EXTENDED REALITY (XR) & GAMIFICATION IN THE CONTEXT OF THE INTERNET OF THINGS (IOT) AND ARTIFICIAL INTELLIGENCE (AI)

By

Georgios Pappas

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Electrical Engineering – Doctor of Philosophy

2021

ABSTRACT

EXTENDED REALITY (XR) & GAMIFICATION IN THE CONTEXT OF THE INTERNET OF THINGS (IOT) AND ARTIFICIAL INTELLIGENCE (AI)

By

Georgios Pappas

The present research develops a holistic framework for and way of thinking about Deep Technologies related to Gamification, eXtended Reality (XR), the Internet of Things (IoT), and Artificial Intelligence (AI). Starting with the concept of gamification and the immersive technology of XR, we create interconnections with the IoT and AI implementations. While each constituent technology has its own unique impact, our approach uniquely addresses the combinational potential of these technologies that may have greater impact than any technology on its own. To approach the research problem more efficiently, the methodology followed includes its initial division into smaller parts. For each part of the research problem, novel applications were designed and developed including gamified tools, serious games and AR/VR implementations. We apply the proposed framework in two different domains: autonomous vehicles (AVs), and distance learning.

Specifically, in chapter 2, an innovative hybrid tool for distance learning is showcased where, among others, the fusion with IoT provides a novel pseudo-multiplayer mode. This mode may transform advanced asynchronous gamified tools to synchronous by enabling or disabling virtual events and phenomena enhancing the student experience. Next, in Chapter 3, along with gamification, the combination of XR with IoT datastreams is presented but this time in an automotive context. We showcase how this fusion of technologies provides low-latency monitoring of vehicle characteristics, and how this can be visualized in augmented and virtual reality using low-cost hardware and services. This part of our proposed framework provides the methodology of creating any type of Digital Twin with near real-time data visualization.

Following that, in chapter 4 we establish the second part of the suggested holistic framework where Virtual Environments (VEs), in general, can work as synthetic data generators and thus, be

a great source of artificial suitable for training AI models. This part of the research includes two novel implementations the Gamified Digital Simulator (GDS) and the Virtual LiDAR Simulator.

Having established the holistic framework, in Chapter 5, we now "zoom in" to gamification exploring deeper aspects of virtual environments and discuss how serious games can be combined with other facets of virtual layers (cyber ranges, virtual learning environments) to provide enhanced training and advanced learning experiences. Lastly, in chapter 6, "zooming out" from gamification an additional enhancement layer is presented. We showcase the importance of human-centered design of via an implementation that tries to simulate the AV-pedestrian interactions in a virtual and safe environment.

Keywords: Gamification, serious games, gamified tools, simulation, eXtended Reality, XR, Internet of Things, IoT, Artificial Intelligence, AI, virtual environments, VE, synthetic data

To my son, Christos, my wife Tonia,
my parents, Christos and Maria,
and my sister, Alexandra

ACKNOWLEDGEMENTS

Participating in and being the first candidate to complete the Dual Ph.D. program between the National Technical University of Athens and Michigan State University was an incredible experience. This would have never been successful without the help and support of certain people.

I would like to thank my two advisors Dr. Konstantinos (Kostas) Politopoulos and Dr. Joshua (Josh) Siegel. Kostas and Josh were always there for me when I needed them. Their help, support, collaboration and friendship will endure beyond these two Ph.D.'s and I will always be grateful for this. Kostas and I have a history collaborating since my undergraduate years, with Kostas serving as my thesis supervisor and teaching me how to think as an engineer in my career and in my life. Josh and I met at the Massachusetts Institute of Technology, where he was the lead instructor of the inaugural Internet of Things Bootcamp and I was a student. Since then, we have collaborated in many capacities including with him advising my work at Michigan State University for this Dual Ph.D. program. Through countless hours working at night, under lockdown, and with online meetings, we completed great cross-disciplinary research and established a long-lasting friendship.

I would also like to thank professors John Papapolymerou (MSU ECE's Department Chair) and Nectarios Koziris (NTUA ECE's Dean) for their support and help and for initiating this unique program that may be the first Dual Ph.D. granting program between Greece and the US. It is no secret that every new program may have difficulties in the process and this was no different, but Drs. Papapolymerou and Koziris were there to help and support throughout the process.

Further, I would like to thank my committee members from both universities for our talks and their support. From NTUA, Professor Emeritus Kyriakos Hizanidis, Professor Glytsis, Professor Tsanakas and Professor Kollias. From MSU, Associate Professor Mi Zhang and Professor of Practice Jeremy Bond.

Special thanks to MSU MI's Professor Carrie Heeter for letting me be a part of the Serious Game Design and Research program and for her support when I faced a serious health issue.

Thank you also to all the members of the Graduate Office at NTUA (Efi Kanta, Anna An-

drikopoulou, Martha Sifaki and Maria Xeidianaki) for their tireless and always immediate help.

Next, I am grateful to the Open University of Cyprus for giving me the opportunity to apply my theories and research work in Distance Learning as a member of the Lab of the Educational Material and Methodology.

This Ph.D. would not have been possible without the scholarship and the fellowship I received these years and I am extremely grateful.

This journey would also not have been successful without my invisible heroes. My love to my son Christos for making my daily life so colorful and my wholehearted thanks and love to my amazing wife, Tonia, for being the person who handled everything in the real world for me when I was lost in my "virtual environments." Thank you also to two of the most important people in my life, my parents, Christos and Maria, who were and always are there for me in good and bad times and supported me in all my educational aspirations since I was a child. And last but not least I would like to thank my incredible sister, Alexandra, for being the constant reminder of creativity in my life.

Important Note: This dissertation is submitted in partial fulfillment of the requirements for the Dual Ph.D. Program between Michigan State University and National Technical University of Athens. A similar version of this document is submitted to both institutions.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

The present cross-disciplinary research will establish a framework and build bridges among various Deep Technologies [2]. Specifically, we bring Gamification and Extended Reality (XR), Internet of Things (IoT) and Artificial Intelligence (AI) together in a way that provides real-time data visualization (IoT as input to XR/Gamification), educational opportunities for humans (Gamification) or even training for machines (XR/Gamification as output to external AIs). Our primary focus will be on XR and Gamification (Figure 1.1).

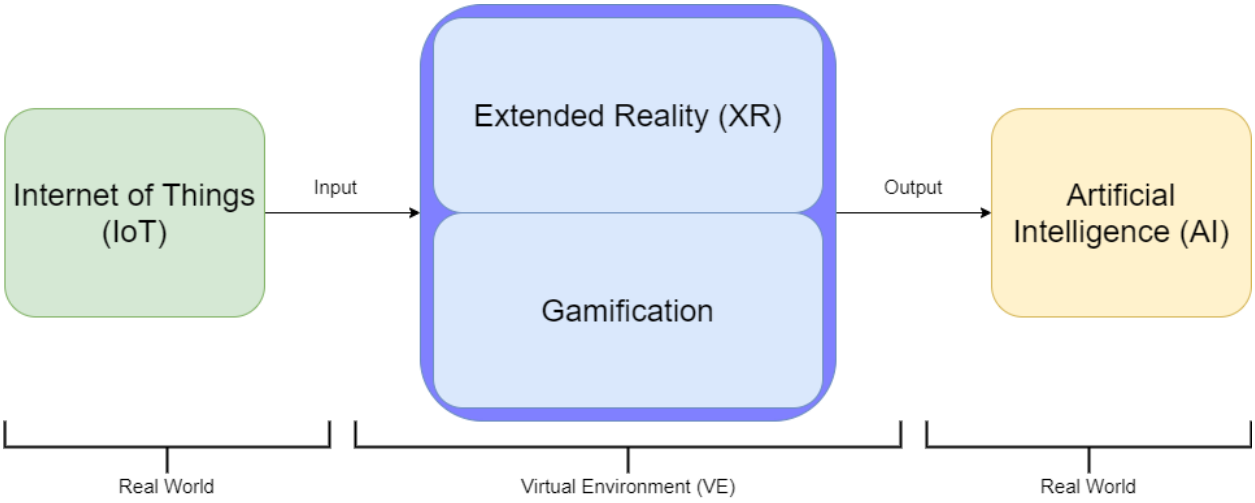


Figure 1.1: Holistic overview of the proposed bridge between IoT, XR/Gamification and AI.

Game development and gamification are concepts usually perceived as entertainment based technologies but modern game engines such as Unity3D or Unreal Engine are very powerful and flexible tools. This is because they provide both 2D and 3D workspaces where in-depth Physics Systems may be applied. Further, the constant updates of the game engine SDKs along with the community work (open source or asset stores) provide academia the means to explore and solve research problems quickly, easily, and inexpensively. While games are usually the "kingdoms of surreal," the engines they are built upon and the virtual environments (VEs) these incorporate, may instead simulate specific research conditions or generate virtual data useful in

real-world applications. Combining game development techniques with industrial XR SDKs like Vuforia (PTC) [3], ARKit (Apple) [4], ARCore (Google) [5] or SteamVR [6], an engineer may turn a virtual environment into an immersive experience and create advanced desktop and mobile applications.

In order to be able to simulate and visualize conditions, the crucial part is not only programming these conditions correctly but also to feed the virtual world with real world data. To achieve the above, we may acquire data from IoT sensor implementations (IoT to XR/Gamification). This first research contribution explores how IoT might work as an input method for data visualization or virtual event triggering in a near real-time or a lowlatency condition.

In order to generate useful artificial data (measurements, images), we must simulate the real world inside game engines. After the simulated world design is complete, in order to capture real event data, we let users test the gamified tools or play the serious games we have developed. The synthetic data is later validated on AI models that we may train (Gamified Digital Simulator [7] - Chapter 4) or are pre-trained using real data (Virtual LiDAR Simulator [8] - Chapter 4).

We further explore options for human training via gamification in Virtual Environments (VE) that may be combined with other layers of VEs to provide a higher level of interactivity to users. Lastly, we focus on user-centered design by not only designing and developing but also performing survey-based research.

1.1 Compartmentalization of the research problem

In the 17th century, the French mathematician and philosopher René Descartes proclaimed: "Divide each difficulty into as many parts as is feasible and necessary to resolve it." Although this portion of wisdom is quite old nowadays, it is remarkable how efficient it proves to be as a methodology in problem solving.

In the present dissertation, to achieve the proposed research goal of combining these technologies (Figure 1.1), the same path was followed. The problem was decomposed into smaller parts and each step was handled individually. The main concept of our work was proved applicable in a

variety of domains and we have already applied it successfully in the area of autonomous vehicles and education.

The figure 1.2 shows how this dissertation is organized. Each of the next chapters plays a significant role and provides something unique to the suggested framework while at the same time, it offers individual contributions.

More analytically:

- **Chapter 2:** The environmental studies tool is featured in this chapter. With this tool, we introduce the combination of advanced gamification techniques with an IoT implementation in the context of distance learning. The IoT is used as an enabler/disabler of objects and phenomena in a virtual and gamified world.

- **Chapter 3:** The VirtualCar implementation is illustrated in this chapter. Similarly to Chapter 2, IoT is combined with gamification again but using industry SDKs we also create immersive XR applications that can also stream IoT data. Also, this time we visualize the IoT sensor data. The context of this research part is the automotive industry.

- **Chapter 4:** This chapter features two implementations, the Gamified Digital Simulator (GDS) and the Virtual LiDAR Simulator (VLS). Both implementations offer unique methodologies of generating synthetic data from virtual environments. This data can train new AI models which may work in the real world or be validated by pre-trained models that had been fed with real data in first place. This chapter proves the connection of gamification with AI via the generation of valuable synthetic data.

- **Chapter 5:** This chapter, although it does not provide any connection with IoT or AI, is important because we dive deeper into gamification which is the core of this research. In this case, novel connections between a variety of virtual layers are combined to enhance the

gamification itself. The context is distance learning and the tool featured is the Cyber Escape Room. This tool offers integration with both Cyber Ranges (Virtual Machines) and Virtual Learning Environments (Moodle).

- **Chapter 6:** This chapter offers an external enhancement to gamification and in expansion the whole framework. Via an advanced simulator that explores the AV-pedestrian interactions in a virtual and safe environment, the user-centered design is showcased since it is a tool that tries to put human aspect at the center of attention and tries to raise awareness on the topic of AVs.

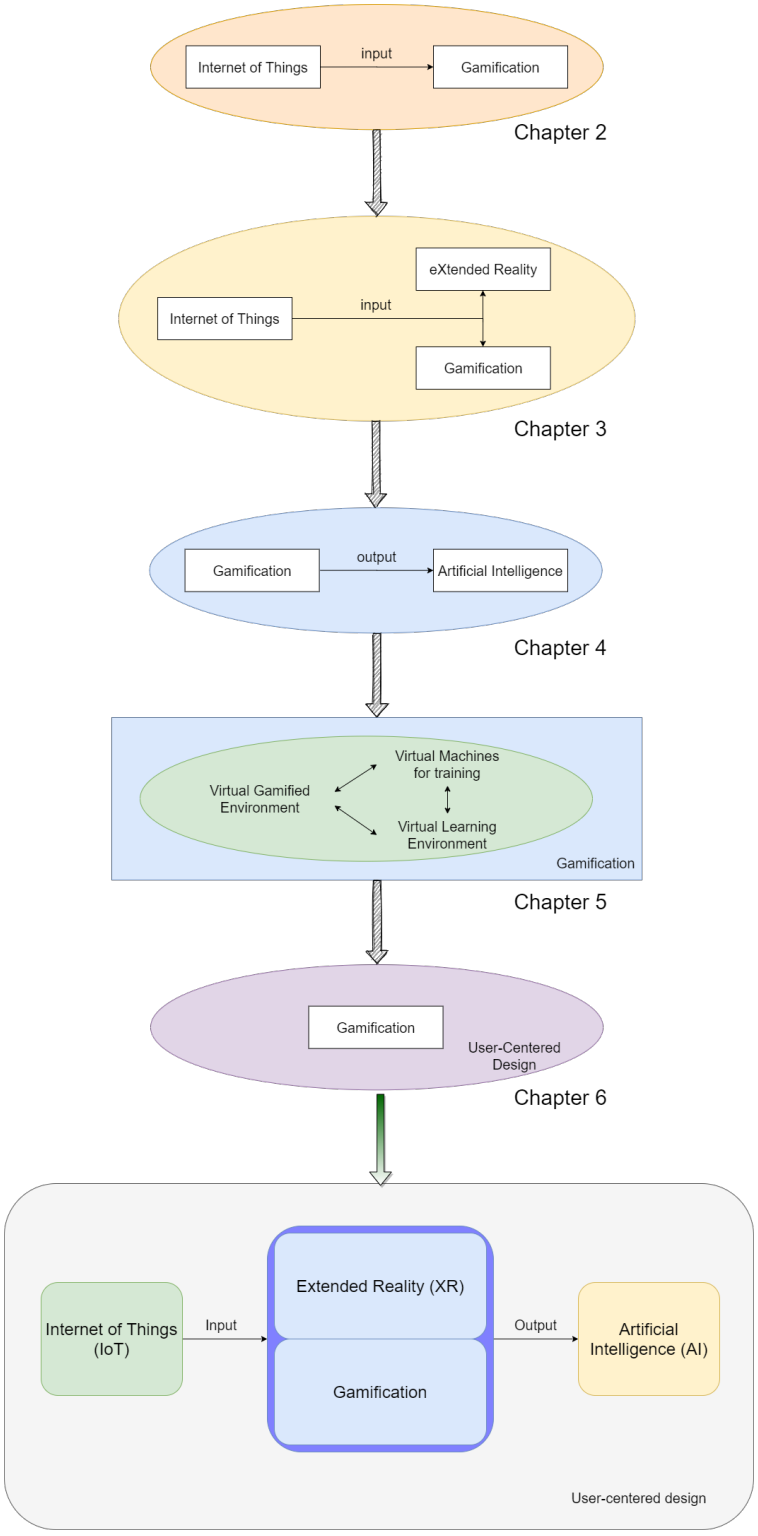


Figure 1.2: Overview of the dissertation chapters. Chapters 2, 3 and 4 prove the suggested holistic framework. Chapter 5 dives deeper into gamification and combines it with other layers of virtuality and chapter 6 enhance gamification with user-centered design. Altogether, they suggest a unified Deep Tech framework for solving various, domain agnostic problems.

CHAPTER 2

COMBINING GAMIFICATION WITH THE INTERNET OF THINGS

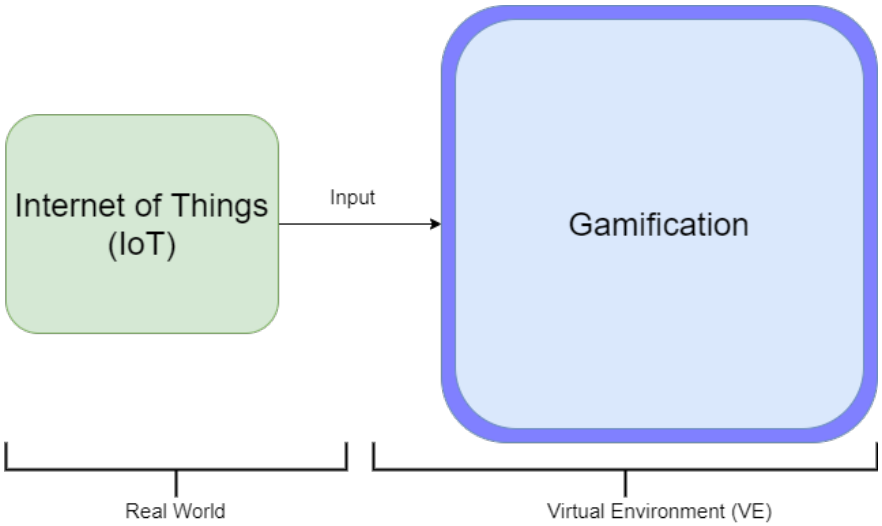


Figure 2.1: Combining IoT with Gamification diagram.

The first part of the research combines IoT along with advanced 3D Gamification techniques to create a novel application for distance learning. This tool is a hybrid synchronous-asynchronous serious game that includes a series of advanced game development implementations in addition to an innovative pseudo-multiplayer methodology. It is related to Environmental studies and provides distance-learning students with the common environment in a digital space for taking measurements for their projects regardless their real life location (e.g., major cities, towns or villages). Also, it features a virtual drone system that is accomplished by using multiple cameras and RenderTextures. Along with it, it provides a teleport system to specific locations where events take place. These events are triggered remotely by faculty (Figure 2.2) via microcontrollers and sensors/buttons, with all students able to watch the devices simultaneously. The microcontrollers send data to the cloud and the tool receives and updates this information into the scene (Figure 2.1). When an attached button switch is enabled, the forest is set on fire and related educational material appears. When it is off, the forest returns to its initial state via internal enabling/disabling mechanisms. One of the main differences between this implementation and the next one (see Chapter 3) is that the IoT

data are not visualized as they are received. Based on the data received, the tool is programmed to activate or deactivate GameObjects (such as fire particles), rather than displaying numbers or binary states.

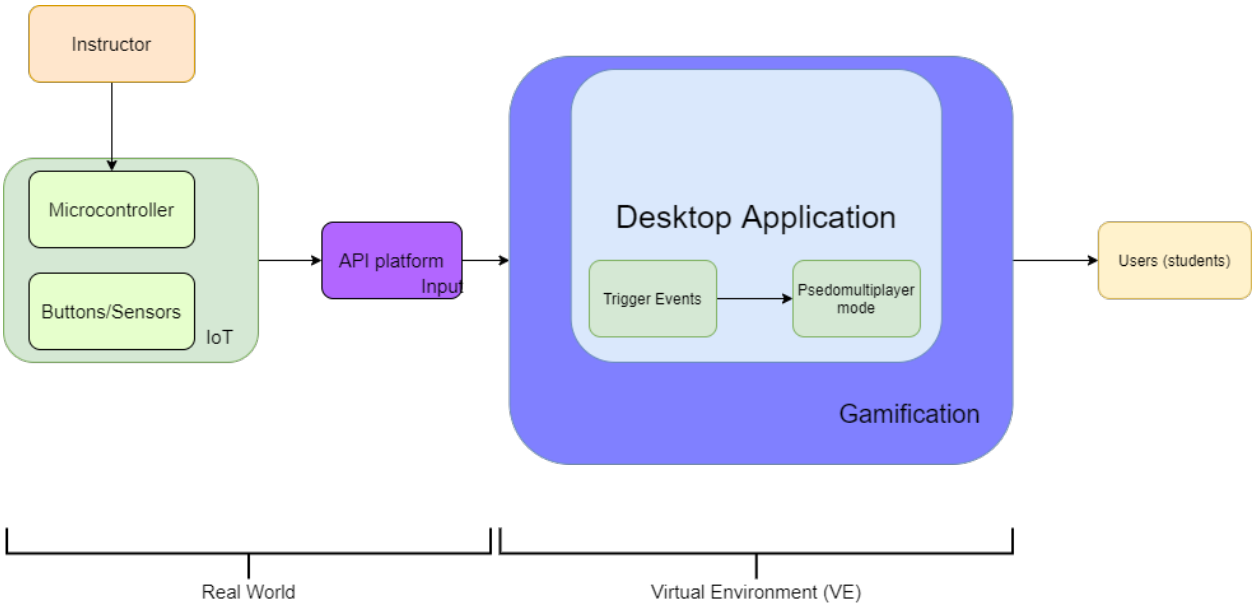


Figure 2.2: Environmental Tools Layout: Combining IoT with Gamification.

2.1 Gamification

Gamification is considered to be one of the most engaging ways to educate or train students in class or remotely in distance learning. At the same time, the Internet of Things is one of the Deep Technologies[2] that will disrupt many parts of our lives. Integrating Internet of Things implementations with gamified applications or tools and gamification will offer some unique and user-student approaches in education. In the present chapter, we introduce the concept of gamification and then provide an overview of the Internet of Things. Next, we showcase an example of how our team designed and developed a training gamified tool applied in the field of environmental studies and how this is integrated with the Internet of Things to enable triggers in a virtual 3D space. Following this, we will provide information on how this innovation was used in a real distance-learning class for Masters degree students.

2.1.1 Serious and Simulation Games in Education and Training

Gamification and serious games have proved to be valuable in training and education [9, 10, 11]. These games or gamified tools aim [12] to simulate realistic virtual environments in safe and accessible conditions and scenarios in which users and particularly students are able to interact in an engaging manner [13]. At the same time, the students not only acquire knowledge in their fields [14], but also learn in a "serious" fun way [15]. Some serious games can be used as virtual labs where experimentation can take place and based on this digital experience [16] and students can be assessed by their instructors [17]. This way, institutions can lower their operating expenses by limiting the purchases in expensive, specialized equipment since students will be using reusable enabling technology to generate flexible digital and simulated equipment. Using this kind of equipment the risk of damage from improper use is also eliminated. Another advantage that this simulated virtual environments offer is that they could be used in students' personal computers, transforming them in remote labs accessible anywhere, any day and anytime.

Studies have shown that advanced and high quality simulators have managed not only to train humans but also Artificial Intelligence (AI) models, in a human-to-machine and machine-to-machine approach [7, 18, 19, 20, 21].

2.1.2 Game Engines

Game development has changed in recent years. Importantly, developers - especially the "indies" or small studios, can buy commodity game development tools rather than creating everything from scratch. In the industry, there are several game engines that make development work easier. This is because most of them offer **advanced physics systems** including **particle systems**, **scripting** in popular or custom programming languages, **collider** and **detection systems**, **input management**, **rendering** and in-game **Artificial Intelligence** for non-player characters (NPCs) [22].

Some of the most popular and widely used are:

- **Unity3D:** This is one of the most powerful engines. It requires programming in C# or

JavaScript. Unity has a large developer community which is a powerful source of help and information. There are abundant free or commercial assets in the Unity Asset Store that can potentially speed up the development process. It has both a free and a professional version [23].

- **Unreal:** Together with Unity, Unreal is considered one of the top choices for game developers. It uses C++ as a programming language and also offers blueprints for non-programmers to work and design logic. Many popular AAA titles are made using Unreal [24].
- **GameMaker:** GameMaker has been a popular choice for non-programmer game developers since it does not require programming skills to create a game with it [25].
- **Godot:** Godot is a free and open-source engine that uses its own programming language GDScript which is similar to Python [26].
- **CryEngine:** CryEngine is another high-end engine and it uses C++, C#, Lua Script Bindings and Flowgraphs as programming languages [27].

2.2 The Internet of Things (IoT)

2.2.1 What is the IoT?

The Internet of Things (IoT) is a network of people, devices, and services [28] that can sense, connect to one another, make inferences, and act(uate) at scale [29, 30] with in excess of 30 billion IoT devices deployed globally today. [31] Sensing allows devices and services to measure themselves and their environments; connectivity allows information to be moved to where it may be aggregated, where it is most useful, and/or where it is easiest to process. Inference turns information into insight, and action and actuation allow insight to affect a system so as to form a closed control-loop based on near realtime and abundant data. This network of connected smart devices leverages pervasive sensing, connectivity, and computation to improve quality of life and deliver social, economic, and other value. [32] The IoT is a Deep Technology that was impossible

a few short years ago and is now easily feasible, pervasive, and invisible [2], made feasible by advances in sensing, storage, remote computing, and more. [32]

Today, tools exist to ease computer interaction with and among connected devices [33] and to simplify the development of analytical tools and feedback systems. These tools drive the development of novel applications at the intersection of traditional products, services, and knowledge silos, leading to increased adoption of the IoT in an ever-accelerating virtuous cycle. In the broadest-possible terms, IoT is a design language and a vocabulary focused on technology and industry convergence and enablement [34] with key tenants being data generation and use at scale.

2.2.2 The Impact of IoT

More people and things join the Internet of Things every day. [35] From these people and things, it is possible to generate more data from more places, allowing for the creation of a growing number of (data informed) products, services, and analytics at the intersection of those existing. [36] Measurement and data capture from diverse systems can lead to enhanced understanding of those systems, their environments and contexts, and the world at large. [37]

One example of data measurement, capture, and analysis takes the form of Digital Twins, which mirror physical systems mathematically within remote computing environments. [32] These Twins can be used for predictive analytics, or more simply, to enable compelling interactive visualizations taking the form of 2D dashboards, 3D rendered desktop applications, or even Virtual Reality. [38] These environments combine with other technological affordances to create opportunities for individuals to engage in experiences otherwise infeasible. IoT has the potential to create new industries, new modes of interaction, and a newly-immersive and responsive world. These worlds need not be purely physical. For example, IoT can combine with technologies such as eXtended Reality (XR) to allow individuals to transcend the limitations of the physical, to engage in experiences taking place in dynamic and responsive virtual environments [17], allowing for participants to engage in new and unique experiences in comfort and safety of their own home.

Beyond smarter visualizations and analytics, the Internet of Things also creates opportunities

through broad-scale data collection enabled by pervasive sensing, connectivity, and ubiquitous computing. Data collection, sharing, analysis, and action at scale [35] may include systems ranging from consumer devices to critical infrastructure [39, 40, 41] such as smart factories and automation systems. [42, 43, 44] IoT has also been used to monitor equipment [45] and to embed intelligence into "dumb" systems. [46, 47] The net result is that IoT combines with other technological affordances such as Big Data and AI to create smarter, more responsive environments ranging from homes to factories, often building upon the power of 1% at scale (e.g. small savings in one factory are insignificant, but when multiplied across an industry, the impact is radical).

2.2.3 IoT Devices and Sensors

Sensors are less expensive, lower power, more accessible, and more pervasive than ever. Sensors can measure parameters about the sensing system itself, or the sensing systems environment. Parameters measured may include environmental data, such as temperature and humidity, contextual data, such as location, acceleration, and rotation, or rich data sources such as audio, images, or video capture. These sensors upload data to constrained computing systems with battery, bandwidth, and storage limitations, though algorithms for data cleaning and analysis are ever-improving to allow for operation on low-power, low-resource systems. The net result is that the IoT is capturing and acting upon more data than wouldve been feasible a few short years ago, leading to smarter and more engaging systems.

2.2.4 Connectivity and Security

Connectivity allows data and intents (commands) to move across system boundaries, both for small-scale networks (few devices somewhat locally operated) and large-scale networks (a large number of potentially-diverse devices operating at massive geographic scale). Connectivity is important for data collection and aggregation, as well as in moving information from constrained-compute systems to environments where enhanced computation is feasible, such as the Cloud, Fog, or Edge. The ability to move data offers huge potential benefits to analytics and system

optimization, though scale also brings about risk. Maximizing the benefit of data aggregation requires thoughtful platform design, standards, and resource-aware IoT implementations. [36] while ensuring security necessitates thoughtful design inclusive of security from the start. [39, 41, 40] Selecting an appropriate architecture can make deploying IoT safely and secure in more places feasible. [36, 28, 35]

2.2.5 Industry Examples of IoT-enabled Training and Education

The presence of technology in classrooms has become the norm; today, it transcends taking notes on laptops and additionally includes more cutting-edge capabilities such as gamification. Gamification provides a means to motivate learning and to provide high-engagement, interactive instruction motivating collaboration, teamwork, communication, and goal-seeking using element of challenge and motivation to encourage positive learning behaviors. [48] Increasingly, *digital* gamification plays an intentional - rather than incidental - role in higher education. Gamified learning, in combination with game-based learning, can support higher education with great benefits across disciplines for teachers and students alike. [49]

Promising results in higher education have led to the exploration of digital gamification in industry, too, with application to professional education in disciplines such as engineering. [50] Studies have shown a positive effect by making subjects more manageable, and increasing student motivation to learn.

More recently, the idea of pairing existing gamification approaches with other emergent technologies such as the Internet of Things, XR, and easy-to-use game engines has started appearing in the industry with application to education. One example of such development is what PTC is doing for training, in using this methodology for worker training or remote worker supervision. Also, some efforts towards this direction have been made by Realism.io, which is an educational organization that creates virtual laboratories to expand student access to educational resources at low per-user cost and with minimal health and safety risks.

2.3 Environmental Studies Gamified Tool

2.3.1 Design and Development methodology of the Environmental Studies Gamified Tool - Methodology

Starting from a simple sketch and some notes (Figure 2.3) on the needed educational elements and 3D objects, we started to design our serious gamified tool with an expectation to simulate a somehow realistic environment [11]. In order to design our tool to be suitable for environmental studies educational purposes and thus, generate a meaningful [51] interactive learning experience to our students, we worked using the research-creation method [52].

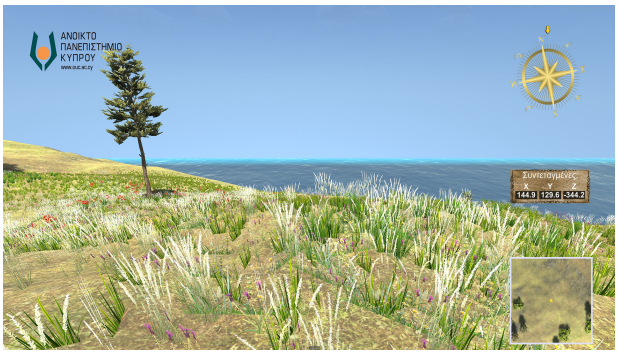
Figure 2.3: Initial sketch ideation and notes on the gamified tool

For the development part of our gamified tool, we worked using the Scrum model [53] of the Agile methodology of software engineering. Our team arranged meetings after specific elements were developed and stable builds were exported. We worked in sprints and continuously iterated [54] on development aspects until we reached a working beta build.

(a) Main Menu

(b) Loading Screen

(c) Main area screenshot 1



(d) Main area screenshot 2

Figure 2.4: Scene Views

2.3.2 Developing the Environmental Studies Gamified Tool

The gamified tool is built upon a powerful Game Engine, Unity [23]. As discussed in subsection 2.1.2, Unity is one of most popular engines in the field of game development and features a physics system suitable suitable for our case and most importantly, a great commercial add-on asset to generate the needed types of terrains. Below, we explain the development of the tool.

Levels - Scenes: The gamified tool contains a main menu (Figure 2.4a), a loading screen (Figure 2.4b) and a main area environment (Figures 2.4c, 2.4d). The main menu handles the scene transitions and the loading screen works as the waiting screen between the main menu and the main area environment.

Features and Mechanics: Once the gamified tool is loaded in the main environment, the user can explore in a **first-person perspective** (first person controller) a **virtual environment** of 4Km^2 (2Km x 2Km terrain). This terrain was developed using a purchased add-on. This way, we created a

land mass (including hills), **a lake**, **different kinds of plants** and **a forest**. We have also developed a **simulated weather system**.

The square area of the environment is also bounded with **invisible colliders**, placed near its limits (about ten meters inside each side). This way users always remain between the necessary area and there is no chance to "fall off" the level.

The tool also contains three user interface (UI) elements at the right side of the screen (Figures 2.4c, 2.4d). These include a **compass**, a **coordinate system viewer** and a **mini-map**. The compass helps the users identify the correct direction for their movements while the coordinate system viewer is providing the in-game coordinates and current location of the player's avatar (first-person camera). The mini-map just gives a small top-down preview of the neighboring area.

In addition to the mini-map that just gives an essence of a certain area, a more powerful **virtual drone system** was implemented too. This system was developed using RenderTextures and UI Panels and gives the opportunity to users for a top-down view of certain areas. Selecting different buttons from the keyboard, it can provide a larger or smaller predefined area (Table 2.1 and Figures 2.5a - 2.5g). An important characteristic of this feature is that these drones can move or rotate along with the users' movement.

Lastly, we also created a **teleport system** where users can change locations immediately and go to predefined coordinates where instructor-triggered events are taking place (see Subsection 2.3.3).

2.3.3 The IoT Implementation

After the main tool was developed as described in section 2.3.2, we developed an IoT implementation. To do so, for the hardware part, we used a breadboard, a microcontroller and a button. The microcontroller was powered via a micro-USB cable connected to a computer. The circuit schematic is presented in Figure 2.6 and was created with the tool "Fritzing" and a photo of the actual implementation is shown in Figure 2.7.

The microcontroller that we were using is named "Photon" and it is designed by the IoT company "Particle". It has a pre-installed WIFI shield and thus, after setting it up we managed to connect it

Drone	Area Covered	Keyboard Button
Drone View 1	$100m^2$ (10m x 10 m)	U
Drone View 2	$400m^2$ (20m x 20m)	I
Drone View 3	$1{,}600m^2$ (40m x 40m)	O
Drone View 4	$6{,}400m^2$ (80m x 80m)	P
Drone View 5	$14{,}400m^2$ (120m x 120m)	J
Drone View 6	$25{,}600m^2$ (160m x 160m)	K
Drone View 7	$40{,}000m^2$ (200m x 200m)	L

Table 2.1: Drone System and Button Configuration

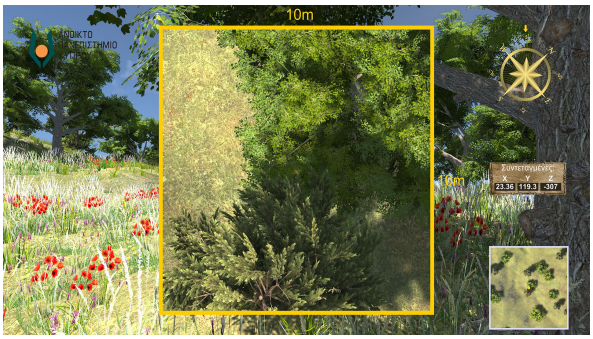
with the web-based IDE and the cloud platform of Particle.

After our device was connected online via WIFI, we programmed our microcontroller in C++, similarly to an Arduino and "flashed" it via the IDE. Our code can be found in Figure 2.8. In short, the code gets the value of the button and translates it into “ON” and “OFF” states. The “ON” state represents when the button is pressed and the “OFF” when it is not. When the circuit is powered on, these values are uploaded constantly to the cloud via WIFI with an interval of t_d=2000ms (2sec) between each measurement (Figure 2.9).

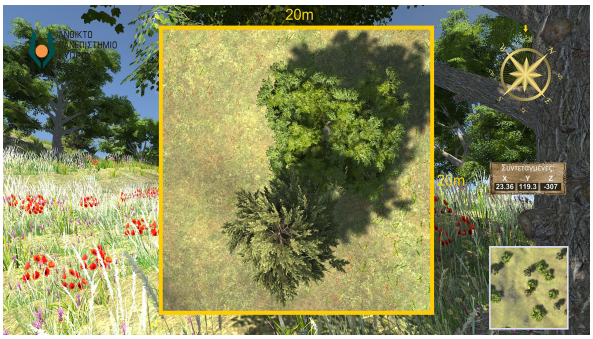
In order to get these values and import them into our gamified tool with C#, we are using the stream link provided. This link also contains an access token (shown but greyed out in Figure 2.9) and uses a security method [55] since our data are not public but private [56]. Practically, we are using the authorization method OAuth [57] which is pre-installed on server side. Using the REST API we are getting the values in the form of JSON (JavaScript Object Notation) files [58].

2.3.4 Overview of IoT-enabled Gamification

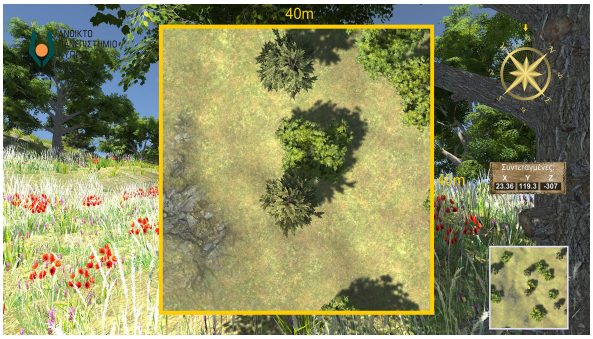
In the Figure 2.10, we can see how this integration works. While the IoT implementation is powered on and connected to the internet, the instructor presses or releases the button switch. Then, via the WIFI, this information (“ON” - “OFF”) is uploaded to the cloud platform. We have programmed



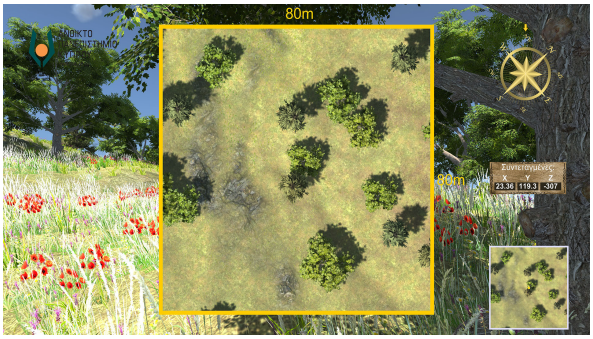
(a) Drone View 1 (10m x 10m area)



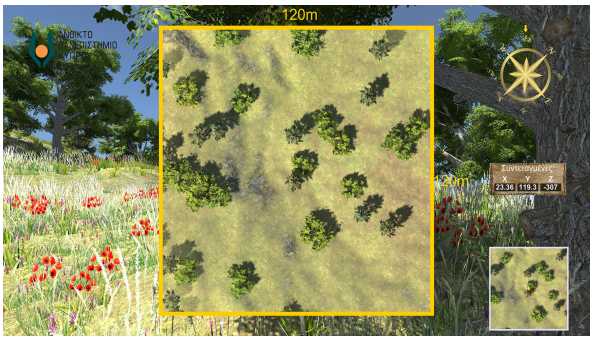
(b) Drone View 2 (20m x 20m area)



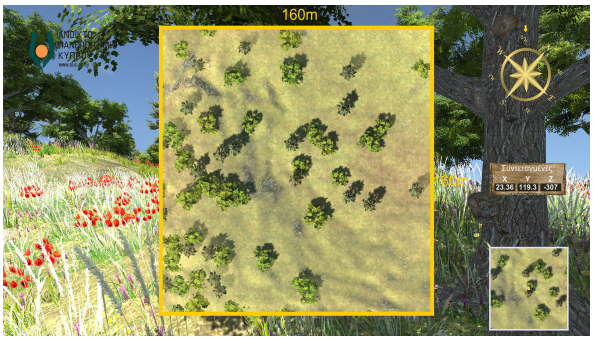
(c) Drone View 3 (40m x 40m area)



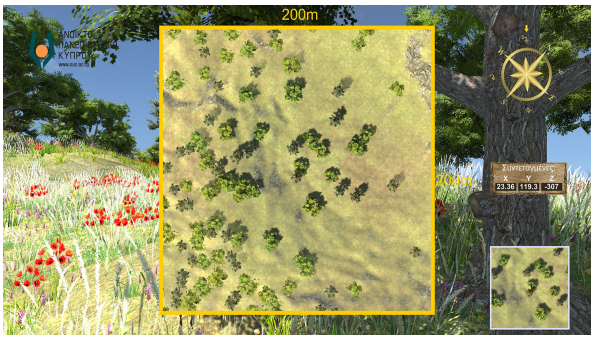
(d) Drone View 4 (80m x 80m area)



(e) Drone View 5 (120m x 120m area)



(f) Drone View 6 (160m x 160m area)



(g) Drone View 7 (200m x 200m area)

Figure 2.5: Various Drone Views

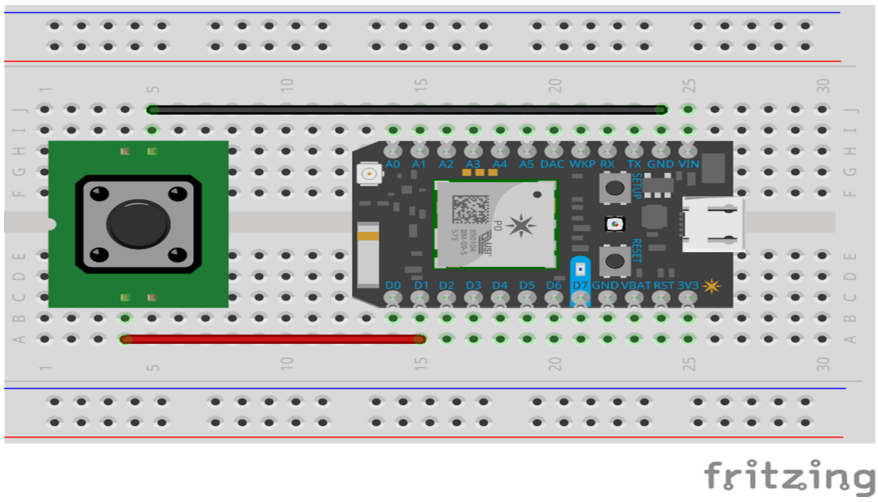


Figure 2.6: IoT implementation schematic

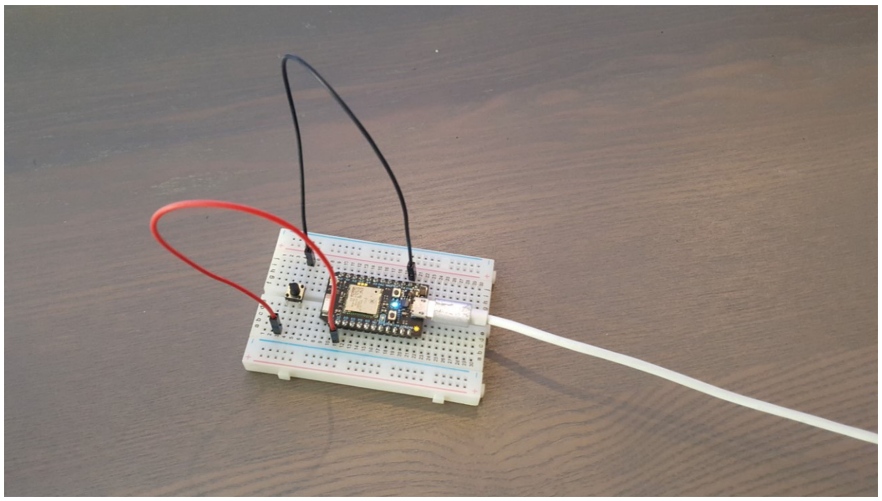


Figure 2.7: The IoT Implementation in reality

our gamified tool in C# to get the secure stream of data. In addition to the 3D environment, the UI elements, the audio and the other C# scripts needed for the tool's development, we are getting the "ON"-"OFF" states. These states are not used as text anywhere, but they are used as triggers, the "IoTriggers", as we named them. While developing the gamified tool, we have also created a fire particle system along with a course-related 3D world space image. Both the particle system and the image are Unity gameObjects that we had set to be inactive unless the tool gets the "ON" state indication. When the instructor presses the button, they enable these gameObjects and thus, both the fire and the course material show up. When they release the button and thus, the IoT

```
int led = D7;
int pushButton = D2;

void setup(){
    pinMode(led, OUTPUT);
    pinMode(pushButton, INPUT_PULLUP);
}

void loop(){
    int pushButtonState;
    pushButtonState = digitalRead(pushButton);
    //String Button_Status = pushButtonState;


 // Particle.variable("Button_Status", pushButtonState);


   if(pushButtonState == LOW){
        digitalWrite(led, HIGH);
        Spark.publish("Button_Status","ON",60,PRIVATE);
//        Particle.variable("Button_Status", pushButtonState);

        delay(2000);
   }
    else{
        digitalWrite(led, LOW);
        Spark.publish("Button_Status","OFF",60,PRIVATE);
  }
}
```

Figure 2.8: Microcontroller code

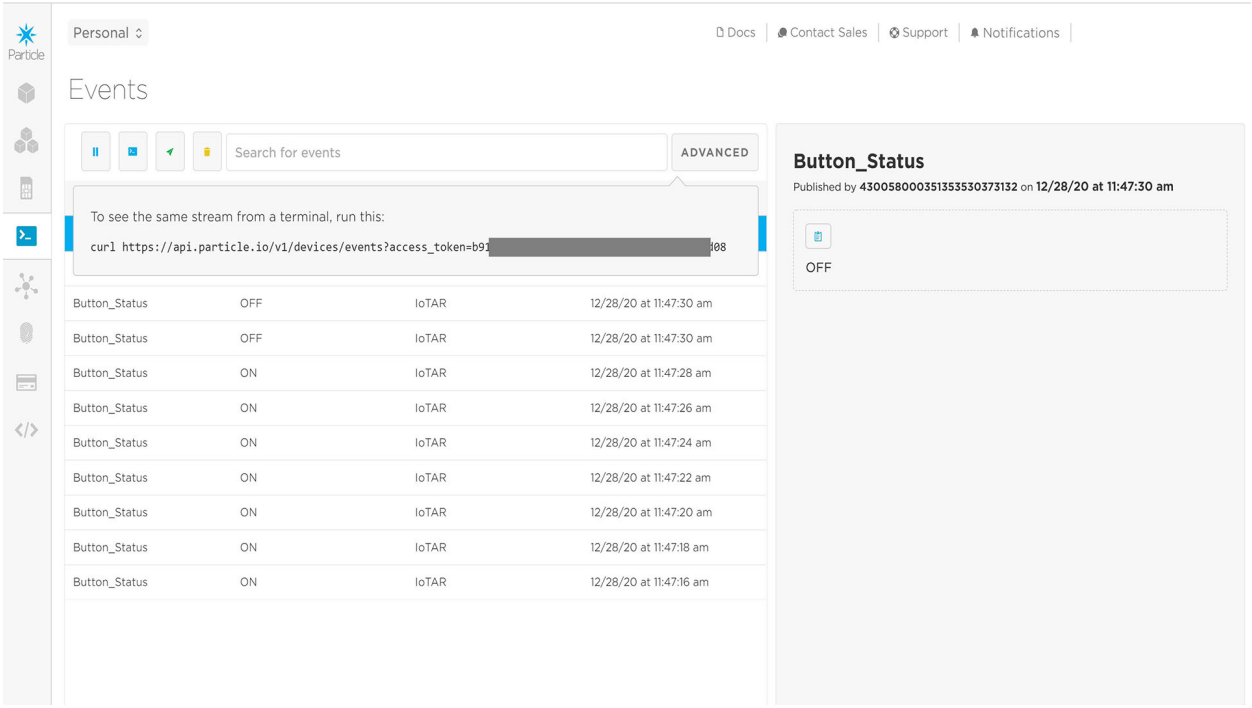


Figure 2.9: ON-OFF Values as uploaded to the cloud platform

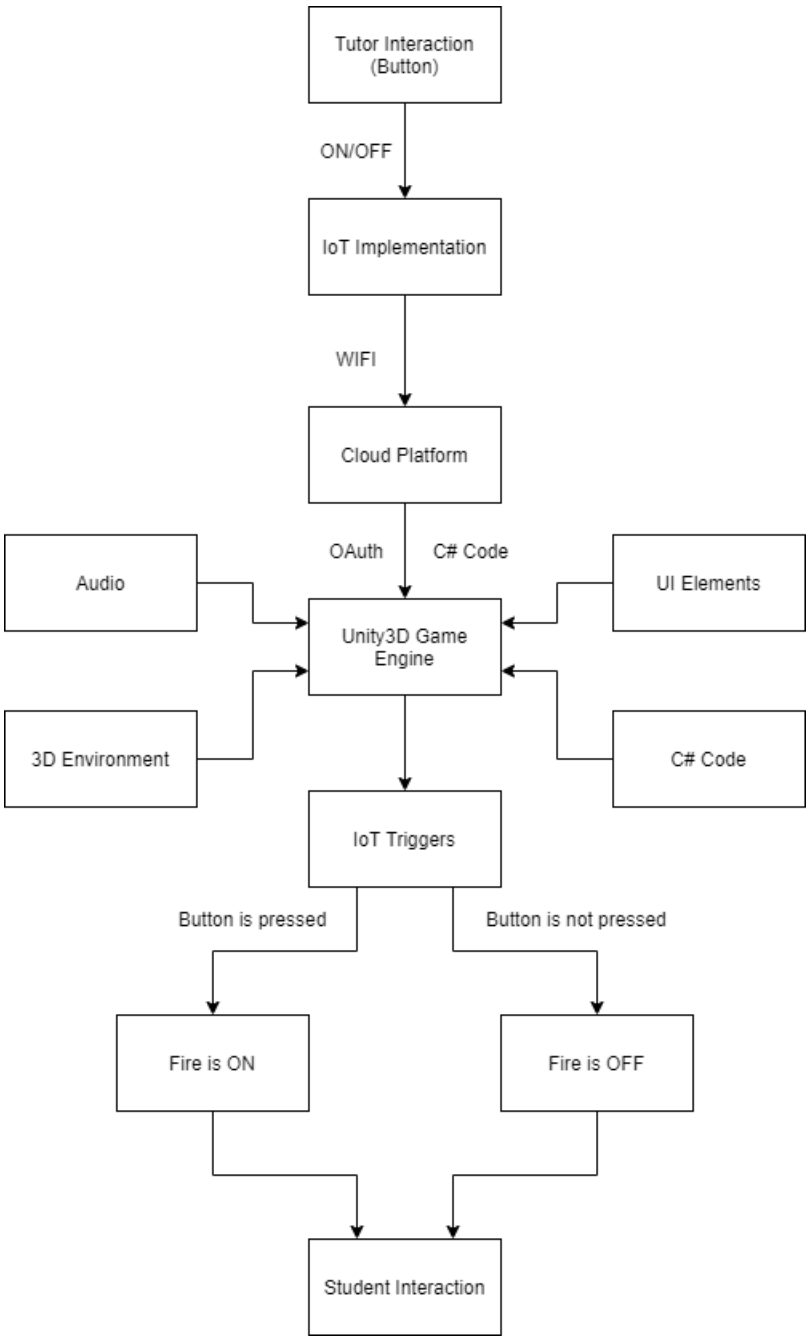


Figure 2.10: Holistic diagram of gamification and IoT integration

Figure 2.11: When the button is pressed, the fire starts burning the trees and the virtual book appears - IoTriggers

implementation is transmitting an "OFF" indication, these gameObjects are programmed to be set inactive once more and the virtual environment will return to its original state.

This way, we have on the one hand the **asynchronous** character of the tool, where students can explore the virtual environment or they can work and get measurements with the drone system for their assignments locally and at their own time and the safety of their houses. On the other had, we have transformed it into a very powerful **synchronous** tool for lectures using the IoT. Using this method, every student that uses the application and is located near the virtual coordinates that the fire incident is taking place can actually view the fire and the additional course material **simultaneously** with their classmates. That is why we had implemented the teleport system (see Subsection 2.3.2, to help students move immediately to the appropriate location for the IoTriggers. This way, a tutor can use the tool while lecturing and provide students with a very engaging interactive experience. We named this mode "pseudo-multiplayer", because even though it is a type of a multiplayer, it differs from a regular game's multiplayer mode (see Subsection 2.3.5).

2.3.5 Multiplayer vs IoT-enabled Pseudo-Multiplayer

Multiplayer is a common mode in modern games. When users are playing multiplayer, they are interacting with other users along with the game itself. Depending on the game, the users are viewing the other users' avatars and they can see them moving or performing various actions. Using modern game development techniques (excluding the game streaming services like Google Stadia [59] and others), users have locally installed games and after they are connected to a server and then a lobby, they are just streaming the position, rotation and scale attributes of other users in their lobby in order to be able to see them moving. At the same time, for other events viewable in multiplayer the Remote Procedure Calls (RPCs) are taking place. This way the packets transmitted between users are limited to the essential ones and thus, are easier to be handled over the network and the performance is better since the ping remains quite low.

The pseudo-multiplayer mode we implemented, does not require our users to be connected to a game server or a specific lobby. Every copy of the gamified tool works as an independent receiver of the IoT information. All students can simultaneously view the events when they are taking place, but even though they could be near or even at the same virtual coordinates with their classmates, they do not see each other. This way, they can attend a lecture and use the tool without even having distractions with each other and by getting a simultaneous by custom instructor-to-student experience.

2.3.6 The use of Environmental Studies Tool in Class

Once the development phase and the major beta testing of the gamified tool were complete, it was uploaded to the moodle-based online platform of the Open University of Cyprus, named "eClass" (Figure 2.12). Along with it a short manual explaining the game controls and the general usage was created and uploaded too. Shortly, after being available online to students, the tool was demonstrated to students during a synchronous meeting of the class in a Blackboard Collaborate session (Figure 2.13).

This tool is primarily designed for distance learning purposes. More specifically, it is in-

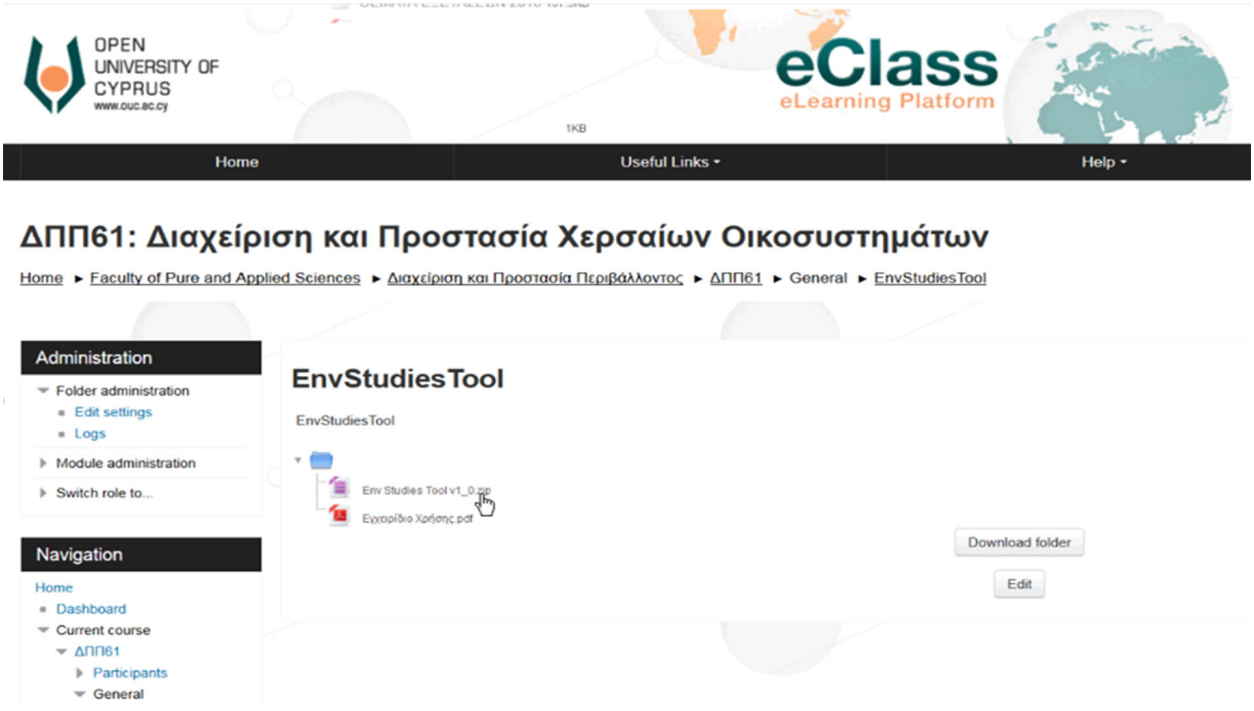


Figure 2.12: The gamified tool on eClass

corporated in Terrestrial Ecosystem Management module of the MSc program of Environmental Conservation and Management to facilitate online teaching of standard field-based methods for ecology and physical geography.

The use of the tool, provided solutions to a number of practical teaching issues and fostered a new learning experience for the students. As is the case in distance learning, students are located in many and different geographical areas. Some are living in major cities and smaller towns, with minimal or no immediate access to semi-natural landscapes. In addition, and since student fieldwork should be done in pairs for health and safety reasons, any related assignment to real world conditions remains problematic (e.g. vegetation sampling/survey).

At the same time, for an instructor it may be hard to create assignments and assess students with no “common ground" for measurements between their class. All the above issues were solved or improved using the environment studies tool.

Some types of activities which are now undertaken using the tool, directly or indirectly, in the aforementioned MSc module include:

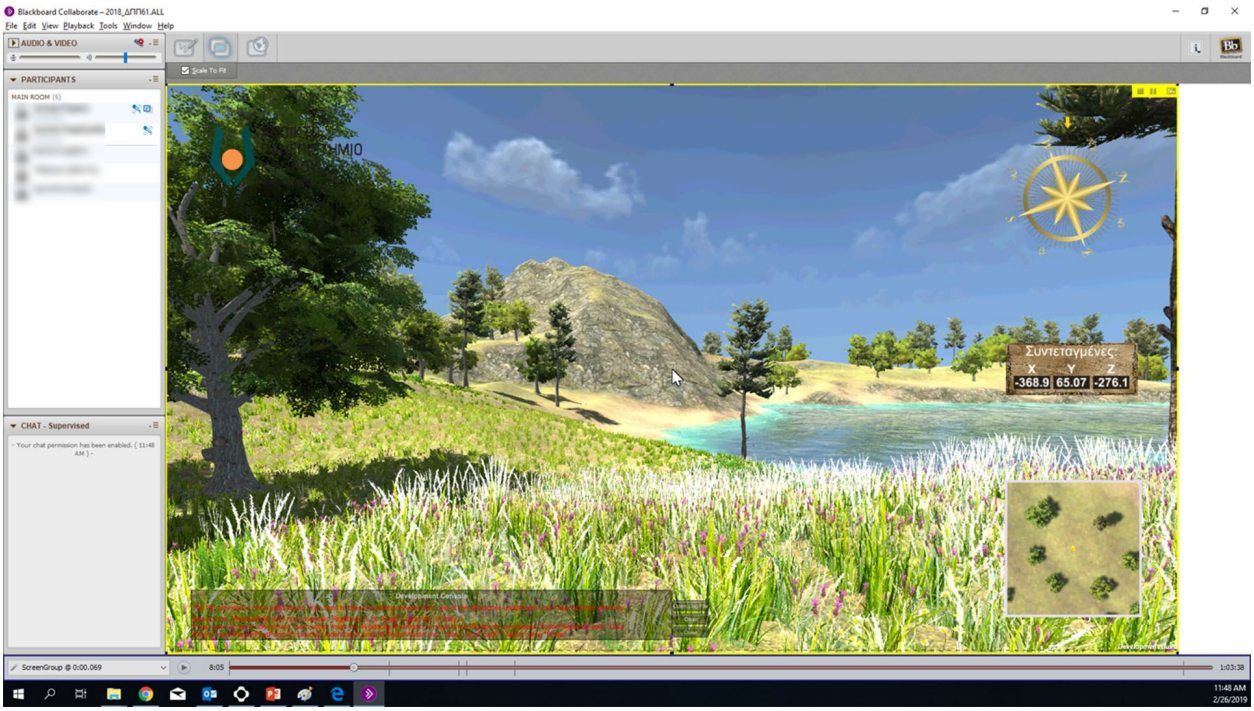


Figure 2.13: Live demonstration of the tool in Blackboard Collaborate

- **Measurement of species abundance:** presence/absence vs qualitative data: using any species of the ones present in the virtual terrain.
- **Biodiversity indices calculation:** using all different species identified in the virtual terrain.
- **Choice of sampling plots size and the minimal area curve:** This activity includes the seven set drone views and two different life forms of the virtual terrain (trees vs grasses).
- **Estimation of species population:** based on the use of pine trees.
- **Landscape Character Assessment:** using land cover and landform and assumed differences in geology of the virtual terrain.

2.3.7 Student Survey and Results

Before making the tool a part of the course core material, we needed to get some feedback from the students. So, after uploading the tool to the eClass platform students, we asked students to download it locally and test it on their computers. Then, we created a user experience (UX) and

anonymous research survey. The UX research methodology we used was survey-based. This is because we wanted to give the participants the flexibility of testing and providing their feedback at their own time, since they are students of an Open University where asynchronous methodology is the norm. Instead of having interviews on specific difficulties, bugs or errors they came across or suggestions the students had, we gave them the opportunity to include these at the end of our short questionnaire in a comment textbox.

Regarding our sample, the average age of the MSc students is 35 years, with a wide range of backgrounds, mostly on science related disciplines. All of them are working in government or private entities dealing with environmental management, located either in Cyprus or Greece in different geographical setting i.e. both in large urban centres and rural areas. The students which took part in this assessment did so on a voluntary basis.

By the time this research was conducted, the tool **had not been associated with learning objectives yet but students were informed about its goals and the IoT part was explained, so the focus is mainly on the use of the tool**. In some statements of the survey, students just give their insights on the upcoming learning goals and tool association. Specifically, students had to give their ratings from 1 to 5 (1 represents the lowest value) for the following statements. At the end, they were also given the option to write a few suggestions for updates and future releases or report any bugs they may have had and difficulties they may have faced. The results on survey statements are presented in Table 2.2 and we got feedback from four students.

- **Statement 1:** My interaction of the tool was pleasant.
- **Statement 2:** My exploration in the virtual environment was easy.
- **Statement 3:** The tool is quite related with the objectives of the course.
- **Statement 4:** The tool will help me better understand the course material.
- **Statement 5:** Having the tool at my own computer and without the need of getting outside for sampling purposes or for taking pictures making helps me in my overall learning process.

	Participant 1	Participant 2	Participant 3	Participant 4
Statement 1	★★★☆☆	★★★★☆	★★★☆☆	★★★☆☆
Statement 2	★★★☆☆	★★★☆☆	★★★☆☆	★★★★☆
Statement 3	★★★☆☆	★★★☆☆	★★★★★	★★★☆☆
Statement 4	★★★★☆	★★★☆☆	★★★★★	★★★★☆
Statement 5	★★★☆☆	★★★☆☆	★★★★★	★★☆☆☆
Statement 6	★★★★☆	★★★☆☆	★★★★★	★★★★★
Statement 7	★★★★★	★★★★☆	★★★★☆	★★★★★

Table 2.2: Survey ratings on various aspects of the tool

	Participant 1	Participant 2	Participant 3	Participant 4
This will improve my overall learning experience	✓	✓	✓	
My learning experience will remain the same				✓
I am not interested in this capability of the tool				

Table 2.3: Survey results on the IoT-related statement.

- **Statement 6:** The tool offers a new and innovative learning experience.
- **Statement 7:** The use of such tools in more courses would make my university experience more interesting.

For the IoT part, students had to answer the following statement and the results are presented in Table 2.3:

- **Statement:** The tool is programmed to connect with IoT implementations (including sensors) and the data visualization can take place within its virtual environment and this can be triggered by your instructor. Thus, during the synchronous lectures you will be able to view the changes in real time on your computers.

In some comments the participants left, they suggested adding more 3D features like roads,

rivers or more types of bushes and trees. Also, some other custom feature suggestions include more information on the types of plants or rocks. On the other hand, some problems they faced, had to do with some application crashes and slow player movement. This is explained by the fact that the tool is quite demanding in resources for graphics and some low-end computers may face difficulties in performance and thus, users may have a worse experience interacting with the tool than others. Further, despite having implemented a teleport system that moved the users to predefined locations, they requested an implementation that would freely move them in specific virtual coordinates by just placing the set of the needed coordinates. Lastly, since the drone system is a core element of the tool, some users suggested adding more views but smaller this time (2m x 2m and 4m x 4m).

Overall, as shown at the previous Tables (2.2, 2.3) and the feedback we got from the students was really good revealing that gamification and IoT-enabled Triggers will definitely improve the overall student learning experience.

2.4 Conclusions and Future Work

As we showcased, gamification can help both students and the academics in various ways. It can provide a really interactive and engaging environment for learning and at the same time, it provides the medium to teach or get measurements that would otherwise be a hard, dangerous or quite expensive task. In addition to this, the IoT offers the opportunity to get sensor measurements (or button indications in our case) that can be visually viewed (sensor data visualization) or can work as triggers, like our "IoTriggers". Using this IoT-enabled gamification approach, we showcased how it can be used effectively with a very positive impact in students' learning experience both in synchronous and in asynchronous learning context.

In the future, for the gamification part, we plan to further develop the 3D environment and add more elements like more kinds of plants and even animated animals created with the photogrammetry methodology. We also plan to include some student suggestions and include more information within this 3D space and add a new teleport system that will enable them to move freely just by entering values of the needed coordinates. Also, we will make some improvements

towards performance and load the 3D elements more efficiently for low-end computers.

Furthermore, we intend to create a Virtual Reality version of the tool for a full immersion. In terms of the IoT part, we are planning to start using the learning material aspect in a large scale adding a lot more gameObjects (like images) associated with the course curriculum. After doing this, we may use multiple IoT implementations that will be visualizing actual sensor data -and not just button triggers- installed on a real environment showing information like temperature and humidity.

2.5 Acknowledgements

This work has been accepted as a book chapter entitled "Handbook of Intelligent Techniques in Educational process" (Springer [60]) . Its use was showcased in collaboration with Beuth University of Applied Arts at Online Educa Berlin conference (OEB 2018) [61] and its impact in distance learning was mentioned among others at the International Technology, Education and Development conference (INTED 2019) [62]. Special thanks to Dr. Vogiatzakis (OUC Environmental Studies), Dr. Politopoulos, Dr. Siegel, for their input contribution and overall collaboration drafting and reviewing the book chapter. The developed application is currently used at the Open University of Cyprus and received the Gold Award at the Cyprus Education Leaders Awards 2019 for the category "Best Learning Experience".

CHAPTER 3

COMBINING XR & GAMIFICATION WITH THE INTERNET OF THINGS

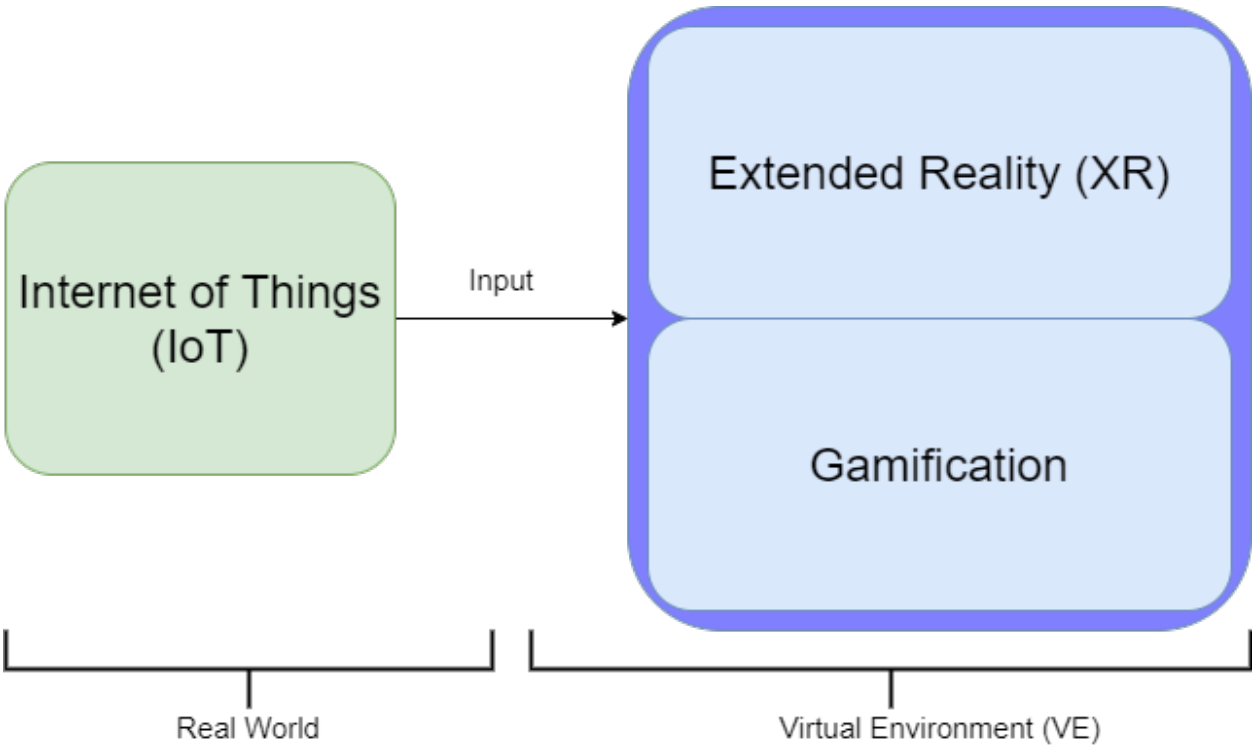


Figure 3.1: Combining XR & Gamification with IoT diagram

In this second part of our research, we demonstrate the combinination of IoT streamable data with XR and Gamification techniques (Figure 3.1) for use within the automotive industry. Here, sensors are installed into a car and its data are automatically uploaded to the Cloud as part of an IoT system. Using authentication protocols (OAuth2.0), we import low-latency streaming data into three different applications (two mobile AR/VR and one desktop application). A user-observer can visualize the real driver's behavior using these applications (Figure 3.2) as if it was a digital twin.The same data may then be used for other applications, such as performance analytics or condition monitoring. The suggested methodology in this implementation and this kind of combination between immersive technologies and XR/gamification is applicable in creating any type of low-cost digital twins.

Before starting showcasing the demonstration, the fundamentals of the reality-virtuality continuum and the terminology of XR will be presented.

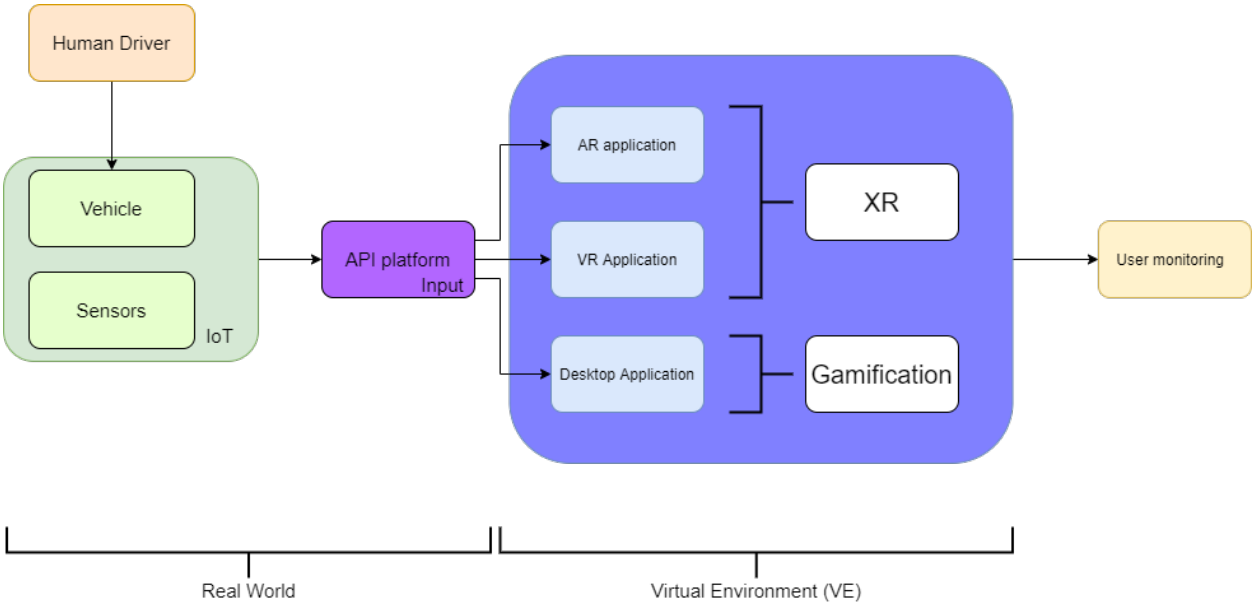


Figure 3.2: Avacar Layout: Combining XR & Gamification with IoT.

3.1 XR Fundamentals

3.1.1 The Reality-Virtuality Continuum

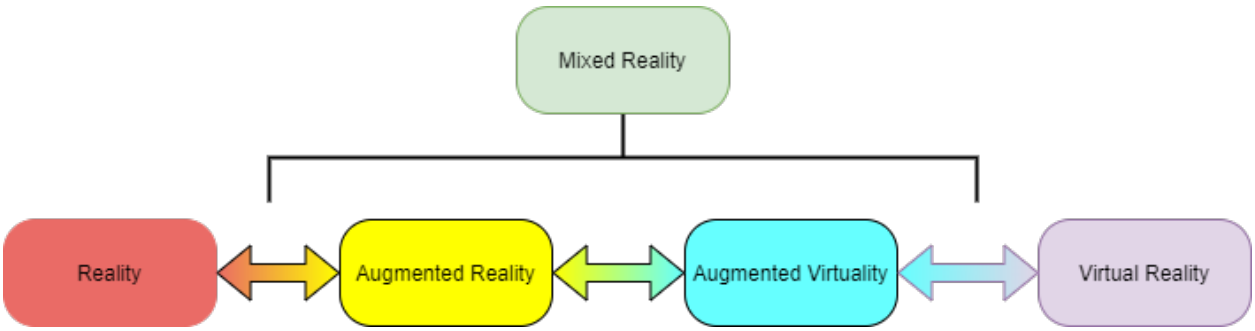


Figure 3.3: The Reality - Virtuality Continuum by Milgram

One of the first efforts to establish a continuum between real world and virtuality took place in 1994, when Paul Milgram and Fumio Kishino introduced the idea of the Reality-Virtuality Continuum [63]. This continuum (Figure 3.3) starts from a real environment (Reality) and ends

with a completely digital or virtual environment, specifically Virtual Reality. As a real environment, we identify an actual environment of the physical world, where Virtual Reality is a completely artificial space. Augmented Reality (AR) adds a layer of artificial objects atop the real world. According to Milgram, AR is a subset of Mixed Reality (MRfa) where the user is in the real world but also able to see artificial objects. Augmented Virtuality (AV) refers to the virtual world where a user can see objects from the real world. An example of Augmented Virtuality is when users see representative hands moving like their own hands inside a VR headset (for e.g. Oculus Quest). In Virtual Reality (VR), a user is inside an artificial environment completely isolated from the real world.

Although Milgram's continuum was accurate for the mid-90s, present technological provide slightly different definitions within this spectrum. The differentiation of technologies in products and applications has become more and more difficult to differentiate from those when Milgram pioneered the continuum.

3.1.2 Industry Perspective on Virtuality

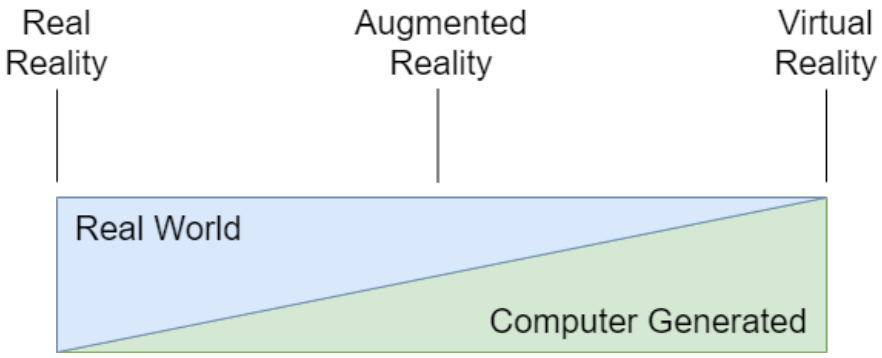


Figure 3.4: Google's Immersive Computing Spectrum

Technological giants like Google [64] (Figure 3.4) and Microsoft [65], (Figure 3.5) have provided their own definitions of Reality and Virtuality and developed both hardware and software (SDKs) supporting development.

Due to technology advances over the last decade, the differentiation between these technologies is increasingly difficult. Although VR's definition remains largely unchanged, since the user is

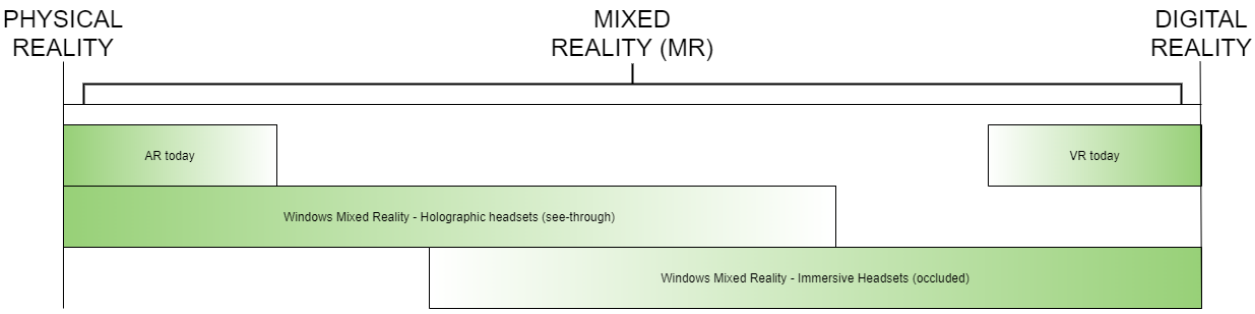


Figure 3.5: Microsoft's Mixed Reality Spectrum

isolated from the real world, AR and Mixed Reality (MR) leave more room for confusion. The main difference is the user interaction involved. In AR, a user can only view artificial enhancements like a 3D model, a text or a video, which appear on top of the physical environment, whereas in MR there is the option of interactivity with these enhancements or even combine artificial spaces.

3.1.3 Differences between VR, AR and MR

Based on the work of Ke, Xiang, and Zhang et al. [66], we may explore the differences of VR, AR and MR (Table 3.1) in terms of immersiveness, interactivity, multiperception and key technologies. Based on the latest advancements, the table 3.1) is further updated in two of its options. The first is that VR now offers real-time interaction (for e.g. seeing hands in VR as if they were real) and the second one is that the virtual spaces cannot be affected in AR. Virtual objects can be affected, though in AR we are within a physical rather than virtual space.

3.1.4 So, what is eXtended Reality (XR)?

Extended Reality or Cross Reality (XR) is the superset that brings AR, VR, MR under one group. The term X is interchangeable depending on what technology we are referring to (Figure 3.6).

3.2 Short Description of the Demonstration

This demo features a virtual 3D car model based off a vehicles data duplicate, an IoT-enabled Avacar. There are three embodiments including a desktop application (VirtualCar), Augmented

Properties	Characteristics	VR	AR	MR
Immersiveness	Virtual Space	✓		✓
	Physical Space		✓	✓
	Real-time Interaction	✓		✓
Interactivity	User and Virtual Model	✓	✓	✓
	User and Physical Space		✓	✓
	Physical Objects and Virtual Models			✓
Multiperception	Virtual Information	✓	✓	✓
	Real Information		✓	✓
	User's Physiological Information			✓
Key Technologies	Simulation Technology	✓	✓	✓
	Display Technology	✓	✓	✓
	Computer Graphics Technology	✓	✓	✓
	Human-Computer Interaction Technology		✓	✓
	Registration Tracking Technology		✓	✓

Table 3.1: AR/VR/MR comparison table

Reality (VirtualCarAR), and Virtual Reality (VirtualCarVR). The Avacar gets information from an external API serving information from a real or representative simulated vehicle, mirroring the sensors present in modern On-Board Diagnostic systems and the location data provided by a mobile devices GPS. In each application, the Avacar appears at specific locations around the world, using the Unity Game Engine to virtually mirror the behavior of the actual or simulated vehicle in real-time.

3.3 Purpose of Demonstration

The purpose of the presented research is to develop a system for mirroring the real or simulated data of a particular vehicle within a virtual environment. Vehicle data may be provided by means of an external file, or a secure API with access to raw vehicle sensor data provided by telemetry [67, 68], vehicle data captured from ambient sensors [69, 70, 71, 72, 68, 73], or simulated sensor data. The virtual environments are built using desktop, Augmented Reality, or Virtual Reality (AR-VR)

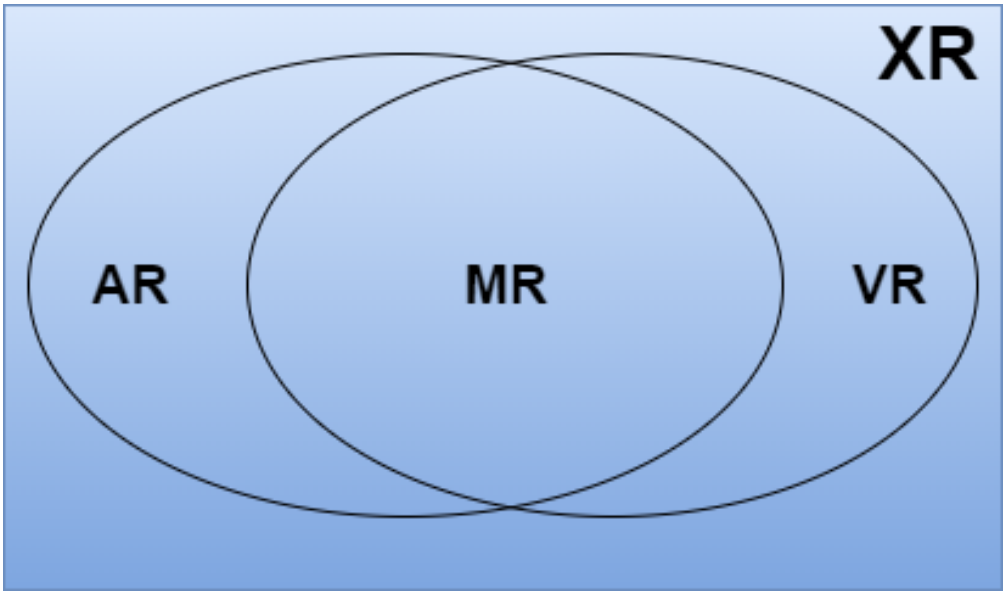


Figure 3.6: eXtended Reality (XR) Venn Diagram

concepts and based upon the Unity3D framework.

To optimize for presentation purposes and ensure the availability of real-time data during the demonstration window, this initial revision uses sensor data stored in a prerecorded file on a remote server. This JSON file [58] is a public sample and not connected to a real vehicle. Its attributes can be edited via external tools (FileZilla), and changes will propagate to the visualization.

Applying authentication methods (oAuth2.0) to our sample, we design a secure end-to-end Internet of Things model implementation [28] for twinning automobiles. At the same time, we created a desktop Application, an Augmented and a Virtual Reality application where users can visualize real-time changes in vehicle speed, engine speed, and location (location changes require the use of Unity3Ds editor mode and a software reload) to fetch the new 3D environment objects.

This application is grounded in the need for improved vehicle data visualization tools. For example, the applications may be used to monitor teenaged drivers, or remotely, for insurance agencies to supervise consenting drivers in order to reduce premiums. It is a first-of-its-kind graphical representation linked to connected cars, with the three applications reflecting contributions to the state of the art in data-informed vehicle visualizations. The underlying architecture has the potential to support applications improving transportation safety, efficiency, and comfort while

building consumer trust in connected cars.

3.4 Technology and Techniques

The demo consists of three parts, all of which are developed using the Unity3D game engine. To ensure performance, low-poly location based terrain is generated by Unitys Wrld plugin at startup. Changing the latitude and longitude within Unitys Editor mode passes new parameters to the virtual vehicle and generates new terrain topologies each time simulation restarts.

The sensor data shown on the upcoming applications dashboard gauge overlays comes from the JSON attributes velocity_kph and RPM (Figure 3.7). When these attributes are changed within the file, the gauge needles move to reflect the changes in real-time. This occurs as part of an on-frame update function wherein Unity updates its environment. The data are scraped using a C# script which can be modified to access both public and private (embedded authorization token) APIs [56].

```
{
    "id": "f61ba3d5-a68a-43eb-a731-0db871b4d3a3",
    "user": {
        "id": "U_ffd955ba63db5c25",
    },
    "type": "notification:speeding",
    "created_at": "2015-04-12T17:45:18.123Z",
    "time_zone": "America/Los_Angeles",
    "lat": 48.85564,
    "lon": 2.297013,
    "accuracy_m": 10,
    "created_at": "2015-04-12T17:45:01.123Z",
    "vehicle": {
        "id": "C_507d6f1bd6d9b855",
    },
    "device" : {
        "id": "021ac91c826b12eca99e685c"
    },
    "velocity_kph": 100,
    "RPM": 4000
}
```

Figure 3.7: Sample JSON file showing operation parameters for a representative vehicle.

3.5 Desktop Application (VirtualCar.exe)

The Desktop Application uses a two-camera model, with the first camera providing a 3rd person birds eye view of the Avacar. The second camera is inset like a mini-map, and provides a view from the virtual drivers seat. This provides a view of the Avacars dashboard, and is representative of what a desktop-based teen tracker application might look like. (Figure 3.8 and 3.12)

Figure 3.8: Desktop Application featuring a two-camera view. The Avacar is viewed from a 3rd person perspective, atop a procedurally-generated map of Cambridge, MA, USA. The minimap shows the drivers view of the dashboard, with real-time gauge updates indicating vehicle and engine speed.

3.6 Augmented Reality Application (VirtualCarAR.apk)

Augmented Reality applications can be both markerless [74] or image targeted. Our AR implementation uses an image targeted model based upon the cross-platform Vuforia plugin. This plugin uses a database of images and for this demo presentation was trained to follow a single marker image (Figure 3.9 and 3.12). When the device camera tracks this image, a three-camera view of the map, speedometer, and tachometer (Figure 3.10) automatically appears with the map orientation varying along with the orientation of the marker image.

Figure 3.9: This is VirtualCars Augmented Reality tracking image. It attains a five-star rating from Vuforias Web Platform.

Figure 3.10: The Augmented Reality Application features a three-camera view; including a 3rd person view of a procedurally-generated map (Cambridge, MA, USA) and dashboard gauge overlays. The tighter view of the map reduces the number of polygons necessary to render.

3.7 Virtual Reality Application (VirtualCarVR.apk)

Virtual Reality is a new and immersive technology that offers advantages to the automotive industry [75]. In our case, the VR implementation uses Google Cardboard in Unity. In this demo, the user sits in the drivers seat and views the virtual dashboard along with outside low poly buildings visible through the windshield. As with the desktop and AR applications, these buildings are also procedurally-generated. (Figures 3.11 and 3.12)

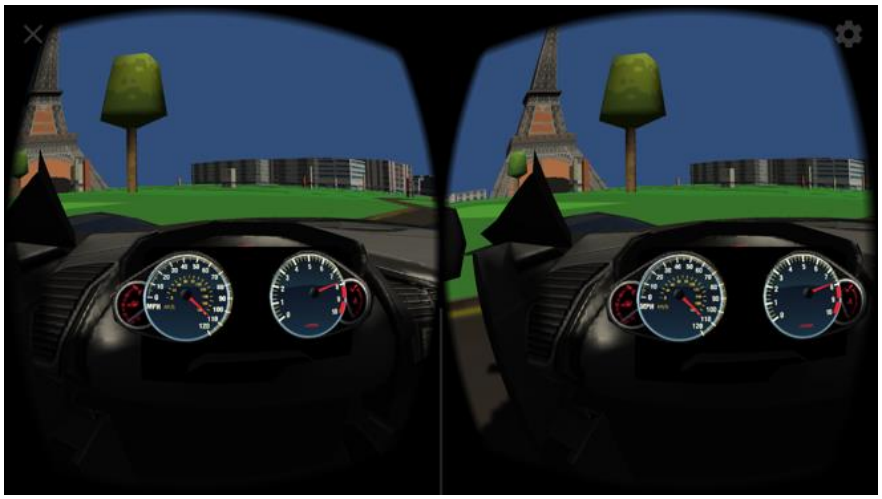


Figure 3.11: Virtual Reality Application featuring two-camera view. The drivers 1st-person view is located within the Avacar and shows the procedurally created map (Paris, FR). At the same time, the user can see the real time indications of Speed and RPM.

3.8 Future Updates

Future improvements will include additional gauges and indicators. The demonstration will also be connected to a live API, like CloudThink [76]. Once these applications are robust, additional indicators will be added to highlight potential problems based upon the real-time vehicle data, for example identifying engine misfires based on acoustic analysis [77].

3.9 Acknowledgements

This research product was a collaboration between the National Technical University of Athens (NTUA) and the Massachusetts Institute of Technology (MIT). Special thanks to Dr. Siegel (MIT Research Scientist at the time) and Dr. Politopoulos (NTUA). This work entitled “VirtualCar: Virtual Mirroring of IoT-Enabled Avacars in AR, VR and Desktop Applications” was presented at the demo track of the International Conference on Artificial Reality and Telexistence & Eurographics Symposium on Virtual Environments 2018 (ICAT-EGVE 2018) and received the Best Demo Award.

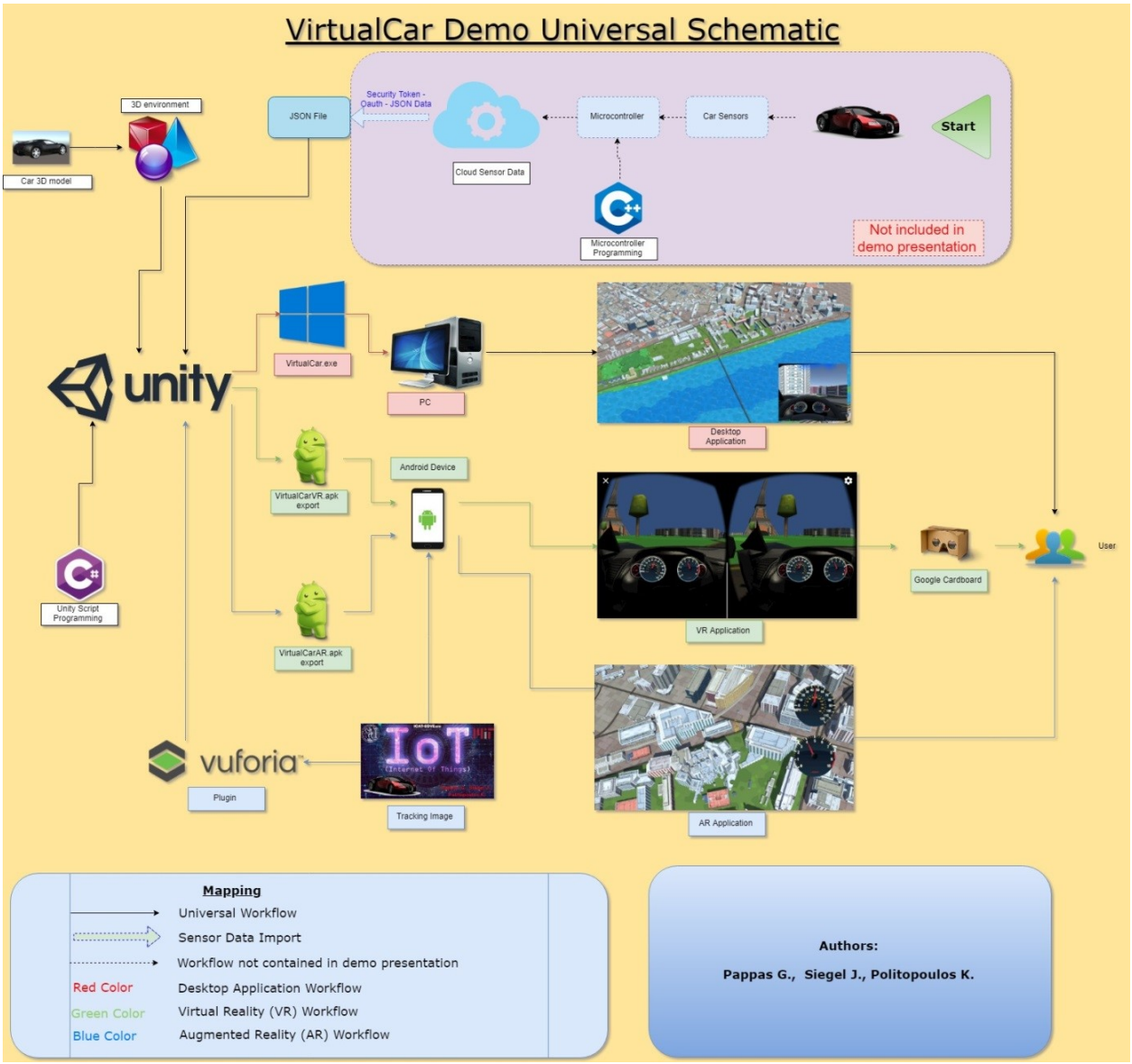


Figure 3.12: VirtualCar Demo Universal Schematic showcases the workflow of all the parts of the demo in a single diagram.

CHAPTER 4

COMBINING GAMIFICATION WITH ARTIFICIAL INTELLIGENCE

This next part of our research combines Gamification with Artificial Intelligence (Figure 4.1). This chapter is divided into two major subchapters, and focuses on the design and development of two gamified tools that may generate virtual and synthetic data that are suitable for creating AI models or can be validated by other pre-trained models created with real data. Both projects are related with to the field of automated or autonomous vehicles.

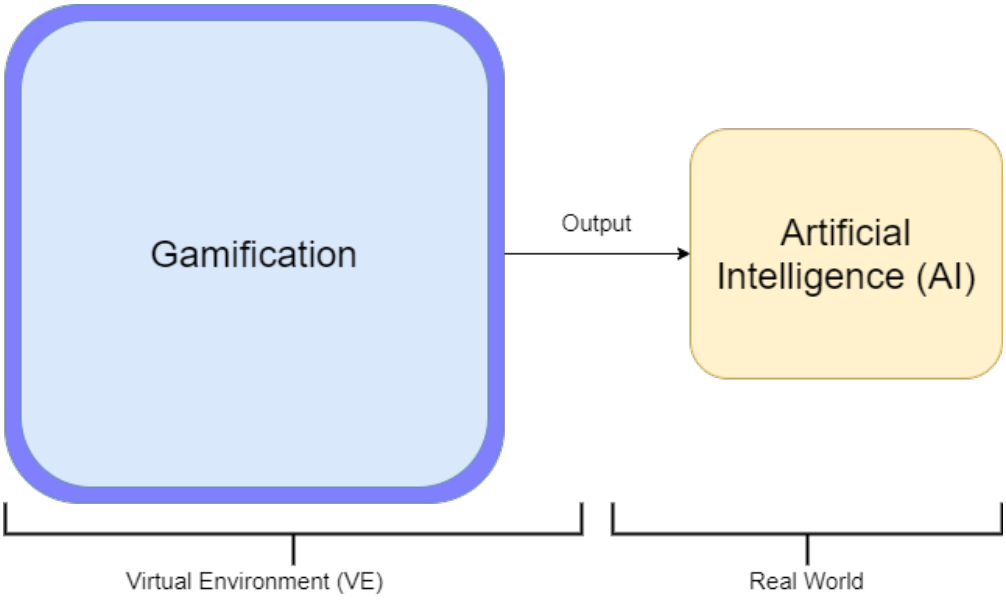


Figure 4.1: Combining Gamification with AI diagram.

The first implementation (see section 4.1) is a playable gamified simulator for capturing synthetic camera data for training a line following algorithm. The game features a virtual buggy that players can drive around a track with a gamepad. This research tool captures sample images and steering angles based on driver's behavior, logging speed, wheel angle, and screenshots that simulate a real RC car's camera to disk. Together with the human driving behavior, a Unity driving AI (rule-based) is developed using Ray-casting cameras and invisible colliders. This way, we managed to also implement an AI-to-AI training (Figure 4.2). After the sampling phase is over, we train a neural network for line following and test it successfully in game and later on a real-world self-driving

radio controlled car, demonstrating the value of the tool in generating transferrable data, particularly for tough-to-capture scenarios.

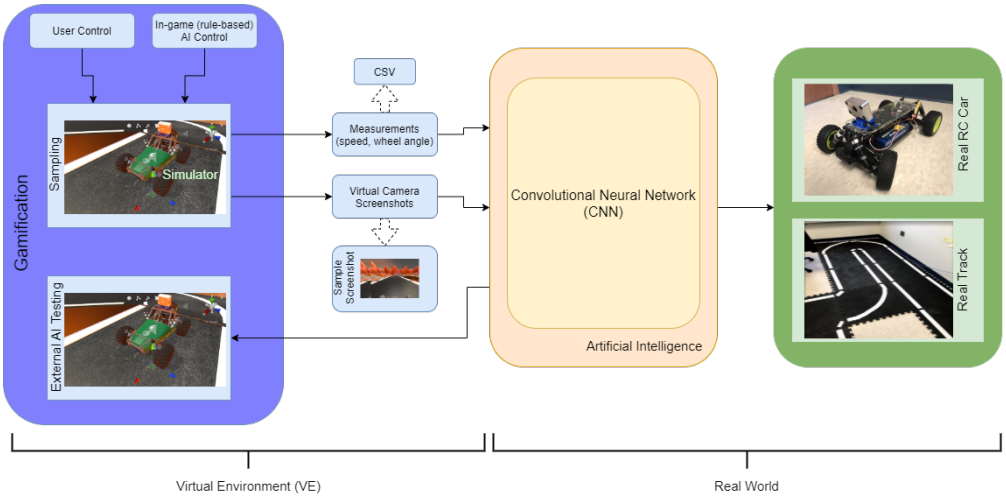


Figure 4.2: Gamified Simulator Layout: Combining Gamification with AI.

The second implementation (see section 4.2) simulates real-world LiDAR (Light Detection And Ranging) sensors. We used open-source tools and libraries and designed and simulated an easy-to-use, end-to-end platform for generation of point cloud data. We demonstrate a novel methodology of a single raycasting process for data sampling that results in accurate filtered data, taking advantage of known ground truth knowledge from Virtual Environments. Our tool has the ability to simulate any industry LiDAR with a simple configuration at the Options Panel. The generated data are validated with a pre-trained AI model (PIXOR), which is developed using real LiDAR point cloud data from KITTI Dataset. Transferrability from sim2real demonstrates that plausible point cloud data may be generated from our easy-to-use tool 4.29.

4.1 The Gamified Digital Simulator

4.1.1 Introduction

Deep Learning requires Big Data to learn behaviors from infrequent edge cases or anomalies. One application making use of Big Data is vehicle automation, where information representing diverse scenarios is necessary to train algorithms resilient to infrequent and high-impact events.

While manufacturers log fleet data (e.g. Tesla capture customer telemetry[78, 79]), it is difficult to validate that data are "clean" (e.g., common drunk, drowsy, drugged or distracted drivers might negatively impact a lane-holding algorithm). Even sober drivers with a poor understanding of the centerline of their vehicle may taint data. Further, the perception of "appropriate" morals and ethics is a driver of automated vehicle growth[80]. To meet the needs of robust, diverse and high quality data reflecting safe and ethical driving, the capture of known-good and large-scale data is necessary. "Wisdom of the crowd" requires *massive* scale, particularly in critical systems[81], so companies may instead capture data from costly trained drivers to maximize quality at the expense of quantity.

There is a need to capture large volumes of high-quality data with minimal supervision, and for inexpensive physical test platforms to validate real-world edge-case performance. This project proposes gamified simulation as a means of collecting bulk data for self-driving and a platform based on commodity hardware for real-world algorithm validation. Such a simulator could capture human- and AI-enabled data to train vision-based self-driving models and validate trained models' domain adaptation from simulator to real-world without explicit transfer learning.

Simulation is well-established[78], and small-scale hardware already tests algorithms in lieu of full-scale vehicles[82, 83], particularly for algorithms that may be dangerous or costly to test on a full-scale physical platform, such as collision avoidance or high-speed and inclement weather operation. Our approach furthers proven techniques to allow the generation of data from *unskilled* drivers and through the validation of resulting algorithms on lower cost and more widely accessible hardware than is used today. This enables large scale, rapid data capture and real-world model validation that may then be translated to costlier and higher fidelity test platforms including full-scale vehicles.

We develop such a system and demonstrate its ability to generate data from untrained human drivers as well as AI-controlled agents. The simulated environment mimics a real-world laboratory, and game mechanics implicitly motivate quality data capture: players compete for high scores or the best time to collect line-following training data. The result is a human-in-the-loop simulation

enabling inexpensive (<$500 platform + <$150 game asset costs), rapid (fewer administrative hurdles to clear for test approval), and high-quality (crowdsourced, configurable and supervised by game mechanics) data collection for certain use cases relative to conventional simulated and full-scale data capture environments and trained safety drivers. Our holistic solution integrates domain randomization, automated data generation, and thoughtful game mechanics to facilitate the bulk capture of useful data and transferrability to a near-disposable platform.

Our solution is unique in enforcing overt and latent rules in data collection: we learn from humans who have adapted to drive well in complex scenarios, while ensuring the data collected are of high quality by encouraging capture of useful data, automatically discarding low-quality data, disallowing specific actions, and enabling increased scale. Scoring mechanics encourage users to collect information for both common and edge-case scenarios, yielding valuable insight into common and low-frequency, high-risk events while the physical platform democratizes access to hardware-in-the-loop validation for budget-constrained developers.

This project's primary contributions are the toolchain comprising the simulator and physical validation hardware. These support a data-generating framework capable of simulating scenarios infeasible to capture in reality, with the benefit of semi-supervised (enforced by game mechanics) and crowd-sourceable (from untrained and unskilled users) data provided by humans testable on real images. Indeed, crowdsourcing has demonstrated success within transportation - for mode identification, parking location, and more[73]. The unique combination of implicitly supervisory game mechanics and crowdsourcing enables an end-to-end software and hardware training and testing platform unlike those used in contemporary research. Proof-of-concept Convolutional Neural Networks, while effective demonstrators of the simulator's ability to generate data suitable for domain adaptation to real hardware without domain adaptation, are peripheral contributions validating our claims of effective simulator and platform design.

4.1.2 Prior Art

4.1.2.1 Simulation and Gamification

One challenge in training AI systems is data availability, whether limited due to the scale of instrumented systems or the infrequency of low-likelihood events. To capture unpredictable driving events, Waymo, Tesla, and others collect data from costly highly-instrumented fleets[78, 79, 84]. To generate data at a lower cost, Waymo uses simulation to increase training data diversity[78] including not-yet-encountered scenarios. Simulation has been used to generate valuable data particularly for deep learning[21, 85]. However, simulated data abide by implicit and explicit rules, meaning algorithms may learn latent features that do not accurately mirror reality and may miss "long tail" events[86]. Though Waymo's vehicles have reduced crash rates relative to those of human drivers[87], there is room for improvement, particularly in coping with chaotic real-world systems operated by irrational agents. The use of thoughtfully-designed games lowers data capture cost relative to physical driving, with game mechanics, increased data volume, and scenario generation supporting the observation of infrequent events.

Researchers modified Grand Theft Auto V (GTA V)[88] to capture vehicle speed, steering angle, and synthetic camera data [89] and Franke used game data to train radio controlled vehicles to drive[90]. However, GTA V is inflexible, e.g. with respect to customizing vehicle or sensor, and data require manual capture and labeling.

Fridman's DeepTraffic supports self-driving algorithm development[91]. However, DeepTraffic generates 2D information, and while DeepTraffic crowdsources data, it does not crowdsource *human-operated* training data.

The open source VDrift [92] was used to create an optical flow dataset used to train a pairwise CRF model for image segmentation[93]. However, the tool was not designed with reconfigurability or data output in mind.

Another gamified simulation is SdSandbox and its derivatives[94, 95, 96] that generate steering/image pairs from virtual vehicles and environments. These tools generate unrealistic images,

may or may not be human-in-the loop, and are not designed to crowdsource training data.

CARLA[97] is a cross-platform game-based simulator emulating self-driving vehicles and offering programmable traffic and pedestrian scenarios. While CARLA is a flexible research tool, users are unmotivated to collect targeted data, limiting the ability to trust data "cleanliness." Further, there is no physical analog to the in-game vehicle. A solution with game mechanics motivating user performance and a low-cost physical test platform would add research value.

Other, task-specific elements of automated driving have been demonstrated in games and simulation, e.g. pedestrian detection [98, 99] and stop sign detection[100]. Some simulators test transferrability of driving skills across varied virtual environments[101]. These approaches have not been integrated with physical testbeds, which could yield insight into real-world operations and prove simulated data may be used to effectively train self-driving algorithms.

There is an opportunity to create an easy-to-use gamified simulator and associated low-cost physical platform for collecting self-driving data and validating model performance. A purpose-built, human-in-the-loop, customizable simulator capable of generating training data for different environments and crowdsourcing multiple drivers' information would accelerate research, tapping into a large userbase capable of generating training data for both mundane and long-tail events.

Such a simulator could create a virtualized vehicle, synthetic sensor data, and present objectives encouraging "good behavior" or desirable (for training) "bad behavior." For example, a user could gain points from staying within lane markers, or by collecting coins placed along a trajectory, or by completing laps as quickly as possible with collision penalties. Collected data would aid behavior-cloning, self-driving models in which input and output relationships are learning without explicit controller modeling. The simulator could be made variable through the use of randomized starting locations, dynamic lighting conditions, and noisy surface textures, compared with more deterministic traditional simulators.

Algorithms trained on the resulting synthetic data will be more likely to learn invariant features, rather than features latent to the simulator design. The result will be improved algorithms capable of responding well to infrequent but impactful edge cases missed by other tools, while a physical test

platform will validate model performance in the real-world and capture data for transfer learning (if necessary).

4.1.2.2 Training Deep Networks with Synthetic Data

To prove the simulator's utility, we generate synthetic images and train deep learning behavior cloning models, then test those models on a low-cost hardware platform. This section explores training models using synthetic data and porting models to the real-word.

Neural network training is data-intensive and typically involves manually collecting and annotating input. This process is time consuming[102] and requires expert knowledge for some labels[103, 104]. Generating high-quality, automatically labeled synthetic data can overcome these limitations. Techniques include Domain Randomization (DR) and Domain Adaptation (DA).

DR hypothesizes that a model trained on synthetic views augmented with random lighting conditions, backgrounds, and minor perturbations will generalize to real-world conditions. DR's potential has been demonstrated in image-based tasks, including object detection[105], segmentation[106] and $6D$ object pose estimation[107, 108]. These methods render textured $3D$ models onto synthetic or real image backgrounds (e.g. MS COCO[109]) with varying brightness and noise. The domain gap between synthetic and realistic images is reduced by increasing the generalizability of the trained model (small perturbations increase the likelihood of the model converging on latent features).

Synthetic data also facilitates 3D vision tasks. For example, FlowNet3D[110] is trained on a synthetic dataset (FlyingThings3D[111]) to learn scene flow from point clouds, and generalizes to real LiDAR scans captured in the KITTI dataset[112]. Im2avatar[113] reconstructs voxelized 3D models from synthetic 2D views from ShapeNet[114], and the trained model produces convincing 3D models from realistic images of PASCAL3D+ dataset[115].

Domain Adaptation (DA) uses a model trained on one source data distribution and applies that model to a different, related target distribution - for example, applying a lane-keeping model trained on synthetic images to real-world images for the same problem type. In cases, models may

be retrained on some data from the new domain using explicit transfer learning. In our case, we use DA to learn a model from a simulated distribution of driving data and adapt that model to a real-world context without explicit retraining.

For generating realistic data to support transfer learning without explicit capture of real-world data, Generative Adversarial Networks (GANs)[116] have been used to generate realistic data[117], 3D pose estimators[118] and grasping algorithms[119]. This work shows promising results, but the adapted images present unrealistic details and noise artifacts.

We aim to develop a simulator based on a game engine capable of generating meaningful data to inform Deep Learning self-driving behavior cloning models capable of real-world operation, with the benefit of being able to crowdsource human control and trusting the resulting input data as being "clean." These models clone human behavior to visually-learn steering control based on optical environmental input such as road lines seen in RGB images. Such a system may learn the relationship between inputs, such as monocular RGB camera images, and outputs, such as steering angles. In this respect, both a vision model and an output process controller are implicitly learned.

4.1.3 Why Another Simulator?

While there are existing simulators[120], our Gamified Digital Simulator (GDS) has three key advantages.

GDS is designed for ease-of-use and self-supervision. A user can collect data (images and corresponding steering angles) without technical knowledge of the vehicle, simulator, or data capture needs. This expands GDS's potential userbase over traditional simulators.

GDS provides an in-game AI training solution that can, without track knowledge, collect data independently. This allows some data to be generated automatically and with near-perfect routing as described in Section 4.1.4.10.

GDS offers a resource-efficient simulation, adapting the graphical quality level of scene elements depending on the importance of each element in training a model. Elements such the track and its texture (Section 4.1.4.1) are high quality whereas the background (Section 4.1.4.3)

uses low-polygon models and low-resolution textures. This reduces the processing power needed to run the GDS, expanding its applicability to constrained compute devices.

4.1.4 The Gamified Digital Simulator: Development Methodology

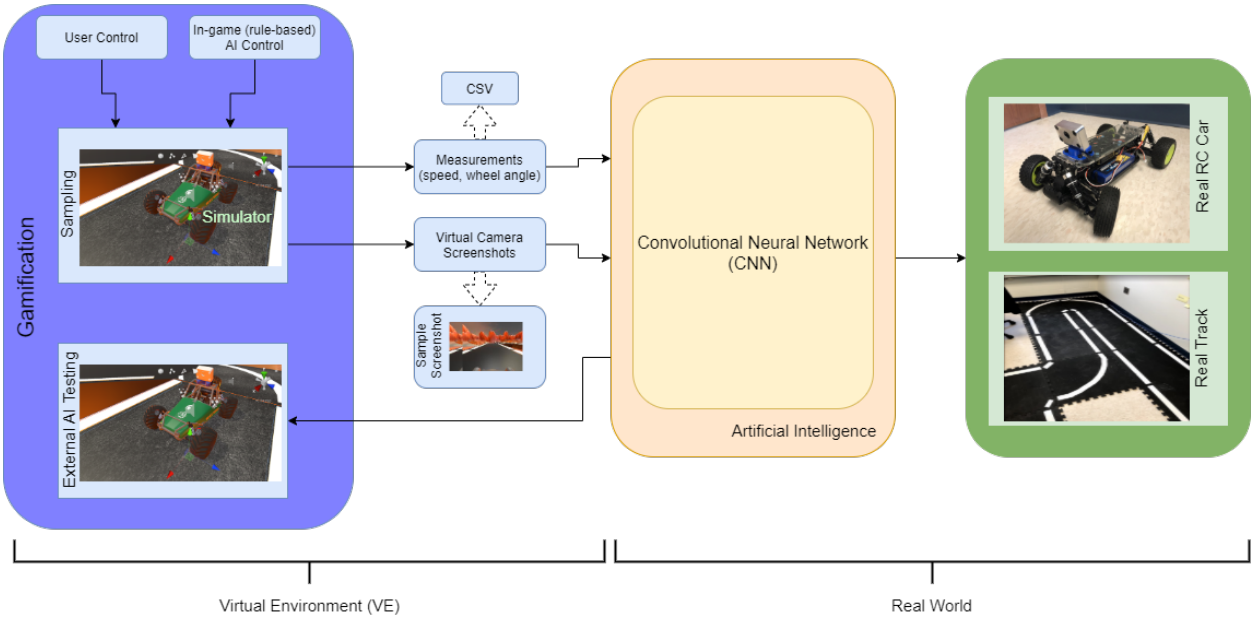


Figure 4.3: The end-to-end toolchain couples a virtual environment with a real-world platform and track setup. Synthetic images, trained by in-game AI and/or human operators, inform self-driving models. These models use behavior cloning to mimic observed behaviors without explicit system modeling.

This section details GDS's development. In Figure 4.3, we overview the GDS toolchain. The virtual car can be driven both by a human (using a gamepad) or a rule-based AI. The simulator generates two types of data: a) images from a virtual camera simulating the physical vehicle and b) a CSV file with speed and steering angle details. To test that these virtual data are effective for training behavior cloning algorithms, we capture data and train models to operate a virtual vehicle agent in the GDS environment and test this model on the physical platform to determine whether the learned models are effective in a different (physical) environment.

The GDS is built upon the cross-platform Unity3D Game Engine. The game consists of four scenes: a) Main Menu, b) Track Selection, c) User Input Mode and d) AI Mode. The Main Menu and Track Selection scene canvases and UI elements help the user transition between modes. From

the Track Selection scene, a user selects one of two identical playable scenes (tracks/environments), one with a human-operated vehicle and the other with an AI-operated vehicle. The AI Scene utilizes an in-game AI or an external script, e.g. running TensorFlow[121] or Keras[122] AI models, as described in Sections 4.1.4.10 and 4.1.7.

We first explore the application design, and then describe the methodology for transforming the game into a tool for generating synthetic data for training a line following model. Gamification allows non-experts to provide high-volume semi-supervised training data. This approach is unique relative to conventional simulation in that it provides a means of crowdsourcing data from goal-driven humans, expediting behavior cloning from the "wisdom of the crowd."

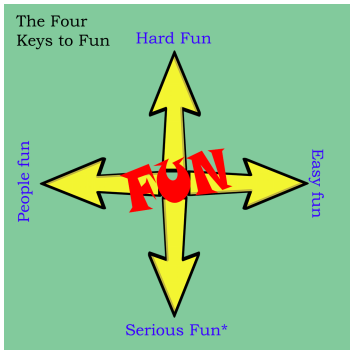


Figure 4.4: "The Four Keys to Fun" by Nicole Lazzaro (simplified). GDS belongs primarily to the "serious fun" category[1].

While designing the GDS, we considered it as both a research tool and a game with purpose [123, 124] and therefore include elements to enhance the user experience (UX). The fun is "serious fun" (Figure 4.4), as defined by UX expert Nicole Lazzaro[1, 15], where users play (or do boring tasks) to make a difference in the real world.

While the simulator cannot be released open-source due to license restrictions on some constituent assets, it is capturing data for academic studies related to training AVs and human-AV interaction and these data will be made available to researchers. The detail provided in this section should allow experienced game developers to recreate a similar tool without requiring extensive research. For researchers interested in implementing a similar software solution, the authors are happy to share source code with individuals or groups have purchased the appropriate license to

the required assets - please contact the authors for more information.

4.1.4.1 Road Surface

Figure 4.5: The initial test track was rectangular and designed as a simple test for the simulator and data collection system.

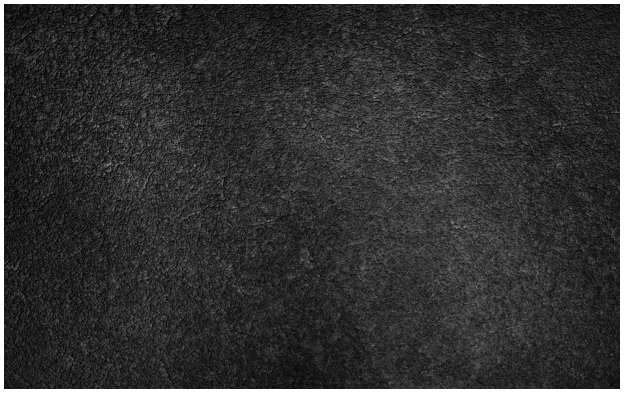


Figure 4.6: This asphalt texture was tessellated across the road surface in later versions of the simulator.

A road surface GameObject simulates a test track (Figure 4.5). We developed modular track segments using Unity's cuboid elements and ProBuilder (Figure 4.7). Modules were given a realistic texture (Figure 4.6) that changes based upon ambient lighting and camera angles, minimizing the likelihood that the neural network learns behaviors based on tessellated edges.

To reduce trained model overfit, we developed "data augmentation" using Unity's *lerp* function to add Gaussian noise creating speckles appearing over time. A second script manipulates the in-game lighting sources randomly so that vehicles repeating trajectories capture varied training

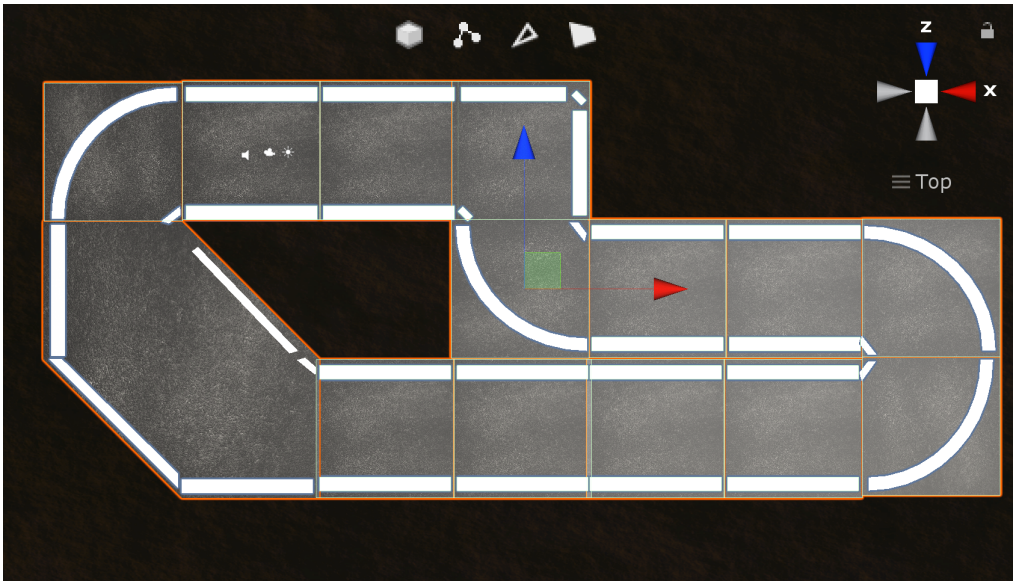


Figure 4.7: The track comprises adjacent modular components.

data. Both techniques support Domain Randomization and improve generalizability for domain adaptation (e.g. sim2real).

4.1.4.2 Game Mechanics

Gamification encourages players to abide by latent and overt rules to yield higher-quality, crowd-sourcable data.

On the rectangular track, we created GameObject coins within the lane markers. Players were intrinsically motivated to collect coins for a chance to "win" (Figure 4.8, though this approach allowed non-sequential collection, with users exiting the track boundaries and reentering without penalty. While we could disregard non-consecutive pickups, failing to do so would contaminate data.

We subsequently developed a system of colliders (green in Figure 4.9) invisible to the camera prevent buggy from traveling outside the white lane makers, ensuring that collected data are always within the lane, reducing contamination. Crash data and preceding/trailing frames were also eliminated from training data.

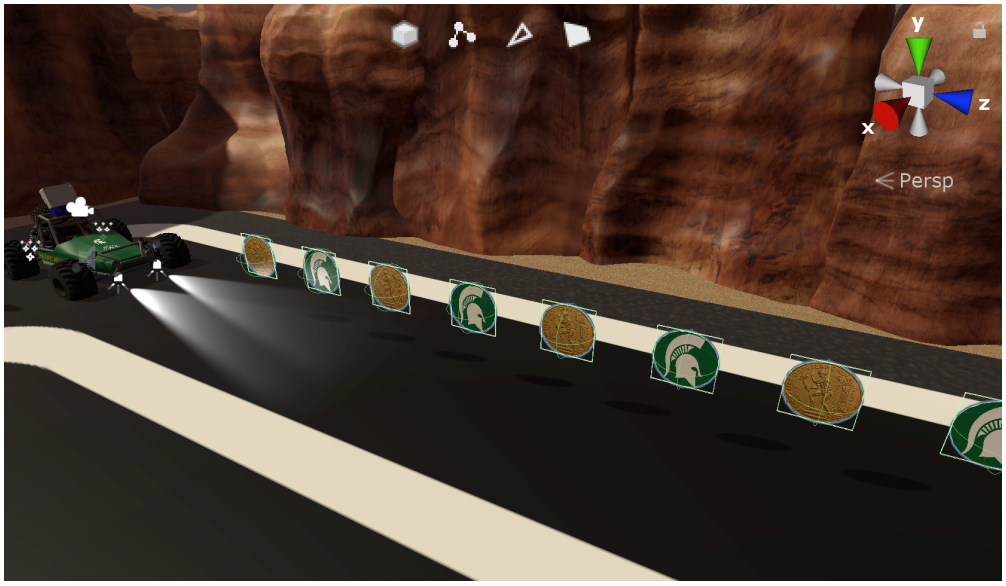


Figure 4.8: An early version of the GDS featured collectable coins worth points to incentive the driver to stay on the road.

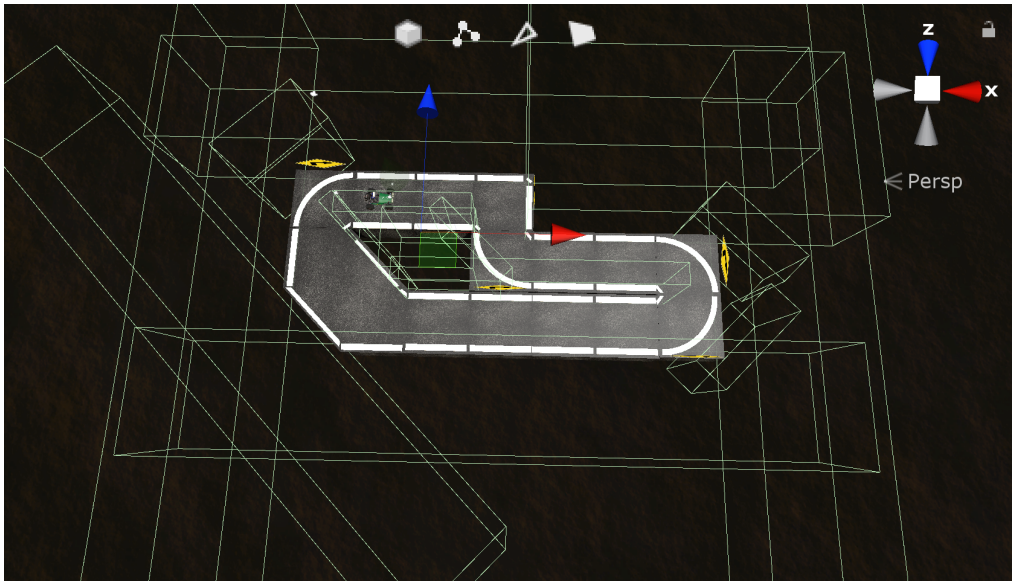
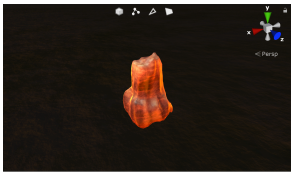


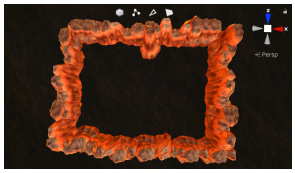
Figure 4.9: We developed an invisible collider system to implicitly force drivers to stay on the road surface.

4.1.4.3 Environment

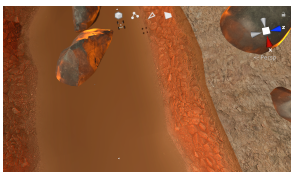
The game has two purposes: to create a compelling user experience (UX) conducive to long play, and to create dynamic conditions to enhance domain randomization and prevent the neural network model fitting to environmental features.



(a) Canyon 3D model



(b) Mountain View

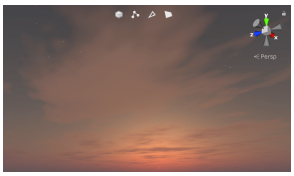


(c) Lava stream terrain

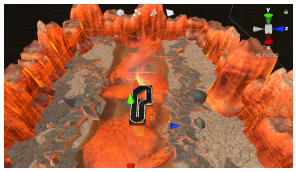
(d) Lava particle effect

(e) Terrain prior to earthquake

(f) Terrain after earthquake



(g) Night Skybox



(h) Combined elements, showing scale of track

Figure 4.10: These figures show the elements comprising the simulated environment and surroundings.

The game environment is a series of GameObjects and rendering lighting parameters. Objects include rocky surfaces and canyon models (Figure 4.10a) scaled and arranged to create a rocky mountain (Figure 4.10b). We included assets from the Unity Asset Store to excite players and create dynamic background images. These include a lava stream particle effect (Figure 4.10c and Figure 4.10d) and animated earthquake models (Figure 4.10e, Figure 4.10f). Combined assets create the experience of driving near a volcano. The dynamic background provides "noise" in training images, ensuring the neural network fits to only the most-invariant features. Finally, a night Skybox is applied (Figure 4.10g). Game environment elements deemed non-critical to the simulation are low-polygon for efficiency.

The perceived hostility of the environment implicitly conveys the game mechanics - the buggy must not exit the track boundaries, or it will fall to its doom. This mechanic allows users to pick up and play the game without instruction.

4.1.4.4 Representative Virtual Vehicle

The virtual vehicle is a multi-part 3D model purchased from Unity's Asset store and customized with CAD from the physical vehicle's camera mount.

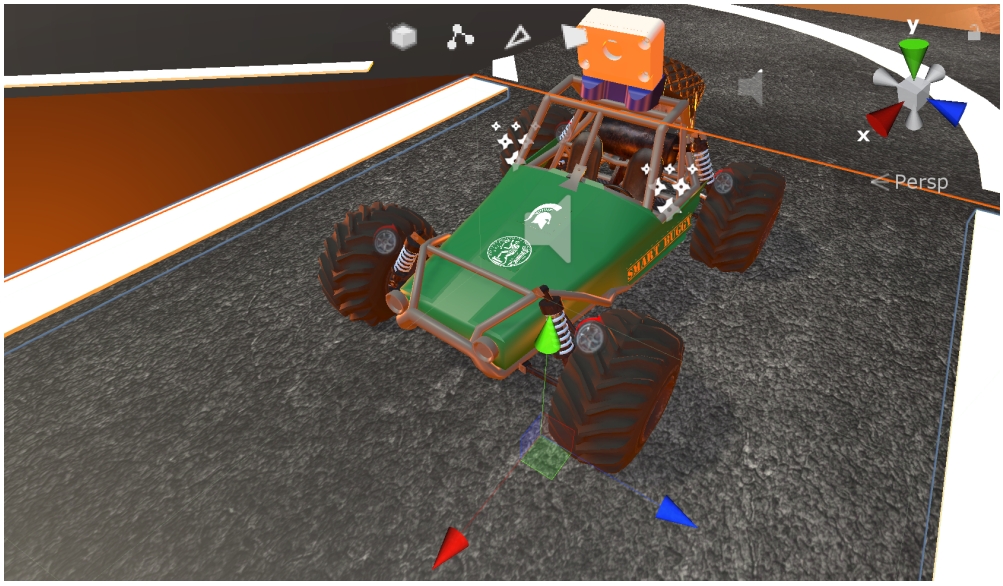


Figure 4.11: Each wheel has its own controller running a high-fidelity physics model. This allowed us to control steering angle, friction coefficient, and more.

To make the buggy drivable, we created independent wheel controllers using a plugin that simulates vehicle physics. Using this model, we set parameters including steering angles, crossover speed, steering coefficient, Ackermann percentage, flip-over behavior, forward and side slip thresholds, and speed limiters. These parameters allowed us to tailor the in-game model to mimic the physical vehicle.

4.1.4.5 Simulated Cameras

The GDS uses a multi-camera system to simultaneously render the user view and export training data. The user camera provides a third person perspective, while a second camera placed at the front of the buggy simulates the RC vehicle's real camera in terms of location, resolution and field of view (FoV). This camera generates synthetic images used for training the neural network model and is calibrated in software. A third camera is placed above the vehicle, facing downwards. This camera provides an orthographic projection and, along with a RenderTexture, creates a mini-map.

Two additional cameras behave akin to SONAR or LiDAR and calculate distance to nearby objects using Unity's Raycasting functionality. These cameras support in-game AI used to inde-

pendently generate synthetic training data (Section 4.1.4.10).

The location of the cameras is linked to the buggy's body. Each camera has differing abilities to "see" certain GameObjects as determined using Unity Layers. For example, the primary camera sees turn indicators directing the user, but the synthetic forward imaging camera ignores these signs when generating training data. In the coin demo, the user could see the coins, but these were invisible to the data-generating camera.

4.1.4.6 Exported Data

We attached a script to the buggy to generate sample data at regular intervals, including a timestamp, the buggy's speed, and the local rotation angle of the front wheels. This information is logged to a CSV in the same format used by the physical platform. Another script captures images from the synthetic front-facing camera at each timestep to correlate the steering angle and velocity with a particular image. The CSV file and the images are stored within the runtime-accessible StreamingAssets folder.

4.1.4.7 Simulator Reconfigurability

Figure 4.12: When the simulator is connected to a multi-display host, the second display allows users to adjust options in real-time. This greatly speeds up tuning the buggy model to match the physical model, and helps make the simulator more extensible to other vehicle types.

In order to simulate the configuration of the physical vehicle, we needed to iterate tests to converge on parameters approximating the physical vehicle. To speed the process of tuning the simulator for other vehicles, options can be changed by editing an external file. Multi-display users can configure options on a second monitor (Figure 4.12). Configuration includes:

1. **Low speed steering angle** - the absolute value of the maximum angle at which the center of the steered wheels is maximum when the vehicle is in low-speed mode
2. **High speed steering angle** - same as above, but in high-speed mode
3. **Crossover speed** - the speed at which the vehicle changes from low- to high-speed mode
4. **Steer coefficient (front wheels)** - the steering multiplier between the steering angle and the actual wheel movement
5. **Steer coefficient (back wheels)** - same as above. Can be negative for high-speed lane changes (translation without rotation)
6. **Forward slip threshold** - slip limit for transition from static to sliding friction when accelerating/braking
7. **Side slip threshold** - same as above, but for steering
8. **Speed limiter** - maximum vehicle speed allowable (also reduces available power to accelerate)
9. **Vertical Field of View** - in degrees, to match physical vehicle camera
10. **Sampling Camera Width** - ratio of width to height of captured image
11. **Sampling Rate** - rate, in *Hz*, of capture of JPG images and logging to CSV file

We use another external file to set the starting coordinates of the buggy (Figure 4.13), helping test humans and AI alike under complicated scenarios (e.g. starting immediately in front of a right

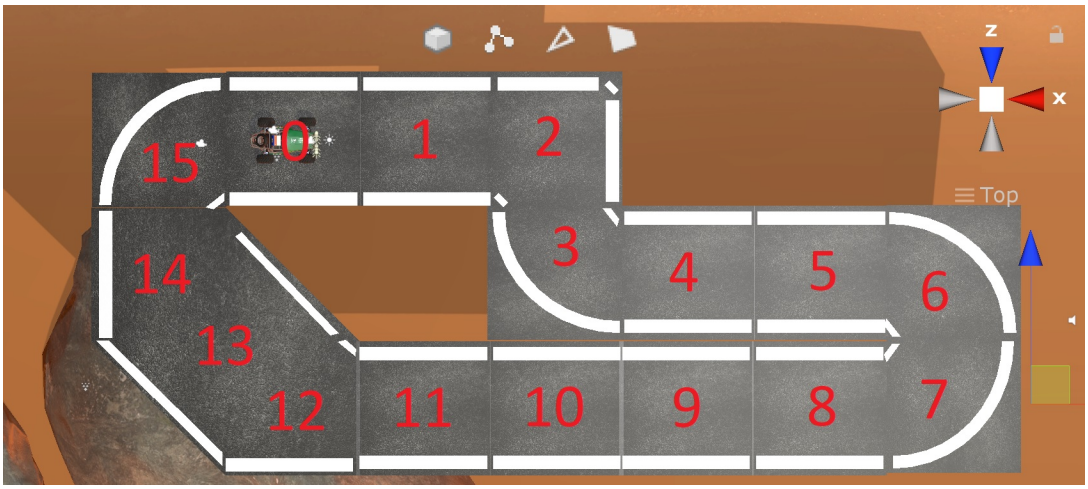


Figure 4.13: This figure shows the starting position options selectable from an external file. Each number refers to the center of the tile, while a starting angle can also be passed as a parameter to the game engine.

turn where the horizontal line is exactly perpendicular to the vehicle, or starting perpendicular to the lane markers).

4.1.4.8 2D Canvas Elements

2D GameObjects show the buggy's realtime speed and wheel angle. There are also two RenderTexture elements, one displaying a preview of the synthetic image being captured (the "real cam" view) and one showing the track on a "minimap." Throttle and brake status indicators turn green when a user presses the associated controls (Figure 4.14). This feature makes it easier for newcomers to quickly capture useful training data.

4.1.4.9 Unity C# and Python Bridge

We test models in the virtual environment before porting them to a physical vehicle. Deep learning frameworks commonly run in Python, whereas Unity supports C# and JavaScript.

We developed a bridge between Unity and Python by allowing GDS to read an external file containing commanded steering angle, commanded velocity, and AI mode (in-game AI – using the cameras described in Section 4.1.4.10, or external AI from Section 4.1.6, where the commanded

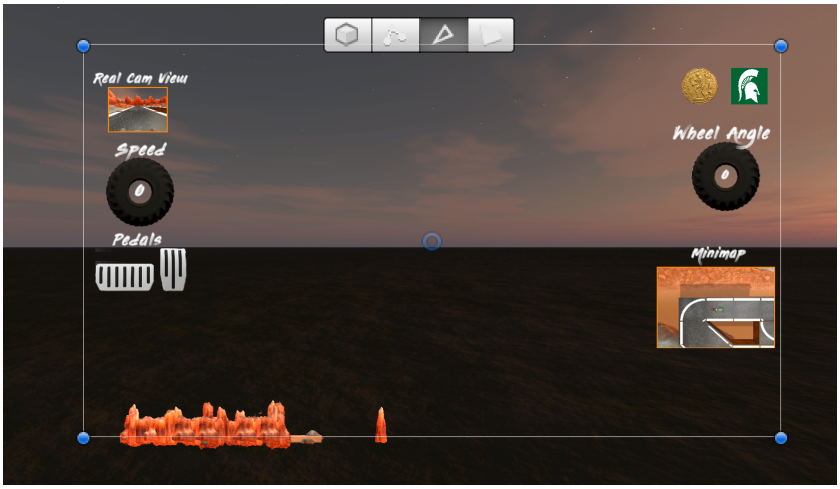


Figure 4.14: 2D Canvas Elements include speed and steering displays, a minimap, the forward camera view, and a minimap.

steering and velocity values are read from the file at every loop).

This approach allows near-realtime control from an external model. Python scripts monitor the StreamingAssets folder for a new image, process this image to determine a steering angle and velocity, and update the file with these new values for controlling steering and velocity at 30+Hz.

Using the PyGame library[125], we also "pass through" non-zero gamepad values, allowing for semi-supervised vehicle control (AI with human overrides). Upon releasing the controller, the external AI resumes control. This allows us to test models in-game and helps us "unstick" vehicles to observe a model after encountering a complex scenario.

4.1.4.10 User Controlled Input and In-game AI Modes

There are two game modes: User Control and AI Control.

In User Control, users interface with gamepads' analog joysticks to control the vehicle.

Under AI Control, the buggy is controlled by in-game AI or an external model (Section 4.1.4.9).

In-game AI captures data without user input. This AI uses a three-camera system (the front camera [the same used to capture the synthetic view image] and two side cameras rotated ±60° relative to the y-axis) as input. Each camera calculates the distance between its position and the invisible track-border colliders.

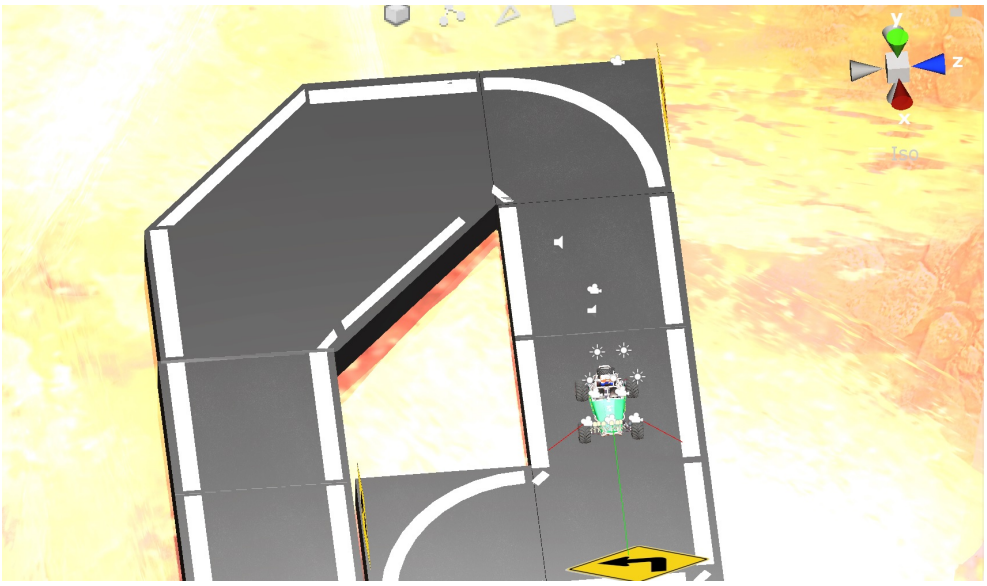


Figure 4.15: This figure shows the cameras measuring the distance to the environment via raycasting. The green line projected from the front camera indicates the longest unoccupied distance to an object, while the two red lines indicate the distance measures for each side camera. The in-game AI keeps these distances roughly equal to center the buggy in the lane.

The camera orientations and "invisible" distance measurements are represented in Figure 4.15 (green represents the longest clear distance, and red lines indicate more-obstructed pathways). The car moves in the direction of the longest free distance. If the longest distance comes from the front camera, the buggy moves straight. If it comes from a side cameras, then it centers itself in the available space. When the front distance falls under a threshold, the buggy brakes in advance of a turn.

As Unity cannot emulate joystick input, we use the bridge from Section 4.1.4.9 to both write and read output for controlling the buggy. The in-game AI model knows ground truth, such as the external positioning of the track's invisible colliders, to effectively measure relative positioning. This allows the virtual buggy to drive predictably and create valuable samples without human involvement. This method generates trustable unsupervised training data for the deep learning network described in Section 4.1.6.

4.1.5 Integrating GDS with An End-To-End Training Platform

The GDS is part of an end-to-end training system for self driving also comprising a physical platform (used for model validation) and a physical training environment (duplicated in the simulated world). Each element is detailed in the following subsections.

4.1.5.1 A Physical Self-Driving Test Platform

Self-driving models used to automate vehicle control are ultimately designed for use in physical vehicles. Full-size vehicles, however, are costly - with a self-driving test platform potentially costing in excess of $250,000, and requiring paid skilled operators for data capture. We therefore created a lower-cost, easy-to-operate and smaller-scale physical vehicle platform and environment mirroring the GDS world to validate model performance.

We considered the TurtleBot[126] and DonkeyCar[127] platforms; both are low-cost systems. The TurtleBot natively supports the ROS middleware and Gazebo simulation tool, however the kinematics of the differential-drive TurtleBot do not mirror conventional cars. The DonkeyCar offers realistic Ackermann steering and a more powerful powertrain with higher top speed, better mirroring passenger vehicles. While the DonkeyCar platform is compelling, it suffers from some limitations: it is primarily a modularized compute module featuring one RGB camera, and this module may be installed onto a range of hardware that may introduce undesirable variability. Integrating the robotics, software, and mobility platform in a tightly-coupled package eliminates variability introduced by changing mobility platforms. The DonkeyCar also does not natively run ROS, limiting research extensibility.

We therefore developed a self-driving platform using hardware similar to the DonkeyCar, and a software framework similar to that of the TurtleBot – a $1/10^{th}$ scale radio-controlled car chassis with Ackermann steering, running ROS. Our hardware platform is a HobbyKing Short Course Truck, which limits variability in experimental design and provides a platform suitable for carrying a larger and more complex payload, as well as capable of operating robustly in off-road environments. The large base, for example, has room for GPS receivers, has the ability to carry a larger battery (for

longer runtimes or powering compute / sensing equipment), and can easily support higher speeds such that future enhancements such as LiDAR can be tested in representative environments. The total cost of the platform is approximately $400 including batteries, or $500 for a version with large storage, a long-range gamepad controller, and a Tensor Accelerator onboard.

Unlike other large-scale platforms like F1/10th[128], AutoRally[129] or the QCar[82], which cost more than our proposed platform, we offer low-cost extensibility, off-road capability, and ruggedness.

Computing is provided by a Raspberry Pi 3B+ (we have also tested with a Raspberry Pi 4 and Google Coral tensor accelerator), while a Navio2[130] provides an Inertial Measurement Unit (IMU) and I/O for RC radios, servos and motor controllers. Input is provided by a Raspberry Pi camera with 130 degree field of view and IR filter (to improve daytime performance), and optionally a 360-degree planar YDLIDAR X4[131] to measure radial distances. The platform receives human input from a Logitech F710 dual analog USB joystick.

The HobbyKing platform is four-wheel drive and has a brushless motor capable of over 20kph. The Pi is vibrationally-isolated on an acrylic plate, reducing mechanical noise and providing crash protection. The camera is mounted atop the same acrylic plate and protected by an aluminum enclosure to minimize damage during collisions. The camera is mounted to a 3D printed bracket, the angle of which was set to provide an appropriate field of view for line detection. A LiDAR, if used, is mounted to this same plate using standoffs to raise the height above the camera enclosure. The vehicle platform is shown in Figure 4.16.

The Raspberry Pi runs Raspian Stretch with realtime kernel provided by Emlid. At boot, the OS launches the $Ardupilot$[132] service, $mavros$[133], and the joy node. If LiDAR is used, the $rplidar$ node is loaded. The user launches one of two Python nodes via SSH: a $teleop$ note, which uses the joystick to control the car and logs telemetry to an onboard SD card at $10Hz$, or a $model$ node, which uses one or more camera images and a pretrained model to command the steering servo to follow line markers using a pretrained neural network. In this mode, the user manually controls the throttle using the F710. Motor and servo commands take the form of a pulse width

Figure 4.16: This 1/10th scale buggy features a Raspberry Pi 3B+, Navio2 interface board, Logitech F710 USB dongle, and a Raspberry Pi camera. Not pictured: LiDAR.

command ranging from 1000-2000uS, published to the $/mavros/rc/override$ topic.

When $teleop$ mode is started, IMU and controller data are captured by ROS subscribers and written to .CSV file, along with the $160x120$ RGB .JPG image captured from $picamera$ at the same time step. An overview of ROS architecture, including nodes and topics, appears in Figure 4.17.

4.1.5.2 Modular Training Environment

We designed a test track using reconfigurable "monomers" to create repeatable environments. We created track elements using low-cost $\frac{1}{2}$" EVA foam tiles. Components included tight turns, squared and rounded turns, sweeping turns, straightaways, and lane changes. Tiles are shown in Figure 4.18.

Straightaways and rounded (fixed-width) corners provide the simplest features for classification; wide, sweeping and right-angle corners provide challenging markings.

The reconfigurable track is quick to setup and modular compared with placing tape on the ground. It also helps to standardize visual indicators, similar to how lane dividers have fixed dimensions depending on local laws. Inexpensive track components can be stored easily, making this approach suitable to budget-constrained organizations. Sample track layouts appear in Figure 4.19.

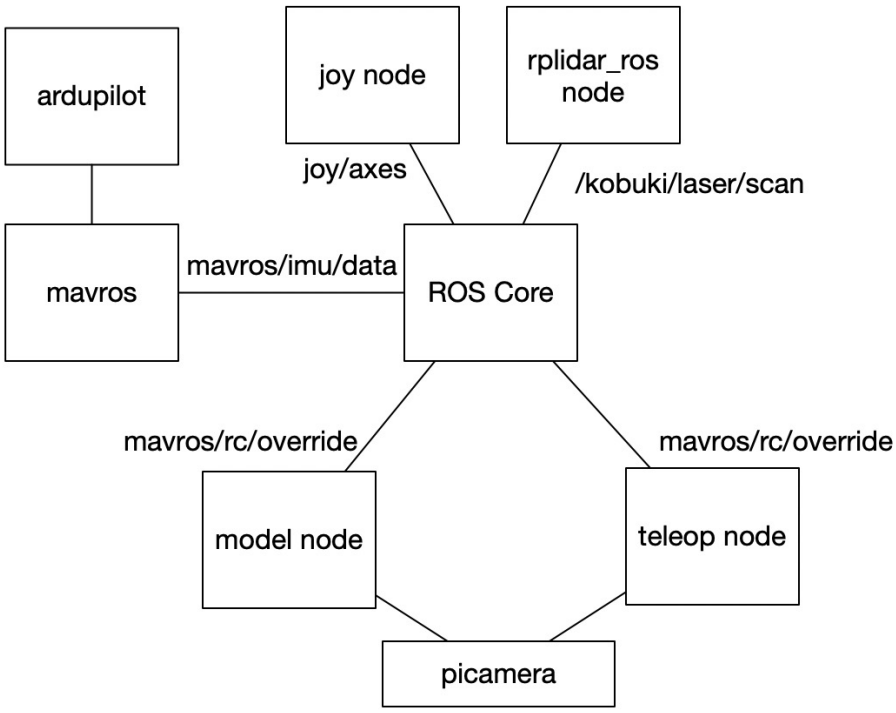


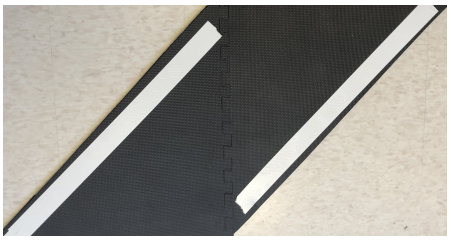
Figure 4.17: A series of ROS nodes communicate with the ROS Core in order to exchange telemetry, sensor data, and control information.

4.1.5.3 Integration with Gamified Digital Simulator

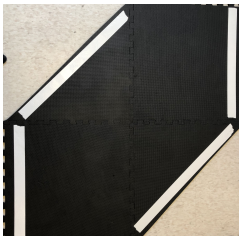
Simulation affords researchers low-cost, high-speed data collection across environments. We use the GDS to parallelize and crowdsource data collection without the space, cost, or setup requirements associated with capturing data from conventional vehicles. We aim for a physical vehicle to "learn" to drive visually from GDS inputs.

In Section 4.1.4, we describe the creation of an in-game proxy for the physical platform. The virtual car mirrors the real vehicle's physics, and driven using the same F710 joystick. The simulator roughly matches the friction coefficient, speed, and steering sensitivity between the vehicles, with numeric calibration where data were readily quantifiable (e.g. field of view for the camera).

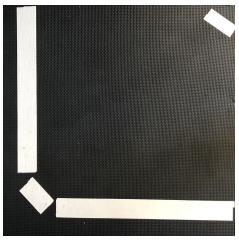
The virtual car saves images to disk in the same format as the physical vehicle to maximize data interoperability, and exports throttle position, brake position, and steering angle to a CSV.



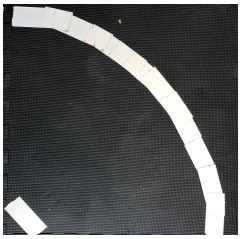
(a) This monomer, made up of two half-tiles, is a high-speed and narrow "lane change."



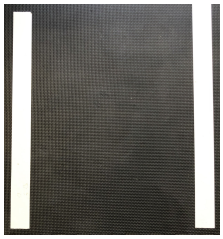
(b) This monomer, made up of two half-tiles and two full tiles, is a high-speed and very wide "lane change" piece. It is difficult to navigate as only one lane marker is visible at a time (or none, if the car is perfectly centered).



(c) This monomer, made up of one tile, is a very sharp 90-degree turn. When approached head-on, there are only two small angled pieces of tape to differentiate left from right turns (the vertical and horizontal lines dominate the field of view).



(d) This monomer is a sweeping turn made up of one tile, and is fairly easy for the vehicle to follow. However, the turn radius is almost exactly the maximum allowed by the car's steering geometry.



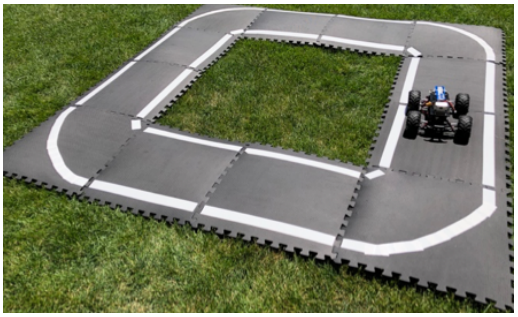
(e) This monomer is a single-tile straightaway, slightly wider than the width of the buggy.

Figure 4.18: These monomers are made of 24" square EVA foam interlocking gym tiles and combine to form a range of track configurations with varying complexity.

4.1.6 Data Collection, Algorithm Implementation and Preliminary Results

For our purposes, the development of a line following algorithm functioning in both simulation and on the physical platform serves as a representative "minimum viable test case," with successful performance in both the simulated and real environment reflective of the suitability of crowd-generated data for training certain self-driving algorithms, as well as of the relevance and tight coupling of both the hardware and software elements designed.

To prove the viability of the simulator as a training tool, we designed an experiment to collect line-following data in the simulated environment for training a behavior-cloning, lane-following



(a) This outdoor track layout is similar to the original rectangular surface used in the coin-collecting game.

(b) This track mimics the track geometry used in the simulator with invisible colliders.

Figure 4.19: We tested multiple physical tracks both matching the layouts in-game and of entirely new designs.

Convolutional Neural Network (CNN). This approach is not designed for State of the Art performance; rather, it is to demonstrate the tool's applicability to generating suitable training data and resulting trained models for operation on the test platform without explicit domain adaptation.

We generated training data both from "wisdom of the crowd" (multiple human drivers) and from "optimal AI" (steering and velocity based on logical rules and perfect situational information). We then attempted to repurpose the resulting model, without retraining, to the physical domain.

We first collected data in the simulated environment by manually driving laps in 10-minute batches. As described in Section 4.1.5.3, the simulated camera images, steering angles, and throttle positions were written to file at $30Hz$. In addition to relying on "wisdom of the crowd" and implicit game mechanics enforcing effective data generation, samples with or near zero velocity, negative throttle (braking/reverse), or steering outside the control limits (indicating a collision) were ignored to prevent the algorithm from learning "near crash" situations. Such data can later be used for specialized training. Rather than using image augmentation, we instead used the in-game "optimal AI" (AI with complete, noiseless sensor measurements and well characterized rules as described in Section 4.1.4.10) to generate additional ground-truth data from the simulator and ultimately used only symmetric augmentation (image and steering angle mirroring to ensure left/right turn balance).

When training CNNs on an fixed-image budget of 50K or 100K images, neither the human-only

training approach, nor the AI-only approach individually resulted in models supporting in-game or real-world AI capable of completing laps. With larger numbers of images from each individual training source, an effective neural network might be trained. Instead, blending the training images 50%/50% from human and AI sources with the same image budget (50 and 100K) yielded models capable of completing real-world laps (occasionally and repeatably, respectively). Our results indicate that the joint-training approach converges more quickly than either source alone and we posit that the combination of repeatability (assisted exploration via in-game AI) and variability (human operation) yields a more effective training set.

Final training data were approximately half human-driven and half controlled by AI with noiseless information. We collected $224,293$ images, almost $450,000$ images after symmetry augmentation. Data capture occurred rapidly; we were able to capture all necessary data. To the point of being faster than physical testing on a full-scale vehicle, it took less time to capture, train, and validate the model than a typical review cycle for experiments involving human subjects –approximately one week. Each image was then post-processed to extract informative features.

We developed an image preprocessing pipeline in OpenCV[134] similar to that proposed in the Udacity "Intro to Self Driving" Nano Degree Program capable of:

1. Correcting the image for camera lens properties
2. Conducting a perspective transform to convert to a top-down view
3. Conversion from RGB to HSL color space
4. Gaussian blurring the image
5. Masking the image to particular ranges of white and yellow
6. Greyscaling the image
7. Conducting Canny edge detection
8. Masking the image to a polygonal region of interest
9. Fitting lines using a Hough transform
10. Filtering out lines with slopes outside a particular range
11. Grouping lines by slope (left or right lines)
12. Fitting a best-fit line to the left and/or right side using linear regression
13. Creating an image of the best-fit lines
14. Blending the (best-fit) line(s) with the edge image, greyscale image, or RGB image

Using a sample CNN to estimate the significance of each preprocessing step, we conducted a grid search and identified a subset as being performance-critical. Ultimately, preprocessing comprised:

1. Conversion from RGB to HSL color space
2. HSL masking to allow only white and yellow regions to pass through into a binary (black-/white) image

3. Gaussian blurring

4. Masking the image to a region including only the road surface and none of the vehicle or environs

This pipeline improved predictor robustness for varied lighting conditions without increasing per-frame processing time compared to raw RGB images. Example input and output from the real and simulated camera appear in Figure 4.20.

We additionally tested camera calibration and perspective transformation, but found these operations to add little performance relative to their computational complexity. The predicted steering angles and control loop update sufficiently quickly that over- or under-steering is easily addressed, and a maximum steering slew rate prevented the vehicle from changing steering direction abruptly, minimizing oscillation.

We trained 25+ CNN variants in Keras[122] using simulated data. For each model, we recorded outsample mean-squared error (MSE). We saved the best model and stopped training when the validation loss had not decreased more than 0.1 across the previous 200 epochs. In practice, this was approximately 400 epochs. In all cases, testing and validation loss decreased alongside each other for the entirety of training, indicating the models did not overfit.

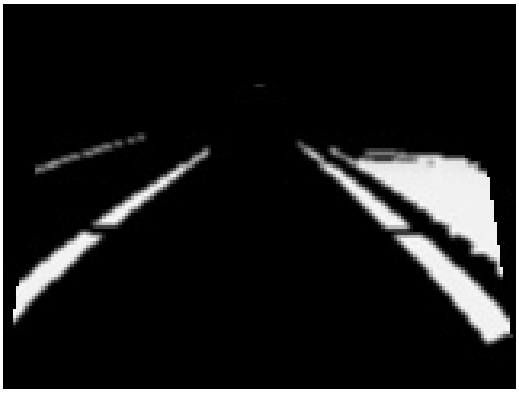
From these results, we selected two CNNs with the best outsample performance: one using a single input image, and one using a three-image sequence (current and the two preceding frames) to provide time history and context. The trained convolution layers were 2D for single-images and 3D for image sequences, which provided the sequence model with temporal context and improved velocity independence. The representative models chosen to demonstrate the simulator's ability to generate data suitable for effective sim2real domain transfer, are shown in Figure 4.21a and 4.21b.

A comparison of the predictive performance of the single-image and multi-image model for the (simulated) validation set appears in Figure 4.22. These plots compare the predicted steering angle to the ground-truth steering angle, with a $1:1$ slope indicating perfect fit.

(a) This is a sample RGB input image from the simulated environment.

(b) This image shows the preprocessed camera input (B&W) for the simulated environment. The lines stand out relative to the rest of the image, improving classifier robustness.

(c) This is a sample RGB input image from the Raspberry Pi camera on the physical vehicle.

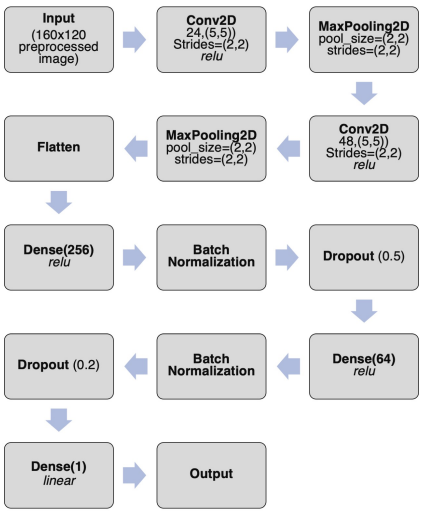


(d) This image shows the preprocessed camera input (B&W) for the real environment. The extracted features are similar between the simulated and physical world, though the real-world has more noise and the white floor is retained as a potential "line feature."

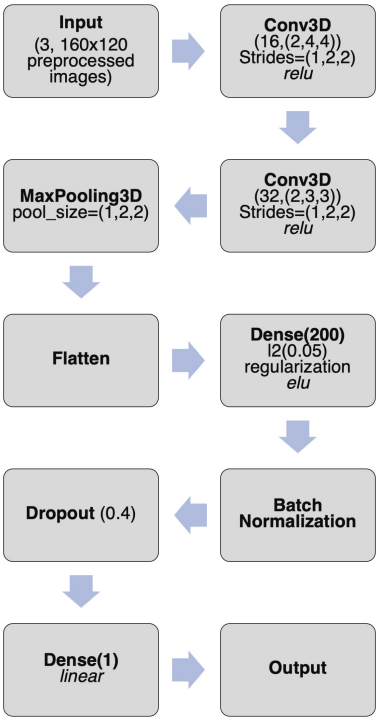
Figure 4.20: Preprocessing images is an efficient process that improves model transferrability between the simulated and physical world.

4.1.7 Model Testing and Cross-Domain Transferrability

We tested the trained model in both virtual and physical environments to establish qualitative performance metrics related to domain adaptation.



(a) This 2D Convolutional Neural Network is the single-image model, taking a greyscale 160x120 image as input and outputting a single (float) angle.



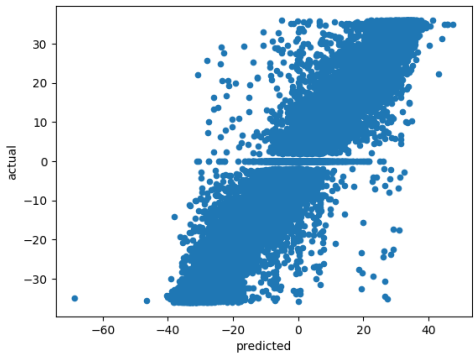
(b) This 3D Convolutional Neural Network is the single-image model, taking three greyscale 160x120 images as input and outputting a single (float) angle.

Figure 4.21: The two models used to predict steering angle from greyscale images are both convolutional neural networks (CNNs) - 2D for the single-image case, and 3D for the image-sequence case. Both models minimize MSE and use the Adam optimizer.

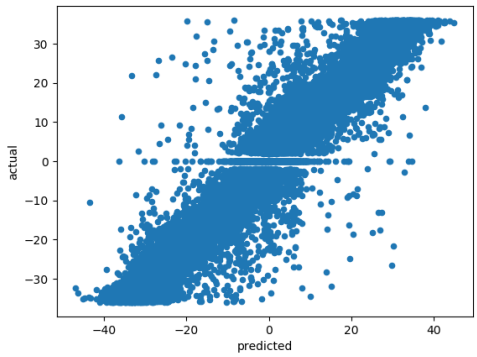
4.1.7.1 Model Validation (Simulated)

We first tested the model in the simulator, using Keras to process output images from the virtual camera. The neural network monitored the image output directory, running each new frame through a pretrained model to predict the steering angle. This steering angle and a constant throttle value were converted to simulated joystick values and written to an input file monitored by the simulator. The simulator updated the vehicle control with these inputs at $30Hz$, so the delay between the image creation, processing, and input was 0.06 seconds or better. This process is described in Section 4.1.4.9.

The simulated vehicle was able to reliably navigate along straightaways and fixed-radius turns. It also correctly identified the directionality of tight and sweeping corners with high accuracy,



(a) This figure shows the predicted versus ground-truth steering angles for the single-image model, with a reasonably strong diagonal component despite few outlying values.



(b) This figure shows the predicted versus ground-truth steering angles for the multiple-image model, with a strong diagonal clustering.

Figure 4.22: Comparing the model performance for the single- and multi-image predictor, using in-game images captured at approximately 30 frames per second.

though it struggled with hard-right-angle turns, suggesting the model identifies turns by looking for curved segments rather than by corner geometry. For some initial starting positions, the simulated vehicle could complete $>$ 20 laps without incident. For other seeds values, the vehicle would interact with the invisible colliders and "ping-pong" against the walls (directional trends were correct, but narrow lanes left little room for error). In these cases, human intervention unstuck the vehicle and the buggy would resume self-driving.

We next transferred the most-effective model to the physical vehicle without explicit transfer learning.

4.1.7.2 Model Transferrability to Physical Platform

We ported the pretrained model to the physical vehicle, changing only the HSL lightness range for OpenCV's white mask and changed the polygonal mask region to block out the physical buggy's suspension , and scaled the predicted output angle from degrees to microsecond servo pulses. There was no camera calibration and no perspective transformation.

As with the simulated vehicle, the buggy was able to follow straight lines and sweeping corners using the unaltered single- and multi-image models, with the vehicle repeatedly completing several

laps. It was not necessary to retrain the model. Both the real car and simulator control loops operate at low loop rates (8-30Hz) and speeds ($\approx$ 10kph), so the classifiers' predicted steering angles cause each vehicle to behave as though being operated with a "bang-bang" controller ("left-right") rather than a nuanced PID controller (see video running a representative image sequence model at low speeds). Though line-following appears "jerky," small disparities between the calibration and sensitivity of the virtual and physical vehicle's steering response minimally impact lane keeping as framerates (enabled with enhanced compute) increase.

We also qualitatively evaluated the single- and multi-image models relative to their performance in the simulated environment. In practice, the model relying on the single image worked most robustly within the simulated environment. This is because the images for training were captured at a constant 30Hz, but the vehicle speed varied throughout these frames. Because the 3D convolution considers multiple frames at fixed time intervals, there is significant velocity dependence. The desktop was able to both process and run the trained model at a consistent 30Hz, including preprocessing, classification, polling for override events from the joystick, and writing the output file, making the single-image model perform well and making the impact of reduced angular accuracy relative to the image-sequence model insignificant as the time-delta between control inputs was only 0.03 seconds.

In the physical world, where the buggy speed and frame capture and processing are slower, the vehicle's velocity variation is a smaller percentage of the mean velocity and computational complexity matters more. As a result, the image sequence provides better performance as it anticipates upcoming turns without the complication of high inter-frame velocity variation. The physical vehicle performed laps consistently with the image-sequence model, though it still struggled with the same right-angle turns as the simulator.

Other models (e.g. LSTMs) may offer improved performance over those demonstrated; an ablation study could validate this. However, those chosen models demonstrate effectively the desired goal of model transferrability and the function of the end-to-end simulator and physical platform toolchain.

These results show successful model transferrability from the simulated to physical domain without retraining. The gamified driving simulator and low-cost physical platform provide an effective end-to-end solution for crowdsourced data collection, algorithm training, and model validation suitable for resource-sensitive research and development environments.

4.1.8 Conclusion, Discussion, and Future Work

Our toolchain uniquely combines simulation, gamification, and extensibility with a low-cost physical test platform. This combination supports semi-supervised, crowdsourced data collection, rapid algorithm development cycles, and inexpensive model validation relative to contemporary solutions.

There are opportunities for future improvement. For example, adding a calibration pattern would allow us to evaluate the impact of camera calibration on model performance. We plan to include simulated LiDAR to improve the simulator's utility for collision avoidance, and to create a track-builder utility or procedural track generator. Incorporating multiplayer, simulated traffic, and/or unpredictable events ("moose crossing" or passing cyclists[135]) would help train more complex scenarios. Alternatively, an extension of the simulator may be used to train adversarial networks for self-driving to support automated defensive driving techniques[136], or high-performance vehicles[137].

Because the simulator is based on a multi-platform game engine, broader distribution and the creation of improved scoring mechanisms and game modes will provide incentive for players to contribute informative supervised data, supporting rapid behavior cloning for long-tail events. These same robust scoring metrics would allow us to rank the highest-performing drivers' training data more heavily than lower-scoring drivers when training the neural network. Some of these can be visible to the user (a "disqualification" notice), while other score metrics may be invisible (a hidden collider object could disable image and CSV capture and deduct from the user's score when the vehicle leaves a "safe" region). Other changes to game mechanics might support the capture of necessary "edge case" unsuitable to generate by other means - particularly with typical research

labs' economic or computational constraints. These adaptations might, for example, contribute to the capture of data from incident avoidance scenarios, or operating in inclement weather, or during particular vehicle failures (such as a tire blowout). Multiplayer driving, with AI or human operated other vehicles, would generate other sorts of edge case data to enhance data capture via naturalistic ablation. This work will require developing a network backend for data storage and retrieval from diverse devices. Changing game mechanics in some respects is simpler and more comprehensively addresses evolving data capture needs than seeking to explicitly codify desirable and undesirable data capture.

There may also be opportunities to integrate the simulator and physical vehicles into an IoT framework[138] or with AR/VR tools[139], such that physical vehicles inform the simulation in realtime and vise-versa.

Finally, crowdsourcing only works if tools are widely available, and we plan to refine and release the compiled simulator, test platform details, and resulting datasets to the public.

4.2 The Virtual LiDAR Simulator

4.2.1 Short Description

LiDAR sensors are common in automated driving due to their high accuracy. However, LiDAR processing algorithm development suffers from lack of diverse training data, partly due to sensors' high cost and rapid development cycles. Public datasets (e.g. KITTI) offer poor coverage of edge cases, whereas these samples are essential for safer self-driving. We address the unmet need for abundant, high-quality LiDAR data with the development of a synthetic LiDAR point cloud generation tool and validate this tool's performance using the KITTI-trained PIXOR object detection model. The tool uses a single camera raycasting process and filtering techniques to generate segmented and annotated class-specific datasets. The tool has the potential to support the low-cost bulk generation of accurate data for training advanced self-driving algorithms, with configurability to simulate existing and upcoming LiDAR configurations, with varied channels, range, vertical and horizontal fields of view, and angular resolution. In comparison with other virtual LiDAR solutions

like CARLA (which has been used to generate some synthetic datasets [140]), this tool requires less game development knowledge and is therefore faster to set up: LiDAR setting customization can be done in a front-end panel enabling users to focus only on the data generation part. The simulator is developed using the Unity Game Engine in conjunction with free and open-source assets, and a build will be shared with the AV community before its eventual open-source release.

4.2.2 Introduction

Automated driving requires detecting and localizing objects [141], e.g. for motion coordination and route planning [8]. In the context of autonomous driving, three dimensional object detection is an essential perceptive element. Most fully and partially automated vehicles are equipped with sensors including LiDAR (Light Detection And Ranging), RADAR (Radio Detection And Ranging), ultrasonic, and camera sensors. In many implementations, LiDAR sensors provide accurate, high-resolution and long-range depth information, while cameras capture detailed semantic information under optimal conditions.

LiDAR data is often represented as a point cloud, though raw point clouds may be complex to process and as a result, different encoding approaches have been explored. Point clouds are a powerful data representation format. By encoding spatial data as a collection of coordinates, point clouds nicely encode large datasets for downstream processing techniques. Primarily, point clouds turn the raw data collected by LiDAR processes into 3D meshes. However, they can also be used to store and manipulate other spatial information. A 3D point cloud is a collection of data points analogous to the real world in three dimensions. Each point is defined by its own position, and, in cases, color. The points can be rendered as pixels to create an accurate 3D model of the object or environment. Point clouds can describe objects and scenes measuring from just a few millimeters to entire cities, depending on the sensor type and sample acquisition method.

While point clouds store data effectively, the ideal data representation for automated vehicles instead allows standard convolution operations to be applied for 3D object classification and detection. Existing representations may be divided into two classes: 3D voxel grids, and 2D

projections. In a 3D voxel grid, each voxel cell can contain a scalar value (e.g., occupancy) or vector data (e.g., hand-crafted statistics computed from the points within that voxel cell), where high-order representations of the voxel grid may be extracted in 3D convolution [142]. Camera data, on the other hand, may be less dense - with even complex implementations comprising red, green, blue, and infrared intensity values for a 2-dimensional pixel range. Multiple camera outputs may be linked to create depth data, or dedicated depth cameras may be used.

Today's LiDAR sensors produce unstructured data in the form of a point cloud containing as many as 10^5 3D points per 360-degree sweep, while cameras may capture 4K video at 60 frames per section, with multiple cameras fusing data to stitch a comprehensive 360 degree view around the vehicle. Perceptive performance from each sensor class individually is improved through the fusion of LiDAR point cloud data with RGB video frames; the resulting volume of data presents a computational challenge for modern scene detectors. There is an unmet opportunity to generate both camera and LiDAR data for use in training data-efficient object representations, as well as classifiers. In the following section, we consider some existing methods for object detection and lead into the development of a novel, game-based synthetic LiDAR simulator.

4.2.3 Object Detection

The last decade has brought about myriad methods exploring the application of Convolutional Neural Networks for accurate 2D object detection, often from a single image [143, 144, 145, 146, 147, 148]. Object detection consists classification and localization. Convolutional Neural Networks (CNN) have shown strong performance in image classification [149], and such networks are used for object detection when running inference over cropped regions representing the candidate objects. Region-based Convolutional Neural Networks (R-CNN) [143] involve region proposals, which help to localize objects within an image. R-CNN first extracts the whole-image feature map with an ImageNet [150], and a CNN then predicts bounding box position per proposal via an Region of Interest (RoI) pooling operation on the whole-image feature map [151]. Since the introduction of class-agnostic object proposals [152, 153], proposal based approaches have become more popular

with R-CNN.

Fast R-CNN [145] is a one-stage end-to-end training process using a multi-task loss on each labeled RoI to jointly train the network. Another improvement is that Fast R-CNN uses a RoI pooling layer to extract a fixed size feature map from region proposals of different size. This operation does not require warping regions and reserves the spatial information of features of region proposals, which leads to further gains in both performance and speed. Faster R-CNN approaches do exist [144, 146], and proposal based object detectors achieve strong performance in many public benchmarks [154, 155].

These algorithms represent advances in object classification from LiDAR and other data, but also highlight a problem: training, testing, and validating algorithms requires abundant data from diverse scenarios to ensure performance.

4.2.4 LiDAR

LiDAR is an active remote sensing system, meaning the system itself emits energy - in this case, light - to measure ambient environmental objects. In a LiDAR system, light is emitted from a rapidly firing laser [156]. Imagine a collimated pulse of photons quickly strobing from a laser light source: light travels to the environment and reflects off of things like buildings, roads, cars and tree branches. The reflected light energy then returns to the LiDAR sensor where the distance travelled is recorded. Measuring distance can take multiple forms; some LiDAR system measure the time it takes for emitted light to travel to the ground and back and convert this into a distance based on the speed of light, while other sensors make use of other measurements, e.g. phase shift of a reflected signal. The measurement method determines the maximum measurement range, speed, accuracy, and precision.

Some LiDAR systems send out pulses of light just outside the visible spectrum and register how long it takes each pulse to return, either rotating or otherwise steering the light to sample the distance to points at various angles relative to the sensor. The direction and distance of whatever the pulse hits are recorded as a point of data [157]. LiDAR units may implement different methods

for multi-dimensional data capture, but generally, they sweep in a circle like a RADAR dish, while simultaneously moving the laser up and down. Once the individual readings are processed and organised within a unified coordinate system, the LiDAR data is transformed into point cloud data.

These measurements may be used in conjunction with sampling data from a GPS or an Internal Measurement Unit (IMU) to provide additional positional accuracy and information regarding the absolute global orientation of the ground plane. The result is that each measurement, which combines the sensor's location with the angle of measure and distance to an object, returns one or more (X, Y, Z) coordinates describing the encountered environment. Relative distances from the sensor to the ground, buildings, forest canopy, motorway overpasses, and anything else that the laser beam encounters during the survey constitutes point cloud data. The specific surface features that the laser encounters can be classified further after the initial LiDAR point cloud is processed. Some LiDAR may fuse data with cameras or other sensors to assign to the point cloud additional values such as reflectivity or color.

As with any emerging Deep Technology [2], LiDAR has its challenges and it is worth noting these to understand both its potential and limitations [157]. The main challenges can be summarized as:

- Line of sight access is needed
- Reflective surfaces can cause problems for the laser
- Bad weather can interfere with data collection

Research underway aims to resolve these and other challenges, particularly within the applied use case of vehicle automation.

4.2.5 Virtual LiDAR Simulator

Perhaps due to the high costs of LiDAR sensors, resources for sample point cloud data are limited. There are some public datasets, like KITTI [112], created from real-world LiDAR sensors - but these public datasets tend not to capture certain critical edge case scenarios, e.g. near collisions, roadway

debris, inclement weather, or off-road operation. In order to address this lack of comprehensive data, we suggest the use of synthetic point cloud data generated in virtual environments [7]. In comparison with real LiDAR point cloud data, the synthetic data can be labeled, annotated, filtered, and segmented, while optionally targeting data capture only from specific virtual objects. Despite appearances of simplified data capture, even this methodology has drawbacks stemming from the lack of easy-to-use, adaptable, and freely available LiDAR simulators. One tool that is broadly used for Autonomous Vehicles (AV) research is the CARLA simulator [158], but this tool requires advanced technical skills and complex setups to function as a highly-adaptable LiDAR point cloud generator. We developed a solution that offers a dedicated means of synthetic point cloud generation.

Figure 4.23: A Main Menu enables scene transitions

Our solution has advantages in comparison with other applications like Carla or AirSim, including:

- It is a dedicated solution for LiDAR sampling, which allows for the simulation of a range of industry-standard and yet-to-be-developed LiDAR

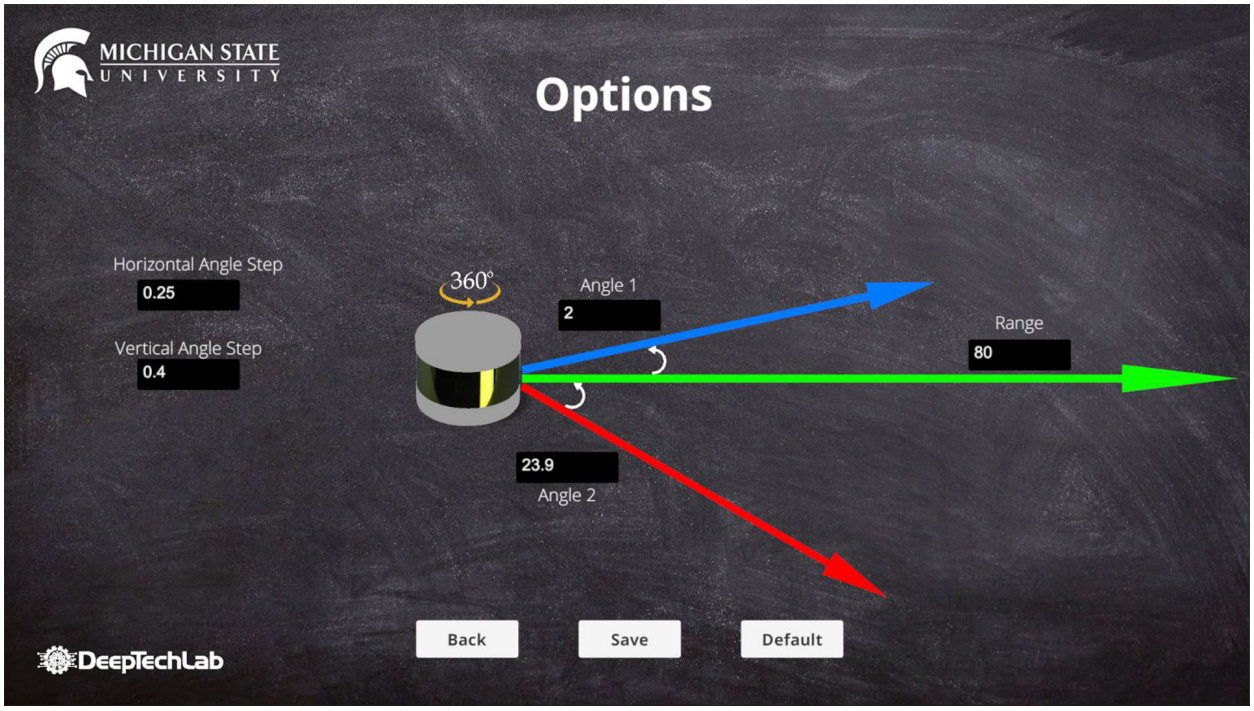


Figure 4.24: The Options Panel provides the opportunity to simulate any existing or yet-to-be-developed LiDAR by graphically altering key values.

- It does not require specific knowledge on software or game development to capture samples - just run the executable and press a key
- The configurability of options is a front-end task
- Samples can be captured from only particular object classes (see subsection "Simulating the LiDAR")
- The Windridge City [159][1] environment offers an High Dynamic Range (HDR) which provides more realistic depictions of color and brightness suitable for training self-drive algorithms

The LiDAR simulator is built using the Unity 3D Game Engine [160, 161]. All assets used to develop this application are free and open-source, while custom scripts emulate the behavior of a LiDAR in a Virtual Environment (VE).

[1] Also available: `https://naturemanufacture.com/windridge-city/`

Figure 4.25: The application's Editor Mode. At center, we can identify the vehicle controller and the single LiDAR camera that performs the sampling in the virtual environment.

4.2.5.1 Simulator Scenes and Mechanics

The LiDAR simulator consists of two main scenes, the Main Menu (Fig. 6.6) and the Main Scene (Fig. 4.26 - 4.27). The Main Menu is used for scene transitions and also links to the Options (Fig. 4.24) (see section Conclusions and Future Work) and the Credits panel.

The Options Panel is where users can edit sensor parameters to simulate a diverse range of LiDAR sensors. Specifically, a user can change both the horizontal and vertical angle steps, two values of vertical angles (in relation to 0 angle) and the range which represents the maximum distance of sampling.

The Main Scene is the core element of the user-facing Windows application. For the main virtual environment, we used the Windridge City asset[159], a free and open source urban area with realistic graphics and diverse traffic objects (lights, signs etc). We additionally used a free 3D model of a car and Unity's default vehicle controller as a placeholder mount for the virtual LiDAR (child gameObject) which allows users to move across the virtual city as if they were driving a real

Figure 4.26: A vehicle controller view while a user moves around the Windridge City environment.

vehicle.

4.2.5.2 Simulating the LiDAR

In order to simulate the LiDAR in the virtual world, we used the datasheet of the Velodyne HDL-64E [162] 64-channel LiDAR and populated these values within the Options panel. With our custom scripts, we simulate the LiDAR's 64 channels using only one game virtual camera to improve capture efficiency. The sensor implementation in game creates a rotating camera (both horizontal and vertical rotation) that spawns and "shoots" small 3D spheres to all directions. If a sphere "hits" a virtual object which is pre-tagged as the target class (in our case, "car") the script automatically saves four values into a single CSV file in the StreamingAssets folder:

- The X, Y, and Z coordinates of the center of the sphere (the sphere's center "sticks" to the virtual object's surface) - Columns 1-3
- A random value ranging from 0.3 to 0.6 representing the reflectance for a car as determined by

Figure 4.27: This screenshot showcases another vehicle controller view where other virtual cars (virtual objects with tag "car") are identified and they are potential sample targets for our LiDAR simulator.

> the KITTI dataset. These values can be generated on a per-class basis (which acts as a means of noise generation for signal augmentation), or else may be pulled from the in-game objects' materials. - Column 4

All 3D models of the virtual environment were tagged with an appropriate object class, which may allow for future use in developing algorithms for classifying and detecting other objects.

The tags we created in Unity are the following:

- **Tree** - trees and other types of vegetation
- **Asphalt** - roads except dirt
- **Sign** - stop or speed signs
- **Lamp** - lamps at the roadside
- **DirtRoad** - dirt roads outside the main city

- **TrafficLight** - traffic lights in the city area
- **Building** - buildings, regardless the type or size

4.2.6 PIXOR

Validating simulator performance required testing synthetic point clouds on algorithms designed for real-world LiDAR data classification to ensure transferability.

4.2.6.1 Introduction to PIXOR

PIXOR [163] is an efficient 3D object detector that generates accurate bounding boxes based on LIDAR point clouds. Bounding box estimates include heading angle and location within 3D space, and predicting this accurately is essential for automated and assisted self driving. The point cloud data from many LiDAR sensors are 3D and unstructured, so standard convolutional neural networks' discrete operations do not apply directly, while 3D convolutions can be computationally expensive [164].

PIXOR acts upon a compact 2D representation of LIDAR point clouds, and is therefore more amenable to real-time inference than a 3D voxel grid representation of the same object. In PIXOR, a bird's eye scene view (BEV) is used, which reduces data dimensionality with minimal information loss. By using BEV there are advantages like being able to apply 2D convolution, which mitigates the overlapping objects in object detection when compared to front-view representation. An overview of the PIXOR 3D object detector is shown in Figure 4.28.

4.2.6.2 Customization

Two-stage detectors, such as Faster R-CNN (Region-based Convolutional Neural Networks) [144] or Mask R-CNN [165] use a Region Proposal Network to generate regions of interests in the first stage and send these region proposals down the pipeline for object classification and bounding-box

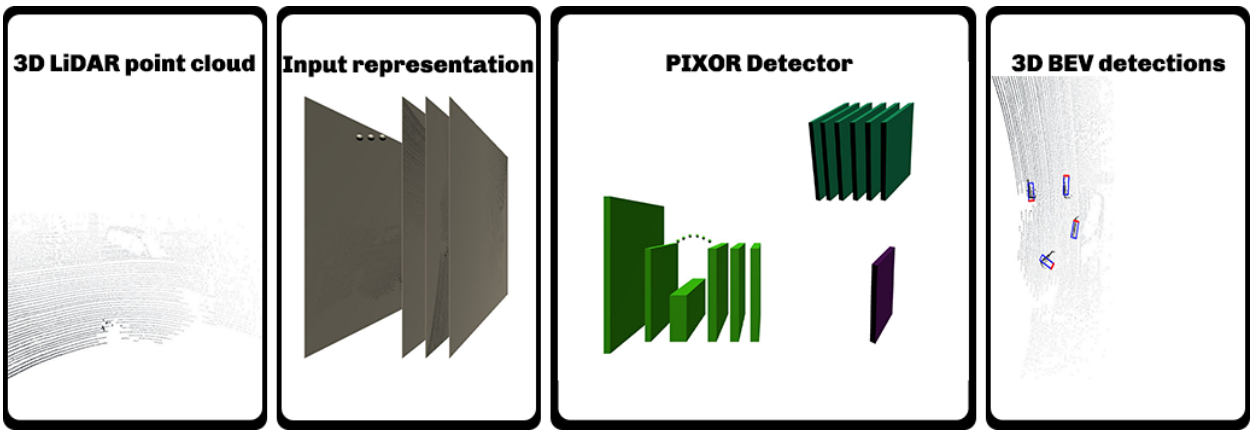


Figure 4.28: Overview of the PIXOR 3D object detector from Bird's Eye View (BEV) of LiDAR point cloud

regression. These models have high accuracy, but poor runtime. Single stage detectors such as YOLO (You Only Look Once) [147] and SSD (Singe Shot MultiBox Detector) [148] are more effective in realtime applications. YOLO and SSD treat object detection as a simple regression problem by taking an input image and learning the class probabilities and bounding box coordinates. Such models offer lower accuracy, but are much faster than two-stage object detectors. For single-class object detection, DenseBox [166] and EAST [167] show that single-stage detectors work well without using manually designed anchors. They both adopt the fully-convolutional network architecture [168] to make dense predictions, where each pixel location corresponds to one object candidate. Recently, RetinaNet [169] showed that a single-stage detector can outperform two-stage detectors if training class imbalance is resolved properly.

PIXOR's region of interest for a processed point cloud is [0, 70] Œ [−40, 40] meters, with a bird's eye view projection with a discretization resolution of 0.1 meters. The height range is set to [−2.5, 1] meters in LIDAR coordinates and all points are divided into 35 slices with bin size of 0.1 meter. The input representation has the dimension of 800 Œ 700 Œ 38. PIXOR uses data augmentation of rotation between [−5, 5] degrees along the Z axis and a random flip along the X axis during training. PIXOR additionally calculates one reflectance channel.

4.2.7 Integration of Synthetic Data with PIXOR - Experimental Setup and Validation Results

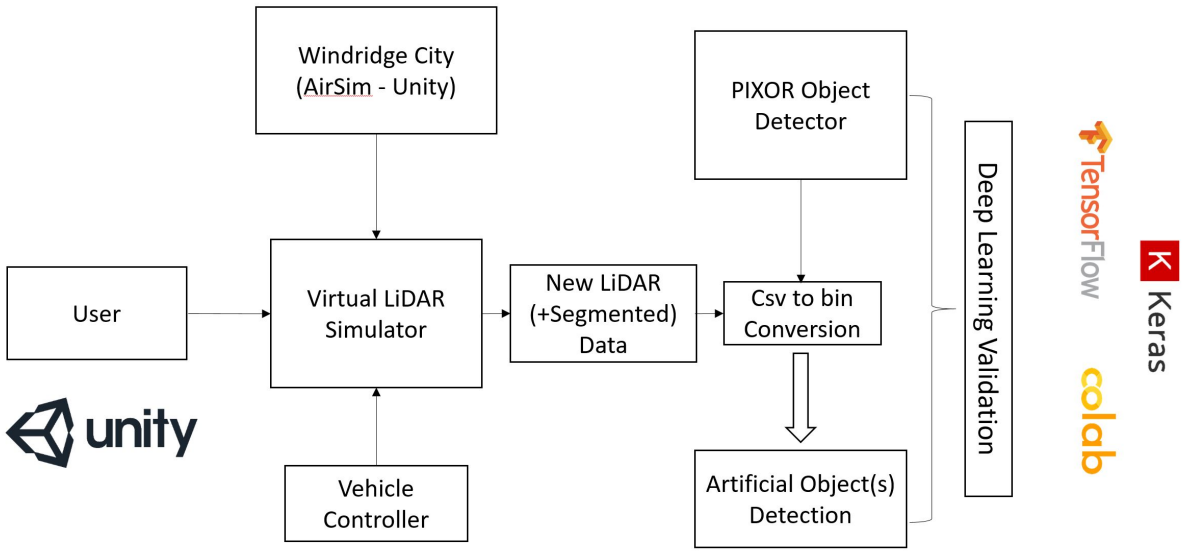


Figure 4.29: The schematic of our end-to-end system showcasing how the dataset is generated in the simulator and validated with the customized PIXOR detector

The Fig. 4.29 shows how this implementation works end-to-end. The simulator provides the default Unity vehicle controller (Fig. 4.26) with which users can move around a realistic environment (Windridge City) using the W, A, S, and D keys. The vehicle controller has the LiDAR simulation camera placed as its child gameobject (Fig. 4.25). When the users identify the exact sampling location and press the "P" key, the vehicle controller is immobilized and the sampling process begins. If during this raycasting process gameobjects with the "car" tag are identified (Fig. 4.27), measurement samples are stored saved to the application's "StreamingAssets" subfolder. The above methodology is handled by a custom script. After the sampling phase is complete, Unity outputs a CSV file.

To validate our sample, we first convert the CSV file into a bin file for validation in PIXOR. Once the conversion is done using the csv2bin tool[2] and fed in as an input to the object detection, the pre-trained model evaluates it and provides the bounding boxes on cars from the Bird's Eye View

[2]Also available: `https://github.com/kkarur/KKGP.ECE884Project.io`

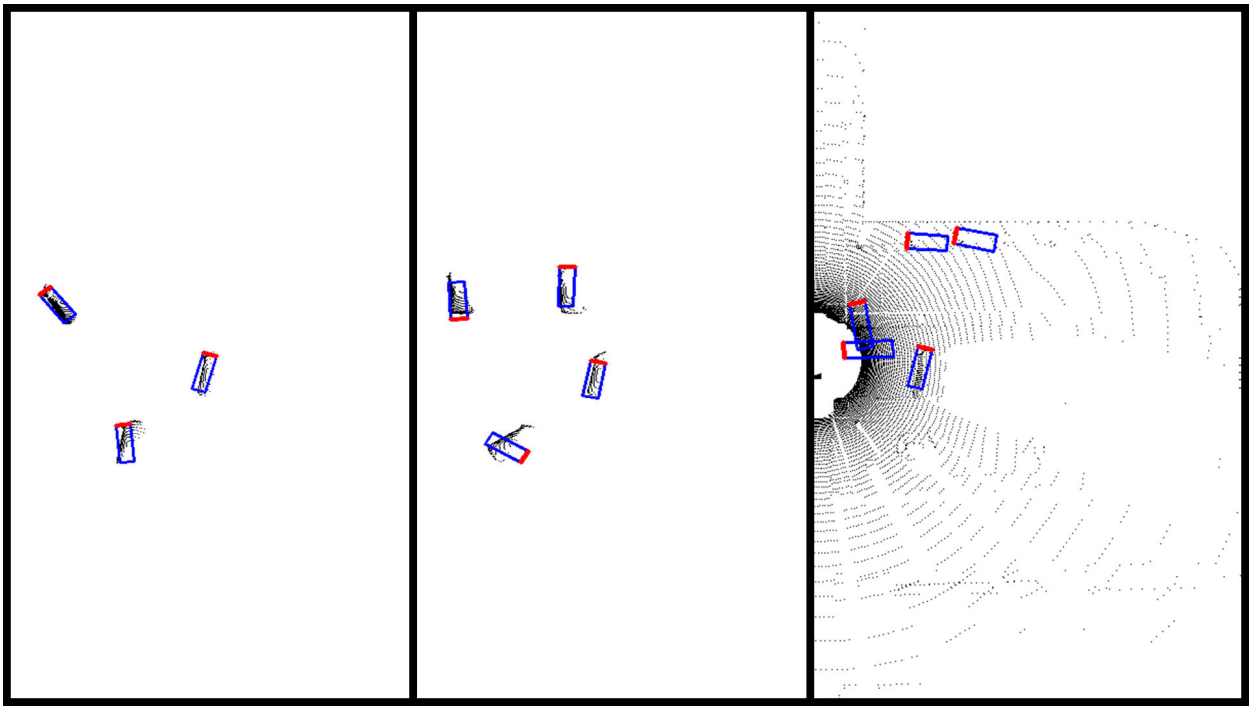


Figure 4.30: In this figure we can see three examples. Starting from left, the first two examples showcase the validation of our synthetic and filtered dataset for three and four cars respectively. The third example showcases the PIXOR output validation for an unfiltered and real dataset from KITTI.

(BEV). The output bounding boxes can be seen in (Fig.4.30) where the cars have been detected by PIXOR object detection. In the first two cases, we can view the validation results from PIXOR model for the synthetic and filtered data of our simulator and in the third one, the results from a real, unfiltered sample from the KITTI dataset.

The time taken to set up the simulator and capture a sample is a matter of minutes, even for untrained users. Validation using the PIXOR pre-trained model was an important step to showcase that segmented and filtered synthetic data generated by our simulator is plausible and reflective of that captured by real-world sensors: we managed to have virtual cars tracked by a pre-trained model trained using real data. This highlights the potential of the simulator for improving current classification and detection models with a variety of scenarios that would otherwise be difficult, unsafe, or costly to capture.

4.2.8 Conclusion and Future work

The present research showcased an end-to-end system that generates synthetic LiDAR point cloud data and validates them via a model pre-trained on real-world data. We have proved that an accurate LiDAR simulator can provide suitable data for Deep Learning training and thus, help to generate data representative of edge-case scenarios in a virtual and safe environment, whereas real-world public data sets must be safety conscious and are volume-limited due to capture cost and complexity. Our work is built upon free and open source assets and libraries, making it easy to redistribute to, adapt for, or recreate within the automotive community. The first released version of the simulator features a simple-to-use tool with a clean design and multiple other advantages which can enable even non-technical individuals to create their samples and datasets.

In the future, we plan to extend the simulator to add noise options to make it even more realistic, and also to train models fully on synthetic data and validate the results using real data. Further, we plan to train hybrid models using both real and synthetic data and compare results with models trained only in simulation or reality to evaluate the significance of the reality gap in LiDAR models. Lastly, we plan to expand the data to other types of vehicles and even traffic signs.

4.3 General Conclusions

The GDS and VLS have proved the connection that advanced gamification may have with AI. Both of these tools that are made using low-cost, free or open source assets, offer valuable synthetic data generation. We have achieved this using novel methodologies like the in-game AI data generation (AI-to-AI training), the user driving behavior capturing and the novel raycasting process for class-filtered point cloud generation. All in all, this chapter reveals the true potential of applying engineering behind the gamification in creating advanced simulators which can further help enhancing AIs. In the future, such simulator solutions will be the main sources for low-cost AI training and the the principle way of improvement of AI models that are originally trained with real data only. The simulators will transform those models into hybrid ones via the simultaneous use of real and synthetic data. The latter will be essentially used to cover edge-case scenarios, including

occasions unsafe for human life or costly cases otherwise difficult to be sampled in real-life.

4.4 Acknowledgements

The GDS research study is a collaboration project between National Technical University of Athens (NTUA) and Michigan State University (MSU) and is published at MDPI “Electronics” journal with the title “Gamified Transfer Learning for Automated Vehicle Line Following.”[7] The project consists of two major parts: the Gamified Simulator and the AI part. The first part along with the in-game AI was developed by NTUA (prior to my enrollment at MSU) and I would like to thank Dr. Politopoulos for his guidance and the external AI development along with the testing phase in a real self-driving RC vehicle was Dr. Siegel’s work at MSU. Special thanks to Yongbin Sun from the Massachusetts Institute of Technology for his contribution on deep networks and synthetic data.

The VLS research study was developed at Michigan State University and is accepted as an abstract at the 2021 SAE World Congress under the title: "Synthetic LiDAR Point Cloud Data Generation Tool and Deep Learning Validation" [8]. I would also like to thank all the team members that participated in this (Karthik Karur, who worked on the Deep Learning Validation part, Dr. Siegel and Dr. Zhang).

CHAPTER 5

COMBINING GAMIFICATION WITH OTHER LAYERS OF VIRTUAL ENVIRONMENTS

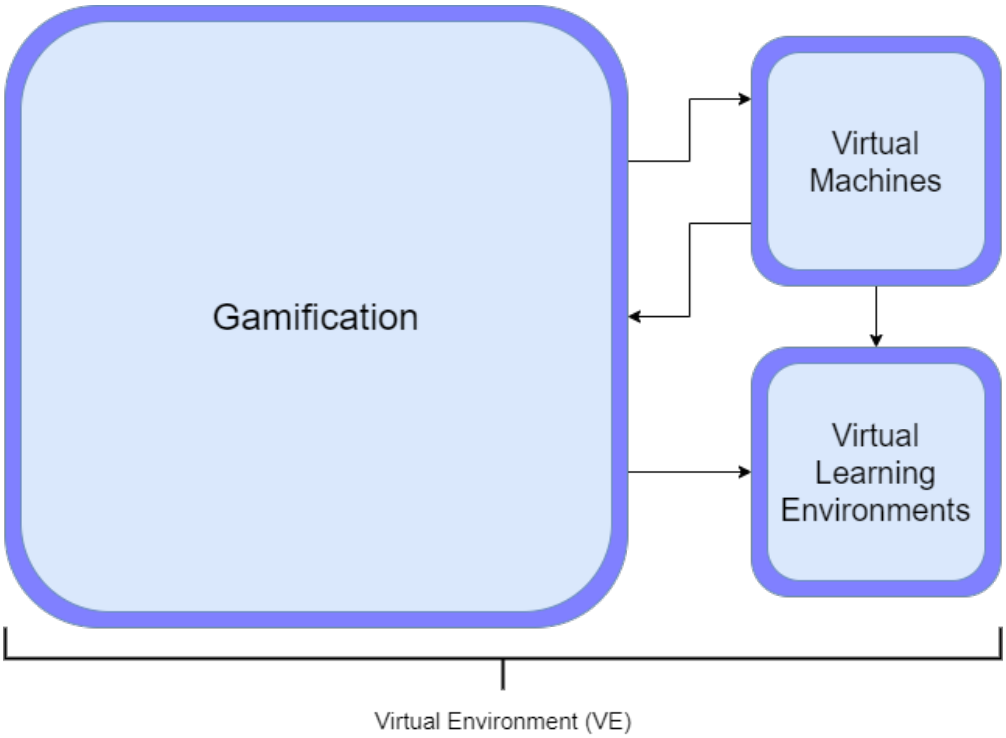


Figure 5.1: Combining Gamification with other layers of virtual environments like Virtual Machines (VMs) & Virtual Learning Environments (VLEs)

This section dives deeper into Virtual Environments (VEs). Although it does not provide any connection with IoT or AI, is important because we dive deeper into gamification and virtual environments which are the most essential parts of this dissertation. Novel connections between different virtual layers that are combined to enhance the gamification itself, will be showcased. The context is distance learning and the tool featured is the Cyber Escape Room. This tool offers integration with Cyber Ranges which are built upon virtual machines and Moodle which is a virtual learning environment. (Figure 5.1).

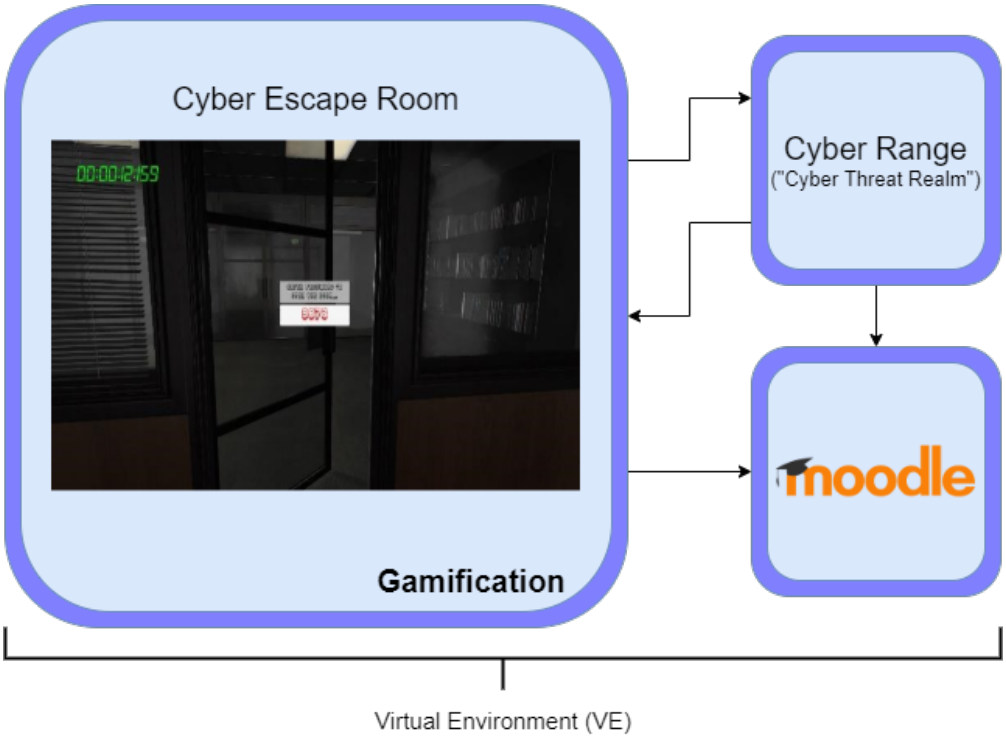


Figure 5.2: Combining Gamification with other layers of virtual environments (Cyber Range - Moodle) diagram.

5.1 Short Description

Serious games and gamification can be robust and rigorous learning tools. They serve as motivation to increase participation during in-class activities and encourages diverse students to cooperate within a safe and goal-oriented learning environment. Instructors may engage with their students and provide feedback using in-game scoring or other mechanisms, and thoughtful level design can create scaffolded learning opportunities for students to build upon foundational knowledge.

Drawing upon the concept of serious games, we developed a cybersecurity-centric, virtual "escape room," in which students must complete tasks of increasing difficulty in order to exit. The game is built on top of a cyber range called "Cyber Threat Realm", a virtualized information and communication technologies (ICT) environment, consisting of virtualized networks, computer systems and applications. In the Cyber Threat Realm, students may explore and exploit systems and their weaknesses without fear of consequence or retribution, allowing for the creation of individual

or team-based "red team- blue team" activities for students to complete. The game is designed so as to provide an easy-to-use environment for participants from a range of technical and non-technical backgrounds, increasing its reach. Envisioned exercises target general cybersecurity awareness, penetration testing, vulnerability exploitation, social engineering, and more.

Cyber Escape Room serves as a cutting-edge platform for teaching and engaging students that want to dive into the cyber security realm.

5.2 Introduction

Serious games and gamification use game design and mechanics to augment non-gaming elements of daily life by emphasizing and playing to user enjoyment and engagement. For example, serious games and gamification can increase participation in activities [13] and motivate competition among groups. Today, serious games and gamification find new homes across domains including in education [9], where they are used to create unique and memorable learning opportunities for students [16], [62], or even in collecting data to improve training data for expert systems[7]. These games are applied within a learning context, where the story is part of a thoughtful process designed to guide students towards specific learning goals. Serious games have also been investigated as a tool for cybersecurity training in cyber ranges [170]. Other related works include platforms used for Cyber Exercises, Capture the Flag events, and training and competence assessment [171], [172]. Another related work identified test beds that could be used in conjunction with educational cyber range exercises [173].

Educational games motivate through their narrative elements and mechanics, such as scoring mechanisms but also by presenting students with unique opportunities that might otherwise not be able to experience. Increased engagement inspires new and continued learning, while game mechanics can encourage diverse students to cooperate towards a common goal, while instructors can silently observe or directly engage with their students in a unique environment.

By allowing students to step outside their typical lives, and into unfamiliar and often inaccessible worlds and scenarios, games provide unparalleled learning opportunities, where students get to

role-play new jobs, see new sights [61], and act without fear of consequence. The ability to replay situations and to study closely cause-and-effect allows self-motivated students to take their education to new heights, while carefully-structured game design can lead participants to expand their horizons, one step at a time.

These capabilities are uniquely-well suited to teaching "risky" concepts, like cybersecurity [14]. Cybersecurity is often difficult to teach, because it requires access to unique tools, complex environments, and a "safe space" where "breaking things" doesn't lead to harm. A gamified cybersecurity training tool can provide students with opportunities to explore without risk, and for instructors to dynamically adapt game mechanics to meet students evolving skill levels.

5.3 The Cyber Escape Room Game

"Cyber Escape Room" is a first-person 3D video game developed with the Unity Game Engine. The game loads into a camera view emulating the players viewpoint, placing them inside a building with physical and digital security infrastructure based on a realistic system. The user must find and interact with a workstation using traditional security tools and techniques to meet objectives of increasing difficulty in order to test their current skills from previously-learned cybersecurity concepts. This requires collecting items and information.

Player actions are split into physical and digital tasks. Physical tasks include item collection (e,g, flash drives) to motivate the need to guard data in the physical realm. These drives contain information necessary to solve riddles and open doors. Digital tasks involve the use of the workstations inside the cyber range.

Completing each task successfully provides students with "decryption codes", necessary to continue their quest, e.g. by providing additional information, items, or access. Gameplay-motivating elements include timers, goal-completion achievements, and leaderboards, both in singleplayer and multiplayer modes. Single player mode follows a particular sequence; multiplayer games may be competitive (1 vs. 1) or cooperative.

5.4 Cyber Range - "Cyber Threat Realm"

The importance of hands on training has been well established, especially in a field such as cybersecurity. Any theoretical background must be enriched and supported through realistic training exercises. In recent years there is an increasing trend in the development and use of online training platforms, even in vocational courses [174]. Realistic training enhances the preparedness of personnel at all hierarchical levels, either horizontally or vertically. Personnel may include administrative or technical cyber-security professionals at all levels. Cyber-Security training is becoming more and more common. Potentially, the only way in which cyber-incidents can be handled and even prevented adequately is through rigorous awareness or training exercises [175]. The need for cybersecurity awareness or deep expertise increases in a linear manner, a fact that increases the importance of educating personnel. More highly skilled cyber-security professionals are needed by the industry as the number of cyber-threats and ingenuity of attackers is growing. To be able to meet the increasing requirements, the synergies of higher education must be considered along with professional training and certification programs, to establish the basics of deep learning in cyber-security. This can be addressed by new teaching and knowledge transfer methods that rely on cyber-ranges and technical exercises. Research also plays an important role in dealing with advanced cyber-threats. The knowledge obtained is also validated with educational scoring mechanisms to enable trainees to efectively gain intuition into their current cyber-security skill set.

A Cyber Range provides the means for hands-on Cyber Security training and can be considered as a powerful tool for providing quantitative and qualitative assessment for such training activities. It is also a sandboxed environment where high risk cyber security exercises can be carried out in a safe manner. A cyber range includes components such as network devices, network management services, servers, workstations, application software and management of computing resources [176]. It allows the creation and execution of a multitude of individualized or team based cyber security training scenarios. Some of the scenarios that can be deployed in a cyber range include cybersecurity awareness, pen-testing exercises, exploitation, social engineering, forensics, network security, malware analysis etc. Training curricula can be also linked to professional certification

programs supported through such learning platforms.

5.5 Design and Development of the game

5.5.1 Application Scenes

In the current development state, the game consists of three scenes: the main scene (Figure 5.3), the singleplayer and the multiplayer scene. Users can explore all the different scenes via a button panel and they can also make a variety of changes concerning graphics and audio in the options section (Figure 5.4).

Figure 5.3: Main Menu Scene

5.5.2 Environment

The main 3D environment comes from a Unity Asset that has been customized. It consists of four floors and each floor has many different rooms. Each room may contain multiple game Objects like desks and computers that users can interact with to complete objectives and escape these rooms (Figure 5.5).

Figure 5.4: Options Panel

Figure 5.5: Screenshot from the main scene environment

5.5.3 Game Mechanics

When the game loads to the main menu, a Virtual Private Network (VPN) application is launched. The player/student has to connect to a specific VPN in order to link the 3D environment of the game with the Cyber Range.

After the player spawns into a room, he or she has to locate different clues to be able to escape from it. These clues can vary. In most cases, users should find a suitable in-game computer and

Figure 5.6: In-game computer interaction

interact with it. When they press the "E" key on their keyboard, a browser link appears (Figure 5.6). Then, users can login with their "Cyber Threat Realm" account to a console and complete specific cybersecurity tasks created by their tutor/instructor. Upon successful completion, they acquire clues like access door passwords. This way, when they approach a rooms door an indication to enter a password appears (Figure 5.7). If students have the correct password, the door opens and they are able to escape from that room (Figure 5.8).

Figure 5.7: Password input field

Figure 5.8: A correct password lets the door open.

Further to the in-game computer interaction, clues can be found from interactable objects like USB Flash Drives (Figure 5.9) or even monitors that are autoplaying custom videos (Figure 5.10). The videos are streamable content coming from external sources. Both videos and USB flash drives can spawn in random locations creating unique user experiences (UX) each time a student plays the game, maximizing replayability and limiting students abilities to cheat.

Figure 5.9: Collecting a USB drive.

Cyber Escape Room also contains various 2D User Interface (UI) elements. One of the most

Figure 5.10: A monitor showing a streamable video

important is the timer (Figure 5.11) that starts once the level is loaded. Players can compete with each other in terms of time of objective completion and they can view their scores just by pressing the "O" key on their keyboard (Figure 5.12).

Figure 5.11: Game timer

Finally, when a student completes the level, the results are sent to the tutor automatically.

5.5.4 Modes

Cyber Escape Room, currently has two main modes: single player and multiplayer. In single player mode, students have to solve a series of cybersecurity tasks as described above. Upon successful completion of all objectives, they complete the level. In multiplayer mode, we are currently developing a 1vs1 scenario. In this case, two players are entering an ICT infrastructure in a building and one is the cyber attacker and the other is the defender. They both have access to a computer that gets them to the "Cyber Threat Realm" interface. If the attacker wins, he or

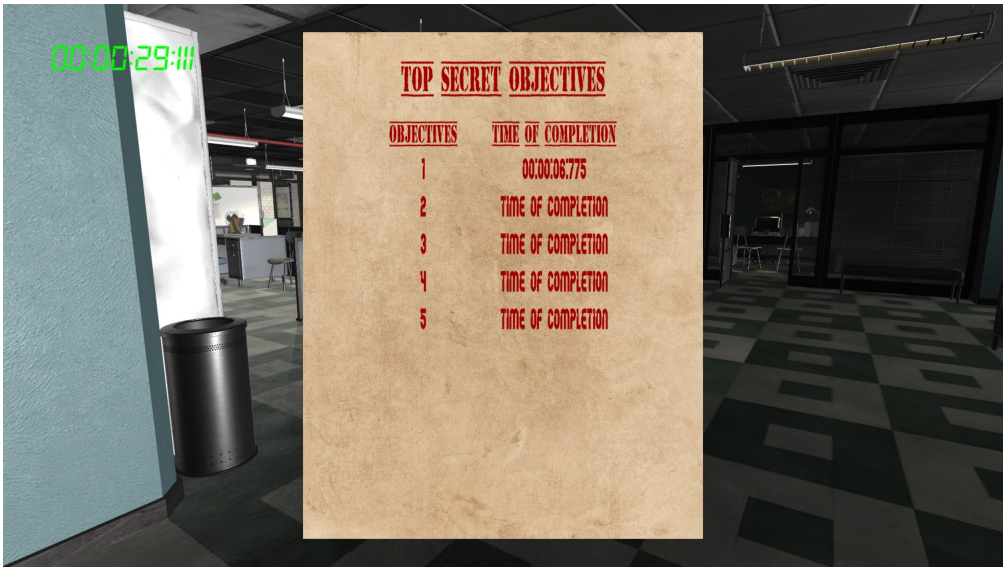


Figure 5.12: Objective completion panel

she acquires the password and then escapes the room. On the other hand, since time is a crucial element, if the timer reaches a certain threshold value and the cyber defense was successful by not letting the attacker escape, then the victory belongs to the defender.

5.6 Learning Management System (LMS) Virtual Learning Environment (VLE)

An LMS platform (Moodle), which is also a Virtual Learning Environment (VLE), resides on top of the Cyber Threat Realm, described in section 2.1, and can also be used to provide visualisation of progressive scores. The platform serves as an educational tool that can automatically import the scores from the video game and by using appropriate scales and weights can add the video game score to other grades comprising the total grade for each user. The total grade is a combination of weights from the score obtained in the videogame along with the grade of any LMS quizzes completed by the users. The users will have to complete a set of quizzes based on the scenario of the videogame to ensure that any taught cyber concepts were understood. Through the use of the LMS platform, users have access to their user profile portal which allows them to view a combination of all evaluations across different exercises and courses they have completed (Figure 5.13).

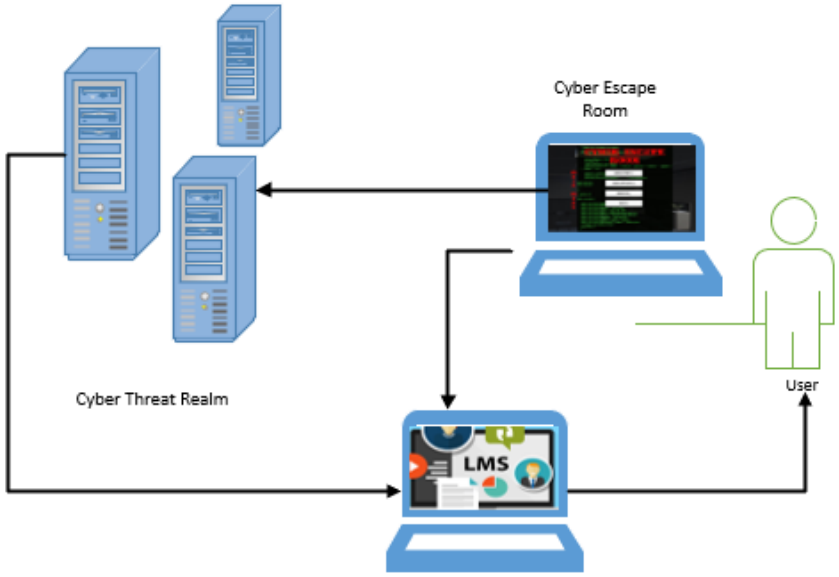


Figure 5.13: Cyber Escape Room - Cyber Threat Realm Flow

5.7 Playtesting Initial Results

The game is currently under development by a group of academics, game developers, and educators. It has not been deployed in the classroom yet, though it has been designed from the ground-up with Cybersecurity Masters degree students in mind. Game design and mechanics are designed with a pedagogical focus, and the authors have conducted small-scale play-testing to establish that the game mechanics work as designed, with four researchers having tested different versions of the application, iteratively modified to improve gameplay and to better-align with teaching needs.

The alpha version of the game included invariant passwords known a priori to the testers, static object locations and repeated video clips. This version of software proved the feasibility of developing a security-focused game and demonstrated the performance of simple game mechanics, while establishing a need for refinements to improve replayability and minimize rote-memorization to ensure that concepts, rather than gameplay mechanics, are learned. This version further concretized the overall arc of the software as being effective, by developing a playable main scene and

implementing an effective transition from gameplay to a completion screen when objectives were met.

A subsequent version of the game, tuned based on develop and author feedback, included pseudorandom password generation each time testers loaded the main scene, as well as increased complexity in identifying the correct clue to extract the password needed to open the door to a locked room. Currently, another revision is in testing, and this is expected to be the final internal build prior to releasing the application to the online platform for student learning. In the next build, clues and objects will spawn to different, predefined locations to minimize the benefit of sharing answers and to increase gameplay satisfaction for students. This modification, along with the random passwords, offers a unique experience every time a tester loads the game.

Across the designed revisions, we have tested gameplay and validated that the mechanics are well-aligned with teaching needs for courses taught by the authors. Further, the software effectively integrates with a Learning Management System (LMS) to correctly-upload user performance metrics (objective completion times and random passwords generated by game).

We are planning to solicit feedback from the students the upcoming semester and based on student sentiment, suggestions, and academic performance, we will make any necessary changes and proceed with releasing a final version the next academic year

5.8 Conclusion and Future Work

The game stands out for its tutor-friendly design, customizability, and extensibility. There are tools for the instructor to change passwords, in-game links to web content, embed external videos or information feeds, and more to provide clues, notifications, or false leads, without involving any programming expertise or intervention. The ability for instructors to reengineer the game without changing source code is a significant advantage that will make this a highly-effective teaching tool suitable for educators of all backgrounds and across grade levels.

In the future, there are additional features and scenarios that can be developed for this game. An important feature we are planning to add is an online leaderboard panel. This will provide an

opportunity for students to view the scores and times of completion of other students, providing a competitiveness element for cyber security training and encouraging students to repeat levels, reinforcing their learning.

The Cyber Escape Room has the potential to change cybersecurity education by creating an easily-accessible and extensible tool that allows students to experiment and receive feedback in a safe and controlled environment. Not only will it allow university students but also distance learning students or even a broader audience of eligible practitioners to engage in active learning of cybersecurity.

5.9 Acknowledgements

The present escape-room-themed game will be used for distance learning purposes, by students of the MSc. program of Cybersecurity (OUC). Together with Dr. Politopoulos (NTUA) and Dr. Siegel (MSU) I would also like to thank OUC teams that played a significant role in the project. The Cyber Range part is fully developed by the "Cybersecurity and Telecommunications Research Lab" of OUC Professor Stavros Stavrou and Dr. Adamantini Peratikou. The Moodle platform is customized and maintained by the OUC eClass team.

This research project was published at the International Technology, Education and Development Conference 2020 (INTED2020). It is entitled as "Cyber Escape Room: An Educational 3D Escape Room Game within a Cyber Range Training Realm." This game received the Gold Award at the Cyprus Education Leaders Awards 2020 for the category "Digital Education".

CHAPTER 6

USER-CENTERED GAMIFICATION DESIGN AND DEVELOPMENT

This chapter offers an addon layer to gamification. Following the previous chapter where we "zoomed in" to gamification to reveal internal interconnections, in this section, we add an external layer of design on it, showcasing the user-centered approach in the designing process. Via the advanced AV-Pedestrian interaction simulator, we suggest that the human factor also needs to be taken into account for the gamification elements of simulators, and thus, with the present study, we put this at the center of the picture. (Fig. 6.1)

This research part is a cross-disciplinary study exploring the pedestrian-autonomous vehicle interactions in a safe virtual environment. In the upcoming sections we first describe current tools for modeling such interactions, and then motivate the need for the design and development of a new application that provides a pedestrian point of view research capability. Following the tool's development phase, we conduct a three-step user experience experimentation in various scenarios and received valuable insights. Participants have the opportunity to answer questions before and after using the application. Behavioral results in virtuality, tended to simulate real life better when

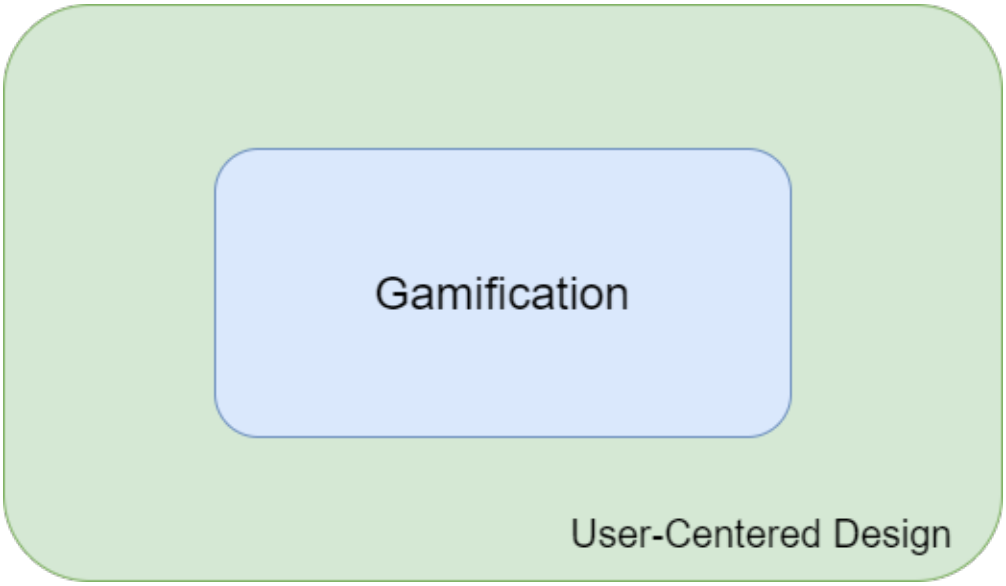


Figure 6.1: User-centered design as gamification enhancement

there were consequences comparing to scenarios where there were no consequences. The tool developed appears to raise participant awareness of autonomous vehicles, which is an important step considering recent public distrust of AVs. Future work will aim to more concretely establish this and other findings.

6.1 Introduction

Due to the advancements of modern technologies automated vehicles (AVs) are an obvious next step in the evolution of the transportation. AVs have the potential to lead positive societal change, including reduced carbon emissions, travel costs, and transit times [177, 178, 179, 180]. Beyond these impacts, AVs bring about the potential for enhanced safety relative to human driving, as human error is a leading cause of road accidents as a result of driver distraction and other factors [178]. Despite AV's potential benefits, there seems to be a distrust about their safety [179, 181]. In particular, AV-pedestrian interaction poses unknowns, in part due to the complexity of capturing real-world data from pedestrians without undue risk. In order to safely conduct this research, there is an opportunity to instead study interactions in (realistic) simulated virtual environments.

Studies have demonstrated simulations' ability to generate realistic data, including for AV-pedestrian interactions [182]. Although there are some great efforts presented in academia, we also noticed the lack of public availability of such solutions [182]. So, this is an important factor for us since our intention is to release our proposed solution as a free-to-use solution useful even to non-technical researchers. On the other hand some other applications are available for public use, such as AirSim and Carla, though those applications are from the perspective of the automated vehicle, while our application is taking a more pedestrian-centered approach. This allows our application to build on prior art while also setting things from a new perspective that will be necessary in automated vehicle research.

Complementing prior art, our team has developed a first-person simulator that allows participants to control a human avatar within an urban environment to allow individuals interact with virtual automated and virtual human-operated vehicles as they attempt to cross a street. A config-

urable application allows researchers to effectively capture representative interactions and can also be used as a way to train participants on AV safety while building trust and awareness, easing the safe and efficient adoption of driverless technologies.

The present manuscript describes the process by which our simulator was developed, tested, and validated. Included is a summary of prior art in simulation related to AV's, and we provide comparison of our simulator to those existing in order to identify opportunities. We then discuss goals and applications of our simulator, and experimental results examining human interaction with our tool and resulting iterative design changes. We close by exploring how our tool can be applied to advance the efforts of the AV industry, and by describing planned next steps in using the simulator to capture data for informing AV design and human interaction.

6.2 Motivation

The Pedestrian Tool is an easy-to-use application that allows researchers to gather data and better understand how the average pedestrian will interact with automated vehicles. One of the main ideas that our team focused on was accessibility, as we wanted to make sure that the tool was quite user friendly. This will allow users to better understand interaction patterns between humans and automated vehicles through data collection using the tool. Although there have been other tools that are similar to what we have created (see subsection 6.3.2), these simulations struggle with accessibility issues. Some simulations are private and therefore lack public support and others can be used only by those with an advanced coding skills. Further to this, other tools are mostly vehicle-centered where our work will try to approach the whole problem from the perspective of a pedestrian. So, in short, we have identified the need of an easy-to-use, plug-and-play and pedestrian-centered gamified research tool for the field. Thus, our application will try to bridge this gap offering the community a tool that can be used by professionals in multiple domains.

This tool will also become important due to approaching reality that automated vehicles are beginning to be introduced into the general public, and with this a lot of questions have been raised about their interactions with humans. There is a general distrust surrounding automated

vehicles due to a lack of understanding of how they work or what they will do, so this tool will allow researchers to better understand how to combat this distrust while also allowing the general public to interact with automated vehicles in a safe and harmless environment. The stigma labeling automated vehicles as 'unpredictable machines' could create additional challenges against their involvement in society and potential commercial failure, and we believe that their benefits far outweigh their drawbacks. So, one of the main goals of the tool is to raise awareness and improve the public perception towards automated vehicles and their safety.

6.3 Prior Art

AV's have the potential to benefit the efficiency and environmental friendliness of our world [178, 179, 180]. As a result, most individuals surveyed would consider using automated vehicles if and when they become available for general public use [178, 181]. However, some people do not trust AV's, citing concerns about privacy, safety, and uncertainties with regards to algorithm performance. [178, 183, 184, 185]. Reig et al. concluded that the general public who do not support AVs may not understand AV capabilities and lack education on the topic [186], suggesting that education may drive support. Even this, however, may not be sufficient to comfort individuals concerned about uncertain algorithm performance, such as how vehicles will perform in the absence of sensor data or external communications [187, 188]. Similarly, a lack of AV data provided to humans may in fact lead to reduced AV performance. Studies are therefore exploring the best ways for autonomous vehicles to communicate information with human drivers and pedestrians in real time, better allowing humans to predict the algorithm's behavior and plan accordingly [180, 189].

Pedestrian/AV safety is an ongoing research concern. Exploration in the area of pedestrian safety has historically been limited by the constant inability to put real human subjects into dangerous situations [182]. Recently, the use of virtual environments to research pedestrian-automated vehicle interactions has been explored, with Deb et al. concluding that virtual environments can simulate pedestrian actions with sufficient realism to mirror real-world results [182]. Jayaraman et al. applied a similar concept, using the Unity Game Engine to develop a simulator that collected information

on pedestrian-AV interactions [190], creating "an accurate long-term pedestrian crosswalk behavior (trajectory) prediction model" from synthetic data [190].

Many of these tools and applications are designed using popular game engines. The Unity Game Engine is one option that has been used successfully in pedestrian simulation [190]. This multi-platform engine allows developers to create simulated environment where they can spawn or destroy of (Game)objects [161, 7]. The design and development process of a tool has become quite easier since a great variety of 3D objects can be acquired through Unity's extensive asset store that eases access to CAD or even scriptable physics models[161, 191]. This flexibility allows researchers to develop simulations, including gamified solutions, iteratively in order to respond to user feedback, particularly at early stages[192].

Another important factor in doing virtual research is determining the ideal mode of interaction (2D, 3D, or some form of XR including VR, AR, or MR). Virtual Reality (VR) allows for participants to feel as though their actions have real consequences [191, 193]. However, working with virtual reality can cause "virtual reality sickness" as a result of prolonged time spent in virtual environments [182], suggesting breaks may need to be part of game design. Despite this, VR has been used successfully in AV studies such as Shahrdar et. al's research. This experiment has participants immerse themselves in a VR simulation and test their trust of virtual AVs after that vehicle made a mistake [183]. The study found that most participants were able to regain trust in an AV shortly after the vehicle had made an error [183]. Extended Reality (XR) and Mixed Reality (MR) can also be used in future experiments involving AVs, allowing for more information to be communicated to passengers and pedestrians as well as helping build trust between humans and AVs [194].

6.3.1 Embodiment in Virtual Environments and Video Games

Virtual Environments and and video games captivate their audiences by allowing individuals access to non-typical experiences. This corresponds with the idea of embodiment, in which a player assumes the role of a virtual character and in so doing subconsciously extends themselves

to inherit that character's goals and aspirations, strengths, and limitations [195, 196]. Embodiment is an important concept in video games as player engagement depends upon how well the player can take control of their virtual character; without sufficient interactivity, games become similar to other forms of media including films or books. In order to have a strong sense of embodiment, one must design a virtual world around the character that helps fully realize the goals of the character while also playing to the characters strengths [195]. This allows the player the ability to use the character's affordances effectively to achieve an end goal.

Embodiment is also important in gamified research, although there is debate about the efficacy of embodiment and its limitations on immersion. Alton raises the question of whether or not one should refer to current virtual character-player relationships as embodiment, because embodiment requires physical reactions and feelings that come as a result of the virtual experience [197]. It may be argued whether or not these criteria are met by VR or even video games in general, which leads to some ambiguity in the term of embodiment in virtual gaming. Although this raises concerns about the idea of embodiment in gamified research, embodiment has been proven effective in virtual environments that allow users to take control of an avatar and explore, e.g. a study done by Kiela et al. found that embodiment in video games is a valid first step to developing AI [198]. In this study, humans virtually simulate what a human would and should do in a situation, and then an AI can uncover patterns from the humans, building upon these to refine an algorithm that can be used to simulate what a human would do in the same situation [198]. This allows algorithms to constantly improve without feedback from human subjects in the virtual world. Another study by Gonzalez-Franco and Peck found that the use of embodied avatars in VR has led to higher levels of user immersion inside simulation [199]. This is important as the more immersed a user is in a simulation, the more realistic the simulation will feel to the user, leading to the creation of better-representative real-world data.

Figure 6.2: Screenshot from the Carla Application

6.3.2 Pedestrian-related research applications and other software

To better understand how our simulation compared to other applications of similar purpose, we decided to explore similar applications to appreciate where other automated vehicle simulators succeeded and fell short as well as how we could set our application apart. In order to do this, we selected three applications, with the first two being chosen as they are some of the most popular in the field of automated vehicle simulation, and the third application as it most closely resembles our goal of focusing more on the pedestrian than the vehicle. The first application was Carla Simulator [158], which was developed by Carla Team. Its an open-source automated vehicle simulator built with the Unreal Game Engine that works with the OpenDRIVE standard to define the roads and other urban settings. As mentioned above, Carla has grown to be one of the most admired options for automated vehicle simulations over the years and so it has built a growing community around it.

The second application that we tested was AirSim [200], developed by Microsoft Research. It is a computer-based simulator that was built for drone and self-driving car research. Like the Carla application, this simulator is built upon the Unreal Game Engine, but there is also an

Figure 6.3: Screenshot from the AirSim Application

experimental version based on Unity in the works. This simulator is primarily used for research purposes surrounding AI and experimentation with deep learning, computer vision, and reinforcement language.

The final application that we looked at when exploring competitive applications was a tool developed by Deb et al. for research at Mississippi State University [179][182]. This simulation is not publicly available, though from the research papers released about it, we can conclude that the simulation focuses on pedestrians crossing an intersection in virtual reality using the HTC Vive headset, which is close to the goal of our team as well. The tool was also developed upon the Unity Game Engine, setting itself apart from the other two applications while also being similar to ours.

In order to compare these applications, we analyzed each application based on five categories, which were:

- **Availability** All simulators are targeting the academic community; which simulators are free and/or open-source?

- **Engines and Technical Development** What game engine are these simulators built on and

Figure 6.4: Screenshot from the Mississippi State Tool

are there any expandability options?

- **Usability**, Type of Application, and Target Audience Identify what the target audience is based on what type of prior knowledge the user needs to utilize the simulator and conduct research with the simulator. Regarding types of application, does the application use VR, desktop, or both kinds of implementation?

- **Realistic Graphics Environment** What is the quality of the 3D environments and models/prefabs used.

- **Pedestrian Point of View** How is the pedestrian feature implemented and could it be used for training or other types of research?

The goal of comparing all of the tools with the aforementioned criteria was to better understand what each of these applications did well so that we could better innovate in our tool while also understanding where these applications did lack so that we could add features to improve the experience and set our application apart from the competition.

	Carla Simulator	AirSim	Mississippi State University Tool
Availability	Free and Open Source	Free and Open Source	Publicly Unavailable

Table 6.1: Availability Comparison

	Carla Simulator	AirSim	Mississippi State University Tool
Unity	No Compatibility	Experimental Compatibility	Compatible
Unreal Engine	Compatible	Compatible	No Compatibility
Technical Development	C++, Python	C++, Python, C#	No Compatibility

Table 6.2: Game Engines and Technical Development Comparison

The first comparison that we made was the availability of each simulator. As can be seen at Table 6.1, Carla Simulator and AirSim, the two most popular tools available, are both free and open source. This fact gives these tools a major advantage over the Mississippi State University Tool as they have had years to build a community that supports their applications, and this is why they are considered some of the most dominant applications in the field. Considering the Mississippi State University tool, it has potential based on research results published about it, but it isnt publicly available and therefore we cant say if it will become as viable as the other two applications.

The next category compares the applications game engine and technical development compatibilities. As shown at Tables 6.2, Carla Simulator is built upon the Unreal Engine and therefore is compatible with it, allowing users with knowledge in C++ to work on it. In addition, some knowledge of Python is also required with the deep learning elements of the self-driving vehicle in the virtual environment. AirSim is also built upon the Unreal Engine, and therefore it requires users to have the same C++ and Python knowledge in order to do any technical developments on the application. However, AirSim also has experimental compatibility with the Unity Game Engine, and therefore it increases its community by allowing for technical developments with the C# programming language and its large community of developers. Finally, the Mississippi State University Tool also taps into this C# community through its use of Unity Game Engine, but as

	Carla Simulator	AirSim	Mississippi State University Tool
Ease of Use	Hard to Use	Hard to Use	Easy to use
Type of Application	Desktop	Desktop	Virtual Reality
Target Audience	Mainly Programmers or Engineers	Mainly Programmers or Engineers	Academia Audience

Table 6.3: Usability and Target Audience Comparison

	Carla Simulator	AirSim	Mississippi State University Tool
Graphics	High	High	Average

Table 6.4: Virtual Environment Visual Fidelity Comparison

mentioned before, it is not publicly available and so no technical development can be made to the application.

The usability and target audience category analysis (Table 6.3) shows that the two main tools, Carla Simulator and AirSim, require strong programming skills in multiple programming languages and are relatively hard to set up, especially for users who are inexperienced. Despite all of their excellent features, this drawback limits the audience of these applications to programmers, engineers, and well-versed automated vehicle enthusiasts. The Mississippi State University Tool succeeds comparatively in this category due to its ease of use and its even easier setup due to it being mostly an export build of a Unity Engine application. This allows the tool to be used by programmers, engineers, enthusiasts, and researchers who may want to conduct research in the simulation relating to automated vehicles and pedestrian interactions. Another major success for the Mississippi State University Tool is that it is a virtual reality application, which allows users to become more immersed in the simulation.

The next category compares how realistic the graphics of each simulation are (Table 6.4). The first two simulations, Carla Simulator and AirSim, both have multiple scenes and use 3D assets that are of AAA quality, allowing for an incredibly immersive experience within the simulation. The

	Carla Simulator	AirSim	Mississippi State University Tool
Pedestrian Point of View	No Pedestrian POV	No Pedestrian POV	Pedestrian POV

Table 6.5: Pedestrian Point of View Comparison

Mississippi State University Tool falls short in this category, despite having average graphics, its graphics dont compare to those of the first two.

The final category is the implementation of the pedestrian point of view in the application (Table 6.5), which is one of the most important categories for our research. Both Carla Simulator and AirSim lack in this category since their focus is on automated vehicles rather than pedestrian research. Despite being excellent applications, there is no user control of pedestrians as they are more AI-oriented. Both of these applications do, however, allow for customizable scenarios with pedestrians, such as volume and diversity of pedestrian populations. The Mississippi State University Tool focuses on the pedestrian point of view, allowing users to embody the pedestrian in virtual reality..

This competitive analysis allowed us to realize that there is an opportunity to create a new pedestrian-centered application that we will be able to use to conduct training research surrounding automated vehicle awareness. So, in order to allow our application to succeed, we needed to create a human-centered application for training and awareness since making another car-focused simulator would make it hard to compete due to the dominance of the applications Carla Simulator and AirSim. We also intend to make our application publicly available (although not open source) as it would allow our application to have more exposure and feedback for future upgrades. Also, ease of use was a priority for our team as we need our tool to be used not only by those well versed in the technical fields but also by participants in research studies and other scientists with diverse backgrounds. Graphics should also be realistic in order to better simulate and immerse users in the experience. Our application focuses on developing a desktop version with a first-person controller, but we do plan on developing and releasing a virtual reality version in the future as well.

Finally, although we developed our application in the Unity Game Engine, some features will be customizable with external files, allowing users and testers to customize scenarios without having to change anything within the source code.

6.4 Pedestrian Tool Design and Development

6.4.1 Methods

The designing process of our tool, was really important. Following the principles of research-creation [52], we needed to create a meaningful [51] interactive experience to our users. Thus, we needed to design a serious gamified tool [11] that builds awareness and at the same time educates [10] the participants on autonomous vehicles via a realistic simulation environment.

Very significant part of our design methodology was the embodiment element since users will take the role a virtual pedestrian. This role of embodiment is explained in more detail at the Subsection 6.3.1.

In terms of development, our application was developed using the Agile methodology and more specifically, the Scrum model [53]. Practically, we developed and conducted parts of our research with stable builds of our tool, working in sprints, while we continued iterating [54] until we reached stable builds.

6.4.2 Development

6.4.2.1 Scenes and Scenarios

The tool is developed in Unity Game Engine. For its development, we used both free and purchased assets from Unity Asset Store. It is a first person perspective gamified application and it features a main menu (Figure 6.6), three different scenario scenes in an urban environment, and a game over scene (Figure 6.8).

Once the game is loaded to main menu, the participant selects one of the three scenarios. These three scenarios are taking place in the same urban environment but the cars (virtual AVs and human

Figure 6.5: In scenario 3, the AVs are identified by the green light underneath. The light appears when the pedestrian is 15 meters or less away from the vehicle.

driven) are having different behaviors.

- **Scenario 1:** All cars are stopping safely before the user-pedestrian. The possibility of being "hit" is very low.
- **Scenario 2:** Some cars may stop before the user-pedestrian. Users do not know which car (AV or human driven) will stop or not. The possibility of being "hit" is high enough.
- **Scenario 3:** Some cars may stop before the user-pedestrian while others will not. In this case, the AVs are identified by a green light indication that appears under the vehicles. This indication starts when the pedestrian-AV distance is smaller than 15 meters. The AVs will always stop before the pedestrian. (Figure 6.5)

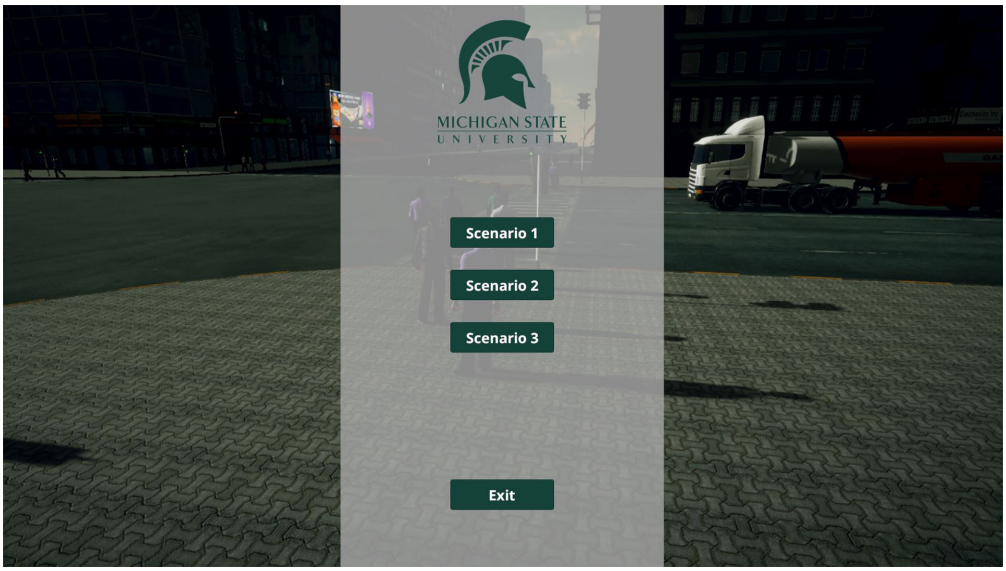


Figure 6.6: Main Menu helps the participants to switch between the various scenarios of the tool.

Figure 6.7: The urban area of the three scenarios features various buildings, vehicles, a skybox and a weather system.

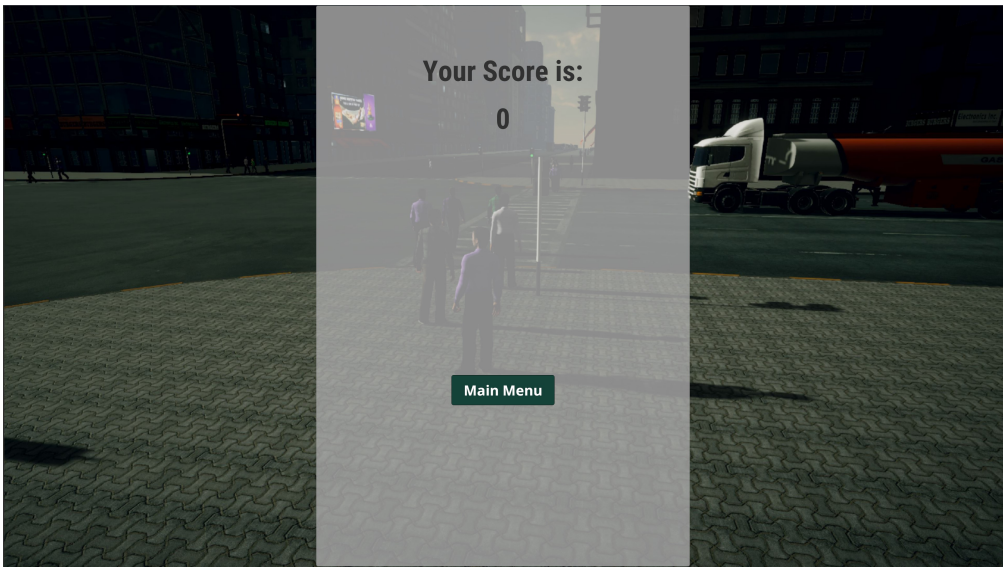


Figure 6.8: The Score-GameOver Scene shows the participants score based on their successful intersection crossings.

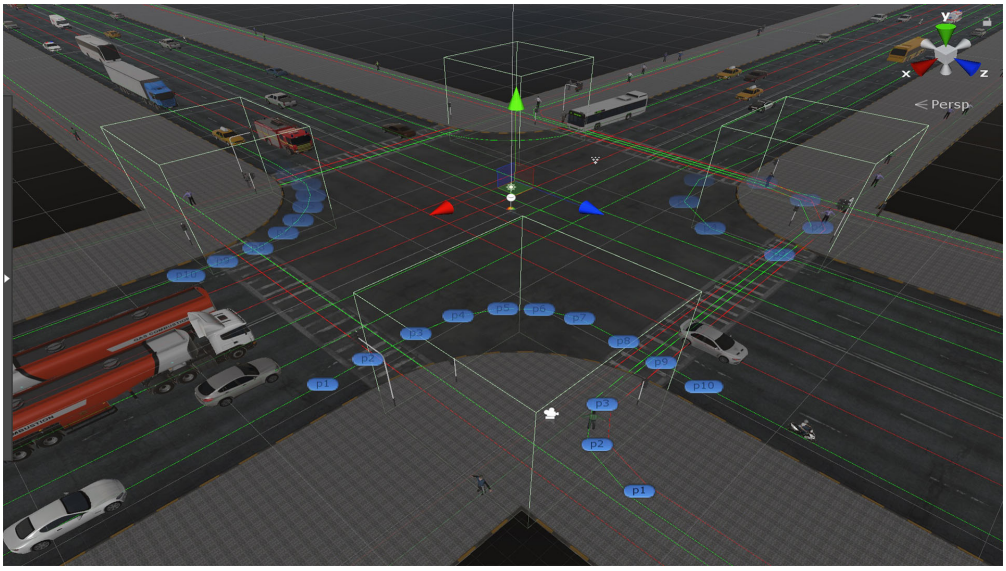


Figure 6.9: The invisible collider system is the main mechanic that the scoring system is built upon.

6.4.2.2 Mechanics over various application builds

The general idea of the training is that the users will try to cross intersections safely as virtual pedestrians while at the same time they develop awareness on self-driving vehicles.

The first build of our application featured only the urban scene and the users could walk and cross intersections freely without any consequence even if they were 'virtually' hit by a vehicle. To make the simulation more engaging and fun [1], since the second version of the tool, there were some new implementations developed. Initially, we created a **scoring system**. Thus, for every successful crossing a user gets 100 points. To achieve this, we created a non-visible collider system (Figure 6.9) that works as trigger. Specifically, for the urban scene there are four colliders, one at each corner of the pavements. Once the user gets into it, a variable gets the name of the pavement and compare it with the last name it had in memory. If it is different (meaning that a user has crossed an intersection), it adds 100 points to the UI scoring system appearing at the top right corner of the screen.

Then, we also implemented a **timer** to limit the time of the training. Most importantly, since the second build of the application, "careless" users may result in losing at the game if they are being "hit" by a vehicle. This **penalization** implementation is accomplished by another collision system achieved between the Player component and the spawning vehicle. Once a user is "hit" or the time ends up, the simulation automatically transitions to the Game Over scene (Figure 6.8) where the final score is shown.

In the third and last major build, we implemented three scenarios (see Subsection 6.4.2.1) with differentiation in vehicle behavior useful for data collection. All the scenarios are based on the same urban scene.

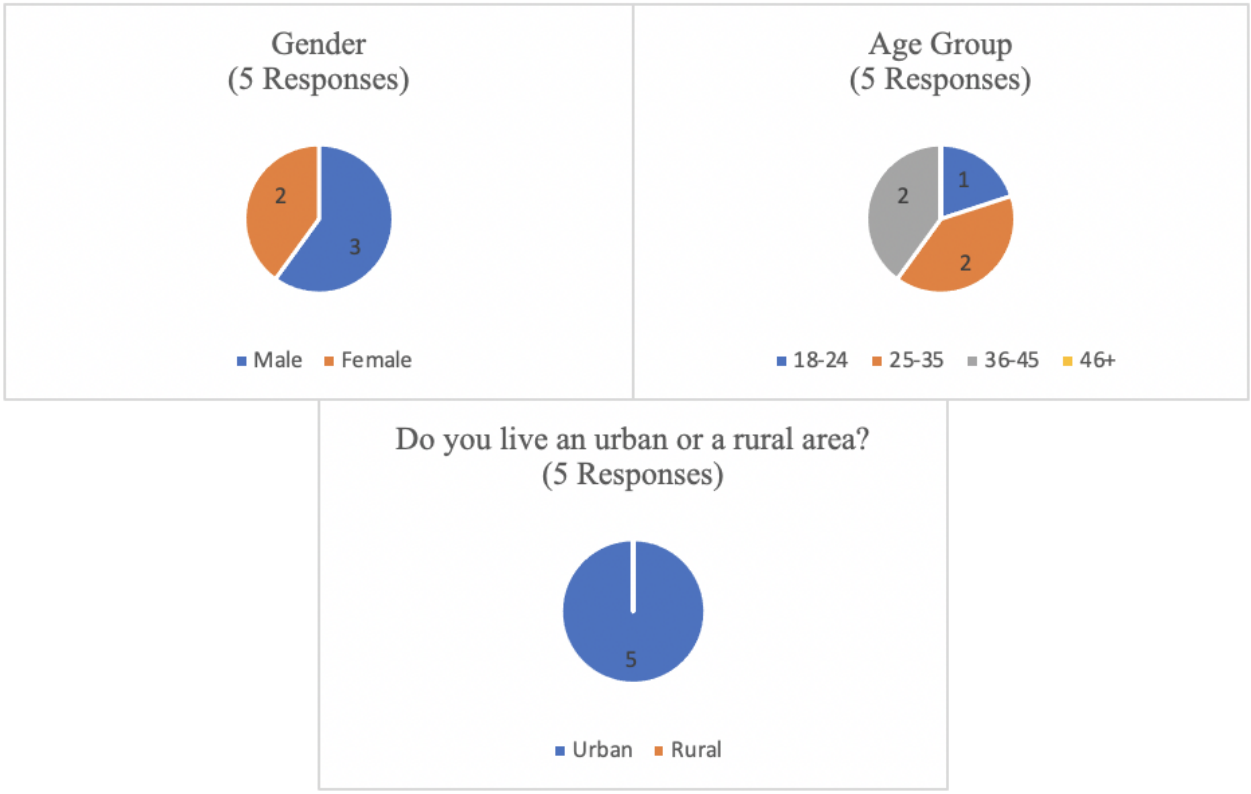


Figure 6.10: Basic demographic information of the participants

6.5 Case Study - UX Experimentation

6.5.1 Step 1: Behavior in Reality vs no Consequence Virtuality

Based on the first build of our application, We conducted a first UX experiment. The goal of this experiment was to understand how people crossed an intersection in a simulation without knowing whether the vehicles in the road were automated vehicles or not, or even if they would stop if the user walked out in front of them. We conducted the experiment with five participants. They were between the ages of 18-45, represented mixed genders, and we purposely chose people who live in urban environments (Figure 6.10), as the first build had only this option available. Demographics for each participant can also be found at the Table 6.6.

We began the experiment by asking each participant the same questions, regardless of demographic. These questions included their feelings about self-driving vehicles such as how familiar they are with self-driving vehicles, if they think self-driving vehicles will become a safer alterna-

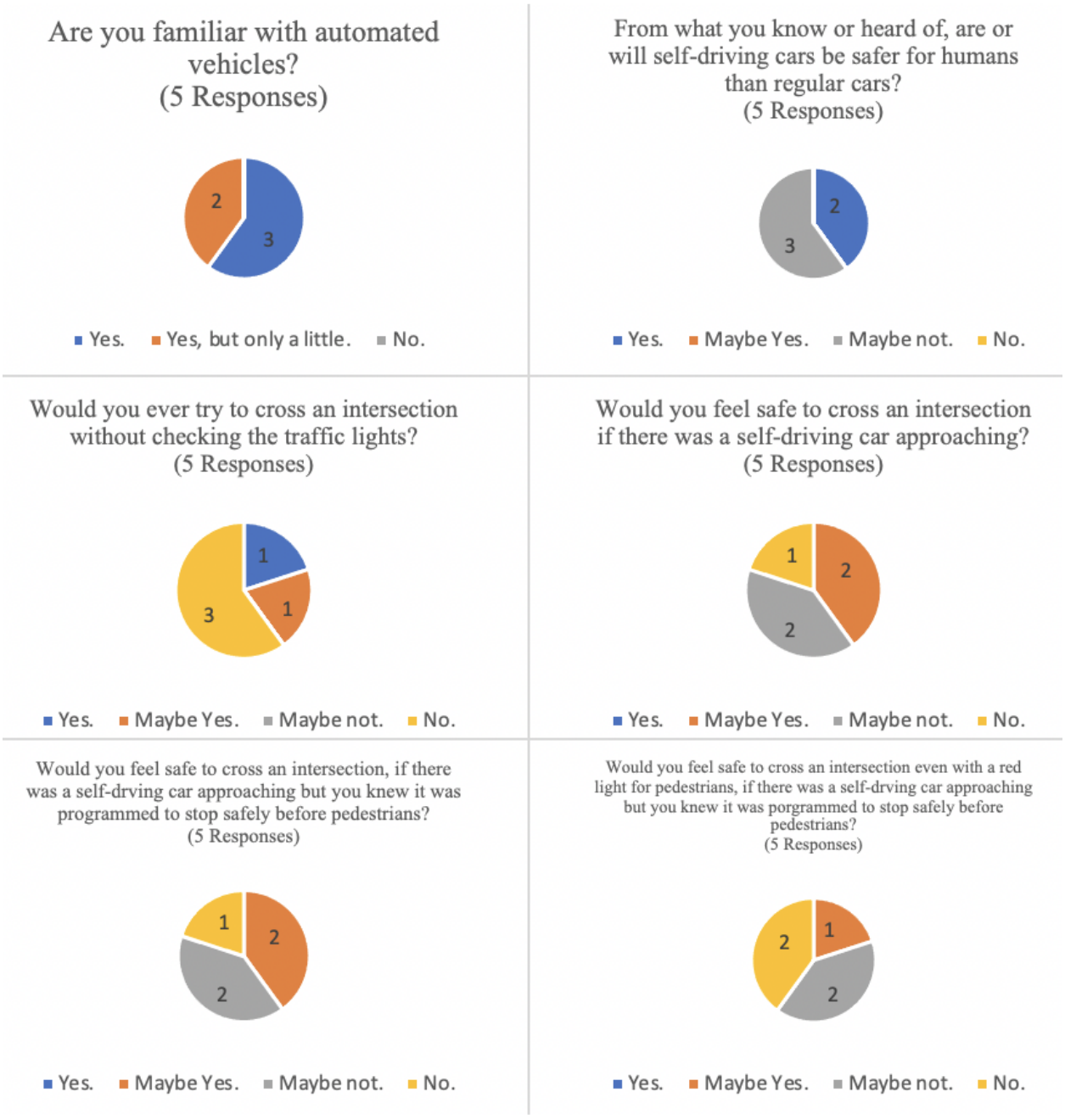


Figure 6.11: Participant responses to questions relating to automated vehicles.

Participant	Gender	Age Group	Area they live in
Participant 1	Male	25-34	Urban
Participant 2	Male	35-44	Urban
Participant 3	Female	35-44	Urban
Participant 4	Male	18-24	Urban
Participant 5	Female	25-34	Urban

Table 6.6: Basic demographic information of the participants

tive to human-driven vehicles, and various questions about different scenarios of whether or not participants would cross the street in scenarios involving driving cars and safety. Responses to these questions are represented at Figure 6.11. After participants finished the survey, they were given five minutes in the simulation to cross the intersections of our virtual city and experience the simulation, during which observers took note of statistics. After the five minutes were up, the participants were asked some follow-up questions, which included whether or not each participant found getting "hit" by the car "fun", what their opinions on the overall aesthetics were, and finally, just stating one thing about their overall experience with the tool. The results of the experiment were tracked using the following rubrics:

- **Rubric 1:** How many attempts of crossing intersections were made?
- **Rubric 2:** How many of the crossing were done safely (checked traffic lights etc.)?
- **Rubric 3:** Did the user try to cross just to check if they will be hit by a vehicle?
- **Rubric 4:** Did the user find the idea of "being hit by a car" in the game "fun"?
- **Rubric 5:** How many times did they walk and how many did they run while crossing an intersection?

This experiment taught us a lot, with one major finding from the results of the automated vehicle questions (Figure 6.10) being that the participants generally didn't trust automated vehicles

Participant	Rubric 1	Rubric 2	Rubric 3	Rubric 4	Rubric 5
Participant 1	10	9	Yes	Yes	8/2
Participant 2	12	6	Yes	Yes	3/9
Participant 3	10	2	Yes	No	5/5
Participant 4	13	2	Yes	Yes	0/13
Participant 5	11	11	No	No	7/4

Table 6.7: Simulation Observations/Question Responses

as being safe in various scenarios. To go along with this, most of the participants wouldn't consider crossing an intersection without checking traffic lights and 80 percent of participants wouldn't cross the intersection if the traffic lights told them that they couldn't cross. This shows that most of the participants generally would not take risks in real life to increase efficiency to get to their destination due to the unsafe implications of their actions. However, despite these results from the preliminary survey, observations from their simulation experiences (Table 6.7) show that four out of five participants still crossed the street in an unsafe manner in the simulation despite their tendencies to do the exact opposite in real life.

This led to our conclusion that people act differently in virtuality than in reality, and the actions of the participants as well as their post-questionnaire results further prove this. Of the four participants who attempted to walk in front of a car, three of them found it "fun" to get "hit" by the car, with one participant citing "being able to push the cars" as one of their favorite features. Based on the participants' pre-survey results, they would never consider actions such as those described above in real life as they would be putting themselves at risk. Participants stuck to this ideology early on the experiment. However, they soon realized that their actions didn't have consequences, so most of the participants decided to be riskier with their intersection crossing and soon experienced being "hit" by a car. This needed to be well improved upon the next versions of the tool to improve realism.

Participant	Gender	Age Group	Experience with AVs (usage)	Experience with AVs (General Knowledge)	Experience with games (especially vehicle-related like GTA)
Participant 1	Male	18-24	★☆☆☆☆	★★☆☆☆	★★★★☆
Participant 2	Male	18-24	★☆☆☆☆	★★☆☆☆	★★★★★
Participant 3	Male	35-44	★☆☆☆☆	★★☆☆☆	★★★★★
Participant 4	Female	18-24	★★★★★	★★★★★	★☆☆☆☆

Table 6.8: Demographic and Experience information on the participants

6.5.2 Step 2: Behavior in Reality vs Consequence Virtuality

Once the second version of the tool was developed, we conducted another UX experiment with four participants this time. the demographics along with more information on the participants are shown at the table 6.8.

This tool version offered a timer, a score mechanism for successful intersection crossings (100 points for each) and most importantly, a penalization system resulting in ending the game when a participant was virtually "hit" by a vehicle.

The users playtested the tool for two minutes each. Due to COVID-19 situation, the tool was sent and the observation took place remotely. One of the main constraints identified was that not all users had high-end computers. Thus, the experience was quite different in terms of realism when played on ultra vs medium or even vs low graphics.

The most important insight that we got by observing the users was that their behavior towards vehicles changed drastically. Although this behavior was not exactly natural, users started to avoid cars and tried either to follow the pedestrian traffic lights as in real life or even to cross more strategically. But this was a big step getting a more realistic behavior in virtuality.

An additional insight we got by observing the users, is that some of them tended to explore the overall map instead of just completing their crossing objective. This behavior is known as an "explorer" type of player based on Bartle's Taxonomy [201] [202]. Although this exploring action was really useful for playtesting purposes and for identifying bugs, it also led us to conclude that training needed to be supervised rather than allowing participants to freely use the application as a game.

6.5.3 Step 3: Testing AV Interaction & Behavioral Intentions

Following the first and second user experience tests, a final experiment was conducted with nine participants to better understand user interactions with autonomous vehicles and predictability. Final sample for this step included seven valid responses for analysis.

Including additional minor updates from UX testing, the tool was equipped with the same timer and scoring mechanism for successful intersection crossings and penalizations for being "hit". Participants were recruited through convenience sampling methods and were again observed remotely due to COVID-19 protocols.

Differing slightly from the prior format, each user interacted with three scenarios for two minutes each. The three scenarios all utilized the same controls and depicted the same environment, but were identifiable as follows: the first scenario included vehicles that always stopped for pedestrians, the second scenario included some vehicles that would stop for pedestrians and some that would not, and the third scenario included some that would stop and others that would not, but autonomous vehicles would indicate their presence with a bright green light when they were within a 15 meter radius from the user.

In order to test the behavior and perceptions across each of these three scenarios, user behaviors were observed and each user was provided with a Qualtrics survey to record their responses. Baseline questions regarding real world scenarios, pedestrian behaviors, and technological and autonomous vehicle familiarity were obtained prior to beginning the pedestrian experience, and then a series of questions were answered following exposure to each of the individual scenarios. Questions following the scenarios focused on the user perceptions of the autonomous vehicle tool and the specific actions taken during their scenario experience. Following the final scenario, additional questions were answered regarding general user experience, usability [203], realism [204], and demographics.

Pre-tests were used to evaluate current behaviors and familiarity with the present topics. This data concluded that the majority of participants would never try to cross an intersection without checking traffic lights in the real world (71%). Additionally, when asked if they would feel safe

crossing an intersection and they knew there was a self-driving car approaching, the majority of participants responded “maybe not”, regardless of the green (OK to cross) or red signal (do not cross) to cross. Additional data concluded that participants were somewhat unlikely to trust AVs to stop for an object or person in the road (M=4.7, Likert scale 1-10) and neutrally to slightly less familiar with AVs to begin with (M=4, Likert scale 1-10).

The data from the post-tests of these three scenarios led to some valuable insight. When asked about the noticeable presence of self-driving cars in the simulation, responses to scenario two (53% noticed) and scenario three (69% noticed) were as expected with an increasing awareness of self-driving vehicles. It is to be noted that some participants (37%) reported noticing self-driving cars in the first scenario when they were not present. Furthermore, when asked about the clear intentions of self-driving vehicles in the simulation, participants in scenario two were mostly unsure (57%) of intentions while a majority of those in scenario three reported that intentions were clear (57%). When asked about their intentions to follow traffic signals in the scenario, the likelihood of following signals progressively decreased from being neutral to moderately unlikely as participants encountered additional scenarios with an increased presence of self-driving vehicles. As for trusting an AV to stop in the simulation, participants became less trusting of the self-driving vehicles as their presence increased across the scenarios (scenario 1: slightly likely, scenario 2: slightly unlikely, scenario 3: moderately unlikely).

When it came to specific user behaviors in the simulation, scenario one had 71% of participants running through the simulation and then scenarios two and three with 86% of participants running through the simulation. It is to be noted that participants had the option to run, walk, or a mix of both. Additionally, participants reported becoming increasingly cautious at the onset of scenario three, as opposed to careless in their behaviors. In scenarios one and two 86% and 71% (respectively) of participants reported being careless, while only 43% reported carelessness in scenario three.

Broadly, general user experience data also confirmed that this simulation was easy to use, realistic, and not too complex. Some qualitative responses suggested additional control options for moving around the environment and additional weather scenarios, so that they could experience

and perceive self-driving cars within differing environments. As for preferences on a consumer level, participants preferred that self-driving cars signal their presence to the public through audio alerts and/or lights or stickers that are more visible to pedestrians (i.e. on tops or sides of vehicles).

Finally, the demographic makeup of this sample was 100% Caucasian, 71% male, ages between 18 and 34, and a majority of participants (57%) self-reporting that their primary residence was in a rural area. This sample size (N=7) and demographic makeup is a limitation of this study that does not allow for capturing the perceptions of a larger, more diverse sample of the population.

6.6 Conclusions

The present research provided some valuable information on the user perspective on AVs. Initially, the users' survey gave us a first indication that people are not so ready to trust autonomous vehicles being on the streets. This distrust does not only include being passengers in such vehicles but also facing them as pedestrians. Further to that, the same group of people, even though they would hesitate crossing an intersection when a self-driving car approached in real life, their behavior was different in the virtual world using the first version of the tool. Specifically, since they had no consequences when they were virtually "hit" by a vehicle they tended to be careless or even abuse cars when they stopped before them. Using the next version of the tool, where a reward and a penalization system were implemented, users started behaving closer to reality. Their approach was more strategic but surely they were more careful in virtuality since being "hit" by a vehicle (self-driven or not) resulted in a game over.

Finally, the final testing version of the tool provided us with both technical and human factor insights. Firstly, it was further confirmed that that the tool itself is usable, realistic, and not too complex for simulation use. Secondly, it was highlighted that users become more cautious when AVs are present and mixed scenarios of AV/Non-AV presence allow for greater uncertainty among participants, when it comes to AV intentions. One contradictory point is that although users are increasingly less likely to obey traffic signals, they are also increasingly less likely to trust AVs. This could speak to the gamification aspect of the simulation, in conjunction with the overwhelming

behavior to run and not walk throughout the simulation. Additionally, users reported the presence of AVs in scenario one, when they were not present. This could be a matter of clarity regarding the scope of the simulation, or an increased uncertainty of AV signaling an intentions. Further research is needed regarding these points and what impact that has on the behavior translation to the real world.

Differences in the whole experience were noted where users tested the tool in medium or low graphics. The performance of their computers played a significant role in how well they were immersed in the application.

Overall, users seemed to enjoy the tool especially, the experienced gamers who were at the same time inexperienced with AVs. They have found the tool easy-to-use since the controls were quite similar to popular games.

6.7 Future Work

The present work resulted in developing a first person gamified research tool about pedestrian-autonomous vehicle interaction. In the future, we intend to create a VR version of the tool for users to have a fully immersive experience. Moreover, we intend to have additional intersections and probably more levels in order for users to have more tasks and they would not have to repeat on the same environment. Regarding the AI pedestrian models we will add more or customize existing ones in order to make the tool even more diverse [205].Upon completing all the development parts, we also plan to release the tool freely for further use and research by the community.

As for opportunities for human factors research, this tool provides a unique option for studying public perceptions and behavioral intentions surrounding AVs. Additional studies could conduct larger, more diverse scale testing, comparative studies of simulation, VR, and reality, additional scenario studies surrounding urban vs rural environments and varying weather conditions, as well as longitudinal human factor studies surrounding trust, acceptance, perceived risk, and interaction preferences with consumer implications.

6.8 Acknowledgements

Although this work has produced some preliminary results already, our team plans to continue data collection in order to gain a better user understanding and get greater insights on AV awareness. We will capture additional data in the US, Greece and Cyprus to expand and diversify our samples. Samples appearing in the below sections were captured within US from subjects participating in an IRB-approved study. I would like to thank all our team members for their work on this project (Dr. Siegel, Jacob Rutkowski and Andrea Schaaf).

CHAPTER 7

OVERALL CONTRIBUTION

7.1 Introduction

As discussed in the previous chapters, this cross-disciplinary research offers two layers of contributions. The first layer of impact is that we managed to prove that diverse Deep Technologies can be interconnected strategically to solve problems across domains. Specifically, starting with XR and Gamification, we initially managed to create bridges with Internet of Things implementations which were working as input streamers. This way, we developed tools for data visualization (low-cost digital twins) for the automotive industry or tools where IoT streamers were working as trigger enablers of events and phenomena in virtuality for distance learning students. At the same time, we designed tools that could generate synthetic data from virtual environments with novel methodologies that could potentially train models of Artificial Intelligence. In total, we presented six projects, each of which has its own contributions to science in addition to the overall framework. The list of contributions (higher/lower level) are presented at the following sections.

7.2 Higher Level Contributions

The present research managed to create bridges among a diverse range of Deep Technologies. Starting with XR and Gamification, this work enabled readers to understand the interconnection with other technologies like IoT and AI, which are usually considered or used independently. While each of them, individually, has a lot to offer in the industry and academia, the proposed holistic framework extends their overall impact a lot further bringing a new way of thinking and approaching them. Having established this framework, we dived deeper into gamification and revealed further connections with other layers of virtuality and also suggested an external layer of design, introducing the human-centered design in gamification and in expansion in all the framework. Finally, as presented, the proposed methodology can be applied in multiple domains

with great success.

7.3 Lower Level Contributions

7.3.1 Environmental Studies Tool (Chapter 2)

- Environmental Studies Tool features a unified environmental solution for sampling and mapping used by distance learning students.

- It provides a custom-made virtual drone system (cameras on multiple heights) that follows the users' movement.

- It is an IoT-enabled application that uses external hardware to trigger in-game events and hidden objects.

- It currently uses a button hardware implementation but this can be expanded to any type of sensors in the future.

- With this solution, we suggest the novel methodology of pseudo-multiplayer. This mode works with IoT and enables instructors to trigger events to multiple users simultaneously with no additional use of server/lobby-based plugins or any other implementations. The main difference from a regular multiplayer is that users can view the events or other objects but not each other. This way we transformed a cutting-edge, asynchronous tool for distance learning into a synchronous one using IoT.

7.3.2 Virtual Car (Chapter 3)

- Virtual Car features three different applications (AR, VR and Desktop) that can be fed with streamers of data and create digital twins.

- Virtual cars, or digital twins in general, are expensive to create. With our solutions, we provide a low-cost methodology for creating IoT-enabled Avacars which are combined and visualized using various immersive technologies.

- Using some commercial assets (under development licenses), we showcased how we can spawn real 3D maps with low polygons in real time which are suitable even for low-end device VR.

7.3.3 Gamified Digital Simulator (Chapter 4)

- The Gamified Digital Simulator is a tool designed for ease-of-use and self-supervision. Using it, someone can collect data (images and corresponding steering angles) without technical knowledge of the vehicle, simulator, or data capture needs. This expands the potential user-base over traditional simulators and simplifies crowdsourcing.

- It provides an in-game AI training solution that can, without track knowledge, collect data independently. This allows some data to be generated automatically and with great results.

- It offers a resource-efficient simulation, adapting the graphical quality level of scene elements depending on the importance of each element in training a model. Elements such the track and its texture are high quality whereas the background uses low-polygon models and

low-resolution textures. This reduces the processing power needed to run the tool, expanding its applicability to constrained compute devices.

7.3.4 Virtual LiDAR Simulator (Chapter 4)

- Virtual LiDAR Simulator is a dedicated solution for LiDAR sampling, which allows for the simulation of a range of industry-standard and yet-to-be-developed LiDAR.

- It is developed using free and open-source tool only.

- It does not require specific knowledge on software or game development to capture samples. A user can just run the executable and press a key.

- The configurability of options is an easy and front-end task.

- Samples can be captured from particular object classes providing a novel segmented and filtered methodology.

- The environment used for the simulation offers a High Dynamic Range (HDR) which provides more realistic depictions of color and brightness suitable for training self-drive algorithms.

7.3.5 Cyber Escape Room (Chapter 5)

- Cyber Escape Room is a gamified tool that offers the means to solve various riddles related to Cybersecurity in a safe and virtual environment.

- Comparing to other solutions it offers randomization in riddles and escaping passwords where similar games or tools tend to offer repetitive and linear approaches.

- A novelty it offers is that it is combined with external browser-based tools like Cyber Ranges (virtual machines) and thus, provide real hands-on training to students and other individuals. Correct answers in a VM generates passwords suitable to escape in the virtual world.

- It offers automatic assessment which is sent to instructors via emails.

7.3.6 AV-Pedestrian Interaction Simulator (Chapter 6)

- AV-Pedestrian Interaction Simulator is one of the few research tools that try to explore the interactions of AVs and Pedestrians in virtual and safe environments.

- Other solutions are focused on the AV perspective but the tool we designed and developed works through the eyes of a pedestrian.

- Studies have shown that AVs could potentially be better that the average driver but the distrust still remains. Having implemented three different scenarios, not only we explore the interactions, the tool tries to raise awareness on AVs to its users.

- This project combines advanced design and development techniques along with human subject research methodologies.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] N. Lazzaro, "The 4 keys 2 fun." [Online]. Available: http://www.nicolelazzaro.com/the4-keys-to-fun/

[2] J. Siegel and S. Krishnan, "Cultivating invisible impact with deep technology and creative destruction," *Journal of Innovation Management*, vol. 8, no. 3, pp. 6–19, 2020.

[3] PTC, "Vuforia." [Online]. Available: https://developer.vuforia.com/

[4] Apple, "ARKit." [Online]. Available: https://developer.apple.com/augmented-reality/arkit/

[5] Google, "ARCore." [Online]. Available: https://developers.google.com/ar

[6] Steam, "SteamVR." [Online]. Available: https://valvesoftware.github.io/steamvr_unity_plugin/

[7] G. Pappas, J. E. Siegel, K. Politopoulos, and Y. Sun, "A Gamified Simulator and Physical Platform for Self-Driving Algorithm Training and Validation," 2021.

[8] K. Karur, N. Sharma, C. Dharmatti, and J. E. Siegel, "A Survey of Path Planning Algorithms for Mobile Robots," 2021.

[9] J. B. Hauge, B. Pourabdollahian, and J. C. K. H. Riedel, "Workshop on the Use of Serious Games in the Education of Engineers," *Procedia Computer Science*, vol. 15, pp. 341–342, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877050912008605

[10] S. L. C. D. R. Michael, *Serious games: Games that educate, train, and inform*. Muska & Lipman/Premier-Trade, 2005.

[11] M. Ott, A. De Gloria, S. Arnab, F. Bellotti, K. Kiili, S. Freitas, and R. Berta, *Designing serious games for education: From pedagogical principles to game mechanisms*, jan 2011, vol. 2011.

[12] L. von Ahn, "Games with a Purpose," *Computer*, vol. 39, no. 6, pp. 92–94, jun 2006. [Online]. Available: https://doi.org/10.1109/MC.2006.196

[13] Y. Yang, Y. Asaad, and Y. Dwivedi, "Examining the impact of gamification on intention of engagement and brand attitude in the marketing context," *Computers in Human Behavior*, vol. 73, pp. 459–469, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0747563217302261

[14] B. Wolfenden, "Gamification as a winning cyber security strategy," *Computer Fraud & Security*, vol. 2019, no. 5, pp. 9–12, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1361372319300521

[15] N. Lazzaro, "Why we Play Games: Four Keys to More Emotion without Story," *Game Dev Conf*, jan 2004.

[16] J. Moizer, J. Lean, E. Dell'Aquila, P. Walsh, A. A. Keary, D. O'Byrne, A. Di Ferdinando, O. Miglino, R. Friedrich, R. Asperges, and L. S. Sica, "An approach to evaluating the user experience of serious games," *Computers & Education*, vol. 136, pp. 141–151, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0360131519300855

[17] G. Pappas, P. Peratikou, J. Siegel, K. Politopoulos, C. Christodoulides, and S. Stavrou, "Cyber escape room: An educational 3d escape room game within a cyber range training realm," in *INTED2020 Proceedings*, ser. 14th International Technology, Education and Development Conference. IATED, 2-4 March, 2020 2020, pp. 2621–2627. [Online]. Available: http://dx.doi.org/10.21125/inted.2020.0788

[18] B. Walther-Franks, J. Smeddinck, P. Szmidt, A. Haidu, M. Beetz, and R. Malaka, "Robots, Pancakes, and Computer Games: Designing Serious Games for Robot Imitation Learning," in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, ser. CHI '15. New York, NY, USA: Association for Computing Machinery, 2015, pp. 3623–3632. [Online]. Available: https://doi.org/10.1145/2702123.2702552

[19] C. Weaver, "The new driver's ED: Game developers teach Cruise's autonomous vehicles to understand gestures made by people on the street," *IEEE Spectrum*, vol. 57, no. 9, pp. 32–37, 2020.

[20] S. R. Richter, V. Vineet, S. Roth, and V. Koltun, "Playing for Data: Ground Truth from Computer Games," aug 2016. [Online]. Available: http://arxiv.org/abs/1608.02192

[21] A. Shafaei, J. J. Little, and M. Schmidt, "Play and Learn: Using Video Games to Train Computer Vision Models," aug 2016. [Online]. Available: https://arxiv.org/abs/1608.01745

[22] D. K., "The Top 10 Video Game Engines," 2020. [Online]. Available: https://www.gamedesigning.org/career/video-game-engines/

[23] Unity, "Unity3D, https://unity.com/." [Online]. Available: https://unity.com/

[24] "Unreal Engine, https://www.unrealengine.com/en-US/." [Online]. Available: https://www.unrealengine.com/en-US/

[25] "GameMaker, https://www.yoyogames.com/gamemaker." [Online]. Available: https://www.yoyogames.com/gamemaker

[26] "Godot, https://godotengine.org/." [Online]. Available: https://godotengine.org/

[27] "CryEngine,https://www.cryengine.com/." [Online]. Available: https://www.cryengine.com/

[28] J. E. Siegel, S. Kumar, and S. E. Sarma, "The Future Internet of Things: Secure, Efficient, and Model-Based," *IEEE Internet of Things Journal*, vol. PP, no. 99, p. 1, 2017.

[29] S. Tedeschi, J. Mehnen, N. Tapoglou, and R. Roy, "Secure iot devices for the maintenance of machine tools," in *Procedia Cirp*, vol. 59. Elsevier, 2017, pp. 150–155.

[30] U. S. G. A. Office, "Internet of things: Enhanced assessments and guidance are needed to address security risks in DOD," jul 2017.

[31] A. Nordrum, "Popular internet of things forecast of 50 billion devices by 2020 is outdated (2016)," *Dosegljivo: https://spectrum. ieee. org/tech-talk/telecom/internet/popular-internet-ofthings-forecast-of-50-billion-devices-by-2020-is-outdated.[Dostopano: 11. 8. 2017]*, 2017.

[32] J. E. Siegel, "Cloudthink and the avacar: Embedded design to create virtual vehicles for cloud-based informatics, telematics, and infotainment," Ph.D. dissertation, Massachusetts Institute of Technology, 2013.

[33] S. Mayer and J. Siegel, "Conversations with connected vehicles," in *Internet of Things*, 2015.

[34] J. Siegel, S. Krishnan, B. Subirana, S. Sarma, J. Merritt, L. Joseph, and R. Arias, "Realizing the internet of things: A framework for collective action," in *World Economic Forum*, 2019.

[35] J. E. Siegel and S. Kumar, "Cloud, context, and cognition: Paving the way for efficient and secure iot implementations," in *Handbook of Integration of Cloud Computing, Cyber Physical Systems and Internet of Things*. Springer Switzerland AG, 2020.

[36] J. E. Siegel, "Data proxies, the cognitive layer, and application locality: enablers of cloud-connected vehicles and next-generation internet of things," Ph.D. dissertation, Massachusetts Institute of Technology, 2016.

[37] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of things (iot): A vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645 – 1660, 2013, including Special sections: Cyber-enabled Distributed Computing for Ubiquitous Cloud and Network Services & Cloud Computing and Scientific Applications âĂŤ Big Data, Scalable Analytics, and Beyond. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X13000241

[38] G. Pappas, J. Siegel, and K. Politopoulos, "Virtualcar: Virtual mirroring of iot-enabled avacars in ar, vr and desktop applications," 2018.

[39] J. Siegel, "Cognitive protection systems for the internet of things," *Homeland Defense and Security Information Analysis Center Journal*, vol. 5, no. 4, pp. 16–20, 2018.

[40] J. Siegel and S. Sarma, "A cognitive protection system for the internet of things," *Security & Privacy*, vol. 17, no. 3, 2019.

[41] ——, "Using open channels to trigger IoT's invited, unintended consequences," *Security & Privacy*, vol. 17, no. 3, 2019.

[42] Y. Sun, A. Armengol-Urpi, S. N. R. Kantareddy, J. Siegel, and S. Sarma, "Magichand: Interact with iot devices in augmented reality environment," in *IEEE VR Workshop on Novel Input Devices and Interaction Techniques*. 10.1109/VR.2019.8798053, 2019, pp. 1738–1743.

[43] Y. Sun, S. N. R. Kantareddy, J. Siegel, A. Armengol-Urpi, X. Wu, H. Wang, and S. Sarma, "Towards industrial iot-ar systems using deep learning-based object pose estimation," in *2019 IEEE 38th International Performance Computing and Communications Conference (IPCCC)*. IEEE, 2019, pp. 1–8.

[44] T. Mustapää, J. Autiosalo, P. Nikander, J. E. Siegel, and R. Viitala, "Digital metrology for the internet of things," in *2020 Global Internet of Things Summit (GIoTS)*. IEEE, 2020.

[45] B. T. Kumaravel, R. Bhattacharyya, J. Siegel, S. Sarma, and N. Arunachalam, "Development of an internet of things enabled manufacturing system for tool wear characterization," in *2017 IEEE 3rd International Symposium in Robotics and Manufacturing Automation (ROMA)*. IEEE, 2017, pp. 1–6.

[46] J. Siegel, S. Pratt, Y. Sun, and S. Sarma, "Real-time deep neural networks for internet-enabled arc-fault detection," *Engineering Applications of Artificial Intelligence*, vol. 74, pp. 35–42, 2018.

[47] J. Siegel, Y. Sun, and S. Sarma, "Automotive diagnostics as a service: An artificially intelligent mobile application for tire condition assessment," in *Lecture Notes in Computer Science: Artificial Intelligence and Mobile Services (AIMS) 2018*, 2018.

[48] B. E. Wiggins, "An overview and study on the use of games, simulations, and gamification in higher education," *International Journal of Game-Based Learning (IJGBL)*, vol. 6, no. 1, pp. 18–29, 2016.

[49] S. Subhash and E. A. Cudney, "Gamified learning in higher education: A systematic review of the literature," *Computers in Human Behavior*, vol. 87, pp. 192–206, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0747563218302541

[50] A. P. Markopoulos, A. Fragkou, P. D. Kasidiaris, and J. P. Davim, "Gamification in engineering education and professional training," *International Journal of Mechanical Engineering Education*, vol. 43, no. 2, pp. 118–131, 2015. [Online]. Available: https://doi.org/10.1177/0306419015591324

[51] E. Z. K. Salen, "No Title," in *Rules of play: Game design fundamentals*. MIT Press, ch. Chapter 3: Meaningful play.

[52] E. Lelièvre, "Research-creation methodology for game research," Nov. 2018, working paper or preprint. [Online]. Available: https://hal.archives-ouvertes.fr/hal-02615671

[53] S. Jesse, *The Art of Game Design: A Book of Lenses.* AK Peters/CRC Press, 2014.

[54] E. Zimmerman, "Play as Research: The Iterative Design Process," 2003. [Online]. Available: https://static1.squarespace.com/static/579b8aa26b8f5b8f49605c96/t/59921253cd39c3da5bd27a6f/1502745178453/Iterative{_}Design.pdf

[55] M. A. Khan and K. Salah, "IoT security: Review, blockchain solutions, and open challenges," *Future Generation Computer Systems*, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X17315765

[56] E. Torroglosa-García, A. D. Pérez-Morales, P. Martinez-Julia, and D. R. Lopez, "Integration of the OAuth and Web Service family security standards," *Computer Networks*, vol. 57, no. 10, pp. 2233–2249, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128613001138

[57] D. Guinard and V. Trifa, *Building the Web of Things: With Examples in Node.Js and Raspberry Pi*, 1st ed. USA: Manning Publications Co., 2016.

[58] K. Afsari, C. M. Eastman, and D. Castro-Lacouture, "JavaScript Object Notation (JSON) data serialization for IFC schema in web-based BIM data exchange," *Automation in Construction*, vol. 77, no. Supplement C, pp. 24–51, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0926580517300316

[59] "Google Stadia, url = https://stadia.google.com/."

[60] G. Pappas, J. Siegel, I. Vogiatzakis, and K. Politopoulos, "Gamification and the Internet of Things in Education," in *Handbook of Intelligent Techniques in Educational process*. Springer, 2021.

[61] S. R. G. T. S. J. P. K. C. C. P. Georgios, "AR/VR/Game-based Edutainment Applications and Real-Time Data Visualisation Technologies for Discovery Learning in the Industry and Distance Education," in *Oeb*, 2018.

[62] A. Avraamidou, S. Lambis, G. Pappas, and C. Christodoulides, "Enhancing Distance Education Students' Learning Experience Through Emerging Technologies," in *INTED2019 Proceedings*, vol. 1, 2019, pp. 3022–3029.

[63] P. Milgram and F. Kishino, "A Taxonomy of Mixed Reality Visual Displays," *IEICE Trans. Information Systems*, vol. vol. E77-D, pp. 1321–1329, dec 1994.

[64] S. Stein, "The VR future is here but no one can agree on a name for it," 2017. [Online]. Available: https://www.cnet.com/news/google-avoids-the-term-mixed-reality-in-its-ar-and-vr-plans/

[65] C. Fink, "War Of AR/VR/MR/XR Words." [Online]. Available: https://www.forbes.com/sites/charliefink/2017/10/20/war-of-arvrmrxr-words/{#}6a257faa8d07

[66] S. Ke, F. Xiang, Z. Zhang, and Y. Zuo, "An enhanced interaction framework based on VR, AR and MR in digital twin," *Procedia CIRP*, vol. 83, pp. 753–758, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2212827119307176

[67] E. Wilhelm, J. Siegel, S. Mayer, L. Sadamori, S. Dsouza, and S. S. Chau, Chi-Kin, "Cloudthink: a scalable secure platform for mirroring transportation systems in the cloud," *Transport*, vol. 30, no. 3, 2015.

[68] J. E. Siegel, D. Erb, and S. E. Sarma, "A survey of the connected vehicle landscape – architecture, enabling technologies, applications, and development areas," *IEEE Transactions on Intelligent Transportation Systems*, 2017.

[69] J. Siegel, R. Bhattacharyya, A. Deshpande, and S. Sarma, "Smartphone-based wheel imbalance detection," in *Dynamic Systems and Control Conference*, 2015.

[70] J. Siegel, S. Kumar, I. Ehrenberg, and S. Sarma, "Engine misfire detection with pervasive mobile audio," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, Cham, 2016, pp. 226–241.

[71] J. Siegel, R. Bhattacharyya, A. Deshpande, and S. Sarma, "Smartphone-based vehicular tire pressure and condition monitoring," in *SAI Intelligent Systems Conference*. IEEE, 2016, pp. 446–455.

[72] J. E. Siegel, R. Bhattacharyya, S. Kumar, and S. E. Sarma, "Air filter particulate loading detection using smartphone audio and optimized ensemble classification," *Engineering Applications of Artificial Intelligence*, vol. 66, pp. 104–112, 2017.

[73] J. E. Siegel and U. Coda, "Surveying off-board and extra-vehicular monitoring and progress towards pervasive diagnostics," 2021.

[74] J. Paulo Lima, R. Roberto, F. Simões, M. Almeida, L. Figueiredo, J. Marcelo Teixeira, and V. Teichrieb, "Markerless tracking system for augmented reality in the automotive industry," *Expert Systems with Applications*, vol. 82, pp. 100–114, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417417302221

[75] G. Lawson, D. Salanitri, and B. Waterfield, "Future directions for the development of virtual reality within an automotive manufacturer," *Applied Ergonomics*, vol. 53, pp. 323–330, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0003687015300260

[76] E. Wilhelm, J. Siegel, S. Mayer, L. Sadamori, S. Dsouza, C.-K. Chau, and S. Sarma, "Cloudthink: a scalable secure platform for mirroring transportation systems in the cloud," *Transport*, vol. 30, no. 3, pp. 320–329, jul 2015. [Online]. Available: https://doi.org/10.3846/16484142.2015.1079237

[77] J. Siegel, S. Kumar, I. Ehrenberg, and S. Sarma, "Engine Misfire Detection with Pervasive Mobile Audio BT - Machine Learning and Knowledge Discovery in Databases," B. Berendt, B. Bringmann, É. Fromont, G. Garriga, P. Miettinen, N. Tatti, and V. Tresp, Eds. Cham: Springer International Publishing, 2016, pp. 226–241.

[78] S. O'Kane, "How tesla and waymo are tackling a major problem for self-driving cars: Data," 2019. [Online]. Available: https://www.theverge.com/transportation/2018/4/19/17204044/tesla-waymo-self-driving-car-data-simulation

[79] J. Stewart, "Tesla's autopilot now changes lanesand you're gonna help it out," 2019. [Online]. Available: https://www.wired.com/story/tesla-navigate-on-autopilot/

[80] E. Kassens-Noor, Z. Neal, J. Siegel, and T. Decaminada, "Choosing morals or ethics: a possible determinant to embracing autonomous vehicles?" 2021.

[81] J. M. Tangen, K. M. Kent, and R. A. Searston, "Collective intelligence in fingerprint analysis," *Cognitive Research: Principles and Implications*, vol. 5, no. 1, p. 23, 2020. [Online]. Available: https://doi.org/10.1186/s41235-020-00223-8

[82] J. Hu, Y. Zhang, and S. Rakheja, "Adaptive trajectory tracking for car-like vehicles with input constraints," *IEEE Transactions on Industrial Electronics*, pp. 1–1, 2021.

[83] B. Balaji, S. Mallya, S. Genc, S. Gupta, L. Dirac, V. Khare, G. Roy, T. Sun, Y. Tao, B. Townsend *et al.*, "Deepracer: Autonomous racing platform for experimentation with sim2real reinforcement learning," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 2746–2754.

[84] A. C. Madrigal, "Inside waymos secret world for training self-driving cars," *The Atlantic*, vol. 23, 2017.

[85] M. Johnson-Roberson, C. Barto, R. Mehta, S. N. Sridhar, and R. Vasudevan, "Driving in the matrix: Can virtual worlds replace human-generated annotations for real world tasks?" *CoRR*, vol. abs/1610.01983, 2016. [Online]. Available: http://arxiv.org/abs/1610.01983

[86] A. J. Hawkins, "Its elon musk vs. everyone else in the race for fully driverless cars." [Online]. Available: https://www.theverge.com/2019/4/24/18512580/elon-musk-tesla-driverless-cars-lidar-simulation-waymo

[87] E. R. Teoh and D. G. Kidd, "Rage against the machine? google's self-driving cars versus human drivers," *Journal of Safety Research*, vol. 63, pp. 57 – 60, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S002243751730381X

[88] Rockstar Games, "Grand Theft Auto," 2014. [Online]. Available: https://www.rockstargames.com/V/

[89] M. Martinez, C. Sitawarin, K. Finch, L. Meincke, A. Yablonski, and A. L. Kornhauser, "Beyond grand theft auto V for training, testing and enhancing deep learning in self driving cars," *CoRR*, vol. abs/1712.01397, 2017. [Online]. Available: http://arxiv.org/abs/1712.01397

[90] C. Franke, "Autonomous Driving with a Simulation Trained Convolutional Neural Network," Master's thesis, University of the Pacific, 2017. [Online]. Available: https://scholarlycommons.pacific.edu/uop_etds/2971/

[91] L. Fridman, B. Jenik, and J. Terwilliger, "Deeptraffic: Driving fast through dense traffic with deep reinforcement learning," *CoRR*, vol. abs/1801.02805, 2018. [Online]. Available: http://arxiv.org/abs/1801.02805

[92] VDrift, "VDrift." [Online]. Available: http://www.vdrift.net

[93] V. Haltakov, C. Unger, and S. Ilic, "Framework for generation of synthetic ground truth data for driver assistance applications," in *Pattern Recognition*, J. Weickert, M. Hein, and B. Schiele, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 323–332.

[94] T. Kramer, "SdSandbox." [Online]. Available: https://github.com/tawnkramer/sdsandbox/tree/donkey

[95] F. Yu, "Train Donkey Car in Unity Simulator with Reinforcement Learning." [Online]. Available: https://flyyufelix.github.io/2018/09/11/donkey-rl-simulation.html

[96] W. Roscoe and T. Kramer, "Donkey Simulator," 2019. [Online]. Available: https://docs.donkeycar.com/guide/simulator/

[97] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, 2017, pp. 1–16.

[98] J. Marín, D. Vázquez, D. Gerónimo, and A. M. López, "Learning appearance in virtual scenarios for pedestrian detection," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, June 2010, pp. 137–144.

[99] H. Hattori, V. N. Boddeti, K. Kitani, and T. Kanade, "Learning scene-specific pedestrian detectors without real data," in *2015 IEEE Conference on Computer Vision and Pattern Recognition*, June 2015, pp. 3819–3827.

[100] A. Filipowicz, J. Liu, and A. L. Kornhauser, "Learning to Recognize Distance to Stop Signs Using the Virtual World of Grand Theft Auto 5," in *Transportation Research Board 96th Annual Meeting*, 2017. [Online]. Available: https://trid.trb.org/view/1439152

[101] J. Togelius and S. M. Lucas, "Evolving robust and specialized car racing skills," in *2006 IEEE International Conference on Evolutionary Computation*, July 2006, pp. 1187–1194.

[102] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, June 2009, pp. 248–255.

[103] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *2015 IEEE International Conference on Computer Vision (ICCV)*, Dec 2015, pp. 3730–3738.

[104] T. Zhou, P. Krähenbühl, M. Aubry, Q. Huang, and A. A. Efros, "Learning dense correspondence via 3d-guided cycle consistency," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 117–126.

[105] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2017, pp. 23–30.

[106] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, pp. 1137–1149, June 2017.

[107] M. Sundermeyer, Z.-C. Marton, M. Durner, M. Brucker, and R. Triebel, "Implicit 3d orientation learning for 6D object detection from RGB images," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 699–715.

[108] W. Kehl, F. Manhardt, F. Tombari, S. Ilic, and N. Navab, "SSD-6D: Making RGB-based 3D detection and 6D pose estimation great again," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1521–1529.

[109] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham: Springer International Publishing, 2014, pp. 740–755.

[110] X. Liu, C. R. Qi, and L. J. Guibas, "FlowNet3D: Learning scene flow in 3D point clouds," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[111] N. Mayer, E. Ilg, P. Hausser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox, "A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4040–4048.

[112] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013. [Online]. Available: https://doi.org/10.1177/0278364913491297

[113] Y. Sun, Z. Liu, Y. Wang, and S. E. Sarma, "Im2Avatar: Colorful 3d reconstruction from a single image," *CoRR*, vol. abs/1804.06375, 2018. [Online]. Available: http://arxiv.org/abs/1804.06375

[114] A. X. Chang, T. A. Funkhouser, L. J. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu, "ShapeNet: An information-rich 3d model repository," *CoRR*, vol. abs/1512.03012, 2015. [Online]. Available: http://arxiv.org/abs/1512.03012

[115] Y. Xiang, R. Mottaghi, and S. Savarese, "Beyond PASCAL: A benchmark for 3d object detection in the wild," in *IEEE Winter Conference on Applications of Computer Vision*, March 2014, pp. 75–82.

[116] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 2672–2680. [Online]. Available: http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf

[117] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb, "Learning from simulated and unsupervised images through adversarial training," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017, pp. 2242–2251.

[118] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan, "Unsupervised pixel-level domain adaptation with generative adversarial networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017, pp. 95–104.

[119] K. Bousmalis, A. Irpan, P. Wohlhart, Y. Bai, M. Kelcey, M. Kalakrishnan, L. Downs, J. Ibarz, P. Pastor, K. Konolige, S. Levine, and V. Vanhoucke, "Using simulation and domain adaptation to improve efficiency of deep robotic grasping," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, pp. 4243–4250.

[120] F. Rosique, P. Navarro Lorente, C. Fernandez, and A. Padilla, "A Systematic Review of Perception System and Simulators for Autonomous Vehicles Research," *Sensors*, vol. 19, p. 648, feb 2019.

[121] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[122] F. Chollet *et al.*, "Keras," https://github.com/fchollet/keras, 2015.

[123] L. von Ahn, "Games with a Purpose," *Computer*, vol. 39, no. 6, pp. 92–94, jun 2006. [Online]. Available: https://doi.org/10.1109/MC.2006.196

[124] B. Walther-Franks, J. Smeddinck, P. Szmidt, A. Haidu, M. Beetz, and R. Malaka, "Robots, Pancakes, and Computer Games: Designing Serious Games for Robot Imitation Learning," in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, ser. CHI '15. New York, NY, USA: Association for Computing Machinery, 2015, pp. 3623–3632. [Online]. Available: https://doi.org/10.1145/2702123.2702552

[125] PyGame Developers, "PyGame." [Online]. Available: http://pygame.org/

[126] Robotis, "PLATFORM - TurtleBot 3." [Online]. Available: http://www.robotis.us/turtlebot-3/

[127] DonkeyCar, "Donkey® Car - Home." [Online]. Available: https://www.donkeycar.com/

[128] M. O'Kelly, V. Sukhil, H. Abbas, J. Harkins, C. Kao, Y. V. Pant, R. Mangharam, D. Agarwal, M. Behl, P. Burgio, and M. Bertogna, "F1/10: An open-source autonomous cyber-physical platform," 2019.

[129] B. Goldfain, P. Drews, C. You, M. Barulic, O. Velev, P. Tsiotras, and J. M. Rehg, "Autorally: An open platform for aggressive autonomous driving," *IEEE Control Systems Magazine*, vol. 39, no. 1, pp. 26–55, 2019.

[130] Emlid, "Navio2 | Emlid." [Online]. Available: https://emlid.com/navio/

[131] Y. Technology, "YD LIDAR X4." [Online]. Available: http://ydlidar.com/product/X4

[132] Ardupilot, "ArduPilot Open Source Autopilot." [Online]. Available: http://www.ardupilot.org/

[133] "mavros - ROS Wiki." [Online]. Available: http://wiki.ros.org/mavros

[134] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[135] J. Barnett, N. Gizinski, E. Mondragon-Parra, J. E. Siegel, D. Morris, T. Gates, E. Kassens-Noor, and P. Savolainen, "Automated vehicles sharing the road: Surveying detection and localization of pedalcyclists," *IEEE Transactions on Intelligent Vehicles*, pp. 1–1, 2020.

[136] P. Gupta, D. Coleman, and J. E. Siegel, "Towards safer self-driving through great pain (physically adversarial intelligent networks)," 2020.

[137] J. Siegel and D. Morris, *Robotics, Automation, and the Future of Sports*. Cham: Springer International Publishing, 2020, pp. 53–72. [Online]. Available: https://doi.org/10.1007/978-3-030-50801-2_4

[138] E. Wilhelm, J. Siegel, S. Mayer, L. Sadamori, S. Dsouza, C.-K. Chau, and S. Sarma, "Cloudthink: a scalable secure platform for mirroring transportation systems in the cloud," *Transport*, vol. 30, no. 3, pp. 320–329, 2015. [Online]. Available: https://doi.org/10.3846/16484142.2015.1079237

[139] G. Pappas, J. Siegel, and K. Politopoulos, "VirtualCar: Virtual Mirroring of IoT-Enabled Avacars in AR, VR and Desktop Applications," in *ICAT-EGVE 2018 - International Conference on Artificial Reality and Telexistence and Eurographics Symposium on Virtual Environments - Posters and Demos*, T. Huang, M. Otsuki, M. Servières, A. Dey, Y. Sugiura, D. Banakou, and D. Michael-Grigoriou, Eds. The Eurographics Association, 2018.

[140] J.-E. Deschaud, "Kitti-carla: a kitti-like dataset generated by carla simulator," 2021.

[141] J. Barnett, N. Gizinski, E. Mondragon-Parra, J. E. Siegel, D. Morris, T. Gates, E. Kassens-Noor, and P. Savolainen, "Automated vehicles sharing the road: Surveying detection and localization of pedalcyclists," *IEEE Transactions on Intelligent Vehicles*, pp. 1–1, 2020.

[142] M. Engelcke, D. Rao, D. Wang, C. Tong, and I. Posner, "Vote3deep: Fast object detection in 3d point clouds using efficient convolutional neural networks," 09 2016.

[143] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 580–587.

[144] R. B. Girshick, "Fast r-cnn," *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 1440–1448, 2015.

[145] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, 06 2015.

[146] J. Dai, Y. Li, K. He, and J. Sun, "R-fcn: Object detection via region-based fully convolutional networks," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, ser. NIPS'16. Red Hook, NY, USA: Curran Associates Inc., 2016, p. 379387.

[147] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788.

[148] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C.-Y. Fu, and A. Berg, "Ssd: Single shot multibox detector," in *ECCV*, 2016.

[149] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," *Neural Information Processing Systems*, vol. 25, 01 2012.

[150] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.

[151] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 9, pp. 1904–1916, 2015.

[152] J. Uijlings, K. Sande, T. Gevers, and A. Smeulders, "Selective search for object recognition," *International Journal of Computer Vision*, vol. 104, pp. 154–171, 09 2013.

[153] J. Pont-Tuset, P. Arbeláez, J. T. Barron, F. Marqués, and J. Malik, "Multiscale combinatorial grouping for image segmentation and object proposal generation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, pp. 128–140, 2017.

[154] M. Everingham, L. Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *Int. J. Comput. Vision*, vol. 88, no. 2, p. 303338, Jun. 2010. [Online]. Available: https://doi.org/10.1007/s11263-009-0275-4

[155] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. Berg, and L. Fei-Fei, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, 09 2014.

[156] "The basics of lidar - light detection and ranging - remote sensing | nsf neon | open data to understand our ecosystems," https://www.neonscience.org/resources/learning-hub/tutorials/lidar-basics, (Accessed on 03/28/2021).

[157] "Lidar vs point clouds: learn the basics of laser scanning, 3d surveys and reality capture," https://info.vercator.com/blog/lidar-vs-point-clouds, (Accessed on 03/28/2021).

[158] A. Dosovitskiy, G. Ros, F. Codevilla, A. López, and V. Koltun, "CARLA: An open urban driving simulator," *arXiv*, no. CoRL, pp. 1–16, 2017.

[159] Unity Technologies, "Windridge city," *Windridge City*, 2019. [Online]. Available: https://assetstore.unity.com/packages/3d/environments/roadways/windridge-city-132222

[160] I. Buyuksalih, S. Bayburt, G. Buyuksalih, A. P. Baskaraca, H. Karim, and A. A. Rahman, "3D MODELLING and VISUALIZATION BASED on the UNITY GAME ENGINE - ADVANTAGES and CHALLENGES," vol. 4, no. 4W4, 2017, pp. 161–166.

[161] A. Juliani, V.-P. Berges, E. Teng, A. Cohen, J. Harper, C. Elion, C. Goy, Y. Gao, H. Henry, M. Mattar, and D. Lange, "Unity: A General Platform for Intelligent Agents," 2018, pp. 1–28. [Online]. Available: http://arxiv.org/abs/1809.02627

[162] Velodyne, "HDL-64E," *HDL-64E: HIGH DEFINITION REAL-TIME 3D LiDAR*, 2018. [Online]. Available: https://www.goetting-agv.com/dateien/downloads/63-9194{_}Rev-G{_}HDL-64E{_}S3{_}SpecSheet{_}Web.pdf

[163] B. Yang, W. Luo, and R. Urtasun, "Pixor: Real-time 3d object detection from point clouds," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7652–7660.

[164] Y. Zhou and O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3d object detection," 06 2018, pp. 4490–4499.

[165] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 2, pp. 386–397, 2020.

[166] L. Huang, Y. Yang, Y. Deng, and Y. Yu, "Densebox: Unifying landmark localization with end to end object detection," *ArXiv*, vol. abs/1509.04874, 2015.

[167] X. Zhou, C. Yao, H. Wen, Y. Wang, S. Zhou, W. He, and J. Liang, "East: An efficient and accurate scene text detector," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2642–2651.

[168] E. Shelhamer, J. Long, and T. Darrell, "Fully convolutional networks for semantic segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 4, pp. 640–651, 2017.

[169] T.-Y. Lin, P. Goyal, R. B. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2999–3007, 2017.

[170] M. Kianpour, S. Kowalski, E. Zoto, C. Frantz, and H. Øverby, *Designing Serious Games for Cyber Ranges: A Socio-technical Approach*, jun 2019.

[171] J. Vykopal, R. Ošlejšek, P. Celeda, M. Vizváry, and D. Tovarák, *KYPO Cyber Range: Design and Use Cases*, jan 2017.

[172] Silensec, "Addressing the Cyber Security Skills Gap:A white paper," Tech. Rep. [Online]. Available: https://www.silensec.com/downloads-menu/whitepapers/item/29-addressing-the-cyber-security-skills-gap

[173] S. Pfrang, J. Kippe, D. Meier, and C. Haas, "Design and Architecture of an Industrial IT Security Lab BT - Testbeds and Research Infrastructures for the Development of Networks and Communities," S. Guo, G. Wei, Y. Xiang, X. Lin, and P. Lorenz, Eds. Cham: Springer International Publishing, 2017, pp. 114–123.

[174] V. E. Urias, W. M. S. Stout, B. V. Leeuwen, and H. Lin, "Cyber Range Infrastructure Limitations and Needs of Tomorrow: A Position Paper," in *2018 International Carnahan Conference on Security Technology (ICCST)*, 2018, pp. 1–5.

[175] G. W. William Newhouse, Stephanie Keith, Benjamin Scribner, "National Initiative for Cybersecurity Education (NICE) Cybersecurity Workforce Framework," Tech. Rep., 2017. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-181.pdf

[176] T. Debatty and W. Mees, "Building a Cyber Range for training CyberDefense Situation Awareness," in *2019 International Conference on Military Communications and Information Systems (ICMCIS)*, 2019, pp. 1–6.

[177] A. Tirachini and C. Antoniou, "The economics of automated public transport: Effects on operator cost, travel time, fare and subsidy," *Economics of Transportation*, vol. 21, no. February, 2020.

[178] J. Piao, M. McDonald, N. Hounsell, M. Graindorge, T. Graindorge, and N. Malhene, "Public Views towards Implementation of Automated Vehicles in Urban Areas," vol. 14, no. 0. Elsevier B.V., 2016, pp. 2168–2177. [Online]. Available: http://dx.doi.org/10.1016/j.trpro.2016.05.232

[179] S. Deb, L. J. Strawderman, and D. W. Carruth, "Investigating pedestrian suggestions for external features on fully autonomous vehicles: A virtual reality experiment," vol. 59. Elsevier Ltd, 2018, pp. 135–149. [Online]. Available: https://doi.org/10.1016/j.trf.2018.08.016

[180] A. Habibovic, V. M. Lundgren, J. Andersson, M. Klingegård, T. Lagström, A. Sirkka, J. Fagerlönn, C. Edgren, R. Fredriksson, S. Krupenia, D. Saluäär, and P. Larsson, "Communicating intent of automated vehicles to pedestrians," vol. 9, no. AUG, 2018.

[181] E. Kassens-Noor, Z. Kotval-Karamchandani, and M. Cai, "Willingness to ride and perceptions of autonomous public transit," *Transportation Research Part A: Policy and Practice*, vol. 138, no. December 2019, pp. 92–104, 2020. [Online]. Available: https://doi.org/10.1016/j.tra.2020.05.010

[182] S. Deb, D. W. Carruth, R. Sween, L. Strawderman, and T. M. Garrison, "Efficacy of virtual reality in pedestrian safety research," vol. 65, no. October. Elsevier Ltd, 2017, pp. 449–460. [Online]. Available: http://dx.doi.org/10.1016/j.apergo.2017.03.007

[183] S. Shahrdar, C. Park, and M. Nojoumian, "Human trust measurement using an immersive virtual reality autonomous vehicle simulator," 2019, pp. 515–520.

[184] C. Ackermann, M. Beggiato, S. Schubert, and J. F. Krems, "An experimental study to investigate design and assessment criteria: What is important for communication between pedestrians and automated vehicles?" vol. 75, no. March 2018. Elsevier, 2019, pp. 272–282. [Online]. Available: https://doi.org/10.1016/j.apergo.2018.11.002

[185] G. Keeling, K. Evans, S. M. Thornton, G. Mecacci, and F. Santoni de Sio, "Four Perspectives on What Matters for the Ethics of Automated Vehicles," no. June, 2019, pp. 49–60.

[186] S. Reig, S. Norman, C. G. Morales, S. Das, A. Steinfeld, and J. Forlizzi, "A field study of pedestrians and autonomous vehicles," 2018, pp. 198–209.

[187] N. Li, I. Kolmanovsky, A. Girard, and Y. Yildiz, "Game Theoretic Modeling of Vehicle Interactions at Unsignalized Intersections and Application to Autonomous Vehicle Control," vol. 2018-June. AACC, 2018, pp. 3215–3220.

[188] B. Chen, D. Zhao, and H. Peng, "Evaluation of automated vehicles encountering pedestrians at unsignalized crossings," 2017, pp. 1679–1685.

[189] J. F. Fisac, E. Bronstein, E. Stefansson, D. Sadigh, S. S. Sastry, and A. D. Dragan, "Hierarchical game-theoretic planning for autonomous vehicles," vol. 2019-May, 2019, pp. 9590–9596.

[190] S. Jayaraman, D. Tilbury, A. Pradhan, and L. . J. Robert, "Analysis and Prediction of Pedestrian Crosswalk Behavior during Automated Vehicle Interactions," 2020.

[191] V. T. Nguyen and T. Dang, "Setting up Virtual Reality and Augmented Reality Learning Environment in Unity," no. November 2018, 2017, pp. 315–320.

[192] B. Morschheuser, L. Hassan, K. Werder, and J. Hamari, "How to design gamification? A method for engineering gamified software," vol. 95, 2018, pp. 219–237.

[193] M. Fernandez, "Augmented-Virtual Reality: How to improve education systems," vol. 7, no. 1, 2017, p. 1.

[194] A. Riegler, A. Riener, and C. Holzmann, "A Research Agenda for Mixed Reality in Automated Vehicles," *ACM International Conference Proceeding Series*, no. November, pp. 119–131, 2020.

[195] J. P. Gee, "Video games and embodiment," *Games and Culture*, vol. 3, no. 3-4, pp. 253–263, 2008.

[196] M. Moghimi, R. Stone, P. Rotshtein, and N. Cooke, "The Sense of embodiment in Virtual Reality," *Presence: Teleoperators & Virtual Environments*, vol. 25, no. 2, pp. 81–107, 2016. [Online]. Available: http://www.mitpressjournals.org/doi/pdf/10.1162/PRES{_}a{_}00135

[197] C. Alton, "Experience , 60 Frames Per Second : Virtual Embodiment and the Player / Avatar Relationship in Digital Games," *Loading: A Journal of the Canadian Game Studies Association*, vol. 10, no. 16, pp. 214–227, 2017. [Online]. Available: http://journals.sfu.ca/loading/index.php/loading/article/view/178

[198] D. Kiela, L. Bulat, A. L. Vero, and S. Clark, "Virtual Embodiment: A Scalable Long-Term Strategy for Artificial Intelligence Research," no. Nips, 2016. [Online]. Available: http://arxiv.org/abs/1610.07432

[199] M. Gonzalez-Franco and T. C. Peck, "Avatar embodiment. Towards a standardized questionnaire," *Frontiers Robotics AI*, vol. 5, no. JUN, pp. 1–9, 2018.

[200] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "AirSim: High-Fidelity Visual and physical simulation for autonomous vehicles," *arXiv*, pp. 1–14, 2017.

[201] R. Bartle, "Hearts, clubs, diamonds, spades: Players who suit MUDs," jun 1996.

[202] A. V. d. S. Marvin Oliver Schneider, Érika Tiemi Uehara Moriya, João Carlos Néto, "Analysis of Player Profiles in Electronic Games applying Bartle's Taxonomy," in *SBC - Proceedings of SBGames 2016*, 2016.

[203] J. Brooke, "Sus: a quick and dirtyusability," *Usability evaluation in industry*, vol. 189, 1996.

[204] R. McGloin, K. Farrar, and M. Krcmar, "Video games, immersion, and cognitive aggression: does the controller matter?" *Media psychology*, vol. 16, no. 1, pp. 65–87, 2013.

[205] Y. B. Kafai, G. T. Richard, and B. M. Tynes, *Diversifying Barbie and Mortal Kombat: Intersectional Perspectives and Inclusive Designs in Gaming*, 2016, no. August. [Online]. Available: http://press.etc.cmu.edu/files/Diversifying-Barbie-Mortal-Kombat{_}Kafai-Richard-Tynes-etal-web.pdf