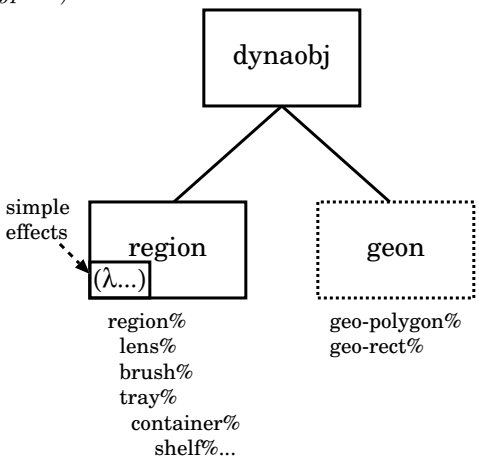
Dynapad Region Tools

Dan Bauer
UCSD Cognitive Science
HCI & Distributed Cogntion Lab

6/25/03

1 Basic Regions

Any dynaobject% may be turned into a region:
(*regionize dynaobj regiontype%*)



This attaches two actors:

The *geon* actor represents the dynaobj's boundary polygon (for intersection, containment, etc.).

The *region* actor determines two properties:

- *Kinetics*: how objects enter, leave, and follow the region. Kinetics depend on the subclass of region% (e.g. lens%, tray%, etc.)
- *Effect*: what happens to objects in the region. In a simple region, the effect indetermined by the region's "effect-lambdas".

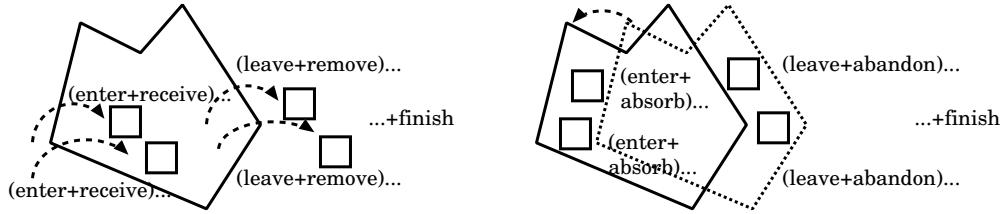
An effect is distributed over several effect-lambdas of these types:

- Enter: called whenever contents enter region (Recieve or Absorb)
  - Receive: called only when contents move into region

  - Absorb: called only when region moves onto content

- Leave: called whenever contents leave region (Remove or Abandon)
  - Remove: called only when contents move out of region
  - Abandon: called only when region moves away from contents

- Finish: called once after all Enter/Leaves



Moving Contents **Moving Region**

Receive/Remove lambdas are used only when object move to/from the region, and Absorb/Abandon lambdas are used only when the regions moves onto/off the objects. An Enter (+ Receive or Absorb) is called for every entering object; a Leave (+ Remove or Abandon) is called for every leaving object. Finally, a single Finish is called.

1.1 Example: magnify-region

```
(define magnify-region%
  (class region%
    (init _obj)
    (super-instantiate (_obj))

     (send this enter-action
           (lambda (obj) (send obj scale 2)))
     (send this leave-action
           (lambda (obj) (send obj scale .5)))))

(regionize poly magnify-region%)
```

1.2 Example: Undoable Effects

Many effects will need to save the state of an object when it enters so that it can be restored when it leaves. In such cases, there must be an interaction between the enter and leave lambdas. To faciliatate such "undoable" ops, you can include an *(undo-on-exit)* function in the enter lambda. The following example saves an object's pen color and restores in on exit:

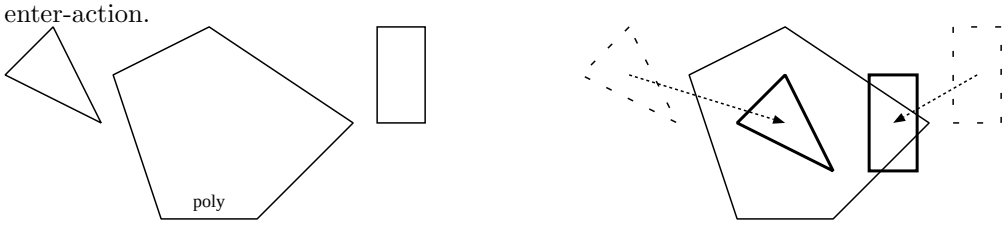
```
(define blueify-region%
  (class region%
```

```
    (init _obj)
    (super-instantiate (_obj))

    (remote-push!
      (lambda (obj) (let* ((old-color (send obj pen))
                           (do-fn  (lambda (obj) (send obj pen "blue")))
                           (undo-fn (lambda (obj) (send obj pen old-color))))
                      (send this undo-on-exit obj do-fn undo-fn)))
      this enter-action)
))

(regionize poly blueify-region%)
```

Note that no explicit leave-action is needed, since it is created on-the-fly by the enter-action.



1.3 Lenses

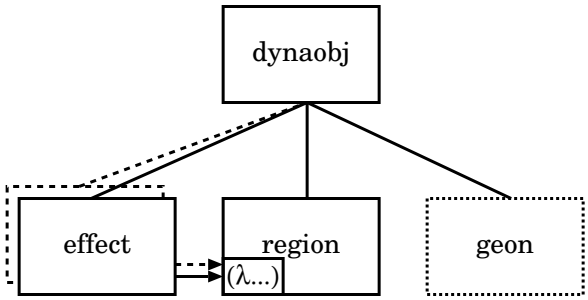
An undoable effect makes the region a kind of "lens" in whose effect only applies while an object is inside the region. But this gets confusing sometimes. A better implementation of lenses is the *lens%* region subclass. It works by copying all its contents and applying the effect to the copy instead of the original. A *lens%* automatically deletes all copies when the original leaves, so an custom leave-action (to undo the effect) isn't needed.

```
(define blueify-lens%
  (class lens%
    ...
    (remote-push! (lambda (obj) (send obj pen "blue"))
                  this enter-action)
))

(regionize poly blueify-lens%)
```

2 Effect "Friends"

When effects are more complicated, it may be better to offload the region's effect into a "friend" object:



Most of the operations needed at Enter or Leave can be methods of the effect object, which simply registers "trigger" lambdas with the region.

For example, the *projector%* below works with a helper object *cells* to produce a layout effect:

```
(define projector%
  (class named-actor%
  (init  region)
  (field cells)  ;helper obj
  ...
  ;register triggers with region
  (remote-push! (lambda (obj) (send cells assimilate obj))
                region enter-action)
  (remote-push! (lambda (obj) (send cells remove obj))
                region leave-action)
  (remote-push! (lambda (obj) (send this render))
                region finish-action)
  ...
))

(make-object projector% (regionize poly lens%))
```