influenced by

- Cinema 4D
- TikZ, LaTeX
- CAD
- programming (e.g. `matplotlib`, `QPainter`)
- paradigms (cf. programming)

influenced by Cinema 4D

- intuituve to use
- few central managers
  - viewport
  - object tree (objects, generators, tags)
  - attribute manager

title

- programming: two assessments
  - quality of code (readability, maintainability, ...)
  - quality of product (no bugs, speed, usable, ...)
- software engineering priciples (SE-principles):
  - avoid code duplication
  - abstraction
  - stick to paradigms (procedural, functional, oo, structural, ...)
  - apply patterns

title

- vector graphics: single assessment
  - beauty of the result (colors, shapes, content, ...)
- fundamental idea:
  - transfer SE-principles to vector graphics

title

- avoid duplicates
- procedural, structural, **object oriented**
- non-destructive

title

- KISS
- user *understands* the application $\Rightarrow$ no unexpected behaviour
- very few entities with clear responsibility
  - object
  - viewport
  - object tree
  - attribute manager
  - tag

object

- has …
    - …a parent and children
    - …attributes
    - …coordinate system (relative to parent, aka. “local transformation”)
- does something
    - displays geometry
    - modifies it’s children
    - acts as group

objects: empty

- supply a new coordinate system
- supply user-attributes
- grouping
- attributes: name, coordinate system

objects: rectangle

- shows a rectangle
- attributes: like empty + width, height, corer radius, …
- similar: *ellipse*, *star*, *n-gon*, …
- can convert to *path*

objects: path

- shows any path (aka. spline)
- attributes: like empty + interpolation, is-closed, ...

objects: generators

object	adds	attributes (plus *empty*-attributes)
mirror	mirrored clone	-
cloner	many clones	count, shape, mode: (linear/circular/path)
boolean	result of boolean	mode: (and, or, xor, not, …)
instance	single clone	ref to source

- mirrored/cloned object: first children
- *boolean*: apply operator to first two (all?) children
- *cloner* and *boolean* make hide their children. Only result is visible.
- *instance*: no special children

object tree

- relationship between objects (parent-child) is crucial
- encourage use of well-designed object tree dialog
- drag-and-drop to set parent/children
- select objects
- manage tags

attribute manager

- each object has attributes (see above)
- almost every aspect of the scene file is an attribute
- display/edit attribute of selected objects
- tags have attributes, too

viewport

- WYSIWYG
- select objects
- manipulate transformation of object (translate, rotate, scale)

tags

- attached to any object, each tag knows its owner object.
- model features that do not fit into the object tree
  - effects: bend, distort, outline, ...
  - constrain attribute
  - style
  - script

scripting

- make attribute system availabe to scripting language
- python ?
- user can define new attributes in any object (user-attribute)
- script-tag can access attributes from owner and its children

templates

- user defines an empty
  - with some user-attributes
  - with script-tag
  - with some children that “do” something
- script-tag uses user-attributes to set attributes of the children
- *template-object*: like *instance*, but with free top-level attributes

features

- multi-selection attribute
  - usable attribute manager, though multiple objects are selected
  - display only intersection of attributes
  - display value only if it is the same
  - set values like +1, $*2$ smartly

Open Questions

- start from scratch or extend existing oss?
  - Inkscape?
- separate material and geometry?
  - uncommon in 2D, but common in 3D
- file format
  - XML, JSON, binary, ...