SimSwarm: Simulating Multi-Agent Systems

Dave Raggett dsr@w3.org
W3C/ERCIM

Swarms are a valuable approach to modelling collections of IoT systems at an abstraction level above that of digital twins.

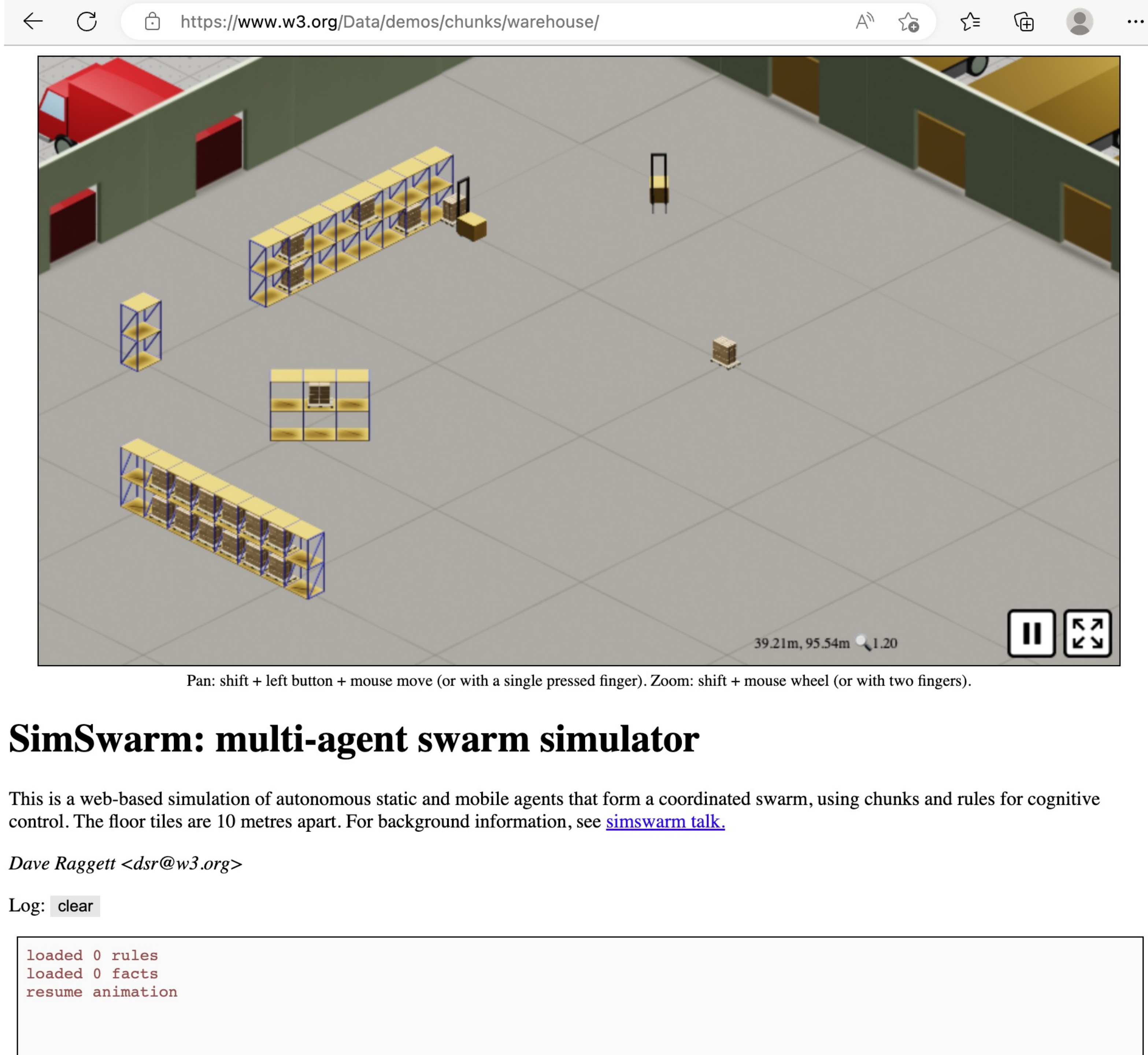
Simulations provide a low cost way to explore different algorithms in advance of real-world deployment.

Cognitive models for low-code development.

SimSwarm: web based isometric swarm simulator

- Multi-agent system
  - Autonomous mobile & static agents that communicate by exchanging messages
- Swarms can be smarter than their components
  - Emergent behaviour
  - Ant colony metaphor
- Smart agents using cognitive control
  - Flexible low-code framework using cognitive rules that operate on knowledge graphs
  - Local or cloud based
- Other agents using hard-coded behaviour
- User interface for high-level monitoring and control

See: https://www.w3.org/Data/demos/chunks/warehouse/



Pan: shift + left button + mouse move (or with a single pressed finger). Zoom: shift + mouse wheel (or with two fingers).

SimSwarm: multi-agent swarm simulator

This is a web-based simulation of autonomous static and mobile agents that form a coordinated swarm, using chunks and rules for cognitive control. The floor tiles are 10 metres apart. For background information, see simswarm talk.

Dave Raggett <dsr@w3.org>

Log: clear

```
loaded 0 rules
loaded 0 facts
resume animation
```

Distribution Warehouse as initial use case

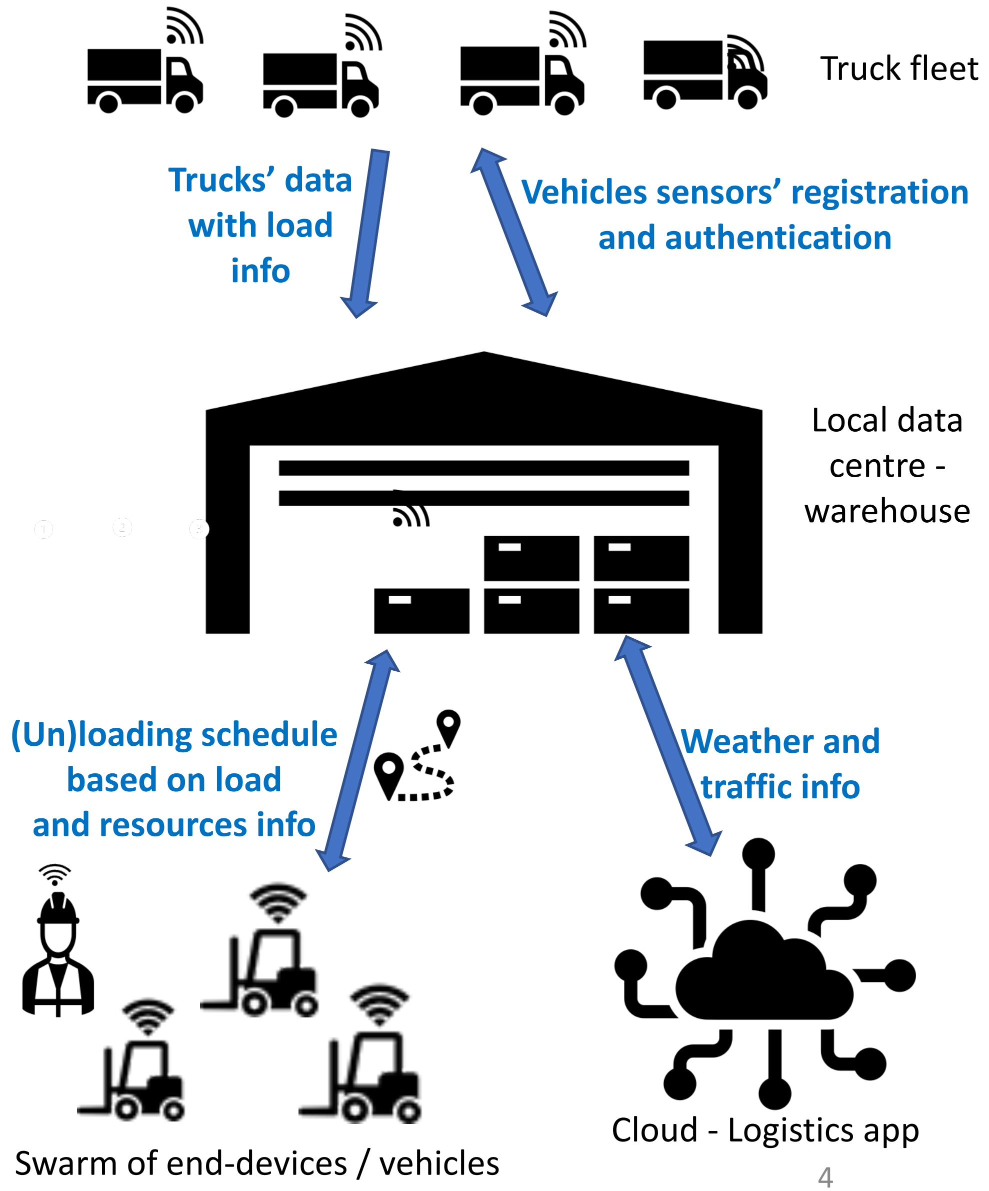
- Swarm of robot forklifts that shift pallets between incoming trucks, storage racks and outgoing trucks
- Based upon message passing between autonomous agents
- Forklifts execute simple plans, plus collision avoidance rules
- Cognitive control to fulfil overall goals for timely and efficient distribution of goods
- 2.5D graphics using isometric game projection
  - 3D models created using Blender + python scripts to export image tiles
  - Rendered on HTML5 <CANVAS>
- Chunks & Rules JS engine
  - W3C Cognitive AI CG, and inspired by John Anderson's ACT-R (@CMU)
- Route planning using A-star biased by pheromone vectors

Use Case: Smart Logistics (INCODE)

Concept:
Load information including transport conditions data (perishables like food) flow from mobile IoT sensors of **the truck fleet** in real time (or also periodically (e.g. daily)) to register and authenticate with the local data centre at the **warehouse**. Static IoT sensors in the warehouse provide inventory status info. SOME of the local data is combined with the cloud intelligence (logistics app). Then the local data centre receives commands to manage a **swarm of end-devices / vehicles** for (un)loading of the trucks according to a schedule (or in real time).

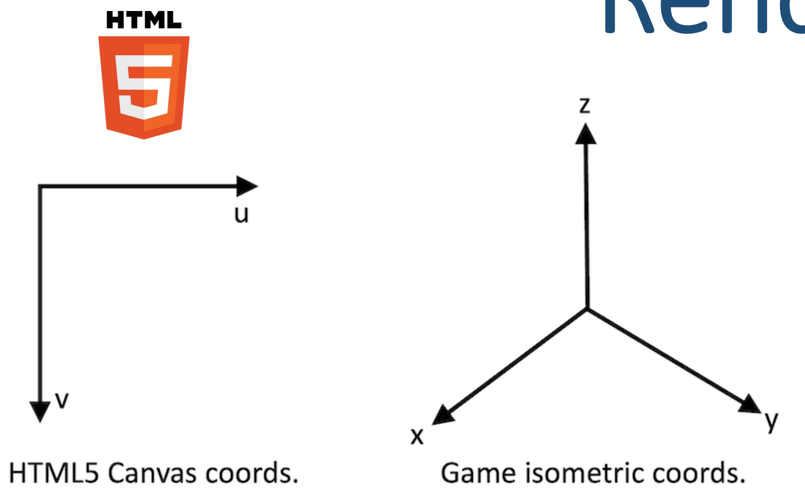
Benefits:
Optimised (un)loading scheduling and allocation of respective resources leads to prevention of damage to perishables and to reduced units costs of using trucks and loading vehicles. In turn, this leads to increased profits of the logistics actors.

Logistics app is a trusted framework for all the logistics actors: road carriers, customs, warehouses, distributors or retailers. Coordinated (un)loading provides for work safety and prevents product mishandling.


Truck fleet
Trucks' data with load info
Vehicles sensors' registration and authentication
Local data centre - warehouse
(Un)loading schedule based on load and resources info
Weather and traffic info
Swarm of end-devices / vehicles
Cloud - Logistics app

HORIZON-CL4-2022-DATA-01-03: Programming tools for decentralised intelligence and swarms (RIA);
Example Project: INCODE, slide courtesy of Jan KOMAREK, CNECT.E.4 Internet of Things

Rendering Algorithms

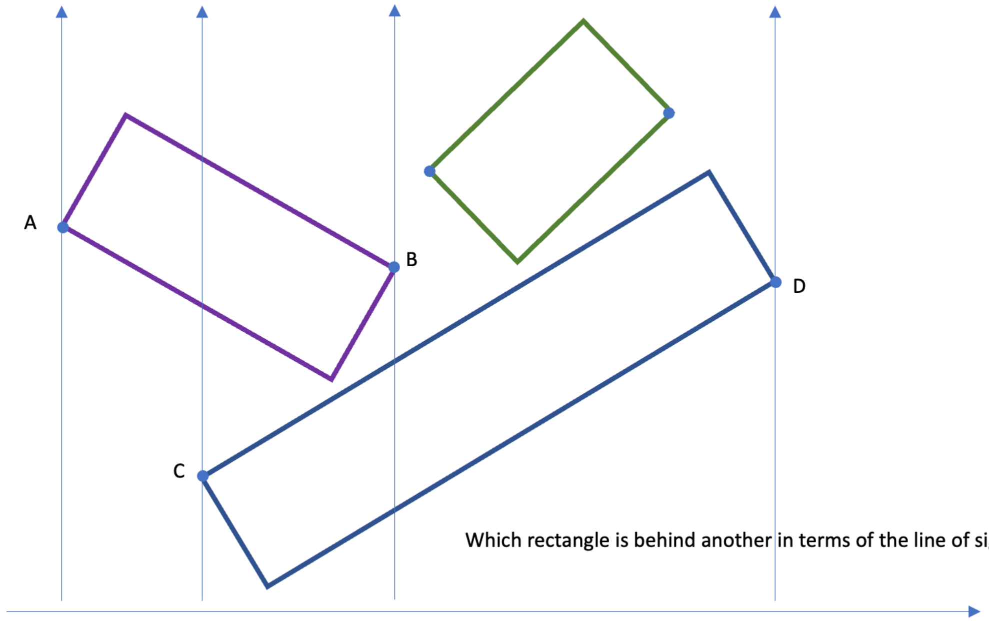


$u = 0.5 \cdot (y - x)$ $\quad x = v - u + z$

$v = 0.5 \cdot (x + y) - z$ $\quad y = u + v + z$

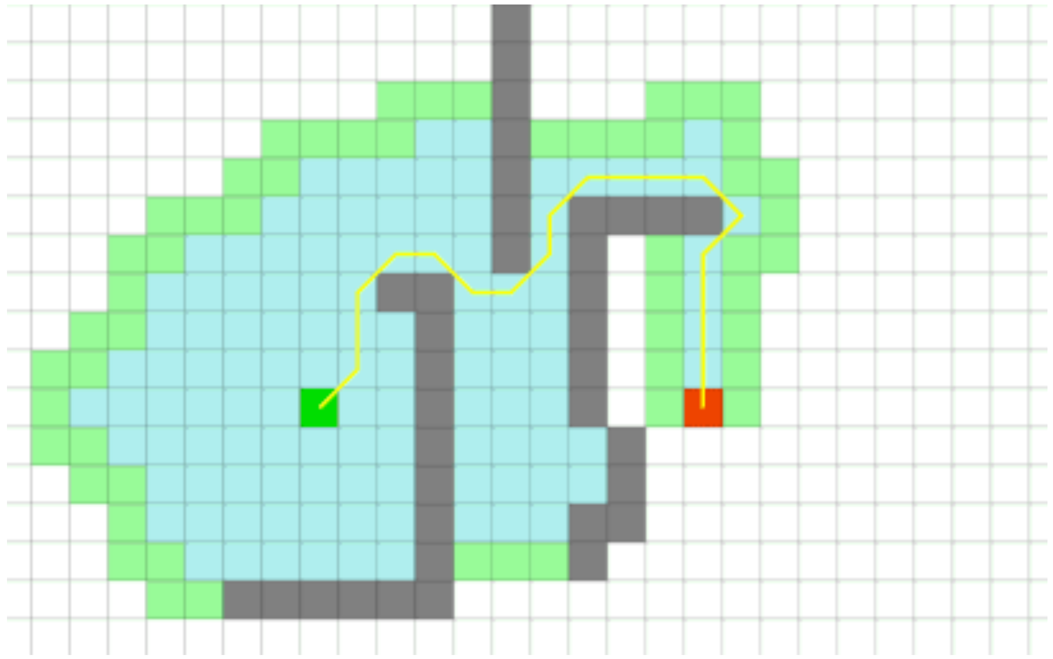
- Orthographic projection equivalent to view from infinity – so that perspective lines are parallel rather than converging
- Commonly used for strategy games, e.g. SimCity, and for Chinese scroll paintings in contrast to European perspective paintings
- Uniform scale simplifies rendering by using precomputed fixed size image tiles

- Rendering sequence determined using spatial sorting based upon component's floor area expressed as a rectangle
  - Map corners to canvas coordinates and sort by u value, where line of sight is along v axis
  - Purple rectangle is behind blue rectangle since line of sight from corner C intersects line from A to B
  - Green rectangle is behind blue rectangle since its corners are between C and D

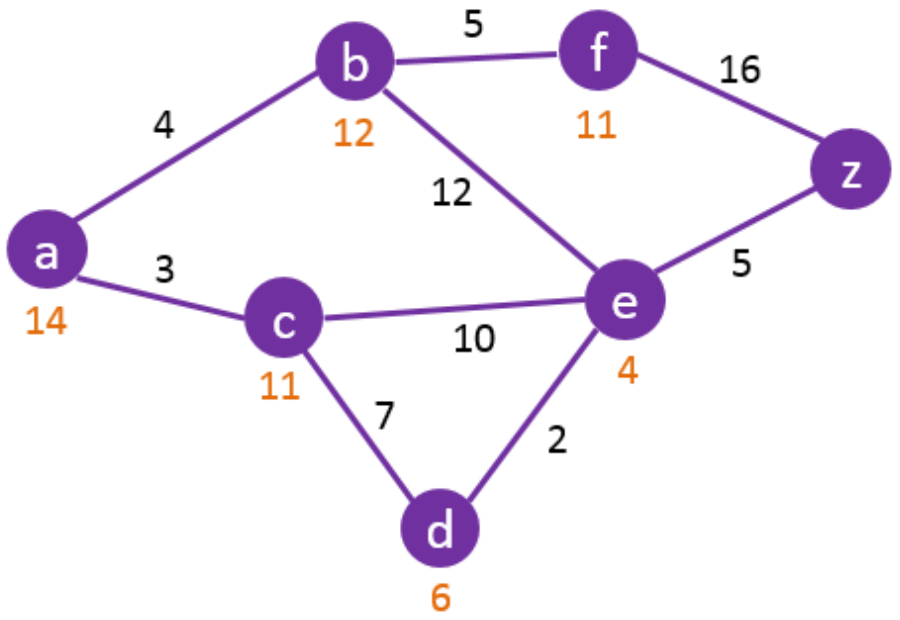


Which rectangle is behind another in terms of the line of sight?

Route Planning



Flood fill algorithm: annotates each cell with minimum distance from the start node.
Figure source: Wikipedia (**license**)



A-star algorithm: beam-search from start node to destination node, where orange numbers are distance as a crow flies from node Z and black numbers are direct distances between nodes.

Route Planning and Traffic Management

- Ant colonies – ants deposit pheromone trails to establish shortest routes between nest and food sources
    - Pheromones slowly evaporate, akin to forgetting curve in human memory
- Adapt A-star to work with matrix map, and associate each matrix cell with pheromone vector indicating the preferred direction of travel
    - Initialise vectors to bias traffic routes
- Traffic management
    - Ad hoc routing – no restrictions
    - Restricted routing – designated paths
- Optimal routing emerging from collective behaviour
    - Emergence of one-way lanes and traffic junctions
    - Reducing distance travelled and delays due to collision avoidance, akin to human streams in busy crowds

Indexing and Neighbours

- Use spatial indexing to limit rendering to the visible part of a very large scene
- Spatial indexing also helps to determine other nearby mobile agents as part of collision avoidance
- Quad-tree indexing using recursive subdivision into rectangles or triangles

- Potential for collisions*
  - Each mobile agent knows its location, speed and direction
  - Shared with nearby agents to detect potential collisions
- To avoid collision, agents either turn away or give way until the path is clear
- Local course recovery after collision avoidance

* Inspired by FLARM systems for gliders and light aircraft, that transmit position and speed vectors for use by intelligent motion prediction algorithms, and warn the pilots as needed.

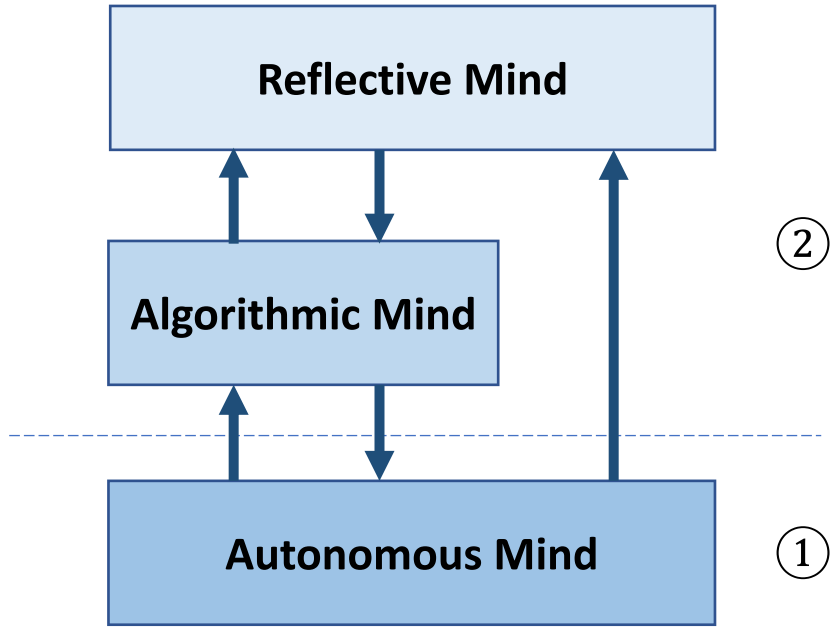
Cognitive Control
using knowledge graphs

- Incoming shipments have low diversity of goods per truck
- Outgoing shipments have high diversity to suit customer needs
- Some goods must be shipped within tight time limits, e.g. fresh vegetables
- Balancing cost of inventory against resilience to disruptions
  - Just-in-time storage and delivery is sensitive to disruptions in supply
- Pallets are assigned to storage areas and outgoing shipments
- Idle forklifts are dynamically assigned to collect specific pallets
  - From incoming shipments to storage
  - From storage to outgoing shipments
- Robot forklifts need downtime to recharge their batteries

Cognitive Agents

- Class of agents that maintain knowledge graphs and apply rules to decide on actions
- Perception is process of building higher level models of the agent's environment
- Cognition is process of reasoning to decide on what actions to take – using a mix of type 1 and type 2 processing
  - symbolic systems using handcrafted knowledge graphs and rules
  - non-symbolic systems using artificial neural networks trained on very large datasets, plus transfer learning on smaller datasets and reinforcement learning with human feedback

Keith Stanovich's Tripartite Model of Mind

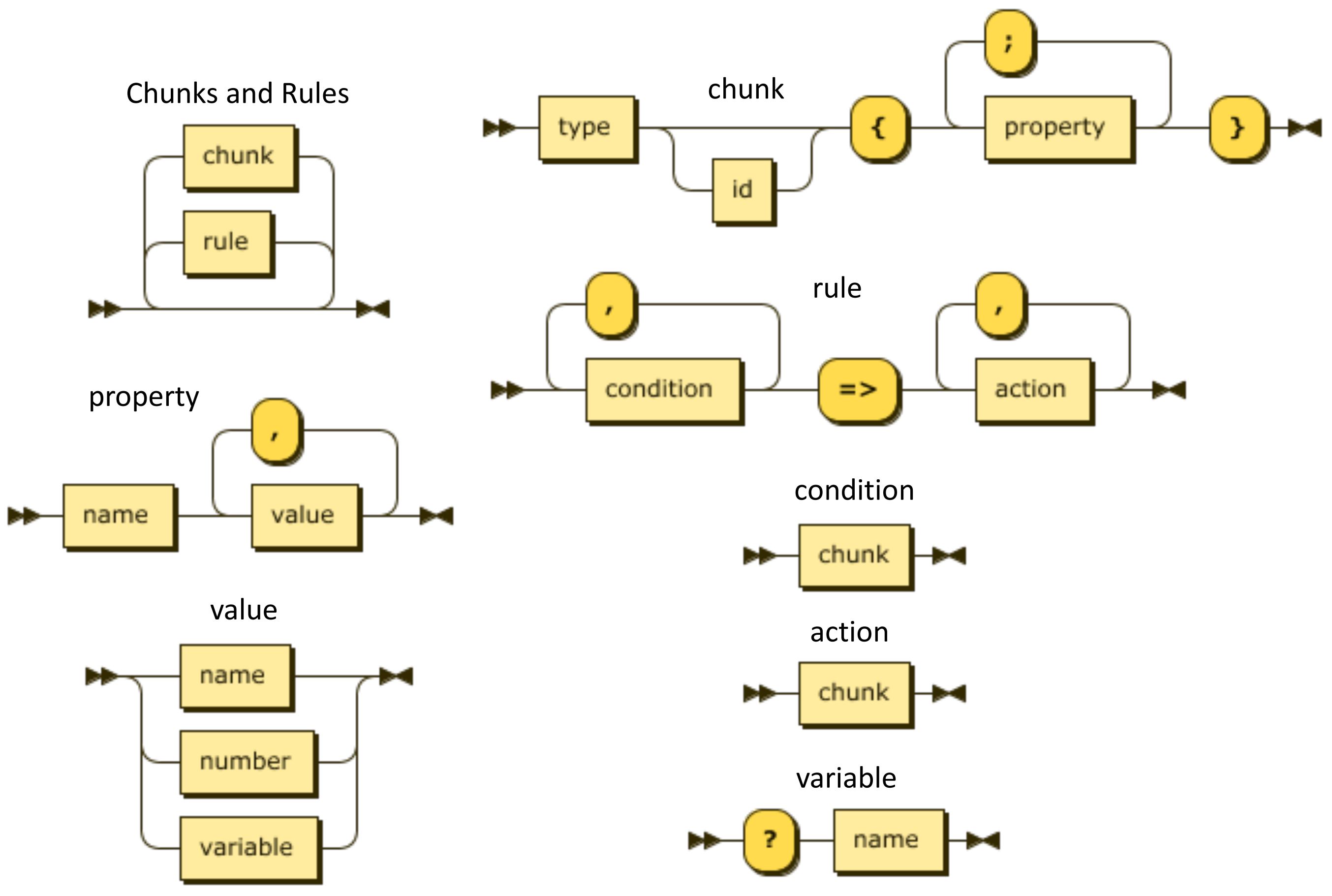


Type 1 processing is fast, automatic, and opaque, e.g. recognising a cat in a photograph or a traffic sign when driving a car.
Type 2 processing is slow, deliberative, and open to introspection, e.g. mental arithmetic. It is formed by chaining Type 1 processes using working memory.

See, e.g. "Dual-Process Theories of Higher Cognition: Advancing the Debate", Evans and Stanovich (2013), along with "Thinking Fast and Slow", Daniel Kahneman (2011)

Chunks and Rules

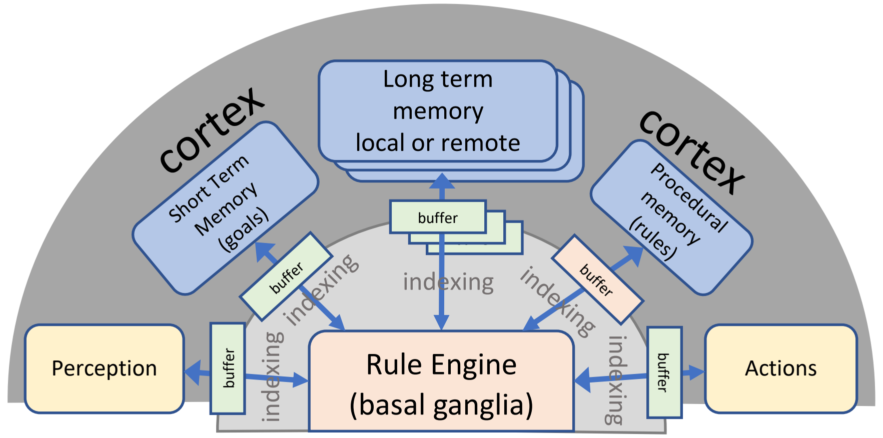
web-based demos for smart homes and factories



names beginning with “@” are reserved, e.g. @do for actions

See W3C Cognitive AI Community Group

Cognition – Sequential Rule Engine



cortex
Short Term Memory (goals)
Long term memory local or remote
cortex
Procedural memory (rules)
buffer
buffer
buffer
indexing
indexing
indexing
indexing
indexing
Perception
buffer
Rule Engine (basal ganglia)
buffer
Actions

Cognitive Buffers hold single chunks
Analogy with HTTP client-server model

- Inspired by John Anderson’s ACT-R and decades of cognitive science research at CMU and elsewhere
- Mimics characteristics of human cognition and memory, including spreading activation and the forgetting curve
- Rule conditions and actions specify which cognitive module buffer they apply to
- Variables are scoped to the rule they appear in
- Actions either directly update the buffer or invoke operations on the buffer’s cortical module, which asynchronously updates the buffer
- Predefined suite of cortical operations
- Reasoning decoupled from real-time control over external actions, e.g. a robot arm

Example Swarm Messages

- Messages signal events, invoke actions and signal their asynchronous results
- Cognitive rules work with chunks, which have an easier syntax than JSON
- Chunks map to RDF using context files similar to JSON-LD
- Swarm agent models describe which messages an agent supports akin to thing descriptions in the web of things
- Messages can be sent one-to-one or one-to-many

Here are some messages* that can be sent to the robot forklifts:

pallets can be on the floor or on a rack or in a truck

Using JSON:	Using chunks:
{"action":"grab", "side":"front", "pallet":"pallet1"}	grab {side front; pallet pallet1}
{"action": "release", "x":20, "y":30, "orientation":45}	release {x 20; y 30; orientation 45}
{"action":"move", "x":20, "y":30, "gear":"forward"}	move {x 20; y 30; gear forward}
{"action":"move", "x":30, "y":20, "gear":"reverse"}	move {x 30; y 20; gear reverse}

* The actual data formats will vary with the protocols used to communicate with IoT devices, and the above is from the perspective of the cognitive rules used to coordinate the swarm.

Swarm Management

Centralised Control

- Swarm coordinator allocates tasks, tracks where everything is, and computes routes for mobile agents
- Messages exchanged between swarm participants and the coordinator (one to one)
- Single point of failure
- But simpler in respect to monitoring and control for business objectives

Decentralised Control

- Tasks allocated in a distributed way 1) request for service, 2) offers of service, 3) requester selects who will do this task
  - Offers can specify a future time, e.g. when a forklift expects to finish its current job
- Topic based message distribution (one to many) – service providers listen on topics relevant to what services they provide
  - Can be limited to nearby participants
- Geospatial streams in place of centralised maps, along with agents that offer routing services

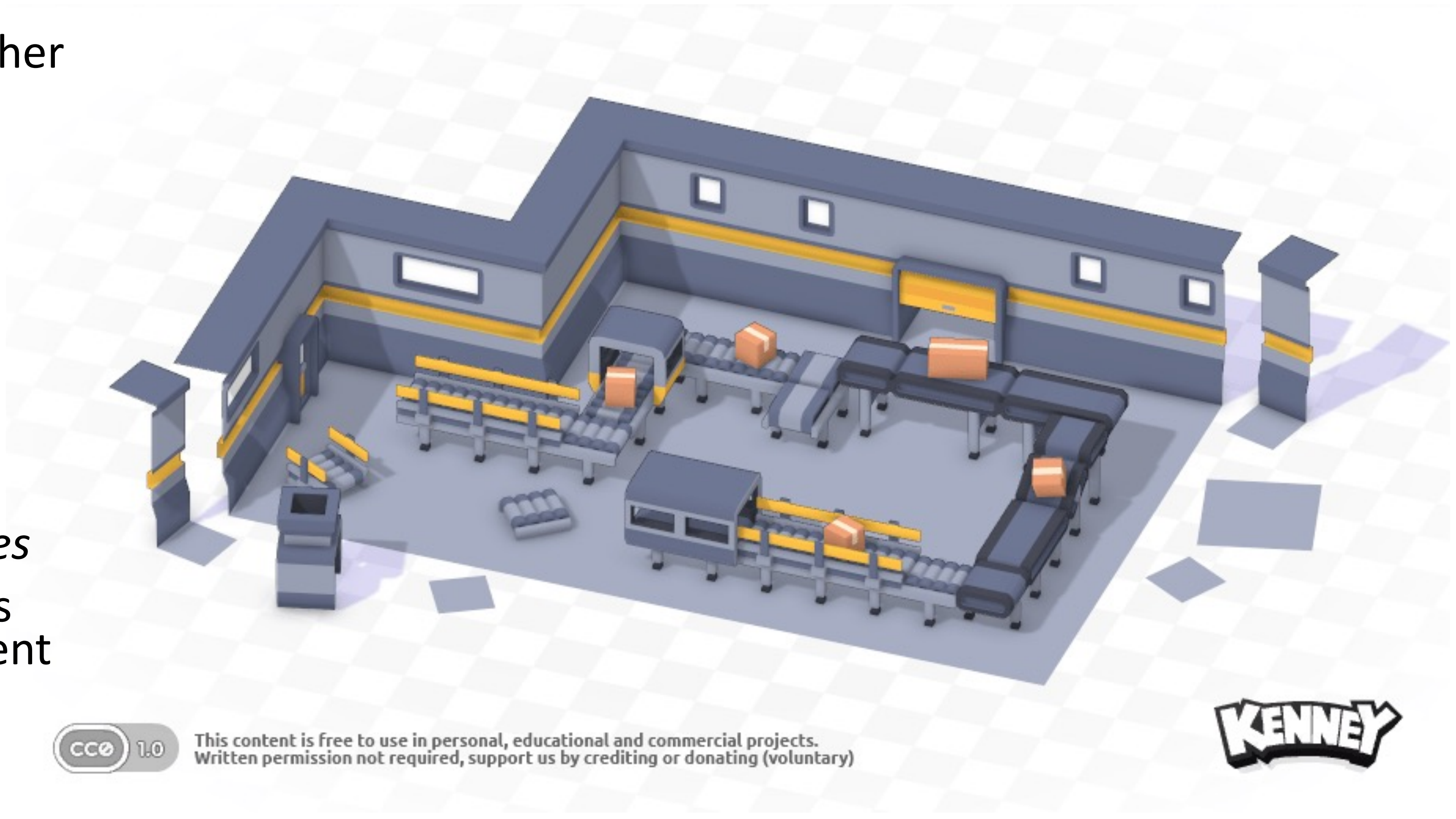
Perception as a distributed process of semantic fusion mimicking the human brain

Questions?

n.b. I hope to work on smart factories as a further use case, and there are many other potential use cases, e.g. urban traffic management, and logistics at container ports.

Factory Simulation

- Smart factories are another potential application
- Robot arms
- Milling machines
- Painting
- Conveyor belts
- Movable racks
- AMRs
- *and many other machines*
- People – as real factories use people to complement automation


cc0 1.0
This content is free to use in personal, educational and commercial projects.
Written permission not required, support us by crediting or donating (voluntary)
KENNEY

See: https://kenney.nl/assets/conveyor-kit

Urban Traffic Management

W3C/ERCIM is interested in developing a web-based demo to complement UC1 and UC2

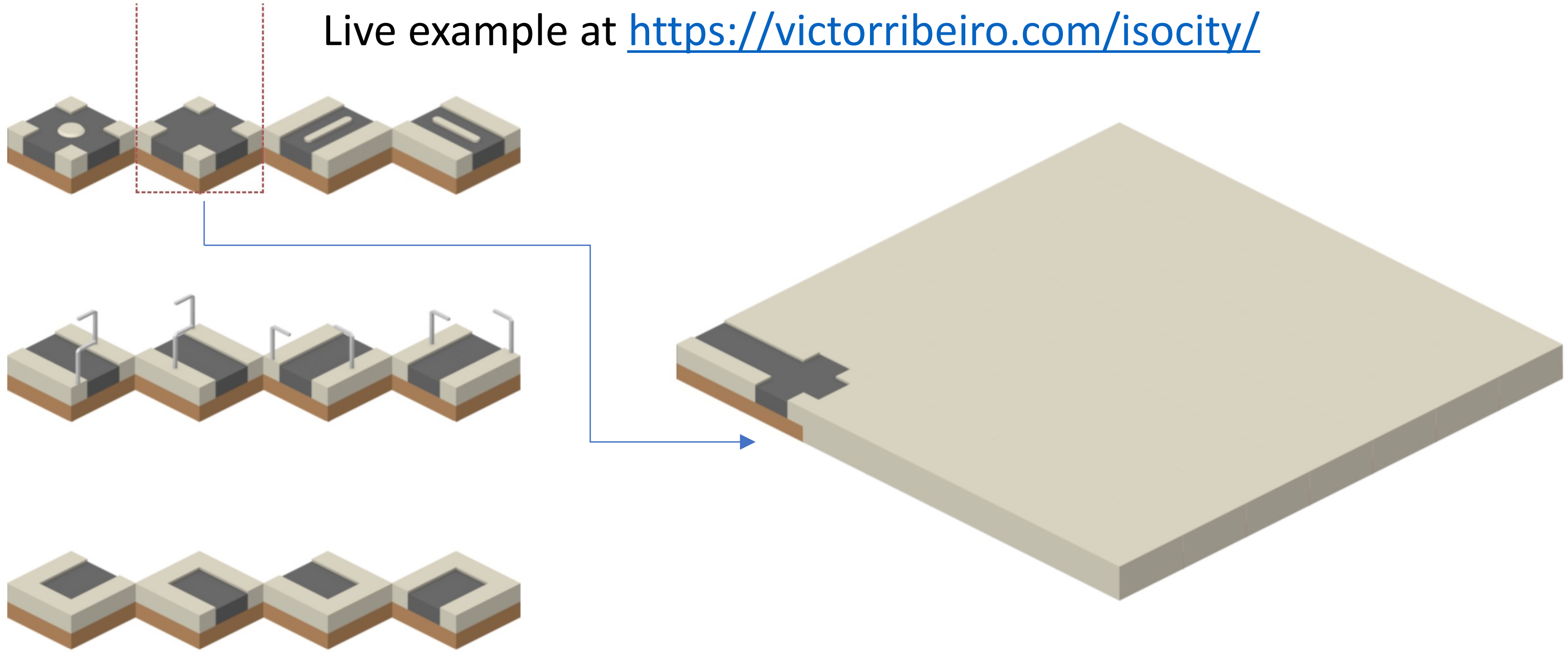
- Isometric projection enabling city scale simulations
- Composed from collections of image tiles
- Generated from good enough 3D models
- Iconic rather than photo realistic
- W3C/ERCIM could fund the creation of the graphics

See: https://github.com/victorqribeiro/isocity

Urban Traffic Management

Live example at https://victorribeiro.com/isocity/

- We could even enable users to build their own cities, e.g. in a web browser
- Can run in a browser
- Pan, zoom and rotate* the field of view



* Through fixed set of angles, e.g. 45° intervals using tile sets, such as below

Urban Traffic Management

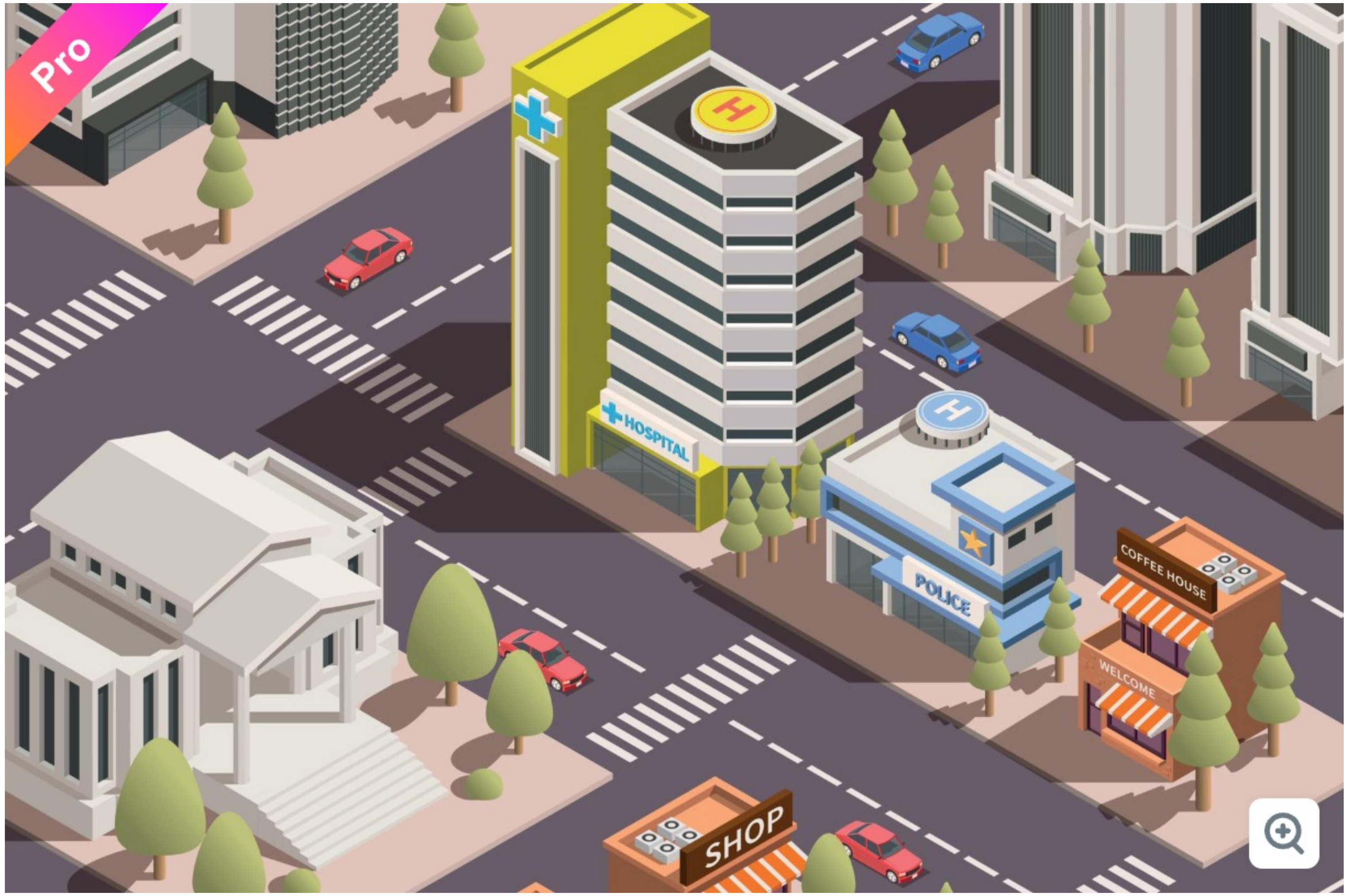
- Here is a larger static example from *shutterstock*
- Many different components that can be configured as needed
- Each component is associated with its behaviour

See: https://www.shutterstock.com/image-vector/isometric-urban-megalopolis-top-view-city-582946510

Urban Traffic Management

- Close up of just one type of road junction
- Challenges for large scale simulations*
- Could include cars, trucks, cyclists, and pedestrians, e.g. mothers with prams, old people with walking sticks, and young children

* Lower fidelity simulation outside field of view



See: https://www.vecteezy.com/vector-art/4564141-modern-city-isometric-illustration

Container Port Logistics

- International trade depends on efficient handing of containers
  - Transported on ships, trucks and trains
  - Minimising delays
  - Maximising throughput
- Warehouses for redistribution when containers carry loads for multiple customers
- Requirements for customs inspections and tariffs

See: https://www.shutterstock.com/image-vector/beautiful-isometric-vector-set-port-terminal-1863706453