Proxima 2.0
WYSIWYG generic editing for the Web

Martijn Schrage (Oblomov Systems)

Doaitse Swierstra

Lambert Meertens

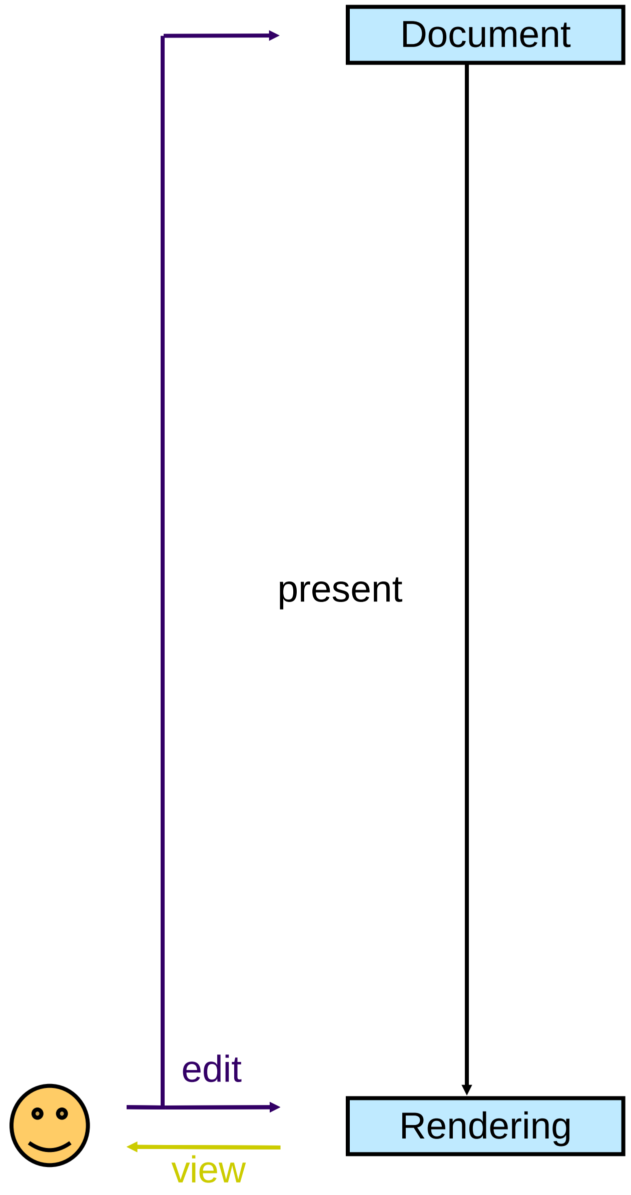
This talk

- Proxima 1.0 overview
- Demo
- Proxima 2.0 plan
- Demo
- Student projects

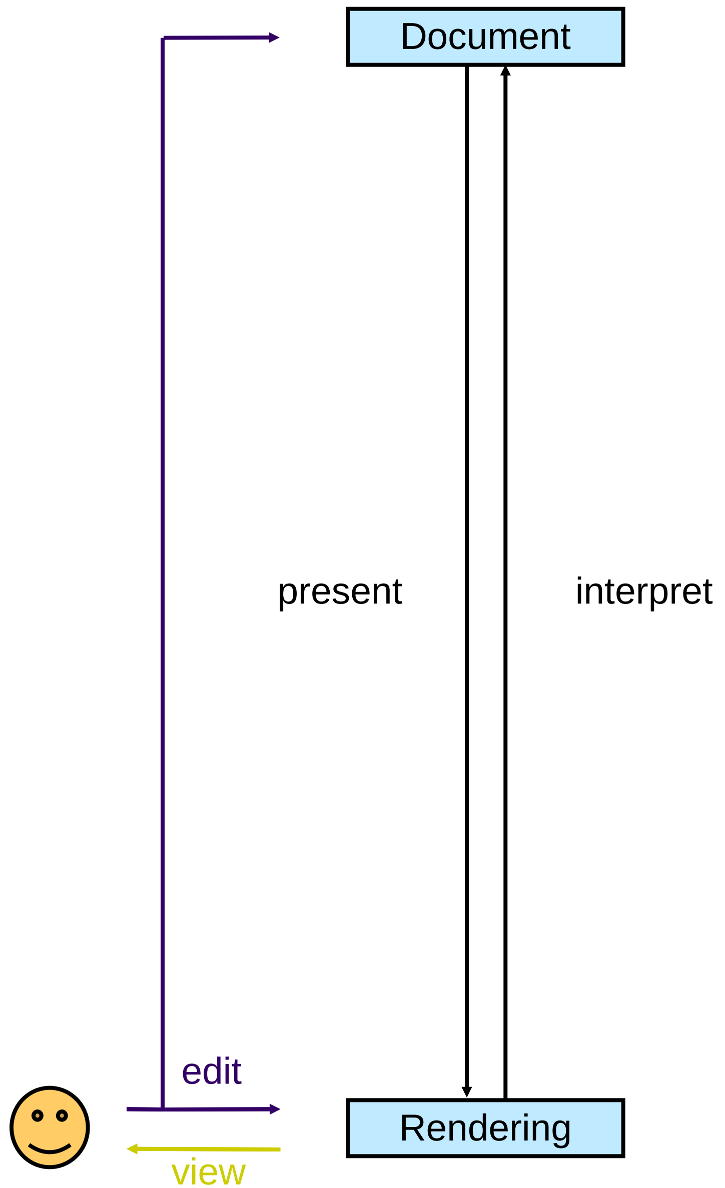
Proxima

- Generic presentation-oriented editor
- Modeless mix of
  - Structural editing: e.g. change section to subsection
  - Free-text editing: e.g. delete [1+2, 5] $\rightarrow$ [15]
- Graphical presentation with derived information
- Applications:
  - Source editor
  - Word processing
  - Form editors
- ~15.000 lines of Haskell

Architecture



Architecture



Presentation process

Document:

```
Root [ Comment ["This", "is", "a", "simple", "expression"]
     , Decl "simple1"
            (IfExp (BoolExp True) (IntExp 1) (IntExp 0))
     ]
```

Rendering:

This is a simple expression

```
simple1 :: Int
simple1 =
  if True then 1
          else 0
```

$$\begin{tikzcd}
\boxed{\text{Document}} \arrow[r, "\text{Evaluation}"'] & \boxed{\text{Enriched Document}} \arrow[r, "\text{Presentation}"'] & \boxed{\text{Presentation}} \arrow[r, "\text{Layout}"'] & \boxed{\text{Layout}} \arrow[r, "\text{Arrangement}"'] & \boxed{\text{Arrangement}} \arrow[r, "\text{Rendering}"'] & \boxed{\text{Rendering}}
\end{tikzcd}$$

Evaluation

Document:

```
Root [ Comment ["This", "is", "a", "simple", "expression"]
     , Decl "simple1"
            (IfExp (BoolExp True) (IntExp 1) (IntExp 0))
     ]
```

Enriched Document:

```
Root [ Comment ["This", "is", "a", "simple", "expression"]
     , TypeDecl "simple1" IntType
     , Decl "simple1"
            (IfExp (BoolExp True) (IntExp 1) (IntExp 0))
     ]
```

$$\begin{tikzcd}
\boxed{\text{Document}} \arrow[r, "\text{Evaluation}"'] & \boxed{\text{Enriched Document}} \arrow[r] & \boxed{\text{Presentation}} \arrow[r] & \boxed{\text{Layout}} \arrow[r] & \boxed{\text{Arrangement}} \arrow[r] & \boxed{\text{Rendering}}
\end{tikzcd}$$

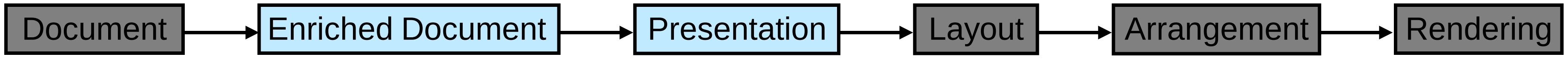
Presentation

Enriched Document:

```
Root [ Comment ["This", "is", "a", "simple", "expression"]
     , TypeDecl "simple1" IntType
     , Decl "simple1"
            (IfExp (BoolExp True) (IntExp 1) (IntExp 0))
     ]
```

Presentation:

```
Col [ With {font}
      (Formatter [ "This", "is", "a", "simple", "expression" ])
    , With {font}
      (Tokens [ Token (1,0) "simple1", Token (0,1) "::"
              , Token (0,1) "Int" ])
    , With {font}
      (Tokens [ Token (1,0) "simple1", Token (0,1) "="
              , Token (1,2) "if", ... ]) ]
```



Presentation

Layout

Presentation:

```
Col [ ...
    , With {font}
      (Tokens [ Token (1,0) "simple1", Token (0,1) "::"
              , Token (0,1) "Int" ])
    , With {font}
      (Tokens [ Token (1,0) "simple1", Token (0,1) "="
              , Token (1,2) "if", ... ]) ]
```

Layout:

```
Col [ ...
    , With {font}
      (Col [ ""
           , Row [ "simple1", " ", "::", " ", "Int" ] ])
    , With {font}
      (Col [ Row [ "simple1", " ", "=" ]
           , Row [ "  ", "if", " ", "True", " ", "then" ]
           , Row [ "           ", "else", ... ] ) ]
```

$$\begin{tikzcd}\text{Document} \arrow[r] & \text{Enriched Document} \arrow[r] & \text{Presentation} \arrow[r] & \text{Layout} \arrow[r] & \text{Arrangement} \arrow[r] & \text{Rendering}\end{tikzcd}$$

Layout

Arrangement

Layout:

```
Col [ With {font}
        (Formatter [ "This", "is", "a", "simple", "expression" ])
    , With {font}
        (Col [ ""
             , Row [ "simple1", " ", "::", " ", "Int" ] ])
    , With {font}
        (Col [ Row [ "simple1", " ", "=" ], ... ]) ]
```

Arrangement:

```
Col(0,0)(80×84)
    [ Col(0,0)(80×24)    [ Row(0,0)(80×12)  [ "This"(0,0)(17×12), "is"(25,0)(6×12), ... ]
                         , Row(0,12)(80×12) [ "expression"(53,0)(27×12)  ]
    , Col(0,24)(75×24)   [ ""(0,0)(0×12)
                         , Row{..} [ "simple1"{..}, " "{..}, "::"{..}, ... ] ]
    , Col(0,48)(80×36)   [ Row{..} [ "simple1"{..}, " "{..}, "="{..} ]
                         , ... ] ]
```

$$\begin{tikzcd}
\text{Document} \arrow[r] & \text{Enriched Document} \arrow[r] & \text{Presentation} \arrow[r] & \text{Layout} \arrow[r] & \text{Arrangement} \arrow[r] & \text{Rendering}
\end{tikzcd}$$

Arrangement

Rendering

Arrangement:

```
Col(0,0)(80×84)
    [ Col(0,0)(80×24)   [ Row(0,0)(80×12) [ "This"(0,0)(17×12), "is"(25,0)(6×12), ... ]
                        , Row(0,12)(80×12) [ "expression"(53,0)(27×12) ]
    , Col(0,24)(75×24)  [ Row{..}  [ ]
                        , Row{..}  [ "simple1"{..}, " "{..}, "::"{..}, ... ] ]
    , Col(0,48)(80×36)  [ Row{..}  [ "simple1"{..}, " "{..}, "="{..} ]
                        , ... ] ]
```

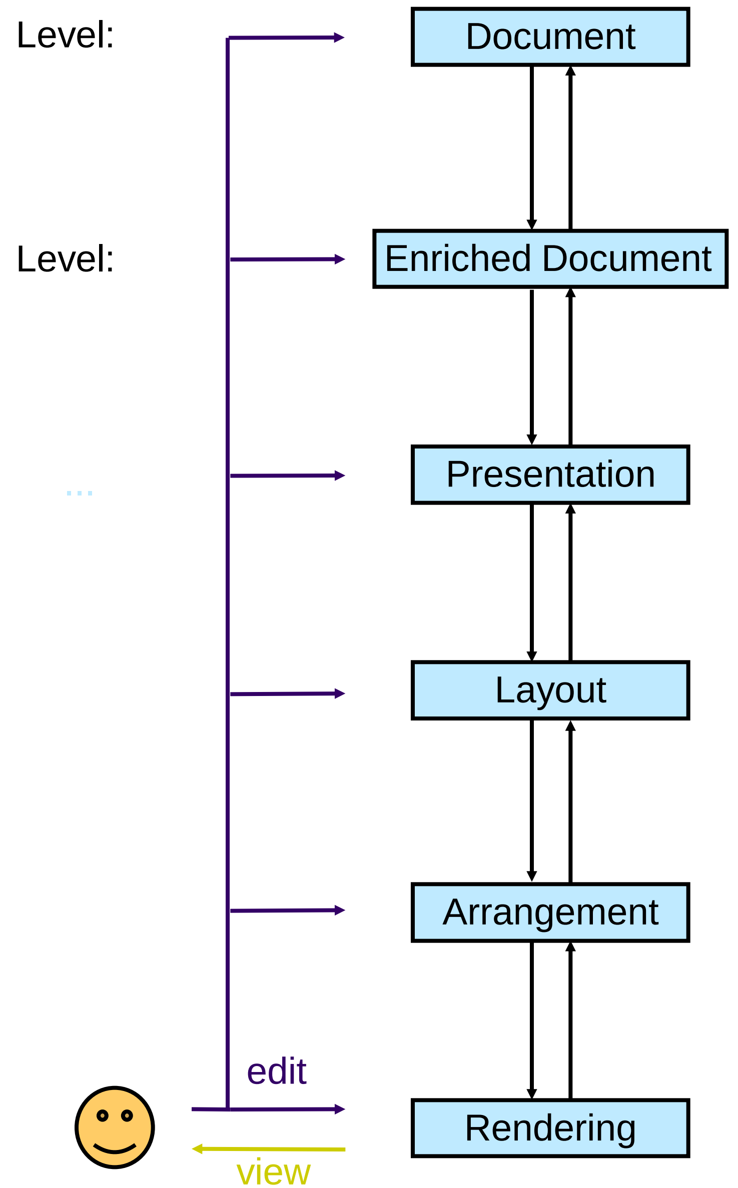
Rendering:

```
This is a simple
expression

simple1 :: Int
simple1 =
  if True then 1
              else 0
```

$$\begin{tikzcd}
\text{Document} \arrow[r] & \text{Enriched Document} \arrow[r] & \text{Presentation} \arrow[r] & \text{Layout} \arrow[r] & \text{Arrangement} \arrow[r] & \text{Rendering}
\end{tikzcd}$$

Rendering

Architecture



Internal document tree

Document + derived values

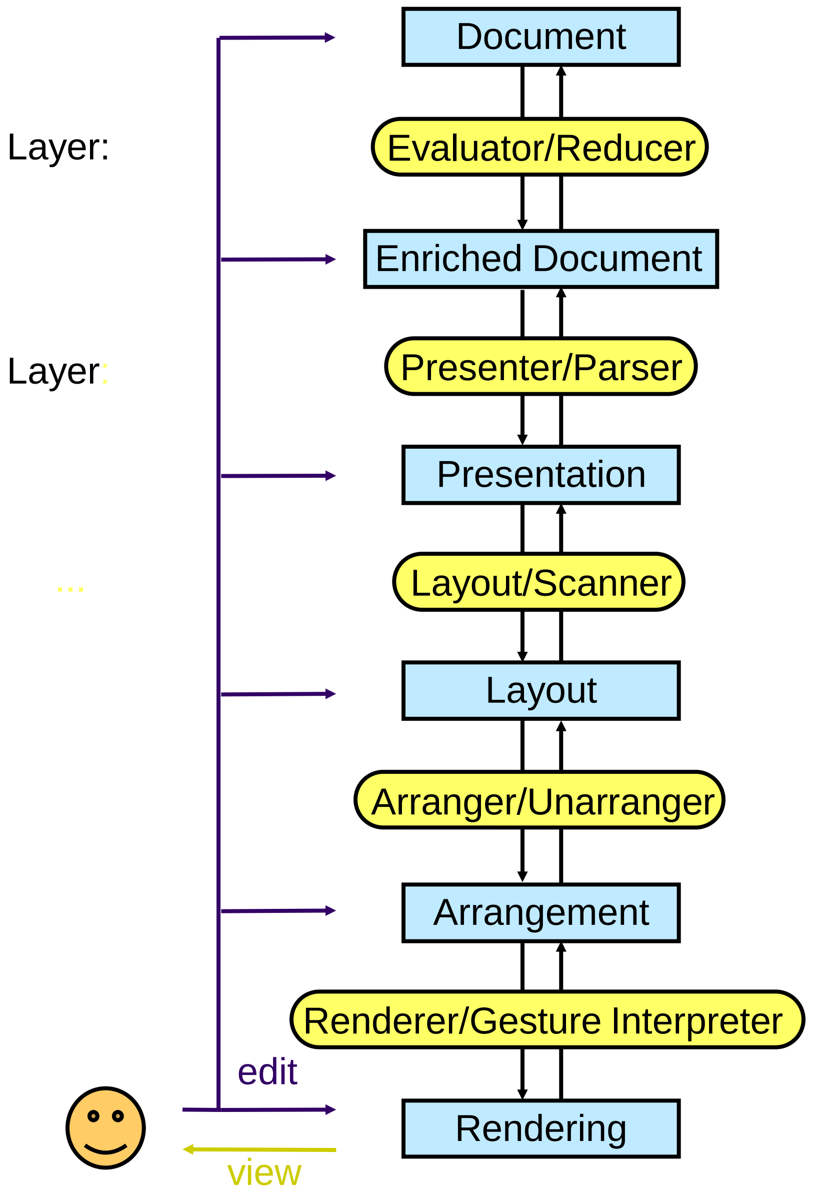
Logical presentation: rows, columns, tokens, etc.

Presentation + explicit whitespace

Presentation with exact positions & sizes

Image

Architecture



Internal document tree

Document + derived values

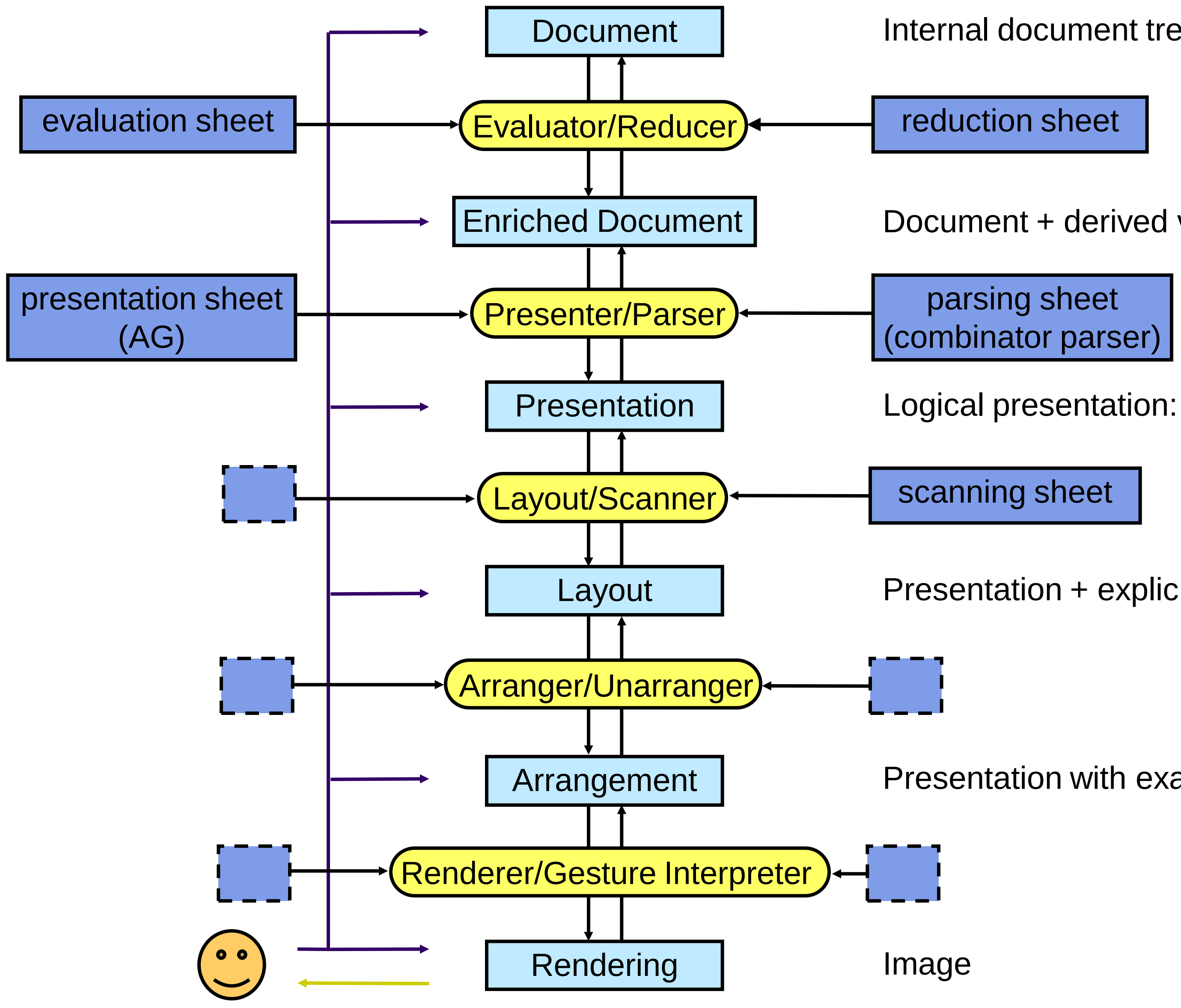
Logical presentation: rows, columns, tokens, etc.

Presentation + explicit whitespace

Presentation with exact positions & sizes

Image

Architecture



Internal document tree

Document + derived values

Logical presentation: rows, columns, tokens, etc.

Presentation + explicit whitespace

Presentation with exact positions & sizes

Image

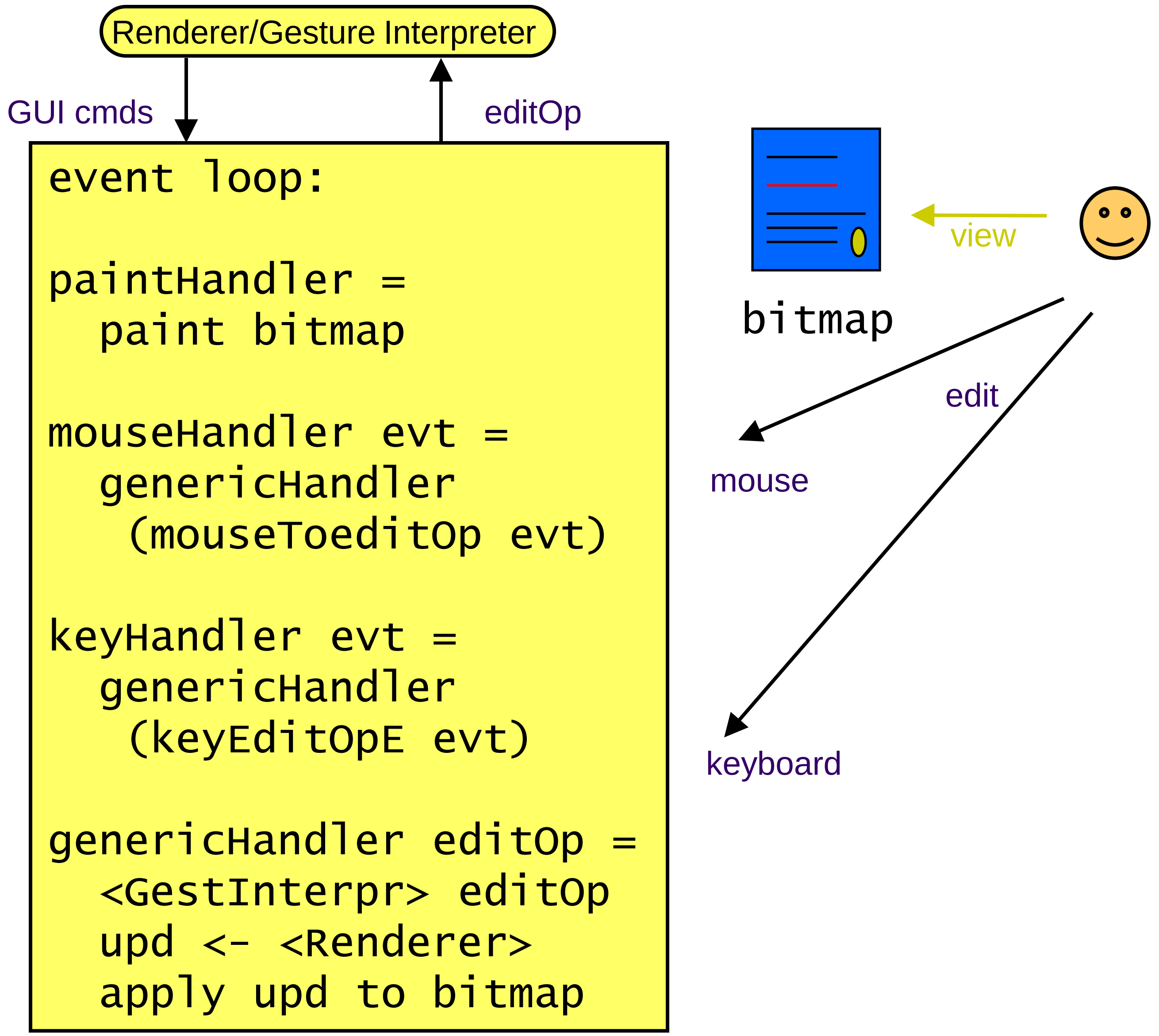
Demo

- Helium editor
  - Functional language similar to Haskell
  - Graphical presentations
  - In-place parse and type errors
  - Derived values in source
  - 1200 lines of code
- Bayesian network documentation editor
  - Documentation for Bayesian Networks
  - Editable graphs with multiple views
  - Word-processor functionality
  - Derived tables
  - 800 lines of code

Proxima 2.0

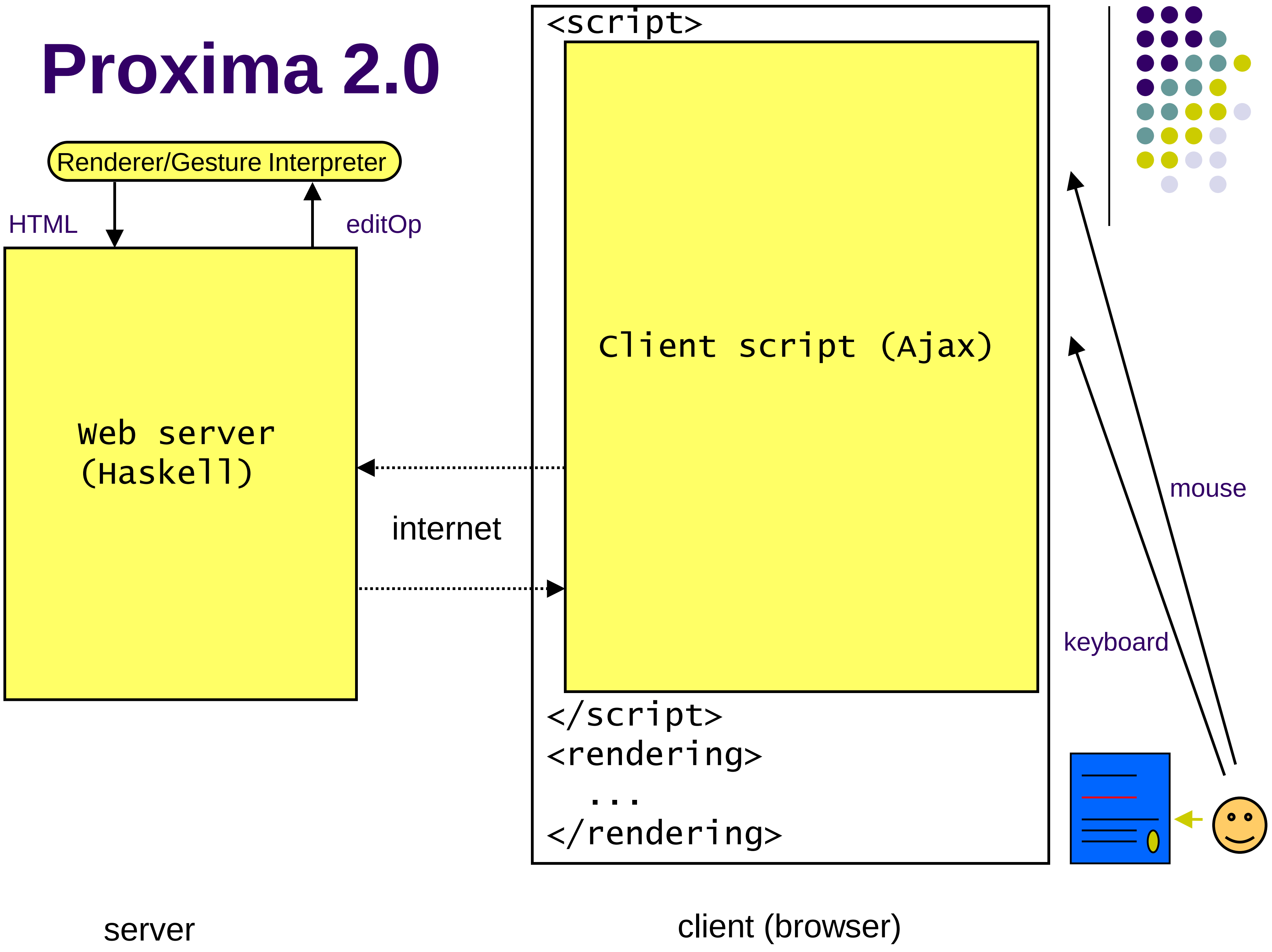
- Proxima 1.0
  - Not a web application
  - GUI dependency makes it hard to build
  - Has to be installed
- Proxima 2.0
  - Server-based with thin Ajax client
  - Server is easy to compile
  - No installation
  - Editors can be part of web-page
  - Runs on small client machines, e.g. iPhones
- Grant from Nlnet foundation
  - Oblomov Systems

Proxima 1.0



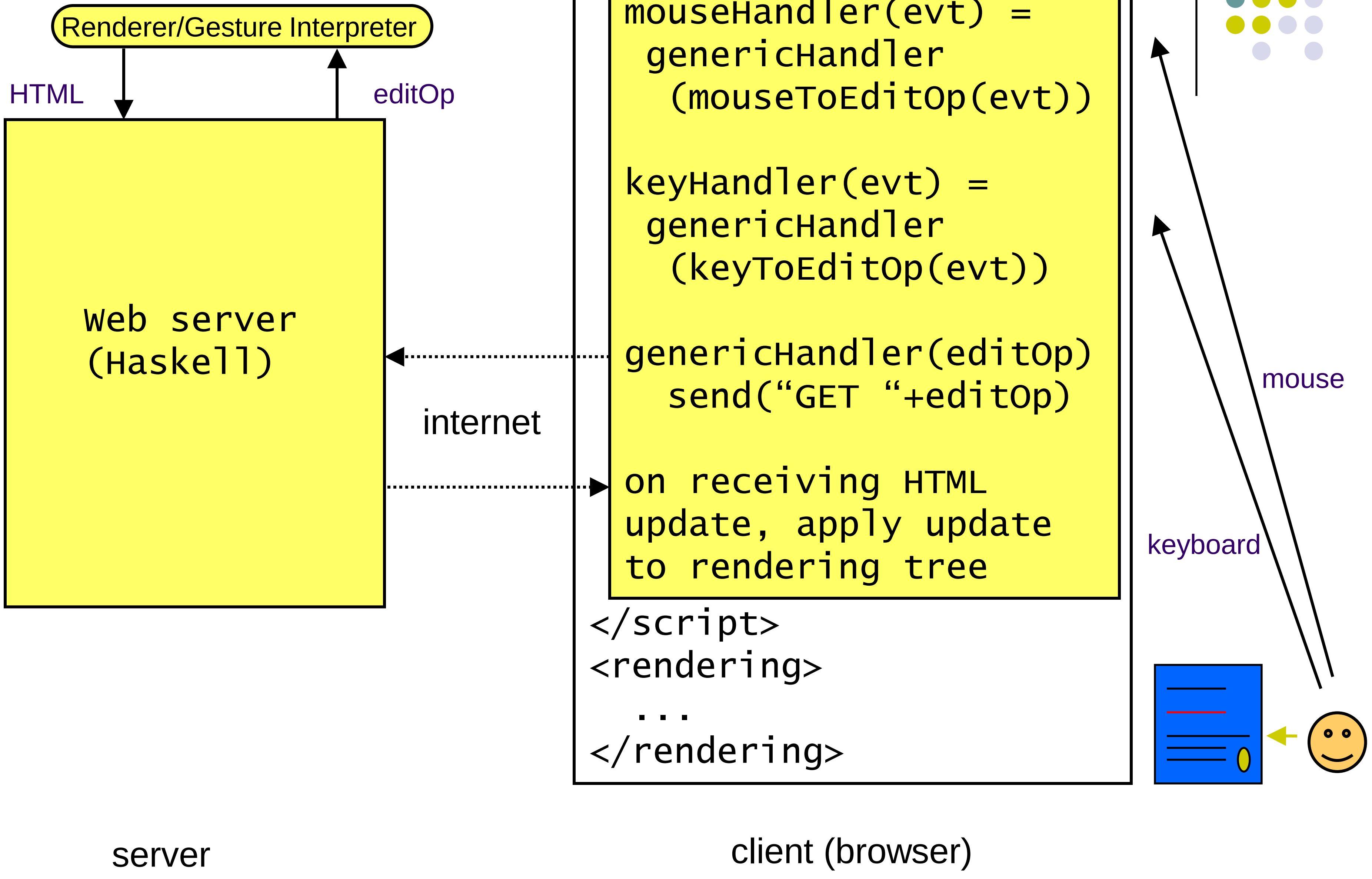
```
event loop:

paintHandler =
  paint bitmap

mouseHandler evt =
  genericHandler
   (mouseToeditOp evt)

keyHandler evt =
  genericHandler
   (keyEditOpE evt)

genericHandler editOp =
  <GestInterpr> editOp
  upd <- <Renderer>
  apply upd to bitmap
```

Proxima 2.0



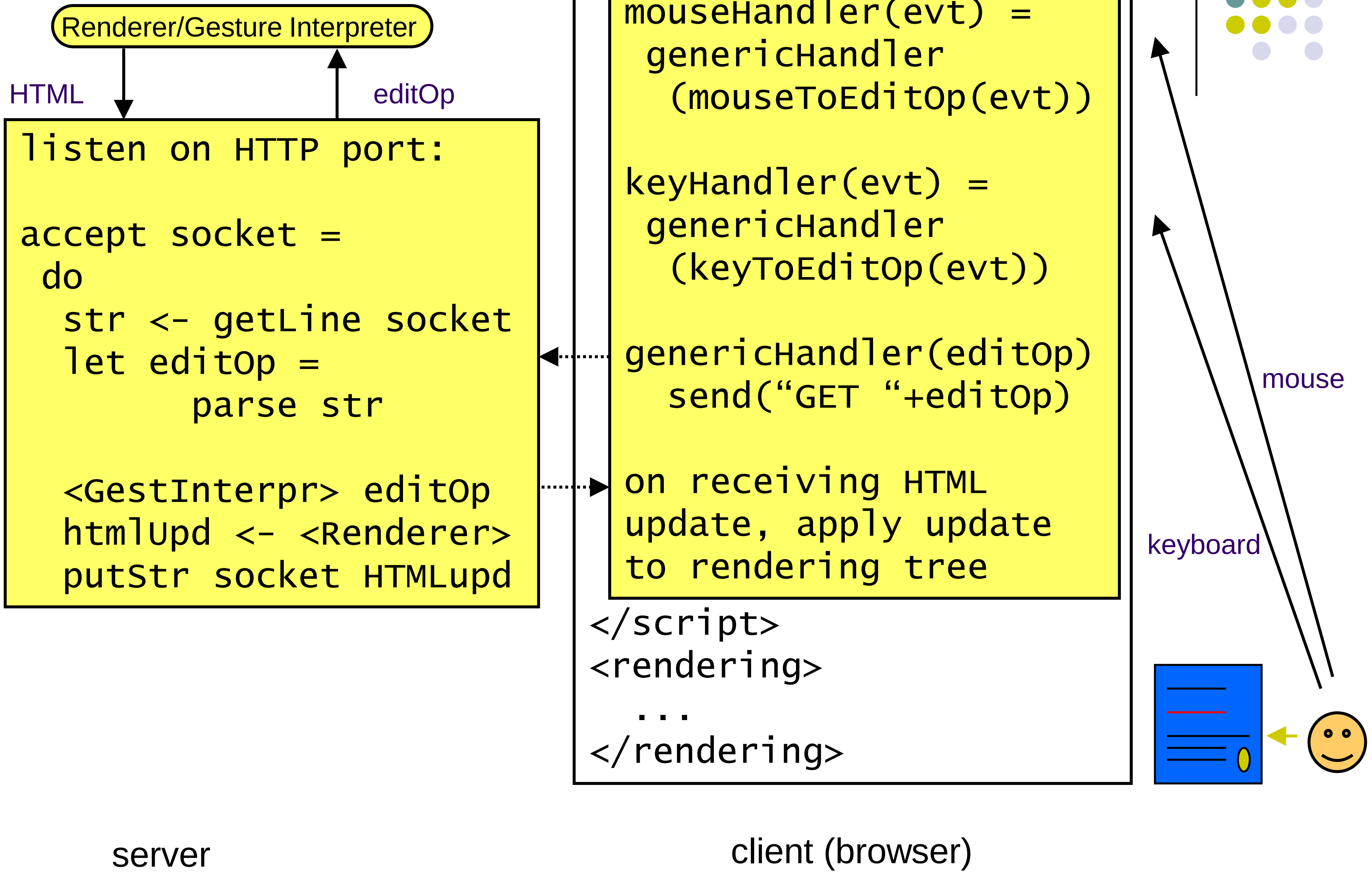
Renderer/Gesture Interpreter
HTML
editOp
Web server (Haskell)
internet
<script>
Client script (Ajax)
</script>
<rendering>
...
</rendering>
mouse
keyboard
server
client (browser)

Proxima 2.0



```
<script>
  event loop:

  mouseHandler(evt) =
   genericHandler
     (mouseToEditOp(evt))

  keyHandler(evt) =
   genericHandler
     (keyToEditOp(evt))

  genericHandler(editOp)
    send("GET "+editOp)

  on receiving HTML
  update, apply update
  to rendering tree
</script>
<rendering>
   ...
</rendering>
```

Proxima 2.0



```
listen on HTTP port:

accept socket =
 do
  str <- getLine socket
  let editOp =
        parse str

  <GestInterpr> editOp
  htmlUpd <- <Renderer>
  putStr socket HTMLupd
```

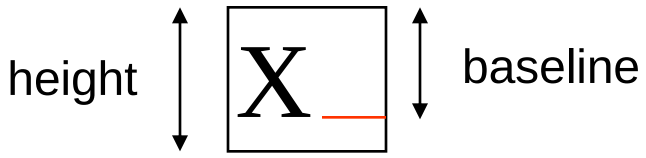
```
<script>
  event loop:

  mouseHandler(evt) =
   genericHandler
    (mouseToEditOp(evt))

  keyHandler(evt) =
   genericHandler
    (keyToEditOp(evt))

  genericHandler(editOp)
    send(“GET “+editOp)

  on receiving HTML
  update, apply update
  to rendering tree
</script>
<rendering>
  ...
</rendering>
```

Renderer

- Convert each arrangement to HTML elements with exact sizes and positions
- `StringA:`
  - convert to HTML
- graphical elements: (`CircleA`, `PolyA`, etc.)
  - convert to SVG
- `ImageA` <filepath>
  - convert to “`<img src='`<filepath>`' />`”
  - Server needs to handle image requests

Font metrics

- Server-side not possible
  - Requires GUI library or parsing of font tables
  - Different browsers may render differently
- Query the client for metrics
- Javascript has no `getFontMetrics`
- Solution:
  - Invisible characters on the page
  - Widths: for each char, measure width of row of 10
  - Height & baseline: render ‘x’ and 0 height element (——)



Popup menus

- Proxima 1.0:
  - Popup menu items are result of function

- Proxima 2.0:
  - Client sends popup request to Proxima server
  - Server returns list of menu items
  - Client shows menu and sends index of selected item
  - Server processes edit command and returns update

Demo Proxima 2.0

- Very basic prototype!

Work plan

- Implement Proxima 2.0
- Speed
  - Drag & drop
  - Text input
  - More incrementality, e.g. block moves
- Session handling
- File handling
- Improve Proxima 1.0
- Build sample editors

Student projects

- Build editors
  - Folding outline task-list editor
  - Sudoku editor

- Multi-user editing in Proxima
- Connecting to a data base
- Generic graph edit model

Questions?

http://www.cs.uu.nl/wiki/Proxima

martijn@oblomov.biz

(or martijn@cs.uu.nl)