

## **Designing Immersive Learning Experiences: Teaching Robotics Programming in Virtual 3D Environments**

*By Carsten Lecon<sup>\*</sup>, Stefanie Soeser<sup>‡</sup> & Andre Stegmaier<sup>°</sup>*

This article presents a teaching scenario using virtual environments, often referred to as part of the ‘metaverse’, to address educational challenges highlighted during the COVID-19 pandemic. The increasing shift to online learning environments emphasizes the need for flexible, anytime-anywhere access to education. Virtual 3D environments offer a level of immersion that surpasses traditional online formats, such as video conferencing, and enable innovative didactic concepts, including collaborative group activities. To illustrate these benefits, we showcase a case study involving the simulation of robot programming within a virtual 3D environment. Additionally, the integration of personal assistants, such as conversational or pedagogical agents, is explored as a means to alleviate teachers' workloads. Advances in artificial intelligence have made natural language interactions with these agents highly effective, further enhancing the potential of virtual 3D environments in educational contexts.

*Keywords:* e-learning, e-teaching, immersion, virtual 3D rooms, gamification, cyber-physical systems

### **Introduction**

The COVID-19 pandemic catalyzed a widespread transition from traditional classroom-based teaching to online platforms, such as video conferencing systems. However, challenges persist even beyond the immediate effects. Current crises, including significant increases in energy costs, have added barriers to attending physical training locations, particularly for external training sessions. At the same time, remote work and learning have become integral to modern educational and professional practices. To address these evolving demands, virtual 3D environments present a compelling option for creating authentic and engaging learning experiences. This article explores the unique characteristics of virtual 3D spaces and their application in higher education and adult learning settings.

During the pandemic, limited preparation time required the rapid adoption of online tools, primarily video conferencing systems, to support synchronous and asynchronous learning. Although effective for continuity, these tools often lack the immersive qualities necessary to foster a robust sense of presence. Virtual 3D environments offer a remedy by enabling participants to interact as avatars ("alter egos") within richly designed spaces conducive to learning. This paper elaborates on the attributes of such environments and demonstrates their potential through concrete educational scenarios.

---

<sup>\*</sup>Professor of Computer Science, Aalen University, Germany.

<sup>‡</sup>Research Assistant, Aalen University, Germany.

<sup>°</sup>Research Assistant, Aalen University, Germany.

The Pedagogical and Technological Framework provides an overview of the fundamental features of 3D virtual spaces. The following sections details a specific learning scenario developed for advanced computer science students, showcasing its integration into existing e-learning infrastructures (Initial phase: Testing with TurtleSim). The scenario emphasizes collaborative group work, with key considerations described in Main Scenario: Autonomous Navigation in a 3D Obstacle Course. Furthermore, the inclusion of pedagogical or conversational agents as virtual assistants is discussed in Next steps. These agents play critical roles, from onboarding learners into the virtual environment to facilitating the achievement of educational objectives. Their functionality extends to assessing and supporting varying levels of competency, as elaborated in the subsequent section. Preliminary results are then presented, followed by the conclusion and an outline of future research directions.

This paper describes ongoing scenarios currently under implementation, with evaluation processes still in progress.

### Virtual 3D Rooms

Immersion plays an important role in a virtual learning environment, that is, ‘immersing’ or ‘sinking’ in an artificially generated environment. Studies have shown that immersion in virtual environments has a major impact on learning efficiency (see, for example, (Pagano, 2013)). This is achieved, for example, by allowing multiple perspectives, situated learning, and transfer (Dede, 2009). Meanwhile, proven immersion environments are virtual 3D environments. (Churchill & Snowdown, 1998) argue that there is a ‘paradigm change’ because such virtual environments ‘provide a space that contains or encompasses data representations and users. Virtual 3D room support supports the fact that the key aspect of many pedagogical methods and theories involves how students learn through observation, imitation, and interactions with other students and the instructor (Beck, 2019).

Regarding the virtual user experience, there are two aspects (Girvan, 2018): sense of presence (“being in a virtual room”) and ‘Shaped by its inhabitants’ (the user ‘inhabitant’ can become a content consumer and a content producer). The user experience is also influenced by technical features (Girvan, 2018): avatars, multiple concurrent users, content creation tools, persistence, and space representation. The interaction of an avatar is significantly influenced by the so-called “embodiment”, e.g. by appearance, gesture options, facial expression, etc. (Churchill & Snowdown, 1998) However, social proximity (Borkovic, Skovira & Kohun, 2021), that is, proximity to someone opposite, can also play a role in a virtual environment. “Walking through” a person is not usually possible, but if one avatar encounters another on their way (for example, if the user in question is inexperienced in controlling his/her avatar) and then automatically stands in front of you, this can lead to a feeling of distress. Since the virtual 3D world is basically just a piece of software, countermeasures can be taken if necessary, for example, by placing a “ban mile” around the avatar in order to maintain the social distance (Borkovic, et al., 2021; Figure 1); this is 1,2 to 2.1 metres.

Figure 1. Proximity (generated in part by AI)



Intime distance, personal distance, social distance

Virtual 3D environments are advantageous in many ways in contrast to, for example, often used 2D video conferences. Simple conversational situations are perceived as more natural than in 2D environments (Garau, et al., 2023). The virtual environment can be comfortable and friendly, so that this environment can trigger relaxation, silence, and pleasure in the user. Induced positive emotions can be a decisive factor for collaborative work and self-regulated learning, which is also reinforced by intensive flow and presence experience (Boekaerts, 1996). In general, for collaborative learning and working, spatial and social immersion are central characteristics of these environments, as well as appropriate usability. Spatial immersion means feeling as being in another place as in reality (Boekaerts, 1996; Zinn, et al., 2016). In (Lai, 2011) it is postulated that the effect of collaborative learning depends on the quality of the interaction. With distributed learning, virtual 3D environments offer better possibilities than, for example, video conferences or asynchronous media such as Wikis or chat groups.

In principle, virtual worlds promote the activity of participants, since the avatar reduces risk perception, especially for shy and insecure students (Lueckemeyer, 2015).

It makes sense to offer the possibility to place (interactive) 3D objects in such a 3D environment, which can also be interacted with in a protected, safe environment (Mantovani & Castelnovo, 2003). In this way, digital twins can also be integrated into the 3D environment (Qi, et al., 2019). The size of objects can also be experienced particularly intensively in a virtual 3D environment.

"Second Life" can be seen as a quasi-pioneer of virtual 3D worlds; this environment is still used in some cases, for example (Dede, 2009) in the context of 'self-efficacy'. In the meantime, there are now many commercial and open source providers of virtual 3D environments.

### Pedagogical and Technological Framework

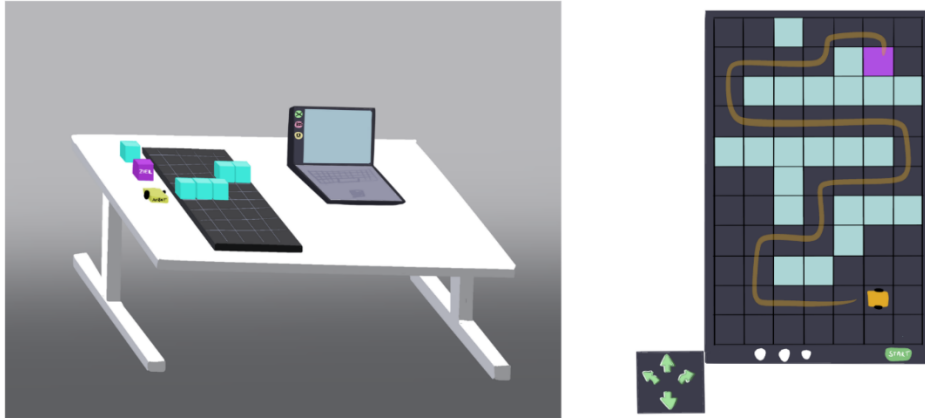
This study explores the implementation of a cyber-physical application in a computer science course where students program a virtual robot to navigate a maze. Learners begin by constructing a parkour using a dedicated editor. This scenario is enriched with a gamification component, where participants, represented as avatars, compete to complete the maze as quickly as possible. The interactive nature of this learning environment not only fosters engagement, but also develops multiple

competencies through group work. Additionally, conversational agents are employed to diagnose programming errors and provide targeted assistance.

This virtual 3D learning environment is used primarily for programming training in universities. While previous efforts focused on teaching basic programming languages to early-semester students (Lueckemeyer, 2015), the present project builds on previous research (Lecon & Herkersdorf, 2014) by challenging students to program algorithms to control a robot navigating a maze. Students first create and animate the robot using 3D modelling tools, followed by group collaboration to program its movements and address challenges such as collisions and item collection. This competitive and gamified approach, in which run times are compared between groups, adds an additional layer of motivation.

The first concept is shown in Figure 2. Each student is provided with a virtual workstation. The grid board on the left side can be used to build the obstacle course. With the laptop on the right side of the desk, the students can set up and execute the code. The concept also includes a top view of a possible result.

Figure 2. Concept, see also (Lecon et al., 2024)



### Integration into the e-Learning Infrastructure

To ensure seamless integration of virtual 3D environments into existing educational frameworks, appropriate interfaces were developed. These interfaces link the Learning Management System (LMS), the 3D environment, and a code repository, usually based on version control systems (Spinellis, 2004). Students access the virtual environment through the LMS, enabling integration of real-time program code analysis and conversational agent support. Robot log data is also transmitted to LMS to track task progress and results.

Future enhancements aim to synchronize real-world robots with their digital twins in virtual space. Real-world data, gathered through sensors, will be used to dynamically generate virtual environments, bridging the gap between physical and virtual learning experiences.

## Collaboration in Virtual 3D Spaces

Collaboration, defined as a joint problem solving effort where participants learn together (Lai, 2011), is a cornerstone of this approach. Unlike cooperation, which involves independent contributions, collaboration emphasizes collective interaction, fostering group synergy. Meta-analyses suggest that collaborative learning particularly benefits lower-performing students, while heterogeneous groups achieve optimal outcomes due to diverse skill sets (Oliver, & Marwell, 1988); Macy, 1999).

In practice, interdisciplinary groups comprising computer science and user experience students have demonstrated superior creativity and functionality in virtual reality projects. Factors like social facilitation also play a role: experienced students often perform better under observation, while less experienced ones benefit from reduced stress due to avatar-mediated partial anonymity (Harkins, 1987).

## AI-Assisted Conversational Agents

The use of AI in education often began with the use of "intelligent" chatbots (Adamopoulou & Moussiades, 2020). AI, especially language models, has now advanced to the point where they can act as individual contact persons ("agents") with natural language input and output (Terzidou & Tsiatsos, 2014; Zargham, et al., 2021). The basis is often ChatGPT, with its strengths, but sometimes also (initial) limitations (Thorp, 2023).

In asynchronous learning settings, conversational agents act as pedagogical supports, assisting students navigate the virtual environment or troubleshooting programming challenges. These agents, represented as avatars, offer natural language communication and gesture-based interactions to enhance immersion. The use of AI-controlled avatars has now also arrived in the educational sector (Doe et al, 2025).

Design considerations for these agents include:

- **Dialogue Mode:** Text-based, speech bubbles, or natural language communication.
- **Appearance and Accessibility:** Visibility on-demand or at predetermined locations, with customisable designs to foster trust.

Two modes of interaction are envisioned: direct student queries and proactive intervention when errors are detected. For example, agents can analyse the source code stored in a version control system (VCS) and provide feedback on errors, suboptimal solutions, or alternative approaches.

## Metrics for Learning Objectives

Learning outcomes are structured using Bloom's taxonomy and its extensions, such as the Anderson-Krathwohl taxonomy (AKT) (Anderson & Krathwohl, 2001). Building on this, there are further simplified (Dreyfus SE, 2004) or domain-specific (Kiesler, 2002) taxonomies. In general, competencies are categorized across cognitive dimensions—remembering, understanding, applying, analyzing, evaluating, and creating. For programming education, competencies include:

- Remembering: Recalling syntax and concepts.
- Understanding: Grasping semantics and development environments.
- Applying: Implementing solutions that adhere to quality standards.
- Analyzing: Identifying optimal algorithms for specific problems.
- Evaluating: Assessing algorithm efficiency.
- Creating: Developing functional software solutions.

The integration of AI-driven conversational agents aligns with these cognitive objectives by offering targeted support and improving student autonomy.

## Results

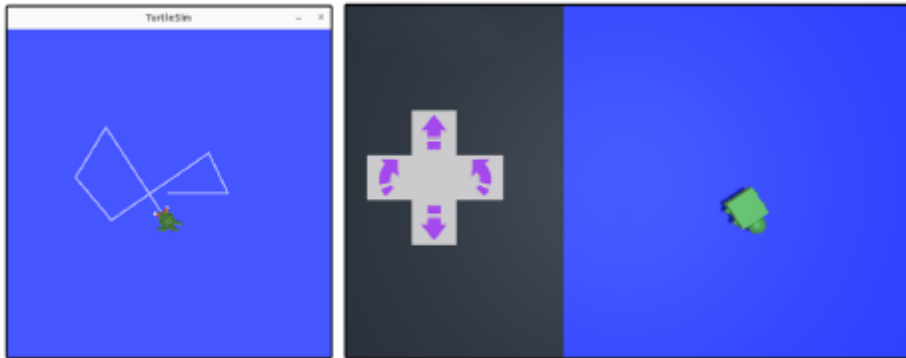
The concept was implemented by developing a robot simulation using the ROS interface (Koubâa (ed.), 2017). To provide consistent development conditions for all students, a standardised virtual machine (VM) was created. This VM includes an installation of ROS2 Iron Irwini, a distribution of the Robot Operating System 2. ROS2 serves as middleware, enabling communication between the various components of the system.

This setup ensures that students can work in a uniform software environment, regardless of the underlying hardware. For the development of the simulation environment, the Unity 3D game engine was chosen, as it is well-suited for creating interactive 3D scenarios. Unity also offers the "Unity Robotics Hub," which facilitated the integration of a ROS2 TCP Connector. This plugin enables bidirectional real-time communication between Unity and ROS2 via standard or custom topics. As a result, Unity can send data to ROS2 and receive ROS-based data in return for visualisation purposes.

### Initial Phase: Testing with TurtleSim

To test the environmental setup before developing the main simulation scenario, "TurtleSim" was used. This lightweight ROS-based simulation tool is typically used for educational and introductory purposes. The initial testing phase served as a technical proof of concept to verify the stability and bidirectionality of the communication between ROS2 and Unity through the ROS2 TCP Connector. TurtleSim was chosen due to its simplicity and seamless integration within the ROS2 framework. It includes implementations of core ROS concepts such as nodes, topics, and message structures.

Figure 3. Bidirectional Communication using TurtleSim



To achieve the simulation result shown in Figure 3, several key steps were taken. The 2D top-down visualisation of TurtleSim was recreated in 3D using Unity. The ROS2 topic “`geometry_msgs/Twist`” was used to transmit linear and angular velocity commands to the simulated turtle. Both environments subscribed to this topic to monitor movement commands, enabling real-time positional updates. Additionally, a custom control menu was implemented in the 3D visualisation, allowing users to send new position data to the TurtleSim topic.

As a result, both the simulation within the ROS2 environment and the Unity 3D environment displayed synchronised movements with minimal delay. This successful bidirectional data exchange between Unity and ROS2 confirmed the reliability of the ROS2 TCP Connector and established the technical foundation for implementing more complex navigation and visualisation features in the main simulation scenario.

Although TurtleSim was not intended for direct student use, this phase was essential to ensure the robustness of the communication architecture before proceeding to student-facing simulations.

### Main Scenario: Autonomous Navigation in a 3D Obstacle Course

After completing the TurtleSim phase, the next step was to develop a scenario that met both technological and pedagogical objectives. A 3D environment was created in which students could test their pathfinding solutions – either as algorithms (see, for example, (Hart, Nilsson & Raphael, 1968)) or by using sensor data. To support this, a dedicated workstation setup was designed, featuring a monitor for saving, loading, and resetting functions. On the other hand, a board with an adjustable grid size was provided. Students could place cubes on the board to act as obstacles. In addition, they could position a robot model as the starting point and a sphere as the destination. If the pathfinding algorithm was successful, the robot would be animated along the grid to the destination point.

### System Architecture Overview

The architecture consists of the following core components:

The **ROS2 back-end**, which runs inside the virtual machine, is responsible for receiving the grid layout, as well as the start and goal positions. Calculate the path and outputs the corresponding movement commands. This is the component in which students implement and test their pathfinding solutions.

The **Unity front-end** handles the visualisation of the robot and its environment. It allows for user interaction and manages the exchange of messages with the ROS2 backend. In the front-end, the user can set up the obstacle course and position the robot and the target. Using buttons on a virtual screen, the player can save, load, reset, or send the course to the ROS2 backend.

### Obstacle Course Creation

The obstacle course was entirely designed in Unity, representing the environment in which the virtual robot would operate.

The grid is generated in an empty GameObject that has been placed on the desk pad. A script called "GridController" has been added to the empty GameObject. The number of rows, the number of columns, and the GameObject to be placed can be transferred to this script via the Inspector. In this script, a two-dimensional array is first created in which the generated objects are stored. The grid is then generated using a function called "BuildGrid". In this function, the width is first read from the transferred GameObject. The objects are then generated in two nested loops and moved to the correct position.

The object that is arranged in the grid is a prefab called "GridField". There are three sub-objects in this prefab. First, there is a visible object in which the player can recognise the individual cells of the grid. There is also an invisible BoxCollider above the visible object. A script called "GridFieldControl" has been added to the collider object to control this field.

The following variables can be set for this script via Unity: two colours can be defined. One is the normal colour, and the other is the colour that the object should have when a block is in the trigger. In addition, the cell object that is to be coloured and the position at which the block is to be located after the snap can be transferred. At the cell start of the script, the colour is first set to the default colour. As soon as an object collides with the collider, it is checked whether it is an object that has the "Gridable" tag. If this is the case, the colour of the cell is changed, and the object is temporarily stored. If the object leaves the collider again, the colour is changed again, and the cached object is deleted. As long as the object is in the collider, a check is made every frame to see if the object is released. If the object is released, the position of the block is adjusted to the previously defined block position, and the collider is deactivated.

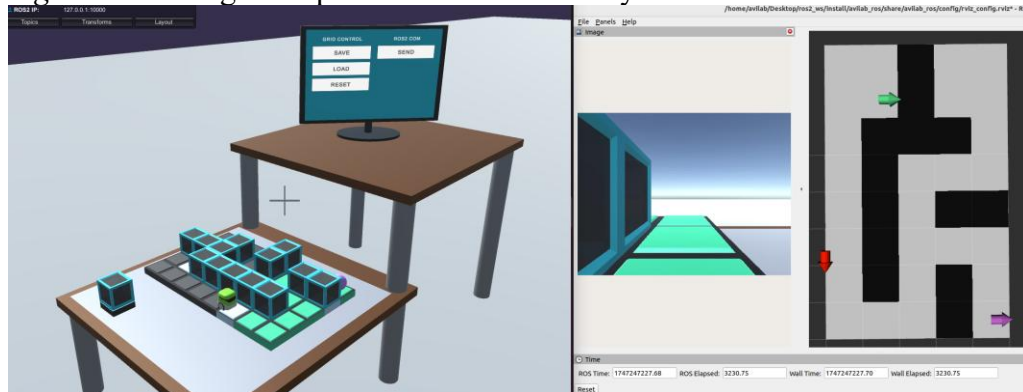
The blocks are generated infinitely via a "BlockSpawner". The "BlockSpawner" also consists of three components. The visible part of the object is similar to the grid cells. There is also an empty object in BlockSpawner that stores the spawn position. The most important part is the collider. This again consists of a BoxCollider and is located above the visible part. The "BlockSpawner" script has been added to this object. In the script, the block to be placed, the position, and the delay can be changed in the Unity Inspector. The script has been programmed so that a new block

is generated in the collider each time a block leaves the collider. This ensures that sufficient blocks are available for the construction of the obstacle course.

An additional object was placed in the middle of the block with a small box collider. This object has been given the tag "Gridable". This additional object ensures that the block is recognised by a maximum of a single grid cell at any time.

A 3D model of a robot has already been created, which is to be edited in the next steps so that it can be placed on the grid (see Figure 4; the (stylized) light green robot is at the front). A target point should be modelled, which should also be placed on the grid.

*Figure 4. Running Example of the Course in Unity and ROS2*



#### Communication between ROS2 and Unity

The communication between ROS2 and Unity takes place via a TCP connection. The virtual machine (VM) running ROS2 was configured to forward TCP port 10000 into and out of the VM. In the ROS VM, port 10000 is defined both in `server.py` during the initialisation of the TCP endpoint and in `endpoint.py`. The IP address used is 0.0.0.0. The ROS-TCP-Endpoint opens a TCP server that handles communication. All messages between Unity and ROS2 are transmitted through this TCP server.

In Unity, the official package "ROS-TCP-Connector" by Unity Technologies was used (Unity Technologies, 2025). The ROS settings in Unity are configured as follows: the option to automatically establish the connection at program start was enabled. The communication protocol is set to ROS2. To establish the connection, the ROS IP address was set to 127.0.0.1 and the ROS port to 10000. In this way, Unity attempts to establish a local connection to the virtual machine and the TCP server running on it.

To obtain runtime information about the connection, the "Unity Robotics Visualizations Package" by Unity Technologies was used (Unity Technologies, 2025). Once the `DefaultVisualizationsSuite` prefab is added to the scene hierarchy, the connection information is displayed in the top-left corner of the screen when starting the application. If no connection is possible, the arrows are coloured red. When the connection is successful, the arrows are displayed in blue. This allows users to immediately assess at program startup whether the system is working correctly or not.

## Expandability through Sensor Integration

To demonstrate the modularity and future potential of the platform, a virtual camera sensor was added to the Unity Scene. The ROSCamera from the UnitySensors package by Field-Robotics-Japan was used as the camera sensor (Field Robotics Japan, 2025). In a separate script, a ROS message of type CompressedImage is sent to the topic `"/camera/color/image/compressed"` at the end of each frame. The camera image is rendered in a RenderTexture, then converted to a byte array and sent to ROS along with the current timestamp. There, it can then be displayed using the Camera component in real time and may be further processed. The system is also designed to support a wide range of additional sensors, such as:

- Point clouds from LiDAR sensors (e.g., for obstacle detection)
- Distance, touch, or temperature sensors
- IMU
- Live data from physical robots, enabling hybrid digital twin scenarios

## Next Steps

The simulation itself is already partially embedded into an existing Unity-based virtual 3D educational environment. This environment allows users to interact as 3D avatars, participate in conferences or presentations, and communicate via voice chat or emotes. It is highly customisable and can be adapted to a wide range of educational use cases. The integration of our simulation within this world opens new possibilities for collaborative learning and realistic training scenarios.

Parallel to the current development, conversational agents are being implemented in a separate subproject. These AI-based agents are designed to provide individualised support to instructors and can also be integrated into our scenario. Their goal is to assist students directly within the simulation, providing guidance, answering questions or explaining tasks—even when a human instructor is not present. This enables a more flexible and autonomous learning experience, particularly valuable in asynchronous or remote educational settings.

The next development steps focus on the following:

- Full integration into the virtual 3D environment: While the virtual environment is currently being successfully simulated, full integration into the interactive world is still ongoing. The necessary communication interfaces, based on the publish-subscribe model provided by ROS2, are already in place and will enable seamless interaction between the simulated systems and the virtual environment.
- Evaluation through practical application: The effectiveness of the system will be evaluated in two educational settings: one in a traditional face-to-face university, where students will collaborate synchronously in groups, and one at a distance learning university, where group work will occur asynchronously. This evaluation will help assess both the technical reliability and the pedagogical value of the solution in different learning contexts.

These next steps aim to ensure that the simulation system not only functions reliably on a technical level but also adds clear value to digitally enhanced teaching and learning scenarios.

### **Conclusion**

This study introduced a robot programming learning scenario within a virtual 3D environment, highlighting its potential to address the challenges of teaching and learning in remote or hybrid contexts. Such environments are particularly advantageous when traditional face-to-face teaching is not feasible, such as during pandemic-related restrictions or in distance learning courses. At the same time, it shows a concrete use case for virtual 3D spaces in education.

Virtual 3D environments offer significant benefits by bridging the methodological and didactic gaps inherent in practical courses, such as programming and laboratory seminars. They provide enhanced spatial and social immersion, allowing for more natural communication compared to two-dimensional platforms. Immersion can be intensified through the use of head-mounted displays (Wu, Yu & Gu, 2022). Our application supports these devices. However, the test runs so far have been conducted directly on the PC, so intensive immersion is still achieved; therefore, maintaining a class set of head-mounted displays is not absolutely necessary.

The described scenario involved developing and implementing algorithms to navigate a robot through a course designed by trainers or learners. This practical result is visualized as a 3D model, where the robot's movements occur in real time within the virtual space, enhancing experiential learning.

Participants, represented as avatars, benefit from the social immersion these environments offer. While avatars enable partial simulation of facial expressions, they also promote objective and focused communication. In addition, computer-generated conversational agents can be integrated to support users, serving as accessible and responsive points of contact for technical guidance and navigation within the virtual environment.

A prototype of this system has been developed as a proof of concept. Future work will focus on evaluating its effectiveness in both traditional university settings as a supplement to classroom teaching and within the curriculum of an online distance learning university.

### **Acknowledgments**

Part of this work is funded through the German Federal Ministry of Education and Research (Project 16INB2005C – AVILAB2).

## References

- Adamopoulou, E., & Moussiades, L. (2020). An overview of chatbot technology. *Artificial Intelligence Applications and Innovations (AIAI 2020)*, IFIP AICT, 584, 373–383. Springer.
- Anderson, L. W., & Krathwohl, D. R. (2001). *A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives*. Addison Wesley Longman, New York.
- Beck, D. (2019). Augmented and virtual reality in education: Immersive learning research. *Journal of Educational Research*, 57(7), 619–625.
- Bdidara, J., & Rusman, E. (2016). Towards a pedagogical model for science education: Bringing educational contexts through a blended learning approach. *Open Learning: The Journal of Open, Distance, and e-Learning*, 32(1), 6–20.
- Bizami, N. A., Tasir, Z., & Na, K. S. (2022). Innovative pedagogical principles and technological tool capabilities for immersive blended learning: A systematic literature review. *Education and Information Technologies*, 1–53.
- Boekaerts, M. (1996). Self-regulated learning at the junction of cognition and motivation. *European Psychologist*, 1(2), 100–112.
- Bloom, B., Engelhart, M. D., Furst, E. J., Hill, W. H., & Krathwohl, D. R. (1956). *Taxonomy of Educational Objectives: Handbook I – Cognitive Domain*. McKay.
- Borkovic, D. J., Skovira, R. J., & Kohun, F. (2021). Virtual social distancing: A digital ethnography of online learning. *Issues in Information Systems*, 22(4), 244–257.
- Chou, C. Y., Chan, T. W., & Lin, C. J. (2003). Refining the learning companion: The past, present, and future of educational agents. *Computers & Education*, 40(3), 255–269.
- Churchill, E. F., & Snowdon, D. (1998). Collaborative virtual environments: An introductory review of issues and systems. *Virtual Reality*, 3(1), 3–15.
- Dede, C. (2009). Immersive interfaces for engagement in learning. *Science*, 323(5910), 66–69.
- Dembski, F., Wassner, U., & Letzgus, M. (2019). *The digital twin: Tackling urban challenges with models and simulations in immersive virtual environments*. eCAADe + SIGraDI Conference – Architecture in the Age of the 4th Industrial Revolution.
- Doe, J., et al. (2025). Pedagogical AI conversational agents in higher education: A conceptual framework and review of the state of the art. *International Journal of Educational Technology*.
- Dreyfus, S. E. (2004). The five-stage model of adult skill acquisition. *Bulletin of Science, Technology & Society*, 24(3), 177–181.
- Field Robotics Japan. (2025). *UnitySensors – Camera Sample*. <https://github.com/Field-Robotics-Japan/UnitySensors/tree/master/Assets/UnitySensorsROS/Samples/Camera>
- Garau, M., Slater, M., Vinayagamoorthy, V., Brogni, A., Steed, A., & Sasse, M. A. (2003). *The impact of avatar realism and eye gaze control on communication in immersive environments*. Proceedings of CHI Conference on Human Factors in Computing Systems, 529–536.
- Girvan, C. (2018). What is a virtual world? Definition and classification. *Educational Technology Research and Development*, 66, 1087–1100.
- Harkins, S. G. (1987). Social loafing and social facilitation. *Journal of Experimental Social Psychology*, 23, 1–18.
- Hart, P. E., Nilsson, N. S., & Raphael, B. (1968). A formal basis for the heuristic determination of minimum-cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2).
- Kiesler, N. (2002). *On programming competences and their classification*. Proceedings of the 20th Koli Calling Conference on Computing Education Research. ACM.

- Koubâa, A. (Ed.). (2017). *Robot Operating System (ROS): The Complete Reference (Vol. 1)*. Studies in Computational Intelligence, 625, Springer.
- Lai, E. R. (2011). *Collaboration: A Literature Review*. Pearson Research Report.
- Lecon, C., & Herkersdorf, M. (2014). *Virtual blended learning: Virtual worlds and teaching scenario integration*. IEEE ICCSE 2014.
- Lecon, C., Soeser, S., & Stegmaier, A. (2024). *Programming teaching in a virtual environment using the ROS interface*. 18th International Education and Development Conference, Valencia, Spain.
- Lueckemeyer, G. (2015). *Virtual blended learning enriched by gamification and social elements in programming education*. IEEE ICCSE 2015, 438–444.
- Mantovani, F., & Castelnovo, G. (2003). *Sense of presence in virtual training*. In: Riva G., Davide F., & Ijsselstein W. A. (Eds.), *Being There: Concepts, Effects and Measurement of User Presence in Synthetic Environments*. IOS Press.
- Macy, M. W. (1999). Learning theory and the logic of critical mass. *American Sociological Review*, 55(6), 809–826.
- Pagano, K. O. (2013). *Immersive Learning*. ASTD Press.
- Qi, Q., Hu, T., Anwer, N., Liu, A., Wei, Y., Wang, L., & Nee, A. Y. C. (2021). Enabling technologies and tools for digital twin. *Journal of Manufacturing Systems*, 58(B), 3–21.
- Spinellis, D. (2004). *Version control systems*. IEEE Software, 22(5).
- Terzidou, T., & Tsiatsos, T. (2014). *The impact of pedagogical agents in 3D collaborative serious games*. IEEE EDUCON 2014, 1175–1182.
- Thorp, H. H. (2023). ChatGPT is fun, but not an author. *Science*, 379(6630), 313.
- Unity Technologies. (2025). *ROS-TCP-Connector – Visualizations Documentation*. <https://github.com/Unity-Technologies/ROS-TCP-Connector>
- Wu, B., Yu, X., & Gu, X. (2022). Effectiveness of immersive virtuality using HMDs: A meta-analysis. *British Journal of Educational Technology*, 51(6), 1991–2005.
- Zargham, N., Bonfert, M., Porzel, R., Döring, T., & Malaka, R. (2021). *Multi-agent voice assistants: An investigation of user experience*. MUM '21 Proceedings, 98–107.
- Zinn, B., Guo, Q., & Saga, D. (2016). Entwicklung und Evaluierung der virtuellen Lern- und Arbeitsumgebung VILA. *Journal of Technical Education*, 4(1), 89–117.

