorSummit: 0@0 side1: 1@0 side2: 0@1
```

Ray.

`<ray> canvas ray: o to: p`

`o` : point or coordinate, the origin
`p` : point or coordinates, point anywhere on the ray
⟶ ray defined by its origin and a second point

Example :

```
| a |
a := canvas point:  1@5.
canvas ray:  0@0 to:  a.
```

Segment.

`<segment> canvas segment:  p1 to:  p2`

`p1` : points or coordinates
`p2` : points or coordinates
$\longrightarrow$ segment defined by two points
Example :

```
| a |
a := canvas point:  5@5.
canvas segment:  10@10 to:  a.
```

Circle.

`<circle> canvas circleCenter:  c to:  p`

`c` : point or coordinates, centre of the circle
`p` : point or coordinates, point on the circle
$\longrightarrow$ circle defined by its centre and a point
Example :

```
| a |
a := canvas point:  1@5.
canvas circleCenter:  a to:  10@4.
```

`<circle> canvas circleCenter:  c radius:  r`

`c` : point or coordinates, centre of the circle
`r` : numeric item or numeric value, radius
$\longrightarrow$ circle defined by its centre and radius
Example :

```
| a r |
a := canvas point:  1@5.
r := canvas freeValue:  4.
canvas circleCenter:  a radius:  r.
canvas circleCenter:  4@4 radius:  5
```

Arc.

`<arc> canvas arc:  p1 to:  p2 to:  p3`

`p1` : point or coordinates, 1^{ere} extremity of the arc
`p2` : point or coordinates representing a point on arc
`p3` : point or coordinates, 2^{eme} extremity of the arc
$\longrightarrow$ arc defined by its extremities and a point
Example :

```
| a b |
a := canvas point:  1@5.
```

```
b := canvas point:  0@5.
canvas arc:  a to:  b to:  -1 @ -2.
```

`<arc> canvas arcCenter:  O from:  A to:  B`

`O` : point or coordinates, centre of the arc
`A` : point or coordinates, origin of the arc
`B` : point or coordinates, so the angle is $\widehat{AOB}$
$\longrightarrow$ arc defined by its centre and angle $\widehat{AOB}$
Example :

```
| a b |
a := canvas point:  1@5.
b := canvas point:  0@5.
canvas arcCenter:  a from:  b to:  -1 @ -2.
```

Polygon.

`<polygon> canvas polygon:  collection`

`collection` : collection of points or coordinates; summits of the polygon
$\longrightarrow$ polygon defined by its summits
Example :

```
| b |
b := canvas point:  1@3.
canvas polygon:  {1@2.  b.  0@0.  d.}.
```

`<polygon> canvas regularPolygonCenter:  center vertex:  summit sides:  number`

`center` : point or coordinates, centre of the polygon
`summit` : point or coordinate, a summit of the polygon
`number` : value item or numeric value, number of summits of the polygon
$\longrightarrow$ regular polygon defined by its centre, one summit and number of summits
Example :

```
| b |
b := canvas point:  1@3.
canvas regularPolygonCenter:  b vertex:  1@1 sides:  7.
```

Geometric transformations.
Geometric transformations are to transform any kind of geometric objects: point, segment, line, ray, vector, circle, arc and polygon.

`<transformed item> canvas rotate:  item center:  centre angle:  angle`

`item` : point, segment, line, ray, vector, circle, arc, polygon
`item` : object to transform
`centre` : point or coordinates, rotation centre
`angle` : value item or numeric value, rotation angle
$\longrightarrow$ transformed object
Example :

```
| c k l |
c := canvas point:  5@5.
k := 3.1415.
l := canvas line:  0@0 to:  5@5.
```

```
canvas rotate: l center: c angle: k.
canvas rotate: l center: 0@0 angle: Float pi / 3.
```

`<transformed item> canvas scale: item center: centre factor: k`

`item` : point, segment, line, ray, vector, circle, arc, polygon
`item` : object to transform
`centre` : point or coordinates, homothety centre
`k` : value item or numeric value, homothety factor
⟶ transformed object
Example :

```
| c k l |
c := canvas point: 5@5.
k := -3.
l := canvas line: 0@0 to: 5@5.
canvas scale: l center: c angle: k.
canvas scale: l center: 0@0 angle: 5.
```

`<transformed item> canvas symmetry: item center: centre`

`item` : point, segment, line, ray, vector, circle, arc, polygon
`centre` : point or coordinates, symmetry centre
⟶ transformed object
Example :

```
| a |
a := canvas point: 4@2.
canvas symmetry: a center: 0@0.
```

`<transformed item> reflect: item axe: axe)`

`item` : point, segment, line, ray, vector, circle, arc, polygon
`axe` : line, symmetry axe
⟶ transformed object
Example :

```
canvas reflect: polygon axe: d1.
```

`<transformed item> canvas translate: item vector: vector`

`item` : point, segment, line, ray, vector, circle, arc, polygon
`vector` : vector or coordinates
⟶ transformed object
Example :

```
| u a|
u := canvas vector: (canvas point: 1@1) to: (canvas point: 3@2).
a := canvas translate: (canvas point: 2@1) vector: u.
```

Example :

```
| u a|
a := canvas translate: (canvas point: 2@1) vector: 2@1.
```

Locus of a point.

`<locus> canvas locusOf: c when: m`

`m` : mobile point on a line

`c` : fixed point depending on the mobile point `m`
⟶ locus
Example :

```
canvas locusOf:  p when:  mobile.
```

Vector.

`<vector> canvas vector:  o to:  e`
`o` : point or coordinates, vector origin
`e` : point or coordinates, vector extremity
⟶ vector
Example :

```
| b |
b := canvas point:  0@5.
canvas vector:  b to:  -1 @ -2.
```

`<vector> canvas vector:  p`
`p` : point or coordinates, the vector coordinates
⟶ vector
Example :

```
| p |
p := canvas point:  5@5.
canvas vector:  p.
canvas vector:  -5 @ -5
```

Numebr.

`<point> pointItem coordinates`
`pointItem` : point
⟶ coordinates (static) of pointItem
Example :

```
| c p |
p := canvas pointOnCurve:  segment at:  0.5.
c := p coordinates.
c x
```

`<value> canvas abscissaOf:  pointOrVector`
`pointOrVector` : point or vector item
⟶ abscissa (dynamic) item of `pointOrVector`
Example :

```
| m x |
m := canvas middleOf:  10@5 and:  7@8.
x := canvas abscissaOf:  m
```

`<value> canvas ordinateOf:  pointOrVector`
`pointOrVector` : point or vector item
⟶ ordinate (dynamic) item of `pointOrVector`
Example :

```
| m x |
m := canvas middleOf:  10@5 and:  7@8.
x := canvas ordinateOf:  m
```

`<value> canvas freeValue:  v`

`v` : initial value
⟶ free value item
Example :

```
canvas freeValue:  (-1 arcCos).
```

`<value> canvas lengthOf:  segment | circle | arc | vector)`

`segment|circle|arc|vector` : segment, circle, arc or vector
⟶ number item, the item length
Example :

```
canvas lengthOf:  v1.
```

`<value> canvas distance:  line | point to:  point2`

`line|point` : line or point
`point2` : point
⟶ number, distance between two points or a point and a line
Example :

```
canvas distance:  l1 to:  a.
```

`<value> canvas slopeOf:  line`

`line` : line
⟶ number, slope of the line
Example :

```
| p |
p := canvas slopeOf:  d.
```

Angle.

`<value> canvas angle:  a to:  b to:  c`

`a` : point
`b` : point, angle summit
`c` : point
⟶ geometric angle $\widehat{abc}$ in the interval $[0\ ;\ \pi]$
Example :

```
canvas angle:  a to:  b to:  c
```

`<value> canvas angle:  v1 to:  v2`

`v1` : vector
`v2` : vector
⟶ oriented angle given two vectors in the interval $]-\pi\ ;\ \pi]$
Example :

```
| v1 v2 a |
v1 := canvas vector:  a to:  b.
v2 := canvas vector:  a to:  c.
a := canvas angle:  v1 to:  v2.
```

Equation.

`<equation> canvas equationOf: lineOrCircle`
`lineorCirlce` : line or circle
⟶ equation of the line or circle
Example :

```
| e d |
d := canvas line: 0@0 to: 15@13.
e := canvas equationOf: e
```

1.2.2 Modification of object attributes

To modify object attributes, we send messages to the objects. So the attributes are always modified after creating the objects.

`item color: aColor`
`item` : any object in the canvas
`aColor` : a `Color`, see methods in this class for existing colors : `Color black, Color red, Color blue, Color orange, Color yellow,...`
Action : modify the item color
Example :

```
pointA color: Color green.
```

`item name: aString`
`aString` : a string, text
Action : rename item
Example :

```
segment name: '[AB]'.
```

`item hide`
Action : hide an item

`item show`
Action : show an item

`line small`
`line` : line item (straight line, ray, circle, lieu, etc.)
Action : small thickness to the line
Example :

```
circle small.
```

`line normal`
`line` : line item (straight line, ray, circle, lieu, etc.)
Action : normal thickness to the line
Example :

```
arc normal.
```

`line large`
`line` : line item (straight line, ray, circle, lieu, etc.)
Action : donne une épaisseur large à une ligne
Example :
```
polygon large.
```

`line plain`
`line` : line item (straight line, ray, circle, lieu, etc.)
Action : plain, undotted, style line
Example :
```
polygon plain.
```

`line dashed`
`line` : line item (straight line, ray, circle, lieu, etc.)
Action : dash style line
Example :
```
polygon dashed.
```

`line dotted`
`line` : line item (straight line, ray, circle, lieu, etc.)
Action : dotted style line
Example :
```
arc dotted.
```

`point cross`
`point` : point item
Action : cross shape for point
Example :
```
a cross.
```

`point round`
`point` : point item
Action : round shape for point
Example :
```
a round.
```

`point square`
`point` : point item
Action : square shape for point
Example :
```
a square.
```

`point small`
`point` : point item
Action : small sized point
Example :
```
a small.
```

`point large`
`point` : point
Action : large sized point
Example :
```
a large.
```

`item moveTo: point`
`item` : point or value item
`point` : coordinates
Action : move the item to the given position, as long as it is possible
Example :
```
| a |
a := canvas point: 0@0.
a moveTo: 5@5.
canvas update
```

1.2.3 Complementary methods

The `DrGeoCanvas` class proposes in the category `helpers` complementary methods to ease the computation of complex, interactive sketches.

`canvas plot: aBlock from: x0 to: x1`
`aBlock` : block of code, with one argument, to describe a function
`x0` : number, lower abscissa of the curve
`x1` : number, higher abscissa of the curve
Action : display the representation curve of the function described in the block of code, in the abscissa interval [x0 ; x1]
Example :
```
| canvas |
canvas plot: [:x| x * x] from: -3 to: 3.
```

`<block> canvas float: float1 at: aPoint from: float2 to: float3 name: aStri`
`float1` : initial value
`aPoint` : left position of the ruler
`float2` : minimum value
`float3` : maximum value
`aString` : name of the value
$\longrightarrow$ block of code returning the current value of the ruler
Action : construct a ruler at the given position with a **decimal** value in the interval [float2 ; float3]
Example :
```
A := canvas float: 1 at: -10@4 from: 0 to: 10 name: 'A'.
F := canvas integer: 3 at: -10@3 from: 0 to: 10 name: 'F' showValue:
true.
A value + F value.
```

There are other variants, some for integer value.

1.3 Gallery of examples

To illustrate the use of Dr. Geo Smalltalk sketch, we present below a small set of examples. It shows you some possibilities and we hope can inspire you for your own need. For each

example, we give the Smalltalk source code, we encourage you to copy and past it in a Dr. Geo workspace and run the code.

1.3.1 Animer une figure

These examples rely on time handling and background process.

A first simple one the understand the concept:

```
| figure p pause |
figure:=DrGeoCanvas new.
p := figure point: 0@0.
pause := Delay forSeconds: 0.2.
figure do: [
   100 timesRepeat: [
      p mathItem moveTo: (p mathItem point + (0.1@0)).
      figure update.
      pause wait]]
```

A second example with a more elaborated sketch:

```
| figure s r u pause |
figure := DrGeoCanvas new fullscreen.
s := figure segment: 0@ -1 to: 4@ -1.
r := figure pointOnCurve: s at: 0.8.
s := figure segment: 0@0 to: 0@1.
u := figure pointOnCurve: s at: 0.7.
u round small; color: Color blue.
1 to: 100 do: [:n|
   u := figure
      point: [:parents| |y t|
         y := parents first point y.
         t := parents second point x.
         (n / 5) @ t * y * (1 - y)]
      parents:  {u . r}.
   u round small; color: Color blue].
pause := Delay forSeconds: 0.1.
figure do: [
   0 to: 1 by: 0.05 do: [:x |
      r mathItem setCurveAbscissa: x.
      figure update.
      pause wait]]
```

1.3.2 Sierpinski triangle

This example largely rely in recursive block of code..

```
| triangle c |
triangle := [:s1 :s2 :s3 :n |
    c segment: s1 to: s2;
        segment: s2 to: s3;
        segment: s3 to: s1.
    n > 0 ifTrue:
        [triangle
            value: s1
            value: (c middleOf: s1 and: s2) hide
            value: (c middleOf: s1 and: s3) hide
```

```
            value: n-1.
        triangle
            value: (c middleOf: s1 and: s2) hide
            value: s2
            value: (c middleOf: s2 and: s3) hide
            value: n-1.
        triangle
            value: (c middleOf: s1 and: s3) hide
            value: (c middleOf: s2 and: s3) hide
            value: s3
            value: n-1.]].

c := DrGeoCanvas new.
triangle
    value: (c point: 0 @ 3)
    value: (c point: 4 @ -3)
    value: (c point: -4 @ -3)
    value: 3.
```

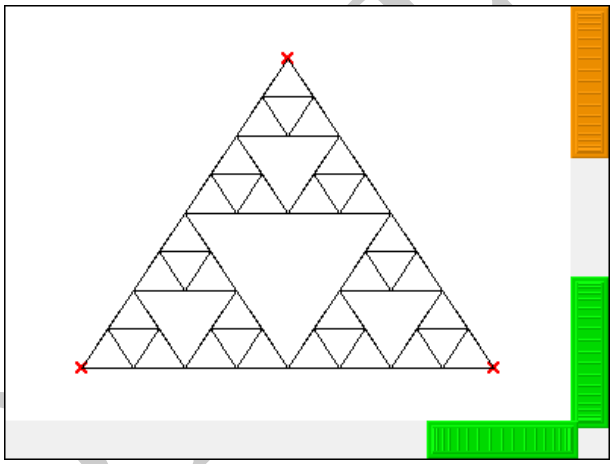

Figure 1.1: Triangle de Sierpinski