Designing Human-Autonomy Teaming Experiments Through Reinforcement Learning

Beau Schelble
Human-Centered Computing
Clemson University
821 McMillan Rd, Clemson, SC 29631
bschelb@g.clemson.edu

Lorenzo-Barberis Canonico
Human-Centered Computing
Clemson University
821 McMillan Rd, Clemson, SC 29631
lorenzo@g.clemson.edu

Nathan McNeese
Human-Centered Computing
Clemson University
821 McMillan Rd, Clemson, SC 29631
mcneese@g.clemson.edu

Jack Carroll
Human-Centered Computing
Clemson University
821 McMillan Rd, Clemson, SC 29631
jcarro5@g.clemson.edu

Casey Hird
Human-Centered Computing
Clemson University
821 McMillan Rd, Clemson, SC 29631
crhird@g.clemson.edu

This paper creates and defines a framework for building and implementing human-autonomy teaming experiments that enable the utilization of modern reinforcement learning models. These models are used to train artificial agents to then interact alongside humans in a human-autonomy team. The framework was synthesized from experience gained redesigning a previously known and validated team task simulation environment known as NeoCITIES. Through this redesign, several important high-level distinctions were made that regarded both the artificial agent and the task simulation itself. The distinctions within the framework include gamification, access to high-performance computing, a proper reward function, an appropriate team task simulation, and customizability. This framework enables researchers to create experiments that are more usable for the human and more closely resemble real-world human-autonomy interactions. The framework also allows researchers to create veritable and robust experimental platforms meant to study human-autonomy teaming for years to come.

INTRODUCTION

Research on human-autonomy teams (HATs) has been on the rise for the past two decades. Many of the studies conducted during this time were able to produce valuable results using the Wizard of Oz (WoZ) methodology, which has a human simulate the role of the AI in the HAT (Maulsby et al., 1993). The usage of WoZ continues today as the use of true AI in HAT research is not a commonplace practice as many studies continue to positively utilize the WoZ methodology (Chen et al., 2017; Fu & Zhou, 2020). This trend is a consequence of the lack of computational power and proper artificial intelligence (AI) development architectures. However, with the advent of cloud services for high-performance computing (HPC) and the release of accessible AI frameworks like TensorFlow and TensorForce, the development and implementation of custom AI for localized simulations is readily achievable (Abadi et al., 2016; Schaarschmidt et al., 2017). With these new tools, researchers are now able to create custom AI for HAT research, significantly improving the validity and generalizability of their results to genuine HATs. This paper creates a framework to showcase how best to leverage these technologies to create actual AI team members by using reinforcement learning (RL) models and creating compliant simulated tasks. The redesign provides an overview of creating the simulated task and agent, with the framework being synthesized from that redesign. By leveraging the technology outlined in this paper, the HAT field can benefit from enhanced validity, generalizability, and the potential for newfound interactions.

Team task simulations have long been used for experiments in the human-human teaming (HHT) literature (Gupta & Woolley, 2018; Smith-Jentsch, 2007). These simulations were meant to create a variety of different task situations and have produced a variety of meaningful results. A strong example of such a simulation, that has long persisted and remains scalable to this day is NeoCITIES. Created as an innovative advancement to Wellens and Ergenr's original CITIES task (Wellens & Ergener, 1988), NeoCITIES serves to provide a simulation for the study of teamwork and decision-making within a command control, and computer-mediated communication (C3) environment (McNeese et al., 2005).

HAT literature has historically utilized simulated team tasks, much like the HHT literature has. Simulated tasks for HATs have ranged from rearranging boxes around an environment with a team to going through an entire UAV mission simulation (Johnson et al., 2009; McNeese et al., 2018). However, as previously stated, the use of true AI is not common in much of this HAT simulated task research (Chen et al., 2017; Fu & Zhou, 2020).

With the technology and framework outlined here in this paper, researchers can implement true AI into their HAT experiments. With this change, there is a great deal of potential for the discovery of new interactions between humans and artificial agents. This paper creates a framework based on lessons learned from an actual redesign of a validated team-based simulation (NeoCITIES) that enabled true AI integration. The framework outlines the core and sub-concepts that researchers can follow to create their task simulations that utilize true AI as well as the differences of HATs as compared

to HHTs. Through the outline of the task simulation redesign, we give a clear example of what goes into such a redesign or build, then pull the core concepts that are required to integrate true RL AI into HAT experiments to form the framework. The ability of the framework to scale, maintain customizability, and have more extensive applications beyond only HAT specific work is then discussed.

NEOCITIES REDESIGN

The culmination of years of enhancements has made NeoCITIES a modular and reliable simulated environment to study teamwork. This fundamental resource allocation task setup has historically led to multiple findings in information science, psychology, and geographical sciences (McNeese et al., 2014). Furthermore, through the repeated use of established measures and metrics, NeoCITIES enables researchers to investigate specific aspects of team cognition and other critical factors in teamwork (Mohammad et al., 2010).

Recently, NeoCITIES was redesigned to integrate RL AI into the overall teamwork simulation. The NeoCITIES redesign is covered in extensive detail in order to potentially aid other researchers in the development of their simulated tasks that support AI integration. These specific details of the redesign empower other individuals to extract the details and lessons learned from this redesign and apply them to their platforms. This redesign also provides context to the framework and its generalizability, with each component stemming from the redesign.

The NeoCITIES simulation has participants respond to a series of emergency events with local government resources, with each event occurring around a fictitious college town (McNeese et al., 2014). The simulation requires a team of three actors, each operating a different government department (police, fire, and hazardous material response). Participants are tasked with assessing each event and appropriately allocating scarce resources. Here, the prior version of NeoCITIES is compared and contrasted to the redesigned version, which consisted of inclusions and advancements to the backend architecture, interface, and AI development & training.

Backend Architecture

Concerning its backend architecture, the prior version of NeoCITIES used a client-server model to display information generated by a simulation engine developed with Java and Adobe technologies. It was a multi-threaded application that handles the event and resource data, maintains the state of the world, and calculates event scores (Hellar & McNeese, 2010). A downside of this design was that it did not support multiple simulations simultaneously, and thus limited researchers to running one simulation at a time in their local environment. A cloud-based design was implemented to rectify this problem, and updates are pushed out to a cloud-based server (Firebase by Google) by clients (participants) on each change in the global state. All clients then see these changes, which are automatically made in real-time via the global state, which each client is listening to through a REST API. Naturally, any time changes are made to the server, the state of the simulation is updated, and each clients' local machine reflects those changes in real-time.

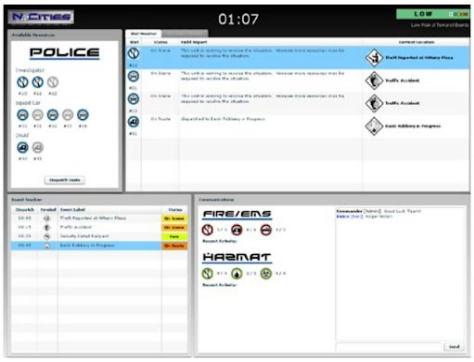

Figure 1. Prior NeoCITIES Design

The Firebase server supports a vast amount of simultaneous sessions because each session is managed by a child in the database's JSON tree, preventing the clients from accidentally updating the wrong session. The last significant way in which the new architecture advances NeoCITIES is by creating an API layer that enables AI agents to interact with the human participants easily, thereby enabling the platform to support HATs for the first time. Specifically, the AI agents can request the state of the simulation through the REST API on Firebase and push actions to the database. This action enables the agents to participate in the simulations in an asynchronous manner similar to human players. Overall, the new architecture is designed to be very modular and thus supports a variety of scenarios with different roles, resources, and the number of players, all controlled by parameters within the JSON structure.

Frontend & Interfaces

The next significant aspect of NeoCITIES consisted of updating the frontend interface. The redesign of the interface sought to preserve many vital features (drag & drop resource allocation, recall, chat, event briefs) seen in Figure 1, while also altering it to more effectively support AI agents & HAT. To start with, Figure 2 shows the default view of the human participants' interface in the redesign. The left side of the UI displays the resources available to the user, while the right side houses the chat suite, which implements a feature that automatically scrolls to the bottom of the chat with each new message. The middle contains the tasks and events currently active, each represented with a card containing various information like required resources, and time remaining. A far more significant change, however, is the implementation of a map locating events and resources. This redesign of NeoCITIES includes a map specifically because AI agents perceive the world through a matrix of numbers that give them a view of the task in its entirety. Meaning that if a map were not included, the autonomous agents would perceive the simulated task in an entirely different manner than their human teammates, with full spatial awareness. This difference would then prevent any accurate or meaningful direct comparisons between HHTs and HATs.

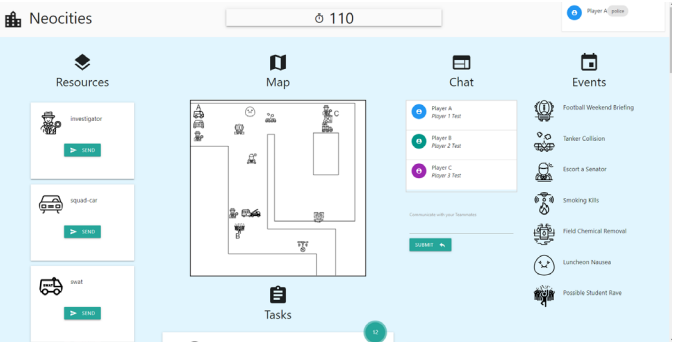

Figure 2. Updated NeoCITIES Design

Development of AI Agents and Training

For the first time, NeoCITIES supports AI agents. The agents are based on RL, which creates its model by making an action based on a state and then receiving a reward. This learning model is a framework within AI training/creation and has received acclaim for superhuman performance in games such as Go, and various Atari games (Hu et al., 1998, p. 199; Mnih et al., 2013). Utilizing RL enables NeoCITIES, and any platform leveraging RL, to access and use some of the most advanced AI algorithms available. RL also allows the artificial agents to work in a variety of different parameters within the simulation it is trained to accomplish. The RL agents used for this redesign were created with TensorForce, which is an open-source RL framework with the goals of delivering usable APIs, code readability, and powerful modularization to develop RL solutions in a variety of applications (Schaarschmidt et al., 2017b, 2018). Part of creating RL agents involves developing a task-specific training environment. The training environment must replicate the specific task mathematically using the Python scripting language. The training environment allows each agent to learn the task through self-play, which in the NeoCITIES redesign, was as part of a three agent team with each agent taking on one of the three available roles within the simulation. More specifically, to the NeoCITIES task, the training environment conveys NeoCITIES to the RL agent as a series of matrices that represent the state of the game as it changes in real-time. This representation of the game in matrices is a vital point to consider in designing these experiments as well as in applied settings as humans and autonomous agents perceive the environments they exist within in radically different ways (the autonomous agent understands through linear algebra representations, while the human perceives using biological mechanisms). In practice, once trained, the environment the agent receives can be slightly tweaked for experiment customizability.

During training, the state of the environment is sent to the agent as a 50x50 matrix containing data on various simulation details like ongoing events, coordinates, & available resources. After receiving the state, the agent responds with an action (which resource to send to which event). This action is then received and executed by the training simulation, with all relevant simulation factors being updated. This new state is then used to determine the reward for the agent, which is based on how much closer necessary resources are to their corresponding event as compared to the previous state. Training the agents takes a considerable amount of time and compute power (CPU compute) as the game must be simulated many times over. Agent models should be saved at regular intervals (5000 iterations for this redesign) and continue until reaching a predetermined number of iterations (200000 for this redesign). The total number of training iterations will increase with things like complex tasks or inadequate reward functions. Once the agent was trained for the NeoCITIES redesign, an API was developed that enabled the trained agent to communicate with the NeoCITIES simulation in much the same way that it communicated with the training environment. This API allows for seamless integration with the user-facing NeoCITIES simulation and thus enables full AI integration. A further potential step in AI training may serve to make the AI more responsive to human actions by logging decisions and actions of trials played by humans and using those trials as part of the simulation the AI uses in training, exposing it to human decision making.

FRAMEWORK FOR HAT EXPERIMENTAL PLATFORMS

As previously mentioned, this framework shown below in Figure 3, is created from the various lessons learned in a redesign of NeoCITIES to support genuine RL AI agents.

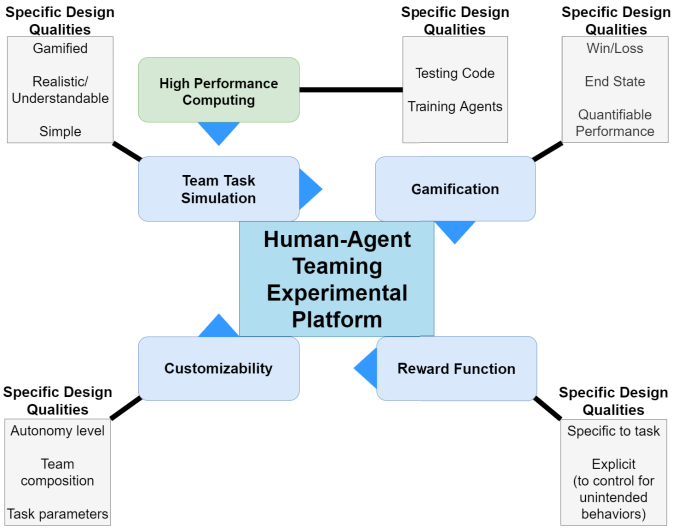

Figure 3. HAT Experiment Platform Framework

The framework consists of various specific design qualities that come together to create the significant core components of the framework. These relationships and distinctions are displayed in the framework with high-performance computing being shown as an enabling factor (green) contributing to the team task simulation, which serves as one of the four core components of the framework. Team task simulation then determines how the simulation will be gamified, which in turn determines how the reward function is designed, and finally customizability of the simulation. Each of the four core components of the framework is interrelated and dependent upon each other, demonstrated in the specific design qualities that each factor within the framework displays, such as the reward function being specific to the task and dependent upon

the team task simulation. This framework conveys the explicit guidelines, standards, and techniques that allow others to complete related projects that provide valuable research outcomes and leverage the potential differences seen in HATs using true AI.

Gamification. A significant feature of RL agents lies in the simulation of the task, which often revolves around learning a game (Mnih et al., 2013; Sun et al., 2019). These agents require a reward function in order to train their algorithms correctly, which means that any task an individual creates must center around a gamified component. There must be a win and loss metric, an ending, and good or bad performance that can be quantified. The fundamental nature of RL makes this component necessary and is also required as the task must be simulated programmatically, using matrices of numbers.

High-performance computing. The need for access to high-performance computing (HPC) was an enormous lesson learned for this project as training agents with 200,000 iterations of the simulated task is very computationally taxing. While these processes can be run and completed on a typical laptop, they take multiple days to complete, and the machine is mostly unusable during that period. Access to HPC machines exists at some universities, but those universities are in the minority. At this point, the benefits of modern cloud services step in with Amazon, IBM, and others offering HPC solutions at scaling costs (High-Performance Computing on IBM Cloud, 2019; HPC at Scale | Amazon Web Services, 2019). These services allow users to test their simulations with the agent on their local machine for free and then send the agent off to train in the HPC cloud service.

Reward function. The reward function serves to indicate to the agent how successful or appropriate their decision was. The reward function must be written to ensure that the agent does not begin to take advantage of the game, as their base function is to maximize the received reward (Schulman et al., 2017). For example, in early iterations of the NeoCITIES agents' reward function, agents never fully completed the event. Agents would have the resource remain close to the event to maximize their score artificially without truly completing the task. This issue was resolved by implementing a special reward once an event was completed to disincentivize this type of action.

Additionally, agents are known to enter a relative state of death or self-extinction under the wrong reward conditions (Martin et al., 2016). Such conditions may occur when the agent is unable to find any positive reward and will make as many actions as it takes to end the current iteration of the game as quickly as it can to minimize the negative rewards it receives. Thus, it is essential to be thoughtful in writing a reward function, ensuring it is advised by these examples and allows the agent to be successfully trained in a reasonable amount of time. Any reward function is also highly dependent upon the gamification of the task simulation, as, without proper gamification, the reward function will be incredibly difficult, if not impossible, to write.

Team task simulation. Creating a simulation that follows the previously mentioned gamification principle while retaining a level of realism or graspable metaphor is vital in HAT, which includes making the task as simple as possible within the bounds of answering the research question. Ensuring that the simulated task meets these tenants allows experimenters to meet the core principles of a team, retain ecological validity, and have participants easily understand the task. Additionally, these types of simulations allow experimenters to easily replicate the game in a mathematical environment for AI integration. Utilizing a realistic and or understandable metaphor for the task simulation enables reliable data that retains ecological validity.

Additionally, with an easily understood game, there is a reduced need for participant training and a reduction in errors. A simple game also translates to reduced time spent coding task simulations, as well as a reduction in time spent training the agent. Examples of such tasks meeting these tenants and utilizing artificial agents are Blocks World for Teams (Johnson et al., 2009), and a treasure-hunting task (Luo et al., 2019). These tasks serve as straightforward examples of simulations that are not too complex for participants, retain realism/graspable metaphor, meet the criteria for a team, and integrate gamification, allowing the integration of true AI for HAT experiments. Gamification is also very interdependent with the team task simulation, and as for the task design, it is entirely dependent upon how the researcher chooses to gamify that task. However, the reward function is entirely dependent upon both of these core concepts, making it a consequence of the decisions made regarding the team task simulation, and the way the task was gamified.

Customizability. The task simulation must also integrate a level of customizability in several key areas to enhance its role as an experimental simulation. The level of autonomy should be customizable based on the role the AI plays within the game. Team composition should also be customizable in order to study different compositions of HATs, on any number of dependent variables. Finally, the task simulation itself must be customizable based on different scenarios. As previously stated, the NeoCITIES scenario accomplishes this through a simple JSON tree within Firebase specifying different lengths of time, number of emergencies, and the resources required for each emergency. This customizability allows for future scalability in the experimental platform for future applications. With high levels of customizability along each of these facets, any task simulation can benefit from a long life span and produce a considerable amount of meaningful results.

DISCUSSION OF FRAMEWORK & WIDER APPLICATIONS

The NeoCITIES redesign gives rise to a framework that engenders several things to the HAT research community. The first being the ability to integrate AI and study human-autonomy teaming in greater detail and with increased validity. The ability to implement UI design changes that accommodate for differences in the way humans and AI perceive the world. Emphasizing the scalability of research platforms to ensure long life and high research output, as well as providing a framework that can extend and generalize to other applications and fields. These contributions are discussed at length through the addition of genuine AI agents, changes in design to

accommodate humans in HATs, and its potential impact on other fields through its scalability.

Addition of AI agents. The redesigned NeoCITIES presents an unprecedented opportunity to study interactions of humans and AI agents. The framework supports and encourages essential components of AI integration, highlighting the specific needs of the AI. These include a task framed as a game, a proper reward function that incentivizes the agents to perform the game in the desired way, and HPC to train the agent within a reasonable amount of time. Thus, with the added ability to evaluate teams working with real AI, researchers can then ask questions that address the quality of interactions in a team and how to best leverage the potential differences between HATs and HHTs.

Human design elements with AI. The newly redesigned NeoCITIES also presents the advantage of more modern and informed design. The updates to the chat interface and the implementation of the map make the NeoCITIES interface much more familiar for modern users. These changes allow for studies that more accurately reflect the current use of technology and the way that humans and autonomous agents interact. This modern UI goes hand in hand with the realism of the task and or having a simple metaphor, enabling the simulation to be readily understood by human and autonomous participants alike.

Broader impacts on other fields. The framework can also be extended to other fields, such as industrial-organizational psychology research, which can incorporate AI agents into organizational teams to investigate performance in uncertain situations. The framework has applications even to less obvious fields like cell biology, which may consider a team task simulation to frame how microorganisms interact or use the concept of a reward function to understand, or perhaps replicate the incentives of cells. This is accomplished with the customizability emphasized by the framework allowing many fields to benefit incorporating AI agents.

CONCLUSION

HAT, while similar to HHT, is still fundamentally different in a variety of ways, and there is not a one to one fit between the two. The design and application differences in HATs need to be considered in the simulations that will form the basis of future research. Differences in team cognition, performance, situational awareness, and processing are a few that must be considered in future research. The framework allows such differences to be recognized and even leveraged in experimental research. This quality makes the framework a valuable tool to use as a springboard for future HAT research, helping to continue driving the field forward.

ACKNOWLEDGMENTS

This research is funded under NSF Award #1829008.

REFERENCES

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., & Isard, M. (2016). Tensorflow: A system for large-scale machine learning. *12th ${$USENIX$}$ Symposium on Operating Systems Design and Implementation (${$OSDI$}$ 16)*, 265–283.

Chen, J. Y., Barnes, M. J., Wright, J. L., Stowers, K., & Lakhmani, S. G. (2017). Situation awareness-based agent transparency for human-autonomy teaming effectiveness. *Micro-and Nanotechnology Sensors, Systems, and Applications IX*, *10194*, 101941V.

Fu, Z., & Zhou, Y. (2020). Research on human–AI co-creation based on reflective design practice. *CCF Transactions on Pervasive Computing and Interaction*, 1–9.

Gupta, P., & Woolley, A. W. (2018). Productivity in an era of multi-teaming: The role of information dashboards and shared cognition in team performance. *Proceedings of the ACM on Human-Computer Interaction*, *2*(CSCW), 1–18.

Hellar, D. B., & McNeese, M. (2010). NeoCITIES: A simulated command and control task environment for experimental research. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, *54*, 1027–1031.

High-Performance Computing on IBM Cloud. (2019, May 29). https://www.ibm.com/cloud/hpc

HPC at Scale | Amazon Web Services. (2019, May 29). Amazon Web Services, Inc. https://aws.amazon.com/hpc/campaigns/hpc-at-scale/

Hu, J., Wellman, M. P., & others. (1998). Multiagent reinforcement learning: Theoretical framework and an algorithm. *ICML*, *98*, 242–250.

Johnson, M., Jonker, C., Van Riemsdijk, B., Feltovich, P. J., & Bradshaw, J. M. (2009). Joint activity testbed: Blocks world for teams (BW4T). *International Workshop on Engineering Societies in the Agents World*, 254–256.

Luo, R., Du, N., Huang, K. Y., & Yang, X. J. (2019). Enhancing Transparency in Human-autonomy Teaming via the Option-centric Rationale Display. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, *63*, 166–167.

Martin, J., Everitt, T., & Hutter, M. (2016). Death and suicide in universal artificial intelligence. *International Conference on Artificial General Intelligence*, 23–32.

McNeese, M. D., Bains, P., Brewer, I., Brown, C., Connors, E. S., Jefferson Jr, T., Jones Jr, R. E., & Terrell Jr, I. (2005). The NeoCITIES simulation: Understanding the design and experimental methodology used to develop a team emergency management simulation. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, *49*, 591–594.

McNeese, M. D., Mancuso, V. F., McNeese, N. J., Endsley, T., & Forster, P. (2014). An integrative simulation to study team cognition in emergency crisis management. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, *58*, 285–289.

McNeese, N. J., Demir, M., Cooke, N. J., & Myers, C. (2018). Teaming with a synthetic teammate: Insights into human-autonomy teaming. *Human Factors*, *60*(2), 262–273.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *ArXiv Preprint ArXiv:1312.5602*.

Schaarschmidt, M., Kuhnle, A., Ellis, B., Fricke, K., Gessert, F., & Yoneki, E. (2018). Lift: Reinforcement learning in computer systems by learning from demonstrations. *ArXiv Preprint ArXiv:1808.07903*.

Schaarschmidt, M., Kuhnle, A., & Fricke, K. (2017a). TensorForce: A TensorFlow library for applied reinforcement learning. *Web Page*.

Schaarschmidt, M., Kuhnle, A., & Fricke, K. (2017b). TensorForce: A TensorFlow library for applied reinforcement learning. *Web Page*.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. *ArXiv Preprint ArXiv:1707.06347*.

Smith-Jentsch, K. A. (2007). The impact of making targeted dimensions transparent on relations with typical performance predictors. *Human Performance*, *20*(3), 187–203.

Sun, Y., Khan, A., Yang, K., Feng, J., & Liu, S. (2019). Playing First-Person-Shooter Games with A3C-Anticipator Network Based Agents Using Reinforcement Learning. *International Conference on Artificial Intelligence and Security*, 463–475.

Wellens, A. R., & Ergener, D. (1988). The CITIES game: A computer-based situation assessment task for studying distributed decision making. *Simulation & Games*, *19*(3), 304–327.