IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 20 August 2023, accepted 4 September 2023, date of publication 11 September 2023,
date of current version 14 September 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3313725

RESEARCH ARTICLE

Procedural Content Generation Using Reinforcement Learning for Disaster Evacuation Training in a Virtual 3D Environment

JIGYASA AGARWAL[1] **AND S. SHRIDEVI**[2]
[1]School of Computer Science and Engineering, Vellore Institute of Technology, Chennai 600127, India
[2]Centre for Advanced Data Science, Vellore Institute of Technology, Chennai 600127, India

Corresponding author: S. Shridevi (shridevi.s@vit.ac.in)

This work was supported by the Vellore Institute of Technology, India.

ABSTRACT This research addresses the need for effective disaster evacuation training methods by proposing a virtual reality system that utilizes Reinforcement Learning Procedural Content Generation (RL-PCG) algorithms. The aim of this study is to provide a cost-effective and safe way to conduct disaster evacuation preparedness training, surpassing the limitations of traditional real-life drills. The paper's objectives encompass the design of a novel 3-layer PCG architecture for generating realistic disaster simulations in virtual reality, the implementation of a working prototype for fire disaster scenarios, and the evaluation of the proposed system's effectiveness through comparison with existing RL agents. Significant findings include the superiority of the RL-PCG agent in generating diverse and realistic disaster scenarios with faster training time and lesser number of steps, even with limited processor capabilities. In conclusion, this research establishes that the RL-PCG Scenario for Disaster Evacuation Training in VR is a more effective method, leading to improved disaster preparedness for individuals, and opens avenues for further advancements in disaster training using virtual reality and reinforcement learning technologies. For a video demo of this work, please visit https://youtu.be/3WZnQOfUP94.

INDEX TERMS Disaster evacuation training, procedural content generation, reinforcement learning, virtual reality.

I. INTRODUCTION

The International Federation of Red Cross and Red Crescent Societies (IFRC)[1] defines disasters as ''serious disruptions to the functioning of a community that exceed its capacity to cope using its own resources''. They can be caused by natural hazards, such as earthquakes, floods, landslides, etc. as well as by man-made and technological hazards like fire. The influence of disasters carries the risk of exposure of the vulnerabilities of a community. Disasters cause huge damage to life and property. The occurrence of a disaster cannot be predicted in advance most of the times. Therefore, the best way to protect ourselves when a calamity arises is to be aware of the possibilities, and prepare ourselves to spring into action whenever a disaster strikes. There are various types of disaster evacuation preparedness training which are conducted by schools, offices, etc. to prepare individuals for disasters. One common method of training is by conducting a drill wherein the conditions during the time of a disaster are replicated, and the people are supposed to evacuate out of that scene based on certain guidelines and procedures. Although effective, these real-life disaster evacuation drills are really expensive and can sometimes be dangerous and life threatening as well [1]. Moreover, it does not exactly replicate the disaster scenario, like the production of fire and smoke, which hinder the visibility during evacuation. Such fire effects cannot be simulated in real-life training drills due to safety reasons. By taking advantage of the rapidly developing technologies, we can simulate each and every detail of such disasters and produce realistic scenarios virtually. Virtual disaster training provides a cost-effective and safe way to conduct disaster evacuation preparedness training. There are many works in literature that use virtual

The associate editor coordinating the review of this manuscript and approving it for publication was Xiaogang Jin.

[1]https://www.ifrc.org/

reality technology for disaster preparedness training [1], [2], [3], [4], [5], [6]. With AR/VR and Metaverse being the latest, most popular and rapidly accelerating field today, it is a great time to switch to virtual methods of training with the fast technological advancement of the HMDs (Head Mounted Displays) by various companies, which would enable people to prepare for disasters in a much more cost-effective and safer manner.

But classical virtual training simulations only prepare the user for a specific type of scenario, e.g. in the event of an earthquake, the training simulation only shows the ceiling debris falling from a particular place. Or in case of a fire simulation, only a specific object is shown to catch fire in every iteration of the training. In real life when a disaster occurs, it can occur at any point of time and can originate from any place. Manually creating so many variations in virtual training is a time consuming and monotonous task for the designers as well as for the developers. In such a situation, if we were to programmatically generate these disaster scenarios, and use intelligent agents to do so, then that would decrease the work to a great extent, and we can focus more on programming the behavior of the NPCs (Non-Player Characters) and other agents in the scenario, rather than in designing them. At the same time, this will make the training simulations more efficient and effective. The practice of programmatically generating content is called Procedural Content Generation (PCG) and it is extensively used in game development to create dynamic content for games. Moreover, with the recent advancement of Artificial Intelligence and Machine Learning algorithms, we can add some level of intelligence to the way that the game content is created. In this work, we have proposed a disaster evacuation training system in VR which uses Reinforcement Learning Procedural Content Generation (RL-PCG) algorithms to generate dynamic scenarios for each iteration of the training for the user, to prepare them to face any kind of disaster situations that may occur.

This research aims to bridge the gap between traditional drills and advanced technological possibilities, ensuring a cost-effective, safe, and efficient method for disaster evacuation training. Through this work, we try to answer the research question: Would an RL-PCG Scenario for Disaster Evacuation Training in VR be a more effective method of training, and would it lead to better disaster preparedness for individuals? The main contributions of this work are: 1) A design architecture of the RL-PCG disaster evacuation training system, 2) A novel 3-layer PCG architecture for creating realistic disaster simulations in VR, 3) A working prototype of the evacuation training simulation, and 4) A study showing enhanced results of our RL-PCG agent as compared to other RL agents in literature. The rest of the paper is structured as follows: Section II highlights the literature survey of our research, along with the feature comparison table of selected works from literature, which are most relevant to our work. In Section III we have elaborated the proposed system and methodology followed by us to develop our working prototype and RL-PCG algorithm. Section IV elaborates the features and game flow of our prototype, along with the alternate game flows. In Section V we discuss our results and compare it with the results of the selected related works from literature, and finally, Section VI concludes the paper and mentions the future direction of our work.

II. RELATED WORKS

This section reviews the works in literature for the underlying concepts and technologies of virtual reality, PCG and machine learning for training and other purposes. At the end of this section, we have elaborated our research findings along with the research gaps in literature identified by us.

A. VIRTUAL REALITY (VR)

XR is defined as ''a group of emerging technologies such as virtual reality (VR), augmented reality (AR), and virtual worlds (VWs) that involve the use of 3D models/simulations across physical, virtual, and immersive platforms'' [7]. Our focus for this work is going to be on virtual reality, in which the user is completely immersed inside the virtual word that they are viewing. Virtual reality has been steadily gaining popularity in recent times, for a variety of purposes. It holds special importance for training purposes, as VR allows for a new cost-effective and safe manner to train individuals for various situations in a non-consequential manner, but at the same time providing a realistic experience. Many works in literature have their focus on building a VR simulation system for conducting emergency disaster evacuation training. Reference [8] is one such systematic review paper that seeks to highlight the trends, technologies, research areas and challenges in XR technology for Disaster Management. The authors have categorized the works into five categories of computer simulation modelling, interaction techniques, training, infrastructure assessment and reconnaissance, and public awareness. The section of this review on Training, specifically Evacuation Training, is the category where our proposed work falls under. In [6], the authors have developed a disaster management training simulation covering disasters like earthquake, typhoon, tsunami and fire, which help the users train and prepare for emergencies when a disaster strikes. In [5], the authors have proposed a VR disaster training system specific to accidents caused by fire. The system has 3 modes to train the users for different kinds of skill sets: evacuations drills, firefighting, and a comprehensive training which is a hybrid of the former two modes. They have used multi-agent systems for their simulation scenarios. A cloud-based collaborative VR evacuation drill has been proposed in [1]. The main aim of their work is to study human behavior by simulating crowd behavior using computer-controlled and user-controlled agents. Another work [3] focusing mainly on flood disasters, shows how we can use Kolb's Learning Method to train individuals for evacuation and rescue during disasters, by ensuring learning motivation among users. The need for simulating realistic disasters has also been emphasized upon in [9] where the authors have used Unity

3D to develop realistic disaster simulations using markerless AR technology.

But all these works depend on manually designing the scenarios and the game assets, which remain the same in all iterations of the simulation. There is no intelligent agent present to dynamically change the scenarios every time the user plays it. There is an AI component present in [4], where the system tries to recreate the scenario of disasters. The system is doing so with the dataset of earthquakes that occurred, inputting that information into a CART algorithm to predict what the scenario would have looked like. This approach heavily depends on the collection and availability of accurate and relevant datasets, which may not always be possible. For this reason, many AR/VR and game developers use Reinforcement Learning (RL), which does not require information to be collected in advance. Further details of RL and its applications in VR simulations are discussed in Section II.C.

B. PROCEDURAL CONTENT GENERATION (PCG)

Procedural Content Generation is the generation of content using algorithms. If procedural is the how then content is the what [10]. Any assets which are used as content for games can be generated procedurally. This includes levels, character models, and even the textures, music, sounds, the story line used in the game. By using PCG, we are letting the computer take some responsibility of the design process, allowing the developers to focus on working on the core functionality of the game. Many algorithms have been proposed for implementing PCG, and a lot of these algorithms are specific to the type of game content that needs to be generated. In [11], the authors have classified the types of game content that can be generated procedurally into categories like game bits, game space, game scenarios, etc. and have listed out the various algorithms that exist to generate each of these game contents. One of the most common algorithms is the Pseudo Random Number generation or PRN which uses seed values to randomly decide on the position of objects in the game, the choice of weapons, etc. like in the work presented in [12] where PRN is used to dynamically generate game levels and difficulty for a 2D game developed using the Unity Game Engine. Other popular algorithms are generative grammars, image filtering, L-Systems, spatial algorithms, etc. A survey of PCG algorithms used in virtual reality environments have been presented in [13]. Most of these algorithms overlap with the classical PCG algorithms mentioned earlier, with specific focus on the balanced compromise between the feeling of presence and realism in VR, and the performance of the PCG algorithms. They have also highlighted the fact that the awareness of using PCG for virtual worlds has increased significantly in recent years, not only through researchers, but mainly through practitioners. One such work, [14] has made use of the surrounding of the user's physical space to create the game path and obstacles in mixed reality.

The use of PCG as a means to retain the interest of users and encourage exploration in games and virtual environments is quite prevalent, and stands as one of the most popular use cases for these algorithms. Such a use case has been explored in [15], wherein the amount of time a user spends interacting with each of the game assets is used as a parameter to generate an open world game for them. Similarly, in [16] the authors have used PCG techniques based on decision trees to make a VR game which offers a unique game experience to each individual based on the time they spend interacting with the various game assets. For example, players interacting more with trees and greenery are given more forest-like scenarios and users interacting more with battle weapons are given a battle scenario. This work has an AI component to it, which allowed the system to take intelligent choices for the scenario generation. Similarly, many researchers have started proposing PCG algorithms with a machine Learning component to generate functional environments that are unique, but at the same time not too random, making the scenarios look logically possible and compatible with the physical laws of the real world. A PCGML framework has been proposed in [17] for automated scene layout generation using neutrosophic evaluation. The game scenes are initially generated randomly after which they are sculpted using a genetic algorithm. This method creates unique and non-repetitive game levels even with several runs of the same algorithm. In [18], the authors present a survey of the various PCG algorithms in literature which make use of ML algorithms like neural networks, LSTM networks, autoencoders, deep convolutional networks, and many others. They have also considered five broad categories of training algorithms: Back Propagation, Evolution, Frequency Counting, Expectation Maximization, and Matrix Factorization. The vast majority of work covered in this paper deals with 2D PCG. Moreover, large datasets are required for these ML approaches. This is where Reinforcement Learning (RL) comes into play, which does not require any complex methods for the acquisition of datasets beforehand. For VR systems, the data is collected through rendering during training. We will discuss the works using RL for PCG in the following section.

C. REINFORCEMENT LEARNING (RL)

Reinforcement learning is the problem faced by an agent that must learn behavior through trial-and-error interactions with a dynamic environment [19]. In a reinforcement learning problem, an agent generates its own training data by interacting with the environment based on its current policy [20]. For this reason, RL is preferred over other classical supervised ML algorithms for AR/VR and game development. A Reinforcement Learning problem can be framed as a typical Markov Decision Problem (MDP) with a set of agent states, S; a set of agent actions A; the probability of a transition at time t from state s to s' under an action α, $\Pr(s_{t+1} = s' | s_t = s, \alpha_t = \alpha)$ and the immediate reward after the transition from s to s' with action α, $R_\alpha(s, s')$. The aim of the RL agent is to choose an action policy $\pi : A \times S \rightarrow [0, 1], \pi(\alpha, s) = Pr(\alpha_t = a | s_t = s)$ that

would maximize its expected cumulative reward. However, in complex simulation environments, the rewards are not direct and immediate, and the action states are continuous instead of being discrete, therefore MDP fails to provide a good result here, and so Deep Reinforcement Learning needs to be used [21]. In Deep Reinforcement Learning, the observations that the agent collects are used as input for the neural network. This neural network then chooses an action based on the current policy. At the end of each episode, the agent receives rewards based on the actions it performs, and the information about the rewards received by it is fed as feedback to the neural network, which then alters its decisions based on the positive or negative rewards received. In works like [22], the authors have used RL to change the behaviour and response of persons playing a game without their prior knowledge, encouraging the players to develop a strategy while playing the game. Authors of [23] and [24] have proposed a deep reinforcement learning approach to model their PCG algorithm to generate unique immersive VR environments for an Industrial Engineering course in order to maintain the students' interest in the coursework. The reward function of their RL PCG algorithm is such that on the basis of the position of a subset of objects in the environment, the RL agent is able to decide the position and orientation of the other objects in the environment. The results of their work are quite relevant for us, and we will be using the results of their work to compare our own work. The same authors have also experimented with this same approach in [25] to generate multi-contextual environments to teach the same underlying academic topic, in this case the mathematical concept of probability. On the basis of the user input for some parameters, the RL agent generates a scenario for the user in VR. The user can tweak the parameters like the probability distribution of certain objects and observe the changes it causes in the environment that is procedurally generated by the RL agent. In the above three works, the authors have used the Proximal Policy Optimization (PPO) algorithm [26], [27] which is implemented using the ML Agents Toolkit of the Unity Game Engine. This toolkit provides another implementation of an RL algorithm called Soft Actor Critic (SAC), which is an off-policy method in which agents learn from stored data of previous episodes to make decisions [20]. The authors of [28] have used the SAC algorithm to procedurally generate dungeons in their game. They have managed to decrease the training time significantly by training the RL agent only using the data structures, removing the need to render the scenes while training the RL agent. The dungeon is generated procedurally with the use of multi-agents, each specialized to render a particular room of the dungeon, which helped them to parallelize the training, and narrow down the source of errors that arose.

D. RESEARCH GAP

To the best of our knowledge and research, there are no PCG environments proposed which use reinforcement learning for generating unique disaster evacuation training scenarios. Although the need for dynamic fire scenarios in simulation is highlighted in [2], the virtual simulations they have created only include 3 pre-planned virtual scenarios for the disaster evacuation training. The authors of [29] have used HTN planning to generate different emergency scenarios with buildings that are collapsed at different places during a disaster, and the parameters can be adjusted according to the desired goals that the user wants to train for, but there is no ML or RL component to aid the scenario generation process. No other work has considered the fact that disaster evacuation training should have unique scenarios in each iteration of the training, and none of them have proposed machine learning aided methods to do so. Table 1 sums up the main works from Section II which we will be using to compare our work with.

III. PROPOSED SYSTEM AND METHODOLOGY

In this work, our aim is to design a Disaster Evacuation Training application using Procedural Content Generation and Reinforcement Learning in a Virtual 3D environment, which allows the user to train for diverse situations and prepare them for a natural disaster, and evaluate this system to test whether it helps to increase the efficiency and effectiveness of training as compared to traditional disaster evacuation training simulations, by comparing our algorithms and reward functions with the ones in related works in the literature.

We propose a virtual reality training simulation game which can simulate fire disasters. The VR environment is an apartment with various floors, and various rooms on all these floors. When the user starts the training simulation, they will be spawned in one of the rooms on one of the floors of this apartment. Spawning means the origination or appearance of the player at a particular point in the game level. The furniture of this room will be procedurally generated using our reinforcement learning procedural level generation algorithm, which will be explained in detail later in this paper. Our aim is to dynamically generate and present scenarios to the user, each time they start the training. To achieve this, we want to create a virtual environment that is unique, and logically correct at the same time. For achieving this, the primary idea is to train the RL (Reinforcement Learning) agents in such a way that they place themselves relative to the position of some other objects in the room.

A. DESIGN ARCHITECTURE

The system design architecture of our proposed system can be seen in Fig. 1. This architecture consists of 3 high-level layers:

1) Unity ML Agents Toolkit: The ML Agents Toolkit[2] of the Unity Game Engine provides implementation of several Deep Reinforcement Learning algorithms which can be used to implement intelligent multi-agent behaviour in virtual environments. The most important step in this is to tweak the hyperparameter values to

[2]https://unity.com/products/machine-learning-agents

TABLE 1. Summary of related works in literature.

Paper	Purpose	PCG Algorithm	ML/RL Algorithm	No. of Agents	Results	Pros	Cons
[29]	Training	HTN Planning	NA	NA	Small, medium and large rooms with 5 high-level goals	Unique scenario generation	Limited scope due to lack of ML/RL component
[23] [24]	Learning	Based on position and behaviour of a subset of game assets	RL (PPO)	3	Trained agent takes less than 1 second to generate new scenario	Unique scenario generation by RL agent	Does not show result with other RL algorithms
[25]	Learning	Based on user input for some parameters	RL (PPO)	3	Mean reward of trained agent, Grocery scene: 2.19, Mfg. scene: 7.13	Multi-contextual scenario generation teaching same underlying concept	Does not show result with other RL algorithms
[28]	Gaming	Based on the relative position of the event rooms	RL (SAC)	4	Mean reward of trained agent, Treasure/shop room: 1.001, Boss Room: 1.772	Reduced training time, less computational resources required, unique scenario generation	RL Agent needs to be re-trained for generating larger dimensions of the maze

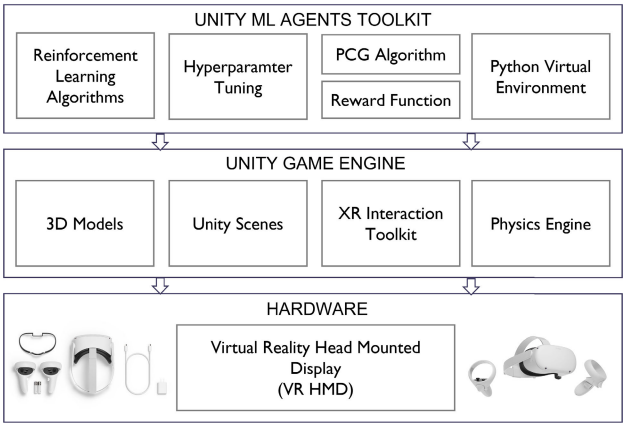


FIGURE 1. System design architecture diagram for the proposed system of the RL-PCG training simulation in VR.

get an optimal result. There are several RL algorithms which can be implemented using this toolkit, depending upon the use case of the virtual environment [30]. For this work, we are using the PPO (Proximal Policy Optimization) algorithm to train the RL agents. PPO is an on-policy gradient method, meaning that unlike popular Q-learning methods such as Deep Q-Networks that learn from stored offline data, PPO learns directly from the experiences it gathers in its environment. This makes it ideal for use in virtual environments where it is very difficult to obtain data in advance. This is also the layer where we will be executing our RL-PCG algorithm for the reward function to train the agents using a python virtual environment along with Unity.

2) Unity Game Engine: The Unity game engine provides us with the platform and tools to make our virtual environment realistic, by making use of 3D models, scenes, interactions and physics mechanisms. We used 3D models of the various objects in the environment like the furniture, and the evacuation equipment like fire extinguisher, etc. We used the physics engine of Unity to simulate realistic disaster scenarios by simulating fire and smoke effects and spreading it around the room utilizing the physics mechanism of the game engine. We can see how this fire effect looks like in the virtual environment in Figure 2. The physics engine was also used to set the velocity and force of the multiple RL agents in the scene.

FIGURE 2. Living room virtual environment with fire and smoke effect.

3) Hardware: The entire evacuation training simulation is deployed on the Meta Quest 2[3] headset for testing. Meta Quest 2 is a mixed reality headset with 6 DoF and 2 controllers, and features like hand gestures and passthrough. Input can be taken through hand gestures as well as through the various buttons of the two controllers.

B. 3-LAYER PCG ARCHITECTURE

In order to achieve a realistic simulation of a disaster in a virtual environment, we implemented PCG algorithms in this work in many layers, with each higher layer dependent on the lower ones. A pictorial representation of this approach can be seen in Figure 3. The proposed architecture for creating such realistic disaster simulations can be visualized as 3 high-level layers:

1) PRN (Pseudo Random Number) Generator: This is the bottom layer of the architecture, which holds the implementation for the most common PCG algorithm, called PRN. This algorithm is used to decide the initial spawn position of the user in the virtual environment.

[3]https://www.meta.com/quest/products/quest-2/

For implementing this, the List data structure of C# is used. The list consists of several Vector3 variables, and each Vector3 variable holds the x, y and z coordinates of a particular spawn position in the virtual environment. The PRN algorithm chooses one of these positions at random, and that is where the user is spawned at in the beginning of the training simulation. The PRN algorithm is also used to place the Fire Source object randomly at some point in the room. There are infinite number of possible positions where the Fire Source might be placed at by the algorithm in any given virtual scenario. On the basis of the position of the Fire Source, the position of the RL agents will be decided.

2) RL-PCG: This layer holds the implementation of our Reinforcement Learning PCG algorithm. We have used the PPO algorithm of the Unity ML Agents toolkit to train the furniture objects, which are the RL-PCG agents in this scenario. Each agent uses virtual sensors to collect information from its surroundings during training, and on the basis of its action policy, it performs actions, receives rewards for those actions in accordance to the reward function, and alters its decisions accordingly in the subsequent episodes. The information obtained from the PRN layer is used as input for deciding the actions of the agents in this layer. The furniture agents place themselves relative to the position of the other objects present in the room, and also relative to each other's positions.
3) Fire Algorithm: After the RL agents have placed themselves in the virtual scenario, we have to generate fire and smoke effects in a logical way in the environment. For this, each of the object in the virtual environment is assigned with properties or tags like flammable, non-flammable, etc. On the basis of the properties assigned to the objects, and the relative position of these objects in the environment, the fire is spread in the scene. For example, if there is flammable object, and if it has been placed close to the fire source, then that object will be the fastest to catch fire. Additionally, the fire will start with a very low intensity, and till the time the user does not extinguish it, the intensity of the fire and the amount of smoke will keep increasing gradually over time. Moreover, if the user starts extinguishing the fire, and then they stop extinguishing it before the fire is completely extinguished, the fire will start regenerating itself after a certain delay of time. The amount of fire that is extinguished per second can also be set in the algorithm.

C. DATASET

Since Reinforcement Learning does not require any dataset to be collected in advance, the RL agent will collect its data from its environment during simulation and training itself. We are collecting the following data points from the virtual sensors of the Unity scene:

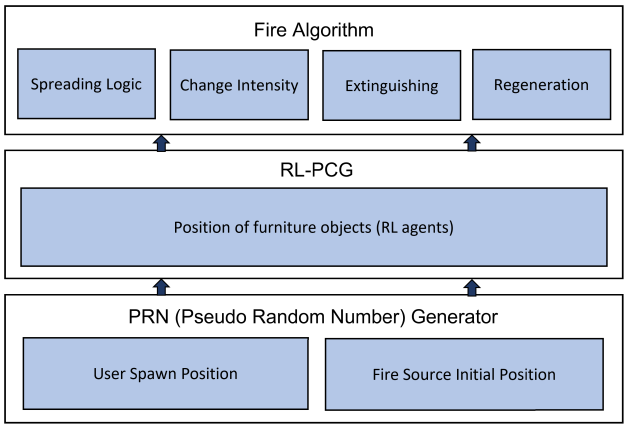


FIGURE 3. 3-Layer PCG architecture for creating realistic disaster simulations in VR.

- The x, y, z coordinates of the position of the fire source object
- The x, y, z coordinates of the position of each of the RL agents (furniture objects)
- The velocity of each RL agent (furniture objects) along its x and z axes

D. CREATING SIMULATION FOR FIRE DISASTER EVACUATION TRAINING IN VR

For creating an evacuation training simulation for disaster preparedness, many strategies have been discussed in various works. In [31], the authors have elaborated a two-step process for creating such environments, with emphasis on the need for presence and effective intensity. In [32], the authors have discussed how naturalistic decision making can be used to form the basis of a learning model which can be used for emergency management training. They have proposed the Recognition Primed Decision making model within NDM to design the training modules. Such a game-based training scenario provides an immersive effect on the learner and aids dynamic decision making in the VR simulation. In [33], the authors have developed a computer-based fire safety virtual evacuation model and tested it with users, to test for symptoms like cyber-sickness, etc. They have also recorded the EEG signals while using the training simulation as a physiological measure. They have divided the development of the virtual environment into various steps, like the development of training content, the assessment tools, training module development and the virtual training module development.

The fire disaster evacuation training simulation in this work has been designed as follows. The RL Agents will be the furniture objects of the room that the user is spawned in. Out of these furniture agents, few of them will be flammable, and the others will be non-flammable objects. Apart from these RL agents, there will be an object in the room which will act as the fire source, i.e., the object from where the fire will originate. In the living room virtual environment, we have a lamp game object, which is acting as the fire source. The

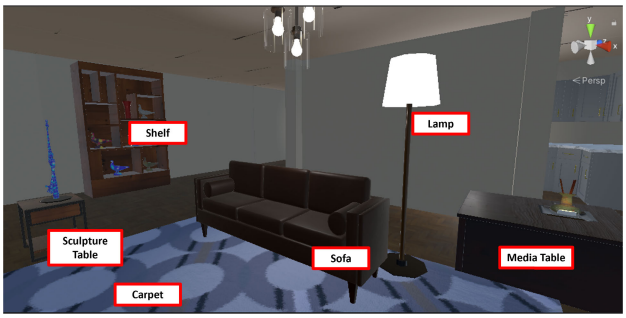


FIGURE 4. Setup of the living room virtual environment with the fire source object and the other furniture RL agent objects for simulating the fire disaster.

position of this lamp in the room will be generated randomly in the beginning of every episode using the PRN (Pseudo Random Number) Generator algorithm of PCG as mentioned in the Layer 1 of the architecture in Figure 3. Once the lamp is placed, the other RL objects of the scene get activated one-by-one, and get placed in the scene. The position of each of these agents will be relative to the other objects in the scenario. For the living room scenario, we have 5 RL agents: sofa, media table, sculpture table, carpet and a shelf. The setup of the living room environment can be seen in Figure 4. Out of these, the sofa, media table and carpet are the flammable objects. All these objects have a Rigidbody component attached with them, which allows us to give force and velocity to the objects, so the objects can move around the room and place themselves in the virtual environment. Our aim is to create a training environment in which a fire disaster is bound to occur. Therefore, the RL agents will place themselves in the room environment in such a way that the flammable objects are placed at an optimal specified distance from the fire source, thus ensuring that if the source object catches fire, it will spread around in the room and the user will need to evacuate out of the situation. Similarly, for the shelf agent, the algorithm will place it close to the walls, as shelves are placed such in a typical room. Whereas for the sculpture table agent, it is placed in such a way that it gets a positive reward for placing itself at the intersection position of two walls, i.e., at the corners of the room. The dynamic positions of the objects in 4 different iterations of training can be seen in Figure 5. We can see that dynamic environments are getting generated every time the application starts (Figure 5). By placing each object (agent) of the scenario one-by-one, each subsequent object (agent) is more sensitive about its surroundings and has to take into account the positions of a greater number of objects before placing itself in the environment. The reward function of the flammable object RL agents can be represented mathematically as follows:

$$R_f = \alpha(d_{ft}) - \beta(d_{ff}) - \gamma(o_f) - \delta(w_f) - \sigma(o_t) \quad (1)$$

where,

- d_{ft} = a binary variable indicating whether the agent is placed at a specified value of optimal distance from the fire source
- d_{ff} = a binary variable indicating whether the agent is placed too close to the fire source
- o_f = a binary variable indicating whether the agent collides with the fire source object
- w_f = a binary variable which indicates whether the agent collides with any of the walls of the room
- o_t = a binary variable which indicates whether the agent collides with any other objects in the room, and
- $\alpha, \beta, \gamma, \delta, \sigma$ are the reward values for each of these variables

IV. PROTOTYPE: FEATURES AND GAME FLOW

A. WRIST UI

In order to evacuate out of the disaster scenario, the user needs to undergo a certain number of tasks. Only after completing all these tasks, the mission will be successful. We have designed 5 tasks for the user, and these tasks are made visible to the user on a panel which is attached to the virtual counterpart of their left hand controller, to make it easily accessible for them at all times. They can toggle this task panel on and off by using the Menu button on the left controller. A visual representation of this wrist UI along with the user tasks can be seen in Figure 6.

B. USER TASKS

As we can see in Figure 6, the user needs to complete 5 tasks in order to successfully complete the fire evacuation training simulation. These tasks are as follows:

- Task 1 - Locate the Fire Source: The first task for the user is to locate where the fire is coming from. In the start of the mission, the user will be spawned randomly in one of the rooms of the apartment, and they will only be able to hear the sound of the smoke alarm alert in the virtual environment. The user needs to move around the apartment using the hand-held controllers and locate the source of the fire.
- Task 2 - Locate the Fire Extinguisher: The next task for the user is to navigate outside the apartment using the hand controllers and locate the fire extinguisher which is placed outside the main door of the apartment, as can be seen in Figure 7.
- Task 3 - Grab the Fire Extinguisher: Next to the fire extinguisher, there will be a panel instructing the user on how to grab the extinguisher using the controller. The user needs to point the raycast of the controller over the fire extinguisher, and press the grip button to grab it.
- Task 4 - Bring the Fire Extinguisher to the Disaster Area: Once the user has grabbed the fire extinguisher, the next task is to bring the fire extinguisher to the disaster area. Once the user accomplishes this, the instructions to use the fire extinguisher to extinguish the fire will appear over the extinguisher, as seen in Figure 8.
- Task 5 - Extinguish the Fire: Once the user has brought the fire extinguisher to the area of the apartment where the fire is spreading, they need to extinguish the fire. To do this, they have to aim the fire extinguisher at the

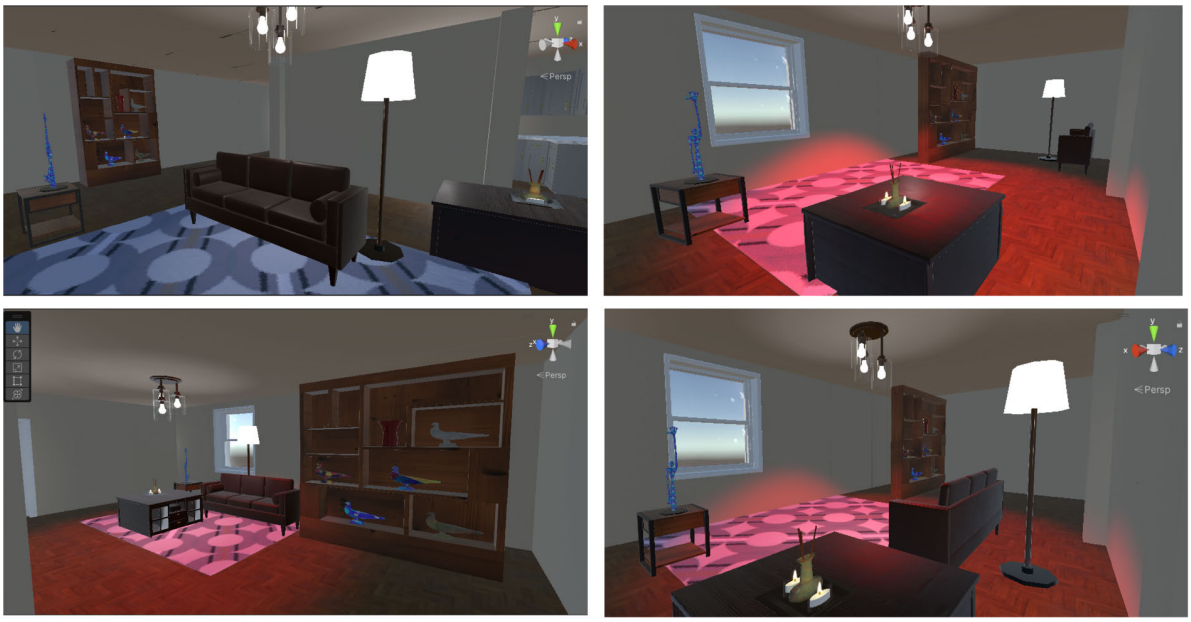


FIGURE 5. Procedurally generated scenarios using RL-PCG which generate different position of the furniture agents in every iteration of training.

FIGURE 6. Wrist UI with task panel attached to left controller.

FIGURE 7. Completion of Task 2, and instructions of performing Task 3 appearing in front of the user on a panel.

base of the fire, and press the trigger button repeatedly. The users will see that the intensity of the fire slowly

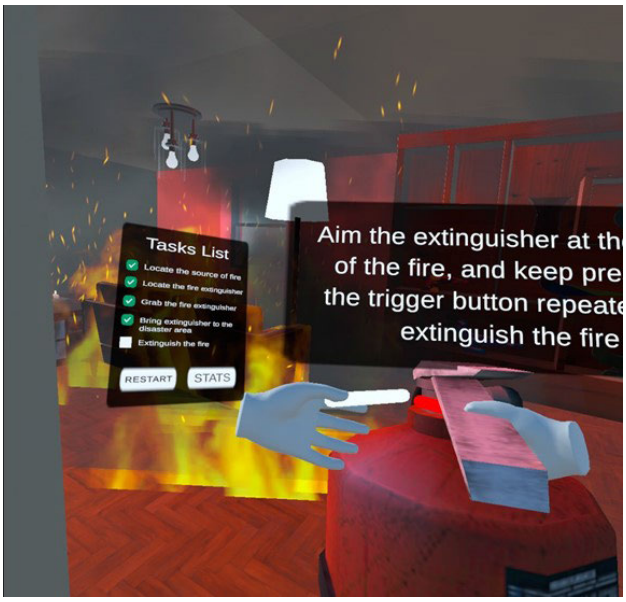
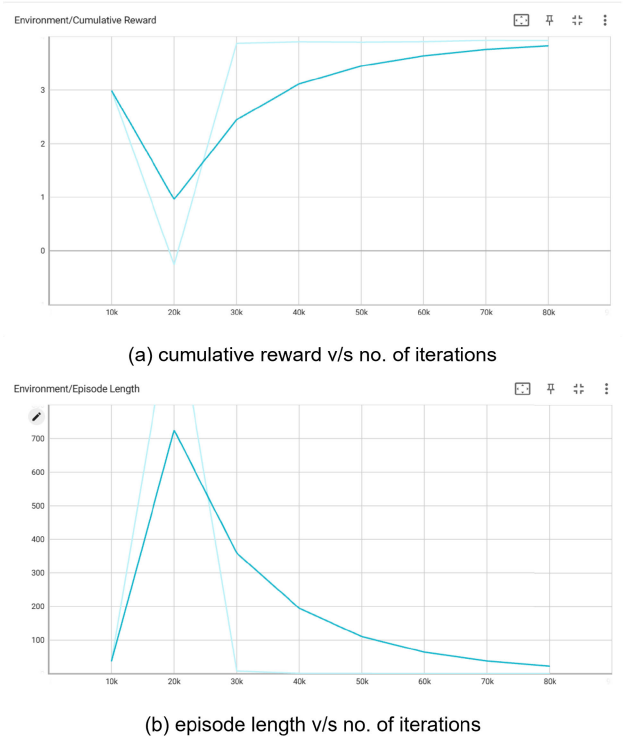


FIGURE 8. The user gets instructions on how to extinguish the fire once they bring the extinguisher to the disaster area.

starts decreasing, and eventually it will be completely extinguished. As per the third layer of our proposed 3-layer PCG architecture (refer Figure 3), if the user does not extinguish the fire completely, the fire will start regenerating itself after a specific time delay, and the user will have to extinguish it again.

C. ALTERNATE GAME FLOW

If we consider the real world, if there is a fire disaster which is not contained in time, it will eventually spread out to such an extent that it will become dangerous for humans to extinguish



(a) cumulative reward v/s no. of iterations

(b) episode length v/s no. of iterations

FIGURE 9. Training results of the flammable object RL agent, graph (a) shows the cumulative reward of the environment over the iterations and, graph (b) shows the variation in length of each episode over the iterations of training.

it at that point. That is why it is important to have a game flow for situations in which the user fails to extinguish the fire. The importance of having a failure mechanism has been emphasized on in great detail in [34] where the authors have built a failure-enhanced disaster evacuation training which creates a bias in such a way that the user is bound to fail during the first iteration in the training virtual environment. For our evacuation training system, in case the user fails to extinguish the fire in time, the fire starts spreading out in the entire room, at which point they get a warning that they need to drop everything and evacuate out of the scenario. At this point, the user needs to follow the exit signs and start moving out of the building by using the stairs. If even after the final warning, the user fails to evacuate out of the building, they will get a ''Game Over'' message indicating that they have failed to move out of the building safely before the spreading of the fire.

V. RESULTS AND DISCUSSION

The results of training the flammable object RL agents can be seen in the subsequent figures. In Figure 9, the first graph shows how the cumulative reward of the environment increases over time, with the progression of the training iterations. The reward first decreases, and then steadily starts increasing until it reaches the end of training. This indicates that the agents are performing the desired actions, i.e. placing themselves close to the fire source, while avoiding collision with the other objects and walls of the room. The second graph shows how the episode length, initially being very low, increases over time, but by the end of the training, it again becomes low. This is happening because during the start of the training, the agent collides with some objects which causes it to gain a negative reward, which ends the episode. Over time, the agent learns to not collide with the objects, thus reducing its negative reward, but since it has not achieved the positive reward, it keeps moving around the room, thus increasing the length of each episode. The episode ends only either when the agent collides with another object in the room, thus obtaining a negative reward, or if it places itself correctly in the room, thus obtaining a positive reward. Through trial and error, the agent eventually learns to obtain the desired positive reward fast, thus ending the episode, which can be seen in the graph as a steady decline in the episode length towards the end of the training. The RL agents in the scene are becoming more aware of their surroundings and are able to take decisions faster, which leads to shorter length of episodes. So, we can see that the length of the episodes is short with high negative rewards in the beginning of the training, and towards the end of the training, the episodes again become short, but with a high positive reward.

TABLE 2. Cumulative mean rewards of the untrained and trained RL agents in the virtual environment.

Cumulative Mean Reward	Sofa	TV Table	Carpet	Shelf	Sculpture Table
Untrained Agent	1.86	-0.25	1.54	-59.92	1.62
Trained Agent	3.23	3.92	3.79	3.68	2.98

TABLE 3. Comparison of our results with selected related works.

Paper	Avg. Training Time	Processor Used	Avg. No. of Steps	No. of Agents
[29]	NA	NA	NA	NA
[23] [24]	3.25 hours	i7	32,000	3
[25]	20.3 minutes	i7	2,400,000	3
[28]	43.15 minutes	NA	260,000	4
Our proposed work	1.06 hours	i5	100,000	5

Table 2 summarizes the results of our RL agents training. We can see the cumulative mean reward of each of the agents in the environment before training (untrained) and after training the agent. We can see that for all the 5 agents, the cumulative mean reward is greater for the trained agents as compared to the cumulative mean reward of the untrained agents.

The comparison of our current work with the selected cited related works has been done in Table 3. In [29], the authors have proposed a system which generates dynamic scenarios for fire evacuation training. But they have not used any RL or ML algorithms to do so. Yet we have included it as an important work for the comparison of our work, since this is

the only work we found in which the authors have considered the need for dynamic generation of scenarios for disaster training, and implemented methods to do so. The work in [23] and [24] has focused on the generation of dynamic scenarios in an Industrial Engineering environment, to maintain the interest of the users by providing them with unique scenarios during each iteration of training. They have parallelized their training and used an intel i7 processor machine to do the training of their agent, whereas our work has trained the agents in 1.06 hours, as compared to their 3.25 hours, by using an i5 processor machine and without parallelizing the training, due to the limitation of the computing power of the hardware used. Despite the limitation of hardware, we were able to achieve good results in lesser amount of time, for greater number of agents (5 agents as compared to 3 agents trained in [23] and [24]). In [25], the authors have developed a system which can generate multi-contextual dynamic scenarios to teach the same underlying academic topic. Although their training time is significantly lesser than ours, due to the fact that they have parallelized their training with 32 parallel environments and worked on the i7 processor for training, we have trained a greater number of agents (5 as compared to 3 in their work) in lesser number of steps (100 thousand steps as compared to 2.4 million steps in their work). In [28], the authors have procedurally generated dungeons using the Unity ML Agents toolkit. Their training time is less than ours, but they have mentioned that their model does not require any visual rendering to be trained, since the environment is grid based and the entire training can be done solely using the data structure in the code itself. Whereas for our training, visual rendering is essential due to the multiple possibilities of the position of objects, which are not discrete fixed values, but continuous values. Moreover, we trained a greater number of agents in lesser number of steps. These results show that we were able to achieve enhanced results in many aspects as compared to the related works in literature.

The efficiency improvements observed in training RL-PCG agents for disaster scenario generation hold significant implications for the overall effectiveness of disaster evacuation training in a virtual reality (VR) environment. The enhanced efficiency, demonstrated by faster training and fewer steps required for convergence, aligns with our overarching goal of creating a practical and impactful disaster preparedness training system. By significantly minimizing the time required to generate diverse and dynamic disaster scenarios, the proposed RL-PCG agents increase the system's readiness for rapid adaptation to evolving disaster situations. This adaptability is vital in training individuals to respond effectively to a wide range of disaster scenarios that might unfold unpredictably. The accelerated training of RL-PCG agents encourages a more iterative approach to scenario generation. With quicker feedback loops, it becomes feasible to iteratively fine-tune and explore a broader spectrum of disaster scenarios. This iterative process enhances the training system's adaptability to address a multitude of disaster variants, ultimately enriching the diversity of training experiences for users. Therefore, the efficiency improvements observed in training RL-PCG agents bear positive implications for the overall effectiveness of disaster evacuation training in a virtual reality environment and reinforce the potential of our system to prepare individuals for a multitude of disaster scenarios efficiently and effectively.

VI. CONCLUSION AND FUTURE WORK

In this study, we introduced a system for training disaster evacuation in a virtual reality environment. We used Reinforcement Learning Procedural Content Generation (RL-PCG) algorithms to make the training scenarios dynamic. Our goal was to create an affordable and secure way to prepare for disasters. We also designed a unique 3-layer PCG architecture to make the disaster simulations in virtual reality feel realistic. We made a working prototype of the evacuation training simulation.

Our contributions materialized in the form of a practical, realistic and dynamic training environment for disaster evacuation scenarios. The RL-PCG agent outperformed other RL agents in terms of lesser training time, lesser number of steps and greater number of agents, with limited processor capabilities.

In the future, we plan to extend our proposed system to simulate other types of disasters such as floods, earthquakes, and tornadoes. We also aim to explore deep reinforcement algorithms, such as Soft Actor Critic (SAC), to see if they improve the RL agents' performance compared to the current PPO algorithm. Additionally, we're considering algorithms that are specifically designed for multi-agent scenarios, like MA-POCA, to see how they perform in cooperative situations [35].

Furthermore, we plan to integrate our proposed system with other immersive technologies such as augmented reality and mixed reality to provide a more immersive and interactive disaster training experience. We also plan to investigate the use of natural language processing techniques to allow users to interact with the training simulation through voice commands.

Finally, we also plan to conduct a user evaluation study in the future to test the effectiveness of our system in terms of increasing disaster preparedness in individuals. We also plan to collaborate with disaster management organizations and emergency responders to test the effectiveness of our proposed system in real-world disaster scenarios and incorporate their feedback into future iterations of the system.

In summary, this study initiates a novel dimension in disaster training, establishing a robust foundation for future exploration, collaboration, and refinement.

ACKNOWLEDGMENT

The authors would like to thank the Vellore Institute of Technology, Chennai, for their support, encouragement, and for providing facilities that aided them in carrying out this research work.

REFERENCES

[1] S. Sharma, S. Jerripothula, S. Mackey, and O. Soumare, "Immersive virtual reality environment of a subway evacuation on a cloud for disaster preparedness and response training," in *Proc. IEEE Symp. Comput. Intell. Human-Like Intell. (CIHLI)*, Dec. 2014, pp. 1–6.

[2] S. P. Smith and D. Trenholme, "Rapid prototyping a virtual fire drill environment using computer game technology," *Fire Saf. J.*, vol. 44, no. 4, pp. 559–569, May 2009.

[3] R. R. Krishnan, S. Sushil, R. Hrishikesh, S. Devadas, A. Ganesh, and G. Narayanan, "A novel virtual reality game for disaster management applications," in *Proc. Int. Conf. Commun. Signal Process. (ICCSP)*, Apr. 2019, pp. 0254–0257.

[4] S. Garcia, P. Trejo, and A. Garcia, *Intelligent VR-AR for Natural Disasters Management*. IntechOpen, 2021.

[5] S. Ooi, T. Tanimoto, and M. Sano, "Virtual reality fire disaster training system for improving disaster awareness," in *Proc. 8th Int. Conf. Educ. Inf. Technol.*, Mar. 2019, pp. 301–307.

[6] A. R. Caballero and J. D. Niguidula, "Disaster risk management and emergency preparedness: A case-driven training simulation using immersive virtual reality," in *Proc. 4th Int. Conf. Human-Comput. Interact. User Exper.*, Mar. 2018, pp. 31–37.

[7] C. Ziker, B. Truman, and H. Dodds, "Cross reality (XR): Challenges and opportunities across the spectrum," in *Innovative Learning Environments in STEM Higher Education: Opportunities, Challenges, and Looking Forward*. Springer, 2021, pp. 55–77.

[8] S. Khanal, U. S. Medasetti, M. Mashal, B. Savage, and R. Khadka, "Virtual and augmented reality in the disaster management technology: A literature review of the past 11 years," *Frontiers Virtual Reality*, vol. 3, Apr. 2022, Art. no. 843195, doi: 10.3389/frvir.2022.843195.

[9] H. Mitsuhara, C. Tanimura, J. Nemoto, and M. Shishibori, "Expressing disaster situations for evacuation training using markerless augmented reality," *Proc. Comput. Sci.*, vol. 192, pp. 2105–2114, Jan. 2021.

[10] R. Watkins, *Procedural Content Generation for Unity Game Development*. Packt Publishing Ltd, 2016.

[11] M. Hendrikx, S. Meijer, J. Van Der Velden, and A. Iosup, "Procedural content generation for games: A survey," *ACM Trans. Multimedia Comput., Commun., Appl. (TOMM)*, vol. 9, no. 1, pp. 1–22, 2013.

[12] G. Nwankwo, S. Mohammed, and J. Fiaidhi, "Procedural content generation for dynamic level design and difficulty in a 2D game using UNITY," *Int. J. Multimedia Ubiquitous Eng.*, vol. 12, no. 9, pp. 41–52, Sep. 2017.

[13] J. Freiknecht and W. Effelsberg, "A survey on the procedural generation of virtual worlds," *Multimodal Technol. Interact.*, vol. 1, no. 4, p. 27, Oct. 2017.

[14] S. Azad, C. Saldanha, C.-H. Gan, and M. O. Riedl, "Mixed reality meets procedural content generation in video games," in *Proc. 12th Artif. Intell. Interact. Digital Entertainment Conf.*, 2016, pp. 22–26.

[15] P. Acevedo, M. Choi, H. Liu, D. Kao, and C. Mousas, "Procedural game level design to trigger spatial exploration," in *Proc. 17th Int. Conf. Found. Digit. Games*, Sep. 2022, pp. 1–11.

[16] J. P. A. Campos and R. Rieder, "Procedural content generation using artificial intelligence for unique virtual reality game experiences," in *Proc. 21st Symp. Virtual Augmented Reality (SVR)*, Oct. 2019, pp. 147–151.

[17] A. Petrovas and R. Bausys, "Procedural video game scene generation by genetic and neutrosophic WASPAS algorithms," *Appl. Sci.*, vol. 12, no. 2, p. 772, Jan. 2022.

[18] A. Summerville, S. Snodgrass, M. Guzdial, C. Holmgard, A. K. Hoover, A. Isaksen, A. Nealen, and J. Togelius, "Procedural content generation via machine learning (PCGML)," *IEEE Trans. Games*, vol. 10, no. 3, pp. 257–270, Sep. 2018.

[19] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *J. Artif. Intell. Res.*, vol. 4, no. 1, pp. 237–285, Jan. 1996.

[20] P. El Sharouni, "Autonomous lane merging: A comparison between reinforcement learning algorithms," B.S. thesis, Dept. Humanities, Univ. Utrecht, Utrecht, The Netherlands, 2021.

[21] E. V. Denardo, "A Markov decision problem," in *Mathematical Programming*. Amsterdam, The Netherlands: Elsevier, 1973, pp. 33–68.

[22] A. Rovira and M. Slater, "Reinforcement learning as a tool to make people move to a specific location in immersive virtual reality," *Int. J. Hum.-Comput. Stud.*, vol. 98, pp. 89–94, Feb. 2017.

[23] C. E. Lopez, O. Ashour, and C. S. Tucker, "Reinforcement learning content generation for virtual reality applications," in *Proc. 39th Comput. Inf. Eng. Conf.*, Aug. 2019, Art. no. V001T02A009.

[24] C. E. López, J. Cunningham, O. Ashour, and C. S. Tucker, "Deep reinforcement learning for procedural content generation of 3D virtual environments," *J. Comput. Inf. Sci. Eng.*, vol. 20, no. 5, Oct. 2020.

[25] J. Cunningham, C. Lopez, O. Ashour, and C. S. Tucker, "Multi-context generation in virtual reality environments using deep reinforcement learning," in *Proc. 40th Comput. Inf. Eng. Conf. (CIE)*, Aug. 2020, Art. no. V009T09A072.

[26] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 1889–1897.

[27] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.

[28] M. W. R. Lopes, "Generating procedural dungeons using machine learning methods: An approach with Unity-ML," Federal Fluminense Univ., Niterói, Brazil, Tech. Rep. 22645, 2020.

[29] K. Hullett and M. Mateas, "Scenario generation for emergency rescue training games," in *Proc. 4th Int. Conf. Found. Digit. Games*, Apr. 2009, pp. 99–106.

[30] J. Hanski and K. B. Bicak, "An evaluation of the unity machine learning agents toolkit in dense and sparse reward video game environments," Uppsala Univ., Uppsala, Sweden, Tech. Rep. 444982, 2021.

[31] J. Tichon, R. Hall, M. Hilgers, M. Leu, and S. Agarwal, "Education and training in virtual environments for disaster management," in *Proc. Assoc. Advancement Comput. Educ. (AACE)*, 2003, pp. 1191–1194.

[32] J. Molka-Danielsen, E. Prasolova-Førland, L. M. Hokstad, and M. Fominykh, "Creating safe and effective learning environment for emergency management training using virtual reality," in *Proc. Norsk Konferanse Organisasjoners Bruk Av Informasjonsteknologi (NOKOBIT)*, 2015, pp. 1–10.

[33] S. Bhide, "Fire safety and emergency evacuation training for occupants of building using 3D virtual simulation," Univ. Central Florida, Orlando, FL, USA, Tech. Rep. 5725, 2017.

[34] H. Mitsuhara, C. Tanimura, J. Nemoto, and M. Shishibori, "Failure-enhanced evacuation training using a VR-based disaster simulator: A comparative experiment with simulated evacuees," *Proc. Comput. Sci.*, vol. 159, pp. 1670–1679, Jan. 2019.

[35] C. Hakansson and J. Froberg, "Application of machine learning to construct advanced NPC behaviors in Unity 3D," Linköping Univ., Linköping, Sweden, Tech. Rep. 1604156, 2021.

JIGYASA AGARWAL is currently pursuing the bachelor's degree in computer science and engineering with the Vellore Institute of Technology, Chennai, India. She is a former XR Research Intern through the Mitacs GRI research internship 2022 with OCAD University, Toronto, Canada. Her research interests include AR/VR, mixed reality, the Internet of Things, cloud computing, and AI/ML.

S. SHRIDEVI received the Ph.D. degree in computer science from Manonmaniam Sundaranar University (MSU), Tirunelveli. She is currently a Professor and the Deputy Director of the Centre for Advanced Data Science, Vellore Institute of Technology, Chennai. She has published many patents and research papers in reputed international journals and conferences and has received best research awards for her works. She received seed funds for her research project and had applied to funds for many government agencies as well. Her research interests include semantic artificial intelligence, web mining, machine learning, and deep learning. She was a University Rank Holder and a Medalist in her master's degree.