Design Materials List, Fall 2014

Cinthia Kim, Aminta Nunez, Serena Takada

Dec 12, 2014

Part 1: Documented Progress

Introduction

For Fall 2014, Aminta Nunez and Cinthia Kim worked on refining the Materials list. The Materials list challenge consists of the extracting and cataloging the materials utilized in the AguaClara plant from the Design files. This challenge is imperative to collect all components into a systematic format; ultimately, allowing the estimation and optimization of the materials needed for an efficient building process and developing an approximate budget. The plan started with the extraction and optimization of baffles, and advanced to the collection of all hydraulic components from the plant. In order to extract the attributes of hydraulic components, hydraulic components were first blocked in MathCad drawing files, then the blocking information was extracted from the AutoCAD drawing to be examined. This method of extracting a materials list is more efficient than the manual method that is currently used by AguaClara members in Honduras. Hence, every time a new plant is constructed, engineers in Honduras use AutoCAD and MathCAD files to laboriously count and measure the pipes and other hydraulic components needed for construction. The materials list under development is not only more efficient, but also more professional. Collaborating with Serena Takada, the team created functions used in the blocking scripts.

Design Details

AguaClara Design Team members have worked on developing a materials list in the past. These members include Kira Gidron during Fall 2012, Annie Ding during Spring 2013 and 2014, and Meghan Furton during Spring 2014. In the past, team members used MathCAD to create an algorithm for the cutting optimization, and a Pipe Matrix for the Stacked Rapid Sand Filter, which needs further modifications. Aside from the algorithms, the cost inputs were also evaluated based off data from 2012. Additionally, the Materials List for the plant in Morocelí created by Drew Hart was used to determine the accessories that should be included and developed in this semester's materials list.

Based on the methods used in the past by team members and the materials list developed by Drew Hart, the team initially planned to extract the materials list from the MathCad files. The accessories needed in the flocculator and entrance tank were extracted from these files and organized into matrices. Later, the accessories in the matrices were called together in one stack. However, the current design tool cannot create an excel file of the materials list from the matrices, and therefore this method was abandoned.

Instead, the team decided to use the blocking commands in AutoCAD to generate a more efficient materials list. Block scripts improve the materials list extraction by first blocking objects, then inserting them with different attributes, and finally extracting the block attributes

into an organized table, which can be used as a materials list. Excluding the pipes that vary depending on nominal diameter and length, the team concentrated on creating a materials list for the hydraulic components and the accessories that depend on only nominal diameters for sizes. The complete work of the Materials List Design team can be found in the Under Development folder in the MaterialsListEntranceTankBlocks file.

Reports

Table I: Past Progress & References

Semester	Designer	Reference
Fall 2012	Kira Gidron	Kira's
Spring 2013	Annie Ding	Annie's
Spring 2014	Annie Ding & Meghan Furton	Annie and Meghan
	Drew Hart	Moroceli Materials List

Part 2: Documented Progress

Accomplishments

10/08/14

Flocculator Baffle Matrix

The method used to extract the desired components for a flexible and comprehensive Baffle Matrix was to create separate matrices for total numbers of baffles for the Upper and Lower Baffles, and for dimensions of the Upper and Lower Baffles. These components were segregated because the units of these two components had to match in order to be stored in one Baffle Matrix. To account for the commercial process, the restriction data of the specific commercial baffles lengths, ordered (8, 10, 12, 14, 16, 18, 20) ft, was vectorized as Lengths of Available Baffles. Note the units have to match the metric units used in the baffle optimization process. For the optimization of the baffles, a Waste array was built to calculate the difference between the ratio of the commercial lengths and the needed baffle length and the greatest integer less than the ratio. Then the total waste matrix produced by each commercial length were determined by the product of Waste and the total number of baffles. From the total waste matrix, the minimum value was extracted and a function was programmed to determine the desired index of the commercial baffle length that produces the least waste.

$$N_{Baffles} := \begin{pmatrix} N_{FlocBaffleLowerTotal} \\ N_{FlocBaffleUpperTotal} \end{pmatrix}$$

$$Baffles := \begin{pmatrix} L_{FlocBaffleLower} & W_{FlocBaffle} \\ L_{FlocBaffleUpper} & W_{FlocBaffle} \end{pmatrix}$$

Figure I. Flocculator Baffle Matrices

The Commercial Restrictions

The commercial baffle lengths available are 8ft, 10ft, 12ft, 14ft, 16ft, 18ft, and 20ft only, with a constant width of 1.07m (3.5105ft). The price of $4.05 per foot were also defined as Price for the Price calculations

$$L_{AvailableBaffles} := \begin{pmatrix} \frac{8}{3.281} \\ \frac{10}{3.281} \\ \frac{12}{3.281} \\ \frac{14}{3.281} \\ \frac{16}{3.281} \\ \frac{18}{3.281} \\ \frac{20}{3.281} \end{pmatrix} \qquad W_{CommBaffle} := 1.07m \qquad Price := 4.05 \frac{USD}{ft}$$

Figure II. Commercial Restrictions

Optimization

To optimize the baffles cuts and have the minimum waste possible, some functions were created to determine the least waste considering all the commercial baffles available.

Waste Function

The waste function determines the waste from the commercial baffles available lengths' cuts. Using the length of the baffles needed as an input, the function calls the commercial lengths available matrix, and calculates the waste, which is the difference between the ratio of the length of the available baffles and the length of baffle needed and the floor of the same ratio.

$$\text{Waste}(\text{length}) := \frac{L_{\text{AvailableBaffles}}}{\text{length}} - \text{floor}\left(\frac{L_{\text{AvailableBaffles}}}{\text{length}}\right)$$

Figure III. Waste Function

Index Function

This function was created to find the position of the minimum waste in the waste array, to be utilized as the index in the commercial lengths available matrix.

$$\text{Index}(\text{Waste}) := \left| \begin{array}{l} i \leftarrow 0 \\ \text{for } i \in 0..6 \\ \quad i \text{ if } \text{Waste}_i \leq \min(\text{Waste}) \end{array} \right.$$

Figure IV. Index Function

Number of Baffles in a Commercial Baffle Function

This function determines the number of whole baffles that can be extracted from a commercial baffle executing the length cuts.

$$N_{\text{BaffleInCommercialBaffle}}(\text{length}, \text{Waste}) := \text{floor}\left[\frac{L_{\text{AvailableBaffles}_{(\text{Index}(\text{Waste}))}}}{\text{length}}\right]$$

Figure V. Number of Baffles in Commercial Baffle

Width Cut Function

The minimum width of the flocculator baffles is 45 centimeters, therefore baffle cuts of the commercial baffles may produce only up to two baffles. In the width cut function, if the width needed of the baffles is less than the half width of the commercial baffles, then two baffles can be extracted from one baffle.

$$N_{\text{WidthCuts}}(\text{Wneeded}, \text{Wcomm}) := \left| \begin{array}{ll} 2 & \text{if } \text{Wneeded} \leq \frac{\text{Wcomm}}{2} \\ 1 & \text{otherwise} \end{array} \right.$$

Figure VI. Width Cut Function

Number of Total Commercial Baffles

Multiple baffles needed can be made from a specific commercial baffle. Hence, this function evaluates the total number of commercial baffles needed by dividing the number of lower or upper baffles needed by the number of lengths or widths possible from the commercial baffle.

$$N_{TotalBaffles}(Nbaffles, Lcuts, Wcuts) := \frac{Nbaffles}{Lcuts \cdot Wcuts}$$

Figure VII. Number of Total Commercial Baffles

Cost Function

The cost function is the product of the number of commercial baffles, the sizes of the baffles, and the price per foot.

$$BaffleCost(n, size, price) := n \cdot size \cdot price$$

Figure VIII. Cost Function

10/31/14

Order of the Materials List Matrices

Following the water flow of the AguaClara plant, the accessories of the hydraulic components were extracted from the design files. In conjunction to the Moroceli materials list, only the necessary accessories were extracted. The nominal diameter, quantity, length, width, and specifications were determined and presented together in a matrix that states the hydraulic component, accessory/pipe, nominal diameter/width, specification, quantity, and length, in that order.

Entrance Tank

The extracted hydraulic components of the entrance tank were rapid mix, overflow drain, gate valve, drainage hoppers, and trash rack. Below is an example of a matrix called MSRapidMix. This is an example for the Rapid Mix:

$$MS_{RapidMix} := \begin{pmatrix} \text{"RM"} & \text{"Elbow"} & \frac{ND_{RMPipe}}{m} & PS_{Default} & 1 & 0 \\ \text{"RM"} & \text{"Pipe Vertical"} & \frac{ND_{RMPipe}}{m} & PS_{Default} & 1 & \frac{L_{RMPipeVert}}{m} \\ \text{"RM"} & \text{"Pipe Horizontal"} & \frac{ND_{RMPipe}}{m} & PS_{Default} & 1 & \frac{L_{RMPipeHoriz}}{m} \\ \text{"RM"} & \text{"Coupling"} & \frac{ND_{RMPipe}}{m} & PS_{Default} & 1 & 0 \\ \text{"RM"} & \text{"Orifice Plate"} & \frac{ND_{RMPipe}}{m} & PS_{Default} & 1 & 0 \end{pmatrix}$$

Figure IX. Rapid Mix Matrix

In the same MathCAD file, the five extraction matrices were created for each component of the Entrance Tank and stacked in MSEntraceTank call script in the Flow Control Drain Grit section of the Entrance Tank. Then an Excel file output was created to extract all the specifications of the pipes and accessories.

Component	Accessories/Pipes	Diameter/Width (m)	PS	Quantity	Length (m)
Trash Rack	Trash Rack	1.238175	0	1	0.3
RM	Elbow	0.254	8	1	0
RM	Pipe Vertical	0.254	8	1	1.9887328
RM	Pipe Horizontal	0.254	8	1	1.8739563
RM	Coupling	0.254	8	1	0
RM	Orifice Plate	0.254	8	1	0
Hopper Drains	Couplings	0.0762	8	3	0
Hopper Drains	Stopper Pipes	0.0762	8	3	1.2297503
Over Flow	Tee	0.1524	8	1	0
Over Flow	Cap	0.1524	8	2	0
Over Flow	Pipe Vertical	0.1524	8	1	0.9777971
Over Flow	Pipe Horizontal	0.1524	8	1	0.427
Flow Control	Elbow	0.1524	8	0	0
Flow Control	Pipes	0.1524	8	1	0.9907015
Flow Control	Gate Valve	0.1524	8	1	0
Flow Control	Male Adapter	0.1524	8	2	0

Table II. Entrance Tank Excel Data Sheet

This table includes all the specifications of the accessories and pipes needed to build the Entrance Tank.

11/21/14

Blocking Commands in Scripts

The team found the unused block function in the Flex Cylinder tab of the MtoATranslator EtFlocSedFi file. The block function receives the name and reference points (base point and insertion point) as inputs and blocks all objects drawn in the current layer.

```
block(title, bp, ip) := | one ← concat("-block", sp, title)
                        | two ← concat(point(bp^⟨0⟩), sp, "all", sp)
                        | three ← concat("-insert", sp, title)
                        | four ← concat(point(ip^⟨0⟩), sp)
                        | five ← concat(sp)
                        | tot ← stack(one, two, three, four, five)
                        | return tot
```

Figure X. Original block code from the MtoATranslator file

However, in the current AutoCAD drawing files, multiple hydraulic accessories and pipes are drawn in a single plumbing layer. Therefore, it is more reasonable to create a function that immediately blocks the last object drawn, rather than creating superfluous new layers for each hydraulic components to block. As a result, the BlockingLast function was created, using the block function as a model, to block the most recently drawn hydraulic component rather than all objects found in the entire layer.

In the original block function, after an object is blocked, the object becomes a block, which disappears from the drawing. The block first disappears in reference to a base point, which can essentially be any point. But for relevancy, the origin point of the entrance tank was chosen for all the hydraulic components of the entrance tank. To insert the block back into the layer at the same position, the insertion point must be the same as the base point. Considering how only the first particular hydraulic component and accessory are blocked to later be copied, which uses a different position, the reference point inputs from the block function were unified as one origin point, reducing the number of inputs.

In overall, the blockinglast function blocks the last drawn object and inserts the block at the given origin.

```
Blocking(BlockName, Origin) := | one ← concat("-block", sp, BlockName)
                               | two ← concat(point(Origin⟨0⟩), sp, "all", sp)
                               | three ← concat("-insert", sp, BlockName)
                               | four ← concat(point(Origin⟨0⟩), sp)
                               | five ← concat(sp)
                               | total ← stack(one, two, three, four, five)
                               | return total
```

Figure XI. Blocking function for all objects

```
BlockingLast(BlockName, Origin) := | Naming ← concat("-block", sp, BlockName)
                                   | OriginPoint ← concat(point(Origin), sp, "1", sp)
                                   | InsertingBlock ← concat("-insert", sp, BlockName, sp)
                                   | InsertionPoint ← concat(point(Origin), sp)
                                   | enter ← concat(sp)
                                   | total ← stack(Naming, OriginPoint, InsertingBlock, InsertionPoint, enter)
                                   | return total
```

Figure XII. BlockingLast Function

To avoid avertible confusion in object selection for blocking, generally the new hydraulic component needs to be blocked immediately after it is drawn. Moreover, every time a new hydraulic component with a different nominal diameter is needed, a new block for that diameter needs to be created because the components can be inaccurately scaled linearly with diameter. Currently, the BlockingLast function, which should be moved to the MtoATranslator file, is defined in the Entrance Tank AutoCAD drawing file.

11/05/14

Attributing Data to Blocks

Blocks can be defined with attributes that are assigned using the $\text{Accessory}_{\text{Att}}$ and Pipe.Att functions. The attributes for hydraulic accessories and pipes include, but are not limited to, diameter, schedule specification, and length. These functions should be stacked right after BlockingLast in the drawing code. It is important to call these functions directly after they are blocked to assure that the correct object is blocked. Given the blockname,the Accessory.att function finds a specific accessory and creates an attribute called “Diameter,” with the tag, “diameter of the pipe.” Then, using the pipediameter input, which is the nominal diameter of the hydraulic component, the function assigns the provided pipediameter to the “Diameter” attribute. Additionally, the attributes that assign numbers must be attributed using the stringit functions that convert the numbers into strings.

$$\text{stringit}_{\text{inches}}(x) := \text{num2str}\left(\frac{x}{\text{in}}\right) \qquad \text{stringit}_{\text{round}}(x) := \text{num2str}\left(\text{round}\left(\frac{x}{\text{m}}, 2\right)\right)$$

Figure XIII. Block Number Insertion for Attributes

AutoCAD has the ability to count how many times a block with the same attributes was drawn. After the attribute is assigned to the block, the block disappears; therefore, the block needs to be reinserted. Pipe_{Att} is a separate block attributing function that essentially works the same way as $\text{Accessory}_{\text{Att}}$ to assign the nominal diameter and length attributes. As mentioned before, the hydraulic components need to be different blocks if they have different diameters.

```
Accessory_Att(PlantComponent, Diameter, Origin, BlockName, Specification) :=
    BlockEditor ← concat("-bedit", sp, BlockName, sp)
    PlantComponentTag ← concat("-attdef", sp, "i", sp, sp, "PlantComponent", sp, "Component of Plant", sp)
    PlantComponentValue ← concat(PlantComponent)
    DiameterTag ← concat("-attdef", sp, "i", sp, sp, "Diameter", sp, "Diameter of Accessory", sp)
    DiameterValue ← concat(stringit_inches(Diameter))
    InsertionPoint ← concat[(point(Origin)), sp, sp]
    SpecificationTag ← concat("-attdef", sp, "i", sp, sp, "Specification", sp, "Accessory Specification", sp)
    SpecificationValue ← concat(stringit_nounits(Specification))
    SaveAttributes ← concat("bsave", sp, "bclose", sp)
    SyncBlocks ← concat("attsync", sp, "s", sp, "1", sp, "y")
    total ← stack(BlockEditor, PlantComponentTag, PlantComponentValue, InsertionPoint, DiameterTag, DiameterValue, InsertionPoint, SpecificationTag, Sp
    return total
```

Figure XIV. Accessory Attributes Function

```
Pipe_Att(PlantComponent, PipeDiameter, Origin, BlockName, Length, Specification) :=
    BlockEditor ← concat("-bedit", sp, BlockName, sp)
    PlantComponentTag ← concat("-attdef", sp, "i", sp, sp, "PlantComponent", sp, "Component of Plant", sp)
    PlantComponentValue ← concat(PlantComponent)
    InsertionPoint ← concat[(point(Origin)), sp, sp]
    DiameterTag ← concat("-attdef", sp, "i", sp, sp, "Diameter", sp, "Diameter of the Pipe", sp)
    DiameterValue ← concat(stringit_inches(PipeDiameter))
    LenthTag ← concat("-attdef", sp, "i", sp, sp, "Length", sp, "Length of Pipe", sp)
    LengthValue ← concat(stringit_round(Length))
    SpecificationTag ← concat("-attdef", sp, "i", sp, sp, "Specification", sp, "Pipe Specification", sp)
    SpecificationValue ← concat(stringit_nounits(Specification))
    SaveAttributes ← concat("bsave", sp, "bclose", sp)
    SyncBlocks ← concat("attsync", sp, "s", sp, "1", sp, "y")
    total ← stack(BlockEditor, PlantComponentTag, PlantComponentValue, InsertionPoint, DiameterTag, DiameterValue, InsertionPoint, LenthTag, LengthValue,
    return total
```

Figure XV. Pipe Attributes Function

12/2/14

Copying Blocks

After the first object is blocked, the other multiples of the same object must be copied or inserted as blocks. Rather than inserting the blocks, which is different from inserting the first block, the copy method was selected because the insertion command opens a dialog box

that disrupts the command line script. Once the command line is interrupted, the rest of the AutoCAD script is lost. To prevent the disruption, the copy method is the better option.

After a block is created, it can be copied and inserted to make replicas. The copyblock function selects the block that was drawn most recently, copies, and inserts the copy into a new insertion point. In copyblock, the origin of the original block should be used as the reference, and the insertionpoint specifies the new coordinates of the block. Attributes are not modified in copyblock and the insertionpoint should not be the relative distance between the first block, but a complete coordinate.

All the while, AutoCAD keeps count of how many times each block is copied. When the data from the AutoCAD file is extracted, the total number of times the block was first drawn and copied can be compiled into a table.

```
copyblock(origin, insertionpoint) :=
    select ← concat("select", sp, "1", sp)
    copy ← concat("co", sp, point(origin), sp, point(insertionpoint))
    enter ← concat(sp)
    tot ← stack(select, copy, enter)
    return tot
```

Figure XVI. Copy Block Function

Trash Rack of Entrance Tank

Originally, the AutoCAD script draws two trash rack sheets using ArrayAll to redraw the first sheet at a relative position. However, the blockinglast function blocks the first sheet and the copyblock function draws the same block at a new insertion point. For the EtTrashRackScript to draw an AutoCAD drawing, the $EtTrashRack_{Block}$ function was included between ViewTop and Freeze, eliminating ArrayAll, to block the recently drawn sheet of trash rack. After the blocking and further attributing, the second sheet of trash rack is copied into a new position.

$$EtTrashRack_{Origin} := EtHopper_{Origin} + \begin{pmatrix} \frac{T_{EtHopperLedge}}{2} - \frac{T_{ConcreteMin}}{2} - \frac{D_{Rebar}}{2} \\ 2 \cdot D_{Rebar} \\ H_{EtHopper} - \frac{3}{2} \cdot D_{Rebar} \end{pmatrix}$$

$$CopyOrigin_{TrashRack} := EtHopper_{Origin} + \begin{pmatrix} \frac{T_{EtHopperLedge}}{2} - \frac{T_{ConcreteMin}}{2} - \frac{D_{Rebar}}{2} + T_{ConcreteMin} + 2 \cdot D_{Rebar} \\ 2 \cdot D_{Rebar} \\ H_{EtHopper} - \frac{3}{2} \cdot D_{Rebar} \end{pmatrix}$$

$$EtTrashRack_{Block} := Blocking("TrashRack", EtTrashRack_{Origin})$$

$$EtTrashRack_{Diameter} := 0$$

$$EtTrashRack_{Att} := Accessory_{Att}("Et", EtTrashRack_{Diameter}, EtTrashRack_{Origin}, "TrashRack", PS_{Default})$$

$$Copy_{TrashRack} := copyblock(EtTrashRack_{Origin}, CopyOrigin_{TrashRack})$$

$$Freeze := FreezeLayer("EtTrashrack")$$

$$EtTrashRackScript := stack(TRLayer, HorizCylinder, ViewRight, ArrayHorizCylinder, ViewTop, VertCylinder, ViewRight, ArrayVertCyl, ViewTop, EtTrashRack_{Block}, EtTrashRack_{Att}, Copy_{TrashRack}, Freeze$$

Figure XVII. Entrance Tank Trash Rack Script

Rapid Mix

In the rapid mix of the entrance tank, one elbow, one vertical pipe, and one horizontal pipe were blocked and attributed. The LFOM orifices and the release hole were disregarded, because the orifices will be drilled into the vertical pipe manually and the release hole has been excluded in the construction process according to Drew Hart.

The elbow, vertical pipe and horizontal pipe are blocked using BlockingLast and attributed as show below.

$$\text{RMElbowLayer} := \text{layer}_{new}(\text{"EtRmElbowPlu"}, \text{green})$$

$$\text{RMElbow} := \text{ElbowSUBF}\left[\text{RMElbow}_{Origin}, \begin{pmatrix} 0 \\ 90 \\ 0 \end{pmatrix} \text{deg}, \text{ND}_{RMPipe}, \text{PS}_{Default}, (\text{"EtCon"})\right]$$

$$\text{EtRmElbow}_{Block} := \text{BlockingLast}(\text{"RmElbow"}, \text{RMElbow}_{Origin})$$

$$\text{FreezeRMElbow} := \text{FreezeLayer}(\text{"EtRmElbowPlu"})$$

$$\text{EtRmElbow}_{Att} := \text{Accessory}_{Att}(\text{"Et"}, \text{ND}_{RMPipe}, \text{RMElbow}_{Origin}, \text{"RmElbow"}, \text{PS}_{Default})$$

Figure XVIII. Rapid Mix Elbow Block

$$\text{RMVertPipe} := \text{PipeSUBF}\left[\begin{pmatrix} \text{RMElbow}_{Origin_0} \\ \text{RMElbow}_{Origin_1} + \text{ElbowRadius}(\text{ND}_{RMPipe}) \\ \text{RMElbow}_{Origin_2} + \text{ElbowRadius}(\text{ND}_{RMPipe}) \end{pmatrix}, \text{L}_{RMPipeVert}, \begin{pmatrix} 0 \\ 90 \\ 0 \end{pmatrix} \text{deg}, \text{ND}_{RMPipe}, \text{PS}_{Default}, (\text{"EtCon"})\right]$$

$$\text{RMVertPipe}_{Origin} := \begin{pmatrix} \text{RMElbow}_{Origin_0} \\ \text{RMElbow}_{Origin_1} + \text{ElbowRadius}(\text{ND}_{RMPipe}) \\ \text{RMElbow}_{Origin_2} + \text{ElbowRadius}(\text{ND}_{RMPipe}) \end{pmatrix}$$

$$\text{RMHorizPipe} := \text{PipeF}\left[\text{RMElbow}_{Origin}, \text{L}_{RMPipeHoriz}, \begin{pmatrix} 90 \\ 0 \\ 0 \end{pmatrix} \text{deg}, \text{ND}_{RMPipe}, \text{PS}_{Default}\right]$$

$$\text{RMVertPipe}_{Block} := \text{BlockingLast}(\text{"RmVertPipe"}, \text{RMVertPipe}_{Origin})$$

$$\text{RMVertPipe}_{Att} := \text{Pipe}_{Att}(\text{"Et"}, \text{ND}_{RMPipe}, \text{RMVertPipe}_{Origin}, \text{"RmVertPipe"}, \text{L}_{RMPipeVert}, \text{PS}_{Default})$$

$$\text{RMHorizPipe}_{Block} := \text{BlockingLast}(\text{"RmHorizPipe"}, \text{EtHopper}_{Origin})$$

$$\text{RMHorizPipe}_{Att} := \text{Pipe}_{Att}(\text{"Et"}, \text{ND}_{RMPipe}, \text{RMElbow}_{Origin}, \text{"RmHorizPipe"}, \text{L}_{RMPipeHoriz}, \text{PS}_{Default})$$

Figure XIX. Rapid Mix Pipe Blocks

The block scripts were inserted immediately after the objects were drawn.

$$RMScript := stack(RMElbowLayer, RMElbow, EtRmElbow_{Block}, EtRmElbow_{Att}, FreezeRMElbow, RMPipeLayer, RMVertPipe, RMVertPipe_{Block}, RMVertPipe_{Att}, RMHorizPipe, RMHorizPipe_{Block}, RMHorizPipe_{Att}, FreezeRMPipe$$

Figure XX. Rapid Mix Script.

12/4/14

Hopper Drain Couplings of Entrance Tank

In the hopper drains, currently only the couplings are blocked and attributed. Originally, the hopper drains were drawn using EtDrainsRight and its mirror function, EtDrainsLeft. Nonetheless, after a thorough inspection of the scripts, the mirror EtDrainsLeft was rarely used. Furthermore, considering how the mirror function could pose problems in the blocking and copying process, the EtDrainsLeft was removed and the copy method was implemented.

$$ThawDrains := ThawMultiple\left(\begin{pmatrix} "EtCon" \\ "EtDrainPipesPlu" \\ "EtWeir" \\ "EtTrashrack" \\ "Cdc" \\ "CdcFittings" \\ "EtDrainStoppersPlu" \end{pmatrix}\right)$$

$$MirrorDrains := Mirror3dAll\left["YZ", \begin{pmatrix} Plant_{Origin_0} - \frac{1}{2} \cdot L_{Sed} \\ 0m \\ 0m \end{pmatrix}, "Y"\right]$$

$$FreezeDrains := FreezeMultiple\left(\begin{pmatrix} "EtCon" \\ "EtDrainPipesPlu" \\ "EtWeir" \\ "EtTrashrack" \\ "Cdc" \\ "CdcFittings" \\ "EtDrainStoppersPlu" \end{pmatrix}\right)$$

$$EtDrainsLeft := stack(ThawDrains, MirrorDrains, FreezeDrains)$$

Figure XXI. Erased EtDrainsLeft

In the copy method, after the first coupling is drawn, blocked, and attributed with a nominal diameter, every drain coupling is copied according to how the original EtDrainsRight decided the number of couplings. The number of drain couplings is determined by the flowrate and EtDrains fully recognizes the corresponding number.

```
EtDrainCouplingsScript := | Local ← stack(DrainCouplingLayer, DrainWeirCoupling)
                          | i ← 0
                          | CouplingOrigin ← [ -(L_EtHopper + T_EtHopperLedge)·i - S_Fitting - ConRadius(ND_EtDrain)
                          |                    (ConRadius(ND_EtDrain) + S_Fitting + L_EtHopperFront)   if EN_DoubleTrain = 0
                          |                    (ConRadius(ND_EtDrain) + S_Fitting + (1/2)·W_Et - T_ChannelWall + T_ConcreteMin)   otherwise
                          |                    -SocketDepth(ND_EtDrain) ]
                          | BlockScriptdraincc ← BlockingLast("DrainCoupling", CouplingOrigin)
                          | DrainCoupling_Att ← Accessory_Att("Et", ND_EtDrain, CouplingOrigin, "DrainCoupling", PS_Default)
                          | Local ← stack[Local, CouplingSUBF[CouplingOrigin + Et_Origin, (0 270) deg, ND_EtDrain, PS_Default, ("EtCon")], BlockScriptdraincc, DrainCoupling_Att]
                          | i ← 1
                          | while i ≤ N_EtFullHopper + |((AddHopper - 2))   if EN_DoubleTrain = 1
                          |                            |1   otherwise
                          |   | CopyingOrigin ← CouplingOrigin
                          |   | CouplingOrigin ← [ -(L_EtHopper + T_EtHopperLedge)·i - S_Fitting - ConRadius(ND_EtDrain)
                          |   |                    (ConRadius(ND_EtDrain) + S_Fitting + L_EtHopperFront)   if EN_DoubleTrain = 0
                          |   |                    (ConRadius(ND_EtDrain) + S_Fitting + (1/2)·W_Et - T_ChannelWall + T_ConcreteMin)   otherwise
                          |   |                    -SocketDepth(ND_EtDrain) ]
                          |   | drain ← copyblock(CopyingOrigin, CouplingOrigin)
                          |   | Local ← stack(Local, drain)
                          |   | i ← i + 1
                          | Local ← stack(Local, FreezeDrainCouplings)
                          | return Local
```

Figure XXII. Drain Coupling Script

After the DrainCoupings function is called in AutoCAD, and the “BCOUNT” command is called, AutoCAD recognises three Drain Couplings for a flowrate of 20L/s.

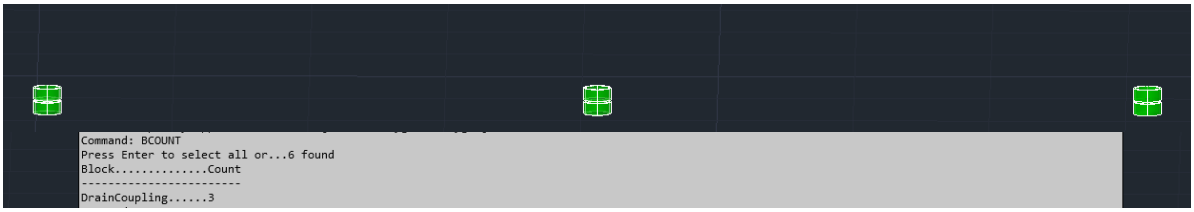

Figure XVII. Drain Coupling AutoCAD Drawing

Data Extraction

Once the AutoCAD draws with the blocks, an output file can be created from the AutoCAD file. When the output file is opened in an Excel file, all the data from the assigned attributes and the number of the blocks are presented in a table. Although, the data extraction was not coded into Mathcad, this method of extracting a materials list is more efficient than the method that is currently used by AguaClara members in Honduras.

Instructions for Data Extraction:

- Data extraction Command
- Create a new data extraction: name file or use an existing template
- Data source: Drawings/sheet set
  include current drawing
- Display: Display blocks only/Display blocks with attributes only
- Category filter: Attribute
- Output data to external file: Give name and location

This materials list can be created by the engineers in the field once the drawing contains all the attributes. It is important to always test the attributes when they are assigned using this tool.

Count	PLANTCOMPONENT	Name	DIAMETER	SPECIFICATION	LENGTH
1	Et	RmVertPipe	10	8	2.13
1	Et	RmHorizPipe	10	8	2.01
1	Et	TrashRack	0	0	
1	Et	RmElbow	10	8	

Table III. Data Extraction of Entrance Tank

Challenges Encountered

1/10/14

Unit Congruency in Matrices

The extraction and vectorization of hydraulic components required the values to have same units or be unitless. In one case, since the units utilized in the system are in metric measurements, the commercials lengths in feets had to be converted to meters. A different case arose in which the vectorized components had to be unitless in order to be combined into one whole matrix or to be utilized in a floor function.

11/7/14

Approach Problems

The design engine extraction program is incongruent to the materials list matrices. Therefore, a new approach is needed in which the design engine can access the materials list. Some considerations include solely extracting the components into variables without constructing matrices; making matrices with an excel output; inserting blocks into all the AutoCAD drawing functions and counting the blocks with command strings. Further suggestions were made, but until the best method becomes clear, the materials list challenge will focus on identifying the variables of all the hydraulic components and compiling them into matrices.

11/11/14

AutoCAD Block Counting

The challenge of AutoCAD Block Counting comprises of inserting block creating commands to all existing and future AutoCAD drawing functions. Essentially, the idea is to count all the blocks by inputting a command script in MathCAD to draw all the accessories and pipes as blocks and extract all the block count information.

11/24/14

Dynamic Blocks

Initially, the use dynamic blocks appeared useful because dynamic blocks could be stretched. When a dynamic block is stretched, the new dimensions are automatically replaces the assigned dimensions in the block attributes. However, a specific origin point, the stretching point, or displacement needs to be eventually cursor selected. Unfortunately, the team was unable to identify a way to select one point of the block using the command line only. Considering how the dynamic blocks also do not stretch for the z-axis, producing distorted 3D objects, the team decided to avoid utilizing dynamic blocks. Rather, the objects should be blocked every time different dimensions are required.

Part 3: Future Work

In the future, Design Team members should work towards compiling a comprehensive and user-friendly list of materials. This semester, to accomplish this goal, matrix extraction and blocking were developed, which can be considered as building blocks for a more extensive future materials list. By implementing and improving these commands, future designers should experiment and create blocks for all accessories and pipes in the MathCAD file or develop a new and more efficient method. Although the team decided to work with blocks, if, in the future, the matrix extraction is selected as the best method, comments should be written to instruct anyone who changes variables that go into the materials list to also change the variables in the excel sheet code to render the MathCAD work accurate and more user-friendly. If the team continues to work with blocks found in the MaterialsListEntranceTankBlocks file of the Under Development folder, variables with the specifications of each accessory should be created for the block attributes. Also, the variables in the blocking functions should be altered so that they are easier for other design team members to understand the blocking functions.

Although not included in the copy function, when blocks are copied, they can be scaled. The future design members should attempt to create a $\text{copyblock}_{\text{pipe}}$ function that scales the length of the pipe depending on the length of the pipe that is needed. Moreover, the Data Extraction Command should be utilized in AutoCAD to compose a materials list table. At the moment, the Data Extraction Command is manually utilized by the team members, but

the process should be automated. Finally, the team should consider reorganizing the MathCAD Drawing files so that each type of hydraulic component is stacked together. This will eliminate the need to re-draw and block hydraulic components when a different object is drawn in between the objects. In other words, all couplings with 2 inch nominal diameters should be drawn together in order for them to be copies of each other. If a pipe is stacked in between, then a new coupling with a 2 inch nominal diameter will need to be drawn, then stacked. In this situation, if one were to copy the most recently drawn object, it will copy the pipe rather than the coupling. Altogether, these goals will ultimately refine the materials list and facilitate the extraction process.