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

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

> *Yilin Lucy Wang and Alicia M. Grubb. Towards a General Solution for Layout of Visual Goal Models with Actors. In Proceedings of the IEEE 28th International Requirements Engineering Conference, 2020.*

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 $M = \langle A, G, R \rangle$
    Constants $C_A, C_N, C_M$        ▷ Constants for Actors, Nodes, and Moves.
    Maximum Layout Iterations maxItr                    ▷ Optional Timeout
    Initial Layout Information initLay ▷ Optional coord. for elements in $M$.
Ensure:
    Final Graph Layout Information

 1: (actorSet, nodeSet) ← INITIALIZATION($M$, initLay)
 2: curCtr = 0                                  ▷ Initializes iteration counter.
 3: while CHECKCOND(curCtr, actorSet, nodeSet, maxItr) do
 4:   for node ∈ nodeSet do
 5:     ADJUST(node, actorSet, nodeSet, False, $C_A, C_N, C_M$)
 6:   for actor ∈ actorSet do
 7:     ADJUST(actor, actorSet, nodeSet, True, $C_A, C_N, C_M$)
 8:   curCtr++
 9: SETCOORDINATEPOSITIVE(nodeSet)
10: GETSIZEOFACTOR(actorSet, nodeSet)
11: CALCULATEACTORPOSWITHREC(actorSet)
12: MOVENODESTOABSPOS(actorSet, nodeSet)
13: return (actorSet, nodeSet)
```

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.

Last updated August 10, 2020.

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 SETCOORDINATEPOSITIVE Helper Function

```
 1: function SETCOORDINATEPOSITIVE(nodeSet)
 2:   maxNXDict ← new dictionary
 3:   maxNYDict ← new dictionary
 4:   for node ∈ nodeSet do
 5:     curActor ← node.actorId
 6:     if typeof curActor = undefined then
 7:       maxNXDict.curActor = 0
 8:     if typeof curActor = undefined then
 9:       maxNYDict.curActor = 0
10:     if curX < 0 then
11:       if maxNXDict.curActor > curX then
12:         maxNXDict.curActor = curX
13:     if curY < 0 then
14:       if maxNYDict.curActor > curY then
15:         maxNYDict.curActor = curY
16:   for node ∈ nodeSet do
17:     curId = node.actorId
18:     node.nodeX = node.nodeX − maxNXDict.curId
19:     node.nodeY = node.nodeY − maxNYDict′curId
```

Algorithm 3 GETSIZEOFACTOR Helper Function

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

Algorithm 4 CALCULATEACTORPOSWITHREC Helper Function

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

Algorithm 5 MOVENODESTOABSPOS Helper Function

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