

Programming Environments for  
**Generative  
Design**

Guilherme  
Ferreira  
Instituto Superior  
Técnico



1. Problem

2. Survey

3. Solution

4. Conclusion

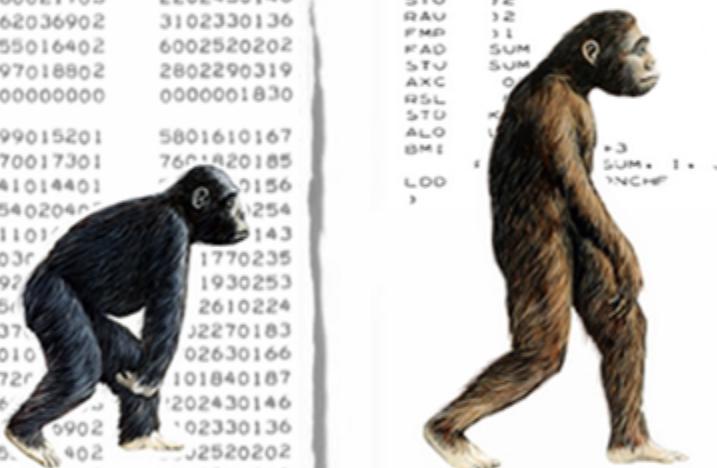
1

# Problem

# The Evolution of Computer Programming Languages

```
1999015201 5801610167
0170017301 7601820185
0141014401 5001530156
0154020402 0501630254
0211014201 4901720143
0203020602 1901770235
0192019902 2201930253
0256015101 5902610224
0137021301 6902270183
0201030401 5702630166
0272017501 8101840187
0160021703 2202430146
0162036902 3102330136
0155016402 6002520202
0197018802 2802290319
0000000000 0000001830

1999015201 5801610167
0170017301 7601820185
0141014401 500156
0154020402 0254
0211014201 143
0203020602 1770235
0192019902 1930253
0256015101 2610224
0137021301 J2270183
0201030401 02630166
0272017501 101840187
0160021703 2202430146
0162036902 02330136
0155016402 J2520202
0197018802 2802290319
0000000000 0000001830
```



Binary

```
STL -1
RAU 8005
MPY *
ALO -1
SLT 0004
ALO 8002
```

```
RAU 0115
STU >1
RAL 8006
STL -1
RAU 8007
MPY *
ALO -1
SLT 0004
ALO 8002
```

```
RAU 0096
STU >2
RAU >2
FMD >1
FAD SUM
STU SUM
AXC O
RSL STU
ALO K
U
BMI >3
SUM. I+J
INCHE
```

```
LDD >
```

```
C 0000 RECTANGULAR MATRIX
C 0000 MULTIPLICATION
DIMENSION A(4*5) *B(5*3)
READ I + A*B
READ I + N*M*L
7 DO 4 J= 1+N
1 DO 4 I= 1*M
6 SUM = 0+0
2 DO 3 K= 1+L
3 SUM=SUM+A(I,K)*B(K,J)
4 PUNCH I+SUM, I+J
8 END
```



Assembly

```
#include <stdio.h>
int main()
{
    printf("Hello world\n");
    return 0;
}
```



Fortran

```
#include <iostream>
int main()
{
    std::cout << "Hello World!";
    std::cout << endl;
    return 0;
}
```



C

```
/* HelloWorld.java
 */
public class HelloWorld
{
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```



C++

Java

```

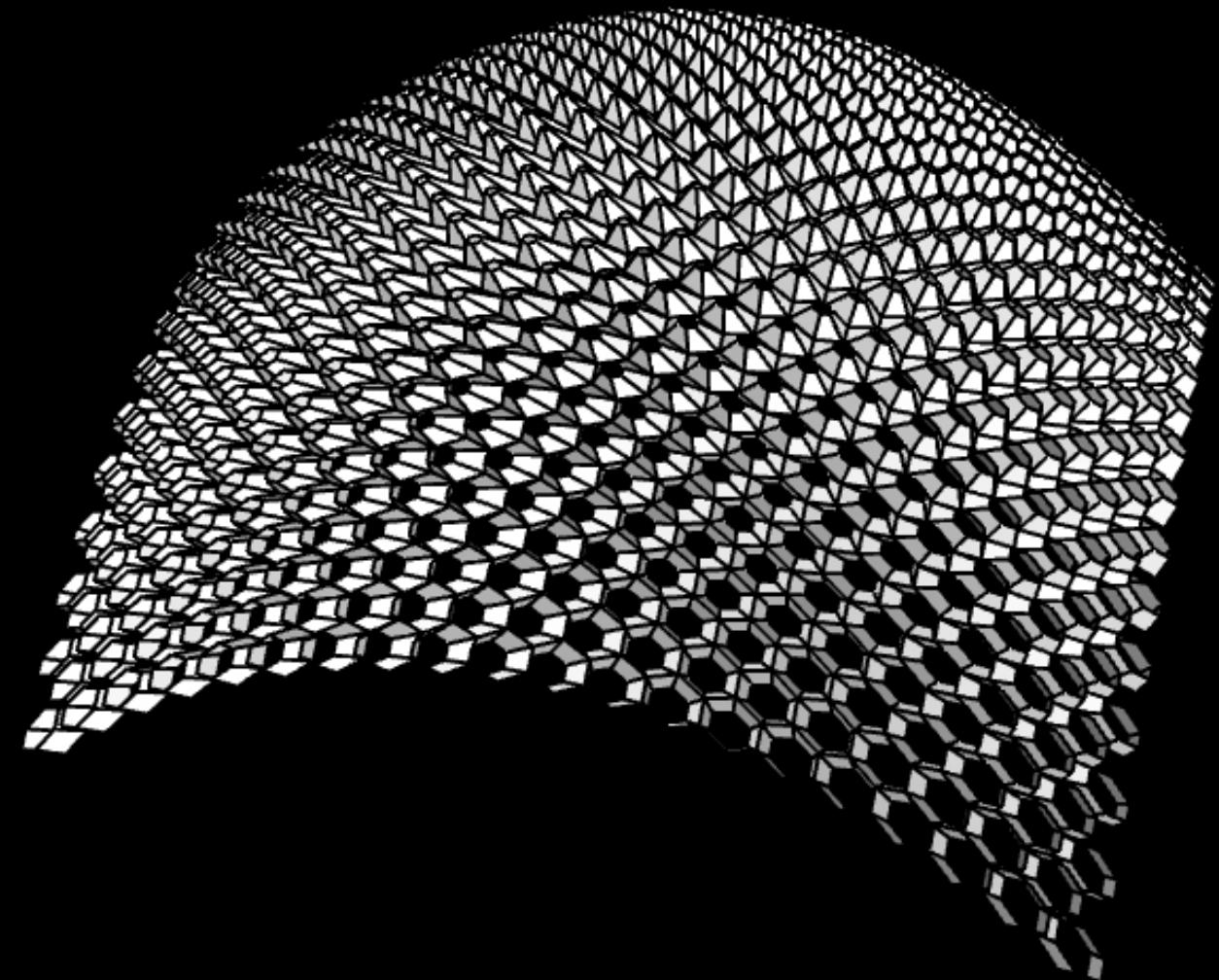
(define (skin-open-srf srf panel panel-f panel-h att)
  (sphere att 0.03)
  (iterate-quads
    (lambda (h0 h1 h2 h3)
      (panel (list-ref h0 4)
             (list-ref h0 3)
             (list-ref h1 4)
             (list-ref h2 1)
             (list-ref h2 0)
             (list-ref h3 1)
             panel-f panel-h att))
    (iterate-quads
      (lambda (p0 p1 p2 p3)
        (let* ((a (ptf p0 p3 0.17))
               (b (ptf p0 p1 1/2))
               (f (ptf p0 p3 1/2))
               (j (ptf p3 p2 1/2))
               (e (ptf b j 2/3))
               (k p1)
               (l p2)
               (c (ptf k l 0.17))
               (d (ptf k l 1/2))
               (w (ptf k l 2/3))
               (g (ptf (ptf p0 p3 2/3) (ptf p1 p2 2/3) 3/2))
               (h (ptf p3 p2 3/2))
               (i (ptf k l 1.17)))
          (panel a b c d e f panel-f panel-h att)
          (list a b c d e f)))
      (transpose-matrix srf)))))

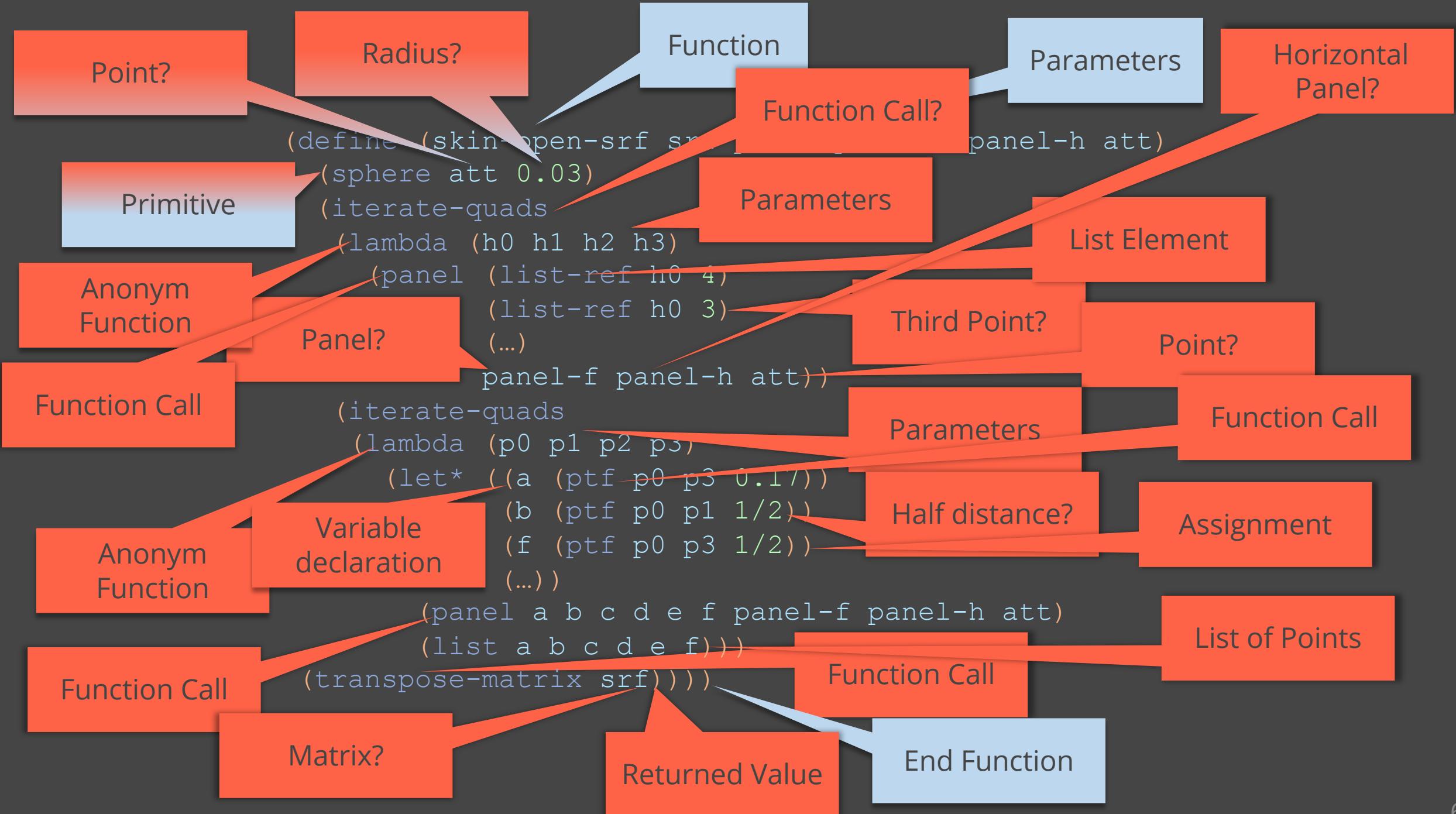
(define (hex-panel-1 p0 p1 p2 p3 p4 p5 f h att)
  ...)

(define (hex-panel-2 p0 p1 p2 p3 p4 p5 f h att)
  ...)

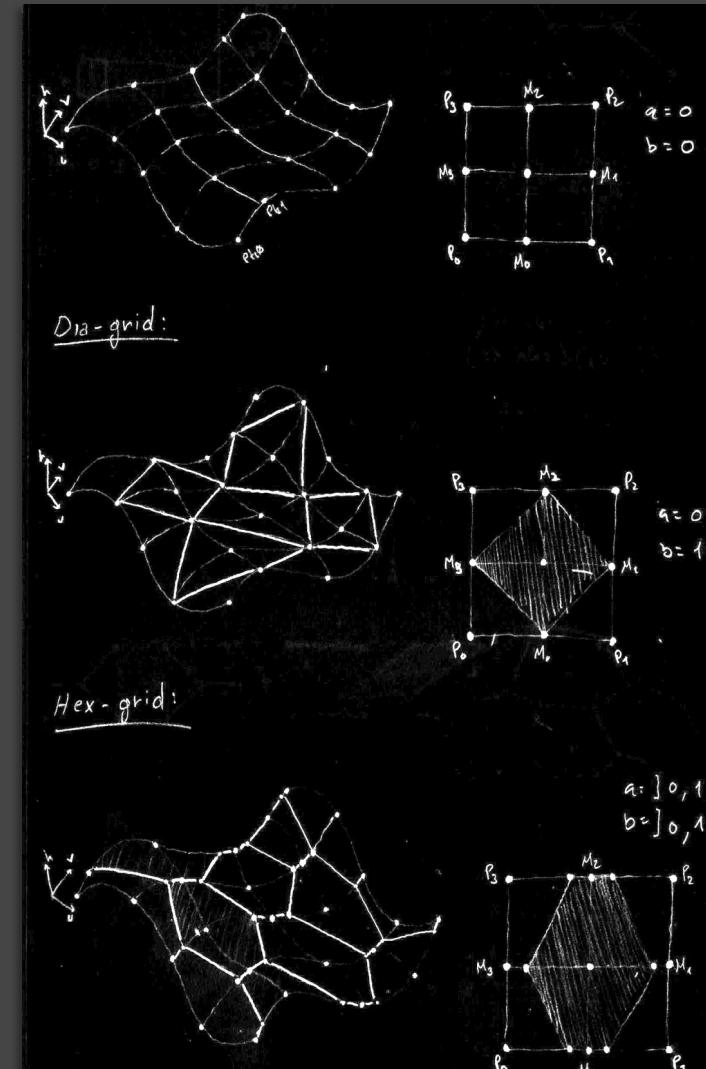
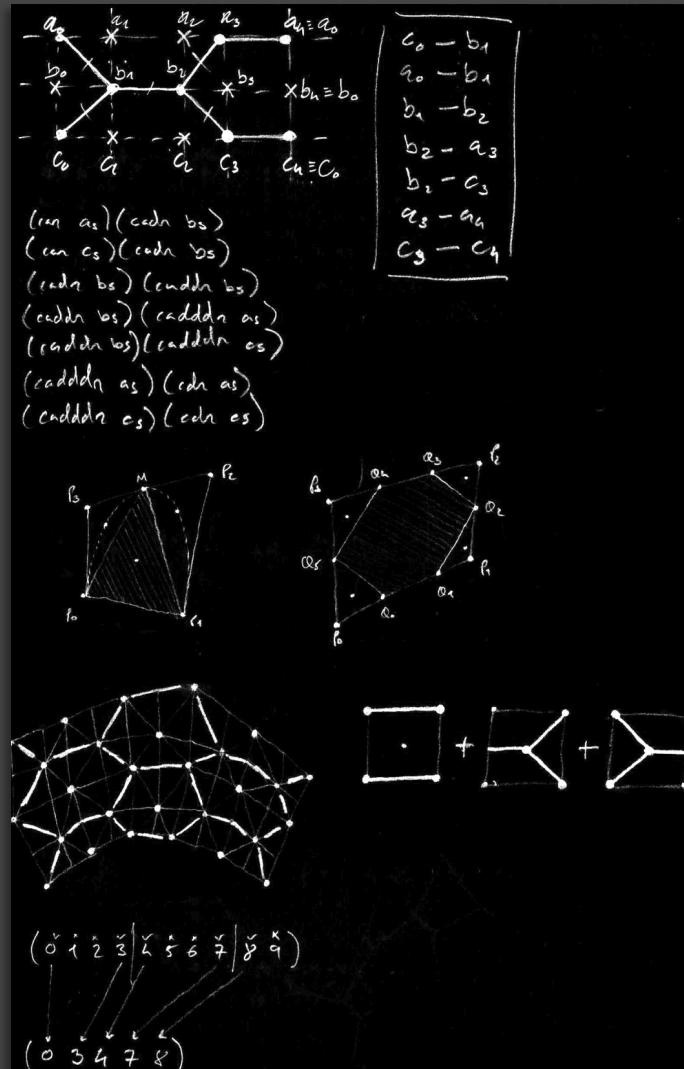
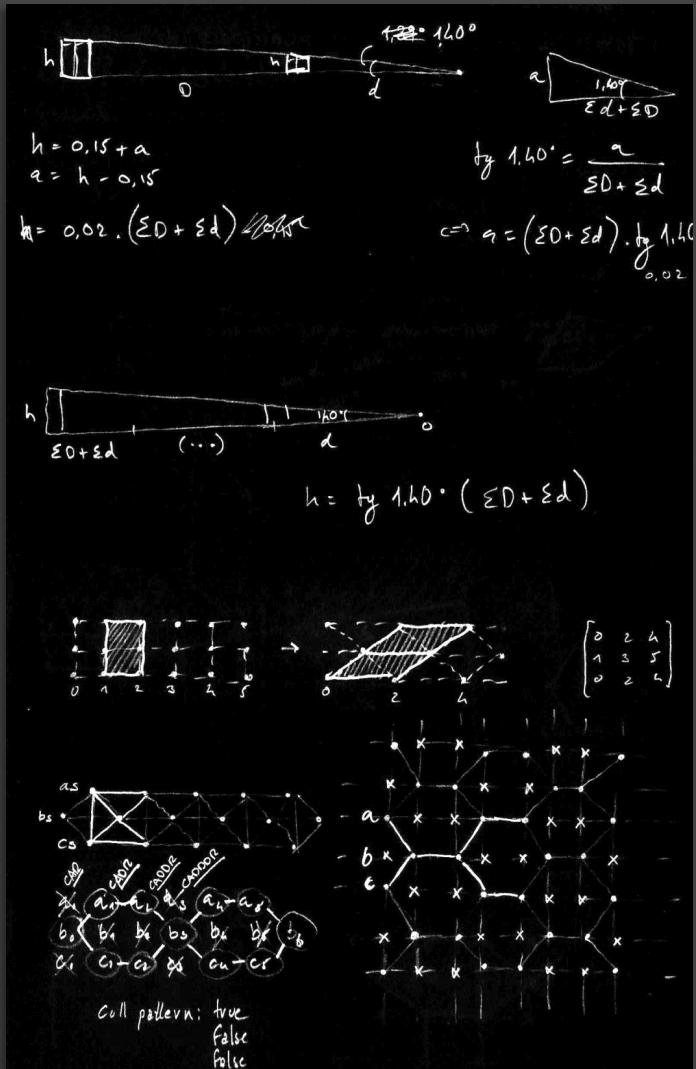
```

# Generative Design Environments

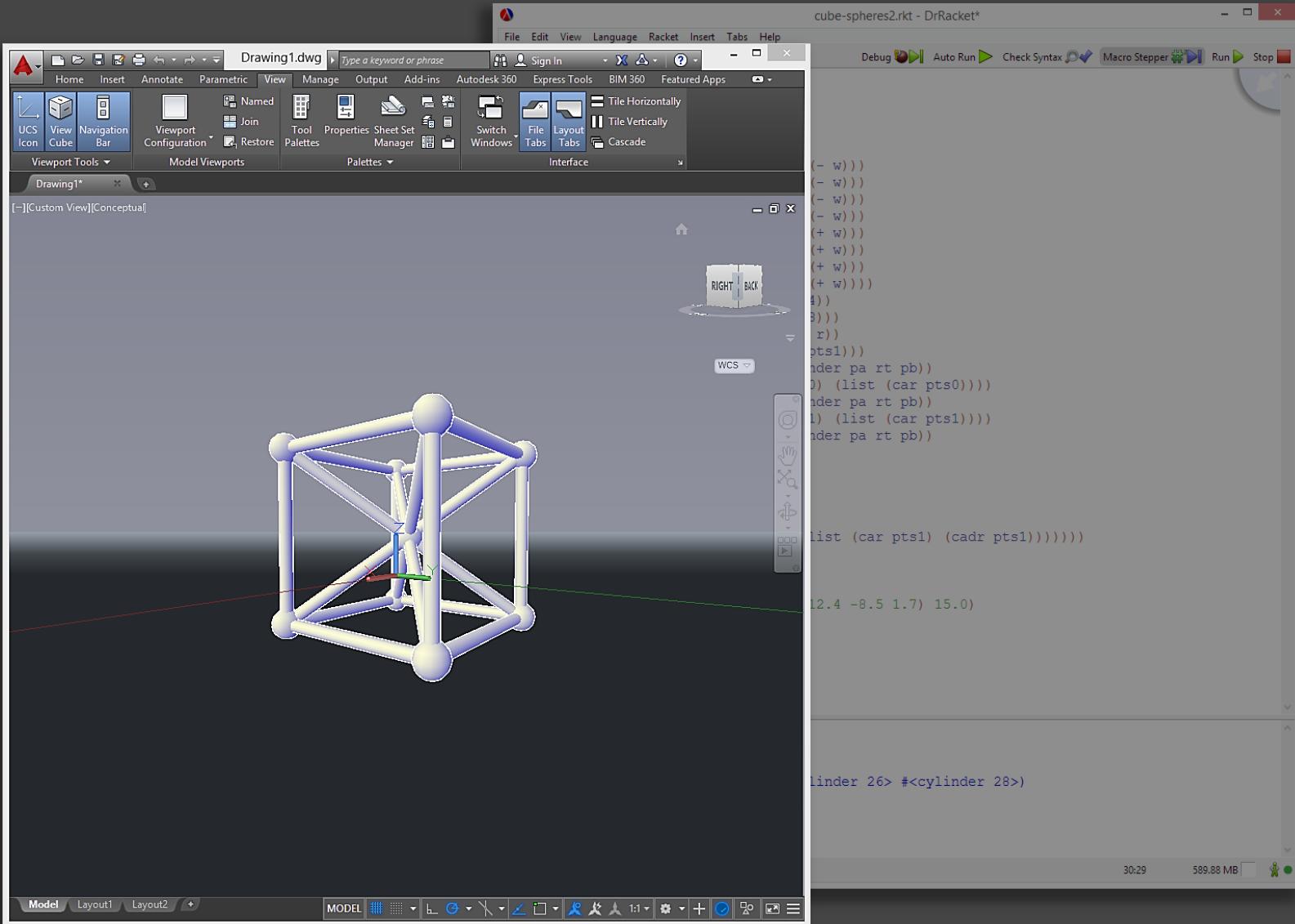




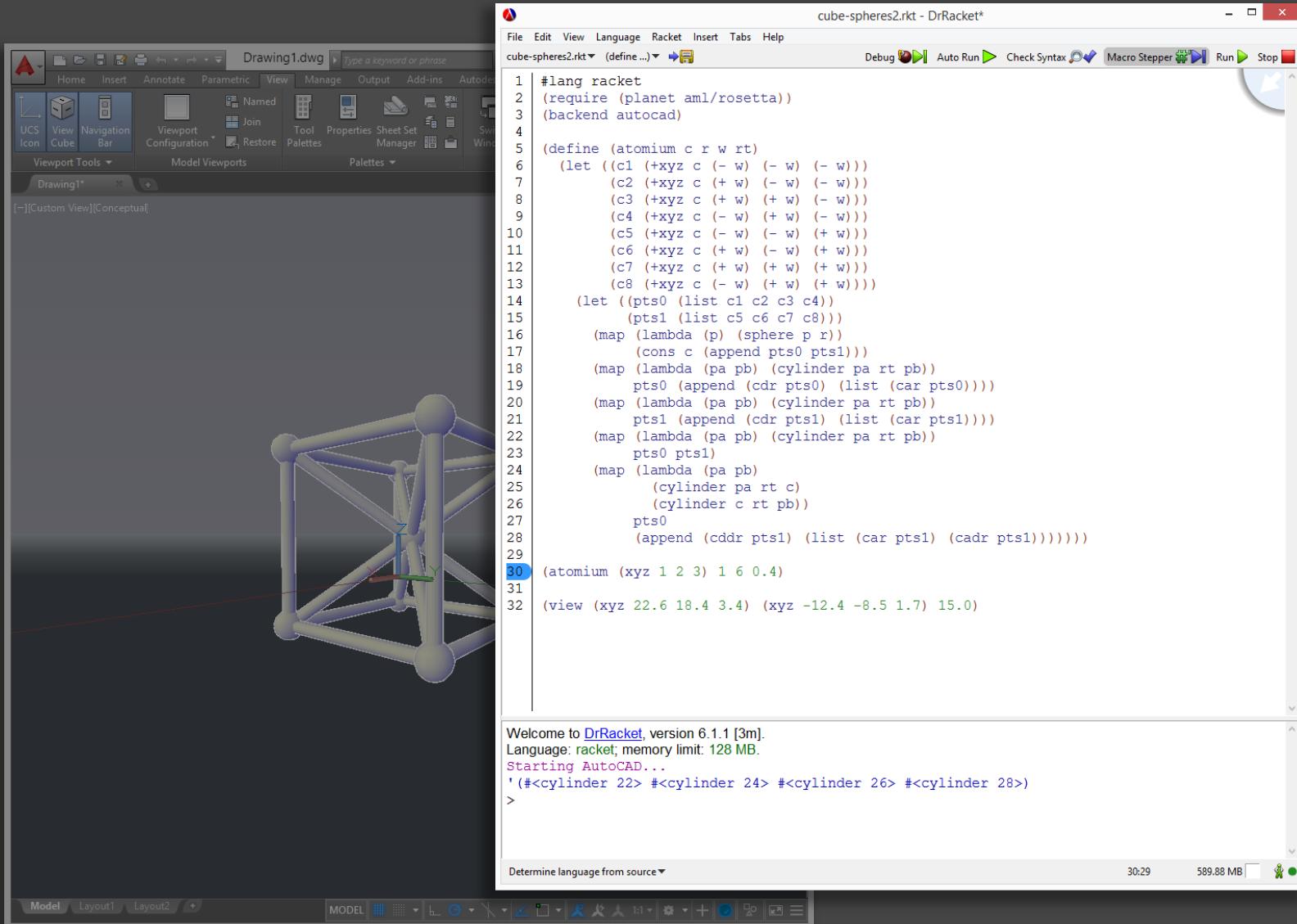
# People understand what they can see



# Creators need a medium connection



# Creators need a medium connection



2

# Survey

# LightTable (Chris Granger et al)

The screenshot shows the LightTable interface. On the left is a code editor window titled "cube.js\*" containing JavaScript code for a 3D cube. On the right is a browser window titled "three.js webgl - buffergeometry" showing a 3D scene with a textured cube.

```
//  
function animate() {  
    requestAnimationFrame( animate );  
    render();  
    stats.update();  
}  
function render() {  
    var time = Date.now() * 0.001;  
    mesh.rotation.x = time * 1.0;  
    mesh.rotation.y = time * 1.5;  
    renderer.render( scene, camera );  
}
```

(apply + [1 2 3 4])

Live

(apply + [1 2 3 4]) => 10

(defn hello-light [msg]  
 (str "Message: " msg))

(hello-light "hello")

(hello-light "hello") => "Message: hello"

```
(defn on-keypress [ch]  
  (swap! pressed-keys assoc (keyword (str ch)) true))  
  
(defn on-keyrelease [ch]  
  (swap! pressed-keys assoc (keyword (str ch)) nil))  
  
(defn pressed? [k]  
  (get @pressed-keys k))
```

keyword  
([name] [ns name])  
Returns a Keyword with the given  
namespace and name. Do not use : in the  
keyword strings, it will be added  
automatically.

# LightTable (Chris Granger et al)

The screenshot shows the LightTable interface. On the left is a code editor window titled "cube.js\*" containing JavaScript code for a 3D scene. On the right is a browser window titled "three.js webgl - buffergeometry" showing a 3D visualization of a textured cube.

```
//  
function animate() {  
    requestAnimationFrame( animate );  
    render();  
    stats.update();  
}  
function render() {  
    var time = Date.now() * 0.001;  
    mesh.rotation.x = time * 1.0;  
    mesh.rotation.y = time * 1.5;  
    renderer.render( scene, camera );  
}
```

The screenshot shows a live editor window with Clojure code. The code defines a function "hello-light" that takes a message and returns a string. It also defines a keypress handler "on-keypress" and a keyrelease handler "on-keyrelease".

```
(apply + [1 2 3 4])  
(defn hello-light [msg]  
  (str "Message: " msg))  
(hello-light "hello")  
  
Live  
(apply + [1 2 3 4]) => 10  
(defn hello-light ["hello"]  
  (str "Message: " "hello"))  
(hello-light "hello") => "Message: hello"
```

The screenshot shows a tooltip for the "keyword" function. The tooltip text is: "([name] [ns name]) Returns a Keyword with the given namespace and name. Do not use : in the keyword strings, it will be added automatically."

```
(defn on-keypress [ch]  
  (swap! pressed-keys assoc (keyword (str ch)) true))  
(defn on-keyrelease [ch]  
  (swap! pressed-keys assoc (keyword (str ch)) nil))  
(defn pressed? [k]  
  (get @pressed-keys k))
```

keyword  
([name] [ns name])  
Returns a Keyword with the given  
namespace and name. Do not use : in the  
keyword strings, it will be added  
automatically.

# LightTable (Chris Granger et al)

The screenshot shows the LightTable interface. On the left is a code editor window titled "cube.js\*" containing JavaScript code for a 3D cube. On the right is a browser window titled "three.js webgl - buffergeometry" showing a 3D scene with a textured cube.

```
//  
function animate() {  
    requestAnimationFrame( animate );  
    render();  
    stats.update();  
}  
function render() {  
    var time = Date.now() * 0.001;  
    mesh.rotation.x = time * 1.0;  
    mesh.rotation.y = time * 1.5;  
    renderer.render( scene, camera );  
}
```

(apply + [1 2 3 4])

Live

(apply + [1 2 3 4]) => 10

(defn hello-light [msg]  
 (str "Message: " msg))

(hello-light "hello")

(hello-light "hello") => "Message: hello"

```
(defn on-keypress [ch]  
  (swap! pressed-keys assoc (keyword (str ch)) true))  
  
(defn on-keyrelease [ch]  
  (swap! pressed-keys assoc (keyword (str ch)) nil))  
  
(defn pressed? [k]  
  (get @pressed-keys k))
```

keyword  
([name] [ns name])  
Returns a Keyword with the given  
namespace and name. Do not use : in the  
keyword strings, it will be added  
automatically.

# LightTable (Chris Granger et al)

The screenshot shows the LightTable interface. On the left is a code editor window titled "cube.js\*" containing JavaScript code for rendering a 3D cube. On the right is a browser window titled "three.js webgl - buffergeometry" showing a 3D scene with a textured cube.

```
//  
function animate() {  
    requestAnimationFrame( animate );  
    render();  
    stats.update();  
}  
function render() {  
    var time = Date.now() * 0.001;  
    mesh.rotation.x = time * 1.0;  
    mesh.rotation.y = time * 1.5;  
    renderer.render( scene, camera );  
}
```

The screenshot shows a Clojure REPL session in LightTable. The left pane shows the input code, and the right pane shows the live results of the evaluation.

(apply + [1 2 3 4])	Live	(apply + [1 2 3 4]) => 10
(defn hello-light [msg] (str "Message: " msg))		(defn hello-light ["hello"] (str "Message: " "hello"))
(hello-light "hello")		(hello-light "hello") => "Message: hello"

The screenshot shows a Clojure function definition and its documentation. The function handles key presses and releases, and checks if a key is pressed.

```
(defn on-keypress [ch]  
  (swap! pressed-keys assoc (keyword (str ch)) true))  
  
(defn on-keyrelease [ch]  
  (swap! pressed-keys assoc (keyword (str ch)) nil))  
  
(defn pressed? [k]  
  (get @pressed-keys k))
```

keyword  
([name] [ns name])  
Returns a Keyword with the given namespace and name. Do not use : in the keyword strings, it will be added automatically.

Rosetta (António Leitão et al)

The screenshot shows the DrRacket IDE interface with the title bar "p.rkt - DrRacket". The menu bar includes File, Edit, View, Language, Racket, Insert, Tabs, Help, and a toolbar with Check Syntax, Debug, Macro Stepper, Run, and Stop buttons. The code editor contains the following Racket code:

```
#lang racket

(require rosetta)

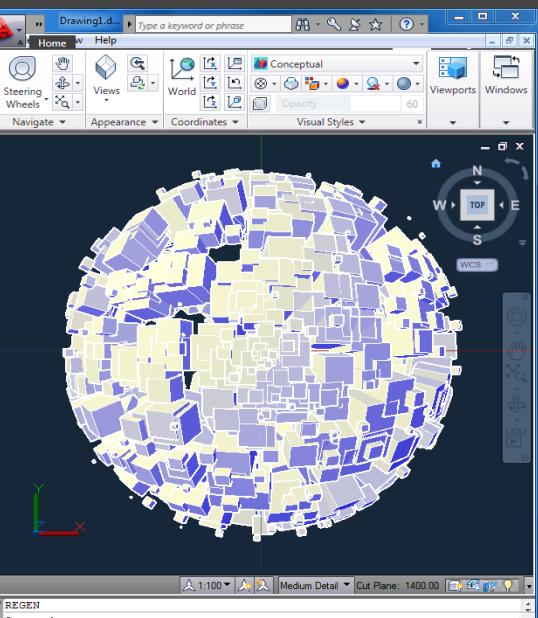
(backend "autocad")

(define (randomize-sshape d1 d2 sr shape-fn)
  (let* ((r (random-interval d1 d2))
         (th (random-interval 0 (* pi 2)))
         (fi (random-interval 0 pi))
         (p (sph r th fi)))
    (move p (shape-fn (- sr r) p)))

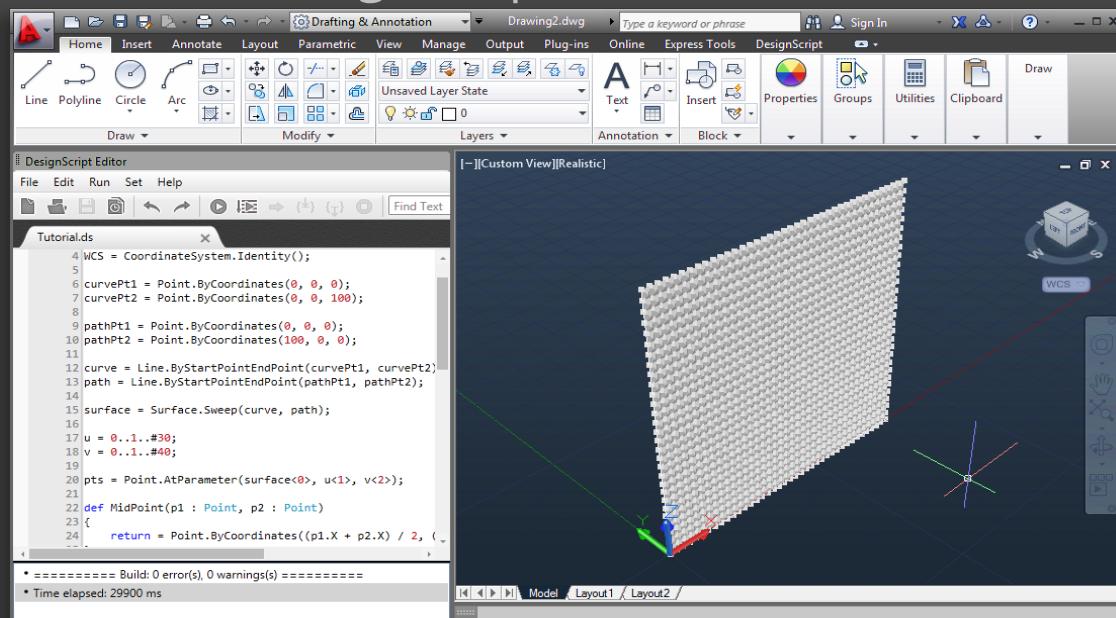
(define (shape-cloud d1 d2 r n shape-fn)
  (for/list ((i (range n)))
    (randomize-sshape d1 d2 r shape-fn)))

(shape-cloud
 4
 5
 5
 600
  (λ (r n) (box-normal r r r n)))

#<autocad-shape>
#<autocad-shape>
#<autocad-shape>
#<autocad-shape>
#<autocad-shape>
```

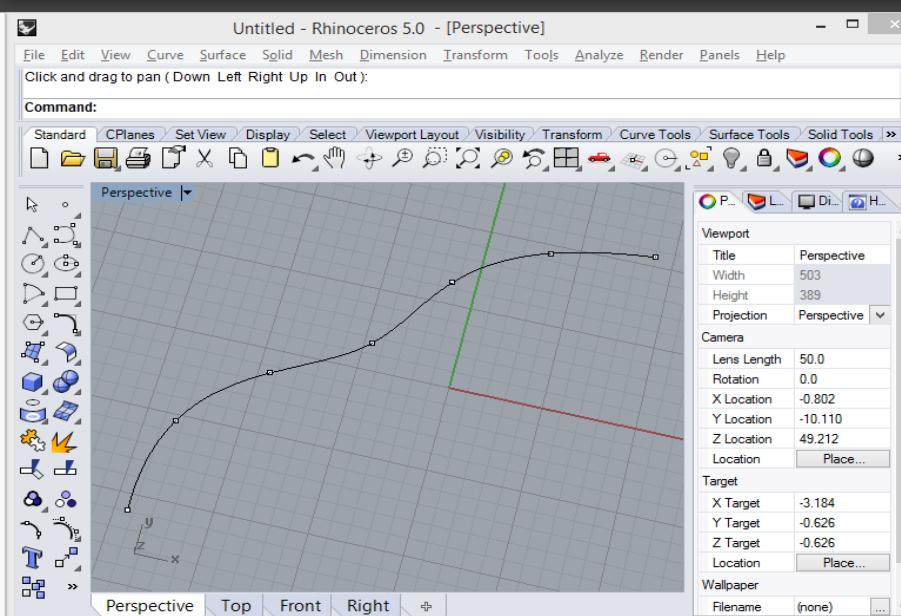


# DesignScript (Robert Aish et al)



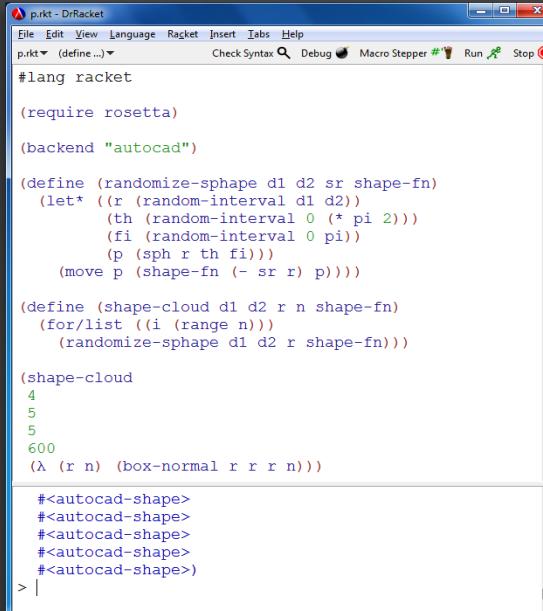
The screenshot shows the RhinoScript Editor interface. The top menu bar includes File, Edit, View, Source, Tools, and Help. Below the menu is a toolbar with icons for search, copy, paste, and others. A search bar labeled <Search> is positioned above the class browser. The class browser on the left lists various Rhino method categories under the 'Rhino' root node. The main window displays a script titled '(untitled)\*' containing the following VBA code:

```
1 Option Explicit
2 'Script written by <insert name>
3 'Script copyrighted by <insert company name>
4 'Script version 16 de junho de 2015 20:32:26
5
6 Call Main()
7 Sub Main()
8
9     Dim strLine
10    Dim divCurve
11
12    strLine = Rhino.GetObject("Select line")
13
14    divCurve = Rhino.DivideCurve(strLine, 6)
15
16    Rhino.AddPoints divCurve
17
18 End Sub
```

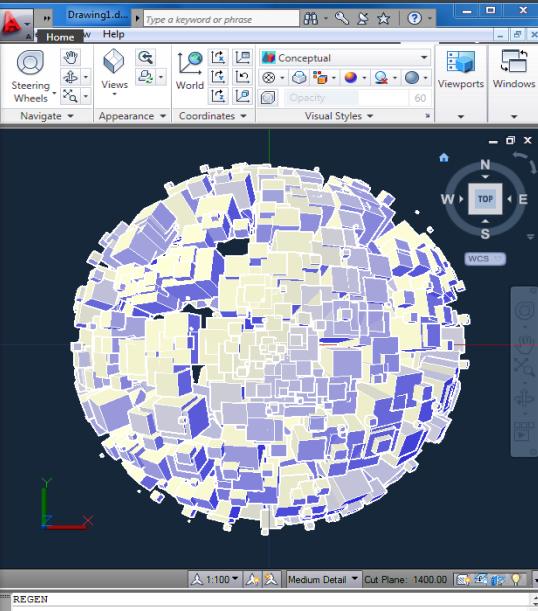


Monkey (David Rutten et al)

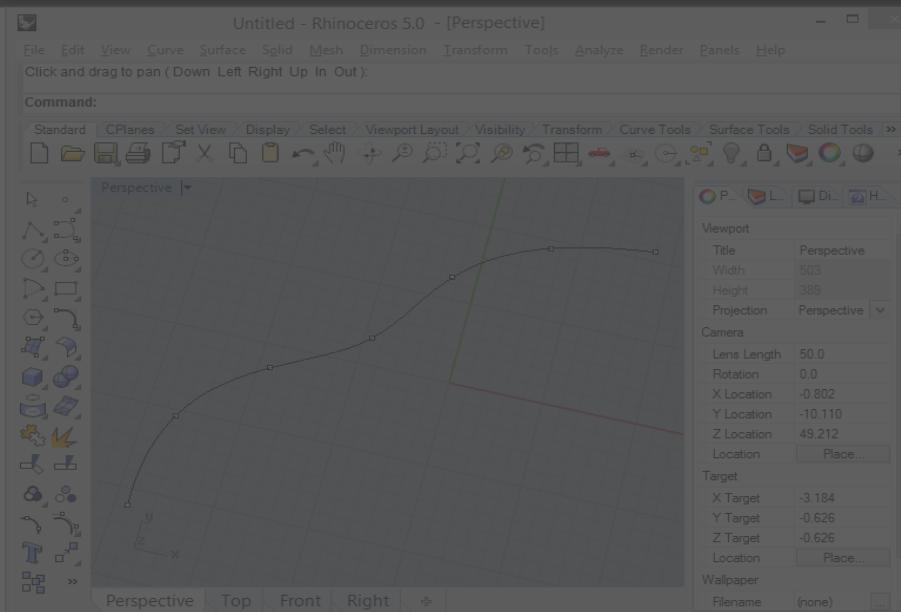
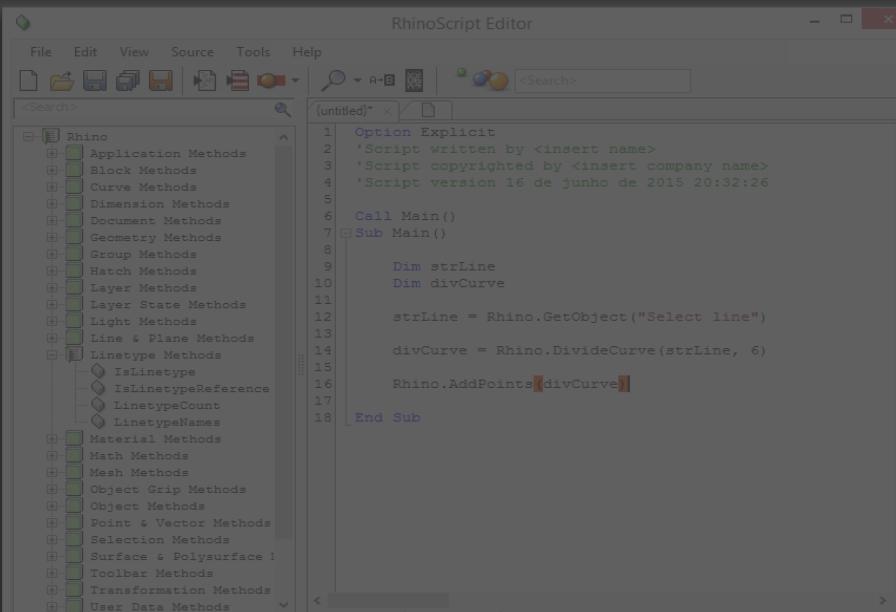
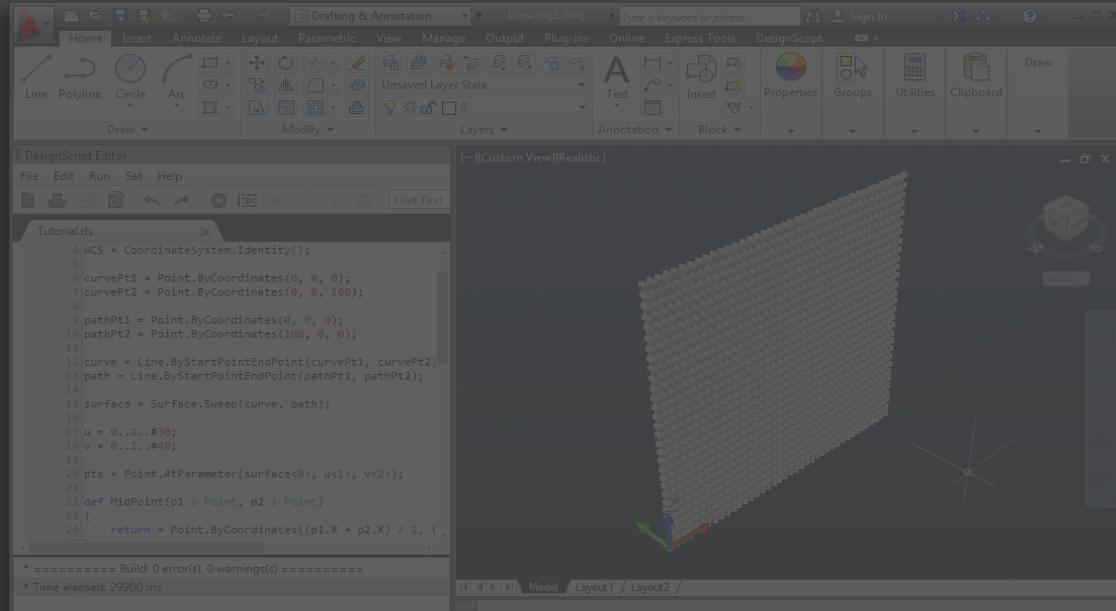
# Rosetta (António Leitão et al)



```
#lang racket
(require rosetta)
(backend "autocad")
(define (randomize-sshape d1 d2 sr shape-fn)
  (let* ((r (random-interval d1 d2))
         (th (random-interval 0 (* pi 2)))
         (fi (random-interval 0 pi))
         (p (sph r th fi)))
    (move p (shape-fn (- sr r) p))))
(define (shape-cloud d1 d2 r n shape-fn)
  (for/list ((i (range n)))
    (randomize-sshape d1 d2 r shape-fn)))
(shape-cloud
  4
  5
  5
  600
  (λ (r n) (box-normal r r r n)))
#<autocad-shape>
#<autocad-shape>
#<autocad-shape>
#<autocad-shape>
#<autocad-shape>
#<autocad-shape>
> |
```

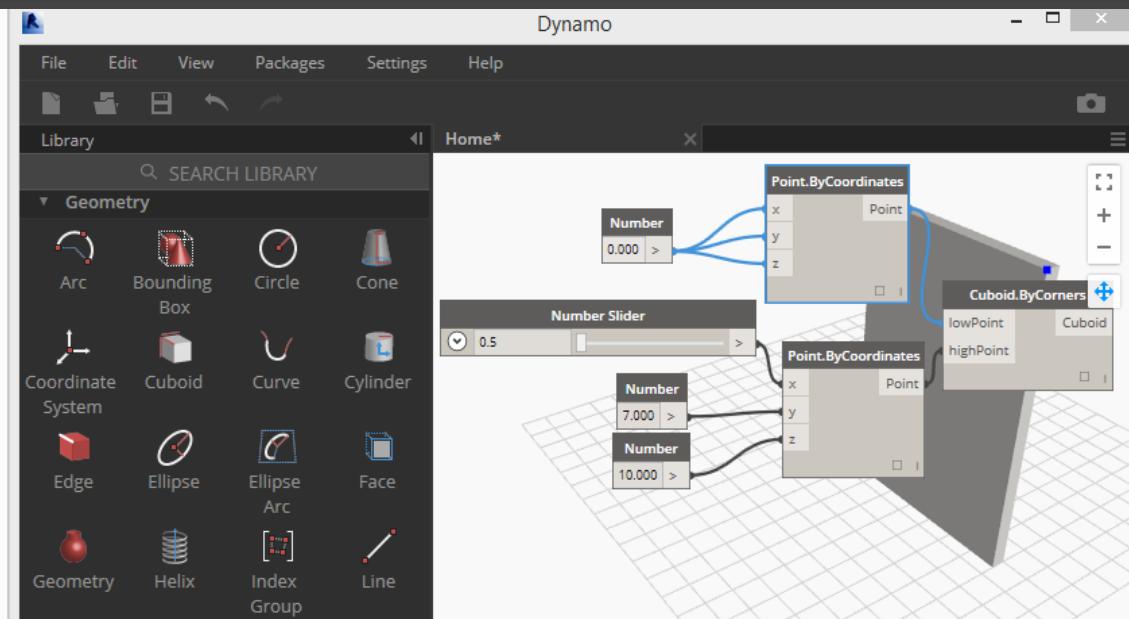
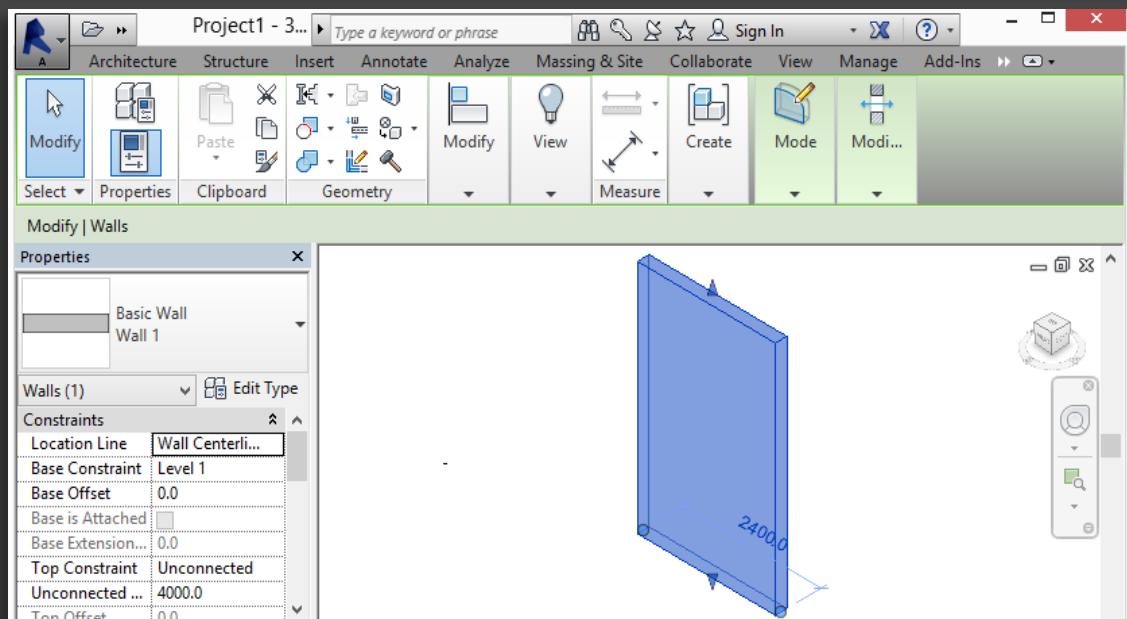
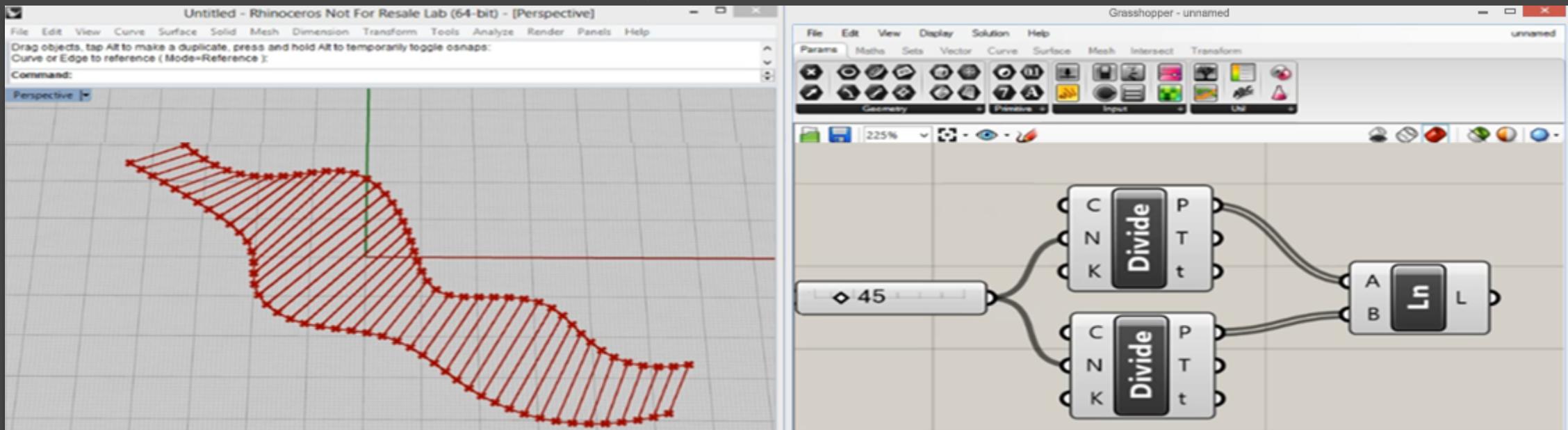


# DesignScript (Robert Aish et al)



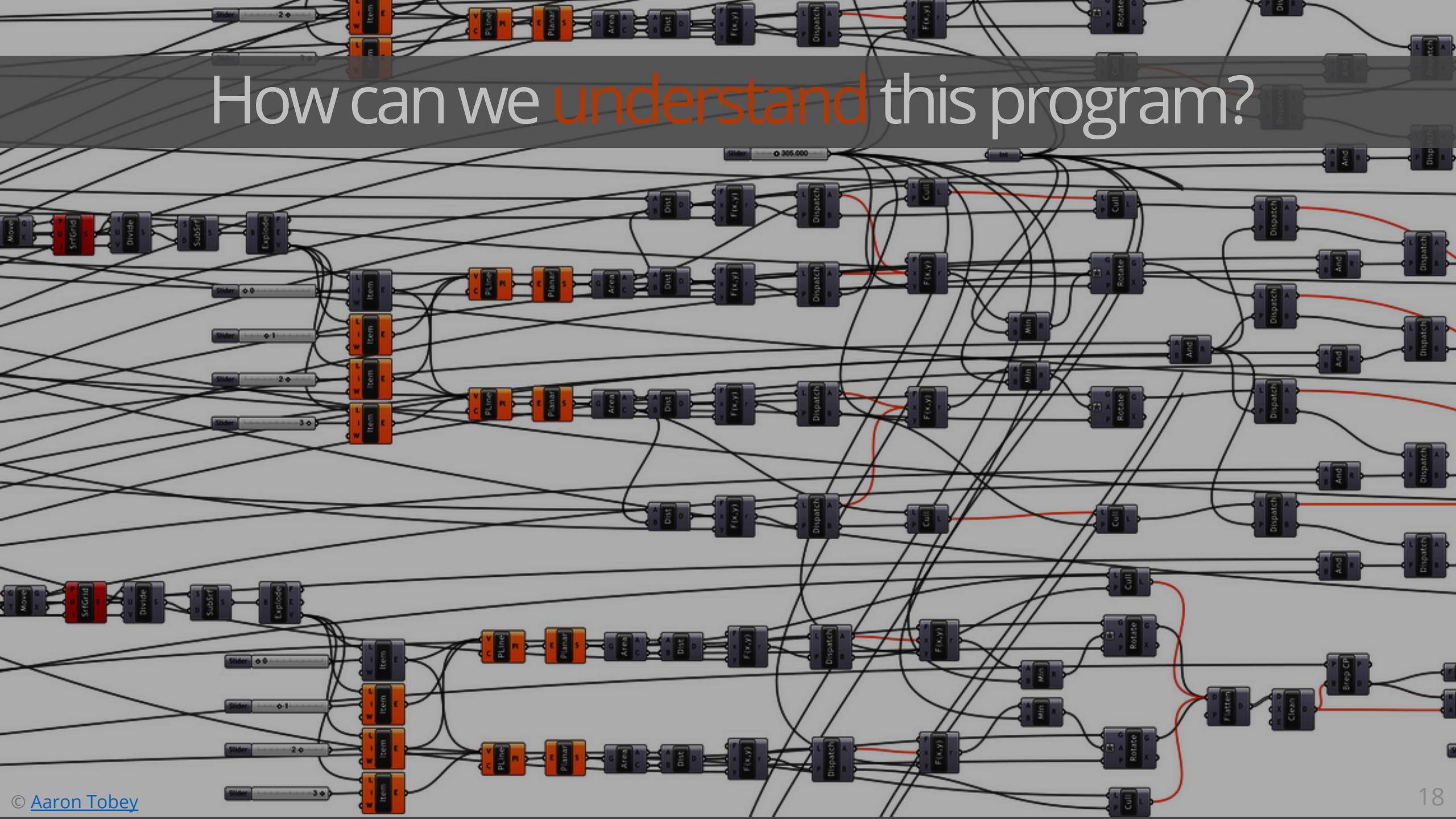
# Monkey (David Rutten et al)

# Grasshopper (David Rutten et al)



Dynamo (AutoDesk)

# How can we understand this program?



3

# Solution

# People understand what they can see

$$h = 0,15 + a$$

$$a = h - 0,15$$

$$h = 0,02 \cdot (\Sigma D + \Sigma d)$$
  

$$h = \tan 140^\circ \cdot (\Sigma D + \Sigma d)$$
  

$$\begin{bmatrix} 0 & 2 & h \\ 1 & 3 & 5 \\ 0 & 2 & 4 \end{bmatrix}$$
  

Coll pattern: true  
false  
false

$$\begin{cases} a_0 - b_1 \\ a_0 - b_1 \\ b_1 - b_2 \\ b_2 - a_3 \\ b_2 - c_3 \\ a_3 - a_4 \\ c_3 - c_4 \end{cases}$$
  

$$(can\ as)(cadn\ bs)$$

$$(can\ cs)(cadn\ vs)$$

$$(cadn\ bs)(cadn\ bs)$$

$$(cadn\ vs)(cadn\ as)$$

$$(cadn\ vs)(cadn\ cs)$$

$$(cadn\ as)(cadn\ as)$$

$$(cadn\ cs)(cadn\ cs)$$
  
  

$$(0 \ 1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8 \ 9)$$

$$a = 0$$

$$b = 0$$
  

$$a = 0$$

$$b = 1$$
  

Dia-grid:

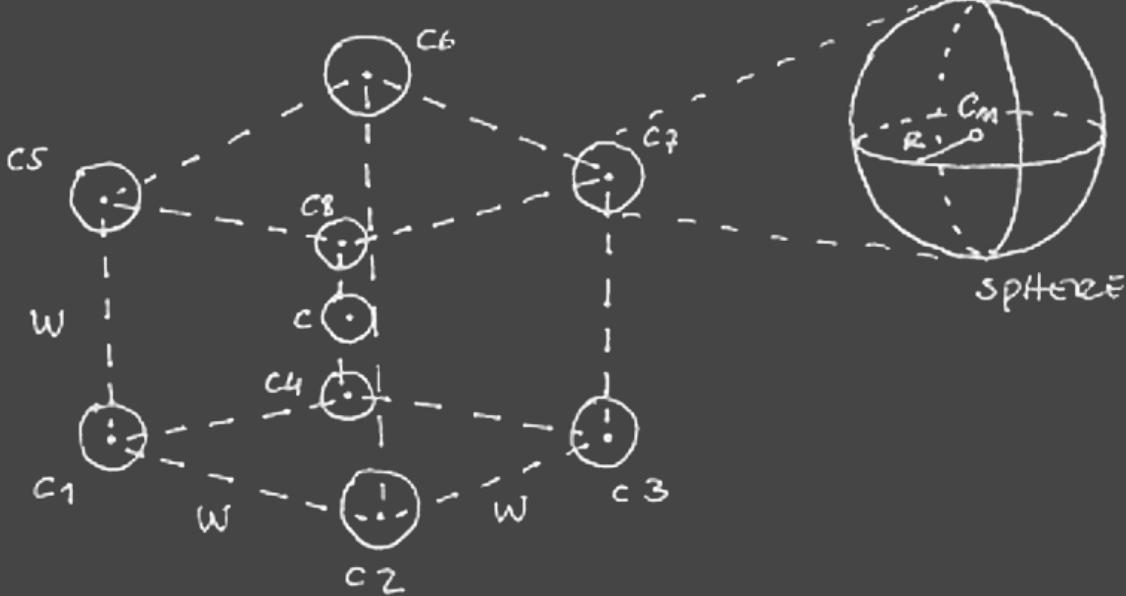
Hex-grid:

$$a: ]0, 1$$

$$b: ]0, 1$$

## Program

```
def cubeSpheres(c, w, r):
```

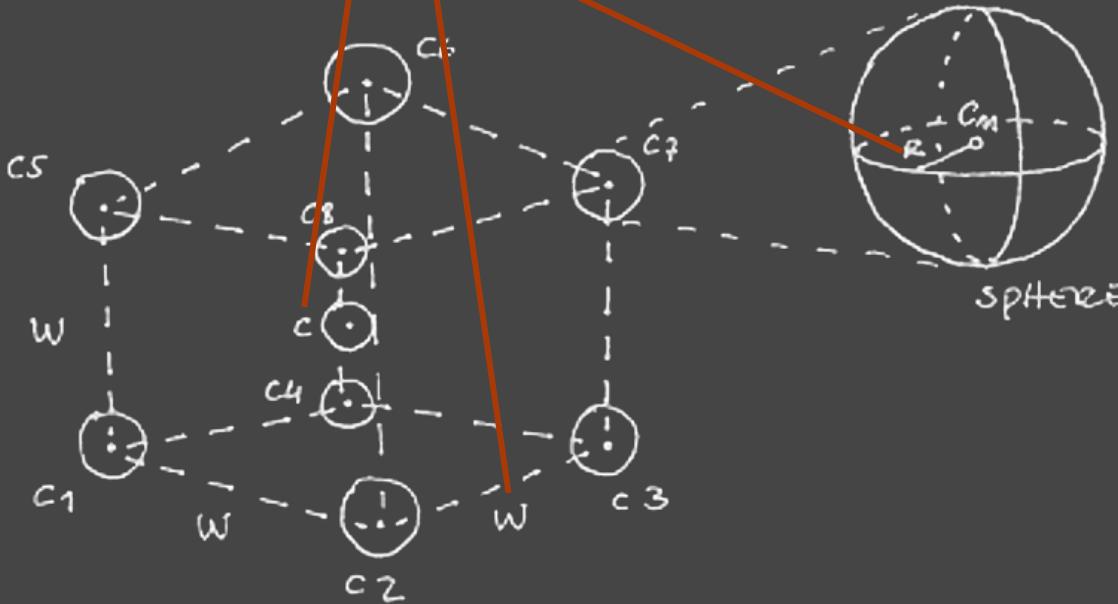


```
c1 = c+xyz (-w, -w, -w)
c2 = c+xyz (+w, -w, -w)
c3 = c+xyz (+w, +w, -w)
c4 = c+xyz (-w, +w, -w)
c5 = c+xyz (-w, -w, +w)
c6 = c+xyz (+w, -w, +w)
c7 = c+xyz (+w, +w, +w)
c8 = c+xyz (-w, +w, +w)
for p in [c1,c2,c3,c4,c5,c6,c7,c8]:
    sphere(p, r)
```

## Documentation

## Program

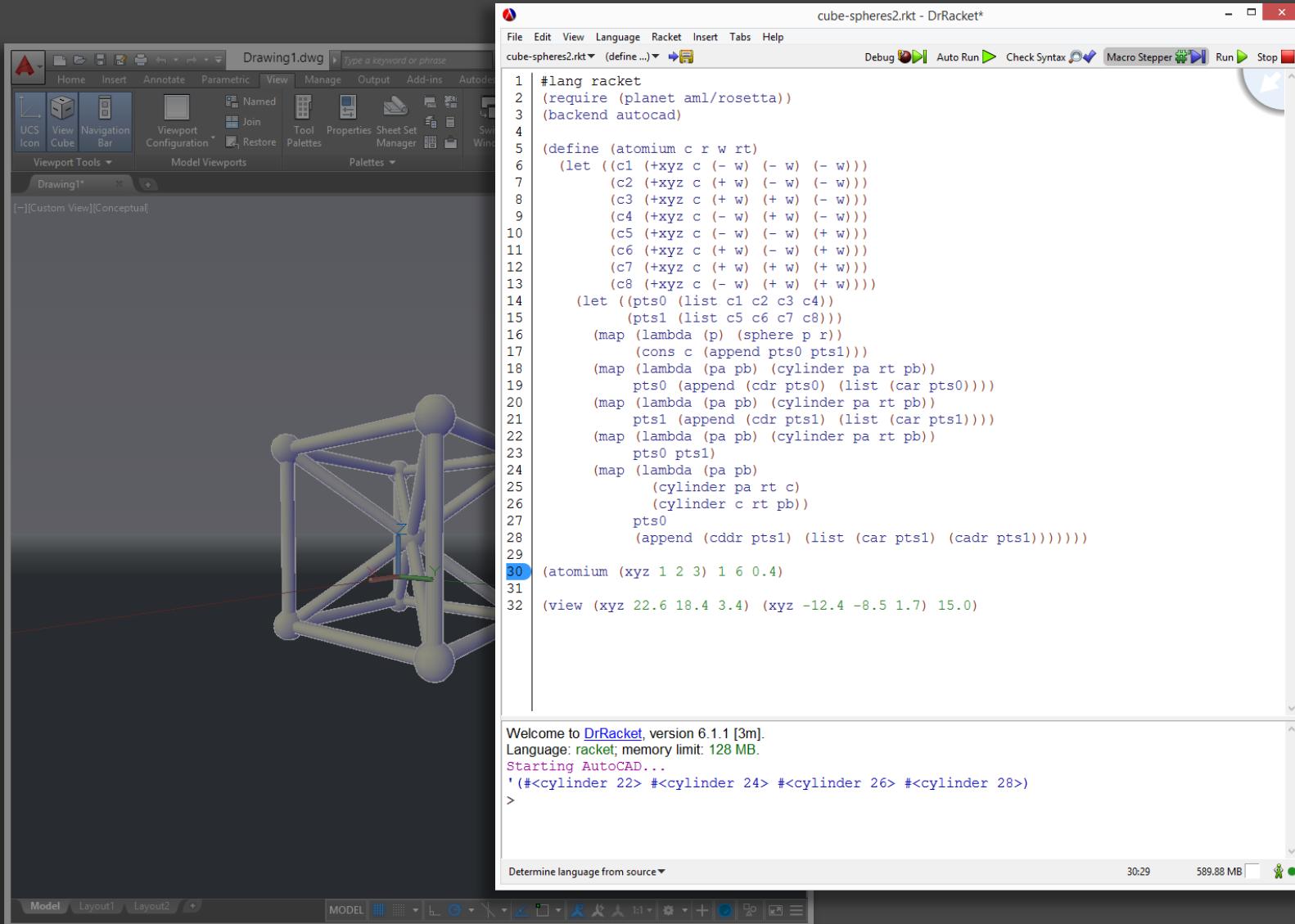
```
def cubeSpheres(c, w, r):
    c1 = c+xyz(-w, -w, -w)
    c2 = c+xyz(+w, -w, -w)
    c3 = c+xyz(+w, +w, -w)
    c4 = c+xyz(-w, +w, -w)
    c5 = c+xyz(-w, -w, +w)
    c6 = c+xyz(+w, -w, +w)
    c7 = c+xyz(+w, +w, +w)
    c8 = c+xyz(-w, +w, +w)
    for p in [c1, c2, c3, c4, c5, c6, c7, c8]:
        sphere(p, r)
```



## Documentation

```
1 #lang racket
2 (require (planet aml/rosetta))
3 (require "sketch-correlation.rkt")
```

# Creators need a medium connection



```
1 #lang racket
2 (require (planet aml/rosetta))
3 (require "tikzgen.rkt")
4 (backend tikz)
5
6 (define (arvore p c a da f)
7   (let ((topo (+pol p c a)))
8     (ramo p topo)
9     (if (< c 2)
10       (folha topo)
11       (begin
12         (arvore topo
13           (* c f)
14           (+ a da)
15           da f)
16         (arvore topo
17           (* c f)
18           (- a da)
19           da f))))))
20
21 (define (ramo p topo)
22   (line p topo))
23
24 (define (folha topo)
25   (circle topo 0.2))
26
27
28
29 ; render
30 (generateTikz)
```

Event  
BUS

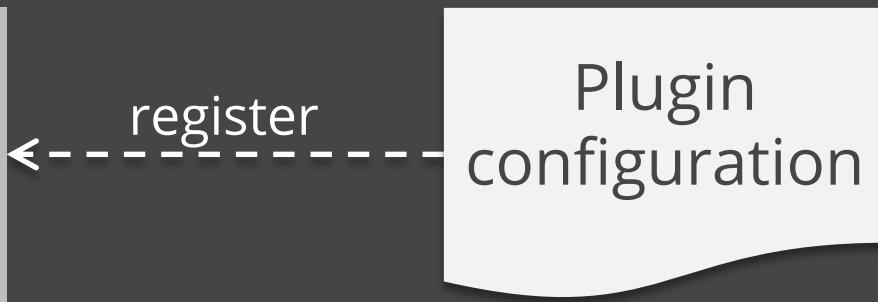
Plugin  
configuration

Editor  
frame

Action  
handler

Tool core

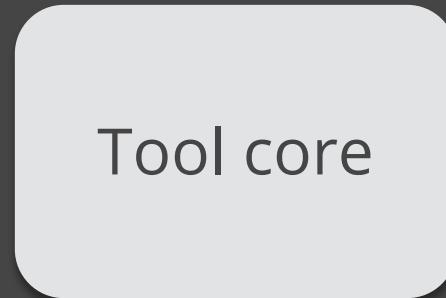
Event  
BUS

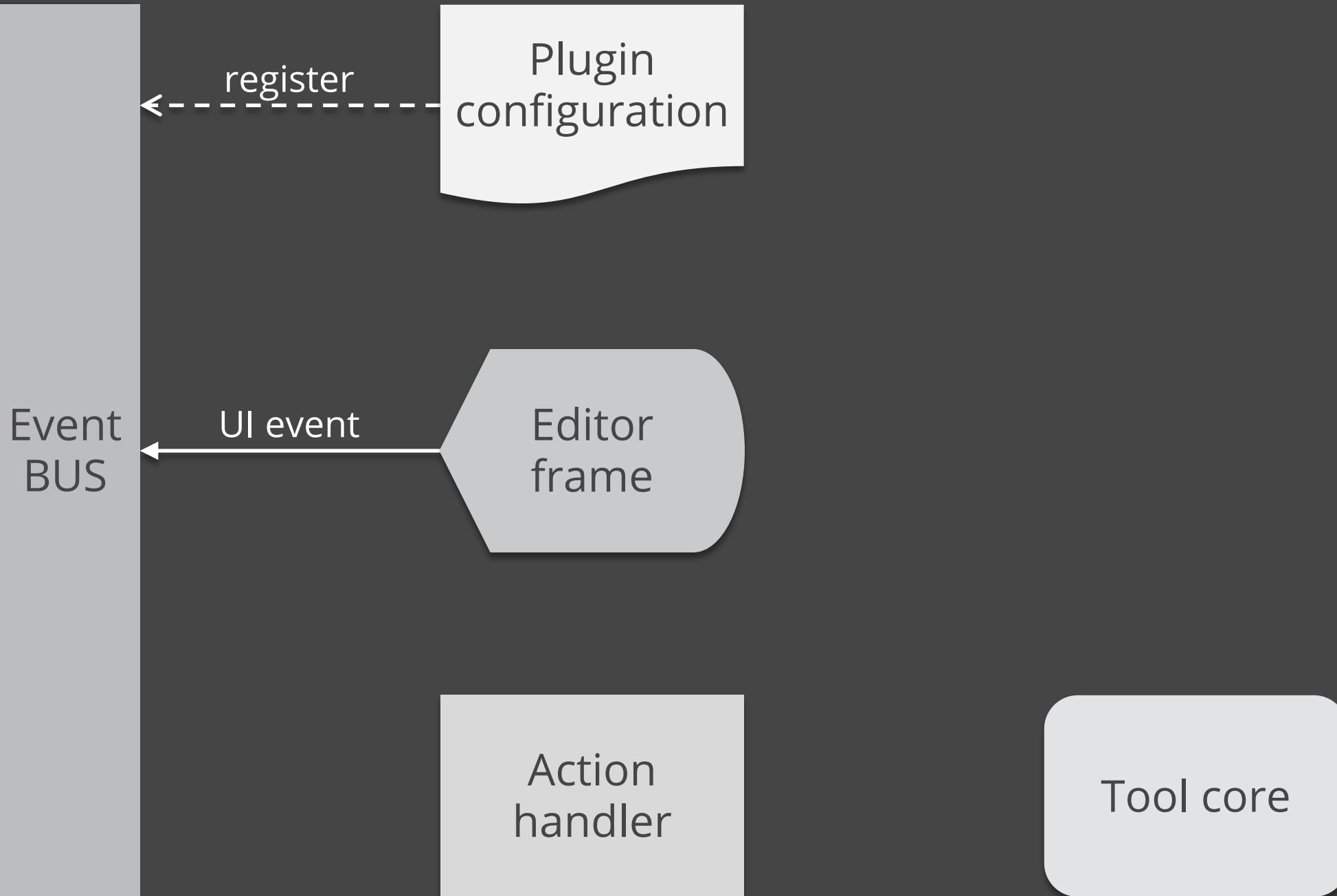


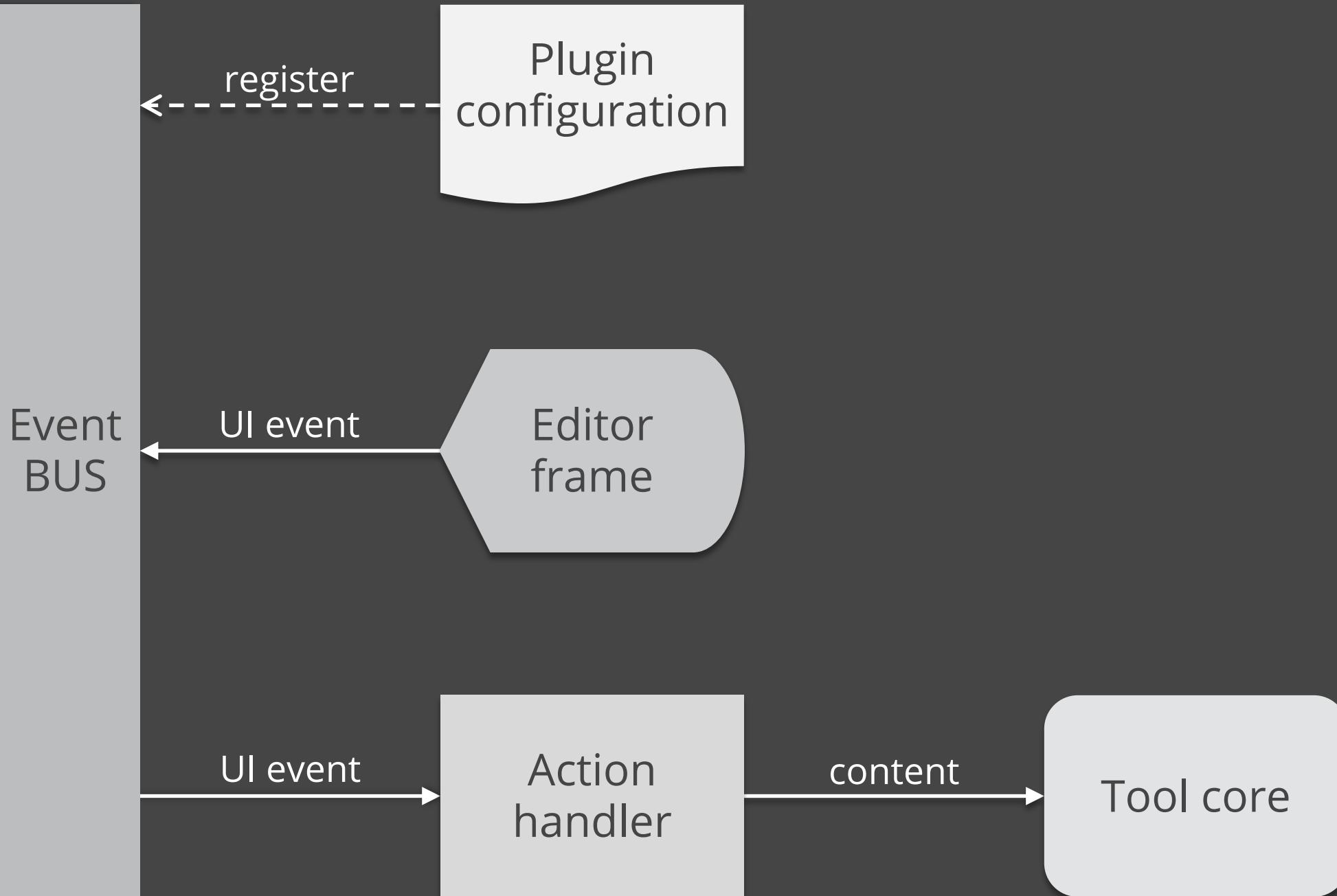
Action  
handler

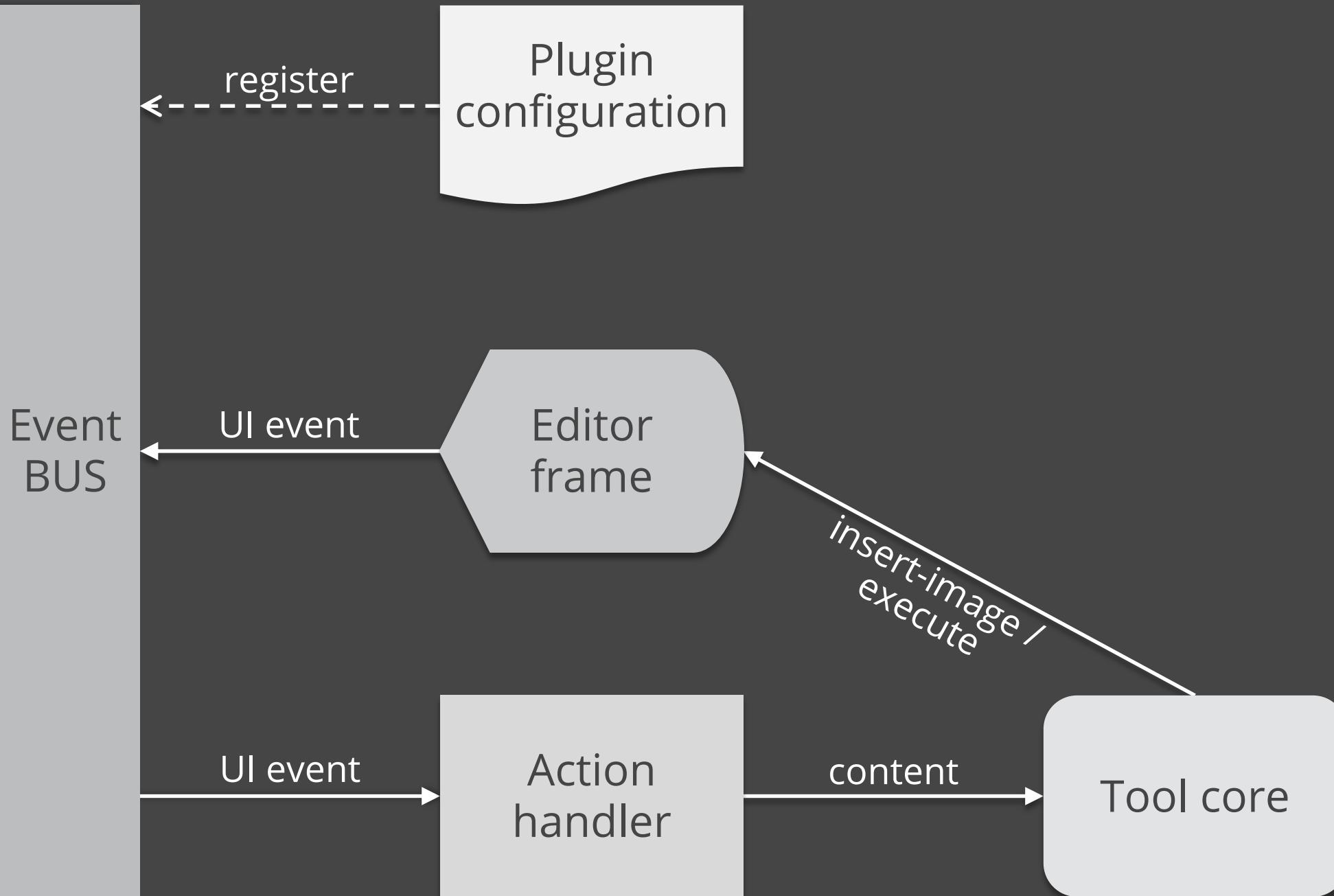


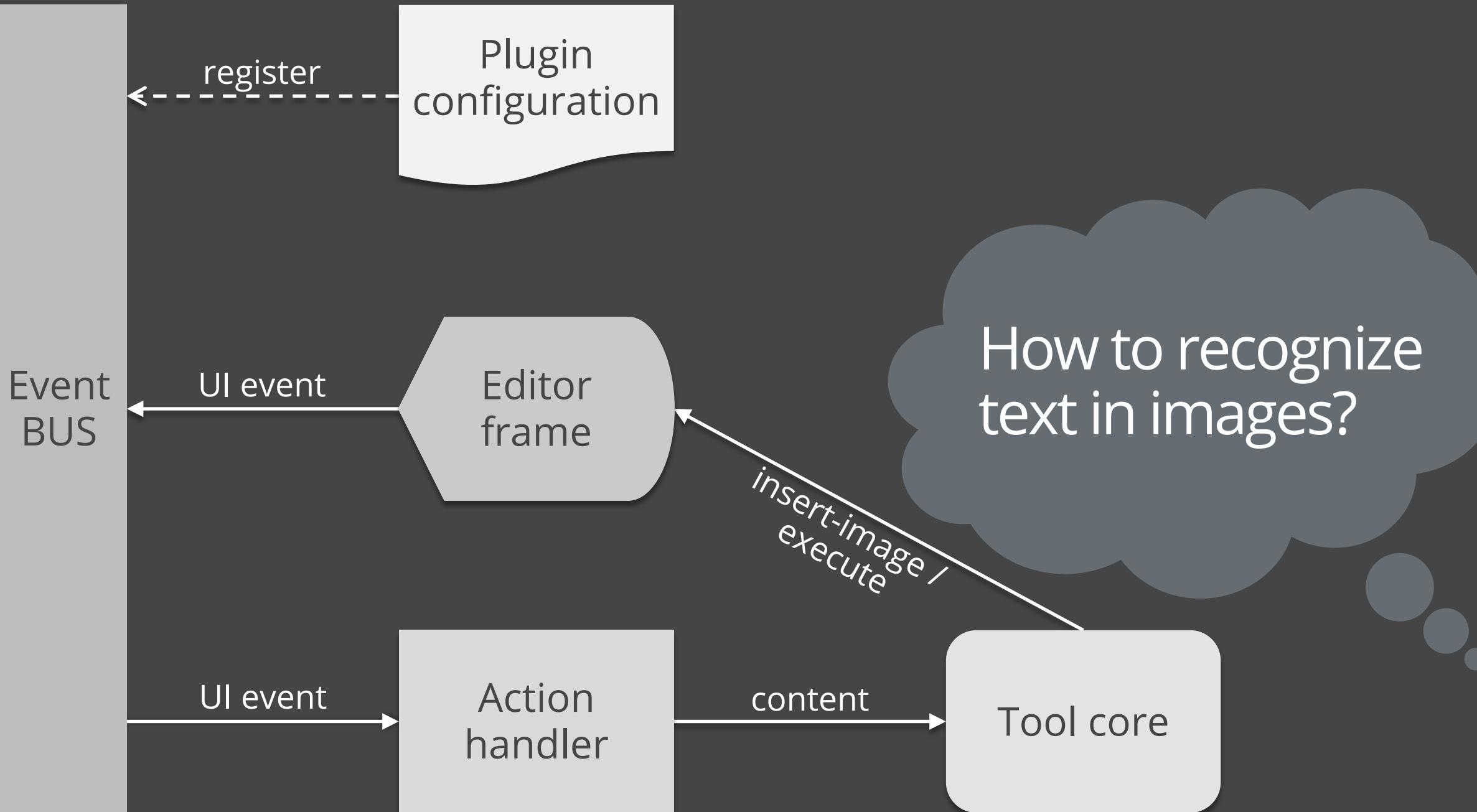
Tool core

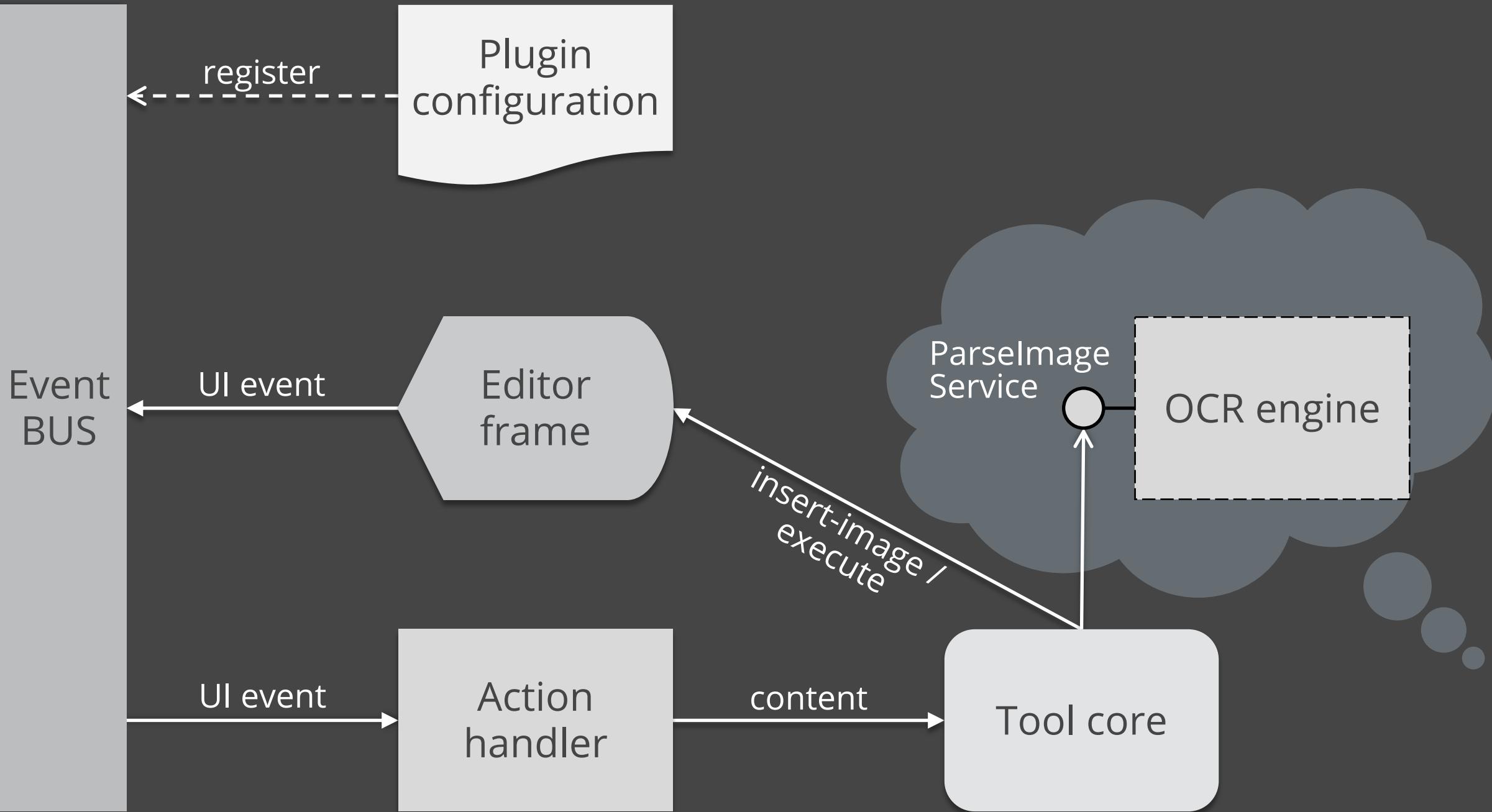












# Evaluation

- Correctness
- Safety
- Performance
- Comparison

# 4

## Conclusion

```

5 (define/img (arvore 1 c a da f) [2 bound occurrences]
6

7 (let ((topo (+pol p c a)))
8   (ramo p topo))

```

- Reduces the effort to understand the code
- GD is a particular case

```

1 #lang racket
2 (require (planet aml/rosetta))
3 (require "tikzgen.rkt")
4 (backend tikz)
5
6 (define (arvore p c a da f)
7   (let ((topo (+pol p c a)))
8     (ramo p topo)
9     (if (< c 2)
10       (folha topo)
11       (begin
12         (arvore topo
13           (* c f)
14           (+ a da)
15           da f)
16         (arvore topo
17           (* c f)
18           (- a da)
19           da f))))))
20
21 (define (ramo p topo)
22   (line p topo))
23
24 (define (folha topo)
25   (circle topo 0.2))
26
27 (arvore (xy 0 0) 30 (/ pi 2) (/ pi 10) 0)

```

- Immediate feedback
- Performance

```

5 (define/img (arvore I c a da f) 2 bound occurrences
6

7 (let ((topo (+pol p c a)))
8   (ramo p topo))

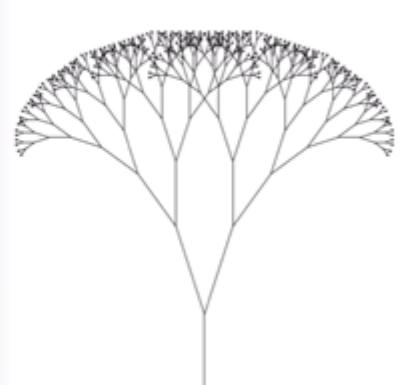
```

```

1 #lang racket
2 (require (planet aml/rosetta))
3 (require "tikzgen.rkt")
4 (backend tikz)
5
6 (define (arvore p c a da f)
7   (let ((topo (+pol p c a)))
8     (ramo p topo)
9     (if (< c 2)
10       (folha topo)
11       (begin
12         (arvore topo
13           (* c f)
14           (+ a da)
15           da f)
16         (arvore topo
17           (* c f)
18           (- a da)
19           da f))))))
20
21 (define (ramo p topo)
22   (line p topo))
23
24 (define (folha topo)
25   (circle topo 0.2))
26
27 (arvore (xy 0 0) 30 (/ pi 2) (/ pi 10) 0)

```

- Reduces the effort to understand the code
- GD is a particular case
- OCR Engine
- Support for image editing
- Immediate feedback
- Performance
- Partial code execution
- Sandboxing



```

5 (define/img (arvore I c a da f) 2 bound occurrences
6

7 (let ((topo (+pol p c a)))
8 (ramo p topo))

```

```

1 #lang racket
2 (require (planet aml/rosetta))
3 (require "tikzgen.rkt")
4 (backend tikz)
5
6 (define (arvore p c a da f)
7   (let ((topo (+pol p c a)))
8     (ramo p topo)
9     (if (< c 2)
10       (folha topo)
11       (begin
12         (arvore topo
13           (* c f)
14           (+ a da)
15           da f)
16         (arvore topo
17           (* c f)
18           (- a da)
19           da f))))))
20
21 (define (ramo p topo)
22   (line p topo))
23
24 (define (folha topo)
25   (circle topo 0.2))
26
27 (arvore (xy 0 0) 30 (/ pi 2) (/ pi 10) 0)

```

- Reduces the effort to understand the code
- GD is a particular case
- OCR Engine
- Support for image editing
- Immediate feedback
- Performance
- Partial code execution
- Sandboxing

Thank you. Questions?