DEEP REINFORCEMENT LEARNING FOR PROCEDURAL CONTENT GENERATION OF 3D VIRTUAL ENVIRONMENTS

Christian E. Lopez[1]
Computer Science, Mechanical Engineering
Lafayette College,
Easton, PA 18042
lopezbec@lafayette.edu
Mem. ASME

James Cunningham
Mechanical Engineering
Carnegie Mellon University
5000 Forbes Avenue
jamescun@andrew.cmu.edu
Mem. ASME

Omar Ashour
Industrial Engineering
The Pennsylvania State University,
Erie, PA 16563
oma110@psu.edu

Conrad S. Tucker
Mechanical Engineering, Machine Learning
Carnegie Mellon University
5000 Forbes Avenue
conradt@andrew.cmu.edu
Mem. ASME

ABSTRACT

This work presents a Deep Reinforcement Learning (DRL) approach for Procedural Content Generation (PCG) to automatically generate 3D virtual environments that users can interact with. The primary objective of PCG methods is to algorithmically generate new content in order to improve user experience. Researchers

[1] Corresponding author. 569 Rockwell Integrated Science Center, Lafayette College, Easton, PA 18042, USA.

*The conference version of this paper received the ASME CIE Virtual Environments and Systems committee best paper award.

have started exploring the use of Machine Learning (ML) methods to generate content. However, these approaches frequently implement supervised ML algorithms that require initial datasets to train their generative models. In contrast, RL algorithms do not require training data to be collected a priori since they take advantage of simulation to train their models. Considering the advantages of RL algorithms, this work presents a method that generates new 3D virtual environments by training an RL agent using a 3D simulation platform. This work extends the authors' previous work and presents the results of a case study that supports the capability of the proposed method to generate new 3D virtual environments. The ability to automatically generate new content has the potential to maintain users' engagement in a wide variety of applications such as virtual reality applications for education and training, and engineering conceptual design.

1. INTRODUCTION

The objective of Procedural Content Generation (PCG) methods is to automatically generate content. Since the 1980s, the gaming industry has been using PCG methods to generate new game levels by manipulating game design elements, such as terrains, maps, and objects [1]. Similarly, researchers have started exploring how automatically generating new content for e-learning applications can help advance Adaptive Instructional Systems, such as Intelligent Tutoring Systems [2], [3]. The ability to automatically generate new content offers several advantages for the design and development of a wide range of applications [4]. For example, automatically generating new content can help reduce the resources needed to develop new applications. PCG methods can help designers explore the design space, and potentially help co-create more creative content. More importantly, content that is automatically generated can be personalized to an individual's unique attributes in order to maximize the user experience

[5]–[7]. The use of PCG methods to generate new content has been shown to improve user experience and engage users (e.g., replay value) [7]–[9].

In recent years, researchers have started integrating Machine Learning (ML) algorithms to automatically generate new content [1], [10], [11]. However, PCG methods that implement ML algorithms require datasets to train their generative models, since these algorithms frequently use supervised learning methods. In contrast, Deep Reinforcement Learning (DRL) based methods are capable of generating efficient representations of complex situations and tasks by implementing sensory input information obtained from simulation environments (e.g., pixels acquired from images of a video game) [12]. Hence, there is no need to capture training data *a priori*, which can help reduce cost [5]–[7].

Given the advantages of the PCG methods and the potential of RL algorithms, this work presents a PCG method based on a Deep RL approach that generates new virtual environments. Figure 1 shows an outline of this method. A Deep RL agent is presented that generates new 3D virtual environments that are validated via a 3D simulation platform. In this work, the term "*virtual"* represents a 3D computer generated (virtual) environment that users can interact with. The RL agent generates new virtual environments according to individuals' preferences for the location of a subset of virtual objects. Once a new 3D virtual environment is generated, the user can interact with it using a variety of interfaces (e.g., immersive VR headset, smartphone, computer). This work extends the authors' previous work [13], and presents the results of a case study

that supports the ability of the proposed method to generate new 3D virtual environments.

2. LITERATURE REVIEW

2.1 Procedural Content Generation

Procedural Content Generation (PCG) can be defined as the field that studies the development of algorithms and methods capable of automatically generating content. The gaming industry has used PCG for decades [1]. Most of the early PCG methods were composed of rules sets and heuristics that guided the content generation process or functions to evaluate the generated content. These heuristics and functions were developed by designers based on their understanding of the application [6], [14]. However, in recent years, researchers have started exploring the use of supervised Machine Learning (ML) algorithms to train generative models capable of automatically creating new content [11].

One of the most well-known projects that integrate supervised ML to generate new game environments is "*Mario AI*" (www.marioai.org) [15]–[17]. Researchers have presented a wide range of PCG methods to automatically generate new environments for a variety of popular games [1], [14]–[17]. For example, Summerville and Mateas [1] introduce a Long Short-Term Memory Recurrent Neural Networks framework to generate new Super Mario Brothers levels. Their model was trained using a corpus of 39 existing levels of Super Mario Brothers, which they were able to augment by using several training techniques (e.g., stacking). Moreover, Summerville et al. [17] present a Bayesian Network to automatically generate level topologies for Zelda-like games. They annotated the

topology characteristics of 38 levels of different Zelda games in order to train their generative model. Justesen et al. [18] attempt to overcome the overfitting problem that arises when training Deep Reinforcement Learning (DRL) agents on static game environments (e.g., training a model to play an Atari game by using just one level) by introducing a search-based PCG. Their progressive PCG method helps control for the difficulty of levels to match the Deep RL agent being trained to play the game. They trained their models using levels from the games of Zelda, Solarfox, Frogs, and Boulderdash. Similarly, researchers have introduce PCG methods based on Markov Chain [19], [20], and Matrix factorization approaches [21]. However, these methods still require human-authored content to train their models.

Most of the current PCG methods that implement supervised ML methods require some initial dataset to train their generative models. In contrast, RL algorithms implement high-dimensional sensory input to generate efficient representations of complex situations and tasks with the use of simulation [12]. Hence, there is no need to capture training data *a priori*. Based on these advantages, this work presented a PCG method to generate new 3D virtual environments based on a Deep RL approach.

2.2 Adaptive Instructional Systems

The field of Adaptive Instructional Systems (AISs) has greatly benefited from integrating methods to generate new content for their adaptive applications [22]. These types of systems require significantly more content than their non-adaptive counterparts since for each adaptation, new content is required [23]. AISs are defined as "*class of*

intelligent, machine-based tools that guide learning experiences by tailoring instruction and recommendations based on the goals, needs, and preferences of each learner [or team] in the context of domain learning objectives" [23]. Intelligent tutoring systems, intelligent method, recommender systems, personal assistants, and intelligent instructional media fall under the umbrella of AISs.

Within this field, RL has been used to model students' learning styles and develop pedagogical policy strategies [3], [9], [24], [25]. However, there has been a limited number of studies that have explored how to automatically generate new content for learning purposes [26], [27]. For example, Hullett and Mateas [8] present an application capable of generating new scenarios for a firefighting training application. The application was able to generate different scenarios of buildings partly collapsed based on the desired skills the users wanted to train on. Smith et al. [28] implement a method for creating levels in a learning application aimed at teaching students about fractional arithmetic. The method implements a constraint-focused generator design approach. Similarly, a learning application that implemented PCG and gamification to engage students in solving math problems is introduced in [29]. This method was founded on template-based and constructive algorithms.

In the context of conflict resolution, a serious game application that combined a Player Modeling and a metaheuristic-search PCG approach is introduced in [30]. This PCG method was driven by a Neural Network used to predict the distribution fairness of the players. The results of this study support the value of PCG to guide the learning of individuals toward targeted learning objectives. Most recently, Hooshyar [7], proposed a

PCG framework for educational game applications based on a Genetic Algorithm (GA) approach. The framework allows designers to control the generation process, given various learning objectives and preferences. In a different study, Hooshyar [26] presents a data-driven PCG approach based on a Genetic and a Support Vector Machine algorithms. They implemented their method in a language learning application and compared the method against a heuristic-based approach. Their results indicate that their data-driven approach was more effective at generating content that matched the performance target of individuals, compared to the heuristic approach. Similarly, Sottilare [23] presents a ML method based on a GA approach to automatically generate new scenarios from a set of parent scenarios for virtual instructional and game-based applications.

The previous studies show how PCG methods can be implemented in learning applications and their potential benefits. These studies also show that researchers are starting to use ML approaches (e.g., Neural Network, Support Vector Machines, Genetic algorithms) to train their PCG models. They train their models on datasets from existing content or datasets containing users' data, which has to be generated or collected *a priori* [7], [26], [30]. The process of generating new content to use as training dataset can require significant time and resources [5]–[7]. In recent years, researchers have started exploring how realistic, synthetic data can be automatically generated [31], [32]. However, while studies have shown that these approaches can generate synthetic datasets that cannot be accurately distinguished from human generated ones [33]–[35], they still require some initial datasets to train their models. In contrast, RL approaches do

not require training data to be collected *a priori* since they take advantage of simulation to train their models. Based on the limitations of supervised ML algorithms and the advantages of RL algorithms, this work presents a PCG method based on a Deep RL approach. The RL agent is trained using a simulation platform to automatically generate new 3D virtual environments, which could potentially be used for learning applications.

2.3 Reinforcement Learning

While traditional supervised ML algorithms require the use of a training dataset, RL methods do not require a training dataset to be collected *a priori* since they take advantage of simulation environments to generate efficient representations of complex situations and tasks [12]. The RL process can be understood as a Markov Decision Processes, were the RL agent connects to a simulation environment via different sensory inputs. The objective of the agent is to develop a model that selects the actions that maximize its long-run reward. In other words, the agent creates the desired action policy by process of trial and error via simulation [36]. Hence, a RL agent can be described as a software agent capable of inducing an action policy in an uncertain environment with delayed rewards [37].

RL methods are suitable for solving learning control problems, which are challenging for traditional supervised ML algorithms and dynamic programming optimization methods [38]. RL agents focus on generating an action policy that can adapt to changes in the environment (e.g., state-space). Researchers have used RL methods to train agents capable of mastering complex tasks at human-level performance [39]–[41]. In recent years, Deep RL algorithms have been implemented to master and perform a wide range of tasks, from Atari games to the Chinese game of Go [39], [40]. Thanks to these advancements, researchers argue that these algorithms will revolutionize the field of Artificial Intelligence [12]. In addition, the rapid development of these RL methods has been encouraged by the rise of easy to use, scalable simulation platforms [42]–[44].

In the context of Adaptive Instructional Systems (AISs), RL based methods have shown promising results in helping personalized narrative-centered learning environments. For example, Wang et al. [45] present a Deep RL framework to personalize interactive-narrative for an educational game. Similarly, Rowe et al. [46] introduce a multi-armed bandit computational formalism, consisting of several components of a Deep RL framework, to generate new training scenario for the Army. The authors also explored Long-Short Term Memory Networks approaches and stated that in future work, they would be implementing RL algorithms to help generate new complex training scenarios.

Table 1 shows a summary of existing work related to methods that automatically generate content (i.e., PCG). This table shows that while PCG methods are frequently used in gaming applications, researchers are starting to explore the use of PCG methods for learning purposes. However, most of the studies on learning applications implement meta-heuristics. In light of the advantages of PCG methods and the potential of RL algorithms, this work presents a PCG method based on an RL approach that generates new 3D virtual environments. The RL agent validates the new 3D virtual environments via a simulation platform; hence, it does not require any training data to be collected *a priori*. The RL based PCG method is implemented in a case study to generate new layouts for a virtual 3D manufacturing environment used for an e-learning application.

In the authors' previous work, initial results of the performance of the RL agent's reward score was presented [13]. The results show that the RL agent did not reach the maximum reward score, but that its reward score was significantly and strongly correlated with the training iterations ($\rho = 0.98$, p-value < 0.001). In other words, the RL agent was

not able to generate a 3D virtual environment that was completely functional, and that maximized the rewards function. However, it managed to model an action policy that maximized the long-run rewards function. Moreover, in the previous work, the training of the agent was not parallelized and the training time-constrained to less than 6 hours due to computational limitations. These factors played a significant role in the performance of the RL agent. Based on these limitations, in this work, the authors extended their previous study by implementing parallelized training over 60 thousand iterations. In addition, the reward function of the RL agent and simulation environment used for training have been enhanced in order to incentivize the generation of more realistic and functional layouts. Finally, the results of a case study that support the capability of the proposed method to generate new 3D virtual environments are presented in this work.

2.4 Reinforcement Learning and Operations Research

The objective of PCG methods to generate new environments given certain criteria is analogous to the Operations Research (OR) problem of Facility Layout Planning (FLP). The objective of FLP algorithms is to identify the optimal arrangement of equipment or facilities in accordance with some criteria and given certain constraints [47]. FLP problems are a NP-complete problem, which means that "*the computational time required to find an optimal solution increases exponentially with the problem size*" [48, p. 25]. This is one of the reasons why researchers have proposed multiple meta-heuristics algorithms to solve the FLP problem, such as Simulated Annealing and Genetic Algorithms

[47]. However, one of the limitations of optimization approaches is that a given optimal solution might not continue to be optimal under a different problem configuration. For example, if an additional constraint is added (e.g., now machine *Z* must be in the coordinates *x* and *y*), the algorithm needs to be run again to find the optimal or near-optimal solution [47], [49]. In contrast, since RL algorithms focus on generating an action policy that can adapt to changes in the state space, they do not require additional training when exposed to a new state (e.g., now machine Z must be in the coordinates x and y).

Due to the advantages of RL algorithms, researchers have explored how to implement RL in combination with metaheuristics with the objective of identifying more efficient methods for solving OR problems [50]–[55]. Recently, some studies have shown promising results of using RL for solving combinatorial optimization problems [56]. For example, RL algorithms have been implemented to tackle classical OR problems like dynamic job shop scheduling problem [57], vehicle touting problem [58], among others routing and scheduling problems [56]. In a recent study, Govindaiah and Petty [59], [60] present the application of a framework that integrates RL algorithms and discrete event simulation to improve the cost efficiency of material handling plans under varying product demands. Their method focused on reducing the cost of material handling plans by changing the routes, timing, and equipment used to transport material between workstations and/or warehouses. However, their method did not consider the locations of the workstations nor warehouses, the reason why it cannot be implemented for FLP [49]. The case study used in this work to test the proposed Deep RL PCG method shares some characteristics with the material handling problem tackled by Govindaiah and Petty

[59], [60]. However, the proposed method focuses on generating new 3D virtual environments by allocating a set of virtual objects. In the case study presented, the RL agent is capable of changing the location of the workstation (i.e., injection molding machine, see section 4) and material handling equipment (e.g., conveyor belts and robot arms). Moreover, the proposed Deep RL PCG method can adapt to changes in the problem space (i.e., state space) without the need for additional training. That is, once the RL agent is trained, it can generate new 3D virtual environments given different injection molding machine locations (see section 5 for results). In contrast, using traditional OR methods would require to run optimization or meta-heuristic algorithms every time the problem space changes when a constraint is added or modified (e.g., now machine Z must be in the coordinates x and y) [47], [49].

3. METHOD

In this work, a PCG method based on a Deep RL approach is introduced. The method is capable of dynamically generating new 3D virtual environments by implementing a RL agent that validates the content via a 3D simulation platform. Figure 2 shows the method of the Deep RL algorithm implemented. In addition, it shows 2D aerial views of the 3D simulation platform used to validate the virtual manufacturing environments generated for the case study (see section 4).

RL problems are framed as Markov Decision Processes, where the agents connect to the simulation environment at a given time *t* via the sensory inputs of state ($\mathbf{S}_t$) that belongs to the set of possible states **S**, and action ($\mathbf{A}_t$) that belongs to the set of possible

actions **A** (see Fig. 2). In each training epoch *t,* the agent observes the current state: $\mathbf{S}_t$ and chooses an action to be executed: $\mathbf{A}_t$. The environment reacts to the action executed and determines the new state to transition: $\mathbf{S}_{t+1}$, as well as the reward signal (i.e., reinforcement signal): R_t. The sensory inputs of the state and action can be in a vector form, containing information about the state of the environment and information regarding the action the agent is taking, respectively. The agent makes decisions based on a *policy* that is defined by a mapping from the state space to a probability distribution over the action space, formalized as $\pi(\mathbf{S}_t) \in P(\mathbf{A})$. In Deep RL, this policy function is realized using a neural network which takes $\mathbf{S}_t$ as input, and generates probabilities for selecting each possible action as output.

The goal of an RL agent is to determine a particular policy π^* which maximizes the long-run reward of the agent. The long-run reward, also known as the return, is used as an objective function over the reward signal itself because it is more stable and less sparse. The return is defined as $\rho = \sum_{t=0}^{T} \gamma^t R_t$, where $\gamma \in [0,1]$ is the discount factor that controls the exponential devaluation of delayed rewards.

In this work, the Proximal Policy Optimization (PPO) [61] algorithm is employed to train the RL agent. PPO is a policy gradient approach to RL based on the Trust Region Policy Optimization (TRPO) algorithm introduced by Schulman et al.'s work [62]. Schulman et al.'s [61] study reveals that the PPO algorithm outperformed other policy gradient algorithms, and provided a more favorable tradeoff between sample complexity, simplicity, and wall-time. Given that a neural network is fully differentiable, gradient ascent can be applied to the policy function directly with respect the advantage estimate

of the policy, a quantity which is related to the expected value of the policy's return (readers are referred to [61] and [62] for additional details).

For generating new 3D virtual environment, $\mathbf{S}_t$ contains enough information to describe the current state of the virtual environment (e.g., location, orientation, and relevant properties of objects in the environment), while $\mathbf{A}_t$ corresponds to ways that the agent can alter the environment. These actions could correspond to determining the location, orientation, or parameters which govern the behavior of the virtual objects in the 3D virtual environment (see Fig. 2). Given this framing, the proposed DRL method can be applied to generate 3D virtual environments for problems that can be framed as arranging multiple individual objects with inherent properties, and that can be expressed in vector form. This enables the use of this method in a variety of application, such as educational and training where new environments are generated for learning purposes, or game applications where new levels are generated for entertainment purposes.

The goal of the RL agent is to develop a model that selects the actions that maximize its long-run reward signal, which takes the form of a scalar value. The elements of the reward function will depend on the behavior that the designers expected the RL agent to model (i.e., learn). The RL agent needs to be rewarded for generating new environments that are functional and not just a random placement of virtual objects. This can be achieved by designing a reward function that incentivizes the generation of functional environments and penalizes nonfunctional ones (e.g., makes parts as in the manufacturing layout example of Fig. 1 & 2). However, a major difference between layout generation problems and other RL problems is that each action (i.e., placement of an

object) cannot be evaluated until the full layout has been generated. This means that every action except the final will have an immediate reward of $R_t = 0$. Thus, the proposed method omits the discount factor from the return, by choosing $\gamma = 1$.

Through its interactions with the simulated environment, the RL agent is trained (i.e., learns) to model an action policy that will maximize its return. Once the RL agent is trained, it will be able to generate new 3D virtual environments given an initial state provided by the user or randomly selected by the agent. In the example shown in Fig.2, this could be the initial location of the injection molding machine. Hence, the RL agent will place the objects in a way that will create a functional virtual 3D manufacturing layout.

4. CASE STUDY

For this case study, the authors used a Virtual Reality (VR) learning application designed to teach Industrial Engineering (IE) concepts (i.e., Poisson distribution, Little's Law, Queuing Theory) with the use of a simulated manufacturing system. Specifically, a manufacturing system that produces power drills was simulated, as shown in Fig.1 & 2. The objective of this VR learning application is to provide a tool with a common theme that educators could use to teach IE concepts and integrate course knowledge into their curriculum [63]. A power drill manufacturing line was selected since previous studies that aim to integrate IE course knowledge have implemented similar power tools [64]. The

virtual environment simulates the initial steps of the process to manufacture a power drill, where the plastic housing is manufactured.

Figure 3 shows, from a user's point of view, a functional layout for this manufacturing system. In this layout, first, an injection molding press produces the plastic housing components. Then, they are cooled down with the use of a conveyor belt. Finally, the plastic housings are placed in a tote with the use of a robotic arm in order to be transported to the assembly line. The 3D virtual environment allows users to interact with virtual objects. For this application, the agent is rewarded based on the efficiency and functionality of the layout generated to produce goods (e.g., the rightmost image on Fig. 2 has a high reward score, while the two other images have a low reward score). Specifically, the reward function used in this case study can be mathematically expressed as follows:

$$R = \beta_1(Tote) - \beta_2(Floor) + \beta_3\sigma_\lambda + \beta_4(Flow) \quad (1)$$

For,

$$Tote = \sum_{p=1}^{\mathrm{P}} \phi_p \quad (2)$$

$$Floor = \sum_{p=1}^{\mathrm{P}} \varphi_p \quad (3)$$

$$\sigma_\lambda = \sqrt{\frac{\sum_{e=1}^{\mathrm{E}}(\lambda_e - \lambda_U)^2}{\mathrm{E}}} \quad (4)$$

$$Flow = \sum_{p=1}^{\mathrm{P}} \sum_{e=1}^{\mathrm{E}} \Delta_{p,e} \quad (5)$$

$$\beta_i > 0 \ \ \forall \ i = \{1,2,34\} \tag{6}$$

Where,

- ϕ_p is a binary variable that indicates if a given part *p* was correctly placed in a tote $\phi_p = 1$ or not $\phi_p = 0$, for *p* ϵ{**P**}
- φ_p is a binary variable that indicates if a given part *p* falls to the floor $\varphi_p = 1$ or not $\varphi_p = 0$, for *p* ϵ{**P**}
- λ_e is a parameter that describes the behavior distribution of equipment *e*, for *e* ϵ{**E**}. (In the case study, this parameter is only applied to the conveyor and injection machine)
- $\Delta_{p,e}$ is a binary variable that indicates if a given part *p* interacted with a given equipment *e*, for *p* ϵ{**P**} and *e* ϵ{**E**}

The reward function shown in Eq. (1), will be maximized when all the parts *p* are placed in a tote and no parts fall on the floor, following Eq. (2)-(3), when the standard deviation of the parameters that describe the behavior distribution of the equipment set {**E**} is minimized, following Eq. (4), and when all the parts interact with all the equipment following the manufacturing process, as shown in Eq. (5). This reward function was designed to reinforce the generation of functional manufacturing layouts that follow the predefined manufacturing process and have a constant flow of part being placed in the tote. This reward function will be computed for every simulation epochs *t* (R_t), as shown in Fig. 2. In addition, in every simulation epoch *t*, the RL agent will be able to control the

placement ($\mathrm{x}_e, \mathrm{y}_e$), and the parameters that describe the behavior distribution (λ_e) of the equipment set {**E**}. The environment will provide the agent with the state information about the placement (x_u, y_u) of the equipment placed by the user {**U**}. The set of equipment {**U**} will allow users to customize the VR environment. In the event that the user does not need to customize the environment, the equipment set {**U**} can be placed randomly, to generate a new environment.

For this application, users can select the location of the injection molding machine, **U** = {Injection molding machine}. On the other hand, the RL agent will manipulate one conveyor belt, one tote, and one robot arm. This means that the set **E** will contain three different equipment (i.e., virtual objects). In order for the agent to be robust to various placements of the injection molding machine, the position of the injection machine was randomly changed at every epoch *t*, and thus to maximize the reward the agent would need to generate a functional layout regardless of the position of the machine. The $\lambda_{conveyor}$ parameter will control the speed of the conveyor, while $\lambda_{machine}$ parameter will control the speed of the injection molding machine. To improve training performance, the RL agent is trained in parallel with multiple layouts. This provides the benefit in improved diversity of training samples by ensuring that the agent is exploring multiple action trajectories simultaneously. In this case study, the agent is trained in parallel on 32 environments.

Finally, in this work, the game engine Unity [65] is used as the 3D simulation platform to train the RL agent. Because of its fidelity, physics simulation capabilities, accessibility, and community support, Unity is widely used by developers [66], [67], as well as by

researchers [68], [33]. Furthermore, Unity ML-Agents Toolkit [44] provides several algorithms and functionalities for the development and design of RL based applications [69]. In this case study, for each simulation epoch *t*, a total of 10 parts were simulated (i.e., p = {1-10}). This number of parts was selected to reduce the complexity of the simulation while allowing the simulation platform to generate the state transition in which the environment reacts to the action executed and provides a reward signal. However, this number can be increased, and the relative difference between the rewards score of layouts would not change. That is, a layout that allows all the parts to fall on the floor will always have a worse reward score than one that places all the parts on the tote, no matter how many parts are simulated.

5. RESULTS AND DISCUSSIONS

The RL agent was trained using an Intel® Core™ i7-4770K 3.50 GHz CPU and 16 GB RAM. A total of 10,000 training iterations (t = 10000) on 32 simulated environments were used to train a RL agent in parallel. This means that a total of 32,000 virtual environments were generated and evaluated to train the RL agent. The total training time was 3.25 hours. In this work, the coefficient of our rewards function (i.e., $\beta_1, \beta_2, \beta_3, \beta_4$) were empirically set to one in order to give the same importance to the elements of the reward function. Figure 4 shows the evolution of the RL agent's average reward over the 32 environments given the training epoch *t*. The *y*-axis shows the bounds of the reward function (i.e., [-11, 21]). Figure 4 shows that the agents' rewards score was significantly and strongly correlated with the simulation iterations (ρ = 0.915, p-value < 0.001). This

indicates that the agent managed to train a model that describes an action policy that maximized the long-run rewards function used in this case study.

To test the performance of the trained RL agent, a total of 512 (i.e., 32*16) new 3D virtual environments were generated and evaluated. This process took 2 minutes and 20 seconds. That is, the trained RL agent takes, on average, 0.27 seconds to generate a new 3D virtual environment. Table 2 shows the number of layouts generated given the rewards score achieved and a description as to why the rewards score was not optimal (i.e., 21). This table shows that, on average, these layouts had a reward score of 13.18 (SD=7.74). This is in line with the average rewards score achieved during the last iteration of the training process (see Fig. 4). It also shows that more than 55.08% of the layouts achieved a reward score greater than 18 and only mismatched the speed between the conveyor and the injection molding machine (i.e., σ_{λ}). Figure 5 shows 2D aerial views of several of the environments generated for testing the performance of the trained RL agent.

The results indicate that the agent managed to train a model that describes an action policy that maximized the long-run rewards function used in this case study. Moreover, the results show that the trained RL agent is capable of generating new 3D virtual environments given different injection molding machine locations without the need for additional training and in less than a second. This is in contrast with common methods used for the FLP, which required to re-run optimization algorithms every time the problem space changes (e.g., the injection molding machine locations changes). This finding shows promising results for using PCG methods based on Deep RL approaches to

generate new 3D virtual environments. The capability to generate new 3D virtual environments given different initial configurations of virtual objects can help personalize applications to an individual's unique preferences.

6. CONCLUSION AND FUTURE WORKS

The ability to automatically generate new content with the use of Procedural Content Generation (PCG) methods offers several advantages for the development and design of new applications. PCG methods can help reduce the resources needed to develop new applications. More importantly, content that is automatically generated can be personalized and adapted to an individual. Implementing PCG methods allows designers to generate new environments that can help improve the overall user experience. Researchers have started developing PCG methods that integrate supervised ML algorithms, which allow designers to generate new content more efficiently compare to heuristics-based methods. However, these algorithms require large datasets to train their generative models. In contrast, Reinforcement Learning (RL) methods do not require any training data to be collected *a priori* since they take advantage of simulation environments to generate efficient representations of complex situations and tasks.

In light of this, a PCG method based on a Deep RL approach that generates new virtual environments is presented. This method trains a model by implementing a RL agent that validates new 3D virtual environments via a 3D simulation platform; hence, it does not require any training data to be collected *a priori*. In this work, a case study is introduced where the proposed method is used to generate new 3D virtual

manufacturing environments, with the intention to teach Industrial Engineering (IE) concepts. The preliminary results indicate that the RL agent was able to model (i.e., learn) a policy that allows it to automatically generate new and functional 3D virtual environments.

The proposed Deep RL PCG approach can help designers automatically generate new content for a wide range of applications. For example, Figure 6 shows how the 3D virtual environment generated for the case study can be integrated into an immersive VR learning application. This immersive VR application can help users learn about IE concepts. The PCG method presented can also be applied to other applications that can benefit from automatically generating new 3D virtual environments (e.g., Adaptive Instructional Systems, adaptive games). Designers can implement this method in their applications by creating a rewards function based on the new environments they would like the RL agent to generate.

While this work presents a novel PCG method based on a Deep RL approach, there still exist a lot of areas for improvement. First, the method should be used to generate other types of 3D virtual environments that differ from the manufacturing layouts used in the case study. Moreover, while the RL approach does not require the collection of data *a priori* since it takes advantage of simulation to train its model, the reward function, which impacts the action policy the RL agent models, can be challenging to design under certain conditions. Furthermore, it could be challenging under certain circumstances to create simulation environments that allow a RL agent to identify the desired action policy. More importantly, future work should explore how integrating the proposed PCG method

into learning applications can impact the motivation and learning of users. For example, as shown in Fig.5, this method can be used to generate new 3D virtual environments for immersive VR learning applications. However, the impact of automatically generating new content on users learning and engagement still has to be tested. Nevertheless, this work presents initial groundwork on integrating RL algorithms to automatically generate new content, which has significant implications for personalized and adaptive systems.

ACKNOWLEDGMENT

This research is funded by the National Science Foundation NSF DUE #1834465. Any opinions, findings, or conclusions found in this paper are those of the authors and do not necessarily reflect the views of the sponsors. The authors would also like to thank the hard work of Bradley Nulsen, Gerard Pugliese Jr., Adith Rai, and Matthew Rodgers on developing and implementing the application used in this work.

REFERENCES

[1] A. Summerville and M. Mateas, “Super mario as a string: Platformer level generation via lstms,” *arXiv Prepr.*, vol. arXiv:1603, 2016.

[2] Iglesias, A., Martínez, P., Aler, R., & Fernando, F. , “Applying reinforcement learning in intelligent tutoring systems,” in *Proc of international conference on new educational environments*, 2002, pp. 11–14.

[3] G. Fenza, F. Orciuoli, and D. G. Sampson, “Building Adaptive Tutoring Model Using Artificial Neural Networks and Reinforcement Learning,” in *Proceedings - IEEE 17th International Conference on Advanced Learning Technologies* , 2017.

[4] J. Togelius, G. N. Yannakakis, K. O. Stanley, and C. Browne, “Search-based procedural content generation: A taxonomy and survey,” in *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 3, no. 3, pp. 172-186, 2011.

[5] R. Bidarra, K. J. de Kraker, R. M. Smelik, and T. Tutenel, “Integrating semantics and procedural generation: key enabling factors for declarative modeling of

virtual worlds," in *FOCUS K3D Conference on Semantic 3D Media and Content*, 2010.

[6] G. Smith, J. Whitehead, and M. Mateas, "Tanagra: Reactive planning and constraint solving for mixed-initiative level design," in *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 3, no. 3, pp. 201-215, 2011.

[7] D. Hooshyar, M. Yousefi, and H. Lim, "A Procedural Content Generation-Based Framework for Educational Games: Toward a Tailored Data-Driven Game for Developing Early English Reading Skills," *J. Educ. Comput. Res.*, vol. 56, no. 2, pp. 293–310, 2018.

[8] K. Hullett and M. Mateas, "Scenario generation for emergency rescue training games," in *Proceedings of the 4th International Conference on Foundations of Digital Games - FDG '09*, 2009.

[9] R. Sawyer, J. Rowe, and J. Lester, "Balancing learning and engagement in game-based learning environments with multi-objective reinforcement learning.," in *International Conference on Artificial Intelligence in Education*, 2017, pp. 323–334.

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

[11] G. N. Yannakakis, "Game AI revisited," in *Proceedings of the 9th conference on Computing Frontiers - CF '12*, 2012.

[12] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Process. Mag.*, vol. 34, no. 6, pp. 26–38, 2017.

[13] C. E. Lopez, O. Ashour, and C. S. Tucker, "Reinforcement learning content generation for virtual reality applications," in *Int. Design Eng. Technical Conf. & Computers and Information in Eng. Conf.*, 2019.

[14] N. Shaker *et al.*, "The 2010 mario ai championship: Level generation track," *IEEE Trans. Comput. Intell. AI Games*, vol. 3, no. 4, pp. 332-347. 2011.

[15] P. Shi and K. Chen, "Learning Constructive Primitives for Real-time Dynamic Difficulty Adjustment in Super Mario Bros," *IEEE Trans. Comput. Intell. AI Games*, vol. 10, no. 2, pp. 155–169, 2018.

[16] M. Guzdial, N. Sturtevant, and B. Li, "Deep Static and Dynamic Level Analysis : A Study on Infinite Mario," in *AIIDE Workshop AAAI Technical Report WS-16-22*, 2016, pp. 31–38.

[17] A. Summerville, M. Behrooz, M. Mateas, and A. Jhala, "The learning of zelda: Datadriven learning of level topology," in *Proceedings of the FDG workshop on Procedural Content Generation in Games.*, 2015.

[18] N. Justesen, R. R. Torrado, P. Bontrager, A. Khalifa, J. Togelius, and S. Risi, "Illuminating generalization in deep reinforcement learning through procedural level generation," in *preprint arXiv*, 2018, p. arXiv :1806.10729.

[19] S. Dahlskog, J. Togelius, and M. J. Nelson, "Linear levels through n-grams," in *Proceedings of the 18th International Academic MindTrek Conference on Media Business, Management, Content & Services*, 2014.

[20] S. Snodgrass and S. Ontañón, “Generating Maps Using Markov Chains,” in *Artificial Intelligence and Game Aesthetics: Papers from the 2013 AIIDE Workshop*, 2013.

[21] N. Shaker and M. Abou-Zleikha, “Alone we can do so little, together we can do so much: A combinatorial approach for generating game content,” in *AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment.*, 2014.

[22] K. Almohammadi, H. Hagras, D. Alghazzawi, and G. Aldabbagh, “A survey of artificial intelligence techniques employed for adaptive educational systems within e-learning platforms,” *J. Artif. Intell. Soft Comput. Res.*, vol. 7, no. 1, pp. 47-64. 2017.

[23] R. A. Sottilare, “A hybrid machine learning approach to automated scenario generation (ASG) to support adaptive instruction in virtual simulations and games,” in *8th International Defense and Homeland Security Simulation Workshop* , 2018.

[24] F. A. Dorça, L. V. Lima, M. A. Fernandes, and C. R. Lopes, “Comparing strategies for modeling students learning styles through reinforcement learning in adaptive and intelligent educational systems: An experimental analysis,” *Expert Syst. Appl.*, vol. 40, pp. 2091–2101, 2013.

[25] A. Iglesias, P. Martínez, R. Aler, and F. Fernández, “Learning teaching strategies in an Adaptive and Intelligent Educational System through Reinforcement Learning,” *Appl. Intell.*, vol. 31, pp. 89–106, 2009.

[26] D. Hooshyar, M. Yousefi, M. Wang, and H. Lim, “A data-driven procedural-content-generation approach for educational games,” *J. Comput. Assist. Learn.*, vol. 34, no. 6, pp. 731–739, 2018.

[27] D. Hooshyar, M. Yousefi, and H. Lim, “A systematic review of data-driven approaches in player modeling of educational games,” *Artificial Intelligence Review*, vol. 52, no. 3, pp. 1–27, 2017.

[28] A. M. Smith, E. Andersen, M. Mateas, and Z. Popović, “A case study of expressively constrainable level design automation tools for a puzzle game,” in *Proceedings of the International Conf. on the Foundations of Digital Games*, 2012.

[29] L. Rodrigues, R. P. Bonidia, and J. D. Brancher, “A math educational computer game using procedural content generation,” in *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*, 2017, p. 756.

[30] C. Grappiolo, Y. G. Cheong, J. Togelius, R. Khaled, and G. N. Yannakakis, “Towards player adaptivity in a serious game for conflict resolution,” in *Proceedings - 2011 3rd International Conference on Games and Virtual Worlds for Serious Application* , 2011.

[31] M. L. Dering and C. S. Tucker, “Generative adversarial networks for increasing the veracity of big data,” in *Proceedings - 2017 IEEE International Conference on Big Data*, 2018, pp. 2595–2602.

[32] C. Beecks, M. S. Uysal, and T. Seidl, “Gradient-based signatures for big multimedia data,” in *IEEE International Conference on Big Data*, 2015, pp. 2834–2835.

[33] C. E. Lopez, S. R. Miller, and C. S. Tucker., “Exploring Biases Between Human and

Machine Generated Designs," *J. Mech. Des.*, vol. 2, no. 141, 2019.
[34] C. Lopez, S. R. Miller, and C. S. Tucker, "Human validation of computer vs human generated design sketches," in *ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, 2018,
[35] Y. Chen, S. Tu, Y. Yi, and L. Xu, "Sketch-pix2seq: a Model to Generate Sketches of Multiple Categories," *arXiv Prepr. arXiv1709.04121.*, 2017.
[36] K. LP, L. ML, and M. AW, "Reinforcement learning : a survey," *Int J Artif Intell Res*, vol. 4, pp. 237–285, 1996.
[37] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, 1998.
[38] X. Xu, L. Zuo, and Z. Huang, "Reinforcement learning algorithms with function approximation: Recent advances and applications," *Inf. Sci. (Ny).*, vol. 261, pp. 1–31, 2014.
[39] V. Mnih *et al.*, "Playing atari with deep reinforcement learning," *arXiv Prepr.*, vol. arXiv:1312, 2013.
[40] D. Silver *et al.*, "Mastering the game of Go without human knowledge," *Nature*, vol. 550, no. 7676, p. 354, 2017.
[41] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529-533, 2015.
[42] M. Johnson, K. Hofmann, T. Hutton, and D. Bignell, "The malmo platform for artificial intelligence experimentation," in *IJCAI International Joint Conference on Artificial Intelligence*, 2016, pp. 4246–4247.
[43] E. Todorov, T. Erez, and Y. Tassa, "MuJoCo: A physics engine for model-based control," in *IEEE International Conference on Intelligent Robots and Systems*, 2012, pp. 5026–5033.
[44] A. Juliani *et al.*, "Unity: A General Platform for Intelligent Agents," *arXiv Prepr.*, vol. arXiv:1809, 2018.
[45] P. Wang, J. Rowe, W. Min, B. Mott, and J. Lester, "Interactive narrative personalization with deep reinforcement learning," in *IJCAI International Joint Conference on Artificial Intelligence*, 2017.
[46] J. Rowe, A. Smith, R. Pokorny, B. Mott, and J. Lester, "Toward Automated Scenario Generation with Deep Reinforcement Learning in GIFT.," in *Proceedings of the Sixth Annual GIFT Users Symposium*, 2018, p. 65.
[47] H. Hosseini-Nasab, S. Fereidouni, S. M. T. Fatemi Ghomi, and M. B. Fakhrzad, "Classification of facility layout problems: a review study," *Int. J. Adv. Manuf. Technol.*, vol. 94, pp. 957–977, 2018.
[48] P. M. Pardalos, D. Z. Du, and R. L. Graham, *Handbook of Combinatorial Optimization*. 2013.
[49] A. Drira, H. Pierreval, and S. Hajri-Gabouj, "Facility layout problems: A survey," *Annu. Rev. Control*, vol. 31, no. 2, pp. 255–267, 2007.
[50] N. Lotfi and A. Acan, "Learning-based multi-agent system for solving combinatorial optimization problems: A new architecture.," in *Proceedings of the 10th international conference Hybrid artificial intelligent systems*, 2015.
[51] S. Martin, D. Ouelhadj, P. Beullens, E. Ozcan, A. A. Juan, and E. K. Burke, "A multi-

agent based cooperative approach to scheduling and routing," *Eur. J. Oper. Res*., vol. 254, no. 1, pp. 169-178, 2016.

[52] H. Samma, C. P. Lim, and J. Mohamad Saleh, "A new Reinforcement Learning-based Memetic Particle Swarm Optimizer," *Appl. Soft Comput. J.*, vol. 43, pp. 276-297, 2016.

[53] M. A. L. Silva, S. R. De Souza, M. J. F. Souza, and S. M. De Oliveira, "A multi-agent metaheuristic optimization framework with cooperation," in *Proceedings - 2015 Brazilian Conference on Intelligent Systems* , 2015.

[54] M. E. Aydin and E. Oztemel, "Dynamic job-shop scheduling using reinforcement learning agents," *Rob. Auton. Syst.*, vol. 33, pp. 169–178, 2000.

[55] Y. C. Wang and J. M. Usher, "Application of reinforcement learning for agent-based production scheduling," *Eng. Appl. Artif. Intell.*, vol.18, no.1, pp. 73-82, 2005.

[56] M. A. L. Silva, S. R. de Souza, M. J. F. Souza, and A. L. C. Bazzan, "A reinforcement learning-based multi-agent framework applied for solving routing and scheduling problems.," *Expert Syst. Appl.*, no. 131, pp. 148–171, 2019.

[57] J. Shahrabi, M. A. Adibi, and M. Mahootchi, "A reinforcement learning approach to parameter estimation in dynamic job shop scheduling," *Comput. Ind. Eng.*, vol. 110, pp. 75-82, 2017.

[58] M. Nazari, A. Oroojlooy, M. Takáč, and L. V. Snyder, "Reinforcement learning for solving the vehicle routing problem," in *Advances in Neural Information Processing Syste*ms, pp. 9839-9849 , 2018.

[59] S. Govindaiah and M. D. Petty, "Applying reinforcement learning to plan manufacturing material handling Part 2: Experimentation and results," in *ACMSE 2019 - Proceedings of the 2019 ACM Southeast Conference*, 2019.

[60] S. Govindaiah and M. D. Pey, "Applying reinforcement learning to plan manufacturing material handling Part 1: Background and formal problem specification," in *ACMSE 2019 - Proceedings of the 2019 ACM Southeast Conference*, 2019.

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

[62] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel, "Trust Region Policy Optimization," *arXiv.org* 1502.05477, 2017.

[63] C. Lopez, O. Ashour, and C. Tucker, "An introduction to CLICK: Leveraging Virtual Reality to Integrate the Industrial Engineering Curriculum," in *ASEE Annual Conference & Exposition*, 2019, pp. 1–12.

[64] J. Terpenny *et al.*, "Product-based Learning: Bundling Goods and Services for an Integrated Context-rich Industrial Engineering Curriculum," in *Annual Conference of the American Society for Engineering Education (ASEE)*, 2018.

[65] W. G. & C. Pope, "Unity_Game_Engine," *UNITY GAME ENGINE Overv.*, 2011.

[66] P. Petridis, I. Dunwell, S. De Freitas, and D. Panzoli, "An engine selection methodology for high fidelity serious games," in *2nd International Conference on Games and Virtual Worlds for Serious Applications*, 2010.

[67] A. Alsubaie, M. Alaithan, M. Boubaid, and N. Zaman, "Making learning fun:

Educational concepts & logics through game," in *International Conference on Advanced Communication Technology, ICACT*, 2018.

[68] J. Cunningham and C. S. Tucker, "A Validation Neural Network (VNN) metamodel for predicting the performance of deep generative designs," in *Proc. ASME Int. Des. Eng. Tech. Conf. Comput. Inf. Eng. Conf.*, 2018.

[69] Y. Burda, H. Edwards, D. Pathak, A. Storkey, T. Darrell, and A. A. Efros., "Large-scale study of curiosity-driven learning," *arXiv Prepr.*, arXiv: 1808.04355, 2018.

FIGURE CAPTIONS LIST

Fig. 1 **Outline of the Reinforcement Learning PCG method**

Fig. 2 **Reinforcement Learning framework representation**

Fig. 3 **User's point of view of a functional manufacturing layout**

Fig. 4 **Reinforcement Learning agent rewards score vs. training iterations**

Fig. 5 **Example of new manufacturing layouts generated by the trained RL agent.**

Fig. 6 **Users interacting with the generated virtual environment using an immersive VR headset**

TABLE CAPTION LIST

Table 1. Summary of related work

Reference	*Meta-Heuristics*	*Supervised ML*	*RL*	*Environment generation*	*Application Context*
[7], [8], [28], [29]	**X**				*Learning*
[9], [26], [30], [45]		**X**			*Learning*
[1], [14]–[18]		**X**		**X**	*Games*
This work			**X**	**X**	*Learning/Games*

Table 2. Summary of environments generated to evaluate the trained model

Reward	*No.*	*Percentage*	*Comments*
21	45	8.79%	Optimal Layout
19	237	46.29%	Mismatched speed only
11	25	4.88%	No robot or conveyor interaction, but matched speed and all parts in the tote
9	128	25.00%	No robot or conveyor interaction, mismatched speed, but all parts in the tote
1	11	2.15%	Robot and conveyor interaction, matched speed, but no parts in the tote
-1	52	10.16%	Robot and conveyor interaction but no parts in bin, mismatched speed
-4	3	0.59%	Conveyor interaction and matched speed, but no robot interaction, nor parts in bin,
-6	9	1.76%	Conveyor interaction but no robot interaction, no parts in tote, and mismatched speed
-11	2	0.39%	No conveyor or robot interaction, mismatched speed,and no parts in the tote