Simpag CogSpace

MAS simulation and construction environment on 2D grid for Simple Agents

MAS: Multi-Agent System

MAS is an environment where intelligent autonomous agents communicate, evolve and change the environment

MAS Main Application Areas

- Information agents
- Distributed problem solving
- Construction of Synthetic Worlds. ALife.
- Collective Robotics
- Multi-agent simulation
- Etc.

Multi-Agent-Based Simulation

- Multi Agent Based Simulation (MABS) is a computer-based simulation where simulated entities are modeled and implemented in terms of agents.

Simulation Types

- MABS support Discrete Event Simulation (DES)
  - *Time driven* - the simulated time is advanced in constant time steps.
  - *Event driven* - the time is advanced based on when the next event takes place.
  - Events are distinct points of time, or discrete time (possibly irregular time intervals).
  - Nothing happens between events
  - Finite number of events
- Continuous Simulation
  - Continuous time.
  - Can be approximated by DES.

MABS Applications

  - Traffic systems
  - Flexible manufacturing systems
  - Computer-communications systems
  - Production lines
  - Flow networks
- Social systems
  - Crisis management
  - Complex security
  - Urban Population Analysis
- Etc.

Existing MABS

- StarLogo (Media Laboratory, MIT) is a specialized version of the Logo programming language well-suited for swarm modeling and Artificial Life projects
- NetLogo is a multiplatform complexity modeling and simulation environment
- Agentsheets is a commercial product that lets end-users build a wide range of applications that include interactive simulations, games, and intelligent web agents through a graphical user interface.
- Swarm is a software package for multi-agent simulation of complex systems. (GPL)
- RePast is a software framework for creating agent based simulations using the Java language (University of Chicago's Social Science Research Computing). provides a library of classes for creating, running, displaying and collecting data from an agent based simulation. In addition, RePast can take snapshots of running simulations, and create quicktime movies of simulations.
- JACK – Java framework to construct BDI agents

Why build yet another MABS system?

- In most cases general purpose MAS does not answer DES requirements.
- Existing MABS have one or more of the following limitations:
  - Support best only some particular agent modeling approach or theory.
    - Swarm modeling: StarLogo, NetLogo, Swarm, Repast
    - Rational (BDI) agents (Jack)
  - Provide limited modeling language : Logo (StarLogo), JACK Agent Language – Java preprocessing extensions to represent BDI elements.
  - Provide little support for time synchronization required by time-based DES simulations (Jack).
- Also existing MABS have limited support for:
  - Modular agent construction
  - Experiment construction and reuse

Main Ideas of SimpAg MAS

- Deictic Representation. Entities in agent environment are represented in two ways:
  - Indexically - in terms of their relation to the agent.
  - Functionally – in terms of the role they play in agent's ongoing projects.
- Running arguments. Responsive, flexible behavior based on continually redeciding what to do (as opposed to plan execution)

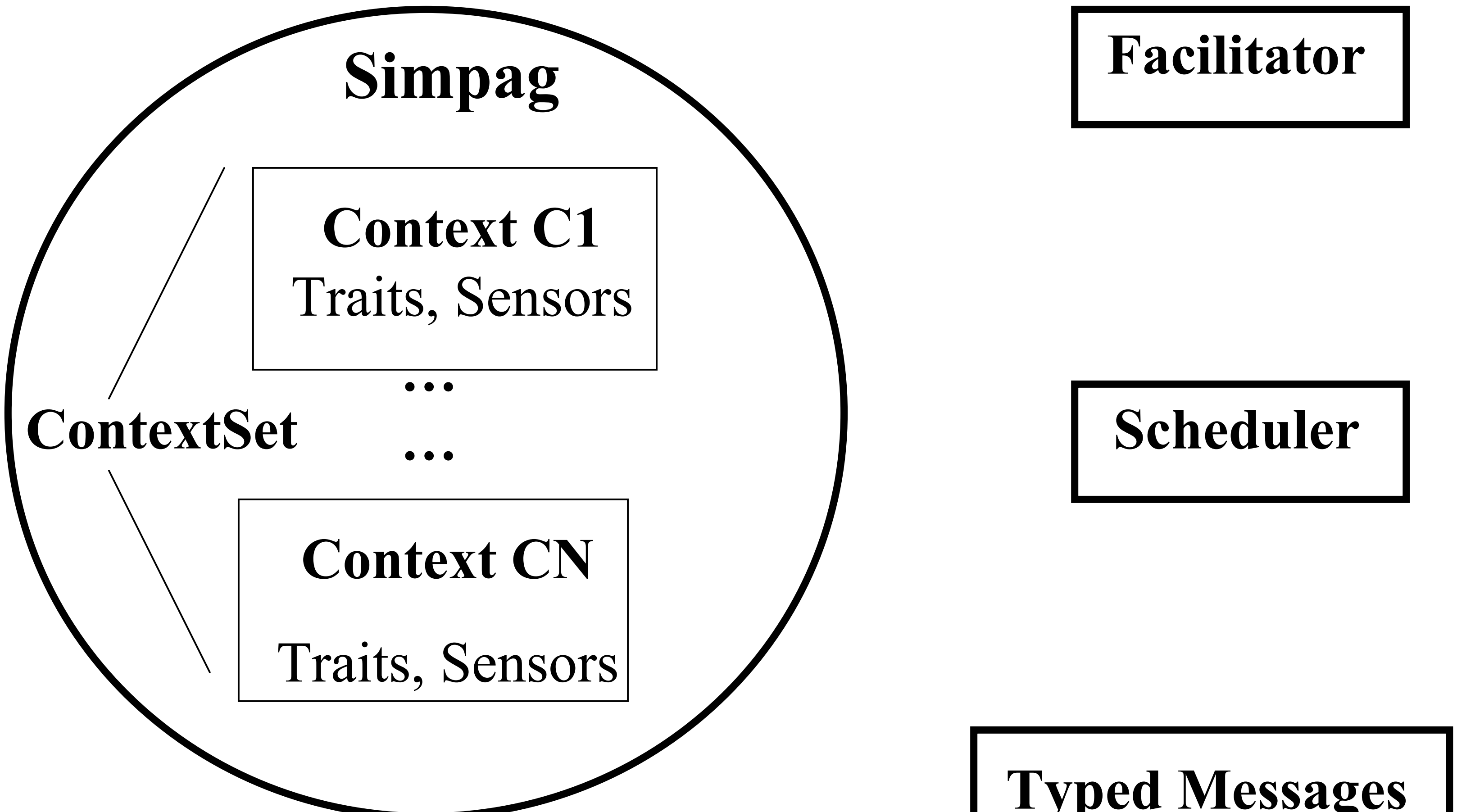
SimpAg MAS goals

- Support different AI methods and agent paradigms, and in particular deictic representation with “running arguments”.
- Provide construction blocks to build holonic agents with various level of intelligence.
- Support fine-grain, distributed modeling.
- Define Simple Agents (simpags) to operate in deictic representation of environment.
- Support collaborative agent behavior based on “running argument” principle.

Simpag MAS Framework. Main Components.

- **Simpag** agent consists of:
  - **ContextSet** containing **Distributed Contexts** that Simpag understands at any given moment – **Simpag Deictic Focus**
  - **Senors**
  - **Traits**
- **Typed Messages -** pass information between sensors and traits
- **Facilitator** – binds sensors and traits in context
- **Scheduler** – provides time driven DES

Simpag Framework: Main Components



Simpag Contexts

- Represent particular agent activity.
- Defines attributes, behaviors and communications specific to some activity.
- Examples of contexts are: room navigation, moving in a group, patrolling, defending perimeter, etc.

Agent behavior in contexts

- Contexts organize improvised agent behavior in **deictic representation** of the environment according to **running arguments** principle.
- Complex behavior is orchestrated as a **context set** of more simple ones.
- Every agent has associated **context set** containing one or more contexts.
- One and the same agent can participate in several contexts at once.

Context Features: Sensors and Traits

- Traits are agent features observable by other agents.
- Sensors are features that agent use to get sensory input from other agents and environment.
- The same sensors and traits may belong to different contexts.
- Context is active when at least one sensor in this context has new value.

Compass Context

Sensor: Direction_To_Target

Trait: Current_Location

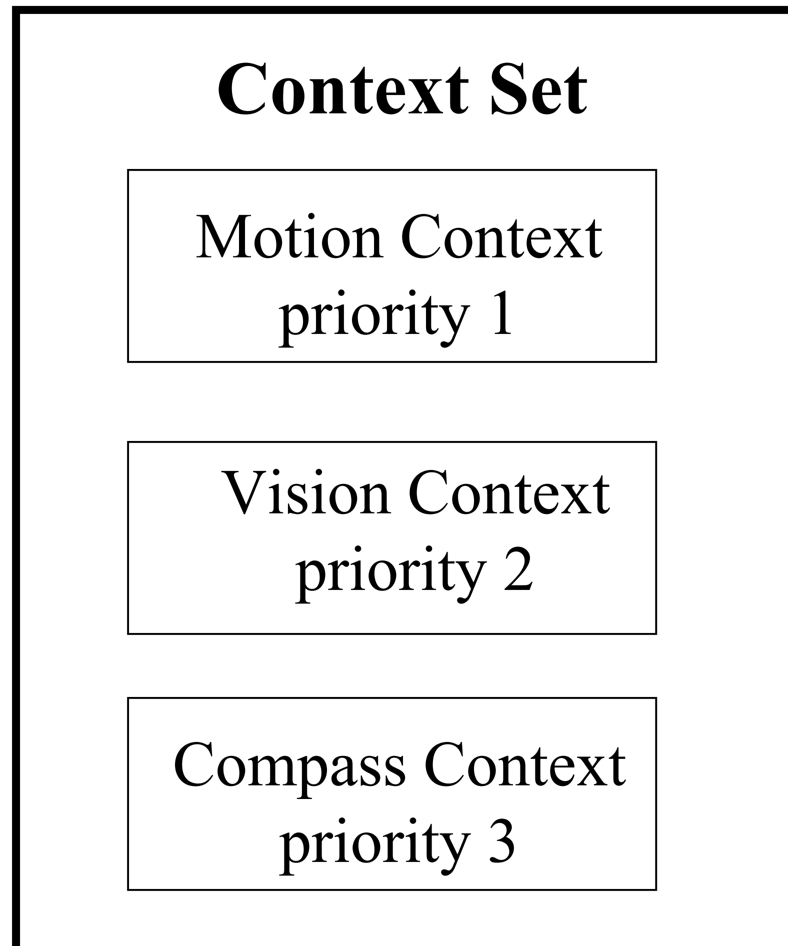
Vision Context

Sensor: Grid_View

Trait: Current_Location

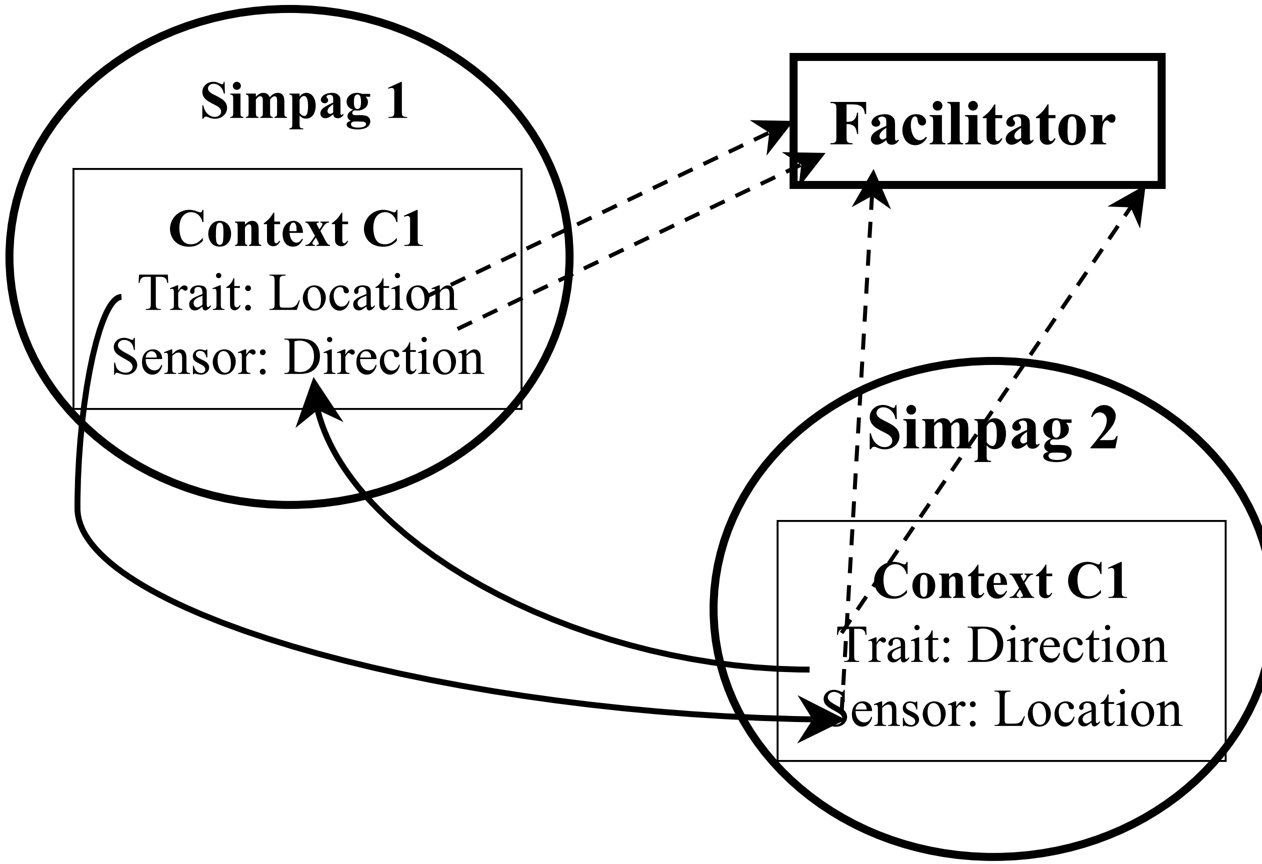
Context Set

- Context represent simple behavior.
- Several contexts are organized in Simpag ContextSet.
- Every context has priority in ContextSet.
- At every simulation step only one context with highest priority becomes active.



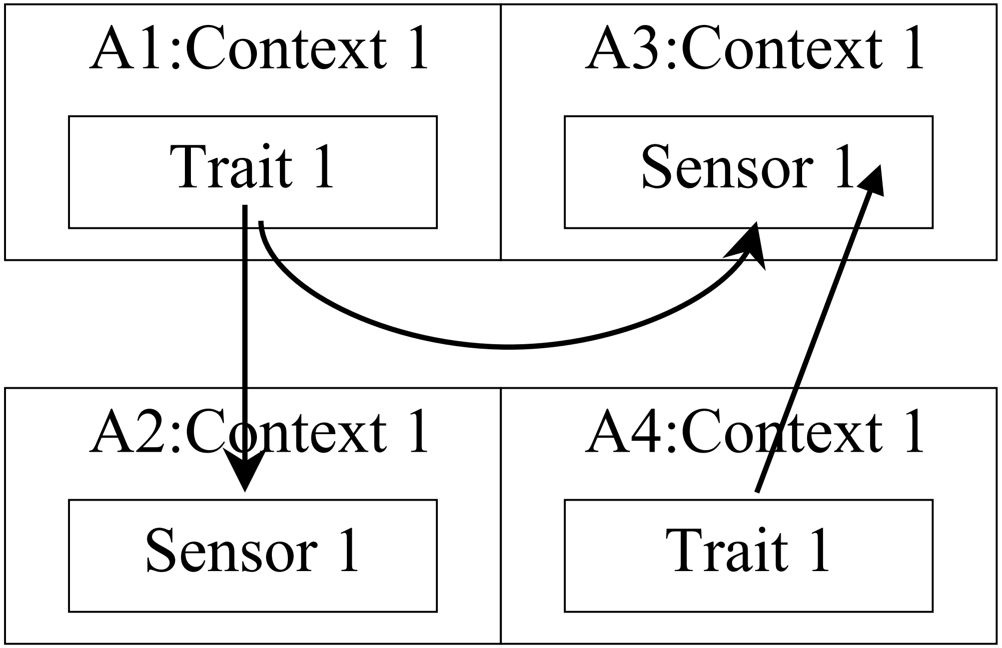
Binding Traits & Sensors

- Simpag publishes its traits and sensors in one or more contexts
- Facilitator binds (→) traits to sensors in the same context instance



Reading from Sensors and Writing to Traits

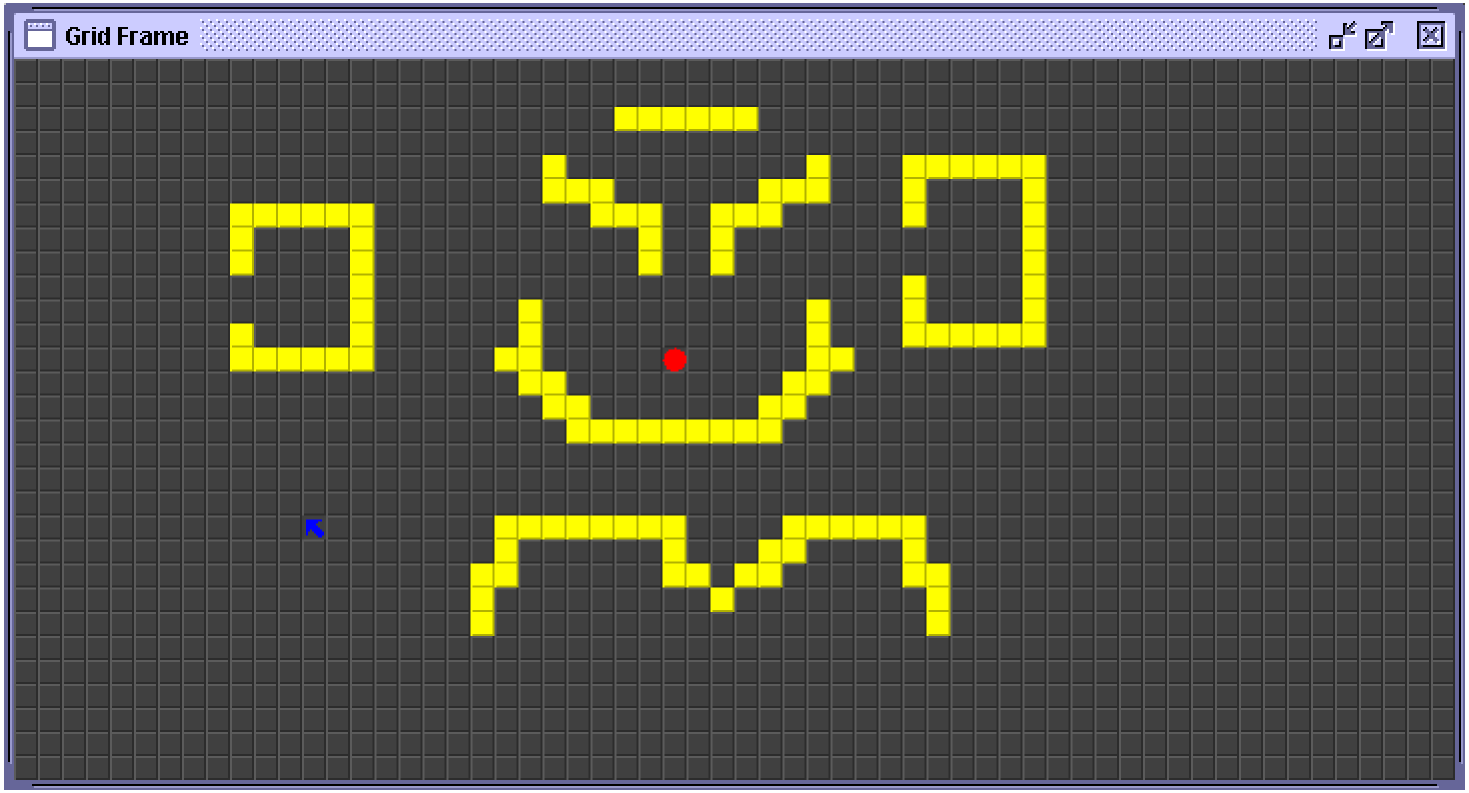
- Writing feature value to trait results in passing Typed Message to sensors bound to this trait
- Reading from sensor returns new feature value if corresponding bound trait was updated on current simulation step.



Simulation Step

- 1. Increment SSN - Simulation Step Number .
- 2. Scheduler: while exist Simpag in the set of running agents, do:
- 3. Scheduler pass control to Simpag ContextSet with SSN as a parameter
- 3.1. ContextSet sets context priority to the highest value defined by this Simpag.
- 3.2. ContextSet gets context from the set with the given priority.
- 3.3. ContextSet checks: if selected context is active then pass control to this context, go to 2. Else decrease priority. Go to 3.2
- 4. Go to 2.

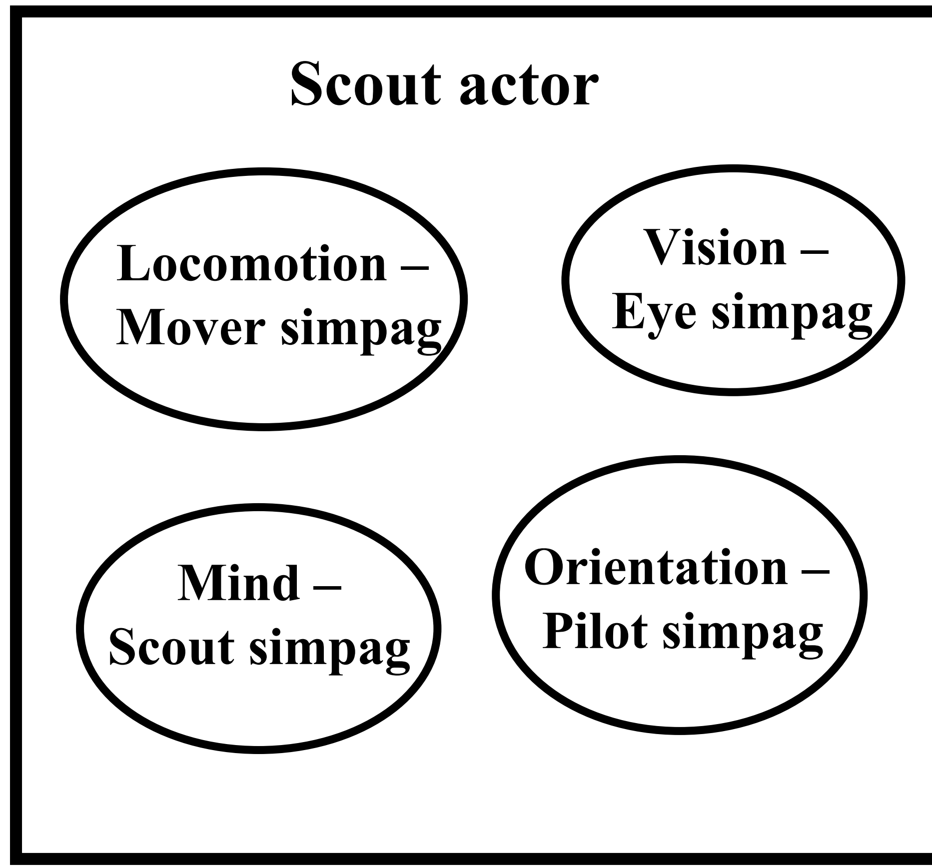
Scout Example



- Scout is an actor that lives in 2D grid space
- Scout goal is to reach target
- While approaching target Scout must walk around different obstacles, such as walls, buildings, etc.

Scout Architecture

- Scout actor is implemented as Simpag MAS that models an artificial being with the following capabilities implemented by separate simpags:
  - Locomotion – ability to move its body on 2D grid. Implemented by Mover simpag.
  - Orientation – ability to calculate direction to the target from current location on 2D grid. Implemented by Pilot simpag.
  - Vision – ability to ‘see’ surrounding objects. Implemented by Eye simpag.
  - Mind – central system that orchestrates other actor capabilities. Implemented by Scout simpag.



Scout Locomotion

- Scout lives and moves on 2D grid according to compass directions: N, NE, E, SE, S, SW, W, NW. Grid can be a toroid or not.
- Locomotion capability is implemented by Mover simpag in **MoverMotion Context** that encapsulates knowledge of world geometry.
- Mover ContextSet has only one context: MoverMotion.
- Mover works with Grid world that allows to:
  - Put Actor to some cell
  - Remove Actor from some cell
- When Mover gets request to move Actor, it validates new location and actually removes Actor from current location and puts it to a new location (cell).
- Mover gets move request in **nextStep sensor** and returns new Scout location in **currentCell trait**.

Scout Orientation

- On 2D grid Scout can use orientation system implemented by Pilot simpag in **PilotCompass Context**.
- Pilot ContextSet has only one context: PilotCompass.
- Pilot knows where target is located.
- When Pilot gets current location (cell) of Scout actor it calculates direction from this location to target.
- Pilot gets Scout location in **currentCell sensor** and returns directions to target (N, NE, E, SE, etc.) in **dirToTarget trait.**

Scout Vision

- Scout has a ‘vision system’ implemented by Eye simpag in **EyeVision Context**.
- Eye ContextSet has only one context: EyeVision.
- Eye gets current location of Scout actor in currentCell trait.
- Eye calculates ‘view raster’ with Scout in its center.
- Every cell in this raster has value NOT_EMPTY_CELL for cells that contain some object(s) or EMPTY_CELL for empty cells.
- Eye returns ‘view raster’ to Scout in **gridView trait**.

View Raster

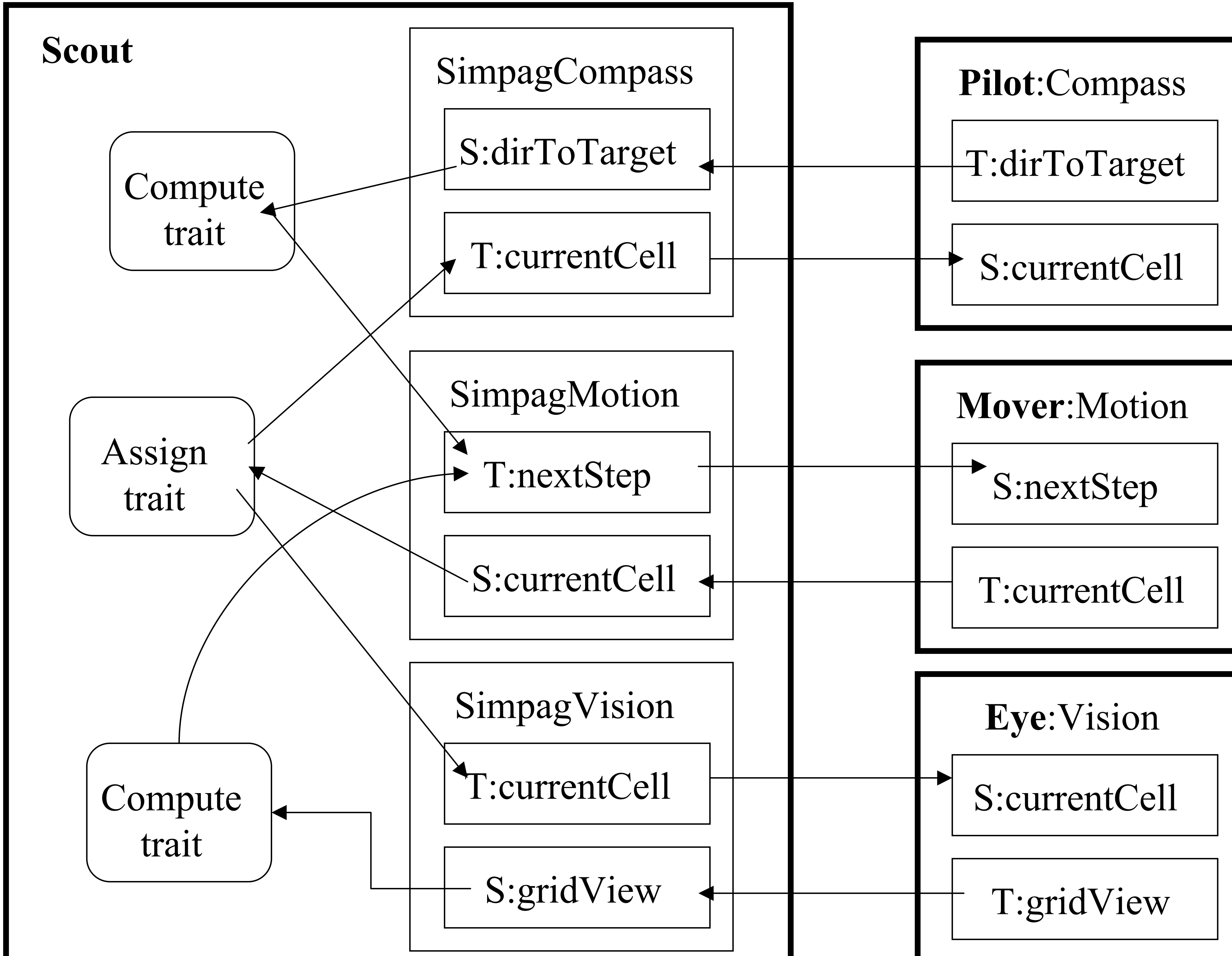
*	*	
	A	

Scout Mind

- Scout Mind is minimal, its only function is to orchestrate Locomotion, Orientation and Vision subsystems of the Scout.
- Mind is implemented by Scout simpag in SimpagCompass, SimpagMotion and SimpagVision contexts that work with corresponding subsystems of the Scout.
- These contexts are orchestrated by Scout ContextSet that assign them the following priorities:
  - SimpagMotion – 1, highest priority.
  - SimpagVision - 2
  - SimpagCompass – 3, lowest priority.

Scout Simulation in Detail

- Next slides show complete Scout MAS implementation with detailed description of context activation in all agents.
- Important notes:
  - Scout is a holonic agent – it consist from 4 simpags.
  - Many Scouts may run in Simpag MAS, thus total number of agents (simpags) is equal to N * 4, where N is a number of Scout Actors.
  - One and the same feature (trait, sensor) may be published in several contexts at once. In this example currentCell trait is published both in SimpagCompass and SimpagVision contexts



Scout
SimpagCompass
S:dirToTarget
T:currentCell
Compute trait
Assign trait
SimpagMotion
T:nextStep
S:currentCell
SimpagVision
T:currentCell
S:gridView
Compute trait
Pilot:Compass
T:dirToTarget
S:currentCell
Mover:Motion
S:nextStep
T:currentCell
Eye:Vision
S:currentCell
T:gridView

Scout Simulation Phase, 1

- This phase starts when:
  - All simpags (Scout, Mover, Pilot and Eye) participating in simulation are created and registered with scheduler.
  - All traits and sensors are bound in all contexts.
- During initialization Scout gets its current location (cell).
- Scout sets its currentCell trait value to initial location.
- SimpagCompass context pass current location info to currentCell sensor in PilotCompass context of the Pilot simpag.
- SimpagVision context pass current location info to currentCell sensor in EyeVision context of the Eye simpag.

Scout Simulation Phase, 2

- Pilot computes direction to target and sets dirToTarget trait to corresponding value.
  – Eye finds if there are any obstacles around current location of the Actor.
  – If obstacles found Eye creates ‘raster view’ were obstacles are marked. Eye sets gridView trait to new ‘raster view’ value.
- If no obstacles detected – Eye does nothing

Scout Simulation Phase, 3

- Scheduler passes control to Scout ContextSet
- ContextSet finds active context of the Scout with the highest priority.
- SimpagMotion context is not active now, as it has no sensor with new message at current simulation step.
- If SimpagVision context has active sensor (in this case gridView sensor has a new message) – it becomes active. ContextSet stops looking for other active contexts because SimpagVision has the highest active priority now which is equal to 2.
- SimpagVision context computes and sets new value for nextStep trait in accordance with the value of gridView sensor.

Scout Simulation Phase, 4

- When there were no obstacles detected by Eye, then SimpagVision context is not active.
- Scout ContextSet selects next active context the maximum priority at this simulation step, which is SimpagCompass context, with priority 3.
- SimpagCompass context computes and sets new value for nextStep trait in accordance with the value of dirToTarget sensor.

Scout Simulation Phase, 5

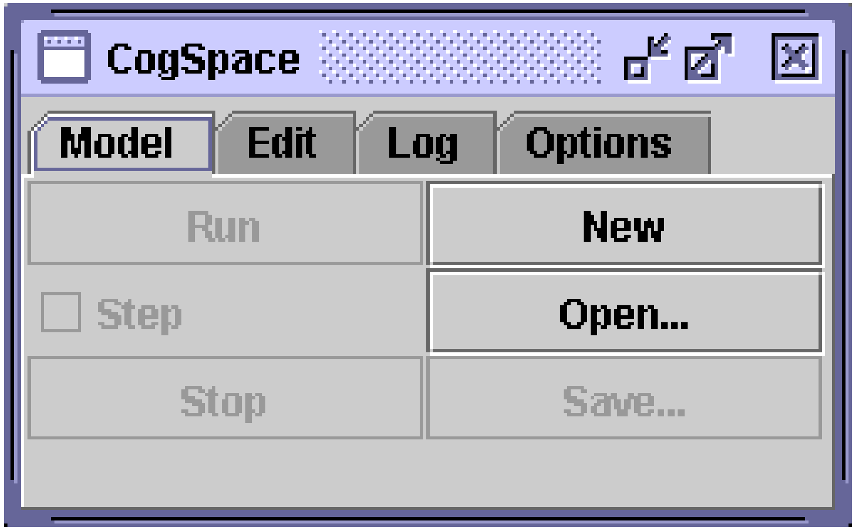
- When nextStep trait is set to a new value in SimpagMotion context then nextStep sensor activates MoverMotion context in Mover simpag.
- Mover moves Actor to a new location and sets new value for currentCell trait in MoverMotion context.
- As a result currentCell sensor activates SimpagMotion context which has the highest priority in Scout ContextSet.
- SimpagMotion context copies value form its currentCell sensor to Scout currentCell trait.
- Again new value in currentCell trait activates PilotCompass and EyeVision contexts in Pilot and Eye simpags respectively.

CogSpace Graphic Environment

- CogSpace provides MAS simulation and construction environment on 2D grid to
- Build:
  - Initial static environment from bricks (cells) of arbitrary configuration (walls, buildings, etc.)
  - Targets – static or dynamic, simulated real-world entities, such as power sources, fuel and ammunition stores, control centers, etc.
- Create initial experiment configuration including static objects and agents (simpags).
- Run simulation and observe it results on 2D grid in continues and step mode.
- Save / restore full state of experiment any time during simulation.
- Provide reusable library of construction elements both for agents and static objects.

CogSpace GUI: Model

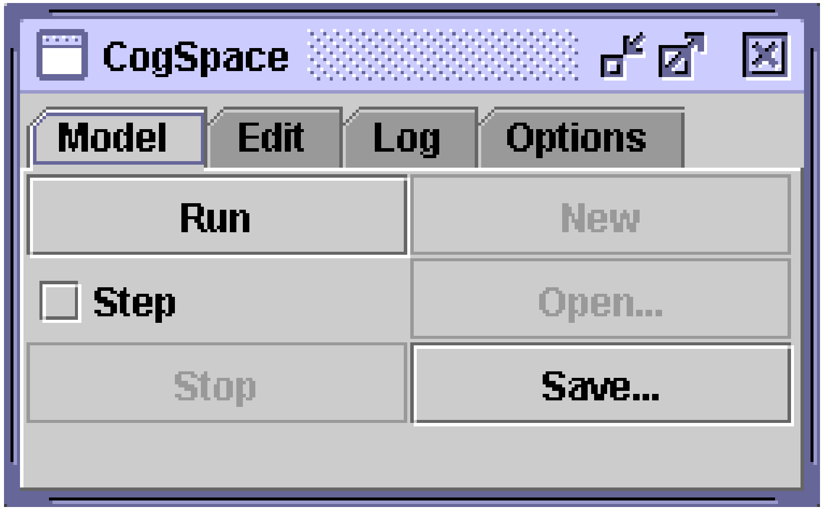
- Model GUI allows to:
- Create New model and corresponding 2D grid
- Open existing model, previously saved (not implemented)



CogSpace: GridFrame

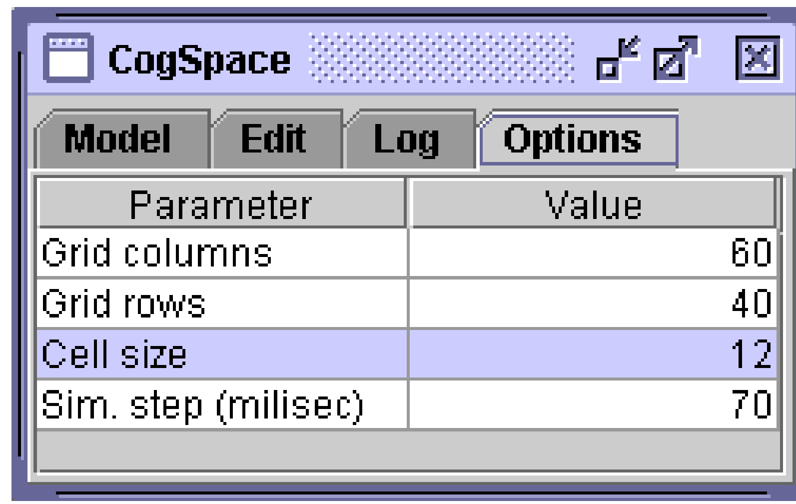
CogSpace GUI: New Model

- Run – start continues simulation. When started Run buttons becomes inactive and Stop button becomes active
- Step – turn on Step mode – one simulation step at a time
- Save – save current model

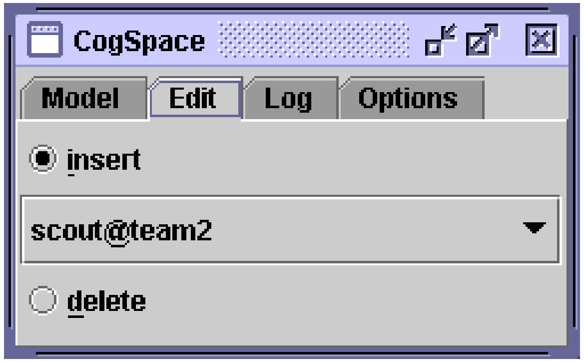


CogSpace: Options

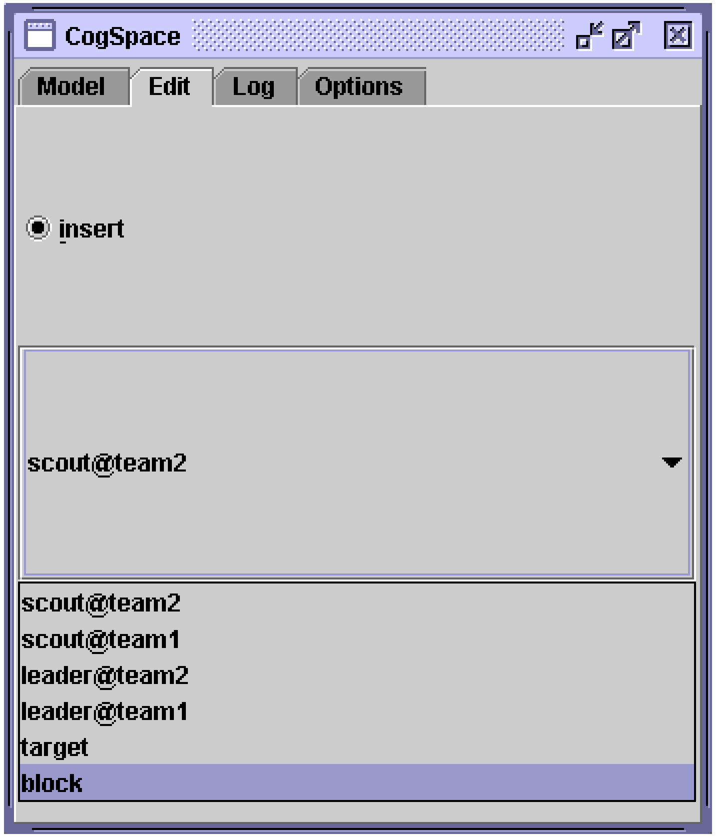
- Oprions GUI allows to set the following options:
- Number of Grid columns
- Number of Grid rows
- Cell size
- Simulation step in milliseconds



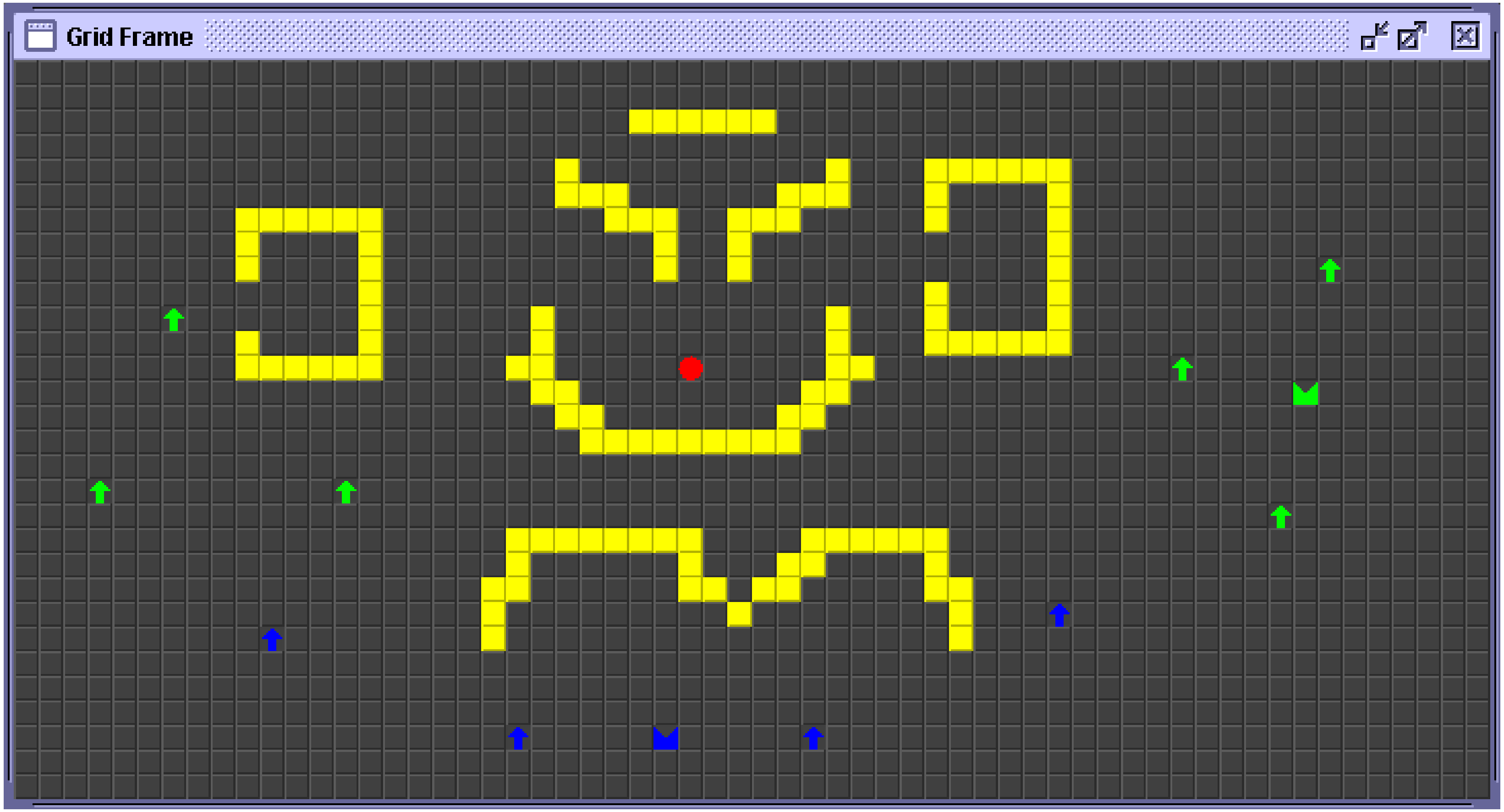
CogSpace: Edit



- Edit panel allows to manipulate with instances of pre-installed objects and simpags:
- Inset instance at cell pointed by mouse in the Grid panel
- Delete instance at cell pointed by mouse in the Grid panel
- Type of pre-installed object can be selected from drop-down menu.

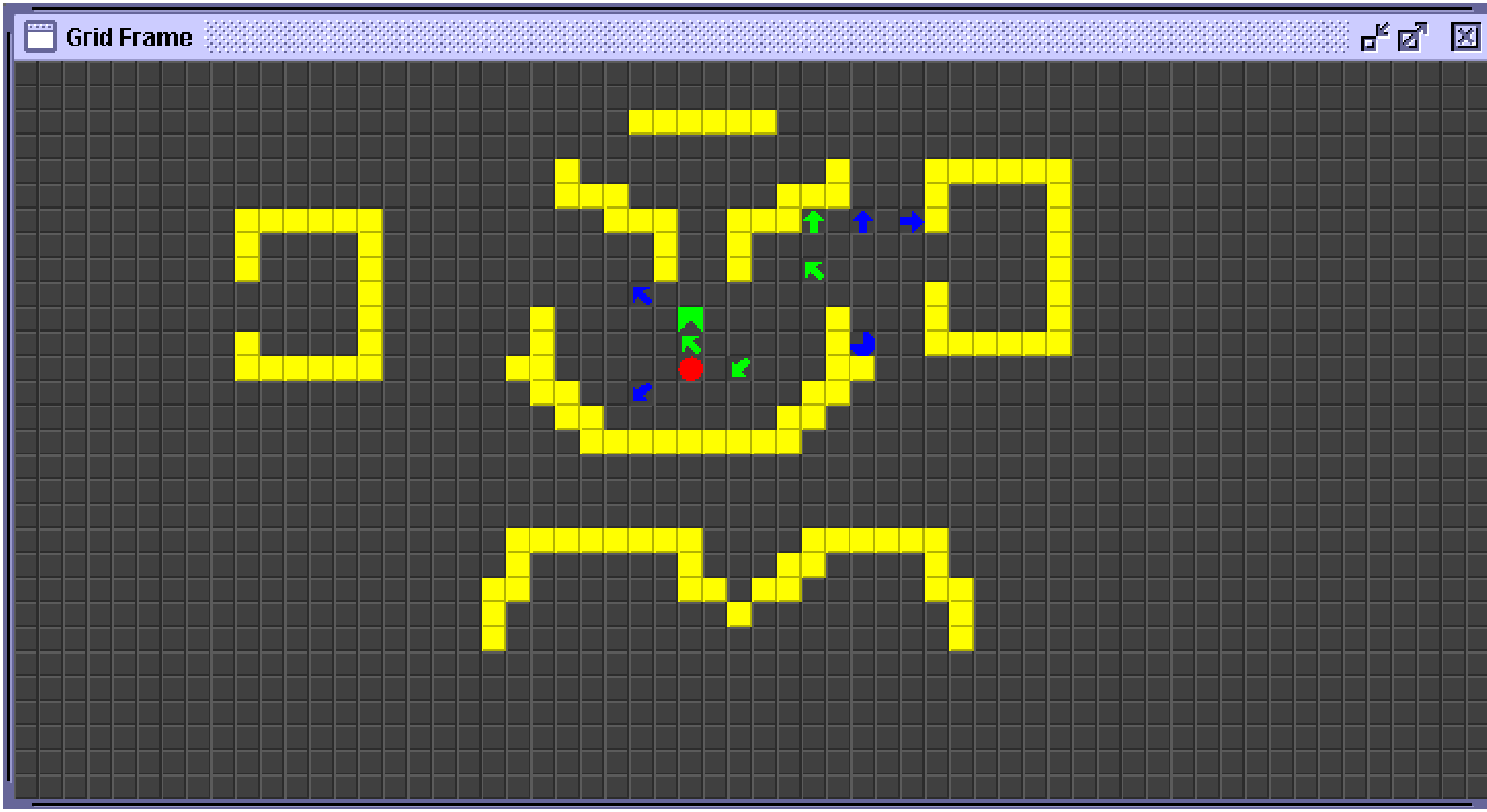


CogSpace: Creating Experiment Environment



- Green and blue widgets represent Scout Actors (discussed above in detail) of two opposing teams.
- Red circle is a target that one team defends from another one.
- Yellow blocks are used to build walls and buildings.

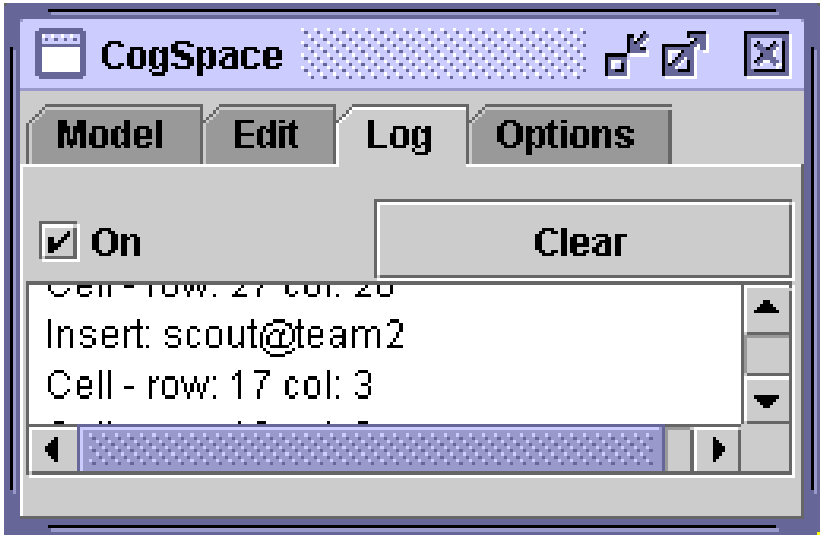
CogSpace: Running Experiment



- Pushing ‘Run’ button starts running experiment in continues mode
- New objects may be inserted / deleted in the Grid without stopping experiment
- Experiment may be stopped with ‘Stop’ button
- When “Step” mode is selected pressing right mouse button on GridFrame advances simulation step by step.

CogSpace: Log

- Log panel can be used by other components of CogSpace to display log and debug information in run time
- Log can be activated and de-activated any time during simulation.
- Clear button – clears log window.



Developing New Simulation

- CogSpace Simpag is an open framework implemented in Java that provides the following means to develop simulations:
- Simpag class library with:
  - Helper classes with default implementations of simpag agents, contexts, traits, sensors and messages.
  - Factory to create and plug-and-play with new implementations of all components.
- CogSpace class library with:
  - Widget library to create Actor and other object representations for 2D Grid world.
  - GUI library to replace / extend current graphical environment.
  - Factory to create and plug-and-play with new implementations of all components.

Future Work

- Future work includes both long and short-term goals.
- Short-term goals includes implementation of:
  - Save/Restore models.
  - Extend context library with new behaviors, in particular group behaviors.
- Long-term goals include:
  - High-level modeling language.
  - Learning agents operating in deictic environment.
  - Construction library.
  - Infinite, unbounded Grid world.
  - Distributed simulation.