Using Subassembly Composer

Subassembly Composer is a very powerful utility that will allow you to create your own custom assemblies. I am going to try and help you understand how to use it to its fullest potential.

First, I will describe the different components of the program. Second, I will describe how some of the features you’ll see will affect the final result you see within the Civil 3D program. Third, I will outline the workflow that I used in creating/using the Boat Ramp Subassembly.

The Components (See Figure 1)

1. The Toolbox – Contains all of the different types of objects that you can use to either draw points, lines or shapes OR control which of those do or do not get shown.
   a. Geometry
Using Subassembly Composer

(continued)

i. Point – an individual point from which a link can be drawn.
ii. Link – a line between two points
iii. Shape – 3 or more links connected and closed

b. Advanced Geometry
i. Intersection Point – will create a point of intersection:
   1. where two links intersect
   2. where one link intersects with an imaginary line drawn from one specified point along a slope
   3. where two imaginary lines each drawn from specified points at specified slopes
ii. Curve –
iii. Surface Link – Creates a link that follows the specified surface between two specified points.
iv. Daylight Rounding –
v. Get Mark Point –
vi. Fillet Arc –
c. Auxiliary (The same as their counter parts shown above but will not show up in the final result – used as temporary/helper items).

⚠️ NOTE: A “normal” link cannot be created to or from an auxiliary point.

2. Flowchart – Shows you the order in which Civil 3D will read the Subassembly. Once you start using conditionals (Decision, Switch), this will become extremely important as this flowchart will allow you to visually see how the Subassembly will flow to and from one element to the various branches of your conditionals.

3. Properties – Lists all of the properties of each element you place in the Subassembly.

4. Input/Output Parameters – The options that the end user will be able to set will be listed here.
   a. The “Side” parameter will always be listed and the name cannot be changed.
   b. Create new parameters by clicking on “Create Parameter”.
   c. If you want to delete one, SINGLE CLICK on it and press the Delete key on your keyboard.

5. Packet Settings
   a. Subassembly Name – The name that will be seen within Civil 3D once you’ve imported the subassembly.

   ⚠️ NOTE: The name MUST begin with ONE letter character and be followed by one or more letters, numbers and/or underline characters.

   b. Description – What the user will see if he hovers the mouse over the subassembly in the toll palette.

   c. Help File – An HTML or CHM file that will be displayed if the user right-clicks on the subassembly within the tool palette and then clicks on “Help”.

   ⚠️ NOTE: I have not been able to get the help file to show up.
Using Subassembly Composer
(continued)

d. Image file – An image file that will be shown on the tool palette after the subassembly has been imported.

6. Target Parameters - Specify what possible targets to search for here. The user will see the possible targets in the corridor properties (Figure 2). In the Boat Ramp subassembly, the targets were named “Surface” and RockBlanketTarget.

![Figure 2](image)

7. Superelevation – Set the options for superelevation here.

8. Event Viewer – If any element in the subassembly causes an error or warning, it will be listed here. If there is a message here, a red circle with an X will appear on the tab name (See Figure 3).

![Figure 3](image)

9. Preview Window – Shows what the subassembly will look like given the defaults you have entered in the various parameters.

NOTE: In the lower left corner, you will see two check boxes: “Codes” & “Comments. If you turn on “Codes”, each Point, Link & Shape will show the corresponding code next to the name. If you turn on “Comments, the comments will show up next to the name (and code if they’re on). Comments are only visible in SAC.

The Effects
When you create a Point, Link or Shape (excluding the Auxiliary types), each of them have a “Code” property. This is the code that will appear within Civil 3D and allow you to create feature lines and shapes.

1. A code entry can actually contain multiple codes as long as they are separated by a comma.
Using Subassembly Composer
(continued)

2. The code(s) must be entered as text strings enclosed in quote marks. If you do not use quote marks, SAC will assume you’re attempting to enter a variable. (The use of a variable here can be an extremely beneficial idea).

3. If you create a Point with a code assigned to it, it will appear in Civil 3D as a Feature line between this point in one assembly instance and its corresponding point within the next assembly instance.

4. If you create a Link with a code assigned to it, it will appear in Civil 3D as a shape in plan view.
   a. The two points within one assembly instance will be connected and will represent one side of a shape.
   b. The two points will be connected to their corresponding points in the next assembly instance to create two more sides of the shape.
   c. And the corresponding points in the next assembly instance will be connected to create the fourth side of the shape.

5. If you create a Shape with a code assigned to it, it will appear in Civil 3d as a shape in Cross Section view (it will appear much the same as it does in the SAC preview window).

The Workflow
First and foremost, to utilize this tool to its fullest potential, you will need to thoroughly plan what it is you want to accomplish.

1. Before even starting SAC, I sat down with a pen and paper and
   a. Sketched out a flowchart listing how it would flow from the attachment point (the baseline in my case) and through each subsequent point.
   b. Listed all the parameters that would be needed (Target Surface, Horizontal Offset Alignments to target, Curb Start, Curb End, Pavement Width etc)
   c. Listed all the different Point Codes, Link Codes and Shape Codes that would be needed.

2. Having my sketch, parameters and codes all laid out, I started SAC and got to work.
   a. Start/Origin represents the attachment point.
   b. First I created a “Sequence” (found under Workflow in the toolbox). In that Sequence, I created several variables representing each of the possible codes/combination of codes.
      i. If I ever need to change the name of a code, I’ll only have to change it ONCE.
      ii. Here are several of the variables listed (there are others but this is just to explain their use):
         1. Top - contains the code “Top”
         2. Daylight – contains the code “Daylight”
         3. RockFill – contains the code “RockFill
         4. Rockfill_Top contains two previous variables concatenated together:
            a. RockFill + “,” Top – because of the way these two codes were entered previously, this will result in the variable being “RockFill,Top”
            c. I created a new point (“P1”) using DeltaX and DeltaY where both are set to zero. The code is “Crown,Top” (using a variable of course).
Using Subassembly Composer
(continued)

d. Then if the station is between the values specified by CurbStart and CurbEnd (Input parameters) which by default are 0+00 and 0+40, we want to draw curb which means that the eop will end at the face of curb. So there is a Decision (if statement) that checks the value of Baseline.Station.
   i. If (Baseline.Station=CurbStart or Baseline.Station=CurbEnd) OR (CurbStart < Baseline.Station < CurbEnd)
      1. Then set PavementWidth (a variable) to PaveW (an input parameter, default is 8”) minus CurbW (another input parameter, default is 6”). Also set a variable “UseCurb” to Yes.
      2. Otherwise, we are outside of the curb stations and should show no curb. So set PavementWidth to PaveW (default is 8”). Also set a variable “UseCurb” to No.

e. Draw the pavement box using the width value specified by the variable “PavementWidth” and the input parameter “PaveH” which is the pavement height (thickness).

f. Now if the variable “UseCurb” is Yes, then draw the curb. If it’s No, then do not draw the curb. However the daylight slope line needs a point to start from. It needs to start from P6 if there is curb and from P2 if there is NO curb. If there is no curb, then P6 will not exist. Therefore in the False branch of this Decision, I created a new point (“P6”) From P2 by DeltaX and DeltaY. Both DeltaX and DeltaY are set to 0.

g. Create an Auxiliary Point (“AP3”) that targets the surface.

h. Now we have another Decision: if AP3.y (the height of AP3) is greater than P6.y (the height of P6), draw the daylight in CutSlope. Otherwise, draw the daylight in FillSlope.

i. From the Daylight point, we have another Decision:
   i. If UseRockBlanket is set to No,
      1. Then we will draw daylight without a RockBlanket. The code assigned will use the Variable Daylight_Top.
      2. Then Show the RockBlanket.
        a. If the daylight point is before the horizontal offset target,
           i. Then we will use a surface link to continue the RockBlanket along the surface at a depth specified by the variable RockBlanketDepth.
           ii. Otherwise, show the RockBlanket ONLY “under” the daylight slope.
Using Subassembly Composer
(continued)

Examples: (Notice the effects of changing the preview values)

Surface is -5 and RockBlanketTarget is 30

Surface is 5 and RockBlanketTarget is 30