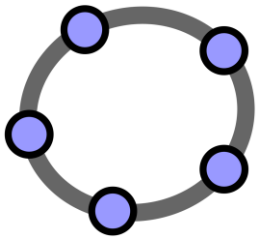


Conditional Visibility & Sequences

GeoGebra Workshop Handout 8

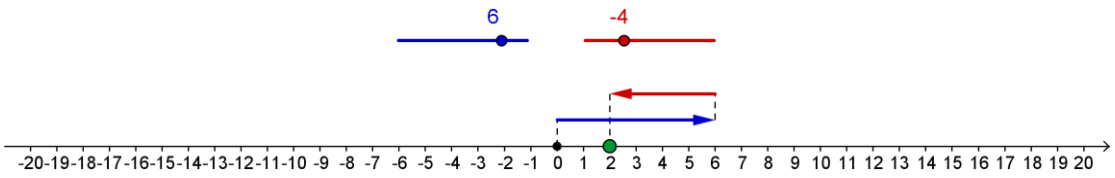
Judith and Markus Hohenwarter
www.geogebra.org

Table of Contents

1. Visualizing Integer Addition on the Number Line

Preparations

- Open a new GeoGebra file.
- Hide the algebra window (*View* menu).
- In the *Options* menu set the *Labeling* to *All new objects.*



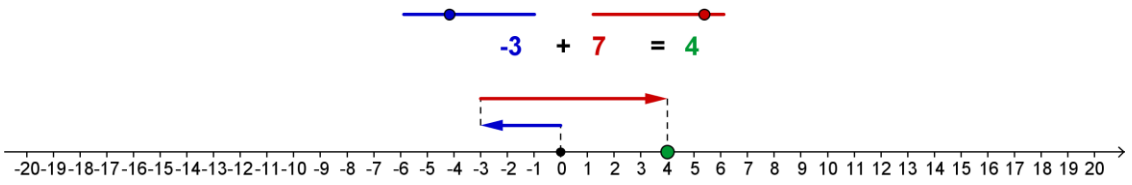
Instructions

1		Open the *Properties dialog for the drawing pad*. Hint: Right click (MacOS: *Ctrl* – click) on the drawing pad and select *Properties*.
2		On tab *xAxis* set the distance of tick marks to 1 by checking the box *Distance* and entering 1 into the text field.
3		Set the *minimum* of the *x*-Axis to -21 and the *maximum* to 21.
4		On tab *yAxis* uncheck *Show yAxis*.
5		Close the *Properties dialog for the drawing pad*.
6	a = 2	Slider for number *a* with *interval* -10 to 10 and *increment* 1.
7	a = 2	Slider for number *b* with *interval* -10 to 10 and *increment* 1.
8		Show the value of the sliders instead of their names. Hint: *Properties dialog – tab Basic – Show label*
9	A	Point `A = (0 , 1)`
10		Point `B = A + (a , 0)` Hint: The distance of point *B* to point *A* is determined by slider *a*.
11		Vector `u = Vector[A, B]` which has the length *a*.

12		Point `C = B + (0 , 1)`
13		Point `D = C + (b , 0)`
14		Vector `v = Vector[C , D]` which has the length *b*.
15		Point `R = (x(D) , 0)`. Hint: `x(D)` gives you the *x*-coordinate of point *D*. Thus, point *R* shows the result of the addition on the number line.
16	A	Point `Z = (0, 0)`
17		Segment `g = Segment[Z, A]`
18		Segment `h = Segment[B, C]`
19		Segment `i = Segment[D, R]`
20		Use the *Properties dialog* to enhance your construction (e.g. match the color of sliders and vectors, line style, fix sliders, hide labels).

Insert dynamic text

Enhance your interactive figure by inserting dynamic text that displays the corresponding addition problem. In order to display the parts of the addition problem in different colors you need to insert the dynamic text step by step.



1		Calculate the result of the addition problem: `r = a + b`
2	ABC	Insert dynamic *text1*: `a`
3	ABC	Insert static *text2*: `"+"`
4	ABC	Insert dynamic *text3*: `b`
5	ABC	Insert static *text4*: `"="`
6	ABC	Insert dynamic *text5*: `r`

7		Match the color of *text1*, *text3*, and *text5* with the color of the corresponding sliders, vectors, and point *R*.
8		Line up the text on the drawing pad.
9		Hide the labels of the sliders and fix the text (*Properties dialog*).
10		Export your interactive figure as a dynamic worksheet.

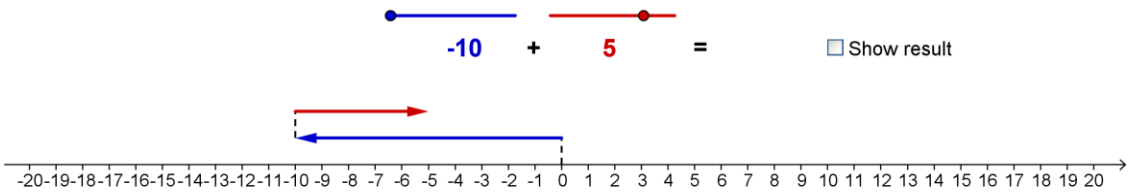
2. Conditional Formatting – Inserting Checkboxes

Introduction of new tool

	Checkbox to show and hide objects **New!** Hint: Click on the drawing pad to open the checkbox dialog window. Enter a caption and select the objects you want to show / hide using the checkbox from the drop down menu.

Instructions

Insert a checkbox into the graphics window that allows you to show or hide the result of the addition problem.



1		Activate tool *Checkbox to show and hide objects*.
2		Click on the drawing pad next to the result of the addition problem to open the checkbox dialog window.
3		Enter `Show result` into the *Caption* text field.
4		From the drop down menu select *text5*. This visibility of this object will be controlled by the checkbox. Hint: You can also click on *text5* in the graphics window to insert it into the list of objects influenced by the checkbox.
5		Click *Apply* to create the checkbox.
6		In *Move* mode check and uncheck the checkbox to try out if *text5* can be hidden / shown.

7	Fix the checkbox so it can't be moved accidentally any more (*Properties dialog*).

Boolean variables

A checkbox to show / hide objects is the graphical representation of a Boolean variable in GeoGebra. It can either be true or false which can be set by checking (Boolean variable = true) or unchecking (Boolean variable = false) the checkbox.

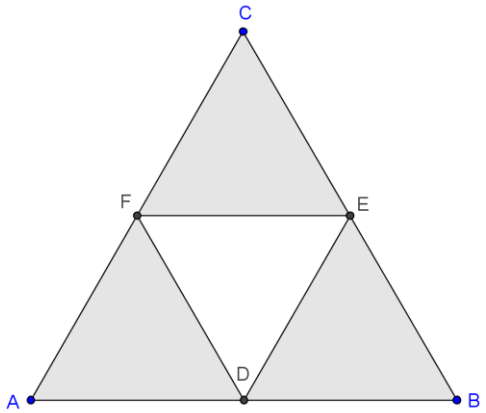
1. Open the *Properties dialog* and click on the + symbol in front of *Boolean value*. The list of Boolean values only contains one object called *j*, which is represented graphically as your checkbox.
2. Select *text5* from the list of objects in the *Properties dialog*.
3. Click on tab *Advanced* and look at the text field called *Condition to show object*. It shows the name of your checkbox *j*.
   Hint: This means that the visibility of *text5* depends on the status of the checkbox.
4. Select point *R* from the list of objects in the *Properties dialog*. Click on tab *Advanced*. The text field *Condition to show object* is empty.
5. Enter `j` into the text field *Condition to show object*. The visibility of point *R* is now connected to the checkbox as well.
6. Repeat steps 4 and 5 for segment *i* which connects the second vector with point *R* on the number line.

Hint: Now the checkbox controls three objects of your dynamic figure: *text5* (which shows the result of the addition), point *R*, and segment *i* (which show the result on the number line).

3. The Sierpinski Triangle

Preparations

- Open a new GeoGebra file.
- Hide the algebra window, coordinate axes, and input field (*View* menu).
- In the *Options* menu set the *Labeling* to *New points only.*



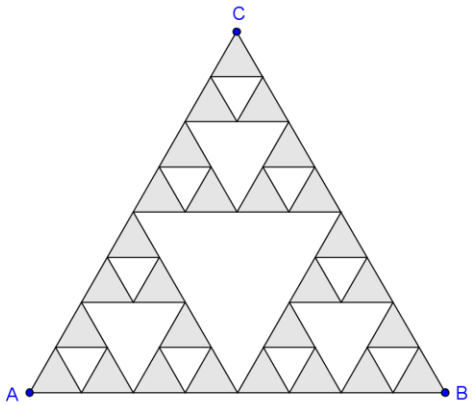
Instructions

You will now learn how to create a custom tool that facilitates the construction of a so called Sierpinski triangle.

1		Arbitrary triangle *ABC*
2		Change the color of the triangle to black (*Properties dialog*).
3		Midpoint *D* of triangle side *AB*
4		Midpoint *E* of triangle side *BC*
5		Midpoint *F* of triangle side *AC*
6		Triangle *DEF*
7		Change the color of triangle *DEF* to white and increase the filling to 100% (*Properties dialog*).
8		Change the color of the sides of triangle *DEF* to black (*Properties dialog*).
9		Create a new tool called *Sierpinski*. Output objects: points *D*, *E*, and *F*, triangle *DEF*, sides of triangle *DEF* Input objects: points *A*, *B*, and *C* Name: *Sierpinski* Toolbar help: Click on three points
10		Apply your custom tool to the three black triangles *ADF*, *DBE*, and *FEC* to create the second stage of the Sierpinski triangle.
11		Apply your custom tool to the nine black triangles to create the third stage of the Sierpinski triangle.

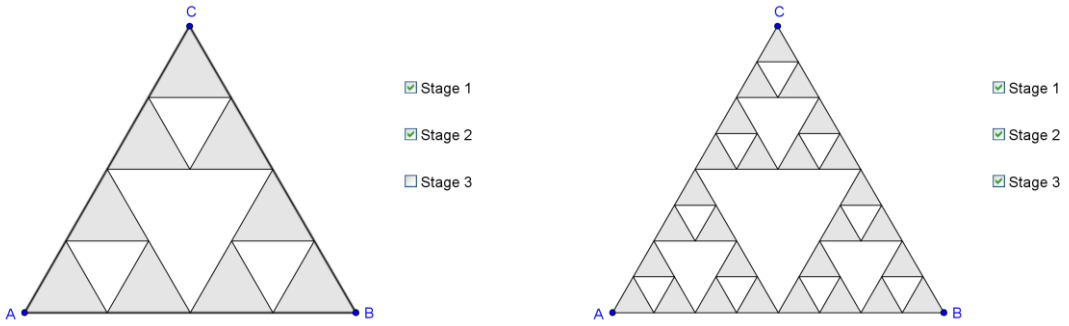
Conditional Visibility

Insert checkboxes that allow you to show and hide the different stages of the Sierpinski triangle.



1		Hide all points except from *A*, *B*, and *C*.
2		Create a *Checkbox to show and hide objects* that shows / hides the first stage of the Sierpinski triangle. Caption: Stage 1 Selected objects: Only large white triangles and its sides.
3		In *Move* mode check and uncheck the checkbox to try out if the white triangle and its sides can be hidden / shown.

4		Create a *Checkbox to show and hide objects* that shows / hides the second stage of the Sierpinski triangle. Caption: Stage 2 Selected objects: Three medium sized white triangles and their sides.
5		In *Move* mode check and uncheck the checkbox to try out if the second stage of the Sierpinski triangle can be hidden / shown.
6		Create a *Checkbox to show and hide objects* that shows / hides the third stage of the Sierpinski triangle. Caption: Stage 3 Selected objects: Nine small white triangles and their sides.
7		In *Move* mode check and uncheck the checkbox to try out if the third stage of the Sierpinski triangle can be hidden / shown.



4. Introducing Sequences

GeoGebra offers the command *Sequence* which produces a list of objects. Thereby, the type of object, the length of the sequence (that's the number of objects created), and the step width (e.g. distance between the objects) can be set using the following command syntax:

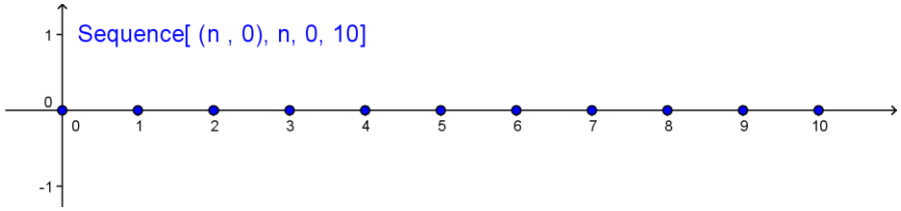
```
Sequence[<expression>, <variable>, <from>, <to>, <step>]
```

Explanations:

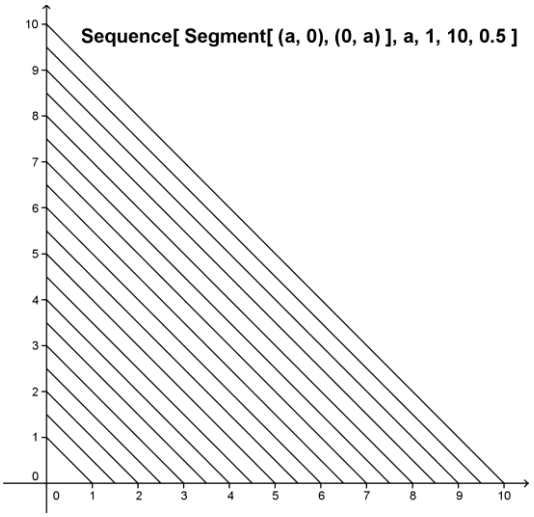
- `<expression>`:
  determines the type of objects created. The expression needs to contain a variable (e.g. `(i, 0)` with variable *i*).
- `<variable>`:
  tells GeoGebra the name of the variable used
- `<from>`, `<to>`:
  determine the interval for the variable used (e.g. from 1 to 10)
- `<step>`:
  is optional and determines the step width for the variable used (e.g. 0.5)

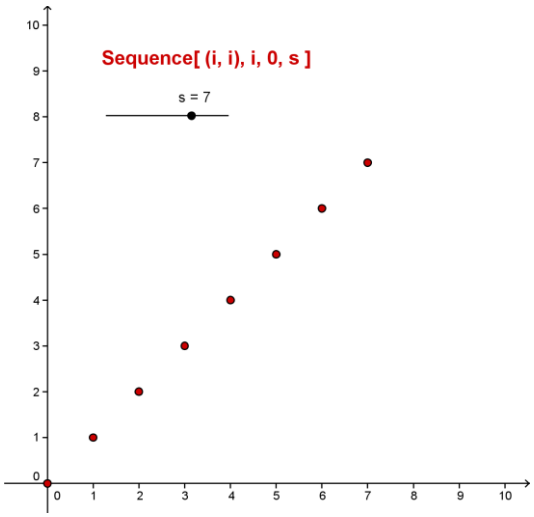
Examples for sequences

- `Sequence[(n, 0), n, 0, 10]`
  - creates a list of 11 points along the *x*-axis
  - points have coordinates *(0, 0)*, *(1, 0)*, *(2, 0)*, …, *(10, 0)*



- `Sequence[Segment[(a, 0), (0, a)], a, 1, 10, 0.5]`
  - creates a list of segments with distance 0.5
  - each segment connects a point on the *x*-axis with a point on the *y*-axis (e.g. points *(1, 0)* and *(0, 1)*; points *(2, 0)* and *(0, 2)*



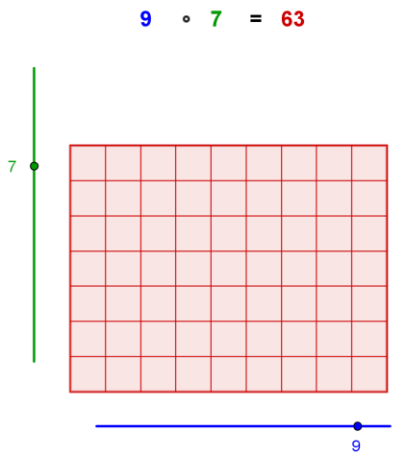


- If *s* is a slider with interval from 1 to 10 and increment 1, then command `Sequence[(i, i), i, 0, s]`
  - creates a list of *s + 1* points whose length can be changed dynamically by dragging slider *s*
  - points have coordinates *(0, 0)*, *(1, 1)*, …, *(10, 10)*

5. Visualizing Multiplication of Natural Numbers

Preparations

- Open a new GeoGebra file.
- Hide the algebra window and coordinate axes (*View* menu).
- Show the input field (*View* menu).
- In the *Options* menu set the *Labeling* to *All new objects.*

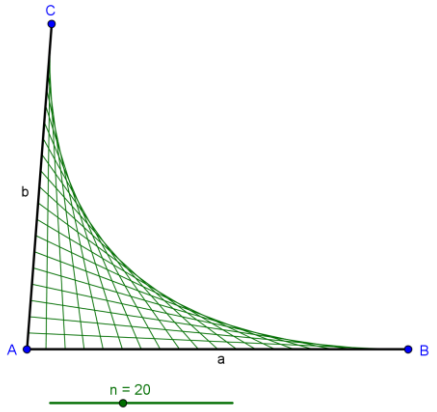


Instructions

1		Horizontal slider *Columns* for number with interval from 1 to 10, increment 1, and width 300
2		New point *A*
3		Segment *a* with given length *Columns* from point *A*
4		Move slider *Columns* to check the segment with given length.
5		Perpendicular line *b* to segment *a* through point *A*
6		Perpendicular line *c* to segment *a* through point *B*
7		Vertical slider *Rows* for number with interval from 1 to 10, increment 1, and width 300
8		Circle *d* with center *A* and given radius *Rows*
9		Move slider *Rows* to check the circle with given radius.
10		Intersect circle *d* with line *c* to get intersection point *C*
11		Parallel line *e* to segment *a* through intersection point *C*
12		Intersect lines *c* and *e* to get intersection point *D*
13		Polygon *ABDC*
14		Hide all lines, circle *d*, and segment *a*.
15		Hide labels of segments

16		Set both sliders *Columns* and *Rows* to value 10.
17		Create a list of vertical segments `Sequence[Segment[A+i(1, 0), C+i(1, 0)], i, 1, Columns]` Note: `A + i(1, 0)` specifies a series of points starting at point *A* with distance 1 from each other `C + i(1, 0)` specifies a series of points starting at point *C* with distance 1 from each other `Segment[A + i(1, 0), C + i(1, 0)]` creates a list of segments between pairs of these points. Note, that the endpoints of the segments are not shown in the graphics window. Slider `Column` determines the number of segments created.
18		Create a list of horizontal segments `Sequence[Segment[A+i(0, 1), B+i(0, 1)], i, 1, Rows]`
19		Move sliders *Columns* and *Rows* to check the construction.
20	ABC	Insert static and dynamic text that state the multiplication problem using the values of sliders *Columns* and *Rows* as the factors: *text1*: `Columns` *text2*: `*` *text3*: `Rows` *text4*: `=`
21		Calculate the *result* of the multiplication: `result = Columns * Rows`
22	ABC	Insert dynamic *text5*: `result`
23		Hide points *A*, *B*, *C*, and *D*
24		Enhance your construction using the *Properties dialog*.

6. Challenge of the Day: String Art Based on Bézier Curves



Bézier curves are parametric curves used in computer graphics. For example, they are used in order to create smooth lines of vector fonts. Let's create some 'string art' based on Bézier curves.

Preparations

- Open a new GeoGebra file.
- Show the input field (*View* menu).
- Hide the algebra window and coordinate axes (*View* menu).
- In the *Options* menu set the *Labeling* to *All new objects.*

Instructions

1		Segment *a* with endpoints *AB*
2		Segment *b* with endpoints *AC*
3		Slider for number *n* with interval 0 to 50, increment 1, and width 200
4		Create `Sequence[A + i/n (B - A), i, 1, n]` Hint: This sequence creates a list of *n* points along segment *AB* with a distance of one nth of the length of segment *a*.
5		Create `Sequence[A + i/n (C - A), i, 1, n]` Hint: This sequence creates a list of *n* points along segment *AC* with a distance of one nth of the length of segment *b*.
6		Hide both lists of points.
7		Create a list of segments `Sequence[Segment[Element[list1,i],Element[list2,n-i]],i,1,n]` Hint: These segments connect the first and last, second and last but one,…, last and first point of *list1* and *list2*.
8		Enhance your construction using the *Properties dialog*.
9		Move points *A*, *B*, and *C* to change the shape of your Bezier curve.
10		Drag slider *n* to change the number of segments that create the Bezier curve.

Note: The segments you just created are tangents to a quadratic Bézier curve.

Task

Create more ‘string art’ with GeoGebra using sequences of points and segments.