INTERNATIONAL JOURNAL
ON INFORMATICS VISUALIZATION

journal homepage : www.joiv.org/index.php/joiv

Autonomous Agents in 3D Crowd Simulation Through BDI Architecture

Sim Keng Wai[a], Cheah WaiShiang[a,*], Muhammad Asyraf bin Khairuddin[a], Yanti Rosmunie Binti Bujang[a], Rahmat Hidayat[b], Celine Haren Paschal[a]

[a]*Faculty of Computer Science and Information Technology, Universiti Malaysia Sarawak, 94300, Malaysia*
[b]*Department of Information Technology, Politeknik Negeri Padang, Sumatera Barat, Indonesia*
Corresponding author: [*]wscheah@unimas.my

***Abstract*— Agent based simulation (ABS) is a paradigm to modelling systems included of autonomous and interacting agents. ABS has been tremendous growth and used by researchers in the social sciences to study socio-environmental complex systems. To date, various platforms have been introduced for agent-based social simulation. They are rule based in any logic, python based in SPADE and etc. Although those platforms have been introduced, there is still an insufficient to develop a crowd simulation in 3D platform. Having a 3D platform is needed to enabling the crowd simulation for training purposes. However, the current tools and platform still lack features to develop and simulate autonomous agents in the 3D world. This paper introduced a BDI plug in at Unity3D for crowd simulation. BDI is an intelligent agent architecture and it is able to develop autonomous agents in crowd environment. In this paper, we present the BDI plug with a case study of Australia bush fire and discuss a method to support autonomous agents' development in 3D crowd simulation. The tool allows the modeller to develop autonomous agents in 3D world by taking the advantages of Unity3D.**

***Keywords*— Autonomous agents; BDI architecture; unity 3D; crowd simulation.**

Manuscript received 15 Dec. 2020; revised 29 Jan. 2021; accepted 3 Mar. 2021. Date of publication 31 Mar. 2021.

I. Introduction

The agent is a piece of software that mimics human intelligence. Agents perceive and influence aspects of their environments, and they can learn. An agent is a computer entity located in an environment to achieve its goals flexibly and autonomously. The agents are in an environment called 'worlds' which can be from simple to complex according to the needs. The agent will also present, derived with human behavior in an organization, to control and structure the constructed model. The agents will interact to transfer information among the agents and the environment in several ways, such as communication, perception, concurrence, cooperation, coordination, and negotiation.

To date, agent technology is widely used in simulation to understand human behavior and emerging behavior. Also known as agent-based simulation or agent-based social simulation, it consists of an organized set of agents that can interact with each other in a virtual world [1].

Various platforms have been introduced for agent-based social simulation. They are rule-based in any logic, python-based in SPADE, etc. Although those platforms have been introduced, there is still insufficient to develop a crowd simulation in a 3D platform. Having a 3D platform is needed to enabling crowd simulation for training purposes. However, the current tools and platforms still lack features to develop and simulate autonomous agents in the 3D world. This paper introduced a BDI plug-in at Unity3D for crowd simulation. BDI is an intelligent agent architecture, and it can develop autonomous agents in a crowded environment. In this paper, we present the BDI plug with a case study of Australia bush fire and discuss a method to support autonomous agents' development in 3D crowd simulation. The tool allows the modeler to develop autonomous agents in 3D world by taking advantage of Unity3D.

A. Related Work

The development of BDI into agent-based simulation has to receive much attention [2]. Three approaches are introduced to implement BDI into agent-based simulation [2]. They are extending the agent BDI programming platform with the features and properties of agent-based simulation. On the other hand, works have been to extend the agent-based

simulation platform like Gama, Analogic, and BDI capabilities. Finally, researchers work on integrating BDI programming platform with agent-based simulation. The summary of the related works in developing BDI into agent-based simulation is shown in Table I.

TABLE I
ACTORS AND ROLE DESCRIPTIONS

Approaches	Works
Extending the BDI platform with simulation capabilities	[9]
Extending the agent simulation platform with BDI capabilities	[4], [5]
Integrating BDI platform with agent simulation platform	[6], [7], [8], [10], [11], [12]
Developing BDI scripting for Unity3D platform	[13], [14]

Work has been done to develop 3D simulation platform to study the problem faced by disabilities in the workplace [3]. The JADE multi-agent platform is adopted with simulation capabilities to enable the modeler to generate 3D models of the workplace. JADE is a multi-agent programming platform. Although JADE supports behavior-based agent development, the author claimed it is a BDI platform.

Taillandier [4] introduced simple reactive models like BDI for GAMA simulation. In this case, GAMA language is adopted to develop BDI architecture when the modeling agent in GAMA simulation platform. On the other hand, [5] has adopted the AnyLogic language to develop BDI agent when simulating an intelligent agent in the AnyLogic simulation platform [5].

Work has been done to integrate JADEX BDI agent into the unreal games engine [6]. The integration of BDI agent platform into game engines is important to model a more realistic computer game character. JADEX [6] is a rational agent development platform where the JADEX system realized on belief, desire, intention (BDI) model at the design and implementation layer. The agents' beliefs, goals, and plans are defined in XML files, and the plan bodies are written in Java.

JADEX BDI reasoning framework is integrated and extended with learning component to Unity3D games engine to control self learning agents in 3D virtual words. The integration of JADEX BDI agent platform to FiVES 3D environment [7]. Finally, GOAL programming is integrated into unreal games engine to develop an intelligent agent for 3D virtual worlds simulation. The same objective was also investigated by the work [8] to connect or integrate BDI JASON into a games engine to simulate a virtual classroom. Jason is a platform for the development of BDI agent systems through AgentSpeak. The AgentSpeak has been one of the most influential abstract languages based on the BDI architecture.

Our experience in integrating BDI platform with games engines has revealed the middleware's complexity when developing BDI agent to control NPC in the virtual worlds. This has been highlighted that cognitive agent development is challenging and there is a need to reduce the complexity of field-expert modeler when adopting BDI agent and reduce the computational cost in deploying cognitive agent architectures [4]. Meanwhile, most agent-based simulations like SWARM, Dollie, Grant, and Hooper [10] provided limited support for the simulation to develop involved cognitive agents and case studies. Based on our experience, some of the platforms are lacking detailed documentation, it ends up implementing the BDI into agent-based simulation or extending the agent-based simulation platform with BDI is a trivial task. Besides, some of the libraries are up to date and it is a challenge to refer to the programmer for further clarification. On the other hand, most of the agent platforms do not have the option to make the simulation in 3D view except for NetLogo but limited.

Our work is in line with Ni *et al* [13] and Sudkhot and Sombattheera [14] that introduced a BDI script at Unity3D to design and develop the virtual world environment at Unity3d. Meanwhile, our ultimate goal is to allow the extensibility and modifiability of the BDI plug among the developer.

II. MATERIAL AND METHOD

Unity3D is a real-time 3D development platform consisting of a rendering and physics engine and a graphical user interface called the Unity Editor [15]. It is a powerful tool for the simulation of visually realistic worlds with sophisticated physics and complex interactions between agents with varying capacities. Unity3D enables real-time simulation in interactive mode and can simulate frames even faster than real-time in offline mode. By developing projects in Unity3D that represent real-world environments, they can holistically understand complex systems to inform key policy or design decisions.

As mentioned before, Unity3D has been used by researchers in modeling and simulating spatial environments. However, Unity3D does not support developing BDI agents. Although there is a BDI tool introduced by Poli [15], there are some limitations. It does not support rules with more than ten subgoals. For example, if you plan to write a complex rule to define a human behaviour with many goals, the user needs to break down the rule into sub-rules of no more than ten subgoals each. Therefore, we built a BDI tool for Unity3D to showcase our proposed method for modeling human decision-making in fire evacuation simulation.

In this paper, our BDI plug in is built using the standard programming language provided by Unity3D. The requirements to develop BDI simulation model in Unity3D are following:

- Create an agent object, attach the “Field of View” Script into it.
- The “Agent” script is the template file for defining the desire and setting up an agent's attribute. User may rename the script name to the role of the agent and set the goals, for example, SubGoal s1 = new SubGoal("belief" ,1 ,true), goals.Add(s1, 1). Then attach it to agent object.
- The “Action” script is the template file for setting up the desire. User may rename the script to the desire of

the agent. Then attach it to agent object.

The main concept behind the tool is the "Field of View" and "Action" scripts. The "Field of View" script is used to perceive the surrounding of an agent and to update its belief. The "Action" script is used to describe the action of an agent. Each of the "Action" scripts represents an agent's action and starts to perform it if the condition or belief is met. All the action will be added to the action list. If the latter evaluates to true, the action is removed from the list, and the next action is executed. If the action list is empty, then the agent does nothing.

To ease using BDI plug in, a method is introduced to systematically model the BDI agent at a higher level of abstraction and transform it into BDI coding through proposed guideline.

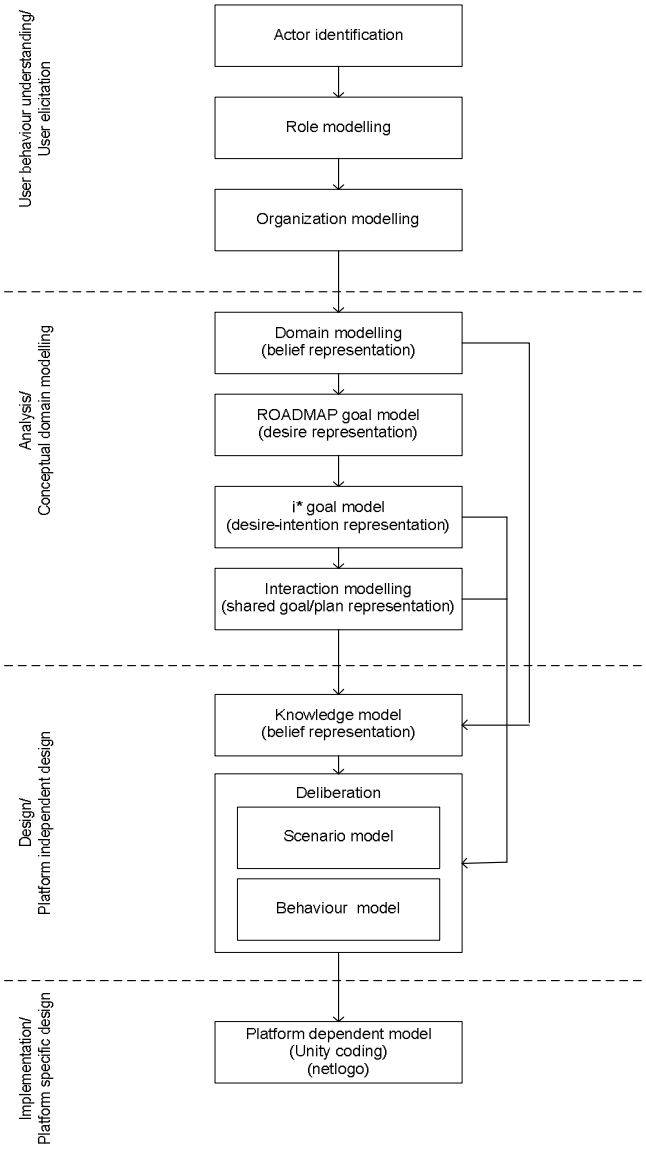

Fig. 1 BDI methodology for human cognition modeling

Fig. 1 shows the modeling of human cognition through the extended AOM. Agent-oriented methodology or modeling is the methodology compliant with model-driven architecture for complex socio-critical system modeling. It has been adopted in games [16], ICT4D [17], mobile application development [18], chemistry simulation [19], smart application [20].

fThe extended AOM covered four phases: user cognitive understanding/ elicitation, analysis/conceptual domain modeling, design/platform independent design, and implementation/platform-specific design. The extended AOM begins with the user cognitive understanding/elicitation phase. This involves identifying actors, the role of actors, and the relationship among actors. After gathering all the information from the first phase, it is then followed by analysis/conceptual domain modeling. It involves understanding the actor's goal, the action, and the plan needed to achieve the goal. It represents the problem domain at an abstract level. After that, the information from the analysis is transformed into design/platform independent design. The design/platform independent design covered the actors' knowledge and the deliberation to achieve the goal. After the design is done, it then transforms the agent models into coding by using the transformation guideline we provided.

In supporting each phase, agent models have been adopted. Table II shows the summarize of the agent models regarding human cognition modeling in each phase. The details of the modeling are presented in [21].

TABLE II
HUMAN COGNITIVE MODELLING THROUGH EXTENDED AOM

Phase 1: User Cognition Understanding/Elicitation	**Human cognition processes**	**Agent models**
Actor identification (Whose cognition we need to capture?)	Understand problem-solving behaviour of individual regarding the problem at hand. In other words,	Organizatio n model Role model
Phase 2: Analysis/Conceptual Domain Modelling (What is the human belief, desire/goal, shared desire/shared goal? How to achieve the human desire analytically?)	Analyze human cognition - belief representation - desire representation - desire intention representation - shared goal representation/s ocial goal	Domain model ROADMAP goal modelling i* goal modelling - i* goal modelling -interaction model
Phase 3: Design/Platform independent design (How to achieve human desire at design level?)	Human cognition design -belief representation -deliberation	Knowledge model Scenario model Behaviour model
Phase 4: Simulation/Platform specific design (How to achieve human desire at programming level?)	Transforming the human cognition model into Netlogo simulation model and Unity3D simulation model	Netlogo human cognition construct Unity3D human cognition construct

Table III shows the mapping of the scenario model and behavior model into Unity3D construct. This includes agent belief and agent attribute. The agent belief is mapped into Unity3D function ModifyState(string key, int value) and the agent attribute is mapped into Unity3D function Start().

TABLE III
MAPPING OF KNOWLEDGE MODEL INTO UNITY3D CONSTRUCT

Knowledge Model		
Model contexts	**Unity3D construct**	**Example of Unity3D syntax**

Agent belief	public void ModifyState(string key, int value)
	{
	if (HasState(key))
	{
	states[key] += value;
	if (states[key] <= 0)
	{
	RemoveState(key);
	}
	} else
	{
	AddState(key, value);
	}
	}
		beliefs.ModifyState ("<belief>", 1);

Agent attribute	void Start() {}	void Start() {
		//set attribute here
		health = 100
		}

Table IV shows the mapping of scenario model and behaviour model into Unity3D construct. This includes agent desire and intention type, agent interaction activity and rules and condition (deliberation). The agent desire and intention type are mapped into Unity3D procedure. The agent interaction activity is mapped into Unity3D function OnTriggerEnter(Collider other). The rule and condition (deliberation) are mapped into "if" or "ifelse" control flow and logic operator.

TABLE IV
MAPPING OF BEHAVIOUR MODEL INTO UNITY3D CONSTRUCT

Behaviour and Scenario Model		
Model contexts	**Unity3D construct**	**Example of Unity3D syntax**

Agent desire and intention type	Procedure	void perceive_fires()
		{
		//do something
		}

Agent interaction activity	void OnTriggerEnter(Collider other)	private void OnTriggerEnter(Collider other)
		{
		//do something
		}

Rule and condition (deliberation)	The "if" or "ifelse" control flow and logic operator	if condition {
		//do something
		}

III. RESULTS AND DISCUSSION

This section elaborates the BDI tool's usage through a case study of fire evacuation during a bushfire. We replicated the case study from Adam and Gaudou [22] to showcase the proposed methodology and tool's feasibility. The scenario is described as following: There are 100 houses in a residential area surrounding the forest. Each house contains of 1 occupant. There are two shelters located at the upper right and bottom left. The occupants are doing their daily routine. Suddenly, there is a bushfire and started to spread over the residential area. The occupants will start to react by defending their home or evacuate to a safe place.

Fig. 2 shows the interface of fire evacuation simulation in Unity. The script "Fire Spawn" is the simulation settings such as general settings, fire settings, building settings and agent settings. Figure 3 shows the parameters we used in our simulation. To replicate the simulation model in [22], we adopted the same parameters, formulas, and calculation.

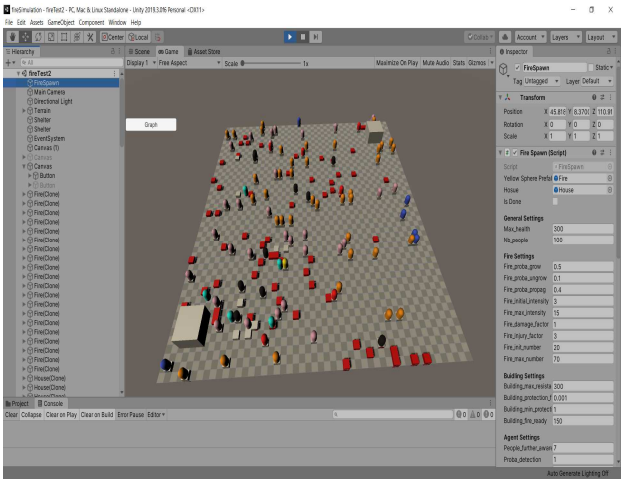

Fig. 2 The interface of the fire evacuation simulation in Unity

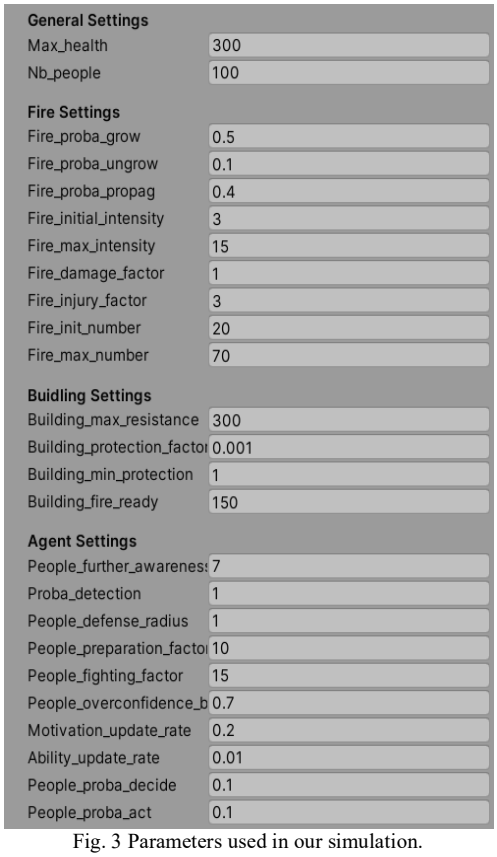

Fig. 3 Parameters used in our simulation.

The purpose of this model is to simulate real human behavior during bushfire. The goal is to raise awareness of the population's real behaviors in crises based on the distinction between objective (capabilities, danger) and subjective (confident, risk aversion) attributes and on individual motivations.

As agents, we defined the occupants randomly distributed in a residential area surrounding by a forest containing 2 shelters. Attributes are randomly initialized (health and resistance between 100 and 200; capabilities and motivations between 0 and 1). There are 3 categories of parameters concerning: fires (probability to grow or propagate, initial intensity, damage factor, etc); buildings (resistance, protection factor, etc); and occupants (confidence bias, perception and action radius, etc).

Six types of human behaviors defined by different colours: dark blue (unaware), pink (indecision), orange (preparing to defend), red (defending), yellow (preparing to escape), light blue (escaping).

Figure 4 shows the perception of the occupant. If there is a fire within the agents' vision, the occupants will update their belief 'hasFire' to true. If the audiences have the belief 'hasFire' = true, then the occupants will start to achieve their goal and plan their actions based on the motivations.

```
void perceive_fires()
    {
        Random.InitState(System.DateTime.Now.Millisecond);
        for (int i = 0; i < fov.fires.Count; i++)
        {
            if (Random.value < (fireSpawn.proba_detection * objective_ability))
            {
                if (!known_fires.Contains(fov.fires[i]))
                {
                    known_fires.Add(fov.fires[i]);
                }
            }

            if (fov.fires[i].transform.parent.GetComponent<FirePropagation>().intensity <= 0)
            {
                known_fires.Remove(fov.fires[i]);
            }
        }

        if (known_fires.Count > 0 && !isPerceived)
        {
            beliefs.ModifyState("hasFire", 1);
            GetComponent<MeshRenderer>().material.SetColor("_Color", new Color32(230, 160, 200, 255));
            fireSpawn.indecisiveNumber++;
            fireSpawn.unawareNumber--;
            isPerceived = true;
        }
    }
```

Fig. 4 Unity3D construct that shows part of BDI decision in fire evacuation

```
if (GetComponent<Agent>().number_of_heatCell == 0 )
{
    GetComponent<Agent>().state = State.STATE_PREP_DEFEND;
    fireSpawn.defendNumber--;
    fireSpawn.prepareDefendNumber++;
    beliefs.RemoveState("fireIsNear");
    beliefs.ModifyState("PreparingDefend", 1);
    GetComponent<MeshRenderer>().material.SetColor("_Color", new Color32(255, 137, 0, 255));

    return true;
   }

if (GetComponent<Agent>().escape_motivation > GetComponent<Agent>().defend_motivation)
{
    GetComponent<Agent>().state = State.STATE_ESCAPE;
    beliefs.ModifyState("isDanger", 1);
    beliefs.RemoveState("fireIsNear");
    fireSpawn.defendNumber--;
    fireSpawn.escapeNumber++;
    GetComponent<MeshRenderer>().material.SetColor("_Color", new Color32(0, 255, 255, 255));
    return true;
}
```

Fig. 5 Unity3D construct that shows deliberation

Fig. 5 shows how deliberation works. If the fire is near to the occupant during achieving the goal "Defend Home", the occupant needs to reconsider the goal and replicate the actions.

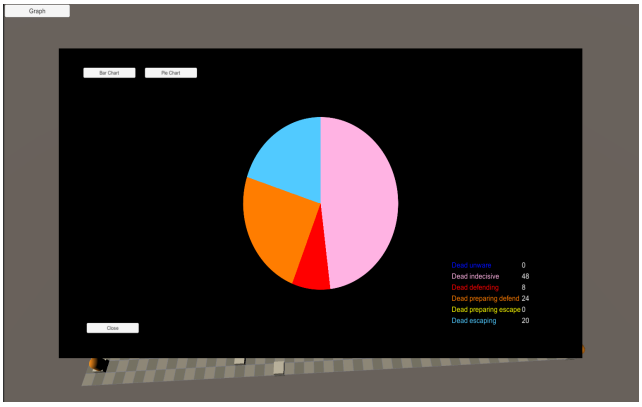

Fig. 6 Causes of death in our simulation

Upon the runtime, the output of the simulation is shown in Fig. 6. With the parameters we used in Fig. 3, we obtained the following distribution of causes of death: 47% of the occupants died while still passive (indecisive); 21% died while escaping; 5% died while defending; the others died while preparing to escape (10%) or preparing to defend (15%), taking by surprised before they could react. From Fig. 6, we found that most of the death is caused by indecisive. The occupant is aware of some fires but unable to decide about how to react. This means that the occupants do not have awareness and knowledge on firebush.

In this case study, we demonstrate how to systematically model a more complex fire evacuation simulation and transform the models into Unity. The methodology can reflect the modeling of human cognition and transform the higher-level model into a simulation model in a systematic manner.

IV. Conclusion

Agent-based social simulation is important to understand real worlds and complex problems. A believable agent in simulation can derive from simple rules based on complex and realistic human cognition like BDI. Although many BDI frameworks of severe games and the integration mechanism of BDI platform into Unity3D have been introduced, exploring adopting agent method into games engines like Unity3D has yet to be explored. In this paper, a BDI plug in is introduced for agent-based social simulation. The BDI plug in is complemented with BDI methodology to bridge the modeling complexity of BDI through a higher-level abstraction and model transformation. In future, more case studies are needed to investigate the usage of the proposed BDI methodology in various domain. In addition, the extensible of the BDI tool from higher level models to low level BDI implementation is worth to explore. We adopt a model driven architecture in this research direction. This is different from middleware introduced in the works [6], [23] With middleware, it is a framework that supports the integration of BDI platform across different agent simulation platforms. On the other hand, there is a need to study the performance cost of the BDI plug in into thousands of agents during simulation to understand the efficiency of the BDI plug-in. The replication of the work [6] for performance study is worth exploring.

Acknowledgment

This project's funding is made possible through the research grant obtained from UNIMAS under the Special MyRA 2018 Cycle [Grant No: F08/SpMYRA/1659/2018].

References

[1] M. Naili, M. Bourahla, and M. Naili, "Stability-based model for evacuation system using agent-based social simulation and Monte Carlo method," *International Journal of Simulation and Process Modelling*, vol. 14, no. 1, pp. 1–16, 2019, doi: 10.1504/IJSPM.2019.097702.

[2] C. Adam and B. Gaudou, "BDI agents in social simulations: A survey," *Knowledge Engineering Review*, vol. 31, no. 3, pp. 207–238, Jun. 2016, doi: 10.1017/S0269888916000096.

[3] D. Singh and L. Padgham, "OpenSim: A framework for integrating agent-based models and simulation components," in *Frontiers in Artificial Intelligence and Applications*, 2014, vol. 263, pp. 837–842, doi: 10.3233/978-1-61499-419-0-837.

[4] P. Taillandier, M. Bourgais, P. Caillou, C. Adam, and B. Gaudou, "A BDI agent architecture for the GAMA modeling and simulation platform," in International Workshop on Multi-Agent Systems and Agent-Based Simulation, 2016, pp. 3–23.

[5] A. Shendarkar, K. Vasudevan, S. Lee, and Y.-J. Son, "Crowd Simulation for Emergency Response Using BDI Agent Based on Virtual Reality," in *Proceedings of the 38th Conference on Winter Simulation*, 2006, pp. 545–553.

[6] L. Braubach, A. Pokahr, and W. Lamersdorf, "Jadex: A BDI-Agent System Combining Middleware and Reasoning." [Online]. Available: http://www.agentcities.net.

[7] A. Antakli, I. Zinnikus, and M. Klusch, "ASP-driven BDI-planning agents in virtual 3D environments," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2016, vol. 9872 LNAI, pp. 198–214, doi: 10.1007/978-3-319-45889-2_15.

[8] J. Nilsson and F. Klügl, "Human-in-the-Loop Simulation of a Virtual Classroom," in *Multi-Agent Systems and Agreement Technologies*, Springer, 2015, pp. 379–394.

[9] A. Barriuso, F. de la Prieta, and T. Li, "An Agent-Based social simulation platform with 3D representation for labor integration of disabled people," in *Advances in Intelligent Systems and Computing*, 2015, vol. 372, pp. 55–64, doi: 10.1007/978-3-319-19629-9_7.

[10] Y. Dollie, W. Grant, and J. Hooper, "Autonomous self-learning agents in 3D virtual worlds."

[11] K. v. Hindriks *et al.*, "Unreal goal bots: Conceptual design of a reusable interface," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2011, vol. 6525 LNAI, pp. 1–18, doi: 10.1007/978-3-642-18181-8_1.

[12] D. Singh, L. Padgham, and B. Logan, "Integrating BDI Agents with Agent-Based Simulation Platforms," *Autonomous Agents and Multi-Agent Systems*, vol. 30, no. 6, pp. 1050–1071, Nov. 2016, doi: 10.1007/s10458-016-9332-x.

[13] L. Ni, V. Gonzalez, J. Liu, A. Rahouti, L. Zhang, and B. P. Taing, "An agent-based approach to simulate post-earthquake indoor crowd evacuation," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Oct. 2018, vol. 11224 LNAI, pp. 568–575, doi: 10.1007/978-3-030-03098-8_43.

[14] P. Sudkhot and C. Sombattheera, "A Crowd Simulation in Large Space Urban," Dec. 2018, doi: 10.23919/INCIT.2018.8584878.

[15] N. Poli, "Game Engines and MAS: BDI & Artifacts in Unity," 2018. [Online]. Available: https://amslaurea.unibo.it/15657/.

[16] G. L. C. Wyai, C. WaiShiang, and N. Jali, "Agent-Oriented Methodology for Designing 3D Animated Characters," *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)*, vol. 9, no. 3–3, pp. 153–158, 2017.

[17] C. WaiShiang, N. Jali, M. A. Khairuddin, and H. Sharbini, "Understanding Technology Changes for ICT4D Projects through

Modelling," *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)*, vol. 9, no. 3–3, pp. 147–151, 2017.

[18] W. Cheah, P. ChinHong, A. A. Halim, and others, "Agent-Oriented Requirement Engineering for Mobile Application Development.," *International Journal of Interactive Mobile Technologies*, vol. 11, no. 6, 2017.

[19] C. W. Shiang, S. Nissom, N. Jali, and S. YeeWai, "Adopting Agent Oriented Methodology (AOM) For Modelling and Simulation in Epidemiology and Ecological Studies," *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)*, vol. 9, no. 2–11, pp. 151–158, 2017.

[20] C. Wai Shiang, B. Tien Onn, F. Swee Tee, M. A. bin Khairuddin, and M. Mahunnah, "Developing agent-oriented video surveillance system through agent-oriented methodology (AOM)," *Journal of computing and information technology*, vol. 24, no. 4, pp. 349–368, 2016.

[21] G. L. C. Wyai, S. K. Wai, C. W. Shiang, and M. A. Khairuddin, "Modelling Human Decision in Fire Evacuation Simulation through BDI Based Cognitive Architecture.," *Solid State Technology*, pp. 2766–2799, 2020.

[22] C. Adam and B. Gaudou, "Modelling human behaviours in disasters from interviews: Application to Melbourne bushfires," *JASSS*, vol. 20, no. 3, Jun. 2017, doi: 10.18564/jasss.3395.

[23] J. van Oijen, L. Vanhée, and F. Dignum, "CIGA: A middleware for intelligent agents in virtual environments," in *International Workshop on Agents for Educational Games and Simulations*, 2011, pp. 22–37.