

Generative Components 101

Lesson 6: Series

What's a Series?

Glossary: () Round Brackets are used to define **functions** like a Series.
Twenty four episodes of ER on a Thursday night is a TV series! Basically a Series is a collection of points along a line or a surface.
We define a Series inside parentheses (**start, finish, increment**)
These inputs will generate and return a collection of numbers from the given start to the given finish, at intervals specified by the given increment.
For example:
Series(1, 10, 1) returns {1, 2, 3, 4, 5, 6, 7, 8, 9, 10} – starts at 1, ends at 10 and is at intervals of 1.
Series(1, 10, 2) returns {1, 3, 5, 7, 9}
Series(2, 3, 0.25) returns {2, 2.25, 2.5, 2.75, 3}

So, when would I need to use a series?

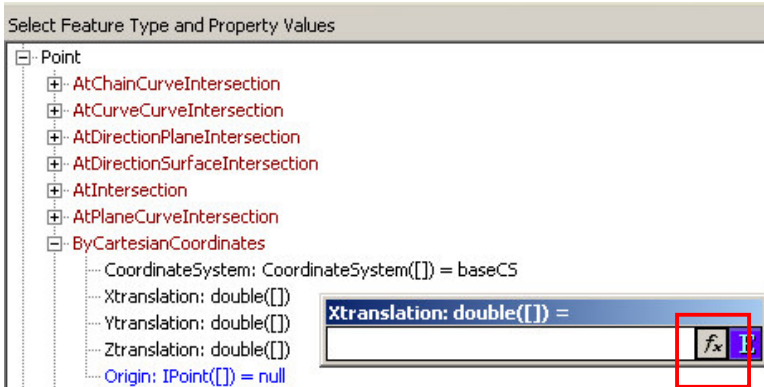
When we want to have multiple sets of points defining our geometry and we know exactly what the increments between those points should be.

Say what?

Ok, lets do an example so you can see what I mean.

Point>ByCartesianCoordinates

In the past we have inserted absolute numbers in the X and Y and Z coordinates. This time we are going to use a Series.



Use the baseCS as the coordinate system.

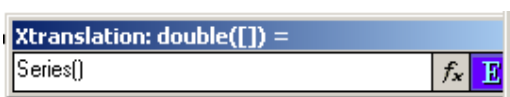
For the X translation we want to insert our Series.

Lets assume we want to create 10 points on the X-axis of equal distance apart. Lets define the (start, finish, increment) to be (5,15,1). A Series is a Function. This means we go to the *fx* tool to pick Series from the Function list box.

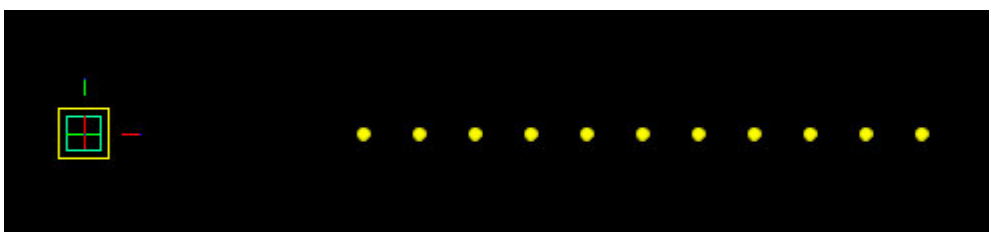
Glossary: Function. Imagine you have lots of apples. If you put those apples into a juicer then you end up with apple juice. The juicer is like a Function. It takes the inputs and turns it into something else. A Series is a Type of Function.
Notice that each Function has a short description which tells you what how to write the syntax (how to write the inputs) and what the Function returns (gives you back).

CScript Functions	
Function	Description
CreateChildFeature(featureTypeName,...)	Returns a newly-created child feature
CreateFeature(featureTypeName)	Returns a newly-created simple feature
CreateTopLevelFeature(featureTypeName,...)	Returns a newly-created top-level feature
CStr(any)	Convert a value to its equivalent string-type representation.
DegToRad(degrees)	Converts from degrees to radians.
DeleteFeature(feature)	Delete a feature from the graph
Distance(point1, point2)	Returns the distance between two points.
DotProduct(vector1, vector2)	Returns the dot product of two vectors.
Exp(n)	Returns e raised to the power of a given number.
First(collection)	Returns the first item in a given collection.
Floor(n)	Rounds a number down to the nearest whole number.
Format(formatString, value [, value...])	Returns a string which is the result of applying the given format string to the given
Last(collection)	Returns the last item in a given collection.
Limit(n, min, max)	Returns a number limited to a given range.
Log(n)	Returns the natural (base e) logarithm of a number.
Log10(n)	Returns the base-10 logarithm of a number.
Max(n1, n2)	Returns the larger of two numbers.
Middle(collection)	Returns the middle item in a given collection.
Min(n1, n2)	Returns the smaller of two numbers.
Pow(x, y)	Returns the result of x raised to the power y.
RadToDeg(radians)	Converts from radians to degrees.
Random([n1 [, n2]])	Returns a random number.
RandomSeed(n)	Provide a starting seed for the random number generator.
Rank(collection)	Returns the number of dimensions in a given value.
Round(n, optional precision)	Rounds a value to a given number of places after the decimal point.
Select(collection, function [, arg...])	Returns a collection consisting of all items from the given collection that satisfy a g
Series(start, final, increment)	Creates a collection from a given start number, final number and increment.
ShowMessageBox(message)	Displays a message box and waits for the user to press the OK button.
Sign(n)	Returns -1, 0 or 1, based on whether a given number is negative, zero or positive.
Sin(angle)	Returns the sine of an angle (which is measured in degrees).
Sinh(n)	Returns the hyperbolic sine of a number.
Sqrt(n)	Returns the square root of a number.
Subcollection(collection, indices)	Creates a collection from specific items obtained from a given collection.
Tan(angle)	Returns the tangent of an angle (which is measured in degrees).
Tanh(n)	Returns the hyperbolic tangent of a number.
Trunc(n)	Truncates a number to an integer by removing its fractional part.
TypeName(any)	Returns a string that identifies the type of a value.
UpdateGraph()	Update the graph with any changes made

Double click on Series or choose Series and click Insert Global Function. Then choose Close at the bottom right of the dialog.



Place your cursor in the brackets to enter the (start, finish, increment). Let's make our series be Series(5,15,1) (Start at 5, end at 15, in increments of 1)
 Lets make the Y Translation 0
 And the Z Translation 0
 Apply.



What did we get? 10 points along the X-axis equidistant apart. (No, not 15 points, we started at 5 not 0 remember.)

Let's now edit the points so that the Series is in the Y direction instead of the X.

How do I edit?

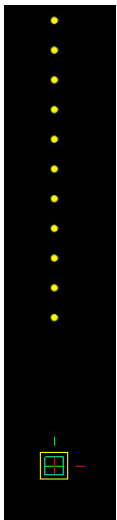
The Edit Feature Tool, remember? Click on one of the points.



```

Point
├── Settings
└── ByCartesianCoordinates
    ├── CoordinateSystem: CoordinateSystem([]) = baseCS
    ├── Xtranslation: double([]) = Series(5,15,1) ({5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15})
    ├── Ytranslation: double([]) = 0
    ├── Ztranslation: double([]) = 0
    ├── Origin: IPoint([]) = null
    ├── X: double = {5.0, 6.0, 7.0, 8.0, 9.0, 10.0, 11.0, 12.0, 13.0, 14.0, 15.0}
    ├── Y: double = {0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0}
    └── Z: double = {0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0}
    
```

You will notice upon editing the full Series of numbers appears. Use copy and paste to transfer the Series(5,15,1) over to the Y Translation and make the X Translation 0.

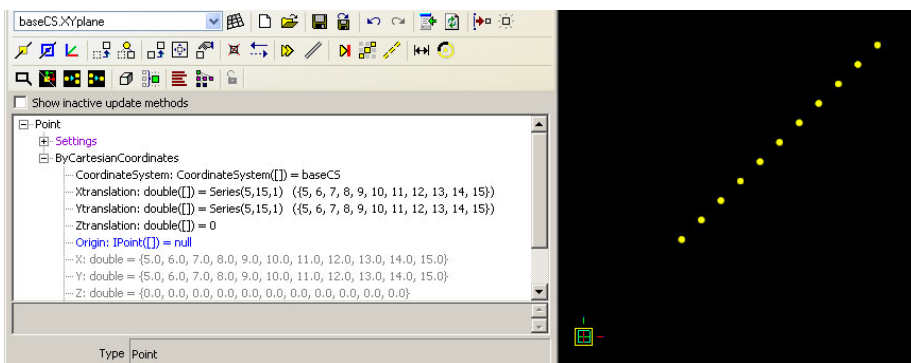


```

Point
├── Settings
└── ByCartesianCoordinates
    ├── CoordinateSystem: CoordinateSystem([]) = baseCS
    ├── Xtranslation: double([]) = 0
    ├── Ytranslation: double([]) = Series(5,15,1) ({5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15})
    ├── Ztranslation: double([]) = 0
    ├── Origin: IPoint([]) = null
    └── X: double = {0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0}
    
```

And try again for the Z location! You may have to rotate the view to see that one!

What happens when I put the Series in both the X and Y translations?



Yep, you guessed (or did you?). The points are now equidistant between the X and Y directions.

Add the series into Z as well and the points will move up into the Z direction too.

Rotate your view to see.

Amazing!

Karen Fugle