

A large, stylized blue graphic of the letter 'G' on the left side of the slide. The 'G' is composed of a thick blue stroke, with a white negative space forming the inner curve and the bottom arm of the letter.

Reinforcement Learning

a gentle introduction & industrial application

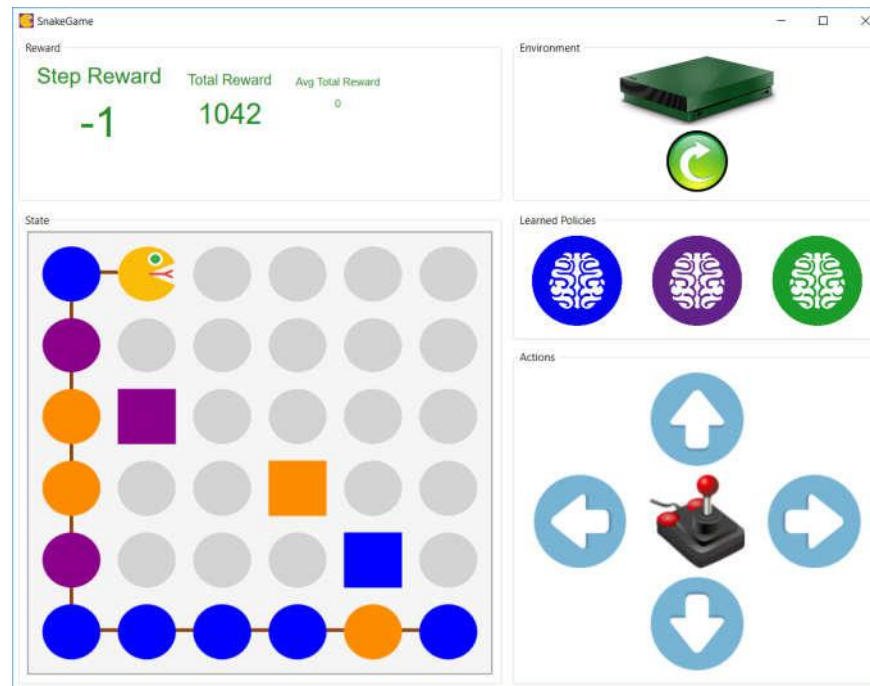
Christian Hidber

Questions: [slido.com](https://www.slido.com) [#bigdata2018](https://twitter.com/bigdata2018)

Learning learning from children



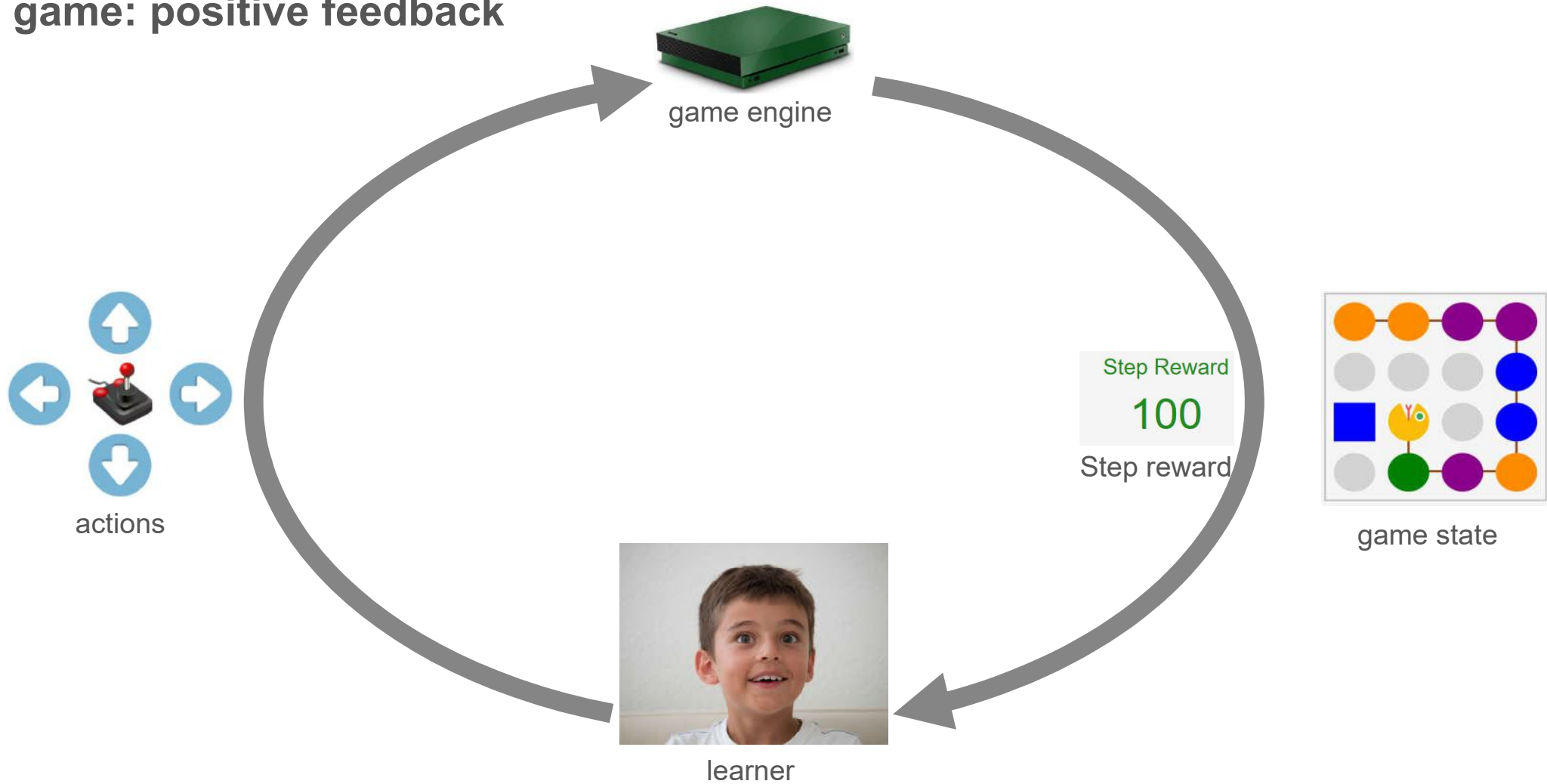
The game: demo



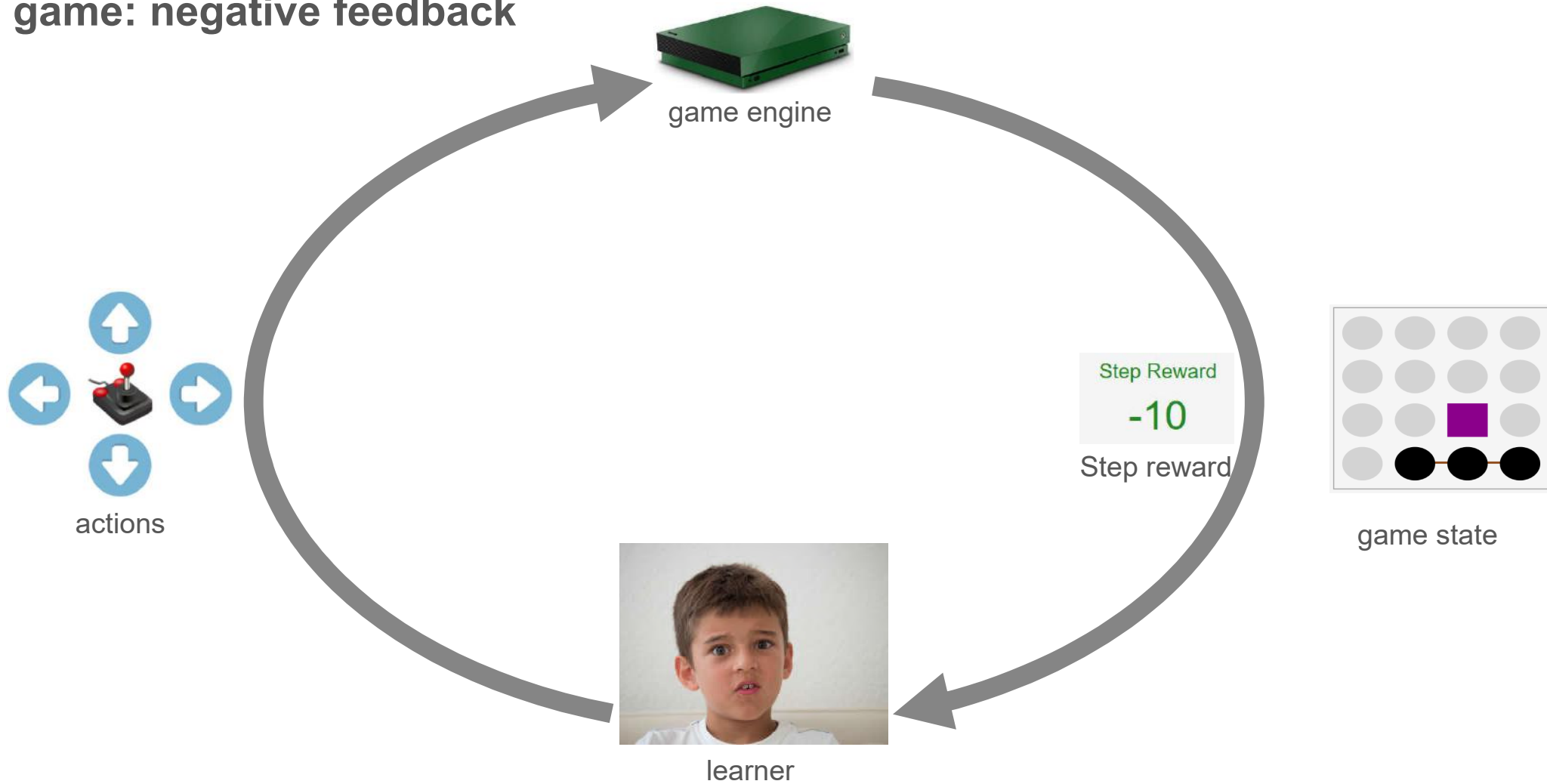
The game: setup



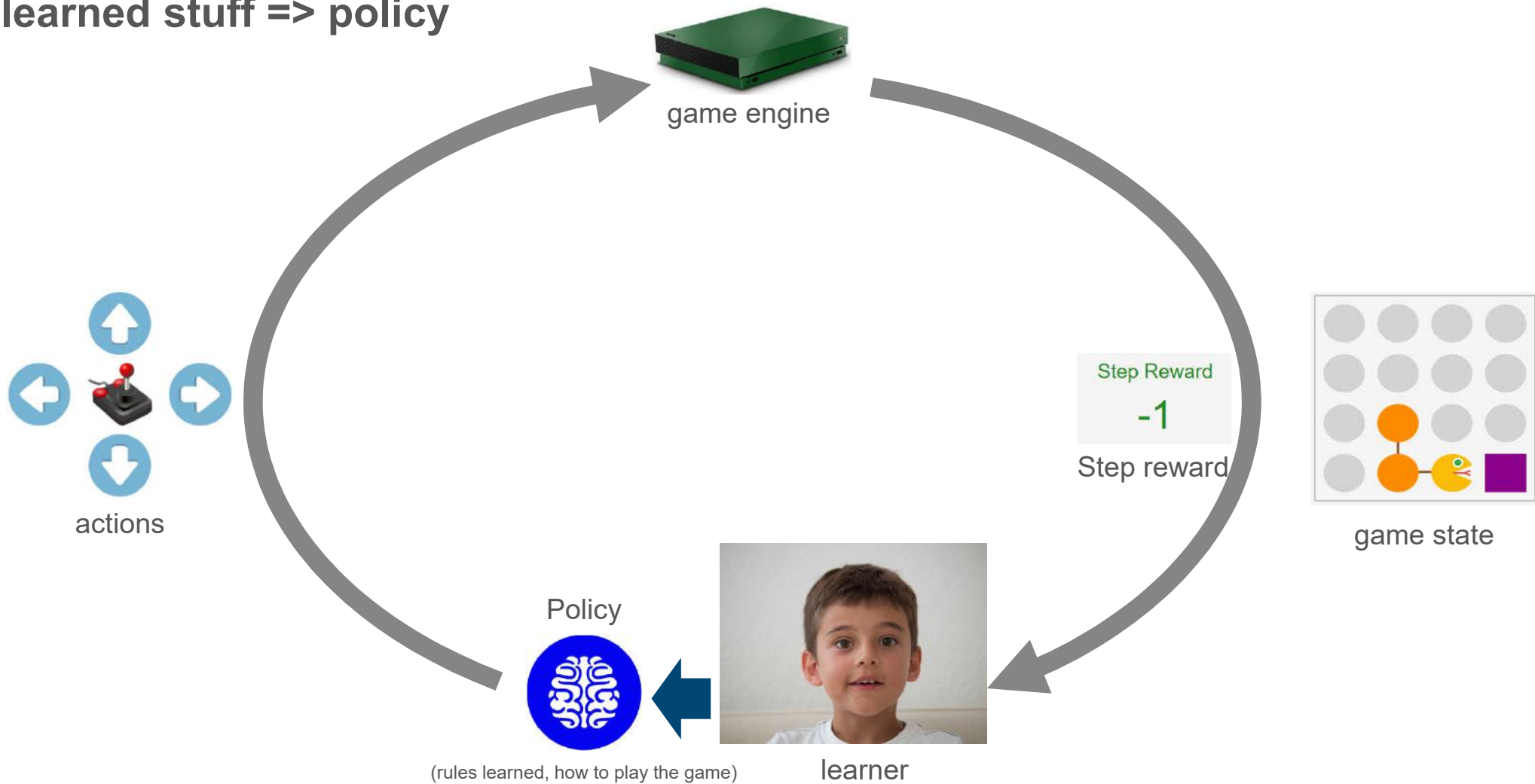
The game: positive feedback



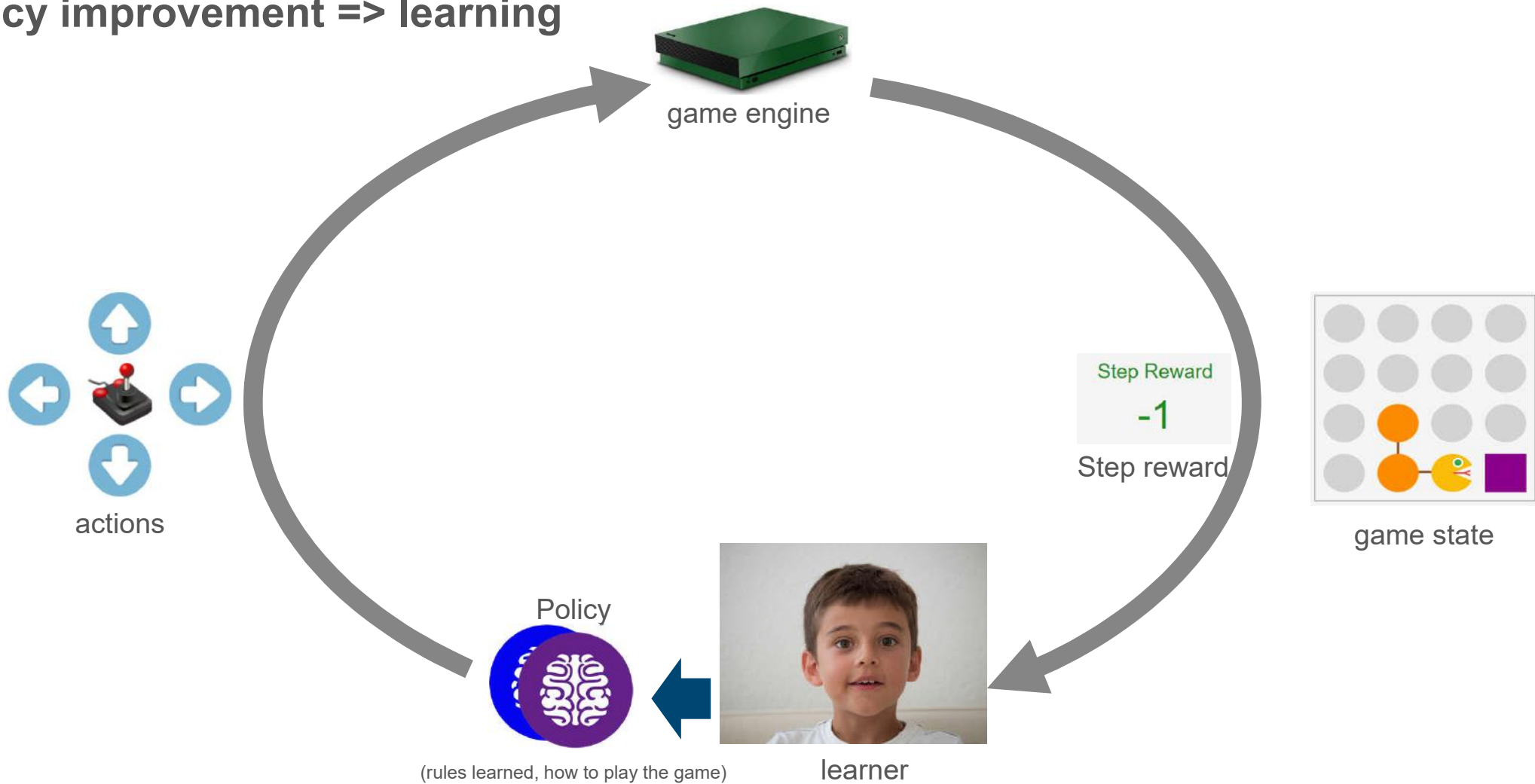
The game: negative feedback



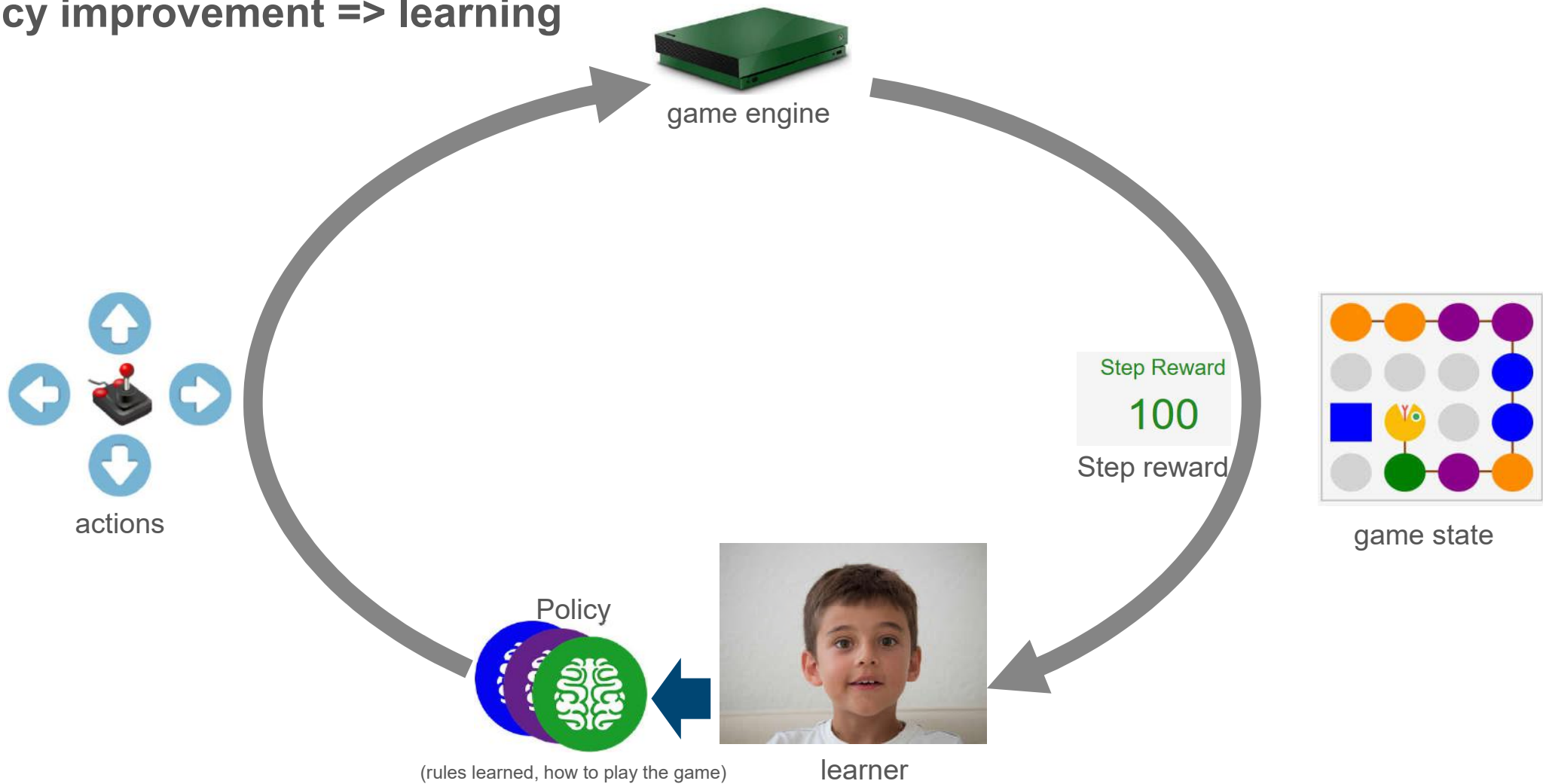
the learned stuff => policy



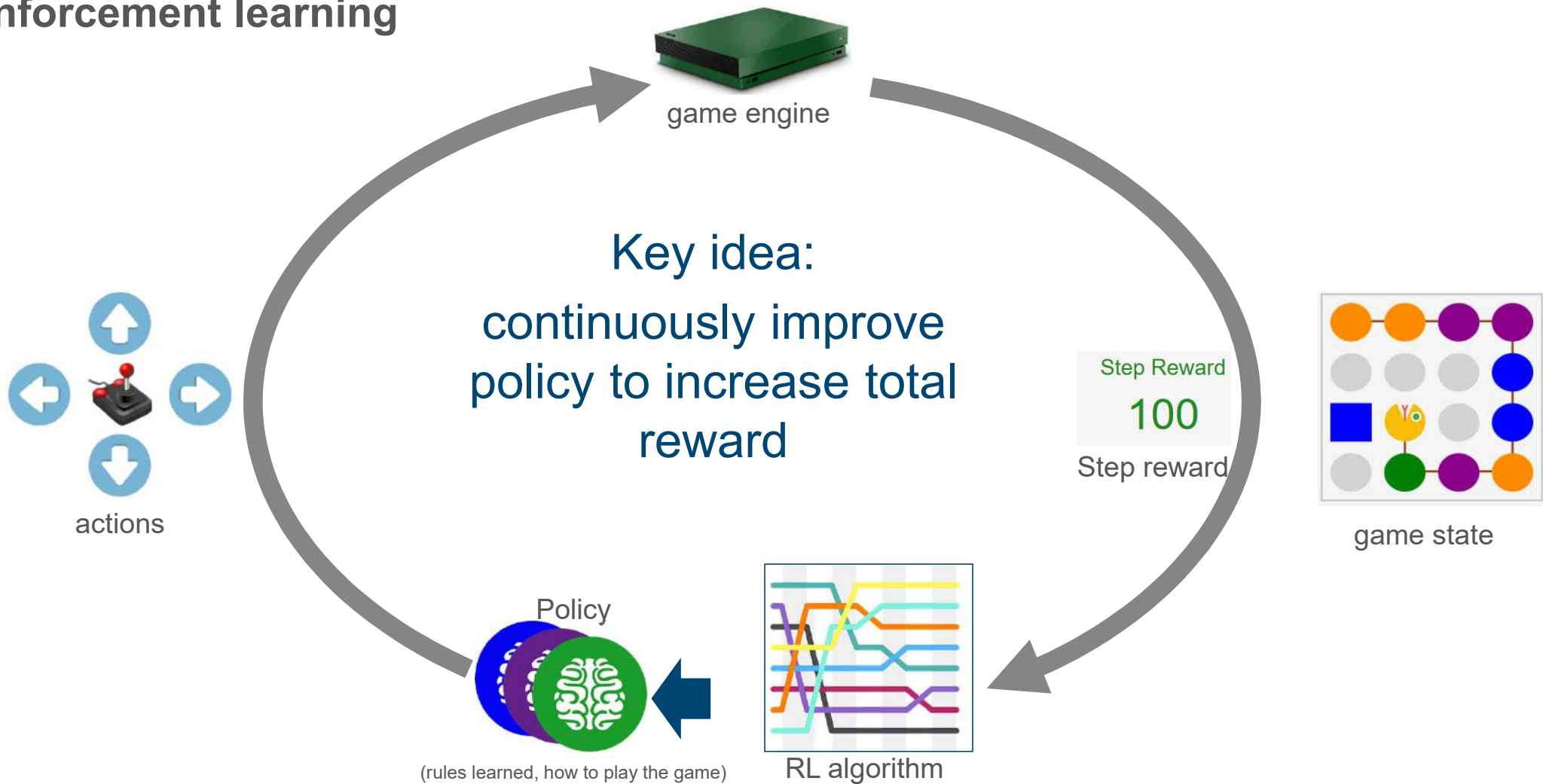
policy improvement => learning



policy improvement => learning

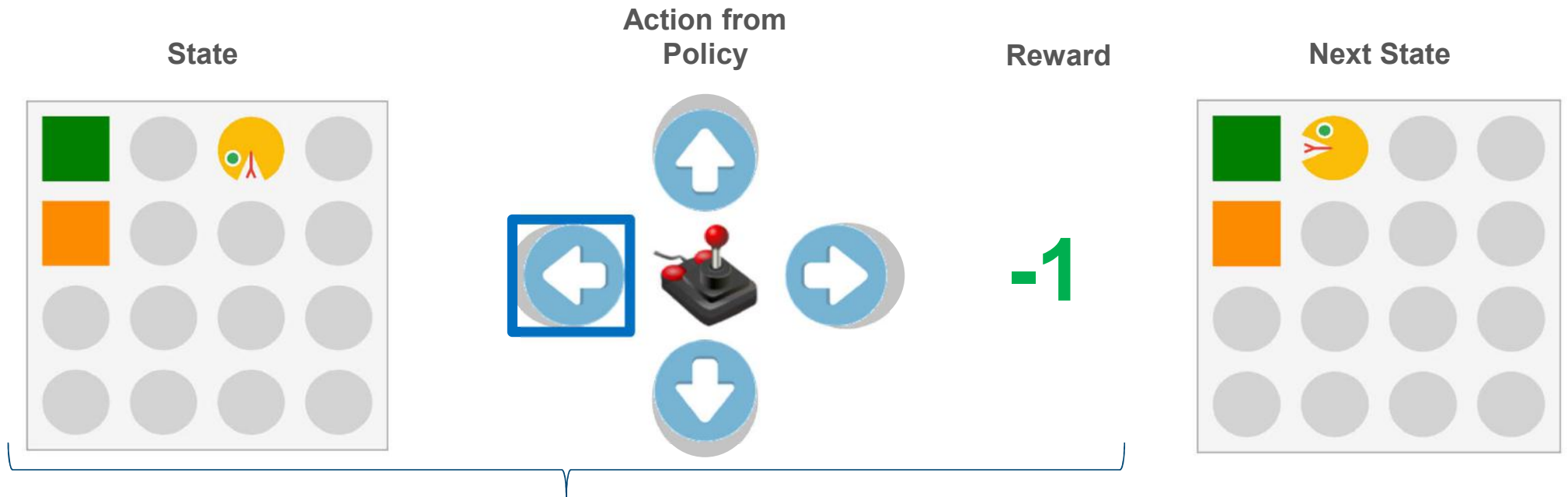


Reinforcement learning



Episode 1 : play with 1st policy (random)

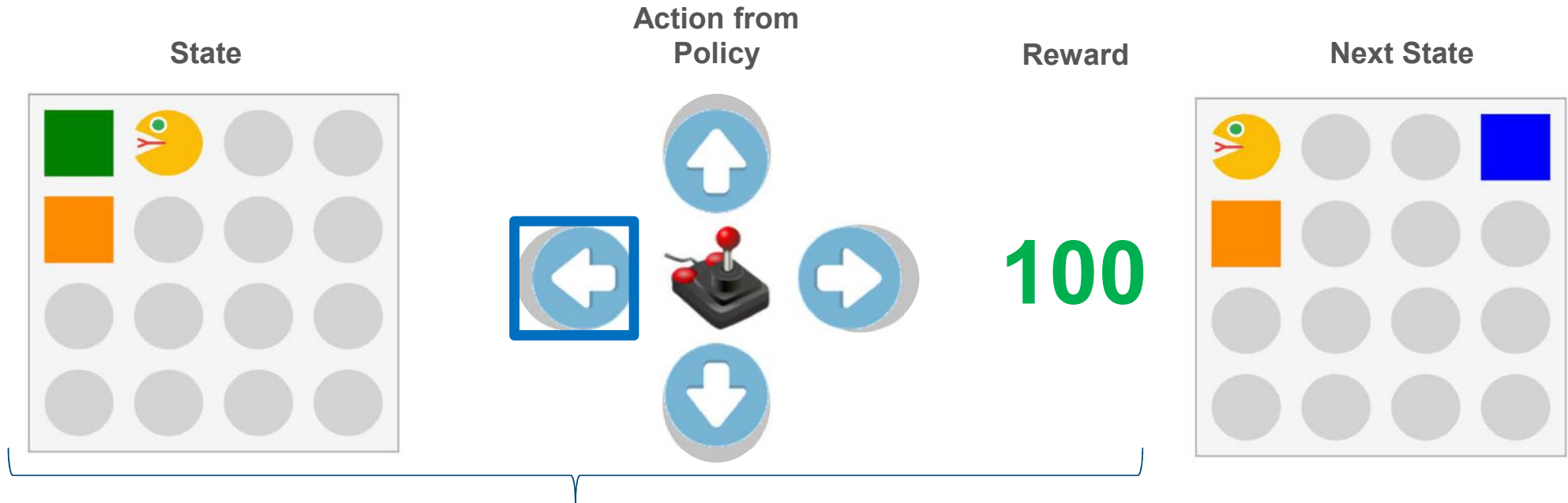

Policy
(rules learned, how to play the game)



1 2 3 4 5 6 7 Step #

Episode 1 : play with 1st policy (random)


Policy
(rules learned, how to play the game)

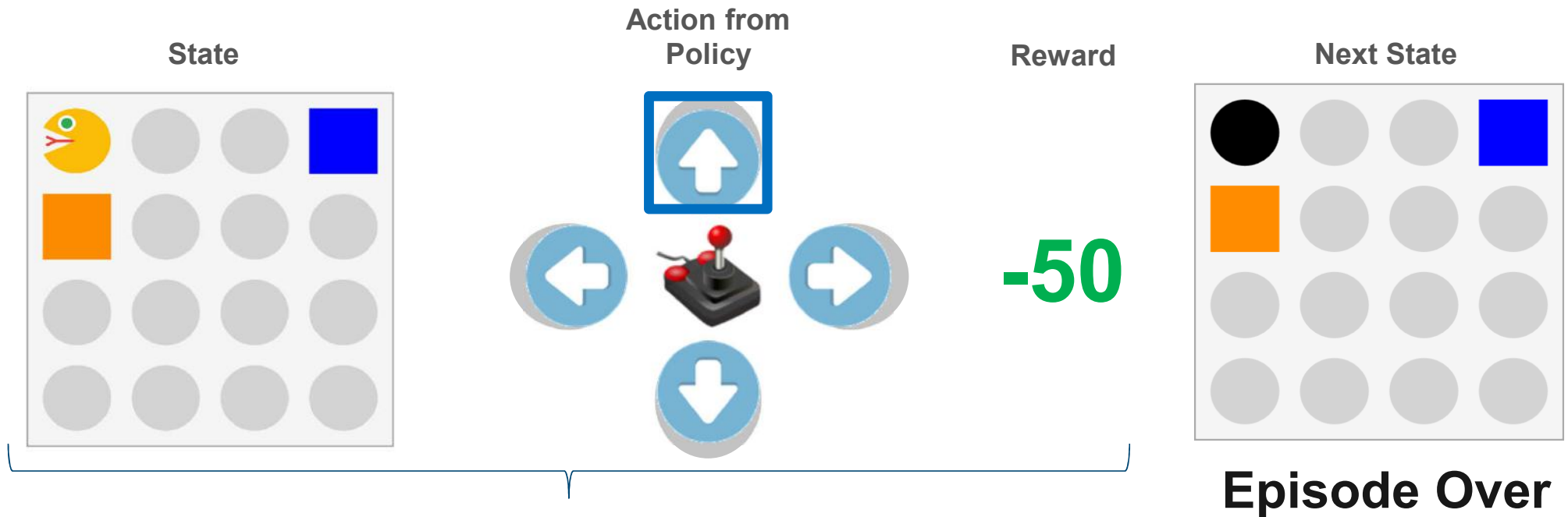


1 2 3 4 5 6 7 Step #

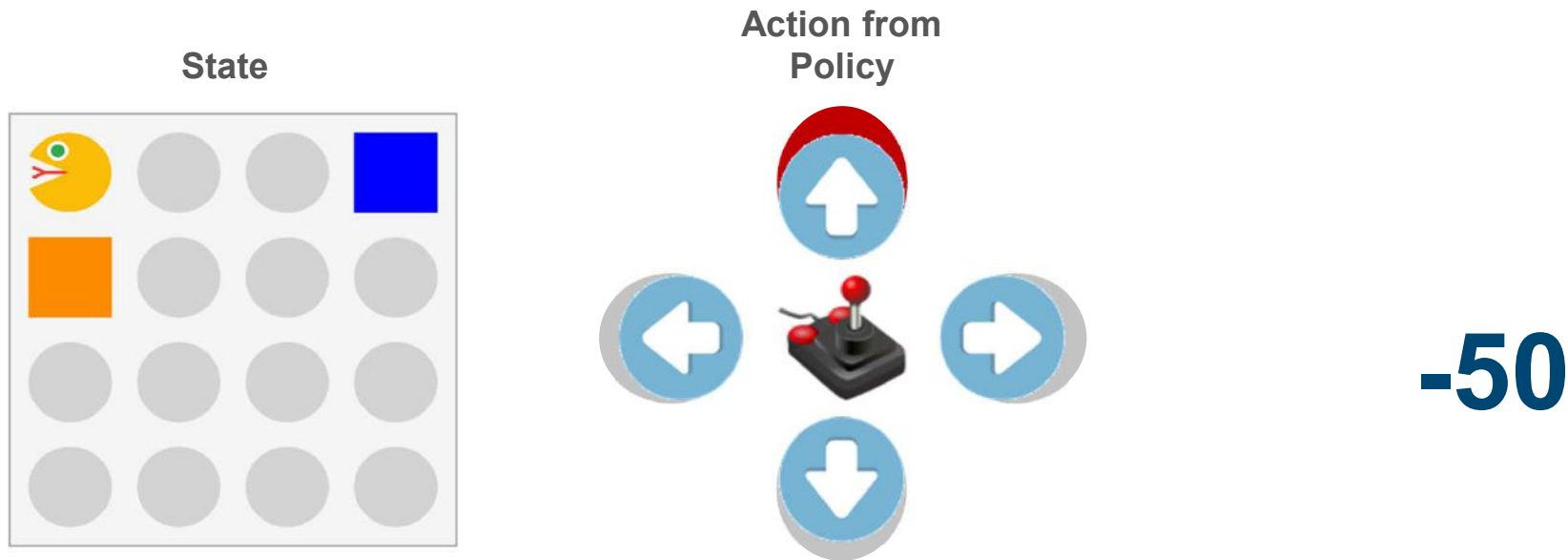


Episode 1 : play with 1st policy (random)


Policy
(rules learned, how to play the game)



Episode 1 : improve 1st policy for state in step 3



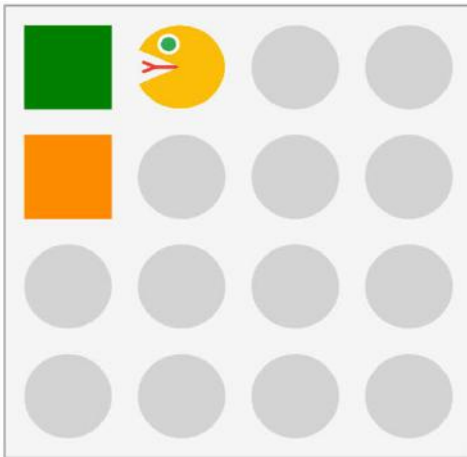


Policy

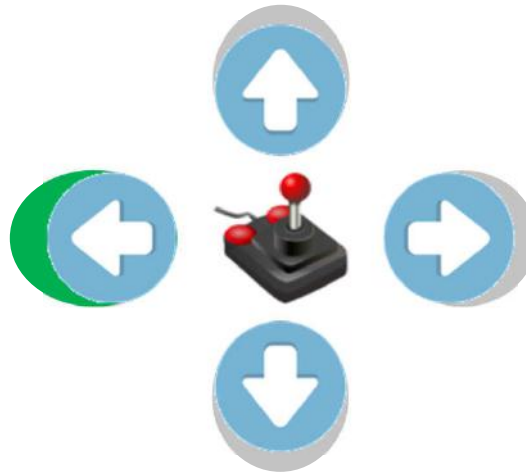
(rules learned, how to play the game)

Episode 1 : improve 1st policy for state in step 2

State



Action from Policy



50 (= +100 -50)

Future Reward

(sum of all rewards from current state until 'game over')



Episode Over

1

2

3

4

5

6

7 Step #

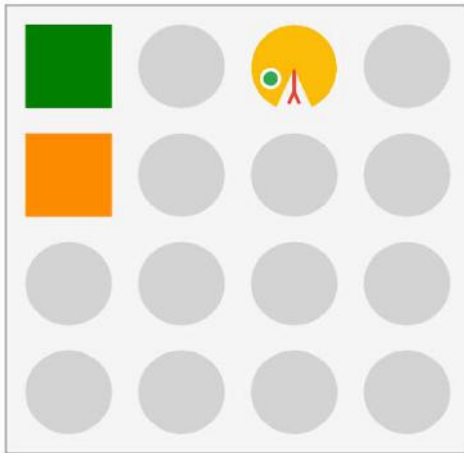


Policy

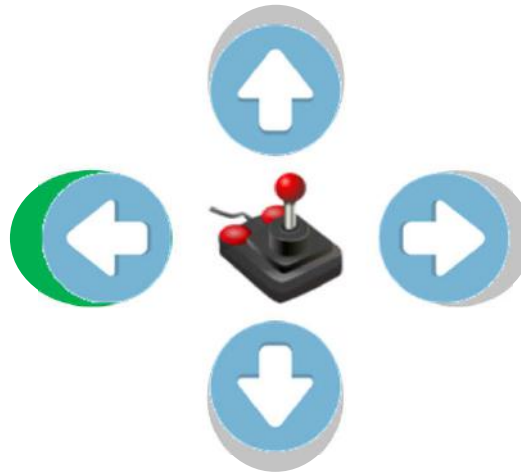
(rules learned, how to play the game)

Episode 1 : improve 1st policy for state in step 1

State



Action from Policy



49 (= -1 + 100 - 50)

Future Reward

(sum of all rewards from current state until 'game over')



Episode Over

1

2

3

4

5

6

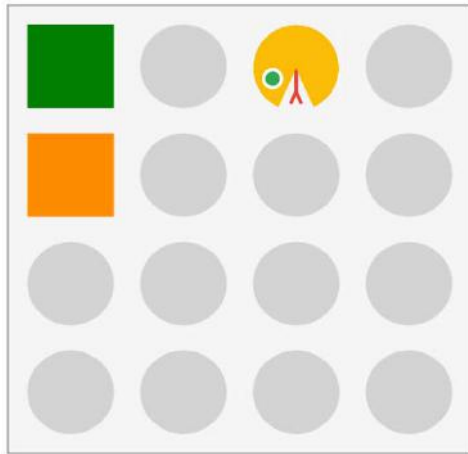
7 Step #

Episode 2 : play with 2nd policy

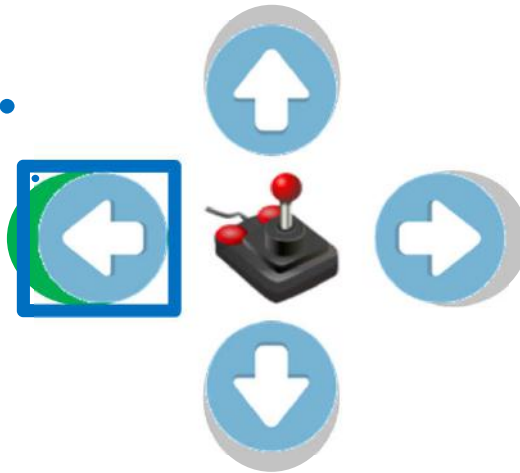

Policy
(rules learned, how to play the game)

Already learned:
go left is ok

State



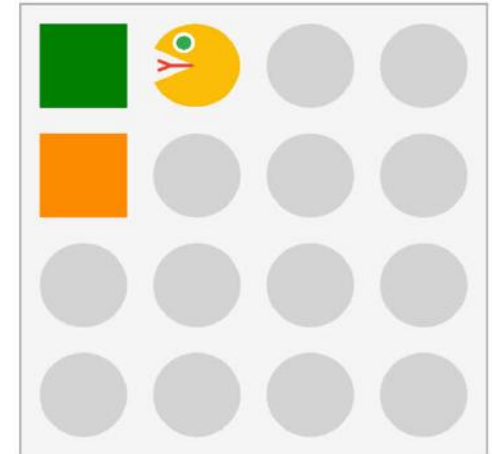
Action from
Policy



Reward

-1

Next State



1

2

3

4

5

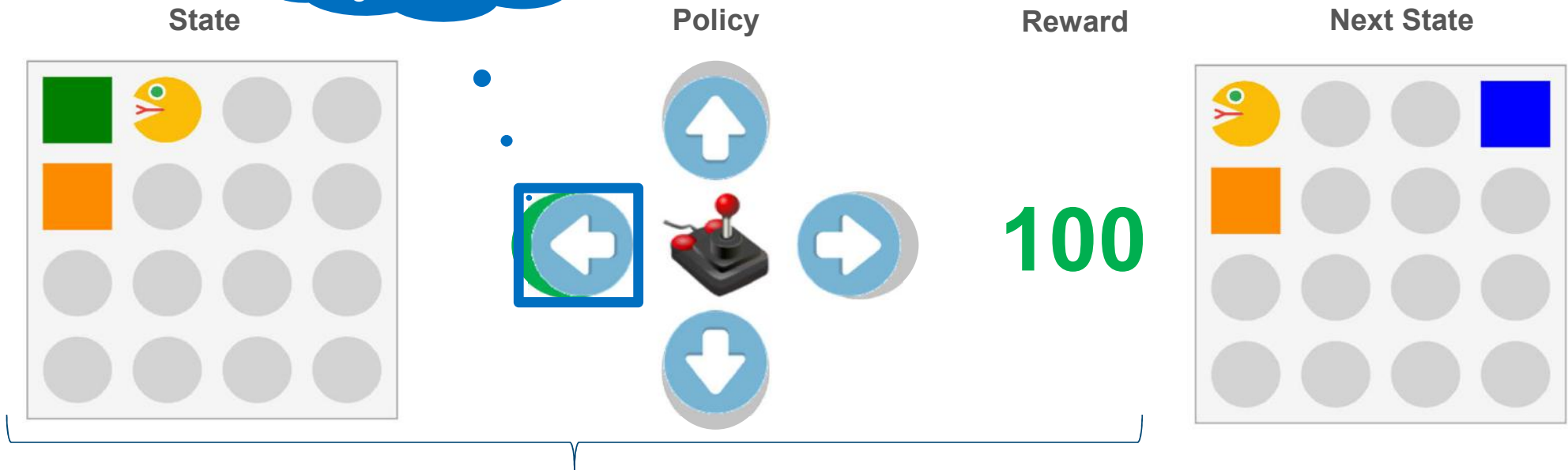
6

7 Step #

Episode 2 : play with 2nd policy


Policy
(rules learned, how to play the game)

Already learned:
go left is ok



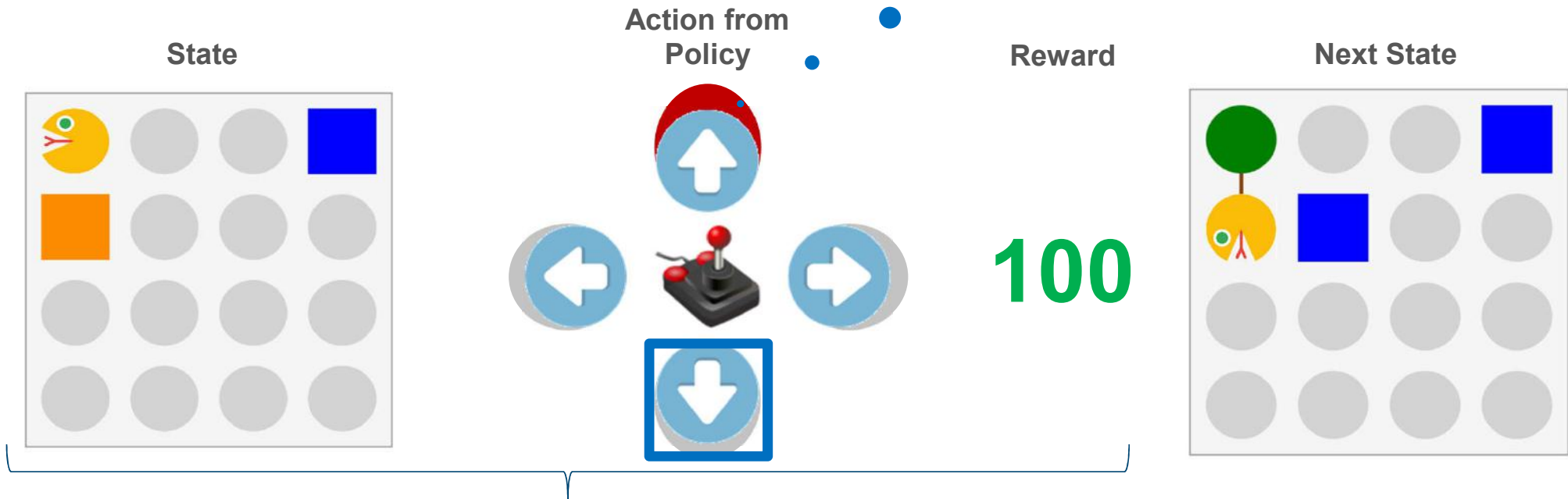
1 2 3 4 5 6 7 Step #



Episode 2 : play with 2nd policy

Already learned:
don't go up


Policy
(rules learned, how to play the game)

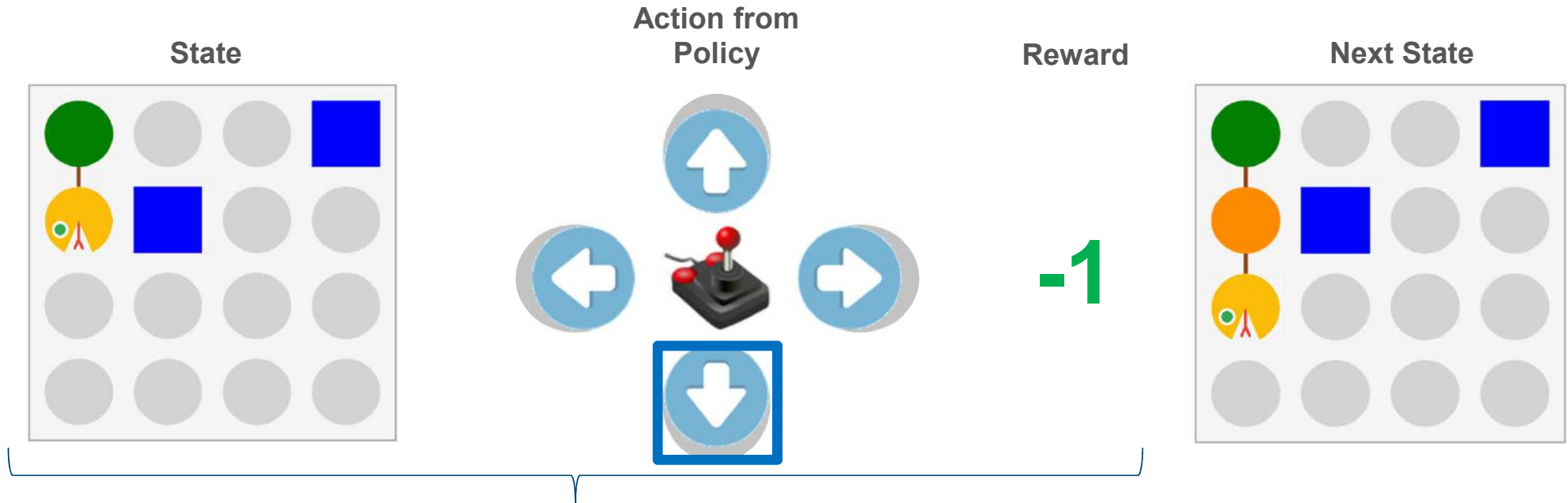


1 2 3 4 5 6 7 Step #



Episode 2 : play with 2nd policy


Policy
(rules learned, how to play the game)



1

2

3

4

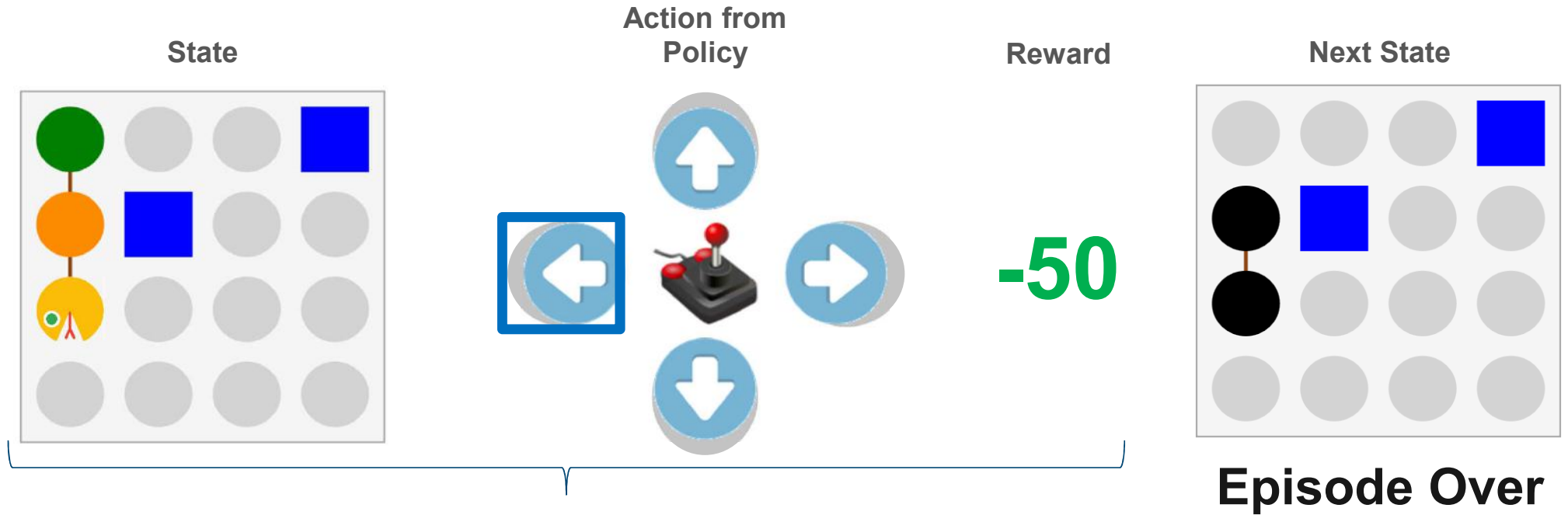
5

6

7 Step #

Episode 2 : play with 2nd policy


Policy
 (rules learned, how to play the game)



1 2 3 4 5 6 7 Step #

Episode Over



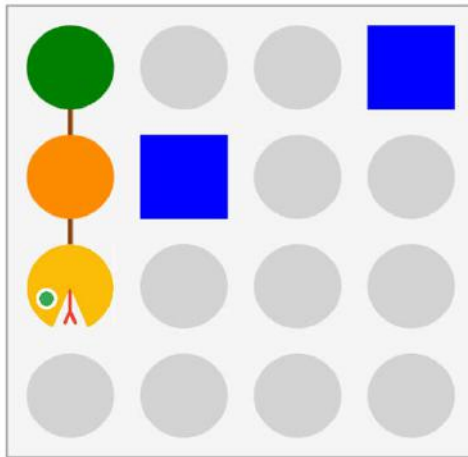


Policy

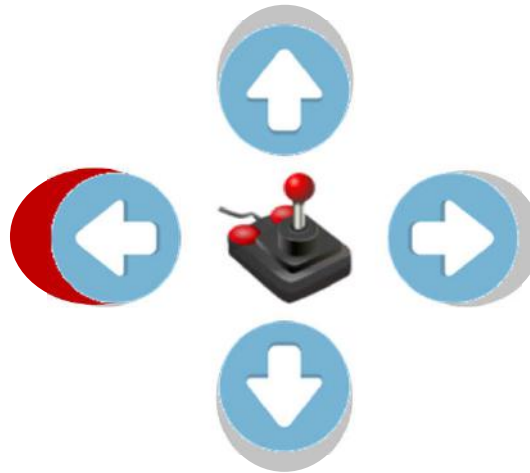
(rules learned, how to play the game)

Episode 2 : improve 2nd policy for state in step 5

State



Action from Policy



-50 (= -50)

Future Reward

(sum of all rewards from current state until 'game over')



1

2

3

4

5

6

7 Step #

Episode Over



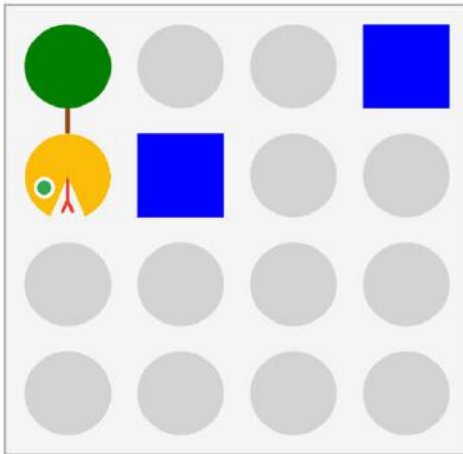


Policy

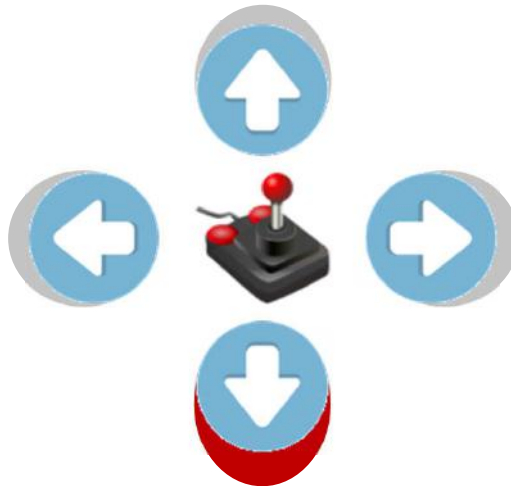
(rules learned, how to play the game)

Episode 2 : improve 2nd policy for state in step 4

State



Action from Policy



-51 (= -1 - 50)

Future Reward

(sum of all rewards from current state until 'game over')



1

2

3

4

5

6

7 Step #

Episode Over



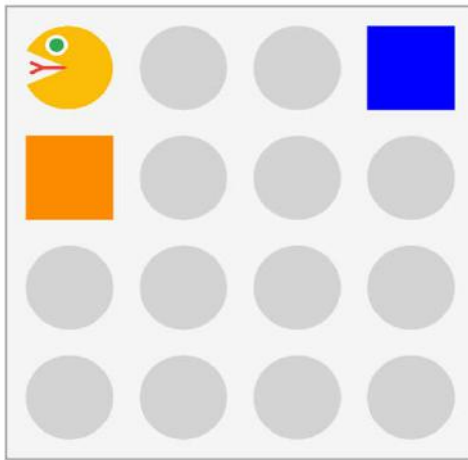


Policy

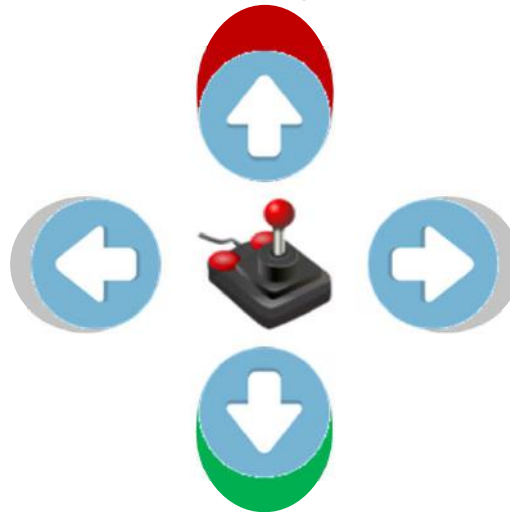
(rules learned, how to play the game)

Episode 2 : improve 2nd policy for state in step 3

State



Action from Policy



49 (= +100 - 1 - 50)

Future Reward

(sum of all rewards from current state until 'game over')



1

2

3

4

5

6

7 Step #

Episode Over

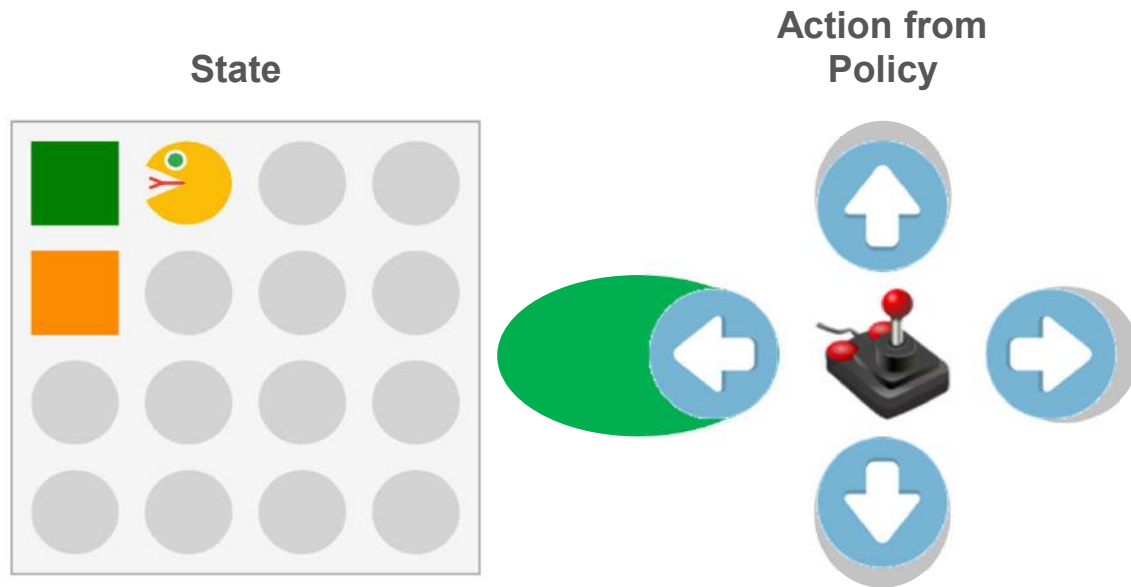




Policy

(rules learned, how to play the game)

Episode 2 : improve 2nd policy for state in step 2



149

(=+100+100-1-50)

Future Reward

(sum of all rewards from current state until 'game over')



1

2

3

4

5

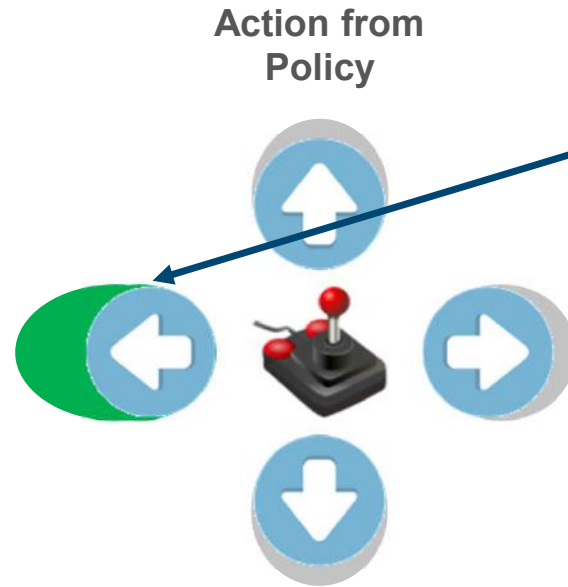
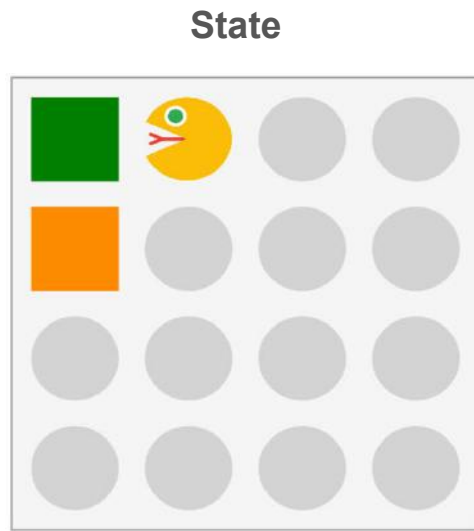
6

7 Step #

Episode Over



Episode 2 : improve 2nd policy for state in step 2



= some running average of old and new value

149 (=+100+100-1-50)

Future Reward

(sum of all rewards from current state until 'game over')



1

2

3

4

5

6

7 Step #

Episode Over

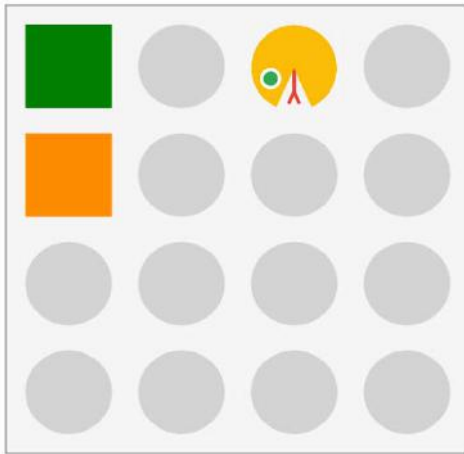


Policy

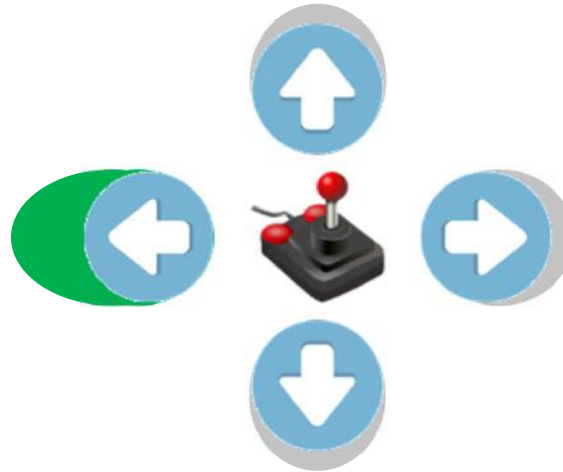
(rules learned, how to play the game)

Episode 2 : improve 2nd policy for state in step 1

State



Action from Policy



148 (= -1 + 100 + 100 - 1 - 50)

Future Reward

(sum of all rewards from current state until 'game over')



1

2

3

4

5

6

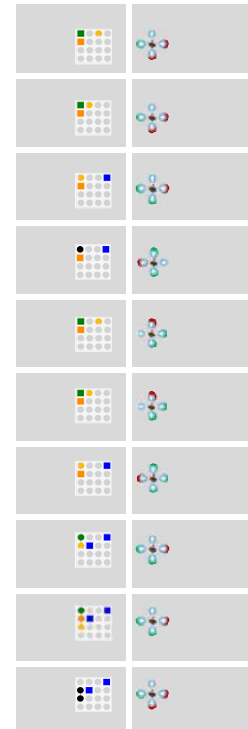
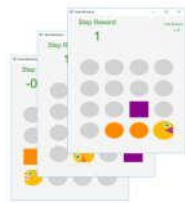
7 Step #

Episode Over

So far



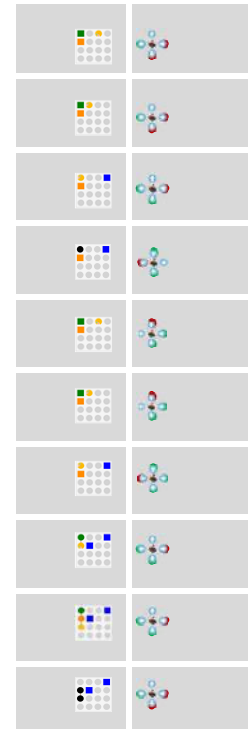
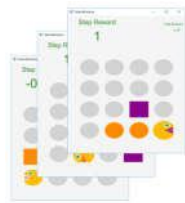
a policy is a map from states to action probabilities



...updated by the reinforcement learning algorithm



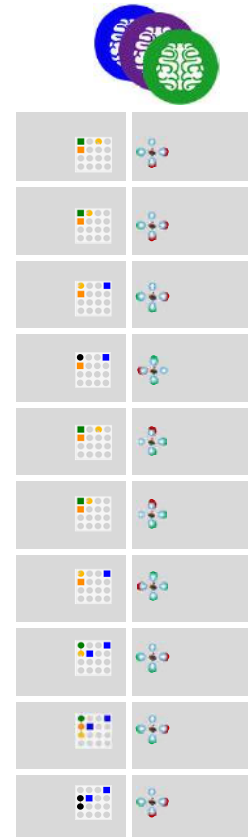
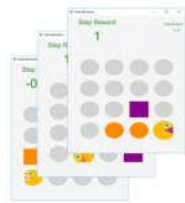
a policy is a map from states to action probabilities



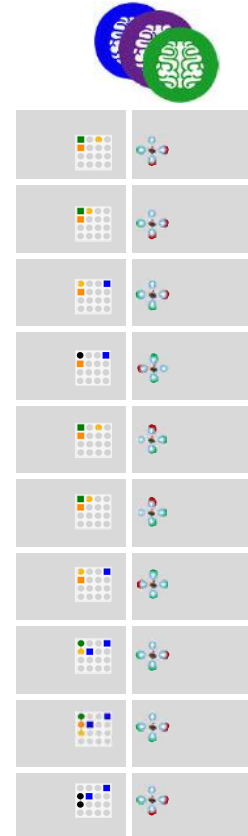
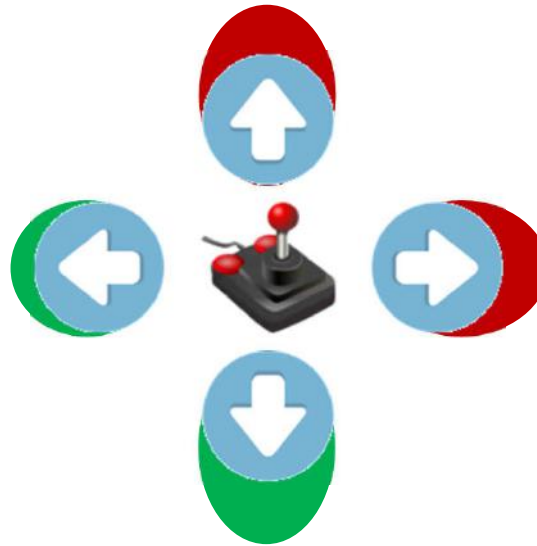
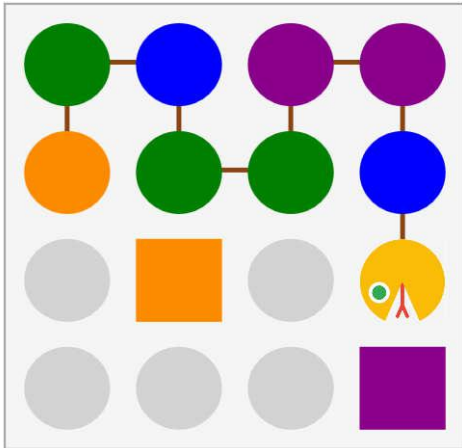
...updated by the reinforcement learning algorithm



a policy is a map from states to action probabilities



After many, many episodes, for each state...





Policy

(rules learned, how to play the game)

Algorithm sketch

Initialize **table** with random action probabilities for each **state**

Repeat

play episode with policy given by **table**

Record $(state_1, action_1, reward_1), (state_2, action_2, reward_2), \dots$ for episode

For each step i

compute $FutureReward_i = reward_i + reward_{i+1} + \dots$

update **table**[state _{i}] s.t.

- **action _{i}** becomes for state _{i} more likely if $FutureReward_i$ is “high”
- **action _{i}** becomes for state _{i} less likely if $FutureReward_i$ is “low”

Algorithm sketch



Policy

(rules learned, how to play the game)

Initialize **table** with random action probabilities for each **state**

Repeat

play episode with policy given by **table**

Record $(state_1, action_1, reward_1), (state_2, action_2, reward_2), \dots$ for episode

For each step i

compute $FutureReward_i = reward_i + reward_{i+1} + \dots$

update **table**[state _{i}] s.t.

- **action _{i}** becomes for state _{i} more likely if $FutureReward_i$ is “high”
- **action _{i}** becomes for state _{i} less likely if $FutureReward_i$ is “low”

Algorithm sketch



Policy

(rules learned, how to play the game)

Initialize **table** with random action probabilities for each **state**

Repeat

play episode with policy given by **table**

Record $(state_1, action_1, reward_1), (state_2, action_2, reward_2), \dots$ for episode

For each step i

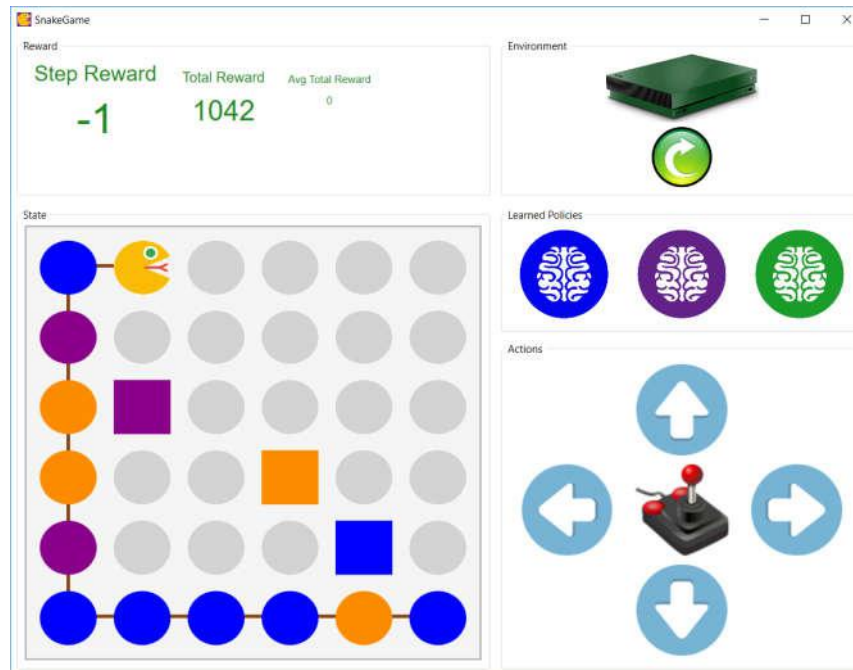
compute $FutureReward_i = reward_i + reward_{i+1} + \dots$

update **table**[$state_i$] s.t.

- $action_i$ becomes for $state_i$ more likely if $FutureReward_i$ is “high”
- $action_i$ becomes for $state_i$ less likely if $FutureReward_i$ is “low”



The game: demo



The bad news: nice idea, but...



The bad news: nice idea, but...

too many states... too many actions

- Too much memory needed
- Too much time





Policy

(rules learned, how to play the game)

The solution

Idea:

Replace lookup table with a **neural network** that approximates the action probabilities contained in the table

Instead of

Table[state] = action probabilities

Do

NeuralNet(state) ~ action probabilities

```
Initialize table with action probabilities for each state
Repeat
  play episode with policy given by table
  Record (state1,action1,reward1),(state2,action2,reward2),... for episode
  For each step i
    compute FutureRewardi = rewardi + rewardi+1 + ...
    update table[statei] ←
      • actioni becomes for statei more likely if FutureRewardi is "high"
      • actioni becomes for statei less likely if FutureRewardi is "low"
```

Change to “play episode with policy given by **NeuralNet**”

Change to “update **weights** of NeuralNet”

Neural nets to the rescue



Policy

(rules learned, how to play the game)

Idea:

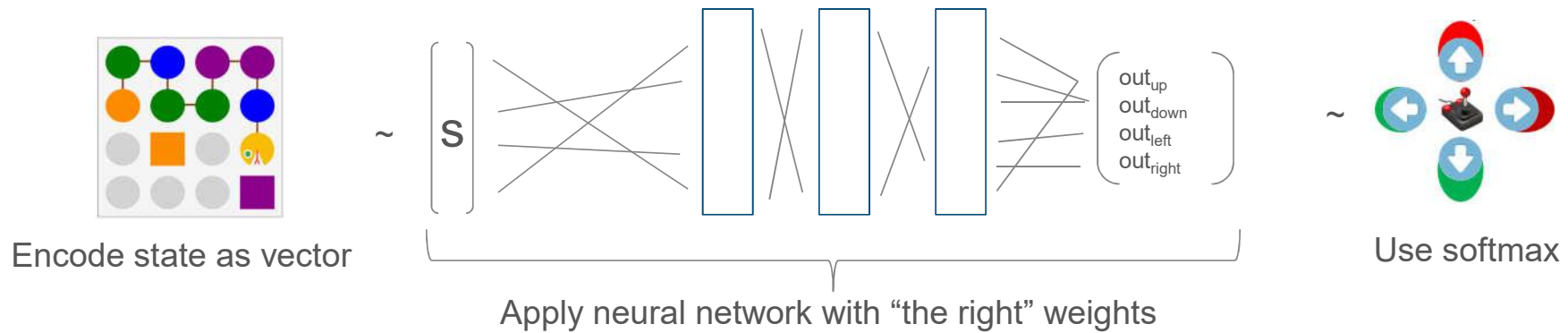
Replace lookup table with a neural network that approximates the action probabilities contained in the table

Instead of

Table[state] = action probabilities

Do

NeuralNet(state) ~ action probabilities





Policy

(rules learned, how to play the game)

Policy Gradient Algorithm sketch

Initialize neuralNet with random weights W

Repeat

play episode(s) with policy given by weights W

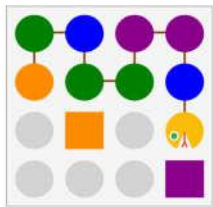
Record $(state_1, action_1, reward_1), (state_2, action_2, reward_2), \dots$ for episode(s)

For each step i

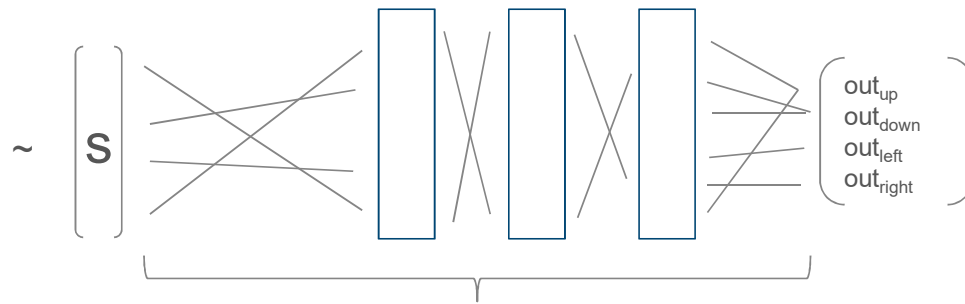
compute $FutureReward_i = reward_i + reward_{i+1} + \dots$

Update weights W

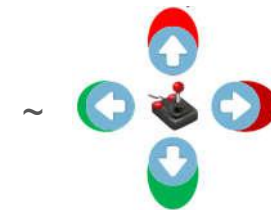
$$W = W + \text{????}$$



Encode state as vector



Weights W



Use softmax



Policy Gradient Algorithm sketch



Policy

(rules learned, how to play the game)

Initialize **neuralNet** with random **weights W**

Repeat

play episode(s) with policy given by **weights W**

Record $(state_1, action_1, reward_1), (state_2, action_2, reward_2), \dots$ for episode(s)

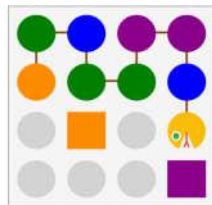
For each step i

compute $FutureReward_i = reward_i + reward_{i+1} + \dots$

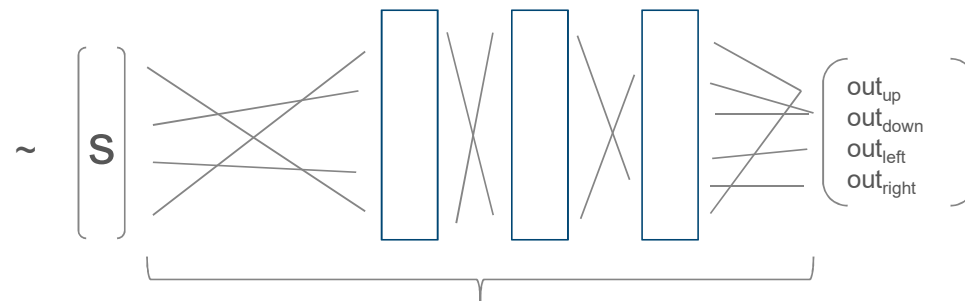
Update **weights W**

$W = W +$

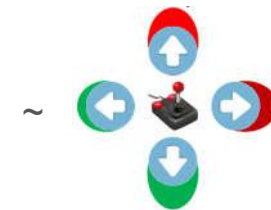
????



Encode state as vector



Weights W



Use softmax



Policy Gradient Algorithm sketch

Initialize neuralNet with random weights W

Repeat

play episode(s) with policy given by weights W

Record $(state_1, action_1, reward_1), (state_2, action_2, reward_2), \dots$ for episode(s)

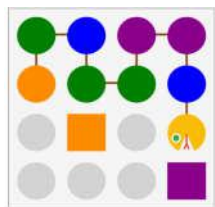
For each step i

compute $FutureReward_i = reward_i + reward_{i+1} + \dots$

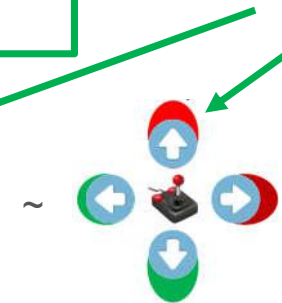
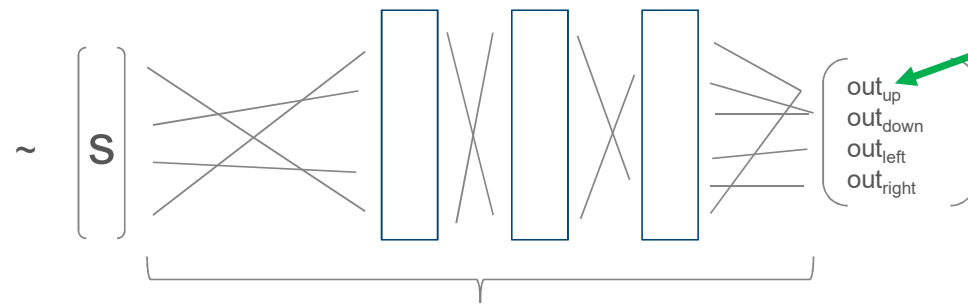
Update weights W

$W = W +$????

Increases out_i



Encode state as vector





Policy Gradient Algorithm sketch

Initialize **neuralNet** with random **weights W**

Repeat

play episode(s) with policy given by **weights W**

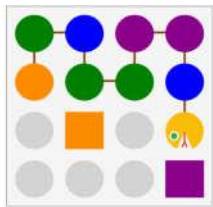
Record $(state_1, action_1, reward_1), (state_2, action_2, reward_2), \dots$ for episode(s)

For each step i

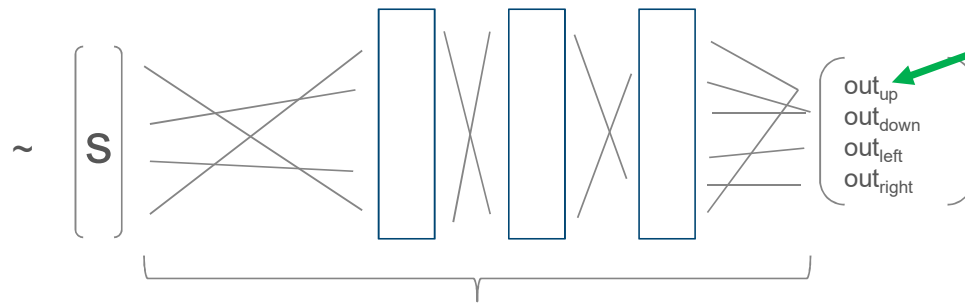
compute $FutureReward_i = reward_i + reward_{i+1} + \dots$

Update **weights W**

$$W = W + \underbrace{\alpha}_{\text{Learning rate}} * FutureReward_i * \underbrace{Gradient_W (neuralNet_W(state_i, action_i))}_{\text{Increases out}_i}$$



Encode state as vector



Weights W



Use softmax



Policy Gradient

$$W = \underbrace{\arg \max_W}_{\text{this is trouble}} \underbrace{E_{\tau \sim p_W}[R(\tau)]}_{\text{expected total reward playing with } W}$$

$= f(W)$

$$W_{k+1} = W_k + \alpha \cdot \nabla_W f(W_k)$$

$$\tau = (s_1, a_1, r_1), (s_2, a_2, r_2), \dots$$

$$R(\tau) = \sum_i r_i$$

p_W = episode probability given the policy defined by the NeuralNet with weights W



Policy Gradient

$$W = \underbrace{\arg \max_W}_{\text{this is trouble}} \underbrace{E_{\tau \sim p_W}[R(\tau)]}_{\text{expected total reward playing with } W}$$

$= f(W)$

$$\tau = (s_1, a_1, r_1), (s_2, a_2, r_2), \dots$$

$$R(\tau) = \sum_i r_i$$

p_W = episode probability given the policy defined by the NeuralNet with weights W

$$W_{k+1} = W_k + \alpha \cdot \underbrace{\nabla_W E_{\tau \sim p_W}[R(\tau)]}_{\text{this is the new trouble}}$$

$$\begin{aligned} \nabla_W E_{\tau \sim p_W}[R(\tau)] &= \nabla_W \int_{\tau} p_W(\tau) \cdot R(\tau) &&= \int_{\tau} \nabla_W p_W(\tau) \cdot R(\tau) \\ &= \int_{\tau} p_W(\tau) \cdot \frac{\nabla_W p_W(\tau)}{p_W(\tau)} \cdot R(\tau) &&= \int_{\tau} p_W(\tau) \cdot \nabla_W \log p_W(\tau) \cdot R(\tau) \\ &= \underbrace{E_{\tau \sim p_W}[\nabla_W \log p_W(\tau) \cdot R(\tau)]}_{\text{good news}} \end{aligned}$$

$$\frac{\nabla_x f(x)}{f(x)} = \nabla_x \log f(x)$$



Policy Gradient

$$W = \underbrace{\arg \max_W}_{\text{this is trouble}} \underbrace{E_{\tau \sim p_W}[R(\tau)]}_{\text{"average" total reward playing with W}}$$

$$\tau = (s_1, a_1, r_1), (s_2, a_2, r_2), \dots$$

$$R(\tau) = \sum_i r_i$$

p_W = episode probability given the policy defined by the NeuralNet with weights W

$$W_{k+1} = W_k + \alpha \cdot \underbrace{\nabla_W E_{\tau \sim p_W}[R(\tau)]}_{\text{this is the new trouble}}$$

$$W_{k+1} = W_k + \alpha \cdot E_{\tau \sim p_W} \left[\underbrace{\nabla_W \log p_W(\tau)}_{\sum_i \nabla_W \text{NeuralNet}_W(s_i, a_i)} \cdot \underbrace{R(\tau)}_{\text{FutureReward}_i} \right]$$

$$W = W + \alpha * \text{Gradient}_W (\text{neuralNet}_W(\text{state}_i, \text{action}_i)) * \text{FutureReward}_i$$

What for ?



no feasible, deterministic algorithm



What for ?



Traditional
Heuristics

Classic
Machine
Learning

Reinforcement
Learning

Automatic solution found in 93.4%

The challenges



manage the water level on the roof
control & steer the water flow
find the right dimensions
save & reliable





Finding the right dimensions



Finding the „right“ dimensions: demo

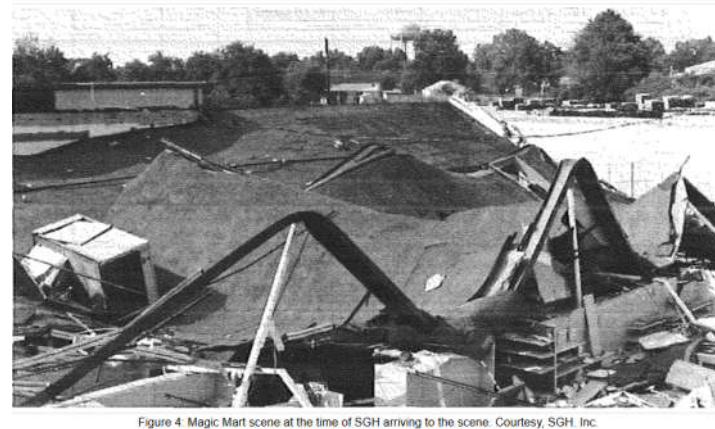
The screenshot displays the GEberit ProPlanner 2018 R2 software interface. The main window shows a hydraulic network diagram with various pipe segments and nodes. An inset image shows a 3D architectural rendering of a building. Below the diagram is a data table for the hydraulic network.

Typ	IS	Q [l/s]	L [m]	RI [m]	V Start [m]	V Ende [m]	g in [m/s²]	g out [m/s²]	q [l/s]	q [l/s]	Zula	Ab-Z [m]
S	1	1.00	1.00	204.0	206.8	0	0	3.7	94	1.0	47	
F	2	315	1.00	1.00	204.0	206.8	-255	2	3.7	94	0.3	51
S	3	315	50.00	50.00	204.0	206.8	-142	-255	3.1	94	0.3	113
S	4	315	6.36		102.0	106.1	-101	-107	1.5	87	0.3	7
S	5	315	3.50		102.0	106.1	-95	-101	1.5	87	0.3	6
S	6	315	12.00		102.0	106.1	-85	-95	1.3	87	0.3	10
S	7	315	13.00		10.8	70.7	-75	-81	1.2	87	0.0	3
S	8	250	15.00		61.2	56.3	-47	-79	1.4	87	0.1	11
S	9	250	15.00		40.8	38.9	-57	-42	0.8	89	0.0	4

On the right side of the interface, there is a product image of a GEberit ProPlanner 2018 R2 component, a 3D model of a pipe fitting, and a small 3D coordinate system icon.

What if...

- Collapsing pipes
- Collapsing roofs
- Clogged pipes
- Façade damages

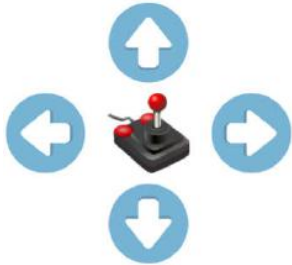


Turning the problem into a game



Designing the Action-Space

Snake game



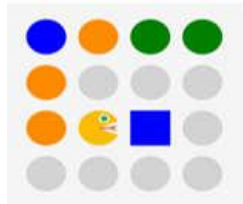
Roof drainage systems



- What actions would a human expert like to have ?
- Are these actions sufficient ?
- Would more / other actions be helpful ?
- Can we drop any actions ?

Designing the State-Space

Snake game



Roof drainage systems

Type	TS	d [mm]	L [m]	H [m]	V Set [l/s]	Q [l/s]	p in [mbar]	p out [mbar]	p at [mbar]	p dm [mbar]	v [m/s]	W [%]	V A min [l/s]	V A [l/s]	Zeta	L R+Z [mbar]	L R+Z [mbar]	R [mbar/s]	Z [mbar]	Remarks
1	25	125	5.00	40.0	40.6	-275	-339	38	81	4.3	91				0.2	80	154	11	20	MR12
1	26	130	5.00	40.0	35.0	-195	-281	38	88	4.2	96				0.2	88	154	13	24	MR12
1	27	130	5.00	33.0	25.3	-113	-182	38	53	8.3	95				0.1	47	168	8	8	MR12
1	28	90	5.00	24.0	17.3	-52	-115	38	51	3.2	100				0.2	83	121	33	12	MR12
1	29	90	5.00	19.0	10.2	6	-16	38	18	3.9	100				0.2	24	39	4	5	MR12
1	30	90	5.00	9.0	4.9	26	-29	26	4	5.9	100				0.2	8	34	3	1	MR12
1	31	75	0.50	0.30	0.2	4.9	-23	21	18	9	1.9	100			0.4	5	28	2	3	MR12
1	32	56	0.09	0.08	0.8	4.9	-45	9	31	2.3	100				0.7	23	23	12	12	---
1	33	75	0.50	0.30	0.0	5.3	-29	18	18	10	1.4	100			0.5	6	34	3	5	MR12
1	34	56	0.09	0.08	0.8	8.2	-56	9	37	2.7	100				0.7	28	38	14	17	---

- What does a human expert look at ?
- Can you switch the experts between 2 steps ?
- Full state vs partial state
- Designing Features



Designing the Reward Function

Snake game

Step Reward

100

Fruit	100
Death	-50
Success	1000
Step	-1

Roof drainage systems



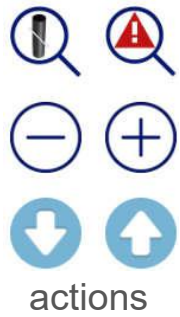
Change Error Count	+/- 1 per Error
Success	100
Step	-0.01

- How would you rate the result of an expert ?
- As simple as possible
- Positive feedback during the game
- Beware of “surprising policies”
- Game over if TotalReward too low

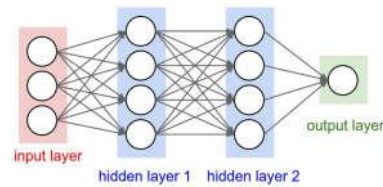
Turning the problem into a game



game engine



actions

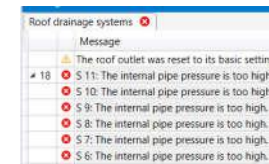


Policy

(rules learned, how to play the game)



RL algorithm



step reward

Typ	TS	d [mm]	L [m]	H [m]	V Set [l/s]	Q [l/s]	p in [bar]	p out [bar]	p at [bar]	p dyn [m]
1	23	125	3.00	48.0	40.4	-278	-220	38	81	
1	28	150	3.00	40.0	32.0	-199	-221	38	80	
1	27	150	3.00	32.0	22.2	-115	-162	38	75	
1	28	90	5.00	24.0	17.1	-52	-115	38	51	
1	29	90	5.00	18.0	10.2	0	-58	38	18	
1	30	90	5.00	9.0	4.9	28	-20	38	4	
1	31	75	8.50	0.0	0.0	-23	-21	38	3	
DWS	12	54	0.00	0.00	0.0	0.0	-45	0	15	
E	31429	75	0.50	0.50	0.0	0.0	-29	14	10	
DWS	34	54	0.00	0.00	0.0	0.0	-58	0	35	

game state

Finding the dimensions with reinforcement learning: demo

The screenshot shows the GEberit ProPlanner 2018 R2 software interface. The main window displays a 3D model of a pipe network with various pipe diameters and lengths. A sidebar on the left shows a 3D coordinate system and a reward system with 'Step Reward 100.99' and 'Total Reward 114.59'. A table at the bottom provides technical data for 'Strang A'.

Typ	TS	d [mm]	L [m]	H [m]	V Sof [l/s]	V [l/s]	p in [mbar]	p out [mbar]	v [m/s]	ψ [%]	Zeta	L-R-Z [mbar]
S	1	315	1,00		204,0	200,8	2	0	3,1	94	1,0	47
F	2	315	3,00	3,00	204,0	200,8	-255	2	3,1	94	0,3	21
S	3	315	50,00		204,0	200,8	-142	-255	3,1	94	0,3	113
S	4	315	6,36		102,0	100,1	-101	-107	1,5	97	0,3	7
S	5	315	3,50		102,0	100,1	-95	-101	1,5	97	0,3	8
S	6	315	12,00		102,0	100,1	-85	-95	1,5	97	0,3	10
S	7	315	13,00		81,6	79,7	-75	-81	1,2	97	0,0	5
S	8	230	15,00		61,2	59,3	-67	-70	1,4	97	0,1	11
S	9	230	15,00		40,8	38,9	-57	-62	0,9	99	0,0	4
S	10	160	15,00		20,4	20,4	-45	-61	1,2	99	0,4	15
S	11	110	0,71		20,4	20,4	-49	-70	2,6	99	0,5	21
E	12	110	0,25	0,25	20,4	20,4	-63	-49	2,6	99	0,3	11
E	13	110	0,20	0,20	20,4	20,4	-81	-63	2,6	99	0,0	1
TRAC	14	90	0,20	0,20	20,4	20,4	-121	-121	2,8	96	0,0	48

Hydraulics Calculation Pipeline



Traditional
Heuristics

Classic
Machine
Learning

Reinforcement
Learning

Automatic solution found in 93.4%

Finds a solution in 70.7%
of the remaining 6.6%

Automatic solution found in 98.1%

Summary

- Turning the problem into a game
- Continuous policy improvement
- No training dataset
- Complements supervised learning



Thank you!

Christian.Hidber@bSquare.ch

W +41 44 260 54 00

M +41 76 558 41 48

<https://www.linkedin.com/in/christian-hidber/>



About Geberit

The globally operating Geberit Group is a European leader in the field of sanitary products. Geberit operates with a strong local presence in most European countries, providing unique added value when it comes to sanitary technology and bathroom ceramics.

The production network encompasses 30 production facilities, of which 6 are located overseas. The Group is headquartered in Rapperswil-Jona, Switzerland. With around 12,000 employees in around 50 countries, Geberit generated net sales of CHF 2.9 billion in 2017. The Geberit shares are listed on the SIX Swiss Exchange and have been included in the SMI (Swiss Market Index) since 2012.



Resources

- Sutton & Barto: Reinforcement Learning, an introduction, 2nd edition, 2018:
https://drive.google.com/file/d/1opPSz5AZ_kVa1uWOdOiveNiBFiEOHjkG/view
- http://rll.berkeley.edu/deeprlcourse/f17docs/lecture_4_policy_gradient.pdf
- <http://karpathy.github.io/2016/05/31/rl/>
- <https://papers.nips.cc/paper/1713-policy-gradient-methods-for-reinforcement-learning-with-function-approximation.pdf>
- <https://arxiv.org/pdf/1707.06347.pdf>
- <https://openai.com/>

