RSVanDerPloegLuaLineDrivenLayout, protocol *hook*.

```
on: shapes edges: edges

   | whxyExternalAddress childrenExternalAddress whgExternalAddress nshapes j addr nedges rootIndex inputDuration parseDuration layoutDuration outputDuration positioningDuration gbreadth gdepth |
   nshapes := shapes size.
   nedges := edges size.

   self vertically
       ifTrue: [
          gbreadth := self gapsPoint x.
          gdepth := self gapsPoint y ]
       ifFalse: [
          gbreadth := self gapsPoint y.
          gdepth := self gapsPoint x ].

   inputDuration := [
                    whxyExternalAddress := Float64TypedExternalAddress
                                              allocate: nshapes * 3.

                    whgExternalAddress := Float64TypedExternalAddress
                                            allocate: nshapes * 2.

                    childrenExternalAddress := Int32TypedExternalAddress
                                                  allocate:
                                                  nshapes + nedges.

                    rootIndex := nil.
                    j := 1.
                    shapes do: [ :shape |
                       | idx wdummy hdummy w h |
                       idx := shape propertyVanDerPloegLuaImplIndex.

                       shape == self rootShape ifTrue: [ rootIndex := idx ].

                       w := shape encompassingRectangle width.
                       h := shape encompassingRectangle height.

                       whxyExternalAddress
                          at: idx put: w;
                          at: nshapes + idx put: h;
                          at: nshapes * 2 + idx put: gbreadth.

                       self vertically
                          ifTrue: [
                             wdummy := w.
                             hdummy := gdepth ]
                          ifFalse: [
                             wdummy := gdepth.
                             hdummy := h ].

                       whgExternalAddress
                          at: idx put: wdummy;
                          at: nshapes + idx put: hdummy.

                       childrenExternalAddress
                          at: idx
                          put: shape propertyVanDerPloegLuaImplChildrenSize.

                       j := j + 1 ].

                    self assert:
                       (rootIndex isNotNil and: [
                           rootIndex between: 1 and: nshapes ]).

                    edges do: [ :aLine |
                       childrenExternalAddress
                          at: j
                          put: aLine to propertyVanDerPloegLuaImplIndex.

                       j := j + 1 ].

                    j := j - 1. "because `j` points to the next available memory location."
                    self assert: nshapes + nedges = j ] timeToRun.

   self liblua withOpenedLibsStateDo: [ :state |
      parseDuration := [
                       self liblua
                          assert: [ :ll |
                             ll
                                luaL: state
                                dostring:
                                   'tidy = require "non-layered-tidy-trees"' ];
                          on: state push: #tidy;
                          lua: state
                          get: -1
                          field: 'reifyflatchunks'
                          remove: true.

                       self liblua assert: [ :ll |
                          ll
                             lua_pcall: state
                             valueWithArguments: {
                                   nshapes.
                                   whxyExternalAddress.
                                   whgExternalAddress.
                                   childrenExternalAddress.
                                   rootIndex }
                          lets: #( r nodes ) ] ] timeToRun.


      layoutDuration := [
                         self liblua
                            on: state push: #tidy;
                            lua: state
                            get: -1
                            field: 'layout'
                            remove: true;
                            lua_newtable: state;
                            on: state push: #r;
                            lua: state set: -2 field: 'root';
                            on: state push: self vertically;
                            lua: state set: -2 field: 'vertically';
                            on: state push: true;
                            lua: state set: -2 field: 'centeredxy'.

                         self liblua assert: [ :ll |
                            ll lua_pcall: state nargs: 1 nresults: 0 ] ]
                            timeToRun.

      outputDuration := [
                         self liblua
                            on: state push: #tidy;
                            lua: state
                            get: -1
                            field: 'flat_xy_into'
                            remove: true;
                            on: state push: nshapes;
                            on: state push: #nodes;
                            on: state push: whxyExternalAddress.

                         self liblua assert: [ :ll |
                            ll lua_pcall: state nargs: 3 nresults: 0 ].

                         self liblua assert: [ :ll |
                            ll luaL: state dostring: 'tidy.free(r)' ].

                         addr := self liblua
                                    lua: state getglobal: 'nodes';
                                    lua_touserdata: state at: -1.

                         self liblua
                            lua: state pop: 1;
                            lua_pushnil: state;
                            lua: state setglobal: 'nodes'.

                         addr free ] timeToRun ].

   positioningDuration := [
                           shapes do: [ :shape |
                              | kk x y |
                              kk := shape propertyVanDerPloegLuaImplIndex.

                              x := whxyExternalAddress at: kk.
                              y := whxyExternalAddress at: nshapes + kk.

                              shape position: x @ y ";
                                 extent: shape extent * Float silverRatio" ] ]
                              timeToRun.

   whxyExternalAddress free.
   whgExternalAddress free.
   childrenExternalAddress free.

   ^ {
        inputDuration.
        parseDuration.
        layoutDuration.
        outputDuration.
        positioningDuration }
```