|  |
| --- |
| Circle Language Spec: Misc Diagram Topics |

## Diagram Metrics

**[Preliminary documentation]**

This is unfinished documentation of lower quality. It only has unfinished descriptions and loose ideas.

### Diagram Metrics

*This text was written in the context of efforts to program a prototype app 'Circle 3'.*

The diagrams of the new computer language are to be drawn out automatically. If they weren’t, this would decrease the usability of the language. These diagram metrics as they are called will be functionally described first not being burdened by any mathematical implications, and after that the math to do so will be designed.

There are several elements to the diagram metrics. Some of them are considered more essential than others. A new version of the computer language is currently being implemented and only the essential diagram metrics concepts will be implemented at first.

The concepts are put more or less in order of priority.

More essential:

- Shape distribution inside a parent

Inside circle, inside other types of shapes.

To keep it simple, lining them up might be adequate.

- Imaginary reference trees

How to keep that overviewable?

At first it will be kept simple. Just distribute the objects over the circle’s circumference and give each object its reference tree up until the point that it needs

- Inward reference lines

This happens in case of fixed logical residences, when the fixed logical residence is actually displayed in the diagram, or one of the qualifiers of it is displayed.

And it also happens in case of arbitrarily chosen qualified references.

< I do not have a clear picture of when imaginary references will appear and when not. >

- Lines going out of the diagram

If the fixed logical residence is out of sight, the highest imaginary reference will get another imaginary references on just outsight the diagram’s upmost parent shape and this imaginary reference will get a line with an open end at which the qualified name of the logical target object reference is displayed.

- Access marks

When to show them, when not to show them.

- Molecular ordering

Shapes and lines being driven into position by certain forces.

> Perhaps with an invisible springs algorithm you can animate everything to equilibrium and you might not have to worry about calculating the exact optimal position.

> Molecular ordering and object order unfortunately might contradict.

- Object order

Will position the sub-objects in a certain order.

- Connectors

Show all connectors all the time? Show them only when you try to connect? Show only the ones that impose access restrictions or things usually not accessible? Sort of summarize them? Towards what direction do they point? When you only show them if you try to connect, perhaps first show the aspect, then after you select the aspect, select the specific connector?

Some of these concepts are less important than others. You need to design the essential ones first and use them in the version currently implemented.

Trivial:

- Names

- Values

- Line types

- Shape types

- Relational rings

Less essential:

- System interface

How to draw it out, when to draw it out, what of it to draw out.

- Nothing

- Shape line types

- Double shape lines

- Line bundling

- Line bundle crossing

Other:

- Zooming navigation

### Brainstorm Text (from original Symbol documentation)

* Element combining
  + Containment
  + Linkage
  + Positioning

< Those molecular like pullings of line connections >

#### Symbol Placement and Proportion

#### Line Alignment

* 1: Line convergence is…
* 2: Dimensional convergence with only target convergance
* 3: Dimensional convergence with source convergance and target convergence
* 4: Disturbance of convergence:
  + Alignment disturbance
  + Angle disturbance
  + Solutions for disturbance
    - Total solution annuls convergence completely
    - Two metric rules create convergence again:
      * Lines go from center to center
      * Symbols are proportionate.
        + Rules:

To their encapsulation level

To the quantity of contained symbols

* + - * + When proportionality causes a deeper object to be smaller, then it works *with* the convergence. Then proportionality causes a deeper symbol to be bigger, it works against the convergence and can actually show divergence. <However lines still group themselves together>. It’s encapsulation level fighting against quantity of contained symbols.
  + Exact and approximate alignment
    - Simple example allows exact alignment. This situation occurs in many places in a diagram
    - More complex example shows inaxact alignment:
      * More than two children
      * Both source and target convergence
      * Multiple lines fight.
* Exaggerated convergence
  + Quanta of a distance unit
  + Varying distance unit over line segments
  + Removing dimensions from the line convergence
  + Varying the presence of dimensions over line segments.
  + Varying Quanta of distance unit between two crossings
    - This rule is a part annihilation of the exaggerated convergence.

I have to have a term for the piece of line in between two crossings: line segment.

Do I need to cover proportionality of symbols here. I could. I could upgrade this subject to symbol metrics in general.

**Sometimes symbol proportionality and placement is compromised for the sake of alignment.**

1: Line convergence is the metric phenomenon, that lines converge towards mutual targets. As they travel, they move closer and closer to one another.

2: When multiple borders are crossed. <a dimensional convergence onstaat >

3: In the previous example only the target symbols were part of an encapsulation structure (or the souce symbols, if you reverse the direction). When both sources and targets are part of an encapsulation structure, there is both groupage by source as well as targets. Actually there is groupage by any crossing and by eventual source and target symbols>

4: The symbols in the diagrams above were properly aligned to show the convergence of lines best. The symbols were lined up by line target. Not like this:

<P: symbols not aligned, lines crossing a lot>

That is one disturbance of the convergence. It causes lines to cross. It’s called order disturbance or cross disturbance. The lack of this disturbance is called source-target alignment. Another disturbance is the following:

<P: aligned more paralel. Convergence angle decreases.>

This disturbance causes the angle of the convergence structure to decrease. This disturbance is called angle disturbance. The below is the extreme example of that.

<P: aligned parallel. Lines melt together.>

To limit the disturbance of line convergence, Symbol should be order by target:

<P: original, neat alignment>

Which is the same as ordering the target symbols by source. Switch the order of either and you ge this effect:

<P: use rainbow colors to denote which order is reversed. Two pictures: in one the sources reversed, in the other the targets.>

The ideal situation would seem to be be if you could always line up symbols by source and target the and put them tegenover elkaar. This is when the row of target symbols and the row of source symbols make no angle and if these rows are perpendicular to the lines’ metric direction.

Exact alignment of sources and targets, however, totally annuls the alignment.

<P: Symbols straight lines, equally spaced>

Lines always go from the center of the source symbol to the center of the target symbol, though. This causes convergence.

<P: previous picture, but lines go to the center>

The reason that lines automatically converge is that they go from center to center. <more pictures explaining that metric fact. Zonneklaar aantonen waar het metrische effect door komt>

The symbol sizes should be more proportionate, though, dependant on their encapsulation level and the amount of symbols it contains.

<P: normal simple convergence picture: proportionate symbols, lines from center to center.>

Making symbols proportionate like that causes more convergence.

##### Old

<>

De binnenkomende lines worden geordend op waar ze vandaan komen en de uitgaande lines worden geordend op waar ze naar toe gaan. Naar welk object ze toegaan.

<Ik moet de uitleg beginnen met een simpele toepassing van line convergence. Ik moet het steeds moeilijker maken en uiteindelijk ermee eindigen dat je kan groeperen op elk mogenlijk criterium.>

<Eigenlijk worden de lines al automatische geordend op waar ze heen gaan en waar ze vandaan komen, met convergence wordt dat nog een zwaarder benadrukt. Ja… dat is zo. Dat is grappig ja. Line convergence is alleen maar een overdrijving van wat al gebeurt. Maybe it’s near a border crossing where convergence is exaggerated. Eigenlijk wordt de convergence niet geëxaggerate. Eigenlijk wordt het ge-deëxaggerate. Je converged ze waar ze eigenlijk divergeren. Convergen doen ze al… zie tekeningen. Als de lijn naar het ene symbol en naar het andere symbol geen grote hoek maken, of zelfs een hoek van 0° dan kan er niet of nauwelijks ‘natuurlijke’ convergence plaats vinden. Stiekum vergroot je de hoek.

Bij overdreven convergence moeten vaste afstanden aangehouden worden.

Worden er nog groeperingen afgestreept onderweg? Nee. Nooit. Er is altijd groepering op alle criteria. Ik bedoel entrances. Op exits wordt niet gegroepeerd, want het is altijd dezelfde die exit <bull, het gaat over meedere lines>. Er is altijd een groepering op entrances en target. Altijd primair op uiteindelijk target, secundair op de parent van de target, ternair op de grandparent van de target, totaan de eerste exit. Shit. Ik ben er nog niet uit.

Een ander voordeel van gedwongen convergence is dat tijdens de rit de afstanden tussen de lines meestal gelijk blijven. Je kunt wel na elke border de convergence steeds wat verder overdrijven.

Convergence gebeurt alleen goed natuurlijk als de children verder uit elkaar staan dan ze wijd zijn… en dat is zonde van de ruimte! Maar over het algemeen, oh en dit maakt alles uit: lines gaan niet naar een symbool toe, maar naar hun middelpunt… behalve als ze naar een child toegaan, want dan wijkt het ag: het gaat niet meer naar het middelpunt. En bij gedwongen convergence wordt er gedaan alsof de children op een rechte lijn liggen, loodrecht op de line direction, zodat de convergence niet over een hoek van 0° verdeeld is… Verder wordt bij gedwongen divergence nauwelijks varierende afstanden tussen lines gebruikt. Of eigenlijk varierende varierende afstanden. Hoe dichter je bij het target komt, hoe minder het varieert of tussen elke twee crossings gaat het minder varieren?

>

##### Old

<Picture showing line convergence>

Line convergence basically means that you group lines that have similar aspects. The usual aspects you’ll group them by is by what they enter or exit. You can also group lines by source and destination.

It’s usually near a border crossing where line convergence takes place.

Say multiple lines exit a parent. Around the border you can group them by where they’re going. Some lines will enter A, some will enter B, others will exit. On the border you group them.

You can also group lines just before the borders and just after. Say before the border you group them by where they came from and after the border you group them by where they’re going.

You can also express both where they came from and where they’re going with line convergence. So called multidimensional line convergence: groups of lines within groups of lines.

You can regroup borders or add more grouping using multidimensional line convergence.

In the analyses of very complex (usually poorly written, ‘spaghetti’-) programs, line convergence serves very well.

To accompany the grouping you can mention the grouping criterium. Or rather: mention with the grouping where the group of lines is going or came from.

Convergence takes place where lines enter or exit borders.

The grouping between borders can become more precise or more general. You might not care about the source of a line in a higher encapsulation level, so grouping can be generalized and possibly more focused on where the lines are going. Ofcouse, if before a crossing the grouping is about where the lines are going, after the crossing, the same grouping is about where the lines came from. Grouping can be about any crossing. On borders or around borders, regrouping can take place and grouping. Ofcourse any other criterium can be grouped by too, but you usually use crossings as the criteria for for line grouping.

A group of lines can be seen like a buslane for lines with certain sources, targets or any selection of its crossing actually.

Grouping lines by next entrance is quite handy in any case. It even evetually takes place anyway: they converge towards what they enter automatically, because they’re all ending up in the same point… at least in the same symbol.

You could also group lines by line type so that you separate object lines, type lines and interface lines.

< In line convergence, show that not only is convergence useful when you see the whole diagram, but also when you don’t see the full diagram, you can still distinct line targets and sources. >

<line convergence, or actually, the curvedness of lines also creates clarity where lines cross eachother. When drawing a diagram you should try to make as little lines as possible cross. When you curve lines you can make crossings of lines more haaks. That makes it clearer to see where each line is going >

< Line convergence is very useful in complicated situations where many things call many other things in a rather disoriented matter. *Spaghetti code* this is called sometimes. In Symbol, the actualy spaghetti becomes visible, because the lines look like spaghetti, and the more lines there are and the more they go though<> eaching, the more it looks like spaghetti >

#### Compromise between Symbol Placement and Line Alignment

<different name>

#### Other Metrics

* Access symbols
* Shape peels
* Line peels
* All other diagram elements
* Identifiers
* Text language
* ‘Specifiers’
* Custom language

#### Ambiguity Risks In Symbol Metrics

* Multiple shape peel types may suggest the existence of another symbol.
* Line convergence may suggest line peels
* Line peels may suggest line convergence
* Line peels may suggest the existance of multiple lines.
* Others

…

#### Notation Choice

< - Implicit calls notation >

<

There may be an automatic choice mechanism in the Symbol programming enviroment. This is a rather complex, thing though, but should be regarded part of the language explanation, not just part of the Symbol programming environment.

Maybe this isn’t an alternative notations subject

>

#### Brainstorm

Do explain that these are standard metrics and that they are not mandatory, obligatory, compulsory, required to obey.

The *start* of an identifier must be closest to the symbol it belongs to. Usually there isn’t enough room for the whole identifier to be closest to the symbol it belongs to.

<I also must find a way to place identifiers there where there’s room…>

#### Line convergence

* + If there are many lines then they can converge. They can be grouped by how many borders they will exit and by what they will enter.
  + The grouping can be multi dimensional <P>.
  + And grouping can take place at multiple borders
  + The grouping between borders can become more precise or more general
  + You can name the groups of lines by what they exit or enter. Sort of line a bus lane can be named by its destination.

-----

2004,

The grouping of lines by caller, callee, etcetera

JJ

### More Ideas

CL,

Suggestions Art, Science & Entertainment,

http://www.visualthesaurus.com/

It has diagram metrics / automatic positioning, like I want it in my own system.

JJ

-----

Diagram Metrics,

2009-04-08

You might want to abstractly express the complexity of an object or method, by making its symbol bigger.

'Size' of a called method could indicate its complexity / how much it does, but it is: what does the method itself do, what do the methods called from it do? But how deep should you go? through the full depth? Because that's how much it does, but sometimes you want to see how much complexity of your own code is below that, so you would only want to see the complexity of some of the modules expressed and ignore the complexities in deeper modules...

JJ

-----

Diagram Metrics,

2009-04-09

Dat is echt heel belangrijk voor Diagram Metrics, dat method size: gewoon zien aan de grootte, wat de hoofdzakelijke procedure is. Dat moet misschien dan ook esthetisch kunnen, maar in eerste instantie qua complexiteit.

JJ

-----

Diagram Metrics,

2009-04-29

Size of a symbol might be set by the user.

It is much easier to develop the new CL when size is determined by the programmer.

Yet, automatic diagram metrics is important,

but user-decided size is effective enough.

Except, that the programmer might not do it,

or the programmer spends too much time on manually setting diagram metrics.

Position, though, is also part manually determined by the programmer, when automatic diagram metrics gives multiple options to positions.

JJ