

# Toy-Window API Referenz

28. Juni 2006

Dieses Dokument beschreibt die API der Implementierung des ConcurrentML Toy-Window-Systems [1] für Scheme 48 und Scsh.

## 1 Modul toy-geometry

- `(make-point x y) => point`  
stellt einen Wert her, der einen Punkt oder einen Vektor repräsentiert. Der Ursprung ist dabei in der oberen linken Ecke, die Y-Achse wächst nach unten.
- `(point+ point1 point2) => point`  
addiert zwei Punkte koordinatenweise.
- `(point- point1 point2) => point`  
subtrahiert zwei Punkte koordinatenweise.
- `(point<? point1 point2) => boolean`  
vergleicht zwei Punkte koordinatenweise.
- `(make-rectangle x y width height) => rectangle`  
stellt einen Wert her, der ein Rechteck repräsentiert.
- `(rectangle-origin rectangle) => point`  
liefert den Ursprung des Rechteck.
- `(rectangle-x rectangle) => int`  
liefert die X-Koordinate des Ursprungs des Rechtecks.
- `(rectangle-y rectangle) => int`  
liefert die Y-Koordinate des Ursprungs des Rechtecks.
- `(rectangle-width rectangle) => int`  
liefert die Breite des Rechtecks.
- `(rectangle-height rectangle) => int`  
liefert die Höhe des Rechtecks.
- `(point-in-rectangle? point rectangle) => boolean`  
testet, ob ein Punkt innerhalb eines Rechtecks liegt.
- `(rectangle+ rectangle point) => rectangle`  
liefert ein Rechteck, das gegenüber `rectangle` um `point` verschoben ist.

- `(rectangle- rectangle point) => rectangle`  
liefert ein Rechteck, das gegenüber `rectangle` um `point` verschoben ist.
- `(rectangle-within rectangle1 rectangle2) => rectangle`  
liefert ein Rechteck der Größe von `rectangle2`, das ggf. verschoben ist, um so gut wie möglich innerhalb von `rectangle1` zu passen.

## 2 Modul toy-display-system

### 2.1 Bitmaps

Ein Parameter mit dem Namen `raster-op` bezeichnet eine logische Verknüpfungsoperation zwischen den Pixeln der Quelle und des Ziels. Die möglichen Operationen werden durch die Symbole `copy`, `xor`, `or`, `and`, `clear` und `set`.

- `(make-bitmap bitmap rectangle) => bitmap`  
erzeugt eine neue Bitmap auf dem Schirm, die ein Kind von `bitmap` ist und sich im Rechteck `rectangle` (relativ zu `bitmap`) befindet. Die Bitmap ist dabei zunächst unter den Kindern von `bitmap` (den „Geschwistern“) ganz vorn.
- `(bitmap-to-front! bitmap)`  
rückt `bitmap` unter den Geschwistern ganz nach vorn.
- `(bitmap-to-back! bitmap)`  
rückt `bitmap` unter den Geschwistern ganz nach hinten.
- `(bitmap-move! bitmap point)`  
verschiebt `bitmap` (relativ) um `point`.
- `(bitmap-delete! bitmap)`  
löscht die Bitmap `bitmap`.
- `(bitmap-clear bitmap)`  
löscht den Inhalt von `bitmap`.
- `(bitmap-draw-line bitmap raster-op point1 point2)`  
zeichnet eine Linie von `point1` nach `point2`.
- `(bitmap-draw-rectangle bitmap raster-op rectangle)`  
zeichnet ein Rechteck, das gegenüber `rectangle` nach rechts und unten um einen Punkt größer ist.
- `(bitmap-fill-rectangle bitmap raster-op texture rectangle)`  
füllt ein Rechteck der Größe von `rectangle` mit der Textur `texture`. Die (für den Moment) einzig sinnvolle Textur ist `texture-solid`.
- `(bitmap-bitblt bitmap1 raster-op point1 bitmap2 rectangle1)`  
kopiert ein Rechteck aus `bitmap1` der Größe von `rectangle1` nach `bitmap2` an die Stelle `point1`. Quelle und Ziel werden mit `raster-op` verknüpft.
- `(bitmap-draw-text bitmap raster-op point text)`  
malt den Text `text` an die Stelle `point`. Dabei ist `point` links und auf der Höhe der Basislinie.
- `(bitmap-text-size bitmap text) => width height ascent`  
ermittelt die Größe des Textes `text`. Dabei ist `ascent` die Höhe des Textes oberhalb der Basislinie.

## 2.2 Initialisierung

- `(make-display label width height) => bitmap mouse-channel keyboard-channel`  
initialisiert das Display-System, erzeugt ein X-Fenster der Breite `width` und der Höhe `height` mit Titel `label` und liefert eine Bitmap für das gesamte Fenster sowie synchrone Kanäle für Maus- und Tastatur-Ereignisse zurück.

## 2.3 Maus-Messages

- `(mouse-message-up? message) => boolean`  
stellt fest, ob der Mausknopf oben ist.
- `(mouse-message-down? message) => boolean`  
stellt fest, ob der Mausknopf unten ist.
- `(mouse-message-position message) => point`  
liefert die Position der Maus zurück.
- `(mouse-message-translate point message) => mouse-message`  
transformiert die Positionskordinaten von `message` relativ zu `point`

## 2.4 Tastatur-Messages

- `(key-message-down? message) => boolean`  
stellt fest, ob die Taste unten ist.
- `(key-message-keycode message) => integer`  
liefert den X11-Keypcode der Taste.
- `(key-message-string message) => string`  
liefert den String für die gedrückte Taste oder einen leeren String für Steuertasten.

## 3 Modul toy-window-system

- `(make-widget-env bitmap mouse-channel keyboard-channel control-in-channel control-out-channel) => widget-env`  
liefert ein Widget-Environment. `Control-out-channel` ist ein asynchroner Kanal, die restlichen Kanäle sind synchron.
- `(make-vanilla-widget-env bitmap) => widget-env`  
erzeugt ein Widget-Environment mit frischen Kanälen.
- `(widget-env-bitmap widget-env) => bitmap`  
liefert die Bitmap eines Widget-Environments.
- `(widget-env-mouse-channel widget-env) => channel`  
liefert den Maus-Kanal eines Widget-Environments.
- `(widget-env-keyboard-channel widget-env) => channel`  
liefert den Tastatur-Kanal eines Widget-Environments.
- `(widget-env-control-in-channel widget-env) => channel`  
liefert den Kontroll-Eingangs-Kanal eines Widget-Environments.

- `(widget-env-control-out-channel widget-env) => async-channel`  
liefert den Kontroll-Ausgangs-Kanal eines Widget-Environments.
- `(make-channel-sink channel)`  
stellt einen Abfluß für den synchronen Kanal `channel` her, der alle ankommenden Werte abholt und wegwirft.
- `(realize-widget bitmap rectangle realization-proc) => value`  
erzeugt ein frisches Widget-Environment innerhalb der Eltern-Bitmap `bitmap` an Position `rectangle` und übergibt dieses an `realization-proc`. Dessen Rückgabewert gibt auch `realize-widget` zurück.
- `(make-window-system label width height) => widget-env`  
initialisiert das Window-System (ruft also `make-display` auf), erzeugt ein initiales Widget-Environment für das gesamte Fenster und gibt dieses zurück.
- `(control-message? thing) => boolean`  
Überprüft ob `thing` ein Control-Message-Record ist.
- `(control-message-type ctrl-msg) => symbol`  
Liefert den Typ einer Kontroll-Nachricht als Symbol zurück. Da es bisher nur einen Typ Kontroll-Nachrichten gibt (für das Löschen eines Widgets), gibt diese Funktion immer `'delete` zurück.

## 4 Modul `toy-button`

- `(make-button label thunk widget-env) => widget-env`  
stellt innerhalb von `widget-env` einen Knopf mit Beschriftung `label` her; bei Betätigung wird `thunk` aufgerufen.

## 5 Modul `toy-frame`

- `(make-frame realize widget-env) => frame value`  
malt innerhalb von `widget-env` einen Rahmen um ein anderes Widget, an das es alle Nachrichten weitergibt. Das andere Widget muß von `realize` erzeugt werden, das als Parameter ein eigenes Widget-Environment übergeben bekommt. Zurückgegeben werden ein Wert, der das Frame repräsentiert, sowie der Rückgabewert von `realize`.

## Literatur

- [1] John H. Reppy. *Concurrent Programming in ML*. Cambridge University Press, 1999. 1