# Ardán: Using 3D Game Engines in Cyber-Physical Simulations (Tool Paper)

Fergus Leahy, Naranker Dulay

Department of Computing,
Imperial College London, UK
f.leahy14@imperial.ac.uk, n.dulay@imperial.ac.uk

**Abstract.** In this paper we present Ardán, a novel simulation platform using a 3D game engine to stand-in for the real world, providing realistic physics and realistic crowds that can interact in real-time with a cyber-physical simulation. Ardán features 3D device (sensor and actuator) placement, flexible time-control, phenomena-on-demand as well visualisation, virtual devices and native application code. The flexibility, control and scalability of Ardán is demonstrated with a corridor case-study that supports upto 200 nodes running at real-time or faster.

## 1 Introduction

Despite the growth in interest in the fields of sensor networks, cyber-physical computing and the Internet of Things, developing and testing distributed networks of devices remains a difficult task. Testing and understanding how the environment interacts with a cyber-physical system (CPS) and vice-versa relies upon deploying devices in the target environment and waiting for or creating the desired phenomena to interact with the CPS. Phenomena can include events such as movement of devices or objects in the environment, passive or active interaction with people (pressing buttons, triggering motion sensors), or other sensor events. Thus, performing test deployments can be time-consuming, difficult and expensive to run repeatedly.

To address this issue various tools and techniques have been developed, such as test-beds [6, 5, 8] and simulators [12, 10, 9, 13, 15]. However, existing tools and techniques aren't adequate for reliably and comprehensively testing these devices in the context of their target environment and the phenomena which may occur within it. Current approaches for testing sensor networks and CPS have focused heavily on accurately simulating devices, the network and power consumption, with great success [12, 10]. However, support for interacting with the environment is limited, typically performed using recorded or designed sensor trace data. This approach can be inaccurate, unrealistic and is restricted to what was recorded.

In this paper we present a novel approach to this problem, integrating a freely available high-performance 3D video game engine (Unreal Engine 4) with an existing sensor network simulation platform (Cooja), creating an end-to-end simulation solution for realistic testing of sensor networks in their target

environments. By integrating a 3D game engine, we are introducing sensor network simulations into virtual reality with a real-time and dynamic virtual world, utilising the game engine's realistic physics and realistic crowds to influence and test the deployed sensor network.

Our contributions consist of:

- A novel 3D simulator for testing cyber-physical systems in a virtual world using realistic physics and crowds to simulate interactions with the environment and people.
- An example case study, the corridor, demonstrating some of the capabilities of Ardán and providing a benchmark for future work.

## 2 Ardán

In this section we give an overview of the main features of Ardán[1] followed by an overview of its design.

### 2.1 Features

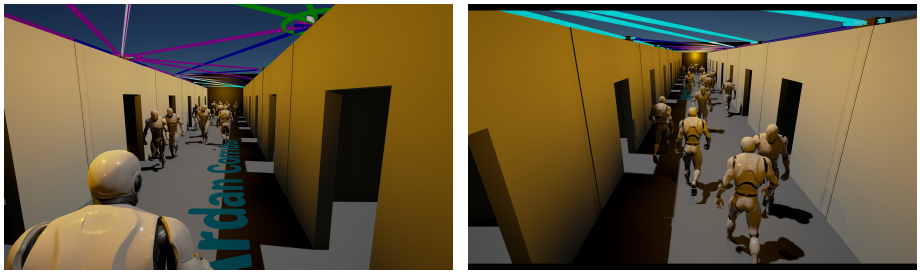
**3D design and placement**. A key part of many CPS projects is understanding how many and where to place devices within the environment to achieve some desired objective, such as device failure tolerance or sensing accuracy. With Ardán developers are able to easily and quickly augment 3D environments with devices, scale up or down the size and topology of the device network as well as move devices around 3D environments to test different configurations.

**Time Control**. Unlike in real-world deployments, Ardán developers are able to control time in the simulated world. Developers can: stop-the-clock, freezing both the simulation and world in time, whilst giving them full control over what they see, allowing more time to observe the environment and move between points of interest; slow down time, giving developers more time to observe or control the simulation; or even speed up time, providing desired results in considerably less time.

**Phenomena-on-demand**. In order to better test and understand sensor network applications in the real-world, developers often need to wait for or even force desired phenomena to occur and then observe how their system reacts. However, exercising control over the real-world can be a difficult and time-consuming challenge, and sometimes not possible (e.g., fire), due to health and safety concerns.

Using Ardán, developers can take direct control of a virtual person or script realistic virtual crowds to carry out tasks, such as walking between points, avoidance, following or interacting with objects. Figures 1 and 2 show people walking up and down a corridor, avoiding each other's path. Unlike using trace data, genuine or created, developers can easily tweak scenarios, such as moving devices, people or adjusting behaviour, to test subtle or significant variations.

[1] Ardán, pronounced "awrd-awn", is the Gaelic word for platform.



(a) Over-the-shoulder view (b) Camera view

**Fig. 1.** 15 people walking up and down the virtual corridor, triggering motion sensors

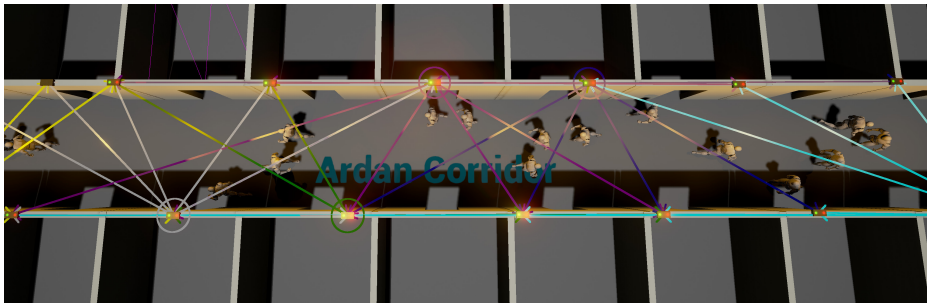



**Fig. 2.** Birds-eye view

**Pattern-matching Eventbus** Logging events in a CPS is vital for post-experiment analysis. Our design utilises the Homework Cache [14], an event publish-subscribe engine as the core communication mechanism, into which events for both the CPS and virtual world are injected. The Homework Cache provides the ability to perform real-time complex event pattern-matching over the event stream to detect phenomena of interest, such as misbehaving nodes or network partitions.

**Visualisation**. Ardán provides tools to overlay visualisations of network and device meta-information on top of the virtual world to help understand how the network is running, allowing developers to see information such as how network paths form as packets are sent, as well as transmissions, receptions, interruptions. In figure 2, sending devices are highlighted with a circle and receiving devices are connected by an arrow to the sender, each device is represented with its own colour, to help differentiate simultaneous transmissions.

**Virtual Sensors and Actuators**. Within Ardán we have modeled several basic sensors and actuators, including motion detectors, buttons, lights and location. These act as virtual hardware for the simulated sensors, allowing the simulation to interact with the virtual world. Virtual sensors can be designed to model a real sensor's behaviour, or be virtually improved to provide higher

accuracy or more features, not possible (yet) with existing hardware, opening the door to experimental virtual hardware prototyping.

### 2.2 Design

Ardán itself is not a single component, but a collection of plugins for facilitating the communication and arbitration between different tools, shown in figure 3. The individual tools act as peers, sharing information with each other to inform their individual simulations/operations, e.g., location updates sent from the 3D game engine to the simulator, affecting radio transmission.

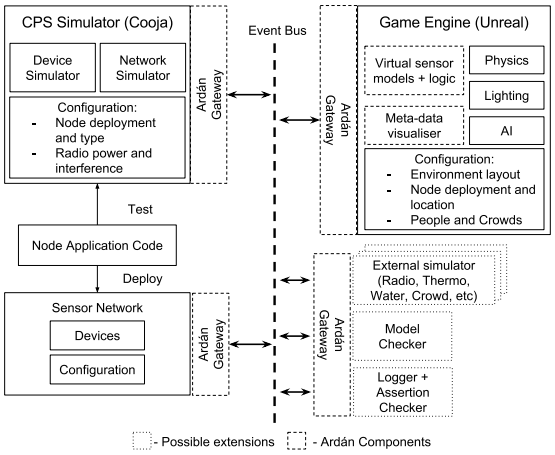



**Fig. 3.** Ardán architecture

While the goal of work is to build an improved simulation experience for testing sensor network applications by integrating a 3D game engine into the pipeline, we also want to create an architecture that can support future integration of tools to improve the field of sensor network application testing.

Thus, rather than integrate the systems directly, our approach uses an event bus and common schema to describe information passed between the different subsystems. This approach reduces the tight coupling between components, allowing individual components to be swapped out or new ones added. We envision the use of tools such as model checking, statistical analysis, unit-testing and advanced simulation for radio and environmental properties.

The wireless sensor network (WSN) simulator, Cooja [12], performs all application and network simulation for the co-simulation and is configured with a set of nodes with their 3D location and native application code written in Contiki to run. As the application code is run in real-time, any sensor- or actuator-based hardware requests, such as sensor reads and actuator commands, are forwarded to the 3D game engine to be performed in the virtual environment.

Within the Unreal Engine we modeled a 3D environment and created a several components to support deploying virtual sensor networks, including device and sensor 3D models, hardware abstractions for the virtual sensors and actuators to sense the virtual world and report back to the simulator over the network bus. Similarly within the Cooja simulator we have modified the simulated hardware to communicate to our virtual hardware in the Unreal Engine.

Key features of games engines, aside from the 3D modelling and graphics tools, are their support for advanced physics and lighting simulations. Typically performed by middleware, physics engines, such as PhysX[3] and Havok[1], provide real-time realistic physics including collision detection, rigid- and soft-body physics, forces and motion, fluid and particle simulation, and destruction. Using these tools, game worlds and the objects within react to the player as we would expect, such as objects falling due to gravity or rebounding after a collision. Similarly, games engines also provide advanced lighting, enabling the use of both static and mobile lighting, with dynamic shadows, occlusion, reflection and refraction. Lighting is key to bringing virtual scenes to life, by illuminating spaces, guiding viewers attention and creating natural divisions between areas.

## 3 The Corridor: A case study

To demonstrate what development of a non-trivial CPS application using Ardán, we devised the following case study, based on an office corridor; we show how it helps developers to test and visualise different scenarios using its novel features, and demonstrate its scalability with up to 200 devices.

The case study focuses on controlling the lighting within a modern day office corridor, with the goal of striking a balance between energy efficiency, effective lighting and user comfort. The ideal corridor lighting scheme should provide a pleasant lighting scheme for users of the corridor, able to adjust based on ambient light levels, gradually illuminating as they progress through it, whilst also ensuring energy is minimised by turning off or reducing the brightness of unused or infrequently used parts of the corridor.

Thus, this provides an interesting and non-trivial task, due to the many ways in which the corridor can be entered/exited or moved around within in; people can enter from the beginning, end or from a room; people can move down the length of the corridor or directly from room to room; people often also stop in the corridor, spending time talking or waiting. Similarly, understanding the ideal number and placement for devices and sensors, and how it affects applications, such as power, reliability, robustness etc. Hence, providing effective lighting schemes can prove difficult to analyse and reason without rigorous testing.

The rest of this section will demonstrate and discuss the use of Ardán to design, analyse and test for our target environment, the corridor, highlighting the features and benefits that the tool provides.

### 3.1 Corridor Setup

We created a virtual corridor within Unreal based on a real corridor within our building, measuring 20x 1.5 meters, with 5 doors spaced evenly on either side. Along the corridor we placed 15 nodes with lights and conical motion sensors attached to the ceiling facing the floor directly below, shown in figure 4.

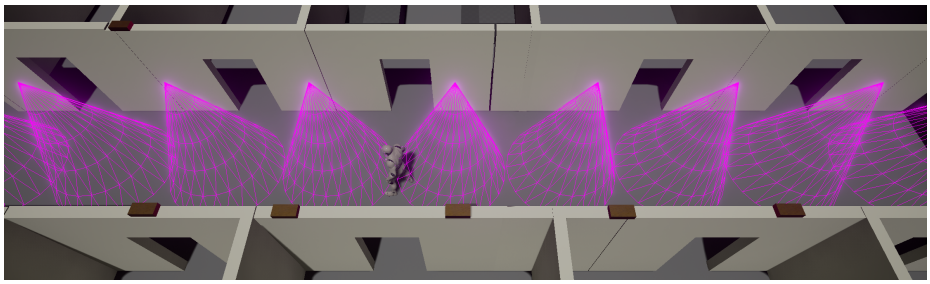


**Fig. 4.** Corridor layout with nodes and conical motion detection zones

To construct the corridor, we used pre-existing models for walls, doorways and lights, making it quick to build the required environment. In addition to these, we created several new 3D models for nodes and a variety of sensors, which can be composed together to create different sensing devices. A drag-and-drop interface is used to place nodes within the newly built 3D environment. The node 3D model is based on a small box with 3 coloured lights, representing the typical LED outputs available on devices, such as the TelosB mote. Each node has a directional light and conical motion sensor attached, giving it the ability to detect when a person moves into range. Nodes also contain parameterised logic to model their functional behaviour within the virtual world, enabling developers to tweak a node's sensing and actuating capabilities to match desired attributes, such as detection accuracy or responsiveness.

The next step was to create the nodes in the Cooja simulator, compiling and loading node application code. Using the IDs Cooja assigns to these nodes, the virtual representations were assigned matching IDs. This is especially important when certain applications are loaded on particular nodes, or when node IDs are used programmatically e.g., for location, routing or ordering.

### 3.2 Lighting Algorithm

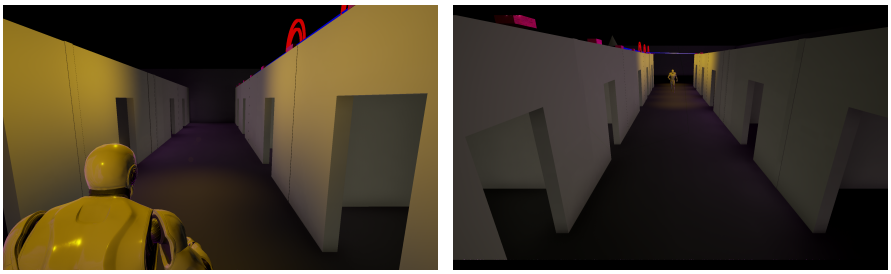
For our case study we developed a basic lighting algorithm which waits for a motion detection event before illuminating its light for 5 seconds and notifying its closest neighbours. If it receives a message from a neighbour, it checks that it's adjacent, before illuminating its light for 3 seconds. Using the tool we iteratively developed the algorithm, with the goal of implementing directional path illumination and loitering detection.

### 3.3 Testing with "What if?" scenarios

When testing CPS deployments, "what if" questions about how the system will perform will naturally arise, in this case we can ask: "what if we move or increase/decrease the number of nodes?", "what if there are crowds of people?", or "what if we place sensors differently or use more/less sensitive ones?". Being able to quickly test and understand what happens to a system in these different scenarios is key to improving its reliability and efficiency.

In order to test our lighting application we devised several test scenarios based on these questions to test both basic and complex situations for which we expect the system to perform correctly with; the complexity of a scenario increases as the number of agents in the scene increases and the pattern of movement changes from simple start to end directions, thus becoming more difficult to visualise and debug conceptually.

We tested two node configurations, with 5 and 15 nodes placed along the corridor. Using Ardán we experimented with node placement, attempting to find ideal placement strategies to improve the responsiveness and efficiency of the lighting. We found that placing nodes outside of office doors key to ensuring adequate lighting for when people enter and leave the corridor.



(a) Over-the-shoulder view (b) Corridor view

**Fig. 5.** Viewing the corridor lighting algorithm from different viewpoints.

We also found by using a greater number of nodes and placing them so that motion detection zones crossed, allowed for the design of a more efficient and pleasant lighting experience. Rather than turning on and off large zones of multiple lights, with fine grained detection we're able to gradually turn on and off light zones, resulting in less lights on at full brightness and a smoother transition from darkness to light. Using Ardán we can also experience this first hand by observing the corridor from the viewpoiont of virtual people as they traverse the space, shown in figure 5a, or view it from a CCTV perspective, shown in figure 5b.

### 3.4 Crowd Control

When simulating a scenario with a single person, we're able to take control and guide them along the corridor using the mouse cursor. However, simulating multiple people it becomes more challenging, choosing their destinations and ensuring they avoid one another. Using the game engine, we're able to take advantage of its path-finding and collision avoidance tools, to better control and direct crowds of people in a simulation. In this case, within our corridor we placed several target locations which the simulated people systematically attempt to walk to, navigating around obstacles and avoiding one another.

### 3.5 Performance

To prove useful Ardán needs to be able to scale to support large CPS consisting of tens if not hundreds of devices whilst ensuring synchronised behaviour at real-time or faster. To demonstrate the scalability of Ardán, we used the case study described in section 3, and scaled the number of nodes and running speed of the simulation. Within the corridor we placed the sensor nodes and motion sensors and implemented the algorithm discussed in section 3.2.

The tests were run on the following spec machine: Xeon E5 1650 6Core with HT, 16GB RAM, 256GB SSD and a sufficiently powerful (MSI GeForce GTX 970) graphics card to support the game engine, otherwise, slow response times between the simulator and 3D game engine would cause the simulation to stall and drop below real-time performance.

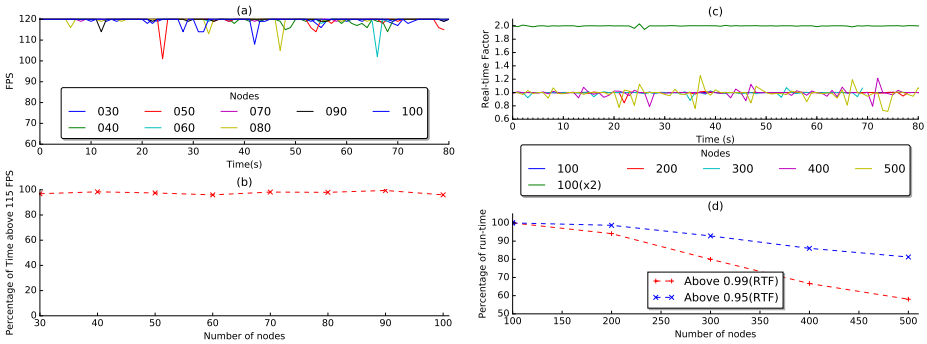



**Fig. 6.** Plot (a) and (c) show game engine FPS and simulator speed with increasingly large network sizes. Plot (b) shows the percentage of time the game engine stays above 115 FPS. Plot (d) shows the percentage of the total run time which the simulation maintains above 99% and 95% of its target speed.

To understand how well Ardán performs, we measured the performance of the individual components, the Cooja simulator and Unreal Engine, measuring

the real-time performance and the frames per second (FPS), for the respective tools. The results in figure 6(c)&(d) illustrate the percentage of time Cooja is able to maintain real-time speed; significant drops, below 95%, indicate the simulation is lagging behind real-time, which could cause temporal bugs or artifacts which would not exist in a real deployment. Similarly, for the Unreal Engine, performance is measured in FPS. Regardless of the FPS the engine always maintains real-time peformance; the FPS represents the overall responsiveness of the engine in terms of physics, input, output and networking, thus, a higher FPS provides more responsiveness when communicating with the Cooja simulator.

The results show that Ardán can support up to 200 sensor nodes running at real-time with the simulator staying reliably synchronised with real-time and thus the game engine. Beyond this, the simulator performance degrades quickly and struggles to keep up with real-time. The results in figure 6(a)&(b) also show that the game engine maintains a consistent FPS above 115, resulting in a smooth simulation and minimal overhead on the Cooja simulator. We also performed tests on running Ardán at faster than real-time, at 200% speed. In this mode, the game engine and its physics engine match the speed of the simulator, resulting in all activity increasing in speed. The results in figure 6 show that roughly half the number of nodes can be simulated in time with the game engine, with minimal fluctuation.

## 4 Related Work

Previous work on sensor network simulators [12, 10, 2, 9], has focused on creating efficient, accurate and scalable solutions, at a variety of simulation levels. However, they provide little support for controlling or simulating external input into the network, relying on the use of hard-coded input, trace-fed data, scripts or manual interaction with the simulated devices. Whilst useful in some cases, these techniques limit the scope of testing, slow to create/update and inflexible.

Recent work by Mueller et al. [11] demonstrates the use of a 3D game engine for prototyping a closed-loop control system for an electric two wheel vehicle. The work utilises a 3D game engine to provide input/output for simulating the interaction with forces in the virtual world. Similarly, 3D simulation has also been used for design and testing of visual algorithms for robotic traversal and navigation of virtual environments [4, 7].

## 5 Conclusion

We demonstrated a novel approach for testing a cyber-physical, through the use of a 3D game engine to simulate physics and crowds. Through the corridor case study, we have shown how this approach can assist development of applications and is scalable. In future, we envision integrating additional tools to Ardán, such as test and monitoring frameworks, and model checking. Recent innovations in virtual reality (VR) headsets, such as the HTC Vive, also offer exciting possibilities into VR cyber-physical application design, placement and testing.

## References


1. Havok. http://havok.com/, 2016
2. NS1/2/3 - Network Simulator. https://www.nsnam.org/, 2015
3. Physx. https://developer.nvidia.com/gameworks-physx-overview/, 2016
4. Agero, C.E., et al.: Inside the virtual robotics challenge: Simulating real-time robotic disaster response. IEEE Transactions on Automation Science and Engineering 12(2), 494–506 (April 2015)
5. Baumgartner, T., et al.: Virtualising testbeds to support large-scale reconfigurable experimental facilities. In: Silva, J., Krishnamachari, B., Boavida, F. (eds.) Wireless Sensor Networks, Lecture Notes in Computer Science, vol. 5970, pp. 210–223. Springer Berlin Heidelberg (2010)
6. Boano, C.A., Zúñiga, M., Brown, J., Roedig, U., Keppitiyagama, C., Römer, K.: Templab: A testbed infrastructure to study the impact of temperature on wireless sensor networks. In: Proceedings of the 13th International Symposium on Information Processing in Sensor Networks. pp. 95–106. IPSN '14, IEEE Press, Piscataway, NJ, USA (2014)
7. Carpin, S., Lewis, M., Wang, J., Balakirsky, S., Scrapper, C.: Usarsim: a robot simulator for research and education. In: Robotics and Automation, 2007 IEEE International Conference on. pp. 1400–1405 (April 2007)
8. Chatzigiannakis, I., Fischer, S., Koninis, C., Mylonas, G., Pfisterer, D.: Wisebed: An open large-scale wireless sensor network testbed. In: Komninos, N. (ed.) Sensor Applications, Experimentation, and Logistics, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol. 29, pp. 68–87. Springer Berlin Heidelberg (2010)
9. Fekete, S., Kroller, A., Fischer, S., Pfisterer, D.: Shawn: The fast, highly customizable sensor network simulator (2007)
10. Levis, P., Lee, N., Welsh, M., Culler, D.: Tossim: Accurate and scalable simulation of entire tinyos applications. In: Proceedings of the 1st International Conference on Embedded Networked Sensor Systems. pp. 126–137. SenSys '03, ACM, New York, NY, USA (2003)
11. Mueller, W., Becker, M., Elfeky, A., DiPasquale, A.: Virtual prototyping of cyber-physical systems. In: Design Automation Conference (ASP-DAC), 2012 17th Asia and South Pacific. pp. 219–226 (Jan 2012)
12. Österlind, F., Dunkels, A., Eriksson, J., Finne, N., Voigt, T.: Cross-level sensor network simulation with cooja. In: Local Computer Networks, Proceedings 2006 31st IEEE Conference on. pp. 641–648 (Nov 2006)
13. Pavkovic, B., Radak, J., Mitton, N., Rousseau, F., Stojmenovic, I.: From real neighbors to imaginary destination: Emulation of large scale wireless sensor networks. In: Li, X.Y., Papavassiliou, S., Ruehrup, S. (eds.) Ad-hoc, Mobile, and Wireless Networks, Lecture Notes in Computer Science, vol. 7363, pp. 459–471. Springer Berlin Heidelberg (2012)
14. Sventek, J., Koliousis, A.: Unification of publish/subscribe systems and stream databases: The impact on complex event processing. In: Proceedings of the 13th International Middleware Conference. pp. 292–311. Middleware '12, Springer-Verlag New York, Inc., New York, NY, USA (2012)
15. Wen, Y., Zhang, W., Wolski, R., Chohan, N.: Simulation-based augmented reality for sensor network development. In: Proceedings of the 5th International Conference on Embedded Networked Sensor Systems. pp. 275–288. SenSys '07, ACM, New York, NY, USA (2007)