Towards a General Solution for Layout of Visual Goal Models with Actors: Supplemental Material

Yilin Lucy Wang
Smith College

Alicia M. Grubb
Smith College, amgrubb@smith.edu

Follow this and additional works at: https://scholarworks.smith.edu/csc_facpubs

Part of the Computer Sciences Commons

Recommended Citation

This Conference Proceeding has been accepted for inclusion in Computer Science: Faculty Publications by an authorized administrator of Smith ScholarWorks. For more information, please contact scholarworks@smith.edu
Towards a General Solution for Layout of Visual Goal Models with Actors: Supplemental Material

Yilin Lucy Wang, Alicia M. Grubb
Department of Computer Science
Smith College, Northampton, MA, USA
\{lwang, amgrubb\}@smith.edu

APPENDIX

This document is the supplemental information for our RE’20 paper, which may be cited as


This document contains an excerpt of the FLAAG algorithm presented in the paper, as well as the helper functions not presented in the paper.

Algorithm 1 Excerpt of FLAAG: Actor-based Goal Model Layout

Require:
- Goal Model $M = (A, G, R)$
- Constants $C_A, C_N, C_M$ – Constants for Actors, Nodes, and Moves.
- Maximum Layout Iterations maxItr

Ensure:
- Final Graph Layout Information

1: \(\text{actorSet}, \text{nodeSet} \leftarrow \text{INITIALIZATION}(M, \text{initLay})\)
2: \(\text{curCtr} = 0\)
3: while \(\text{CHECKCOND}(\text{curCtr}, \text{actorSet}, \text{nodeSet}, \text{maxItr})\) do
4:   for \(\text{node} \in \text{nodeSet}\) do
5:     for \(\text{actor} \in \text{actorSet}\) do
6:       if \(\text{typeof curActor} = \text{undefined}\) then
7:         curActor = \text{node.actorId}\)
8:       if \(\text{typeof curActor} = \text{undefined}\) then
9:         maxNXDict = 0
10:        if \(\text{curX} < 0\) then
11:          if \(\text{maxNXDict} \cdot \text{curActor} > \text{curX}\) then
12:            maxNXDict = curX
13:          if \(\text{curY} < 0\) then
14:            if \(\text{maxNXDict} \cdot \text{curActor} > \text{curY}\) then
15:              maxNYDict = curY
16:        for \(\text{node} \in \text{nodeSet}\) do
17:          if \(\text{curId} = \text{node.actorId}\) then
18:            node.nodeX = node.nodeX - maxNXDict \cdot curId
19:            node.nodeY = node.nodeY - maxNYDict \cdot curId

Next, Algo. 3 calculates the width and height of each actor by determining the differences between the largest and smallest values for the node coordinates that belong to the actor. Algo. 3 takes in the actor and the coordinates of the intentions in each of the actors. Using this information, Algo. 4 finds the final positions for the actors by first sorting the $x$ coordinate and then sort the $y$ coordinate of each actor. The arrangement of the actors is completed from the upper left to the bottom right, where subsequent actors are placed at the bottom right of the previous actor. Finally, Algo. 5 finds the final position for the nodes by adding the $x$ coordinate and $y$ coordinate of the actor, to which the node belongs, to the relative coordinates of each node.

Algorithm 2 \text{setCoordinatePositive} (\text{nodeSet})

1: function \text{setCoordinatePositive}(\text{nodeSet})
2:   maxNXDict \leftarrow \text{new dictionary}
3:   maxNYDict \leftarrow \text{new dictionary}
4:   for \text{node} \in \text{nodeSet} do
5:     curActor = \text{node.actorId}\)
6:     if \(\text{typeof curActor} = \text{undefined}\) then
7:       maxNXDict \cdot curActor = 0
8:     if \(\text{typeof curActor} = \text{undefined}\) then
9:       maxNYDict \cdot curActor = 0
10:    if \(\text{curX} < 0\) then
11:      if \(\text{maxNXDict} \cdot \text{curActor} > \text{curX}\) then
12:        maxNXDict \cdot \text{curActor} = \text{curX}
13:      if \(\text{curY} < 0\) then
14:        if \(\text{maxNYDict} \cdot \text{curActor} > \text{curY}\) then
15:          maxNYDict \cdot \text{curActor} = \text{curY}
16:   for \text{node} \in \text{nodeSet} do
17:     if \(\text{curId} = \text{node.actorId}\) then
18:       node.nodeX = node.nodeX - maxNXDict \cdot curId
19:       node.nodeY = node.nodeY - maxNYDict \cdot curId

Here we describe the helper functions listed on Lines 9–12 of Algo. 1. We use these helper functions after the relative positions are established for each actor and intention (i.e., nodes). Some of the relative coordinates that are generated by the force-directed algorithm are negative numbers. Since we calculate positions of intentions within actors from the upper left corner, Algo. 2 sets the relative coordinates to positive numbers. Algo. 2 takes in the nodeSet and assigns the coordinates of each node to positive numbers by adding the largest absolute value of the coordinates.

Algorithm 3 \texttt{GetSizeOfActor} Helper Function

1: function \texttt{GetSizeOfActor}(actorSet, nodeSet)
2: \hspace{1em} maxPXDict $\leftarrow$ new dictionary
3: \hspace{1em} maxPYDict $\leftarrow$ new dictionary
4: \hspace{1em} minPXDict $\leftarrow$ new dictionary
5: \hspace{1em} minPYDict $\leftarrow$ new dictionary
6: for node $\in$ nodeSet do
7: \hspace{1em} curX = node.nodeX
8: \hspace{1em} curY = node.nodeY
9: \hspace{1em} curActor = node.parent
10: if typeof maxPXDict.curActor = undefined then
11: \hspace{1em} maxPXDict.curActor = 150
12: if typeof maxPYDict.curActor = undefined then
13: \hspace{1em} maxPYDict.curActor = 100
14: if typeof minPXDict.curActor = undefined then
15: \hspace{1em} minPXDict.curActor = 150
16: if typeof minPYDict.curActor = undefined then
17: \hspace{1em} minPYDict.curActor = 100
18: if maxPXDict.curActor < curX then
19: \hspace{1em} maxPXDict.curActor = curX
20: if maxPYDict.curActor < curX then
21: \hspace{1em} maxPYDict.curActor = curX
22: if minPXDict.curActor > curX then
23: \hspace{1em} minPXDict.curActor = curX
24: if minPYDict.curActor > curX then
25: \hspace{1em} minPYDict.curActor = curX
26: for actor $\in$ actorSet do
27: \hspace{1em} actorId = actor.nodeId
28: if typeof maxPXDict.actorId = undefined then
29: \hspace{1em} maxPXDict.actorId = curX
30: if typeof maxPYDict.actorId = undefined then
31: \hspace{1em} maxPYDict.actorId = curX
32: if typeof minPXDict.actorId = undefined then
33: \hspace{1em} minPXDict.actorId = 0
34: if typeof minPYDict.actorId = undefined then
35: \hspace{1em} minPYDict.actorId = 0
36: \hspace{1em} x = maxPXDict.actorId - minPXDict.actorId + 300
37: \hspace{1em} x = maxPYDict.actorId - minPYDict.actorId + 200
38: \hspace{1em} actor.sizeX = x
39: \hspace{1em} actor.sizeY = y

Algorithm 4 \texttt{CalculateActorPosWithRec} Helper Function

1: function \texttt{CalculateActorPosWithRec}(actorSet)
2: \hspace{1em} actorsXSorted = sortActorX(actorSet)
3: \hspace{1em} actorsYSorted = sortActorY(actorSet)
4: \hspace{1em} curX = 0
5: \hspace{1em} curY = 0
6: for actor $\in$ actorsXSorted do
7: \hspace{1em} actor.nodeX = actor.nodeX + curX
8: \hspace{1em} curX += curNode.sizeX
9: for actor $\in$ actorsYSorted do
10: \hspace{1em} actor.nodeY = actor.nodeY + curY
11: \hspace{1em} curY += curNode.sizeY

Algorithm 5 \texttt{MoveNodesToAbsPos} Helper Function

1: function \texttt{MoveNodesToAbsPos}(actorSet, nodeSet)
2: \hspace{1em} actorId = node.parent
3: for node $\in$ nodeSet do
4: \hspace{1em} if actor.nodeId = actorId then
5: \hspace{2em} curX = node.nodeX
6: \hspace{2em} curY = node.nodeY
7: \hspace{2em} node.nodeX = curX + actor.nodeX + 150
8: \hspace{2em} node.nodeY = curY + actor.nodeY + 100