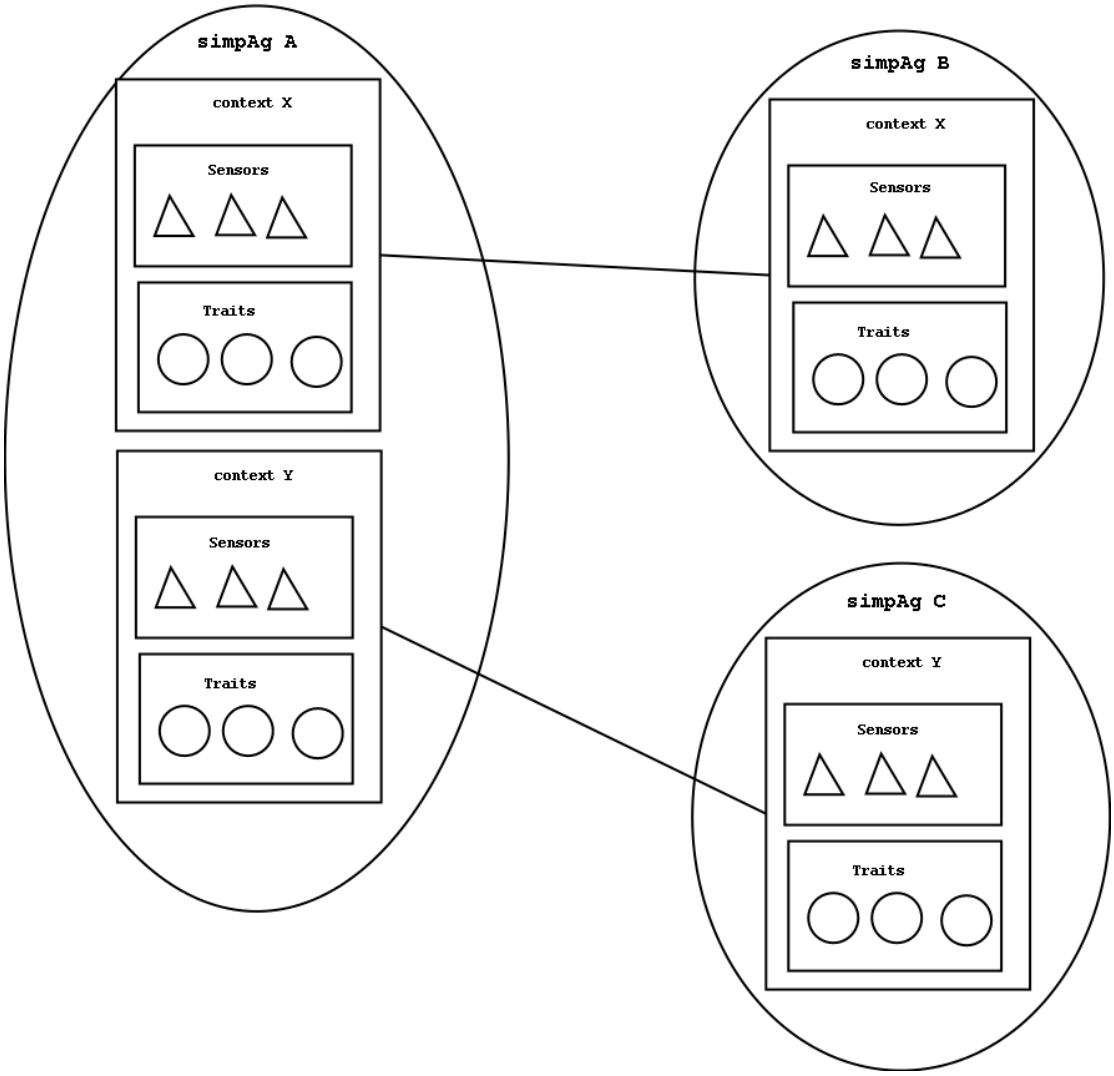
Constructing Federations from Simple Agents (simpAgs) for Pengi World

Dmitri O. Kondratiev, Vasily V. Suvorov

R&D, Luxoft, 9-b Dmitrovskoye shosse
Moscow 127434, Russia.

dkondratiev@luxoft.com, vsuvorov@luxoft.com

simpAgs, Contexts and Traits



simpAgs Framework

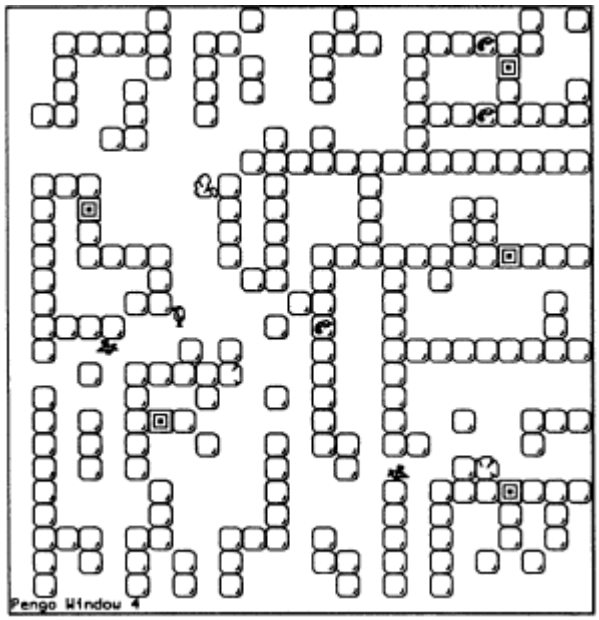
- builds agent federations according to different decomposition principles from simple agents or **simpAgs**
- hierarchies of recursive and self-similar agents - holons
- simpAg provides **live environment context**

simpAg building blocks:

- sensors
- traits - properties visible in context
- evolvable behaviors – alow simpAgs to learn

Modeling Pengi world with simpAgs

Pengo Game



Pengi simpAg Federation controls penguin in Pengo Game according to **Deictic representation** principles, first presented by Philip E. Agre in his work "The Dynamic Structure of Everyday Life" (Artificial Intelligence Laboratory, Cambridge, MA).

Deictic representation: "individuate things in an agent's world ***indexically*** – in relation to the agent's body and identity – and ***functionally*** – in relation to the agent's ongoing goals and projects" (P. Agre)

Pengi can focus on its environment with deictic entities implemented with **simpAgs sensors and traits**:

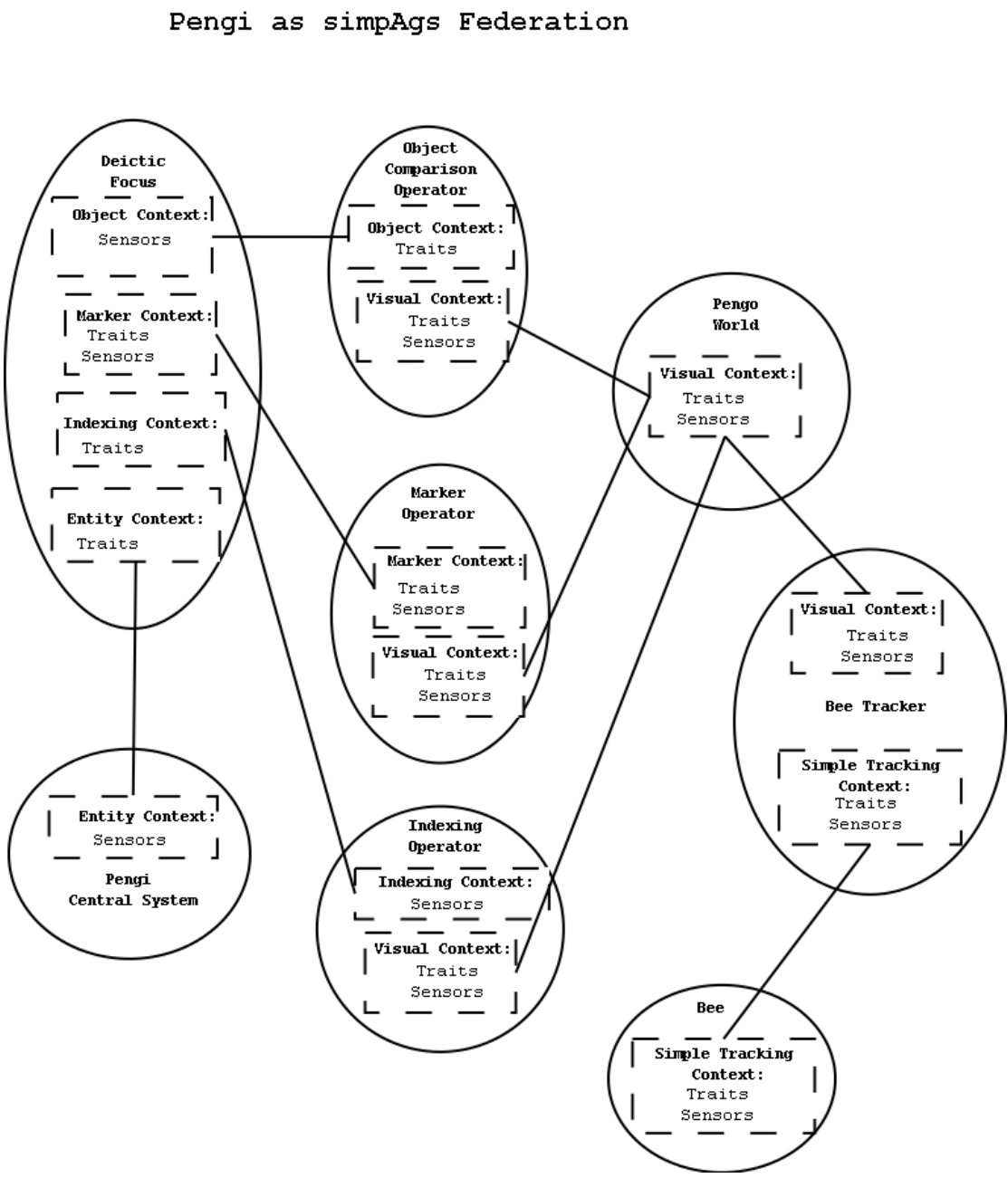
- The-ice-cube-I-am-kicking
- The-direction-I-am-headed-in
- The-bee-I-am-attacking
- The-bee-on-the-other-side-of-this-ice-cube-next-to-me

Entities can have aspects:

- Is-running-away-from (entity: The-bee-I-am-attacking)
- Is-closer-to
- Is-moving-away-from

Pengi Deictic Focus simpAg provides many contexts:

- Deictic Entity context
- Marker operator contexts
- Indexing operator contexts
- Object comparison operator context



Pengi Visual System

Consists from:

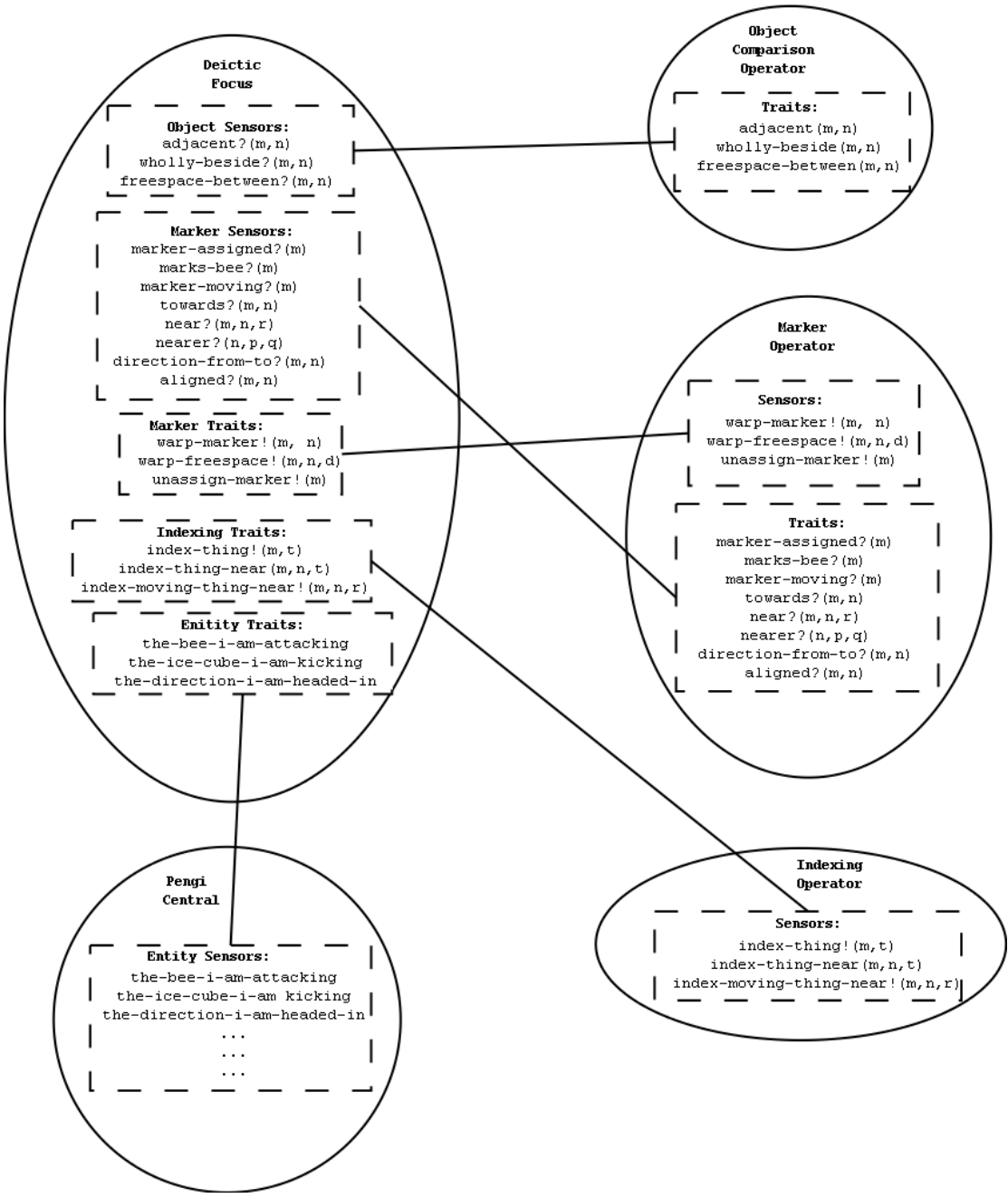
- Visual Objects
    - Penguins
    - Bees
    - Ice Cubes

Provides:

- Visual Operators

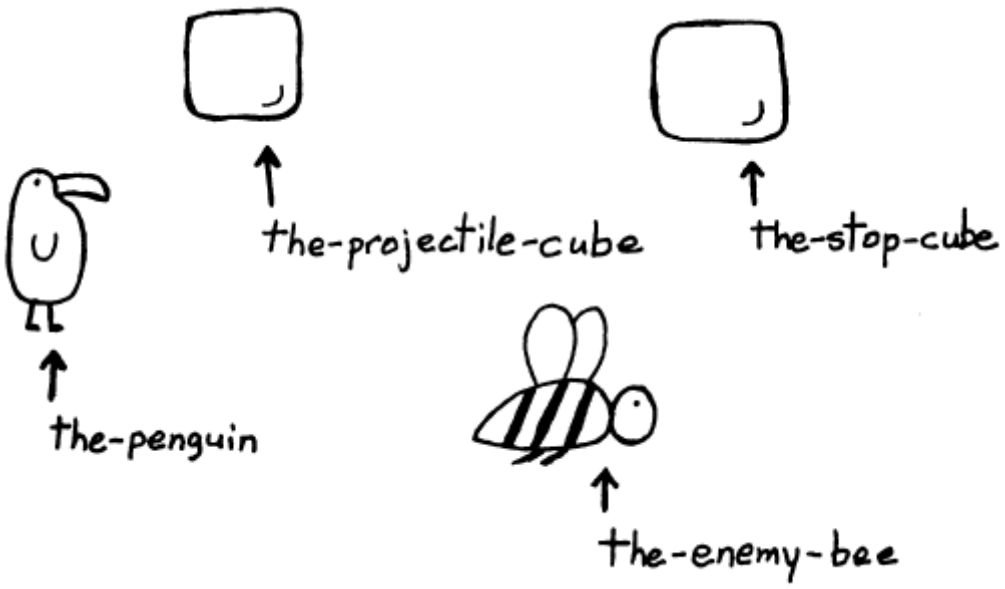
- Indexing operators
- Marker assignment operators
- Marker inspection operators can track markers
- Marker comparison operators
- Object comparison operators

Sensors and Traits Represnting Deictic Entities and Visual Operators



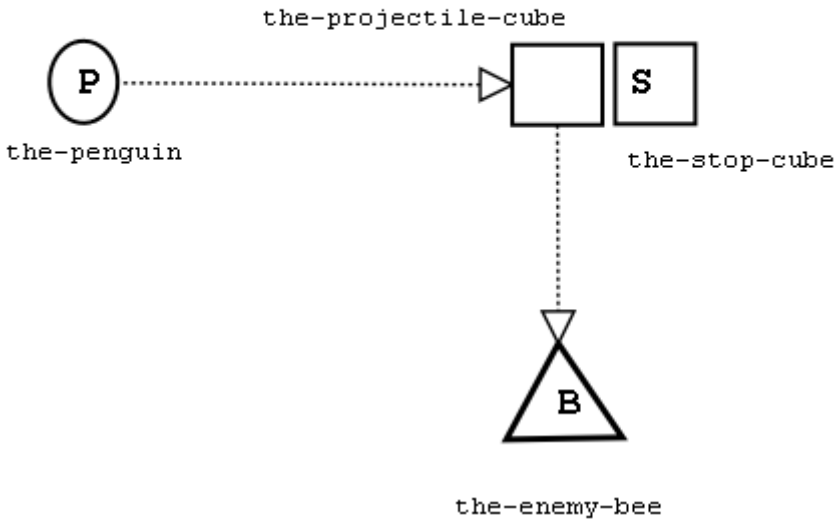
Behaviors

The-two-ice-cube-trick (phase I)




```
(setTrait
    (my-next-move-in-direction-to
        (direction-to
            (the-location-to-hit-cube-to
                (the-new-location-of-projectile-cube-in-bee-channel)
            )
        )
    )
)
```

The two-ice-cube trick (phase II)




```
(if
    (in-bee-channel and not at-hit-location
        (setTrait
            (my-next-move-in-direction-to
                (direction-to
                    (the-location-to-hit-cube-in-bee-channel)
                )
            )
        )
    )
)
```