GEFÖRDERT VOM

Bundesministerium
für Bildung
und Forschung

Formalizing Security Properties of User Interfaces

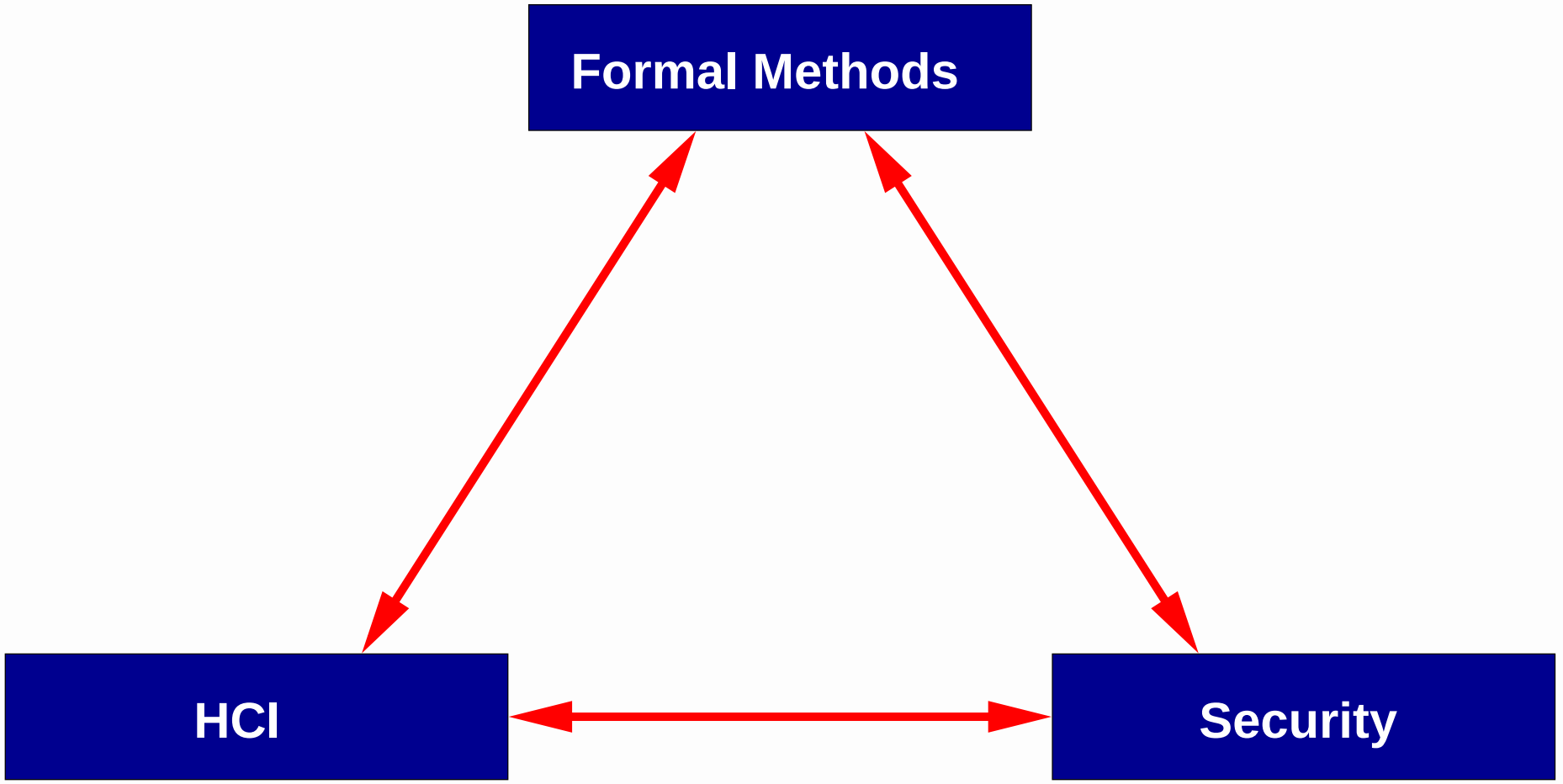
Gerd Beuster

gb@uni-koblenz.de

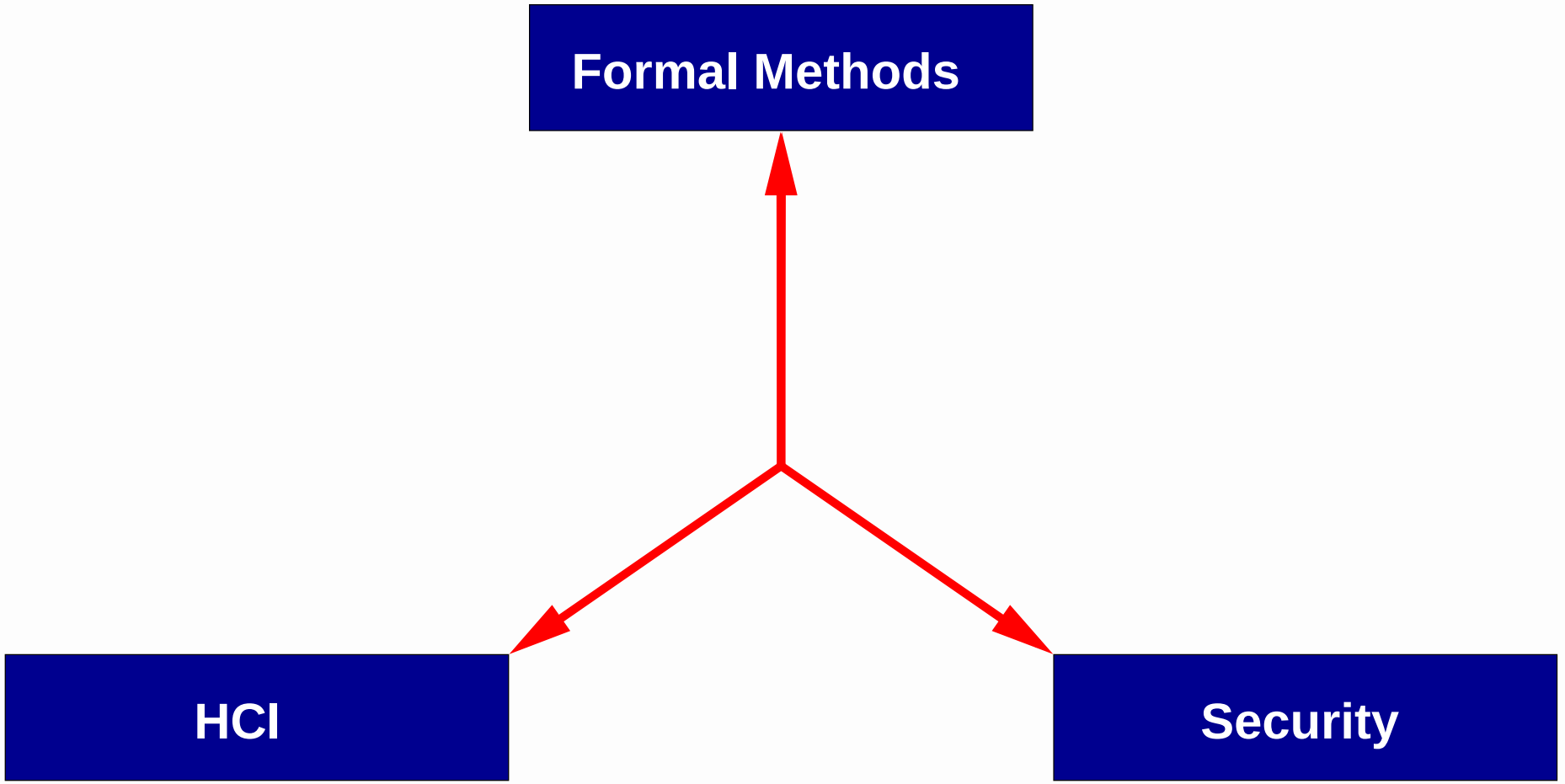
Universität Koblenz-Landau

verisoft

Formal Methods, HCI, Security



Formal Methods, HCI, Security



Motivation

Bringing together formal methods, HCI, and security, because. . .

- user interfaces of security-critical systems become more complex.
- for some systems, security lies in the user interface.
- an increasing number of real-world attacks is targeted at the user interface.

Steps

1. Formalize user interface
   (User Interface $\Longleftrightarrow$ Formal methods)
2. Define requirement for a secure user interface
   (Security $\Longleftrightarrow$ HCI)
3. Formalize UI security requirements
   (Formal methods $\Longleftrightarrow$ Security)
4. $\Longrightarrow$ Formal model of secure user interfaces
5. (User model?)

Steps

1. Formalize user interface
   (User Interface $\Longleftrightarrow$ Formal methods)
2. Define requirement for a secure user interface
   (Security $\Longleftrightarrow$ HCI)
3. Formalize UI security requirements
   (Formal methods $\Longleftrightarrow$ Security)
4. $\Longrightarrow$ Formal model of secure user interfaces
5. (User model?)

Expected Functionality

Generic, modern user interface

- Keyboard/mouse input
- Text or bitmap output
- Multiple screen areas/windows

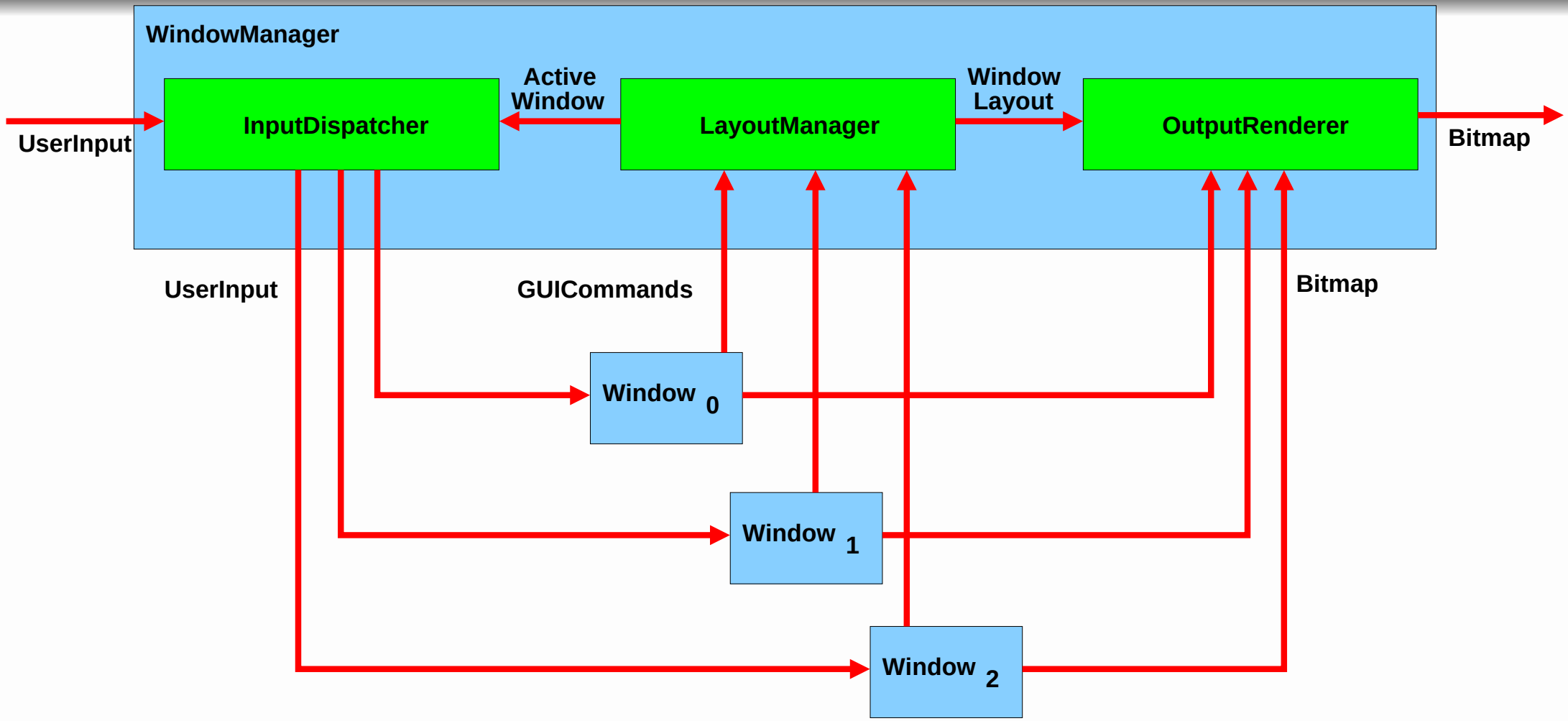
Formalize User Interface

There are many methods to treat user interfaces formally. . .

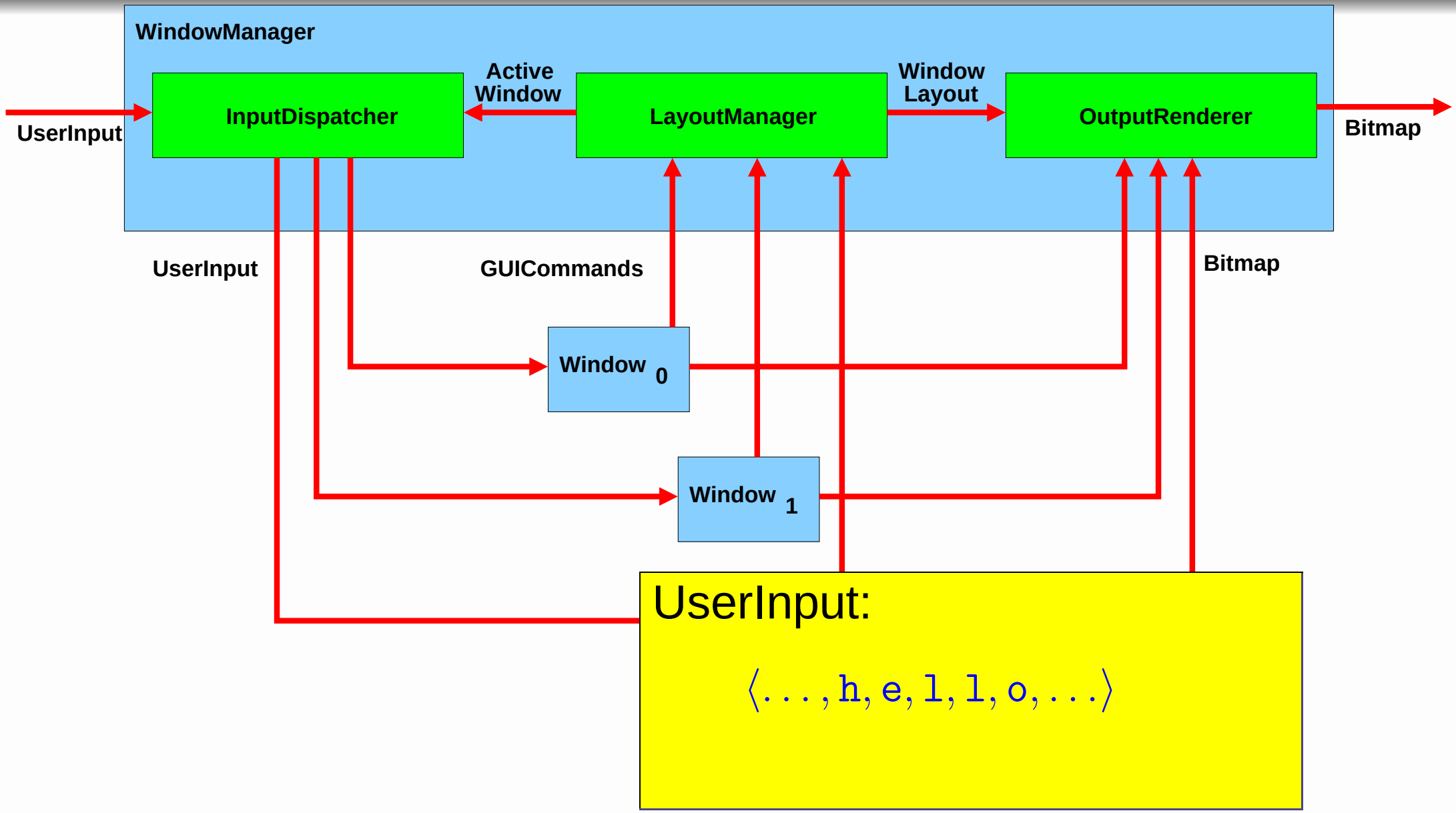
. . . for our purposes, process algebras seem to be a good choice.

- formal description of interfaces
- can handle concurrency
- support different levels of abstraction
- agnostic to internal properties of components/data
- tool support

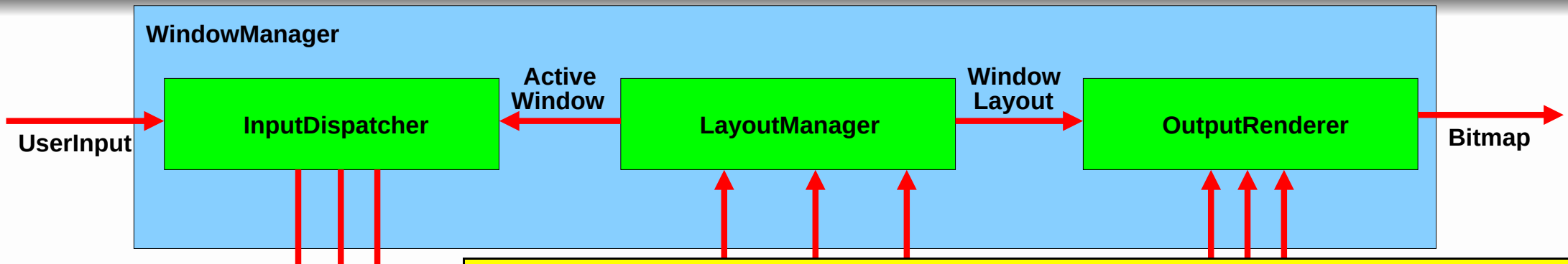
Generic model of multi-window applications



Generic model of multi-window applications



Generic model of multi-window applications



UserInput

InputDispatcher

in $i : UserInput$

in $j : ActiveWindow$

out $o[\mathbb{N}] : UserInput$

$\forall k \in \mathbb{N} : f[\langle\rangle](i, j, k) = o[k]$

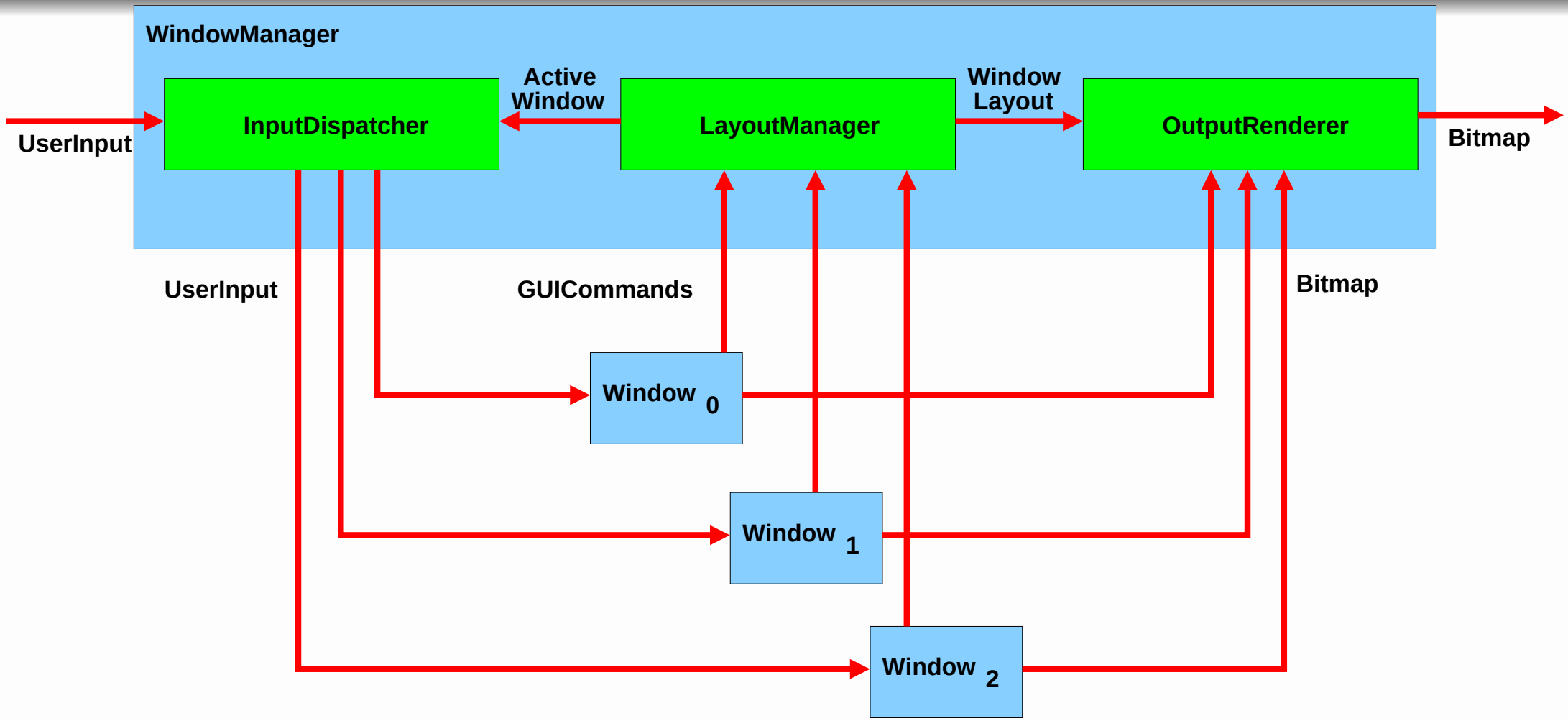
where f so that

$f[m](as, \langle b\rangle \frown bs, l) = f[b](as, bs, l)$

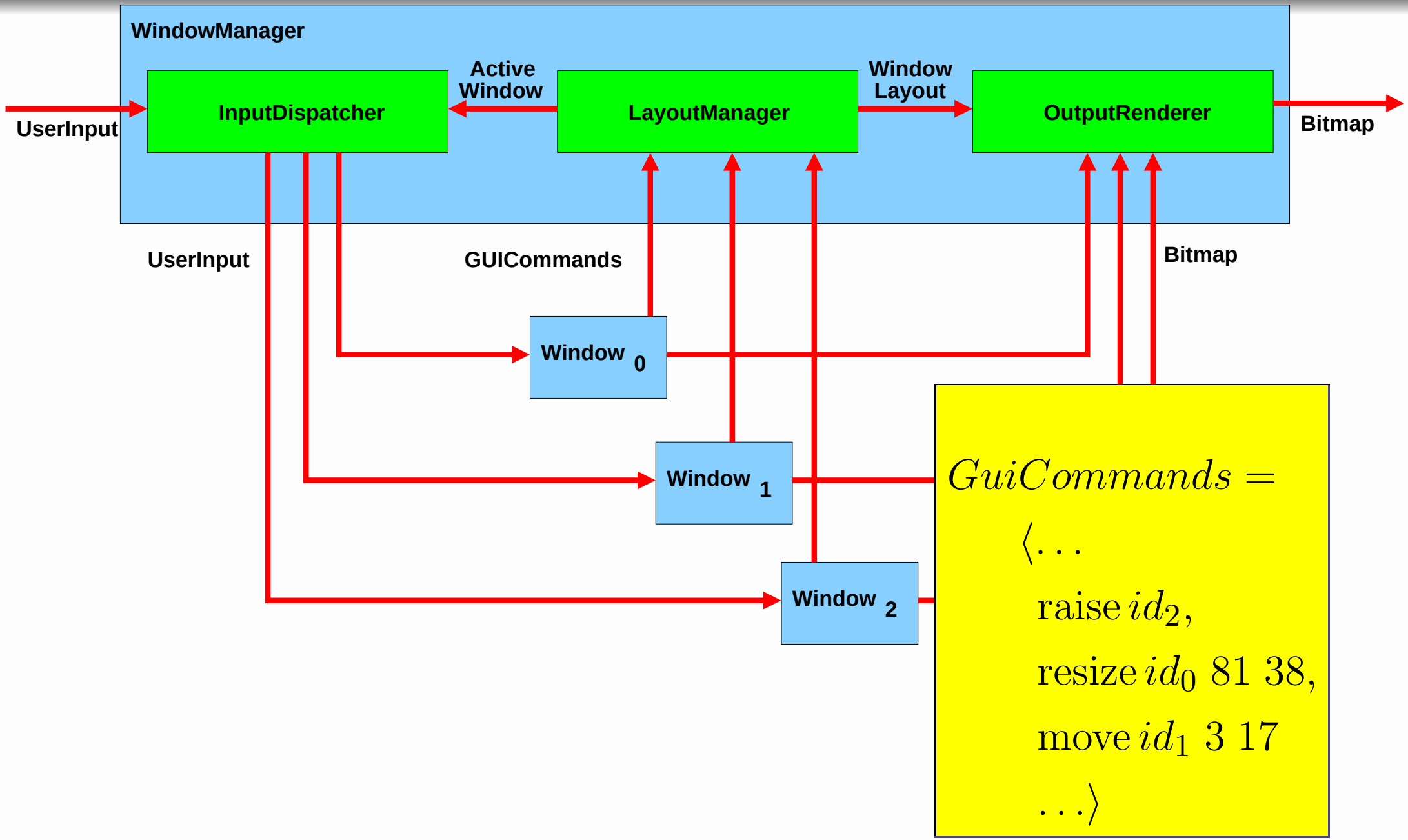
$f[m](\langle a\rangle \frown as, bs, l) = f[m](as, bs, l)$ if $l \neq m$

$f[m](\langle a\rangle \frown as, bs, l) = \langle a\rangle \frown f[m](as, bs, l)$ if $l = m$

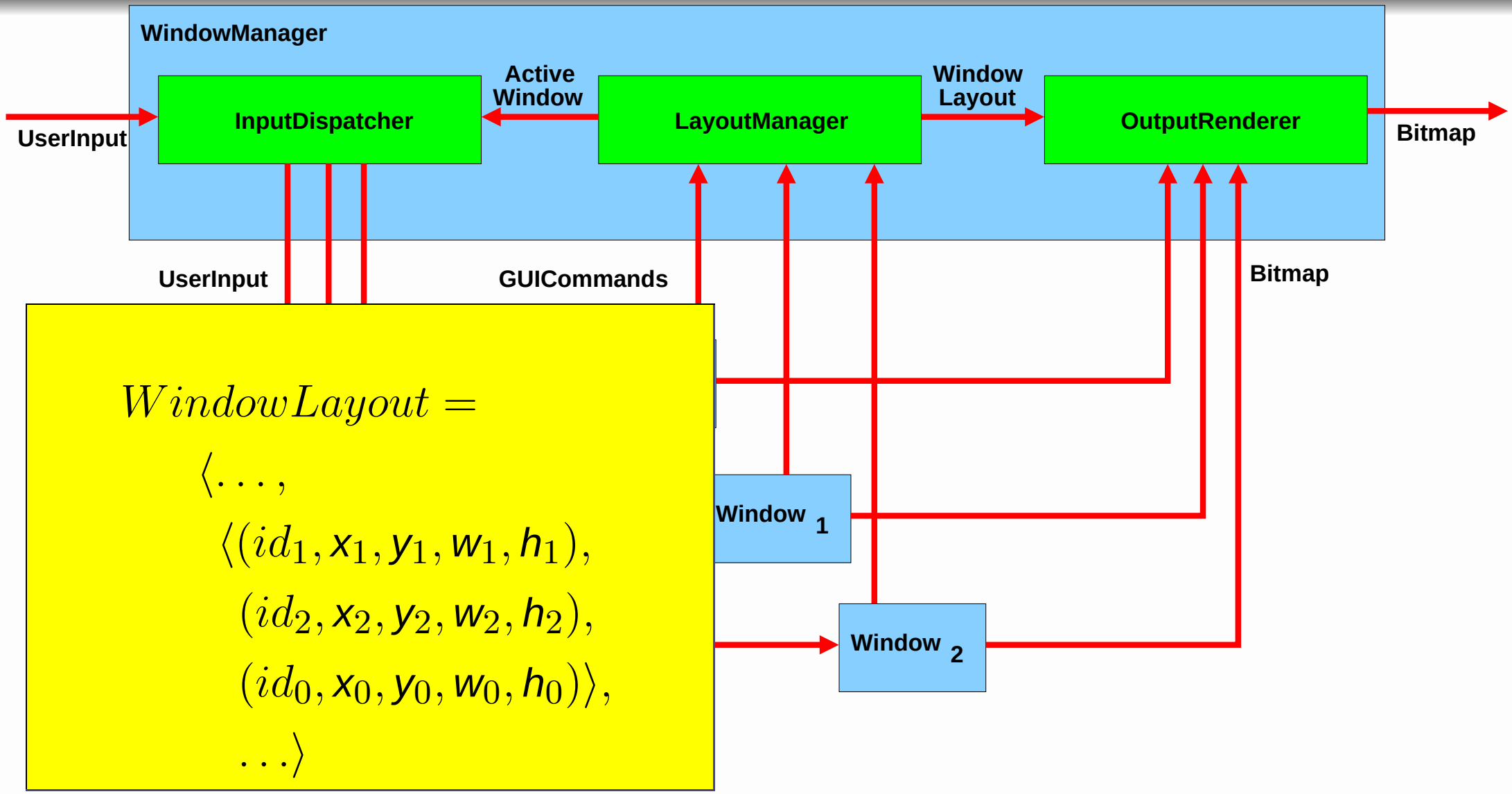
Generic model of multi-window applications



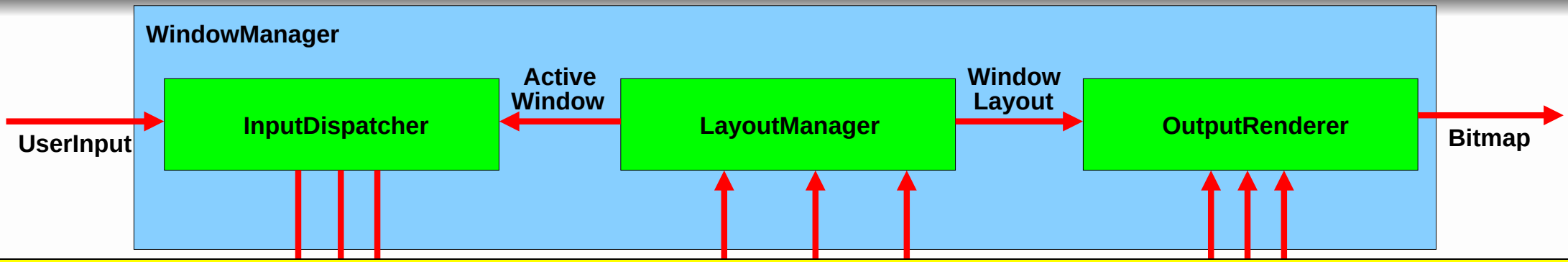
Generic model of multi-window applications



Generic model of multi-window applications



Generic model of multi-window applications

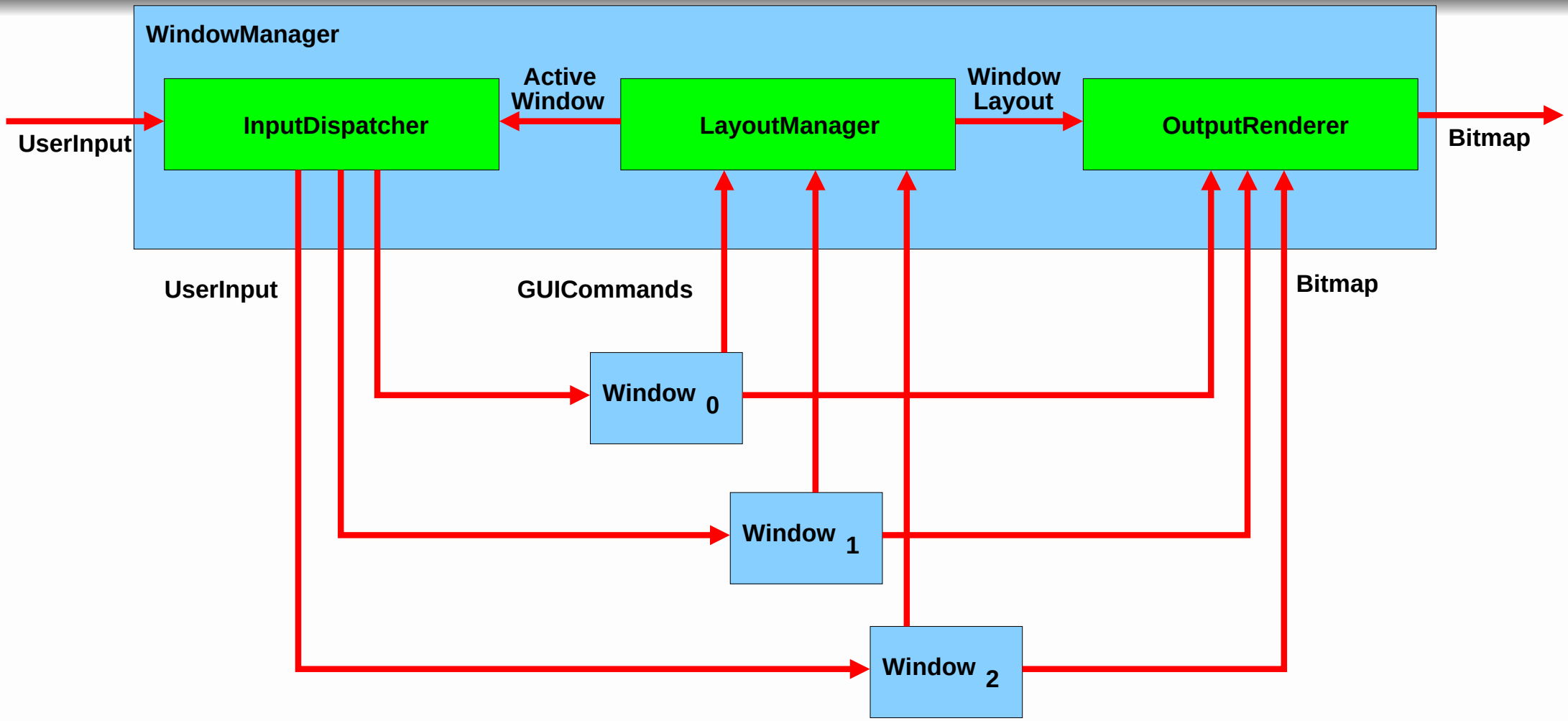


LayoutManager

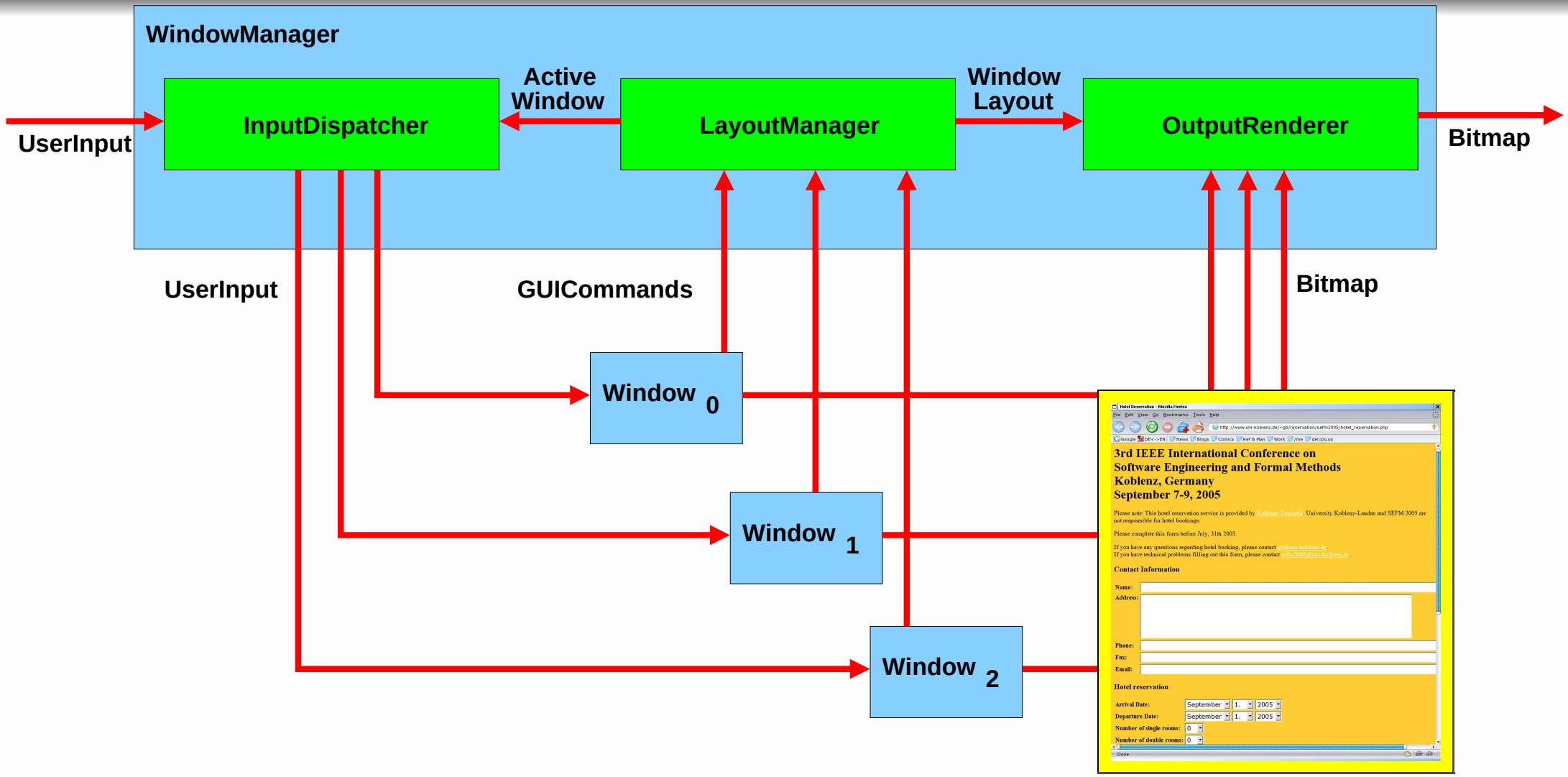
in $i : guiCommand$
out $p : WindowLayout$
out $q : ActiveWindow$

$m[\langle\rangle](i) = t$ with $p = \mathrm{map}(\Pi_0, t);\ q = \mathrm{map}(\Pi_1, t)$
where m so that :
$m[s](\langle \text{raise d} \rangle^\frown i) = (raise(s,d), \langle d \rangle)^\frown m[raise(s,d)](i)$
$m[s](\langle \text{move (d,x,y)} \rangle^\frown i) = (move(s,d,x,y), \langle\rangle)^\frown m[move(s,d,x,y)](i)$
$m[s](\langle \text{resize (d,w,h)} \rangle^\frown i) = (resize(s,d,w,h), \langle\rangle)^\frown m[resize(s,d,w,h)](i)$

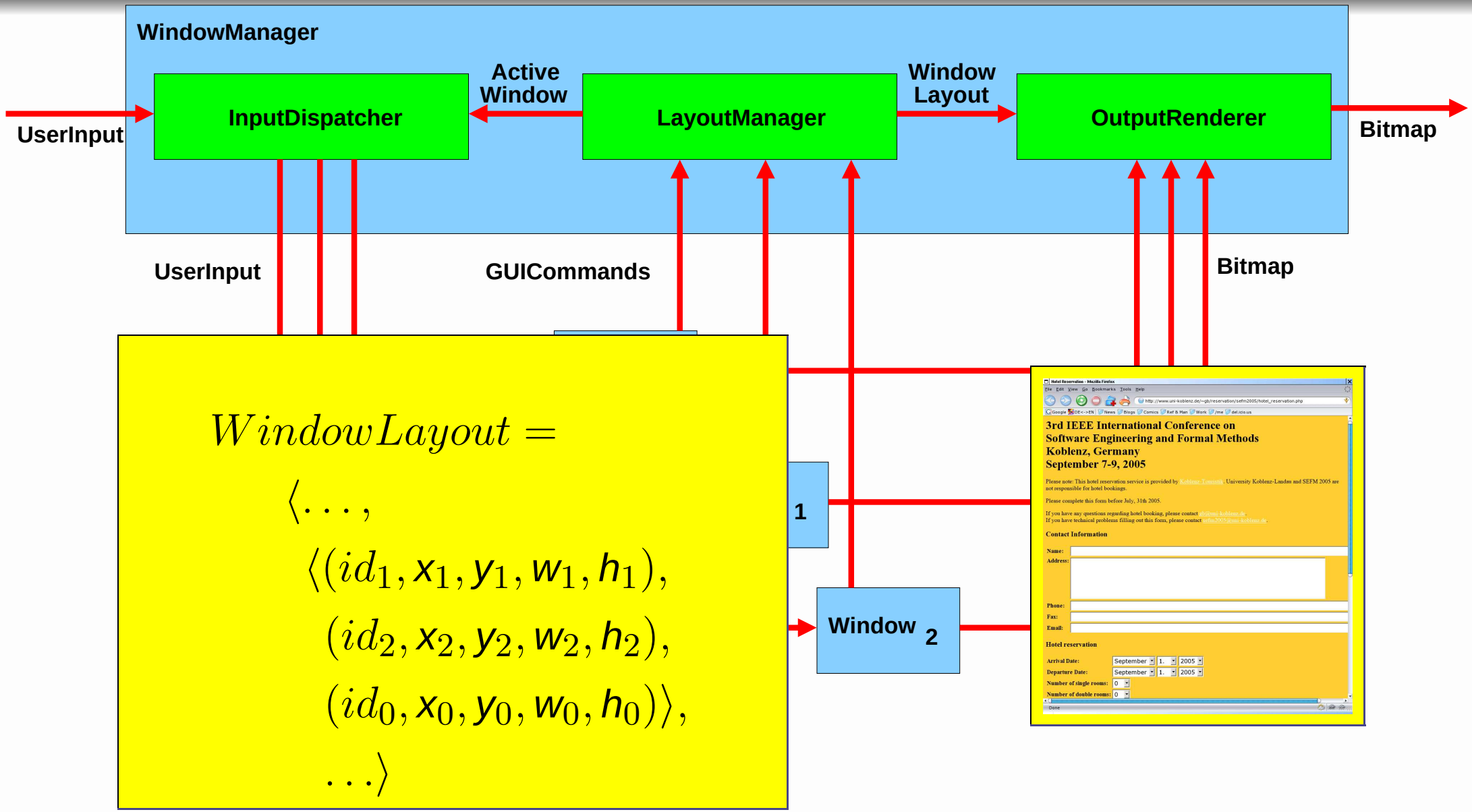
Generic model of multi-window applications



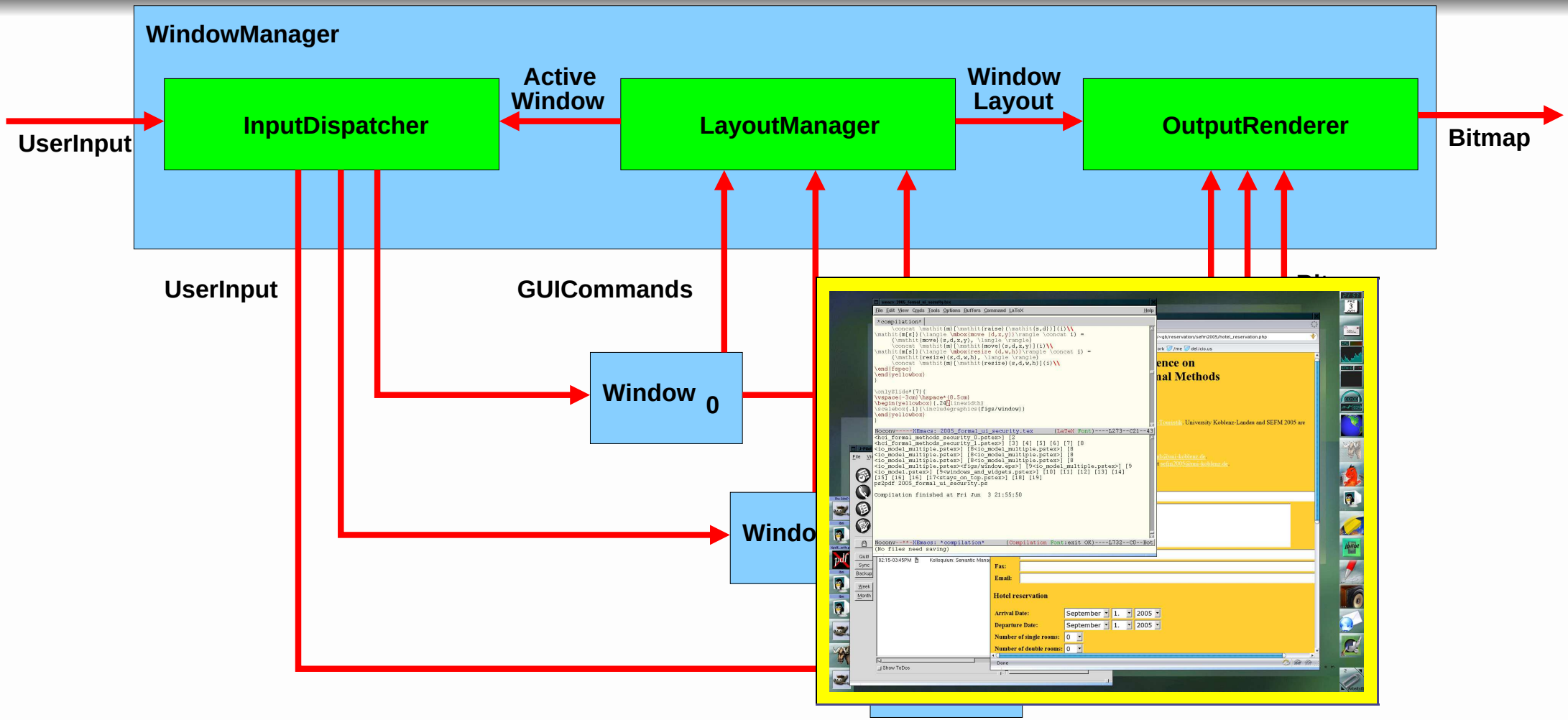
Generic model of multi-window applications



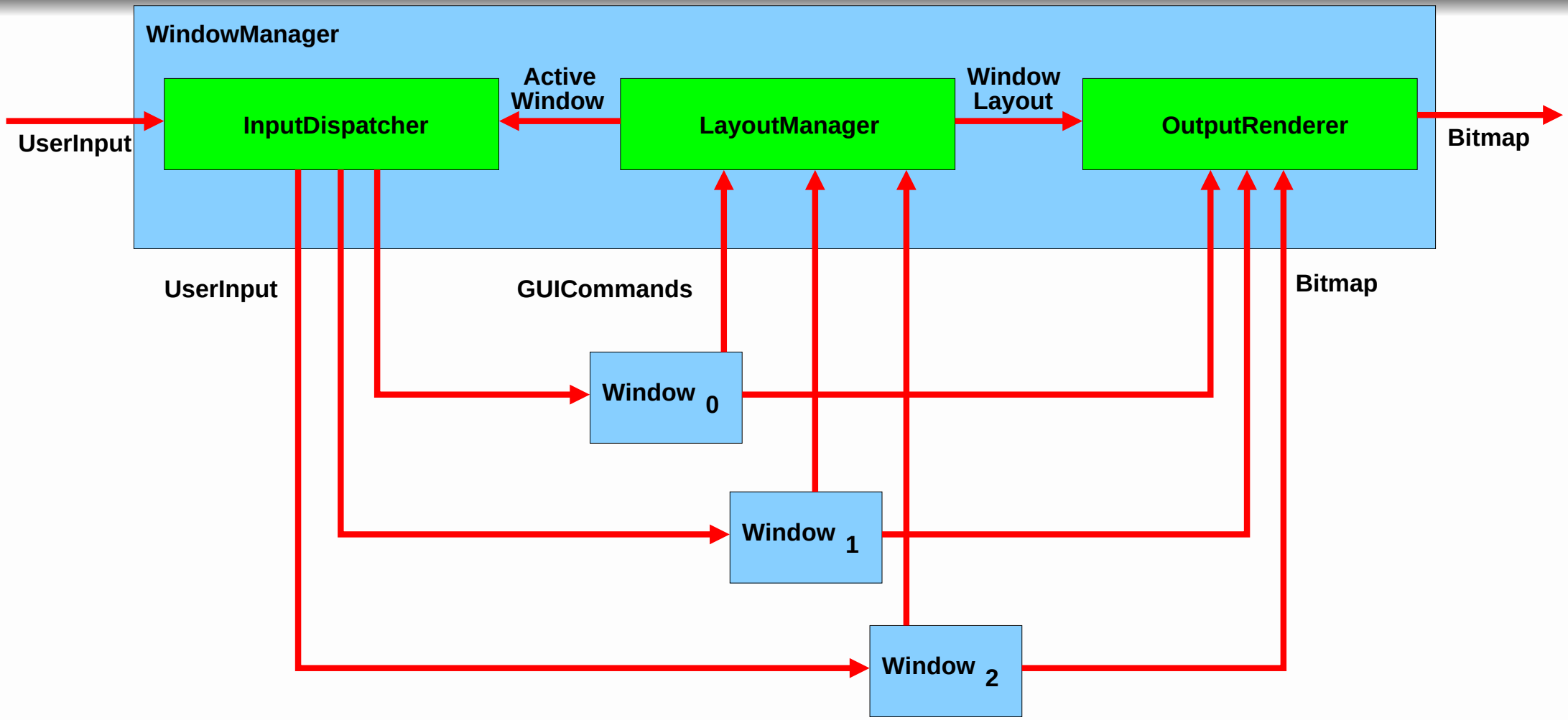
Generic model of multi-window applications



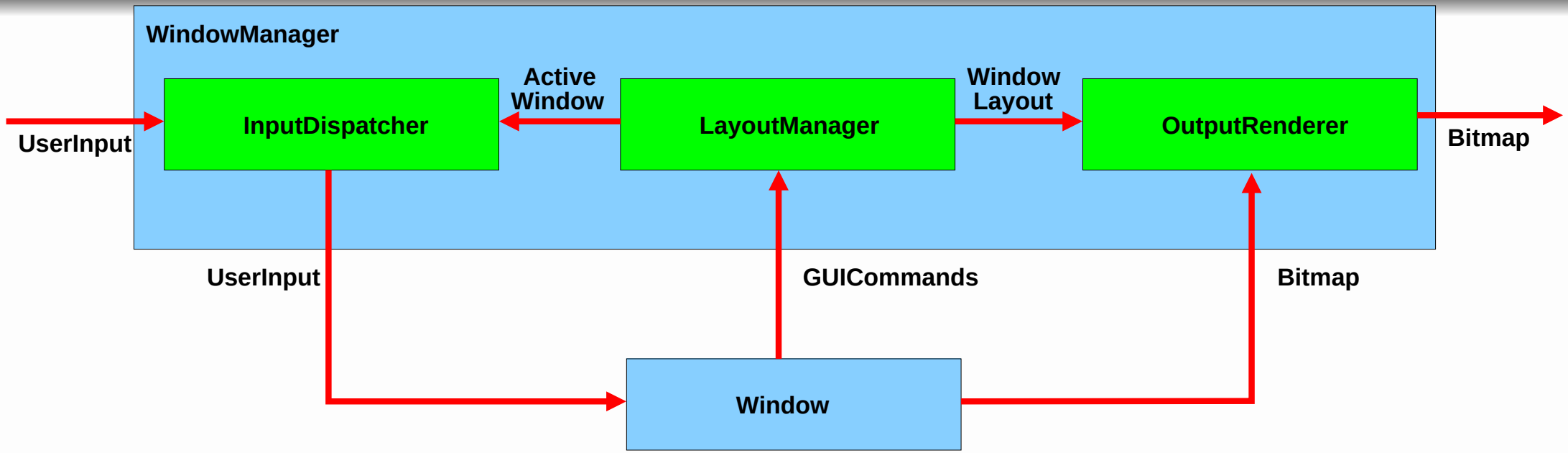
Generic model of multi-window applications



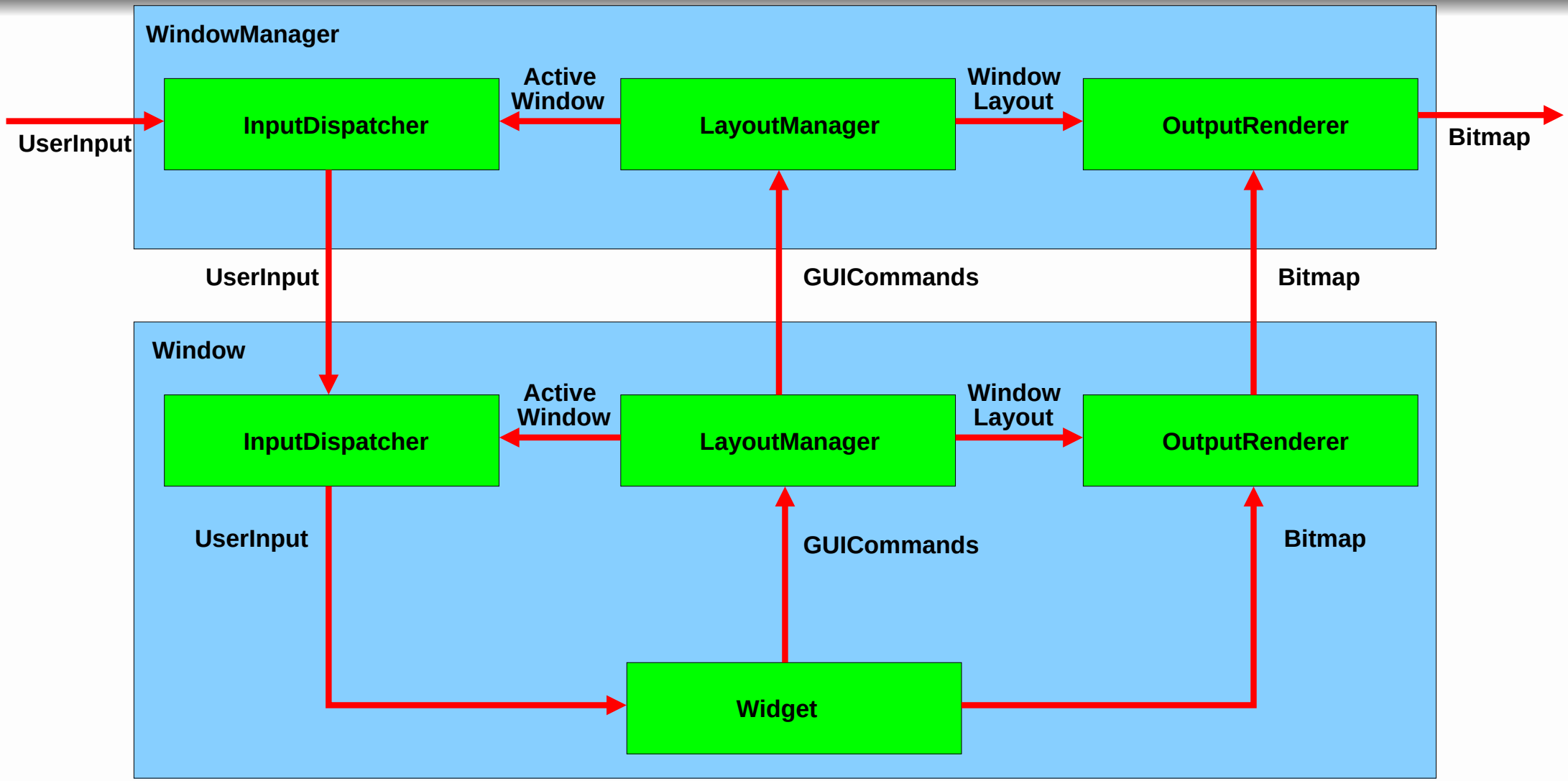
Generic model of multi-window applications



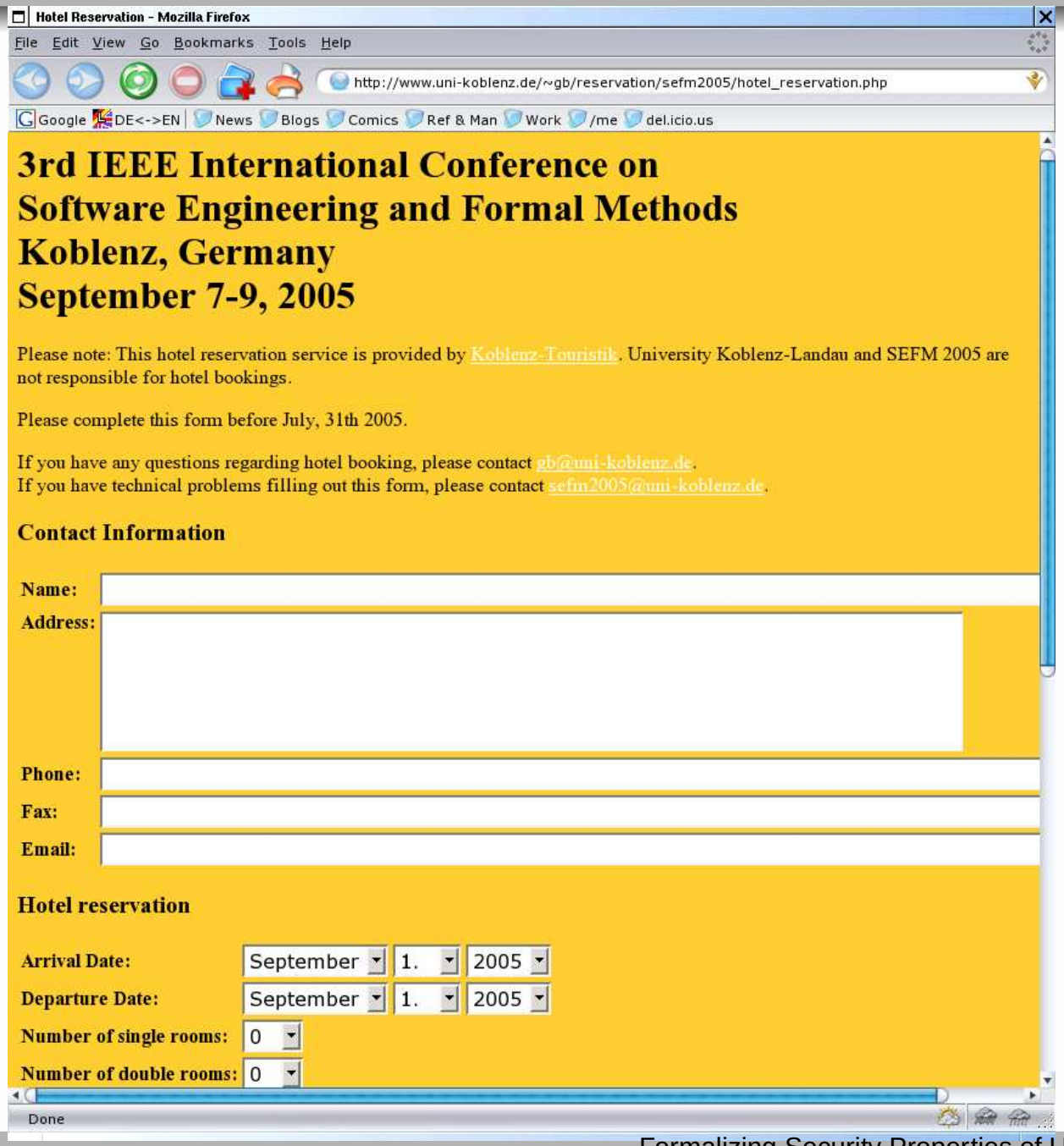
Generic model of multi-window applications



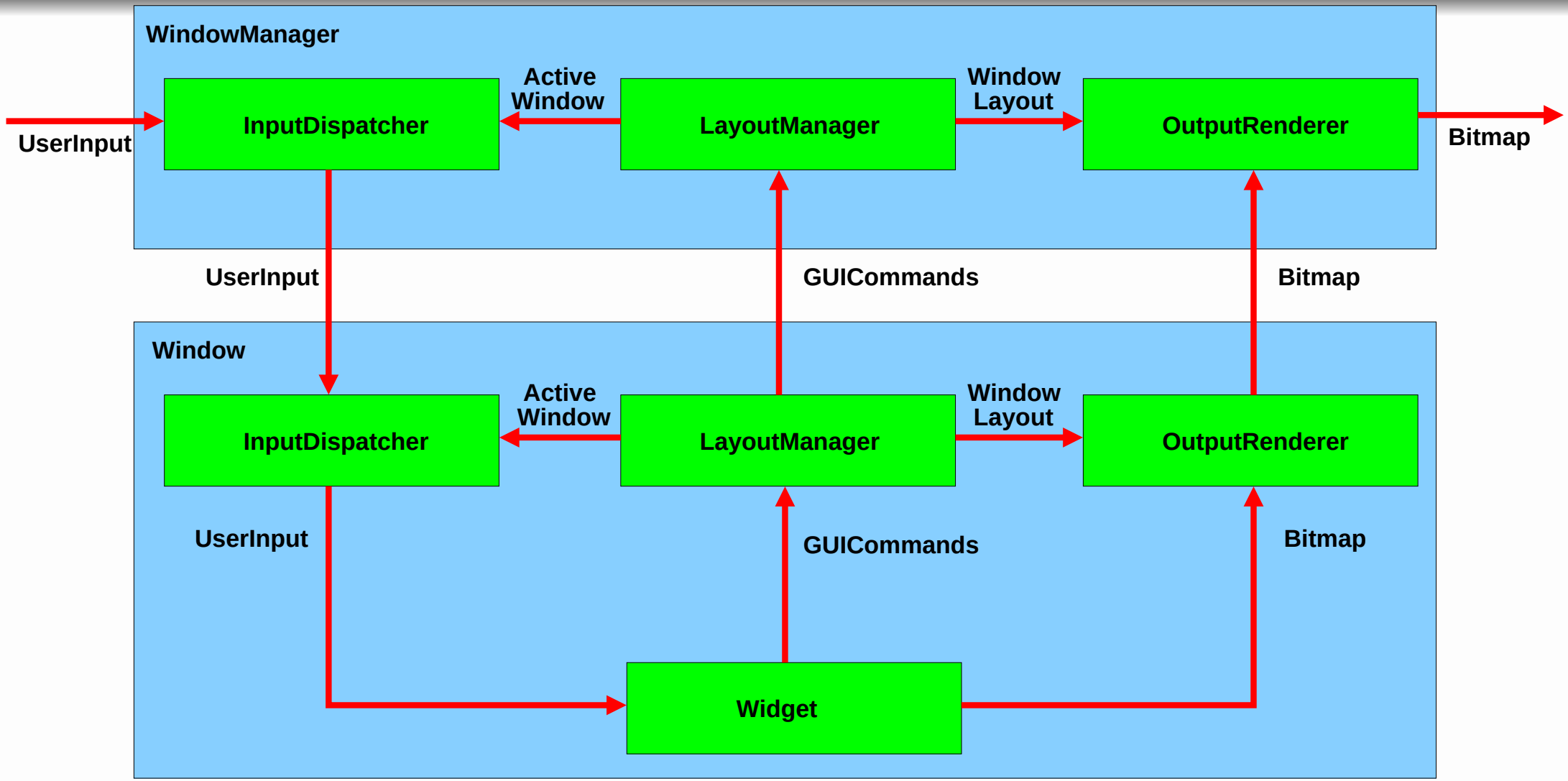
Generic model of multi-window applications



Generic model of multi-window applications



Generic model of multi-window applications



Steps

1. Formalize user interface
   (User Interface $\Longleftrightarrow$ Formal methods)
2. Define requirement for a secure user interface
   (Security $\Longleftrightarrow$ HCI)
3. Formalize UI security requirements
   (Formal methods $\Longleftrightarrow$ Security)
4. $\Longrightarrow$ Formal model of secure user interfaces
5. (User model?)

Steps

1. Formalize user interface
   (User Interface $\Longleftrightarrow$ Formal methods)
2. Define requirement for a secure user interface
   (Security $\Longleftrightarrow$ HCI)
3. Formalize UI security requirements
   (Formal methods $\Longleftrightarrow$ Security)
4. $\Longrightarrow$ Formal model of secure user interfaces
5. (User model?)

Secure User Interface Requirements

Defined by

- Expected Functionality
- System Environment / Attack Scenarios
- Required Security Level

Secure User Interface Requirements

Defined by

- Expected Functionality
- System Environment / Attack Scenarios
- Required Security Level

Required Security Level

- *Provable secure* against attacks
- Compatible to standard security catalogs
- = Allows certification beyond highest standards (EAL7+)

Secure User Interface Requirements

Defined by

- Expected Functionality
- System Environment / Attack Scenarios
- Required Security Level

Attack Scenarios

- Standardized catalogs
- Application specific attacks

Secure User Interface Requirements

Defined by

- Expected Functionality
- System Environment / Attack Scenarios
- Required Security Level

System Environment

- Physical secure system
- Multi-user, multi-tasking computer systems
- Part of I/O data may come from third parties

Expected Functionality

Generic requirements

- No eavesdropping on I/O devices
- Allow to place constraints on I/O behavior
  - Input possible only before/after certain events
  - Restrict access to screen areas/windows
  - Enforce properties of output data (font, size, color, no mimikry, stays on top. . .)

Steps

1. Formalize user interface
   (User Interface $\Longleftrightarrow$ Formal methods)
2. Define requirement for a secure user interface
   (Security $\Longleftrightarrow$ HCI)
3. Formalize UI security requirements
   (Formal methods $\Longleftrightarrow$ Security)
4. $\Longrightarrow$ Formal model of secure user interfaces
5. (User model?)

Steps

1. Formalize user interface
   (User Interface $\Longleftrightarrow$ Formal methods)
2. Define requirement for a secure user interface
   (Security $\Longleftrightarrow$ HCI)
3. Formalize UI security requirements
   (Formal methods $\Longleftrightarrow$ Security)
4. $\Longrightarrow$ Formal model of secure user interfaces
5. (User model?)

Expected Functionality

Constraining I/O behavior

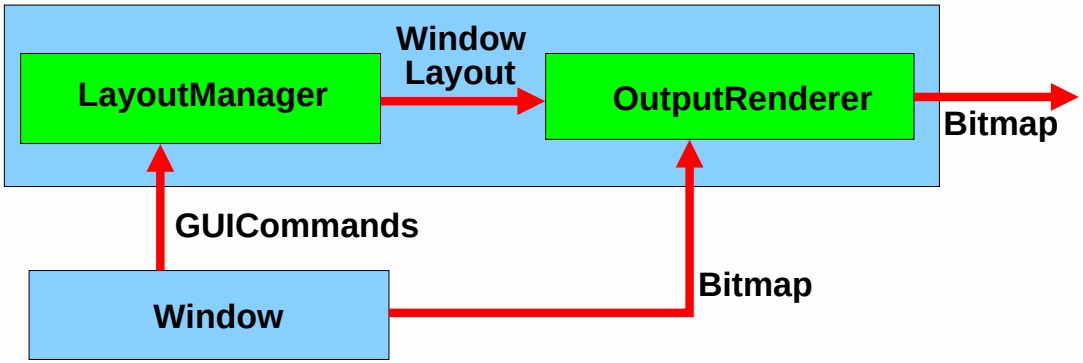
- Input possible only before/after certain events
- Restrict access to screen areas/windows
- Enforce properties of output data (font, size, color, no mimikry, stays on top …)

Expected Functionality

Constraining I/O behavior

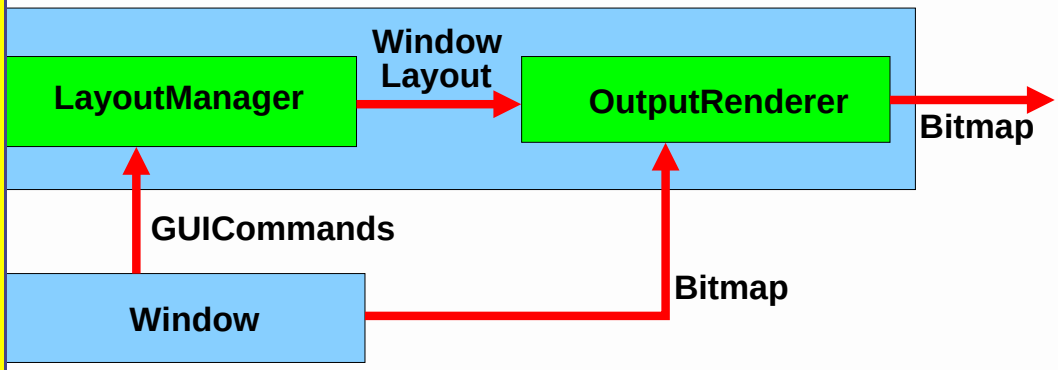
- Input possible only before/after certain events
- Restrict access to screen areas/windows
- Enforce properties of output data (font, size, color, no mimikry, stays on top …)

Example: Stays on Top



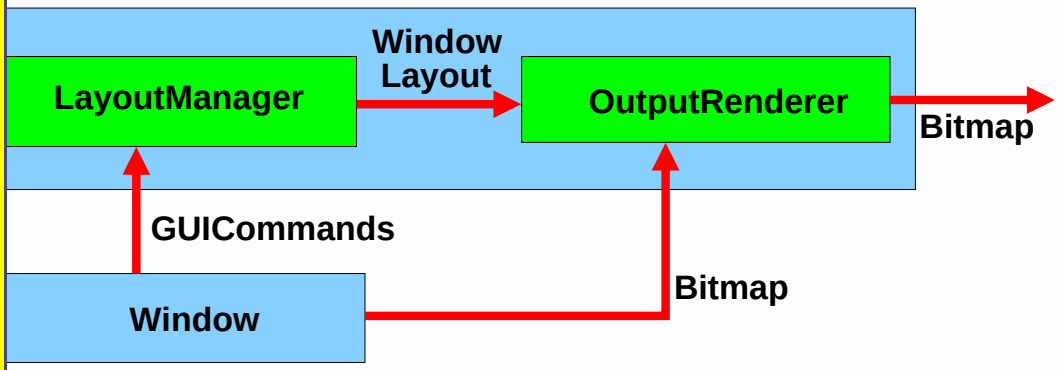
Example: Stays on Top

$$
\begin{aligned}
&WindowLayout = \\
&\quad \langle \ldots, \\
&\quad\quad \langle (id_1, x_1, y_1, w_1, h_1), \\
&\quad\quad\quad (id_2, x_2, y_2, w_2, h_2), \\
&\quad\quad\quad (id_0, x_0, y_0, w_0, h_0) \rangle, \\
&\quad\quad\quad \ldots \rangle
\end{aligned}
$$



Example: Stays on Top

$$WindowLayout = \langle \ldots, \langle (id_1, x_1, y_1, w_1, h_1), (id_2, x_2, y_2, w_2, h_2), (id_0, x_0, y_0, w_0, h_0) \rangle, \ldots \rangle$$



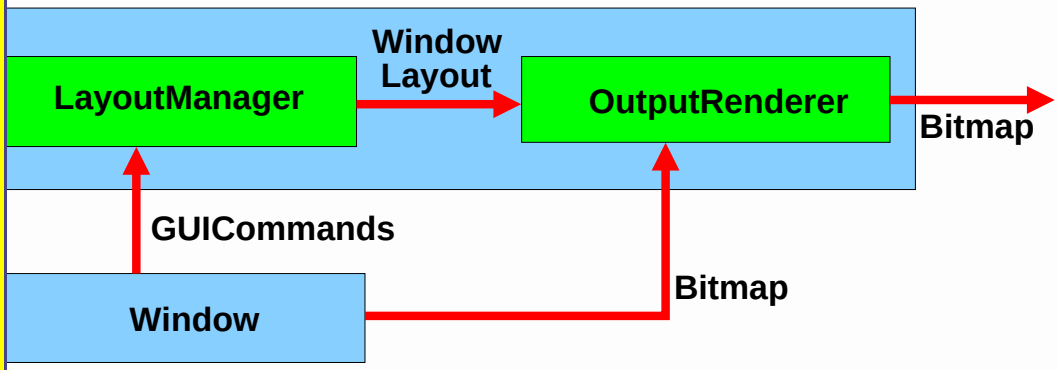
$\text{LayoutManager}'$

in $i : guiCommand$
out $p' : WindowLayout$
out $q : ActiveWindow$

$$\text{LayoutManager}(i, p, q) \wedge (\Pi_0.\text{ft}.p' = id_1) \wedge (\text{rt}.p' = \text{map}(p, (\lambda e.\text{if } e = id_1 \text{ then } \langle\rangle \text{ else } e)))$$

Example: Stays on Top

$$WindowLayout = \langle \ldots, \langle (id_1, x_1, y_1, w_1, h_1), (id_2, x_2, y_2, w_2, h_2), (id_0, x_0, y_0, w_0, h_0) \rangle, \ldots \rangle$$



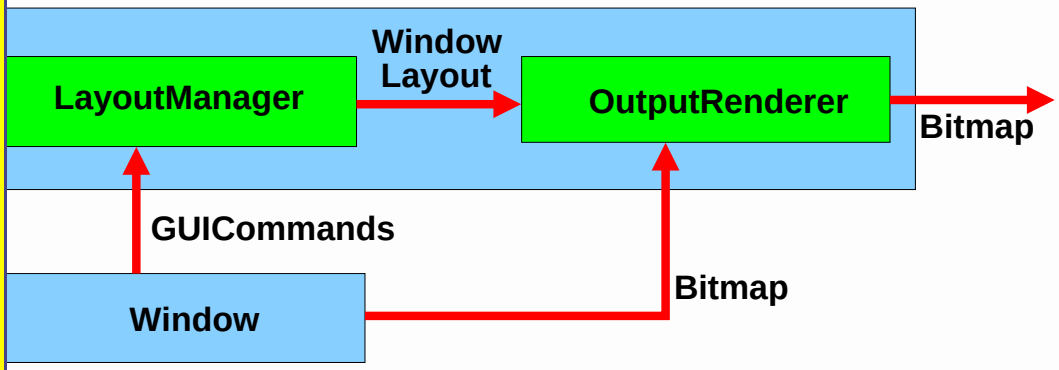
$\text{LayoutManager}'$

in $i : guiCommand$
out $p' : WindowLayout$
out $q : ActiveWindow$

$\text{LayoutManager}(i, p, q) \wedge (\text{ft}.p' = (id_1, 0, 0, 800, 200)$
$\wedge (\text{rt}.p' = \text{map}(p, (\lambda e.\text{if } e = id_1 \text{ then } \langle\rangle \text{ else } e)))$

Example: Stays on Top

$$WindowLayout = \langle \ldots, \langle (id_1, x_1, y_1, w_1, h_1), (id_2, x_2, y_2, w_2, h_2), (id_0, x_0, y_0, w_0, h_0) \rangle, \ldots \rangle$$



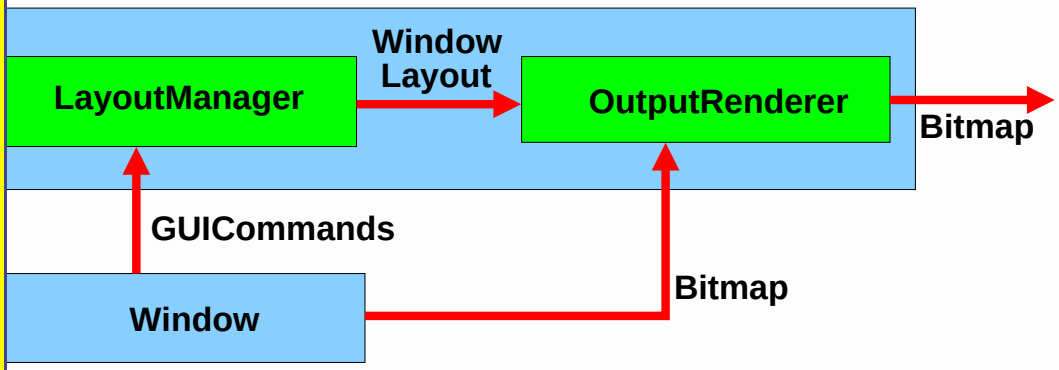
$\text{LayoutManager}'$

in $i : guiCommand$
out $p' : WindowLayout$
out $q : ActiveWindow$

$$\forall x, y \in p' : overlap(x, y) \rightarrow x = y$$

Example: Stays on Top

$$WindowLayout = \langle \ldots, \langle (id_1, x_1, y_1, w_1, h_1), (id_2, x_2, y_2, w_2, h_2), (id_0, x_0, y_0, w_0, h_0) \rangle, \ldots \rangle$$



OutputRenderer′

in $i : Bitmap$

out $o : Bitmap$

$$\forall x, y, x', y' \in o : adjacent((x, y), (x', y')) \rightarrow (contrast(o(x, y), o(x', y')) = 0) \vee (contrast(o(x, y), o(x', y')) > minContrast)$$

Steps

1. Formalize user interface
   (User Interface $\Longleftrightarrow$ Formal methods)
2. Define requirement for a secure user interface
   (Security $\Longleftrightarrow$ HCI)
3. Formalize UI security requirements
   (Formal methods $\Longleftrightarrow$ Security)
4. $\Longrightarrow$ Formal model of secure user interfaces
5. (User model?)

Steps

1. Formalize user interface
   (User Interface $\Longleftrightarrow$ Formal methods)
2. Define requirement for a secure user interface
   (Security $\Longleftrightarrow$ HCI)
3. Formalize UI security requirements
   (Formal methods $\Longleftrightarrow$ Security)
4. $\Longrightarrow$ Formal model of secure user interfaces
5. (User model?)

Applying it to the real world...

Verisoft Email Client

Formalizing parts of the Common Criteria

Simple window manager for capability-based systems

Steps

1. Formalize user interface
   (User Interface $\Longleftrightarrow$ Formal methods)
2. Define requirement for a secure user interface
   (Security $\Longleftrightarrow$ HCI)
3. Formalize UI security requirements
   (Formal methods $\Longleftrightarrow$ Security)
4. $\Longrightarrow$ Formal model of secure user interfaces
5. (User model?)

Steps

1. Formalize user interface
   (User Interface $\Longleftrightarrow$ Formal methods)
2. Define requirement for a secure user interface
   (Security $\Longleftrightarrow$ HCI)
3. Formalize UI security requirements
   (Formal methods $\Longleftrightarrow$ Security)
4. $\Longrightarrow$ Formal model of secure user interfaces
5. (User model?)