

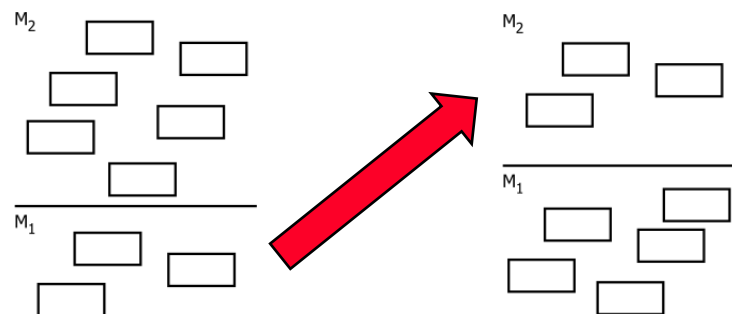


# ***On the Execution of Deep Models***

Colin Atkinson, Ralph Gerbig and Noah Metzger



- We see potential for the application of deep models in
  - Representation of execution state
  - Online influencing of model execution
  - Pause/resume model execution
- In “two-level models” workarounds are applied, such as
  - Model Copies
  - UML Profiles
  - Annotation Models
  - Promotion transformations



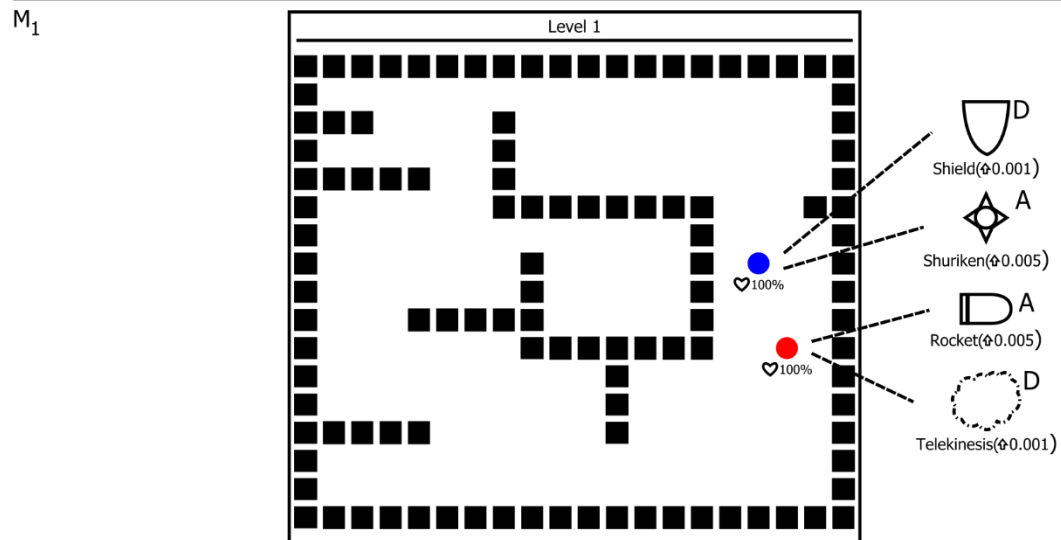
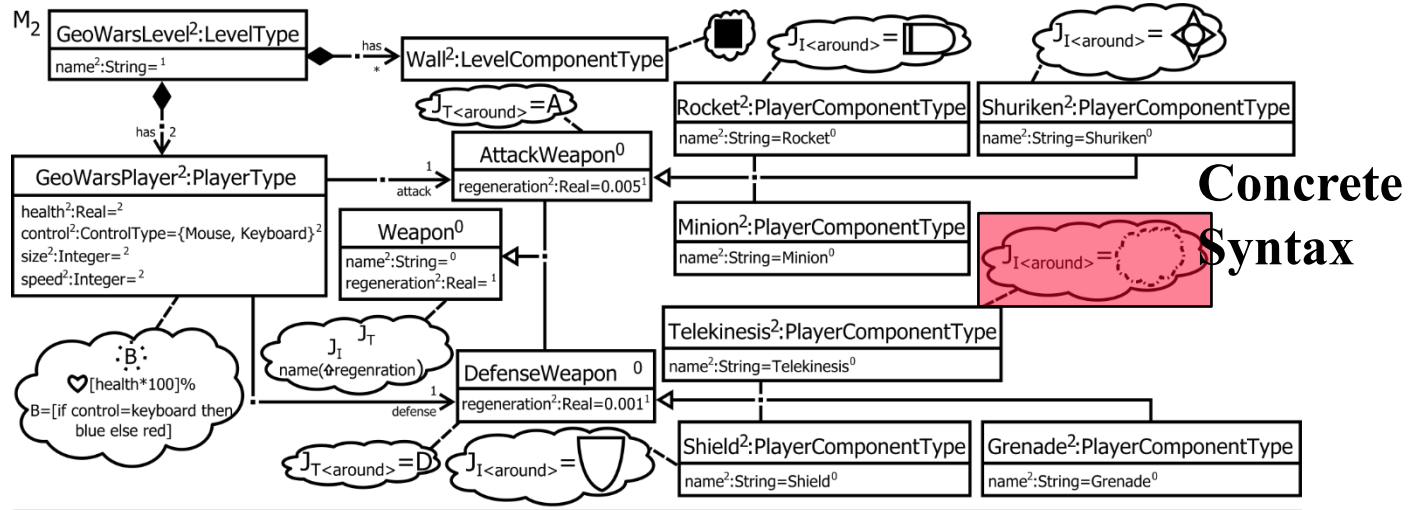
# The Example



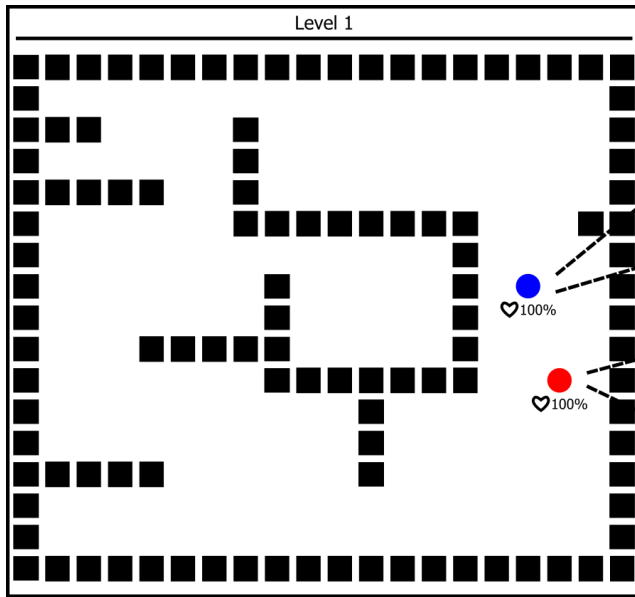
- Two players play against each other from the birds eye perspective
- The game is an executed deep-model
- The model and game are connected via sockets
- Changes in to the model/game are immediately reflected at the other end.
- A third player can change the model at execution time and changes take immediate effect





Linguistic	Property	Value
Ontological	control	Keyboard
	health	1.0
Visualization	size	30
	speed	3

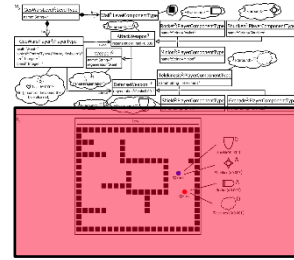
# The Game in a "two-level" Style



# Execution of Model



-  D  
Shield( $\phi$ 0.001)
-  A  
Shuriken( $\phi$ 0.005)
-  A  
Rocket( $\phi$ 0.005)
-  D  
Telekinesis( $\phi$ 0.001)



Level: Address Field Start Fight

Juggernaut

Health: [Blue bar]

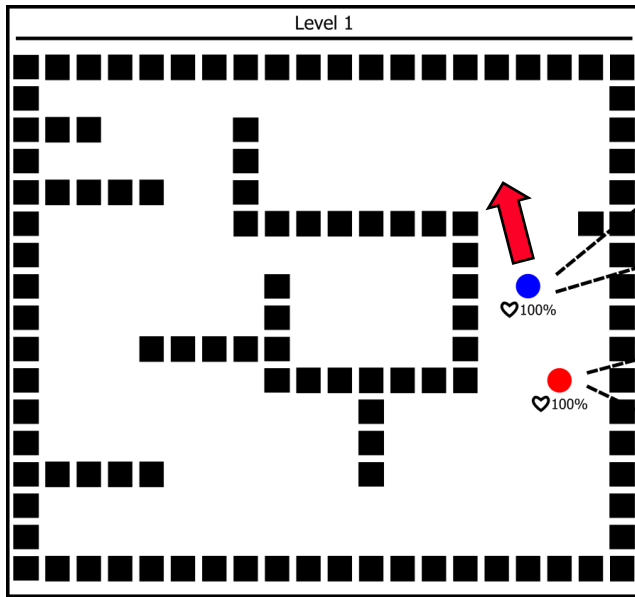
Offensive: [Blue bar] Defensive: [Blue bar]

Ninja

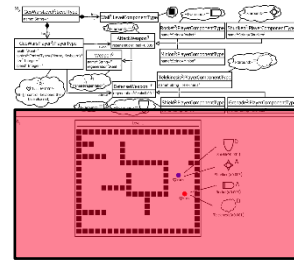
Health: [Red bar]

Offensive: [Red bar] Defensive: [Red bar]

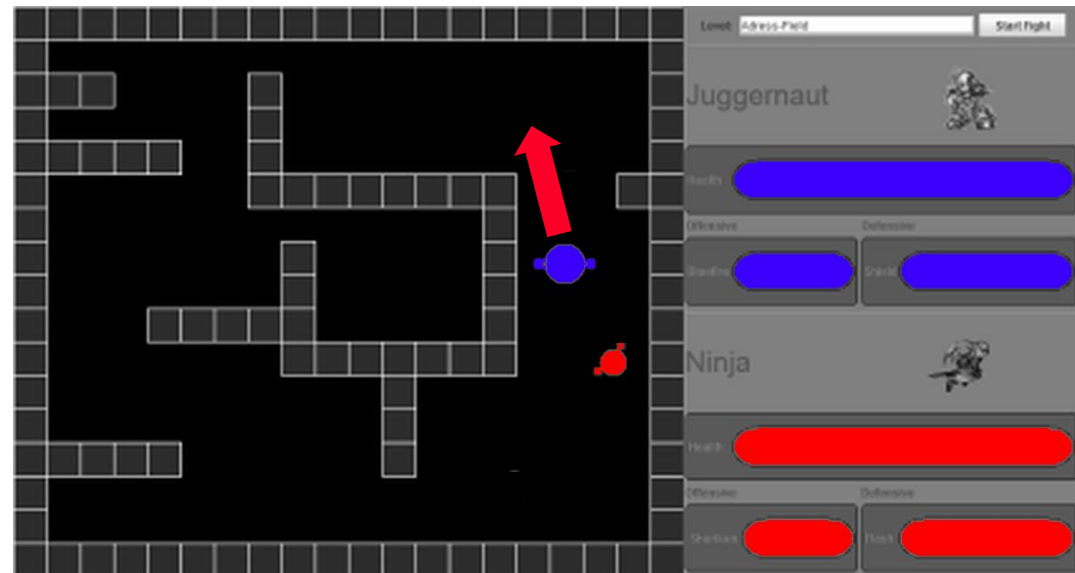
# Representing Execution State 1/2



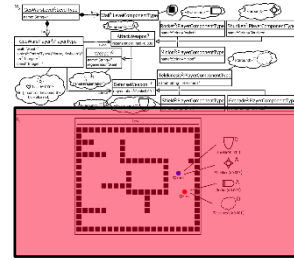
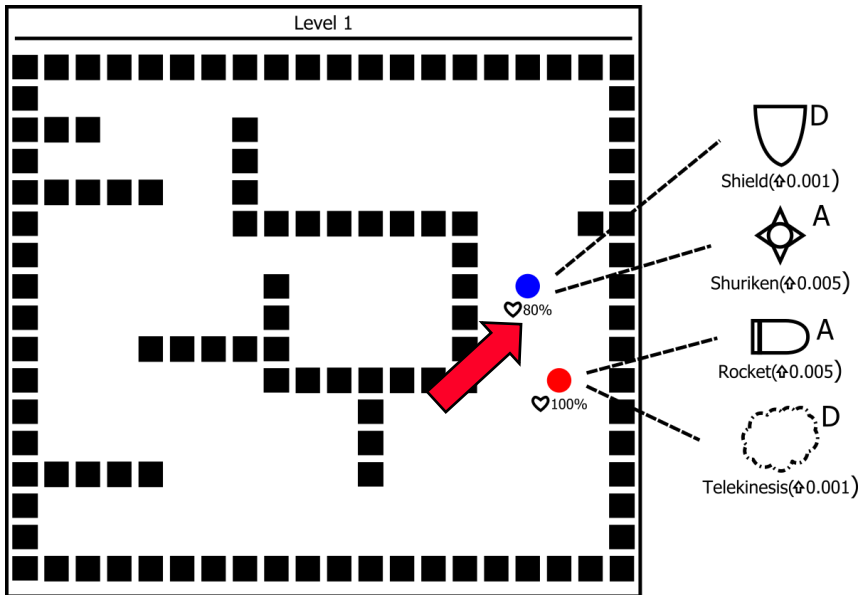
- Shield (D)  $\phi 0.001$
- Shuriken (A)  $\phi 0.005$
- Rocket (A)  $\phi 0.005$
- Telekinesis (D)  $\phi 0.001$



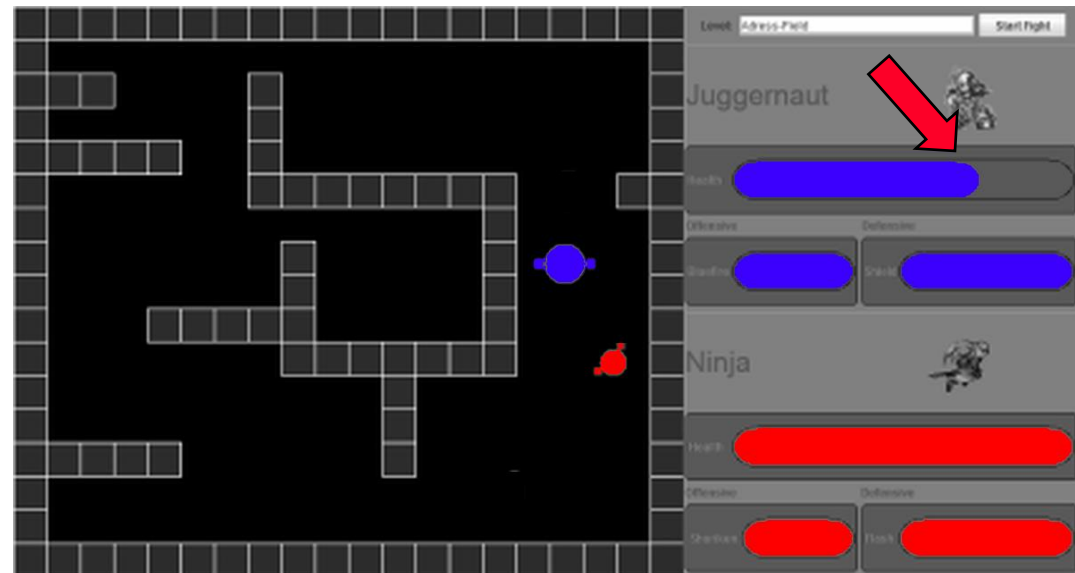
- How can a move be represented at M1?



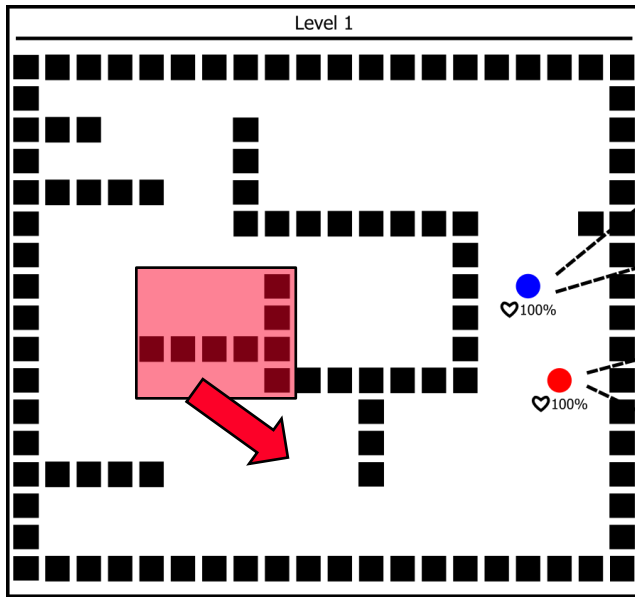
# Representing Execution State 2/2



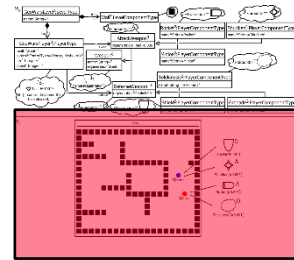
- How can a change to health be represented at M1?



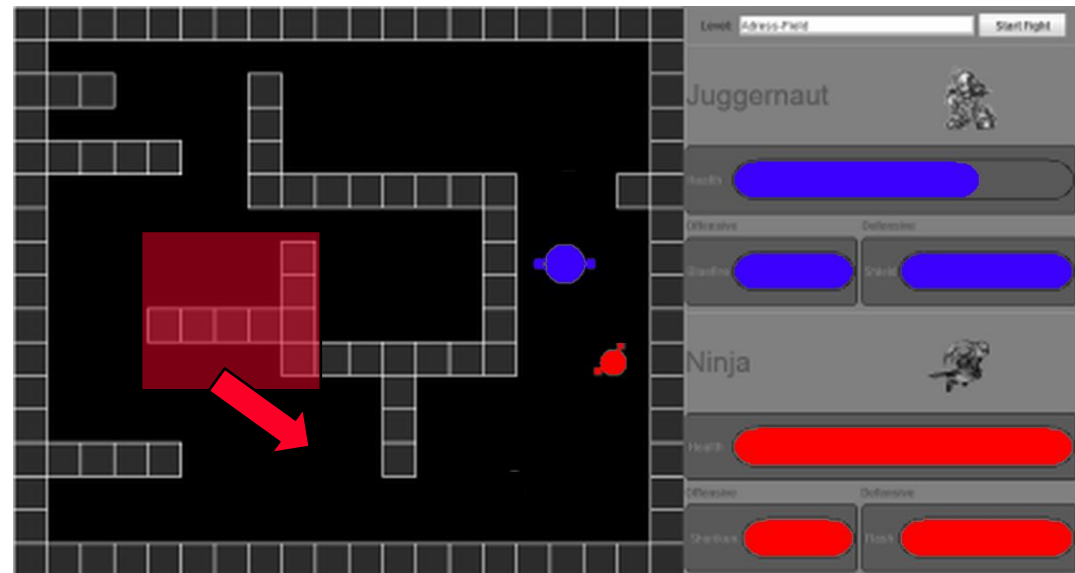
# Influencing Model Execution 1/2



- Shield( $\phi$ 0.001)
- Shuriken( $\phi$ 0.005)
- Rocket( $\phi$ 0.005)
- Telekinesis( $\phi$ 0.001)

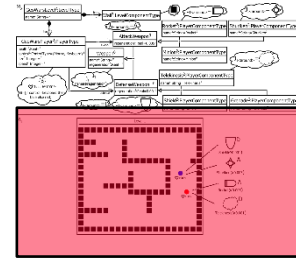
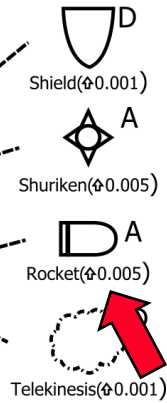
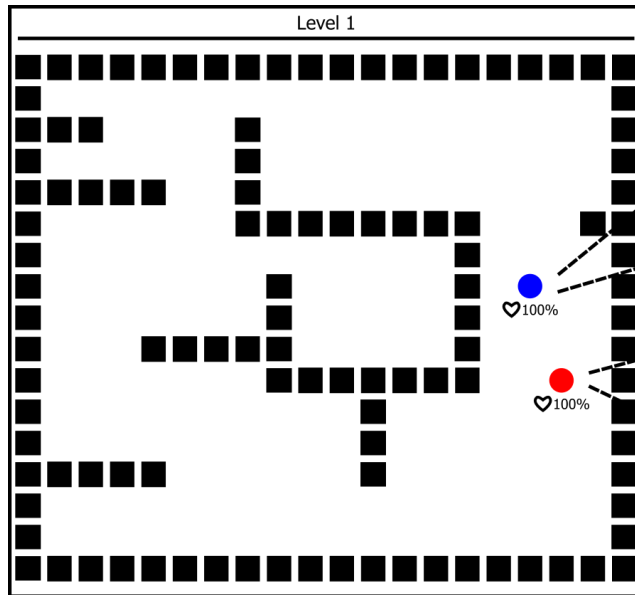


- I want to move a wall during model execution!

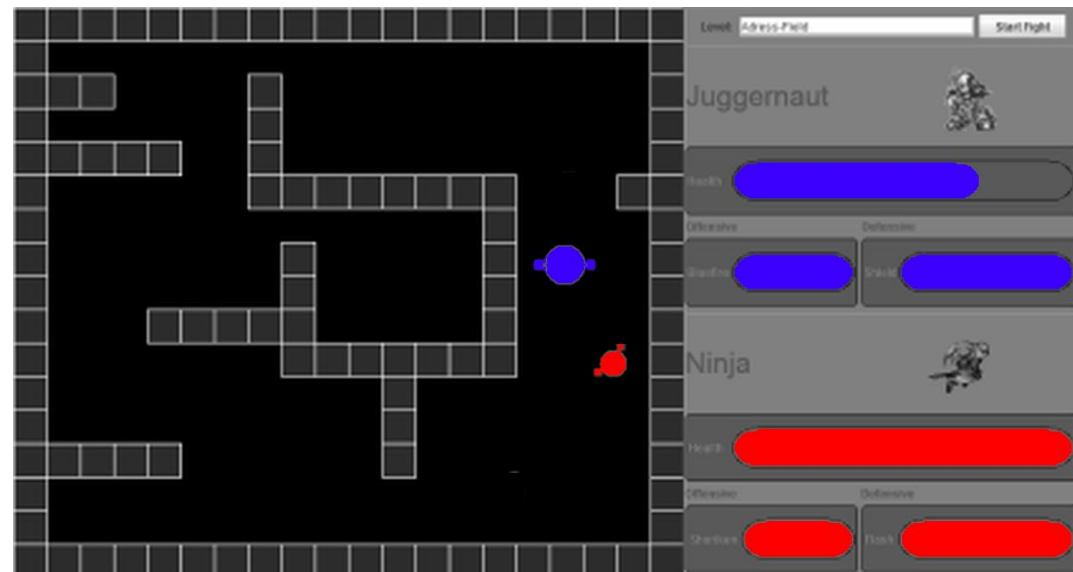




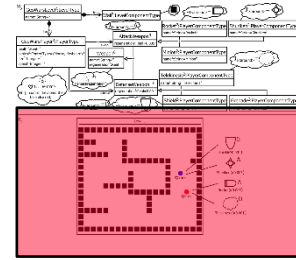
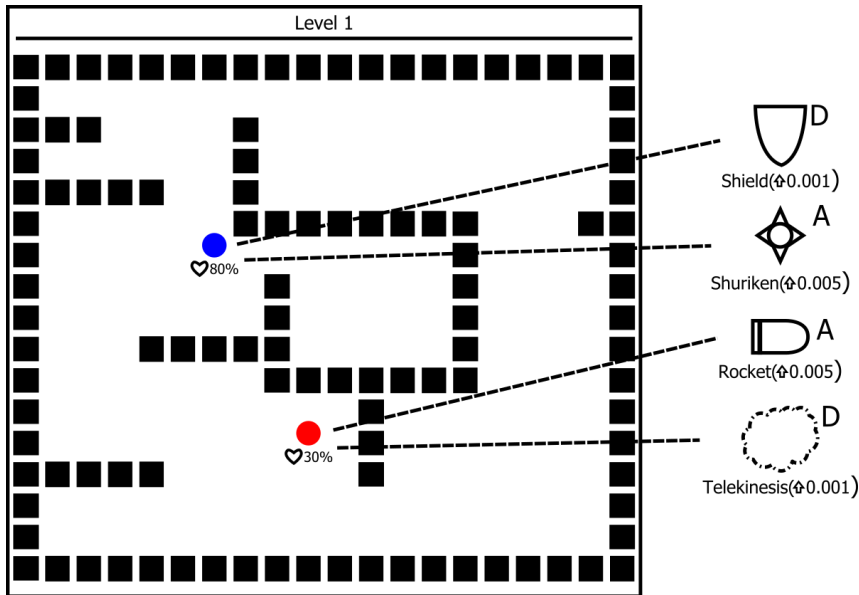
# Influencing Model Execution 2/2



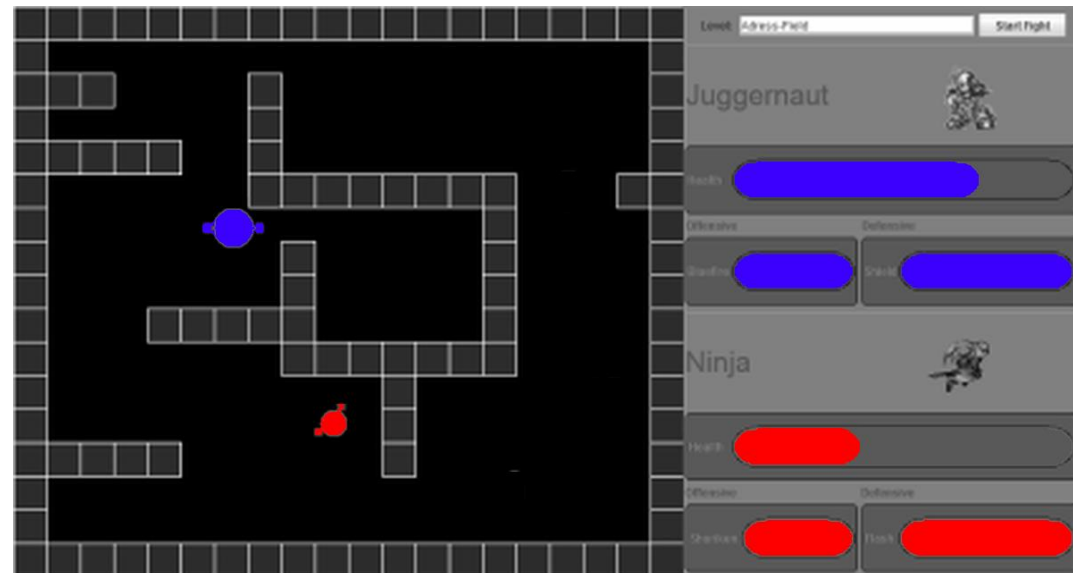
- I want to change the regeneration speed of a weapon during execution!



# Pause/Resume Model Execution State



- I would like to pause the game and resume later!





## The Execution Blueprint is polluted through

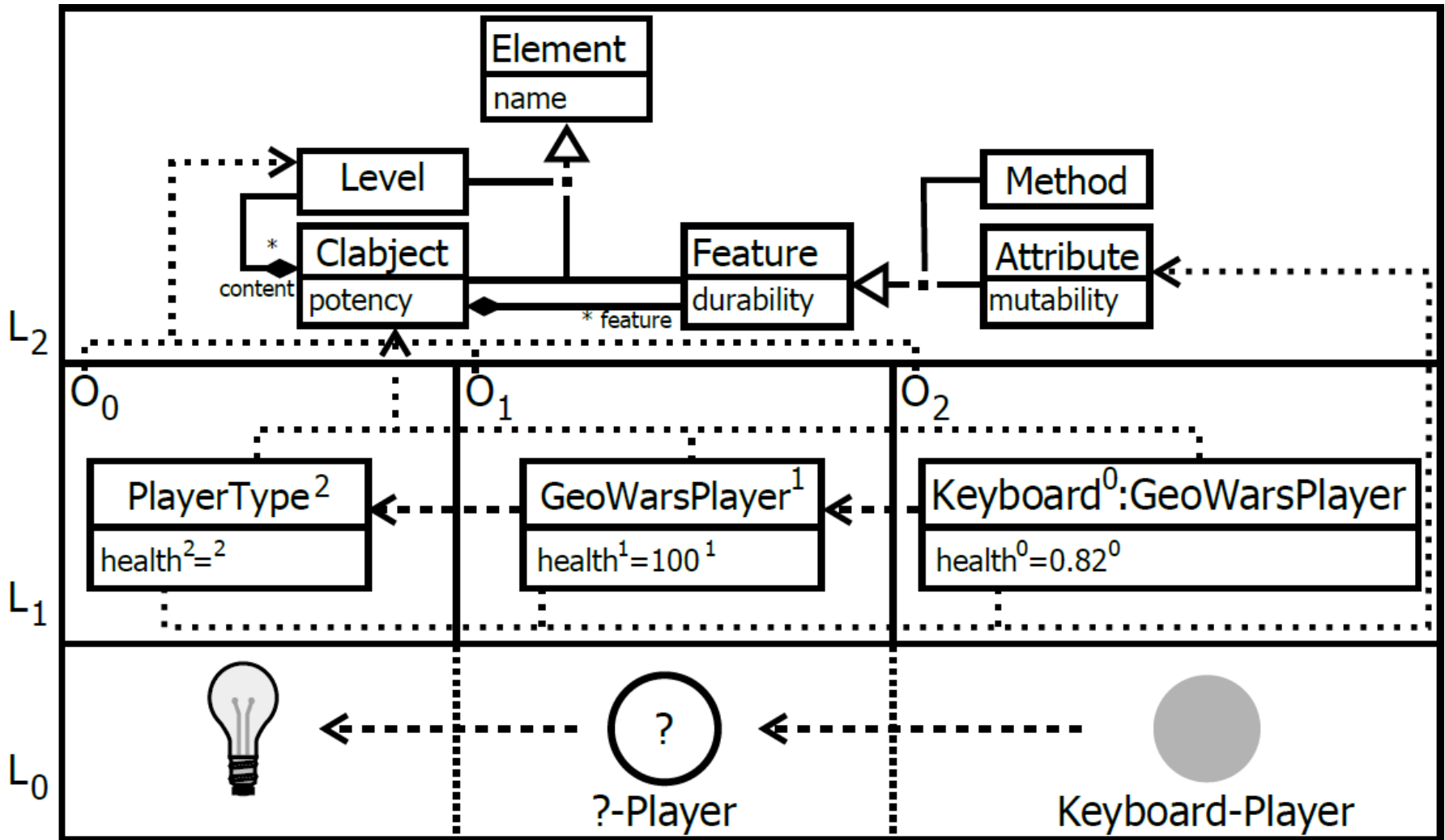
- Transforming the blueprint to a execution state model
- Applying the changes to all future executions of this model

## One Solution

- Copy the blueprint for each execution

## Problem

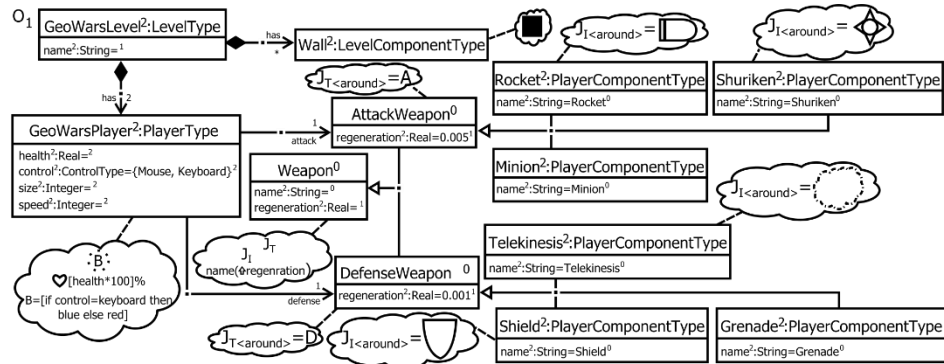
- Checking of conformance of a blueprint copy and the original blueprint is not covered by modeling frameworks
- Keep evolution of blueprint and all copies in synch



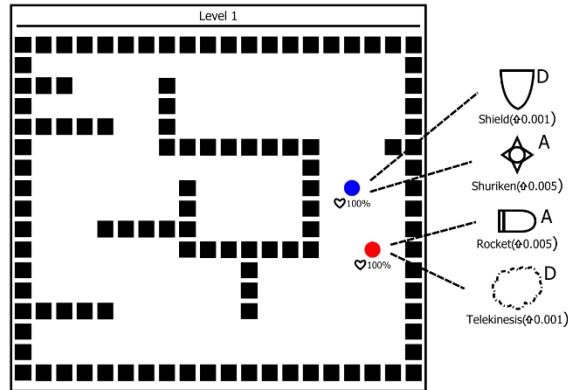
# The Deep Model Version



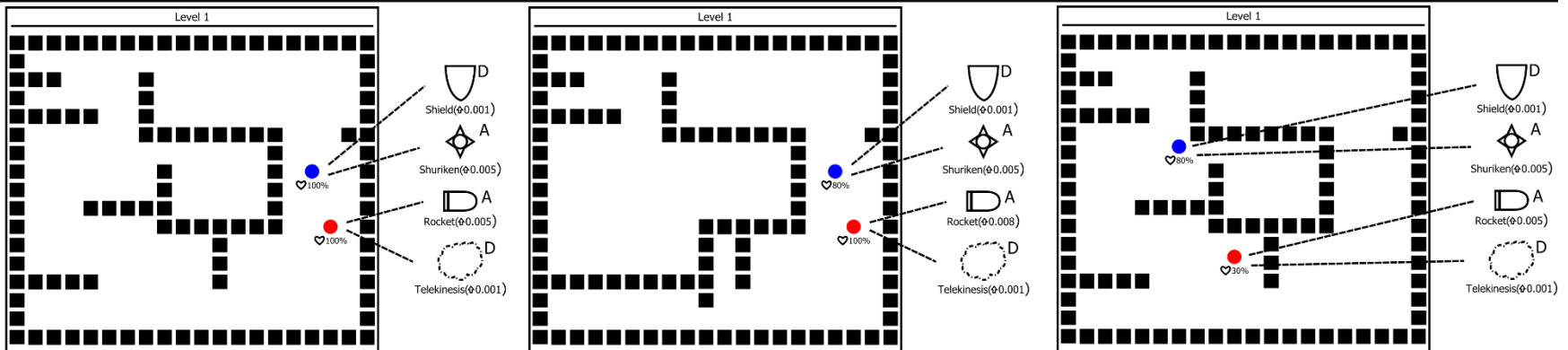
O<sub>1</sub>



O<sub>2</sub>



O<sub>3</sub>





<http://www.melanee.org/videos/>



- All scenarios presented here can be supported with additional coding effort today. But deep-modeling provides these features out of the box.
- The instance level can be used to ....
  - ... represent execution state information ...
  - ... allow modification at execution time ...
  - ... suspend / resume models ...
- ... in an optimal way.
- Semantics of models can be defined through
  - Deep ATL Transformations
  - Java Plug-ins
  - other ways have to be explored yet (e.g. deep action language)
- The next steps are
  - Test if fUML is implementable in Melanee and if it brings advantages over two-level technology
  - Provide a network protocol to query and manipulate deep models



- Visit us!
  - Melanee Webpage - <http://www.melanee.org>



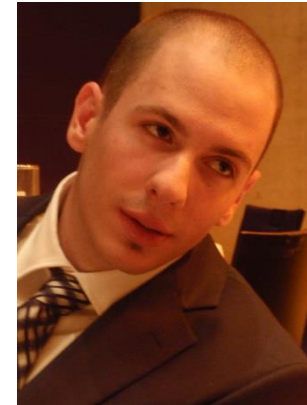
Colin Atkinson

atkinson@informatik.uni-mannheim.de



Ralph Gerbig

gerbig@informatik.uni-mannheim.de



Noah Metzger

nometzge@mail.uni-mannheim.de