An Open Simulator framework for 3D Visualization of Digital Twins

Ashish Joglekar
AI and Robotics Technology Park (ARTPARK)
Indian Institute of Science
Bangalore, India
ashishj@iisc.ac.in

Gaurav Bhandari
RBCCPS
Indian Institute of Science
Bangalore, India
gauravbhandari11@gmail.com

Rajesh Sundaresan
Electrical Comm. Engg. and RBCCPS
Indian Institute of Science
Bangalore, India
rajeshs@iisc.ac.in

2022 IEEE International Conference on Internet of Things and Intelligence Systems (IoTaIS) | 979-8-3503-9645-4/22/$31.00 ©2022 IEEE | DOI: 10.1109/IoTaIS56727.2022.9975980

***Abstract*—Production Digital Twins (DTs) mirror and interact with the production lines that they model through the Industrial Internet of Things (IIoT) based bidirectional data flow pipelines. There is a need for interactive 3D visualization of DTs to unlock the promised capabilities for real time monitoring, optimization, reconfiguration, maintenance and control of the production process. DTs based on open source frameworks like SimPy lack an interactive 3D visualization frontend. This paper proposes a generic open source framework for the 3D visualization of any Discrete Event Simulation (DES) based production DT. As an example, an interactive 3D visualization of a SimPy based DT of a real Surface Mount Technology (SMT) Printed Circuit Board (PCB) line is presented. We visualize machine states, process flow, energy and throughput metrics of the DT and the real line in 3D. We believe that the proposed 3D visualization framework can help ease model validation efforts and can enable interactive "what if" analysis and control for optimization of the production process.**

***Index Terms*—Digital twin, 3D visualization, user interface, discrete event simulation, IRC, assembly line**

I. Introduction

A Digital Twin (DT) is a virtual representation of a physical product or process. A production DT is a virtual model that reflects the state of a production/assembly line. The actual assembly line (henceforth real line) is instrumented with sensors to capture production process states and other vital metrics. Data from the real line is then relayed to the DT using the modern sensing-cum-networking paradigm of Industrial Internet of Things (IIoT). Once the DT is updated with data from the real line, it can be used in process monitoring, "what if" simulations, optimizations, planning/design, maintenance, production process management and safety applications [1].

A discrete event simulation (DES) model is one of the technology enablers for the development of a production DT [3], [4]. For DTs to be useful to factory floor managers, an intuitive user interface (UI) is needed. Commercial DES tools do provide graphical user interfaces to improve user experience [5]–[7]. However, free and open source DT frameworks like SimPy lack inbuilt and interactive visualization of the simulated production process [5]. As examples, the work of [8] is a project that visualized a SimPy based DES in Maya, which is a commercial 3D animation and rendering tool. SimPy results were visualized in 2D in [9]. A few open source DES frameworks like DESMO-J provide 2D and 3D user interface libraries (as extensions to the underlying framework) but these are still in development [10]. In [11] multiple DT instances were simulated and state variables were visualized on isochrone maps with seamless replay and jump-to-point functionality. The work in [12] combined DT with Virtual Reality (VR) technology. None of these, nor any other work to the best of our knowledge provide any generic open source 3D visualization framework for DES based production DTs.

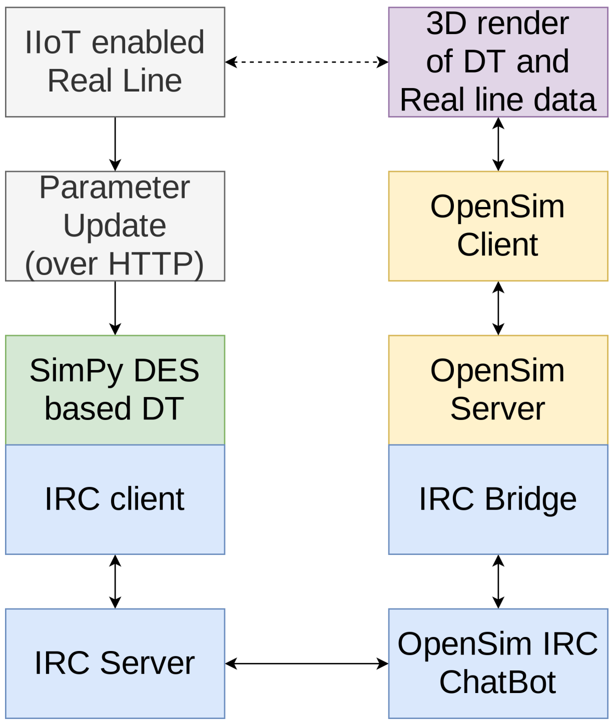
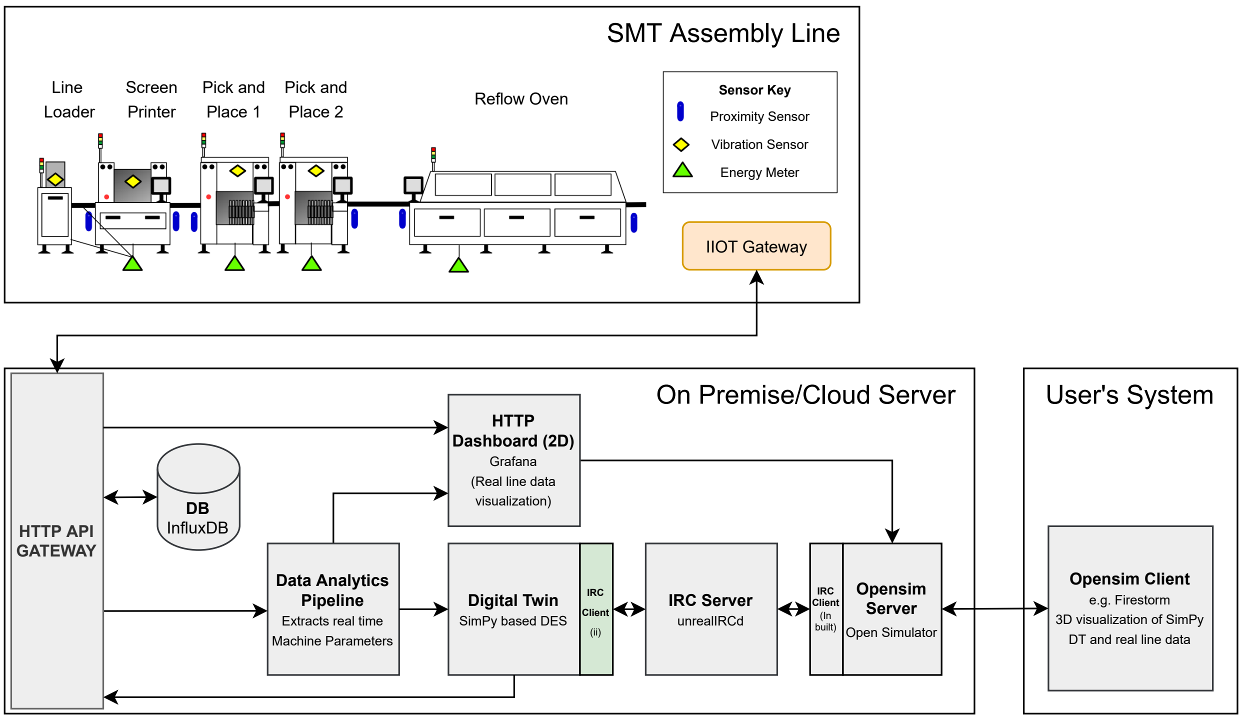


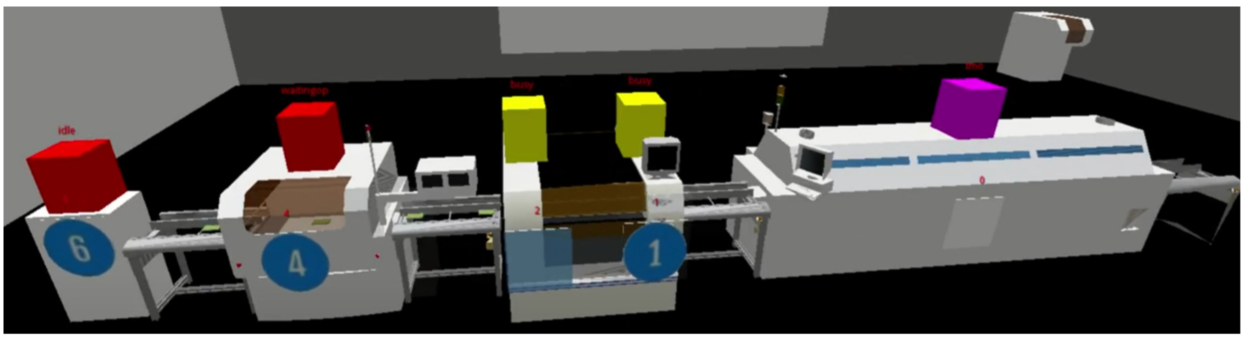
Fig. 1: Proposed 3D visualization framework

To fill the above-mentioned gap, we propose a generic 3D visualization framework for production DTs that is agnostic to the underlying DES framework, see Fig. 1. In our solution, we have leveraged the capabilities of the open source multi plat-

This work was supported by MHRD, SERB IRRD of DST, and TCS under UAY grant number IISc001



(a) Real, instrumented SMT PCB line [2] and architecture of the proposed 3D visualization framework



(b) DT visualized in an OpenSim Client on a user's system in 3D

Fig. 2: Instrumented SMT PCB line and its corresponding DT

form 3D applications server called Open Simulator (OpenSim) [13]. Our 3D visualization framework allows for a) creation of the virtual factory floor/ assembly line in 3D by importing computer-aided design (CAD) models b) re-configuration of the SimPy based digital twin of the assembly line using a simple drag and drop UI in 3D (which is currently done through configuration files) c) simultaneous visualization of the data from the real line and its digital twin in a virtual 3D environment d) deployment of virtual Human Machine Interface (HMI) panels in the 3D virtual environment for remote control of the real line e) visualization of human agents interacting with the line including manual interventions to trigger events during a DES run, and g) ability to visualize state of concurrent DTs to perform "what if" analysis.

Though the architecture of our 3D visualization framework is generic, we present its use for a specific case study. Fig. 2 shows the block diagram schematic of our real instrumented Surface Mount Technology (SMT) assembly line and its corresponding digital twin rendered in OpenSim.

Section 2 of this paper gives a brief description of a SimPy DES based DT of a real SMT assembly line. We argue the need for an interactive 3D visualization framework after considering the lack of visibility of the complex DT's internal states and processes. In section 3, we present various components of the software architecture needed to build a generic DT 3D visualization framework. These include a) OpenSim 3D visualization server and client b) an Internet Relay Chat (IRC) bridge between SimPy and OpenSim c) 3D rendering of a virtual factory floor in an OpenSim client including scripting of objects/processes, and d) concurrent visualization of multiple DT configurations and real time data from the physical line. We also highlight the use cases of our interactive

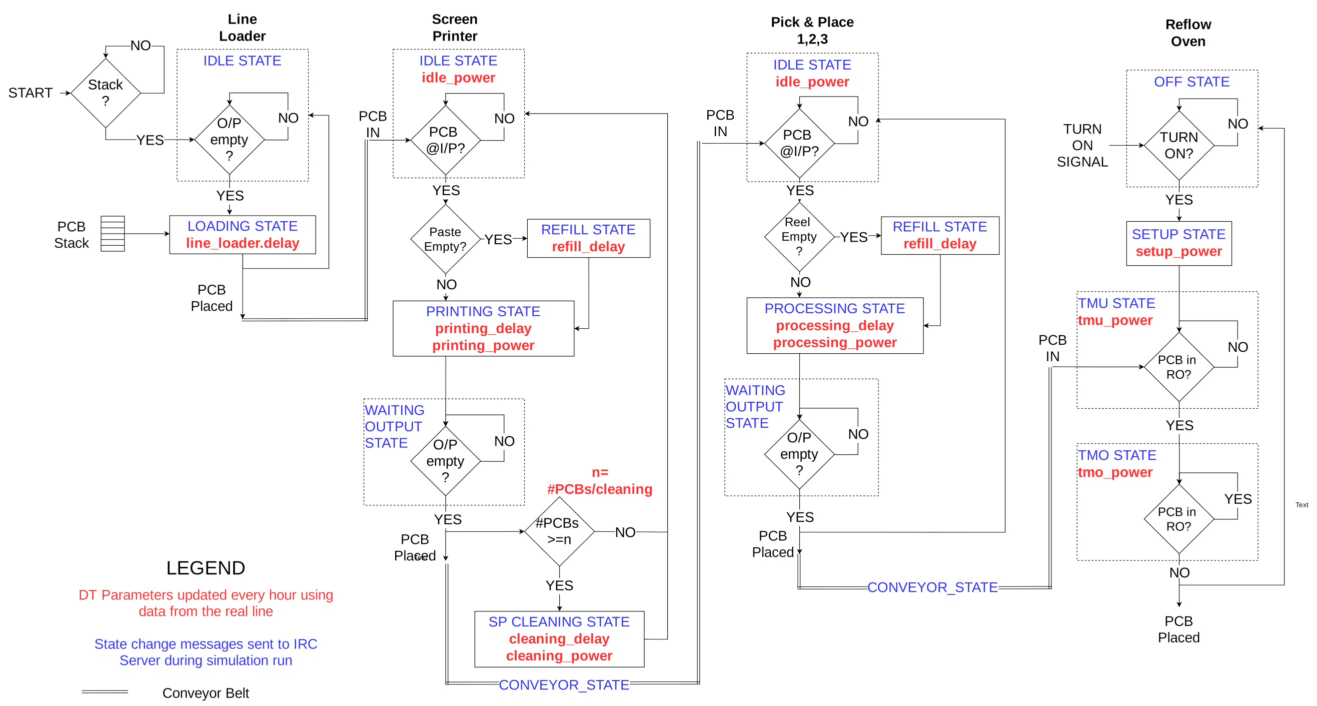


Fig. 3: DES of an SMT PCB line. Parameters that receive hourly updates and state change messages are highlighted

3D visualization framework with a case study on its use in the energy optimization of a real SMT Printed Circuit Board (PCB) assembly line. Finally, in section 4, we list features that can be added to our 3D visualization framework.

II. SimPy DES based DT of an SMT Line

Let us consider a production DT of a real SMT PCB assembly line based on a discrete event simulation (DES) SimPy model [2]. Fig. 2a is a schematic representation of a typical PCB SMT line. Fig. 2a also shows the logical and physical architecture of the proposed 3D visualization framework. Fig. 2b shows the Digital Twin of this line visualized in 3D using our OpenSim based framework. The machines on the line and human agents interacting with the line were modelled in [2] as SimPy processes. Fig. 3 shows the states that were modelled in SimPy to capture the behaviour of every real machine and intervening agents on the line (in blue). We instrumented the real line with sensors to infer various machine parameters and processing states. The corresponding energy consumption pattern was also recorded. This data was fed to the DT and the parameterized SimPy model of every machine was updated on an hourly basis. This data driven model of the line was able to achieve accurate estimates of throughput and energy use. The parameters that received hourly updates are highlighted in red in Fig. 3.

Fig. 3 reveals that even a simple assembly line like an SMT PCB line has many complex states that need representation. Presenting these states on a console makes it difficult for an observer to understand trends and make decisions on how to optimize the line. One is presented with a report only at the end of the simulation run with no/limited visibility of internal process states during the simulation run. Human intervention (during a run) also becomes difficult with this approach. Comparison of two concurrently running DTs, to explore what if scenarios, is also difficult with this approach particularly when the DES framework lacks an intuitive user interface. There is therefore a need for an advanced visualization framework to unlock the promised capabilities of the production DT.

III. Proposed 3D-visualization framework

Fig. 1 shows the various elements of the visualization framework.

We have leveraged the capabilities of the open source multi platform 3D applications server Open Simulator (OpenSim) [13] for 3D visualization of any DES based DT. The OpenSim framework follows a client server architecture. The client renders the 3D environment while the server listens for connections, manages regions and users. Any OpenSim compatible client can be configured to connect with the OpenSim server (yellow boxes).

OpenSim provides an inbuilt IRC bridge module. The IRC based Application Programming Interface (API) between any DES based DT and OpenSim (blue boxes) is our key contribution that enables the proposed generic 3D DT visualization framework. Data flow over this IRC bridge between OpenSim and the SimPy based DT is bidirectional. This bidirectional API allows for a) visualization of the digital twin's state in the 3D virtual environment in real time, b) re-configuration of the digital twin by broadcasting relative placements of assembly line components (once they are dragged and dropped

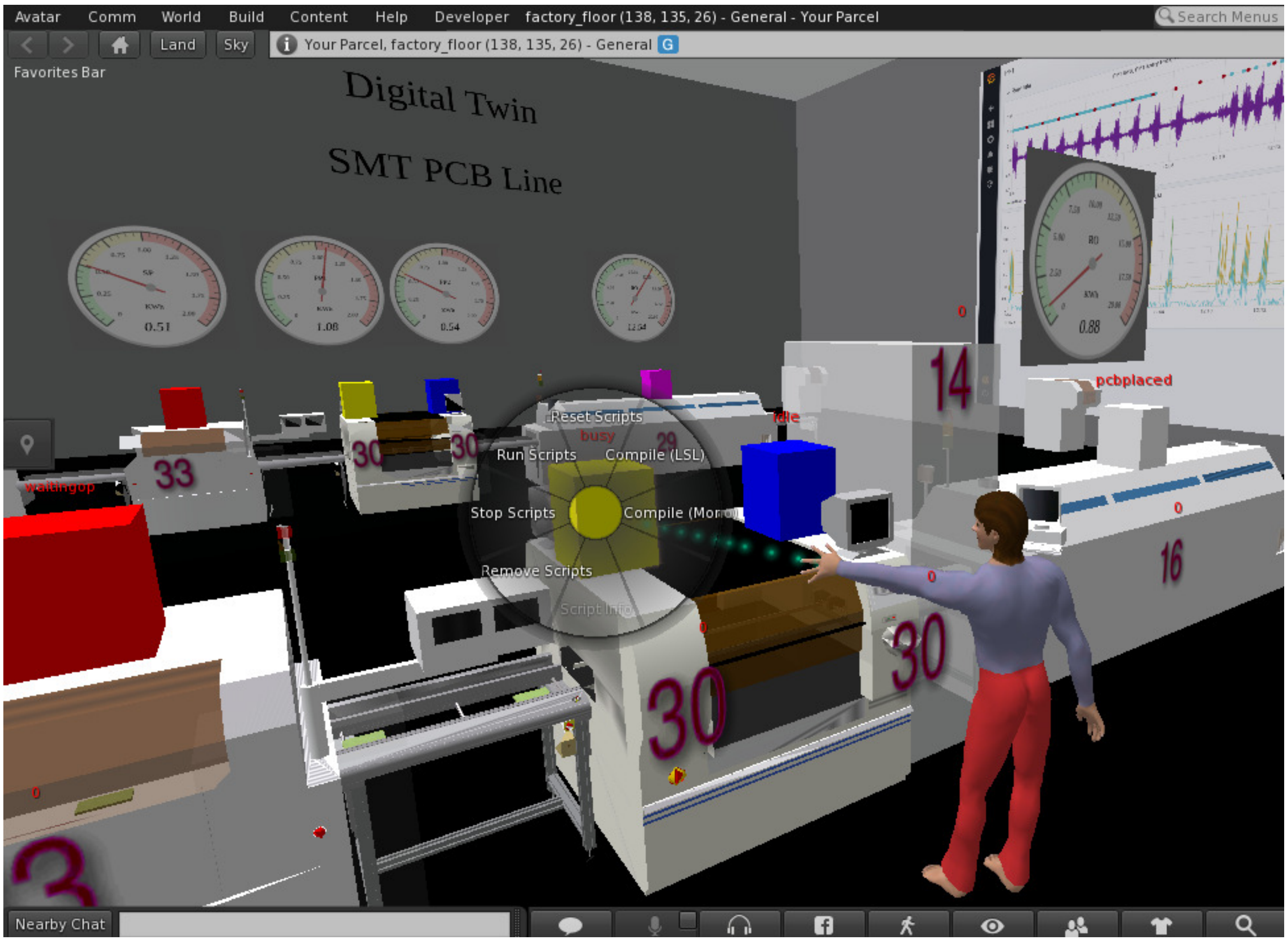


Fig. 4: Visualization of concurrent DTs. Also shown is a virtual human avatar interacting with the line using HMI panels

in 3D) to create a configuration file that specifies the layout and interconnections in a format compatible with SimPy (thus avoiding the need to directly edit SimPy's configuration files), and c) passing manual process triggers to SimPy to analyse certain test cases during a SimPy run.

At the back-end, the SimPy models are updated hourly with process and machine parameters being inferred from observations on the real instrumented line.

The main advantage of the proposed software architecture is that it can scale with the complexity of the production process. Multiple IRC servers may be instantiated for scalability. The IRC bridge module also supports a multi-channel mode which can connect to multiple IRC channels across multiple IRC servers.

IRC is not the only communication bridge between OpenSim and the external real world. Data from the real line (indicated by the dotted line link in Fig. 1) can also be visualized as an Hypertext Markup Language (HTML) data dashboard in the virtual world. Interactive virtual HMI panels can also be deployed in the 3D environment for remote control of the real assembly line process.

A. *OpenSim's IRC bridge*

Objects in OpenSim communicate with each other in the virtual world over various chat channels. An object can be scripted to listen or talk on a specific channel in single channel mode or on multiple channels in multi-channel mode. For communication with external entities, the OpenSim server provides an inbuilt IRC bridge module. An IRC client can use this bridge to relay all chat messages from any configured IRC server to a specific configured channel in the virtual world. All objects in the virtual world can listen and talk on this channel.

Each OpenSim client is also associated with an *avatar*. An avatar is a virtual representation of the user that can interact with objects in the virtual world. Avatars also join the IRC server automatically via the IRC bridge on login. In the context of a production assembly line, the avatars may represent workers interacting with the line.

We propose an API that uses the IRC bridge to visualize a DT in OpenSim. If we associate each machine in the DT with an IRC chatbot, messages can be relayed to and from the DT to the 3D virtual world. This approach is also agnostic to the underlying DT framework. A messaging API has been developed to gain visibility into the simulation's internal states and process variables. Each IRC chatbot sends messages corresponding to the machine's current state. Events in the simulation trigger these IRC messages indicating the current state. Example process states specific to the SMT PCB assembly line are marked in blue in Fig. 3.

Certain IRC client configuration scripts are run to create chatbots with specific nicknames for every state and/or process variable to be visualized. Table I lists the DES states variables and their representative chatbots for our SMT PCB line use case.

The current implementation uses the ii IRC client [14] to send messages from Simpy to Opensim over the IRC bridge. An IRC directory tree is instantiated with the following path structure "path-prefix/server-url/channel-name/in(out)" during the ii IRC client's initialization. Any message written to the "in" FIFO file from SimPy is sent over the IRC bridge. There is a unique "in" file for every nickname and channel. SimPy can also receive messages over any IRC channel by reading contents of the "out" file. For example, the screen printer's state can be changed to the printing state by writing the keyword "printing" to "spstate/127.0.0.1/opensim/in". The keyword is chosen from the machine specific set of keywords defined in the corresponding state array in Simpy, see Table I. The appropriate keyword is written to the "in" file after every state change event in Simpy.

IRC guarantees a delay Quality of Service (QoS) metric of $< 200msec$ on a bandwidth constrained network. With a DES time step of $0.5sec$ this network delay metric is acceptable. IRC guarantees zero data loss by using a robust re-transmission policy [15]. Multiple IRC servers may be instantiated for scalability. We use an IRC chat client to debug messages on the IRC channel. In our implementation, the DES and the IRC server have been hosted on the same server computer.

B. *Virtual object scripting in OpenSim*

We can define the behaviour of the virtual objects in OpenSim through object scripting. An Open Source Scripting Language (OSSL) script can be embedded in every object that is imported (as a CAD model) or created in the virtual world. The script can act to animate or change state of objects on receiving a message trigger from the IRC relay channel. Other examples of interactive object scripts include movement of components/ robotic arms on the virtual line, visualization of humans interacting with the line, etc.

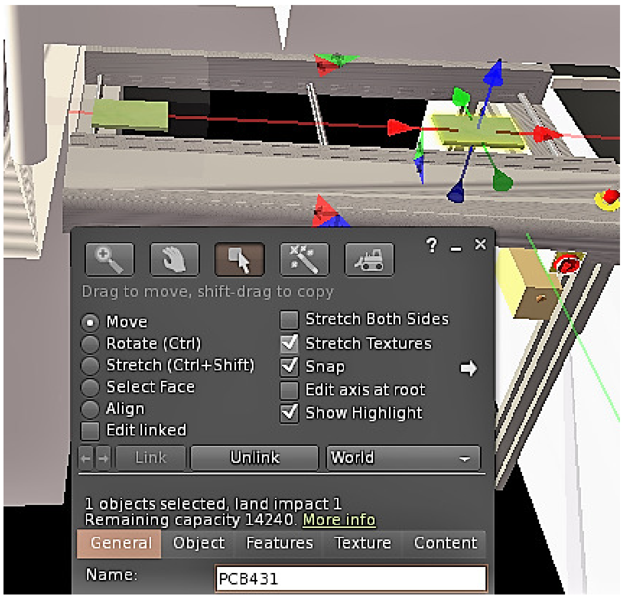
C. *3D Visualization in the OpenSim client*

The user can import 3D CAD mesh files using the OpenSim client's Graphical User Interface (GUI). This is a one-time process when setting up the virtual factory floor. The imported CAD model can then be dragged from the inventory and dropped at any location in the virtual factory floor. Custom 3D objects can also be created using the build menu of the client. Every object can be embedded with a custom OSSL script to define object behaviour. The client features an inbuilt script editor and compiler.

TABLE I: IRC messages corresponding to each chatbot

Chatbot Name	Description	Message Types
lloaderstate	Line Loader States	pcbplaced, idle, loading, unloading
spstate	Screen Printer States	cleaning, printing, waitingop, pcbplaced, spstall
ppstate	Pick and Place Machine States	waitingop, pcbplaced, busy, idle, ppstall
rostate	Reflow oven states	tmun, tmo, setup, pcbplaced
sptoppbelt	Conveyor Belt state form SP to PP	2^n occupancy states
rfobelt	Conveyor Belt state at entry of RO	2^n occupancy states

Fig. 4 shows the ability to visualize concurrent DT runs. Fig. 4 also shows how human avatars may be visualized interacting with the line. Virtual HMI panels may also be deployed to pass manual event triggers to the DT or to send control commands to the real line. In this specific use case, two SMT PCB line configurations have been visualized on the 3D factory floor; one with and one without a line buffer interspersed between the pick and place machine and the reflow oven. Certain performance metrics have also been visualized in real time on the virtual factory floor during the concurrent DT runs. For example, every machine's throughput has been visualized using numbers projected on each machine as textures on 3D objects. Energy use has been presented using gauges displayed on top of relevant machines. Machine states have been visualized by dynamically changing the color of cubes placed on top of every machine. This DT based "what-if" analysis helped identify the optimum line configuration for energy use [2]. The 3D visualization provided an intuitive spatio-temporal context

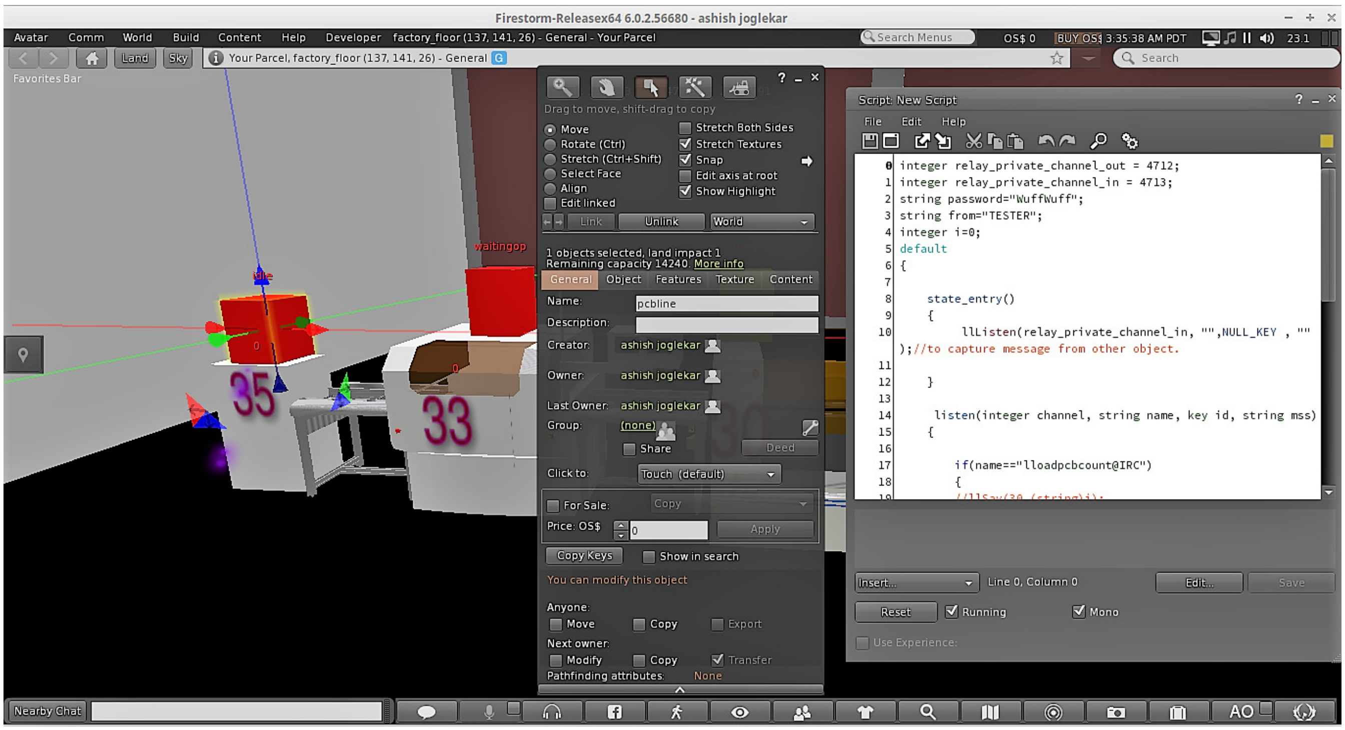

Drag to move, shift-drag to copy

● Move
○ Rotate (Ctrl)
○ Stretch (Ctrl+Shift)
○ Select Face
○ Align
☐ Edit linked
☐ Stretch Both Sides
✓ Stretch Textures
✓ Snap
☐ Edit axis at root
✓ Show Highlight

1 objects selected, land impact 1
Remaining capacity 14240. More info

Name: PCB431

Fig. 5: Custom object builder panel. A PCB was created using this tool



Name: pcbline
Creator: ashish joglekar
Owner: ashish joglekar
Last Owner: ashish joglekar
Group: (none)
Click to: Touch (default)
Price: OS$ 0
1 objects selected, land impact 1
Remaining capacity 14240 More info
You can modify this object

```
integer relay_private_channel_out = 4712;
integer relay_private_channel_in = 4713;
string password="WuffWuff";
string from="TESTER";
integer i=0;
default
{

    state_entry()
    {
         llListen(relay_private_channel_in, "",NULL_KEY , ""
);//to capture message from other object.

    }

     listen(integer channel, string name, key id, string mss)
    {

        if(name=="lloadpcbcount@IRC")
        {
```

Line 0, Column 0

Fig. 6: Inbuilt OSSL script editor in the OpenSim client

to the DTs internal states and variables. For example, the floor manager could, at a glance, identify bottlenecks on the line for a given line configuration, without the need to parse the complex output logs generated by concurrent DT runs.

Fig. 5 shows the drag and drop interface to create custom objects in the virtual world. For our SMT line, a PCB object was created using these build tools. This PCB is spawned by the line loader script and moved around by the scripts embedded in the conveyor belt depending on the state change updates sent by the DT over the IRC link. The script embedded in the virtual conveyor belt updates the board's position on the belt depending on the 2^n possible occupancy states communicated by the conveyor modelled in the DT.

Fig. 6 shows the client's inbuilt script editor. A user script can be attached to any object in the virtual world.

A demo video of the SMT line DT visualized using our framework is available at [16].

IV. Conclusions and Future Work

This paper proposes a framework for 3D visualization of data from the real line and its concurrently running Digital Twin(s). The bidirectional IRC based communication bridge that connects the DT to the open source 3D virtual environment OpenSim is agnostic to the underlying DES framework. This is our key contribution. The framework allows a user to a) import CAD objects as mesh models in the virtual world b) program custom object behaviours using scripts that are embedded in the virtual object and c) render performance metrics/process states as object textures or fully featured HTML dashboards. Depending on the DT use case, human avatars may also be visualized on the virtual factory floor. HMI panels may also be deployed on the virtual factory floor for remote control of the real line and/or its DT. We believe that the proposed 3D visualization framework can help ease model validation efforts and enable interactive "what if" analysis and control for optimization of the production process.

The proposed framework has been used to visualize data from an SMT PCB line and its Digital Twin. Visualization of concurrent DT runs has been useful to verify the effect of process or line re-configurations. We are currently working on three additional features that leverage the capabilities of the bidirectional data link between the DT and OpenSim.

1) Re-configuration of the line requires modifications in both the SimPy code and the 3D CAD model. We want to simplify this process as part of the next version of this framework. An intuitive drag and drop abstraction layer can be created for re-configuring the simulation's line connectivity and process parameters. For instance, when a user adds or removes a machine in the 3D virtual world, the file that describes line connectivity in SimPy can be modified automatically before the next or concurrent simulation run.
2) We also want to enable feature rich virtual HMI panels to allow real time remote control of the assembly line process.
3) A user currently interacts with his avatar and objects in the virtual world using a keyboard and mouse. Upcoming VR technologies may help make this interaction

more immersive, thus improving the situational awareness of the factory floor manager.

Acknowledgment

This work was supported by UAY Scheme (project no. 13) by MHRD, Govt. of India in collaboration with Tata Consultancy Services (TCS). We deeply thank Vinyas Innovative Technologies for permitting us to use their facility for instrumentation and analysis. Authors also thank Dr. Devadatta Kulkarni, Dr. Jefferey Tew and Mr. Vijayanand Parabgoanker from TCS for their valuable inputs.

References

[1] A. Madni, C. Madni, and S. Lucero, "Leveraging Digital Twin Technology in Model-Based Systems Engineering," *Systems*, vol. 7, no. 1, p. 7, Jan 2019. [Online]. Available: http://dx.doi.org/10.3390/systems7010007

[2] N. Karanjkar, A. Joglekar, S. Mohanty, V. Prabhu, D. Raghunath, and R. Sundaresan, "Digital Twin for Energy Optimization in an SMT-PCB Assembly Line," in *2018 IEEE International Conference on Internet of Things and Intelligence System (IOTAIS)*, 2018, pp. 85–89.

[3] W. Kritzinger, M. Karner, G. Traar, J. Henjes, and W. Sihn, "Digital Twin in manufacturing: A categorical literature review and classification," *IFAC-PapersOnLine*, vol. 51, no. 11, pp. 1016 – 1022, 2018, 16th IFAC Symposium on Information Control Problems in Manufacturing INCOM 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2405896318316021

[4] C. Cimino, E. Negri, and L. Fumagalli, "Review of digital twin applications in manufacturing," *Computers in Industry*, vol. 113, p. 103130, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0166361519304385

[5] G. Dagkakis and C. Heavey, "A review of open source discrete event simulation software for operations research," *Journal of Simulation*, vol. 10, no. 3, pp. 193–206, 2016.

[6] "Prespective: Digital Twin Software for Unity," https://prespective-software.com/.

[7] "Siemens Tecnomatix for Plant Simulation," https://www.plm.automation.siemens.com/global/en/products/tecnomatix/.

[8] B. Xhika, "DES with SimPy and Maya," https://sourceforge.net/projects/dessimpymaya/.

[9] B. T. Ersson, L. Jundén, U. Bergsten, and M. Servin, "Simulated productivity of one- and two-armed tree planting machines," *Silva Fennica*, vol. 47, 01 2013.

[10] J. Göbel, P. Joschko, A. Koors, and B. Page, "The Discrete Event Simulation Framework DESMO-J: Review, Comparison to Other Frameworks and Latest Development," 05 2013, pp. 100–109.

[11] L. Atorf and J. Roßmann, "Interactive Analysis and Visualization of Digital Twins in High-Dimensional State Spaces," in *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, 2018, pp. 241–246.

[12] V. Havard, B. Jeanne, M. Lacomblez, and D. Baudry, "Digital twin and virtual reality: a co-simulation environment for design and assessment of industrial workstations," *Production & Manufacturing Research*, vol. 7, no. 1, pp. 472–489, 2019.

[13] "OpenSimulator Open Source Project," http://opensimulator.org/.

[14] "Open Source ii IRC Client," https://tools.suckless.org/ii/.

[15] Y. Chen, T. Farley, and N. Ye, "QoS requirements of network applications on the internet," *Inf. Knowl. Syst. Manag.*, vol. 4, no. 1, p. 55–76, Jan. 2004.

[16] "Demo Video of our proposed 3D visualization framework," https://www.youtube.com/watch?v=Jtchcw8KPM4.