applied sciences

MDPI

Article

Using Game Engines for Visuo-Haptic Learning Simulations

David Escobar-Castillejos [1,2,*], Julieta Noguez [2], Roberto A. Cárdenas-Ovando [2], Luis Neri [2], Andres Gonzalez-Nucamendi [2] and Víctor Robledo-Rella [2]

1 Department of Surgery & Cancer, Faculty of Medicine, Imperial College London, London SW7 2AZ, UK
2 Tecnologico de Monterrey, School of Engineering and Science, Ave. Eugenio Garza Sada 2501, 64849 Monterrey, N.L., México; jnoguez@tec.mx (J.N.); a01122353@itesm.mx (R.A.C.-O.); neri@tec.mx (L.N.); anucamen@tec.mx (A.G.-N.); vrobledo@tec.mx (V.R.-R.)
* Correspondence: d.castillejos@imperial.ac.uk or a01170737@itesm.mx

Received: 22 May 2020; Accepted: 22 June 2020; Published: 30 June 2020

Abstract: Technological advances have been the main driver of enhancing human–computer interaction and interactive simulations have experienced exponential growth in recent years. However, visual and auditory channels are usually the only ones considered for educational simulations even though the sense of touch is also an important one. Touch allows us to recognize and interact with our surroundings. A common way to develop a visuo-haptic simulation in the area of interactive systems is by using a graphic and physics-based engine orchestrated with a haptic rendering framework. However, new solutions, such as professional game engines, have enabled the development of high-quality applications in much shorter time. In this paper, a novel architecture for fast development of interactive visuo-haptic applications in game engines is discussed. To validate the proposed architecture, the Haptic Device Integration for Unity (HaDIU) plugin was implemented. Simulations were implemented to verify the operability of haptic devices. Each scenario was properly modelled and has different haptic objectives. Furthermore, to validate that the usage of this approach provides better visualizations than an existing single purpose application, an experimental study was performed. Results suggest that by using this approach faster development of interactive visuo-haptic simulators can be achieved than using traditional techniques.

Keywords: haptic devices; game engines; haptic rendering and modeling; simulation; e-learning; physics and engineering; higher education

1. Introduction

Significant changes in education have occurred through the years thanks to the evolution of technology. New technological advances focused on education have emerged, such as E-learning and technology enhanced learning, and simulators have been one of the most practical solution to enhance learning [1]. Moreover, visual and auditory channels usually are mostly considered in their development. Tactile sensations have been considered in these simulators by the use of input devices such as mouse devices, keyboards and touch screens. These are considered to be primary input sources. However, recent advances in the development of input devices also consider haptic feedback [2–5] (Figure 1). They enable users to experience the sense of touch and collaborate with sight to interact with virtual environments. Users can feel, palpate, and experience feedback forces in virtual environments. In addition, some haptic devices provide a low-cost solutions that can enhance realism of the virtual simulations [6]. Haptic devices have been used in many fields such as navigation, e-commerce, game, and even arts [3]. The areas with the most applications are medical training and education in general [4,5,7,8]. There have been studies to evaluate user acceptance of virtual simulators [9]

suggesting that virtual environments help users to experience scenarios in multi-sensory approaches. Consequently, simulators that incorporate haptic devices pose an important educational alternative to gain knowledge and skills. Moreover, learning and training are usually associated as a trial-error process and these provide interactive security and comfort conditions to let users practice without any danger [10]. Nevertheless, simulators that pair them have to provide proper interaction between the haptic device and the virtual objects since people use visuo-motor reflexes to acquire knowledge.

(**a**) (**b**)

Figure 1. Commercial haptic devices: (**a**) Novint Falcon haptic device and (**b**) Geomagic Touch, (previously SensAble Phantom Omni).

Visuo-haptic interactive simulators are often built in C++ by using OpenGL and/or using physics engines, such as NVIDIA PhysX (NVIDIA Corporation, California, U.S.). In order to add haptic connectivity, virtual simulators use a haptic rendering framework to control the haptic device. While the haptic interaction is commonly provided by the underlying haptics framework, new tools to develop 3D interactive applications have appeared. Among them, game engines are frameworks enabling developers to create high-quality applications quickly and easily. Game engines are environments designed to develop video-games. They provide built-in components, such as a rendering engine and a physics engine for collision detection and response, that facilitate the development of 2D or 3D applications. Nevertheless, game engines do not consider haptic rendering features in their functionality, so that must be outsourced.

We introduce a novel architecture that facilitates the development of visuo-haptic interactive simulations. It combines the use of game engines and haptic devices by different manufacturers. Unity was chosen as the testing game engine for this study (Unity Technologies, California, U.S.). A key challenge, and the contribution of our work, is the way we incorporate several and different brand haptic devices into Unity by using third-party solutions that are dependent on the CPU's architecture or do not consider multi-device operability. Our approach was planned to allow any interactive application developed in game engines to incorporate multi-device connectivity. These functionalities were implemented in two different modules: (1) Haptic Connectivity (HC) and (2) Haptic Manager (HM). The proposed architecture is capable of incorporating multiple devices in the same application, either the same or different brand, without the need of performing additional adjustments.

To validate the proposed architecture, the Haptic Device Integration for Unity (HaDIU) plugin was created. HaDIU is a C# and C++ plugin and it enables interconnectivity and functionality of multiple devices, as well as the generation of haptic rendering in game engines. Unity is used to perform some of the physics calculations and graphic visualization. Meanwhile, HaDIU is in charge of connecting haptic devices, sending their data to Unity, calculating interaction forces, and deploying force feedback on them. The proposed approach was tested using Geomagic Touch (previously SensAble Phantom Omni) and Novint Falcon haptic devices. However, Force Dimension haptic devices were also considered during development, such as Omega and Delta ones. This feature was considered to enable researchers acquire haptic devices according to their needs. Finally, using our approach, several simulations have been developed to test HaDIU, and a physics virtual simulator called Unity Forces was created. This simulator describes blocks on a rough surface [7,11] and it let the user feel the friction force between the blocks and the surface and the weight of each virtual block. Therefore, the main hypotheses to be proved in this work are as follow:

- The architecture facilitates the development of multi-device visuo-haptic interactive applications by using game engines.
- Visuo-haptic virtual environments can be developed faster and they provide better visualizations than current visuo-haptic environments which use traditional simulation techniques.

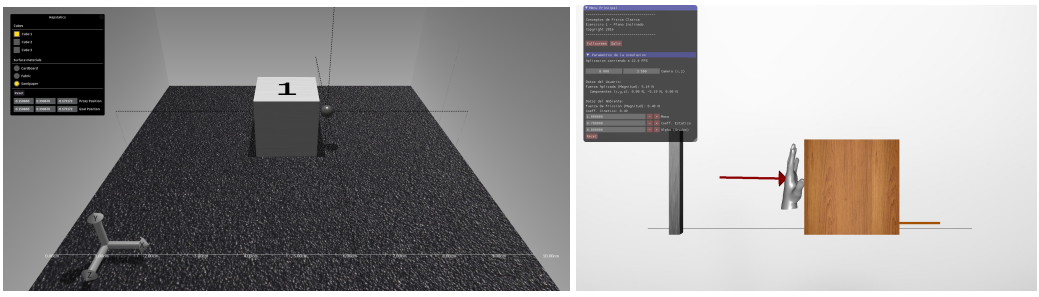
2. Related Work

2.1. Visuo-haptic Learning Simulations

Han and Black developed a computer-based gear simulation [12] by using Microsoft DirectX (Microsoft Corporation, Washington, U.S.) and Adobe Flash (Adobe Inc., California, U.S.). The simulation interface was designed with moving images in a Flash animation. The authors used the Microsoft Sidewinder Force Feedback 2 Joystick (Microsoft Corporation, Washington, U.S.) as haptic device. In this simulator, four gear combinations are shown to illustrate how two intermeshed gears create mechanical advantages. Through the visual channel, information was delivered about how fast each gear rotates, how much input force was needed to rotate the gear on the left and how much output force was generated by the gear on the right. The haptic device gave participants the actual feeling of the input force that they should use to rotate the gears.

Hamza-Lup and Baird created a visuo-haptic friction scenario [13] where the users perceived interaction forces originated during friction. The authors employed the H3DAPI for haptic rendering and used Python. The visual component of the simulator consists of an inclined plane, a set of floating menus for the configuration of the experiment, and the haptic interaction point (HIP). For force feedback, they used a Novint Falcon haptic device because of its compatibility with the H3DAPI, its haptic resolution characteristics, and its costs. Users can interact with the virtual block and plane, and feel the resulting forces.

Yuksel et al. developed *HapStatics*, another friction scenario in [11]. The interactive visuo-haptic simulation was implemented in C++ using OpenGL, and GLSL (Figure 2a). The force feedback was generated by the Novint Falcon device. For haptic rendering, they used Chai3D. They used a 3D cursor to indicate the position of the haptic cursor as well as shadows. Additionally, they limited the position of the haptic device by mounting virtual walls in the simulation environment. Finally, Yuksel et al. took advantage of the additional affordances of computer simulations to provide visual feedback, such as arrows indicating forces, color-coding, etc.



(**a**) (**b**)

Figure 2. (**a**) visuo-haptic friction simulationcreated by Yuksel et al. [11] and (**b**) Friction scenario created by Neri et al. [14].

Neri et al. designed and implemented four classical physics scenarios using haptic devices [14]. One of them was also a friction scenario (Figure 2b). The authors used a Novint Falcon haptic device to provide force feedback in the simulations, and the simulators were created using Chai3D. Parameters in each simulation can be controlled using a graphical user interface (GUI). Interaction forces were calculated according to the physics of each scenario, and real-time visualization was achieved. Another work by Neri et al. described the use of haptic devices to improve the learning

process of basic physics concepts from electromagnetism [7]. Their implementation allows the use of a SensAble Phantom Omni or Novint Falcon. They designed three scenarios: point charge, line charge, and plane charge. The point charge and the line charge simulations were developed as 3D environments due to the nature of their interaction forces. In their simulations, the HIP was displayed as a small sphere and its interaction forces were computed to avoid penetration. Finally, the plane charge simulator is symmetrical in 3D and was developed as 2D representation showing the x and y-axes. Additionally, the authors implemented guiding lines connecting the haptic cursor with the projection to the axis of the Euclidean coordinate system. This approach was considered to provide guidance during the navigation of the HIP in the 3D environment.

Shaikh et al. created a visuo-haptic laboratory for improving electromagnetism concepts [8]. Novint Falcon haptic devices were used to provide force feedback with three degrees of freedom, and the Chai3D framework was chosen for graphic and haptic rendering. As the user moves the avatar in the 3D space, the user can experience force fields generated by each type of charge. Lastly, the haptic experience is coupled with a real-time visual animation of the test charge being manipulated, the electric charge distribution and its resulting electric field (field lines).

Host et al. developed a simulator to understand electric fields around molecules [15]. The system uses a semi-immersive mirrored display, where 3D stereo-rendered graphics are reflected. As a haptic device, the simulator uses Phantom Desktop. The authors modeled an H_2O and a CO_2 molecule with the van der Waals surface. The HIP is used to explore the molecule's electrostatic potential at its position. Additionally, the authors added visual field lines for each scenario.

2.2. Unity and Haptic Devices

The use of haptic devices in Unity can be found in different studies. In Medicine, Gaudina et al. designed video laparoscopic training exercises and developed the eLaparo4D system [16]. The system eLaparo4D was constructed as a node.js application server, and it uses HTML5 with the Unity web engine plugin to give users an on-line platform. In the simulator, two Phantom Omni devices are used as tool handlers (grasper, hook or scissors) and a third one is used to move the camera within the virtual abdomen. To connect them, the authors used an in-house solution to add haptic interactivity in Unity.

Another medical simulator is the one developed by Pavaloiu et al. [17]. They created a virtual reality dental medicine training simulator. They used a Phantom Omni haptic to navigate and perform the tasks in the virtual scene. To enable haptic rendering, they used the Haptic Plug-In for Unity 5 [18], which is based on Geomagic's OpenHaptics software development kit (SDK) (Geomagic, Inc., Morrisville, NC). The simulator includes visual modifications and sound effects when medical procedures are performed in the oral cavity.

In the area of myringotomy, which is a surgical incision into the eardrum, Huang et al. implemented a VR-based environment to simulate all surgical aspects of myringotomy and tube placement [19]. The simulator uses a Phantom Omni haptic device for 3D positioning of surgical instruments, such as the virtual microscope, blade, and forceps. OpenHaptics SDK was also used for force rendering and positioning of the haptic device. Finally, the anatomical structures were modeled using the MIPAV (Medical Image Processing, Analysis, and Visualization) software package (Bethesda, MD) and Geomagic Studio (Geomagic, Inc., Morrisville, NC). Nguyen et al. created a knee bone drilling simulator to examine the use of visuo-haptic virtual simulations as affordable training solutions [20,21]. The authors implemented the simulation in Unity, and the simulator uses two haptic devices, a Novint Falcon and a Geomagic Touch. These are used to perform surgical drilling through bone in total knee arthroplasty. The authors used OpenHaptics SDK and the falconunity solution [22] to provide connectivity with the haptic devices of each brand. However, they do not mention if both devices can interact at the same time in the virtual environment.

Berney et al. developed a 3D immersive spatial cognition trainer with haptic interaction [23]. This simulator allows users to practice 3D manipulations through four levels of increasing difficulty.

The aim of this study was to measure if mental rotation and visualization of viewpoints tests improve spatial abilities. The simulator was built using Unity and two Geomagic Touch haptic devices. The interaction with the haptic device is implemented through a custom plug-in the authors developed.

Lastly, in the area of robotics, Andaluz et al. developed a simulator to control an emulated manipulator with a haptic device [24]. The simulator allows performance analysis of different schemes in autonomous control, tele-operated control, or both. They developed it using Unity for graphics and MATLAB (MathWorks, MA, USA) for the control algorithms and connectivity with the haptic device. The authors used a Falcon haptic device to control the simulated robot arm.

Virtual environments that incorporate haptic devices provide an important alternative for interactive training and learning environments. The above-presented simulators allow proper simulations; however, they usually are developed in long periods of time. Additionally, some studies only focus on functionality and leaves the improvement of visual realism as future work. On the other hand, current implementations only use game engines for visualization or only consider the use of one type of haptic device during simulation. Moreover, the falconunity solution requires older Falcon drivers. Moreover, it uses a server for connectivity in Unity if one does not have a professional edition. By using a server approach, applications needs to open an external software, which should not be needed to establish connections with haptic devices. Therefore, the aim of this study is to propose a multi-device visuo-haptic development architecture for game engines, and to validate it, the architecture was implemented in Unity.

3. Proposed Solution

Inclusion of haptic feedback in visuo-haptic applications requires the implementation of libraries to provide correct connectivity between application and haptic devices. Usually, manufacturers of haptic devices give SDKs to support haptics rendering in virtual environments. However, game engines does not have native functions that supports them, and third-party solutions are dependent on the CPU's architecture or do not consider multi-device operability. Additionally, the creation of visuo-haptic solutions usually involves long development times due to the high complexity that graphic and physics engines require to provide proper visualizations. Therefore, the use of game engines could ease production of visuo-haptic environments. In the following subsections, the proposed approach is described.

3.1. Proposed System Architecture

This work proposes a novel architecture to develop multi-device visuo-haptic solutions in game engines. This architecture considers key-features that they provide in simulations. In Figure 3, the high level architecture diagram is presented. Each of the components that make up this architecture are described in detail in the following paragraphs.

- **Haptic Connectivity (HC)**: This module is in charge of establishing connection between multiple haptic devices and the Haptic Manager (HM) component. Using the HC component, information of haptic devices, such as its maximum linear force, stiffness and damping, can be obtained. Moreover, haptic devices' positions, buttons' states, and orientation are read and sent to the HM component. By using this approach, all visual and physics interactions between the haptic device and virtual objects are delegated to the game engine, and the calculation of interaction forces and their control are generated within the HM component. Finally, this module is the one in charge of retrieving and deploying the interaction force calculated in the HM component.
- **Haptic Manager (HM)**: This module is in charge of retrieving haptic devices' information from the HC component and calculating interaction forces using the game engine's physics module. This component is in charge of initializing a thread to enable proper interactions of haptic devices, and it sends their information to the game engine. Using this data, the game engine is able to display the positions of the haptic device in the virtual scenario. Finally, interaction forces between the haptic device and virtual objects are computed using collision parameters from the

physics module of the game engine. After their calculation, the generated forces are controlled, according to the haptic devices' models that are being employed, and they are sent to the HC component to send force feedback to the user.

- **Game engine**: Most game engines have the following four sub-modules. In the following paragraphs each one is described:

  * **User Interaction**: game engines allow developers to create user interfaces (UIs). Their systems make it easy to display UIs that are positioned in the world among other 2D or 3D objects in the Scene. Moreover, modern applications often need to support a wide variety of different screen resolutions, and game engines' UI systems usually include a variety of tools for this purpose that can be combined in various ways. Finally, by using UIs, the mouse device, and keyboard, one can modify parameters in the simulation to modify the interactivity in the scene.

  * **Physics**: To provide convincing physical behavior, an object in an interactive application must accelerate correctly and be affected by collisions, gravity and other forces. Game engines' built-in physics modules provide components that handle the physical simulation for developers. Virtual objects can behave passively in a realistic way by modifying few parameter settings. Moreover, by controlling the physics using scripts, developers can give an object the behavior of a vehicle, a machine, or even a piece of cloth. Finally, when haptic devices are incorporated, it is necessary to send a force to the haptic; then, by using their collision detection algorithms, the HM component can calculate interaction forces and perform the necessary control algorithms to send the force feedback to the haptic device.

  * **Graphics**: game engines are equally suited to create 2D or 3D applications or both. Materials and textures are rendered on the surface of virtual objects to make them appear as realistic environments. The Camera can be used in perspective or orthographic mode, which can help developers generate views that enhance interactivity. Finally, light sources can be set-up easily around the world.

  * **Visual Rendering**: game engines perform the necessary processes to compile and create the virtual environment and display it in the video device.

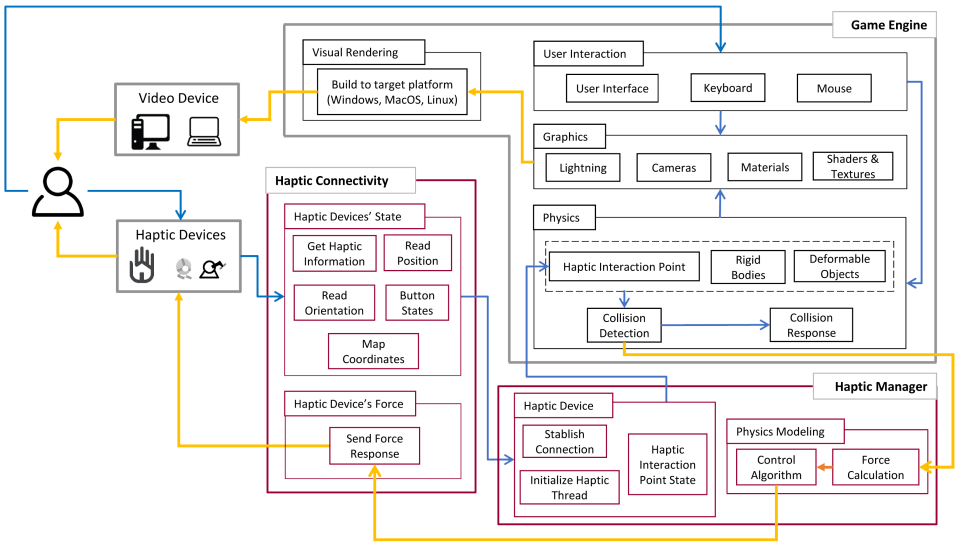


Figure 3. Proposed architecture for haptic devices integration in game engines.

3.2. *HaDIU: Haptic Device Integration for Unity*

As stated in the introduction, Unity was chosen as the testing game engine. It is a cross-platform game engine developed by Unity Technologies based on drag-and-drop functionality and uses C# or

JavaScript as its programming language. To validate the proposed architecture, the Haptic Device Integration for Unity (HaDIU) plugin was developed. It was created by the Cyber Learning and Data Science Laboratory that belongs to the Research Group of Product Innovation from the Tecnologico de Monterrey, Campus Mexico City. HaDIU enables the integration of haptic interaction techniques in Unity. HaDIU was developed using C++ and extending *Chai3d*'s haptic devices connectivity functions for the HC component, C# for the HM component. In the following paragraphs, the description of each HaDIU's components is described. Figure 4 describes the deployment diagram of the proposed architecture in Unity.

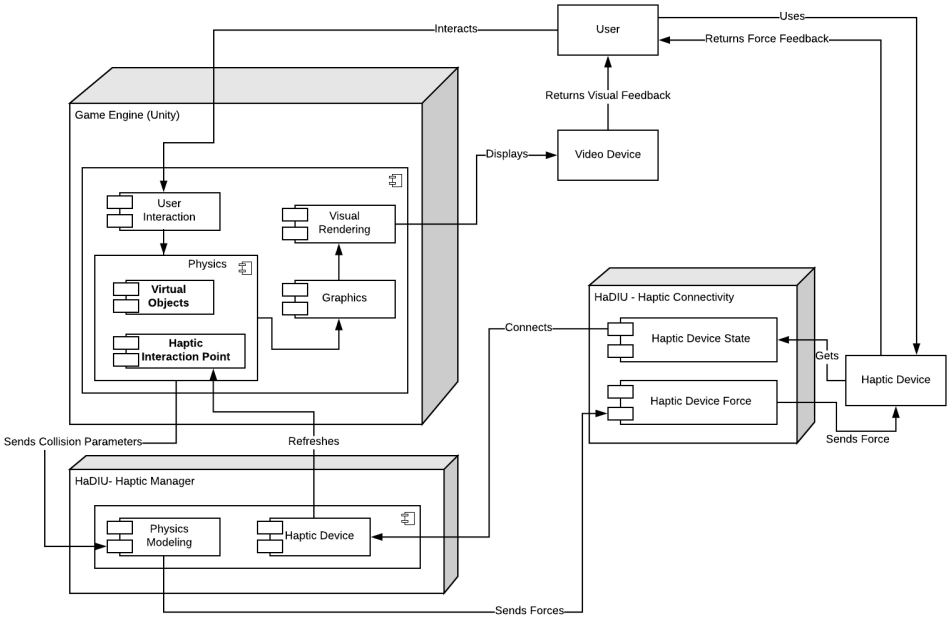


Figure 4. Deployment diagram of the proposed architecture in Unity.

3.2.1. HaDIU—Haptic Connectivity (HC) Component

The first functional part of HaDIU is its HC component. Unity has extensive support for native plugins written in C/C++, or other languages. They allow Unity to integrate middleware libraries or existing code. HaDIU uses and enhances *Chai3d*'s functions for haptic devices connectivity. To develop HaDIU, object-oriented programming (OOP) and a factory design pattern were used (Figure 5).

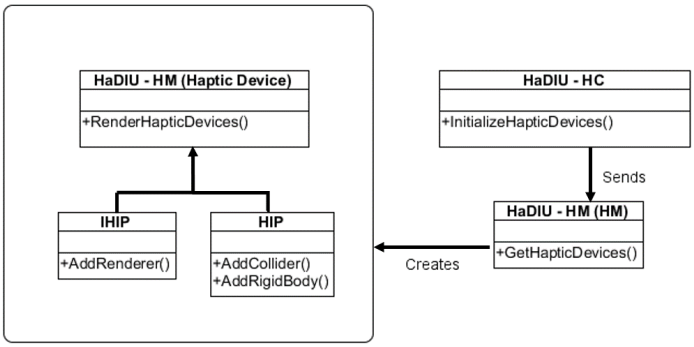


Figure 5. Factory design pattern used during the development of Haptic Device Integration for Unity (HaDIU).

The OOP approach was chosen to avoid duplication of code. Furthermore, OOP provides a clear modular structure for applications, which facilitates the implementation of the proposed architecture. On the other hand, the factory design pattern was chosen due to its ability to create objects without

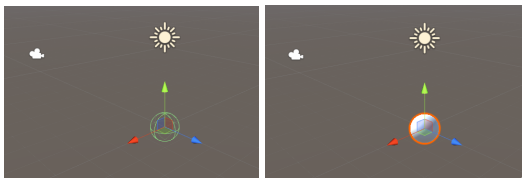
exposing the instantiation logic to the user and refers to the newly created object through a common interface. In other words, HaDIU is responsible for managing the hardware drivers of each haptic device, without the intervention of the user. This approach facilitates the development of interactive virtual environments with multiple haptic devices.

Due to the fact that the user does not have contact with the logic of the haptic device objects created in the factory, class methods were written to retrieve and send information to haptic devices, such as their positions, button states, and orientations. In addition, these methods fetch haptic devices' model information that can be used by developers to apply control algorithm techniques and send proper interaction forces to haptic devices. Finally, this module is in charge of sending the controlled interaction force generated in HaDIU's HM component to each haptic device.

During the development of this plugin, it has to be noted that two adjustments were needed to provide proper connectivity with haptic devices. The first one was a conversion of haptic devices coordinates to Unity ones. This was needed to work with coordinates consistent with Unity's own. Methods were deployed in HaDIU to perform this modification. First, a 3D vector structure was defined to store the new haptic positions. After that, a method was created in the HC component to map haptic devices coordinates to Unity ones. The same process was performed to map the orientation of haptic devices. The last adjustment was the implementation of wrapper functions in the HM component to specify linkage to HaDIU's methods through a dynamic-link library (DLL) file. This aspect was considered to enable the use of this native library in Unity.

3.2.2. HaDIU—Haptic Manager (HM) Component

The final functional part of HaDIU is its HM component. According to the deployment diagram, three Unity elements are needed to visualize and operate haptic devices. The first one is an HaDIU operability C# script. This script calls linkage functions in the native library and enable the haptic devices' connectivity. To properly use it, HaDIU's DLL needs to be imported to Unity. The second one is the HM object. This is an object that will only have a HM C# script component. This script will be in charge of connecting the HM component to the HC one to initialize the haptic devices' operability, to retrieve their information, and to send calculated interaction forces to each haptic device. In the first part, the HM C# script starts a thread to enable proper interactions of haptic devices. This thread is set to 1000 Hz, which is the necessary frequency to operate haptic devices. Then, it gets the haptic devices' information and sends it to the haptic devices' objects in the virtual scene. Finally, it retrieves interaction parameters from Unity's physics engine, and it calculates and calibrates the resulting interaction force that occurs during the interaction of the haptic device object with other virtual objects via the desired position and the haptic device position difference. Lastly, the third Unity element considered in this component is a haptic device virtual object. The haptic device object is composed of two spheres objects. One is the haptic interaction point (HIP), and the other one is the ideal haptic interaction point (IHIP). The HIP one only has a sphere collider and a rigid body component (Figure 6a), while the IHIP sphere has the visual renderer part of the haptic device object (Figure 6b). This approach was implemented according to the haptic rendering concepts described in [2].



(**a**) (**b**)

Figure 6. (**a**) Haptic interaction point object and (**b**) ideal haptic interaction point object.

Once the haptic device object is implemented, an HIP C# script component needs to be created. This gets the haptic device's position, orientation, and buttons' states from the HM C# script. Additionally,

it is the one in charge of detecting collisions in the environment using Unity's "on collisions" functions. It has to be noted that each haptic device needs its corresponding HIP and IHIP sphere objects in the scene. Finally, by using this approach, the sequence diagram presented in Figure 7 describes how all the proposed architecture's components collaborate during the communication with haptic devices, the interactivity of the haptic device and virtual objects, the calculation of interaction forces, and the transmission of visual and force feedback to the user. The Unity graphics represents the visual rendering or graphic processes, the Unity physics represents all the physics calculations. Unity is designed to run the physics thread while the graphical display is unfolding, and it handles the synchronization between the visual update rate and the frequency of the physics system, which can be set up according to the needs. Once Unity has started, the HaDIU script starts and the communication is made with the haptic devices, both the reading of the positions and the sending of the feedback force. Being a separate thread of execution, it allows haptic devices to maintain a refresh rate above 1 Khz. Moreover, this thread allows synchronization between Unity's physics thread and the haptic control thread.

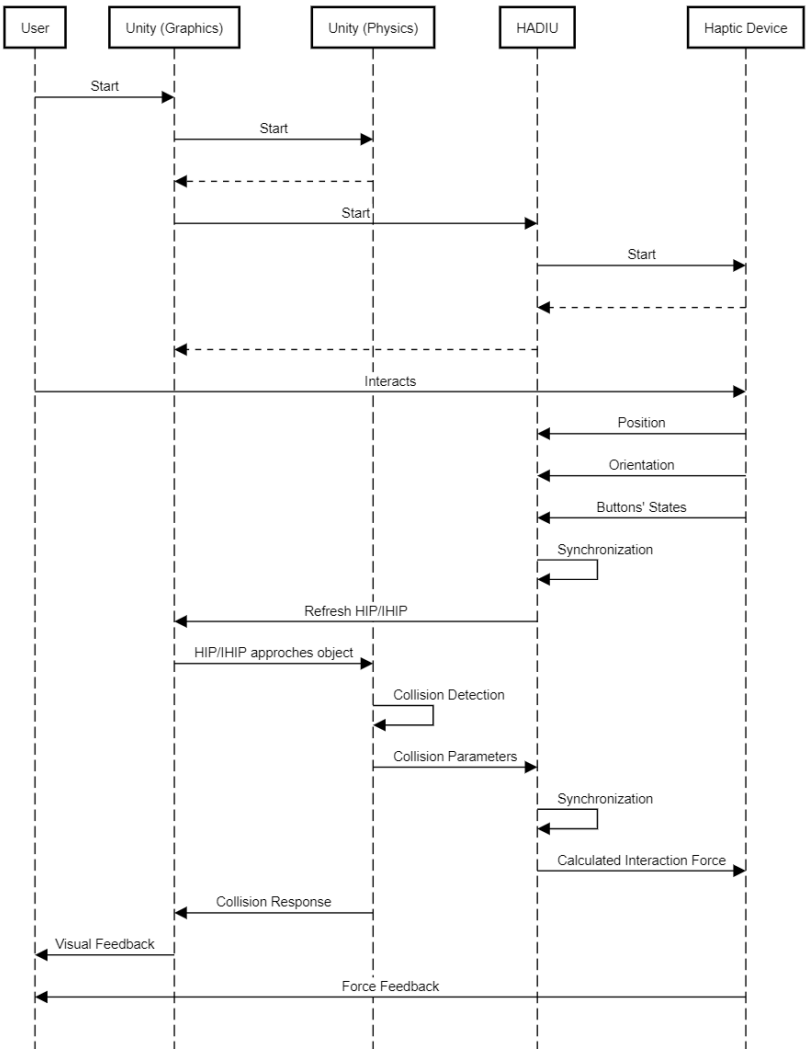


Figure 7. HaDIU's sequence diagram that describes how all the proposed architecture's components collaborate during the communication with haptic devices and calculation of feedback force according to their interactions with virtual objects.

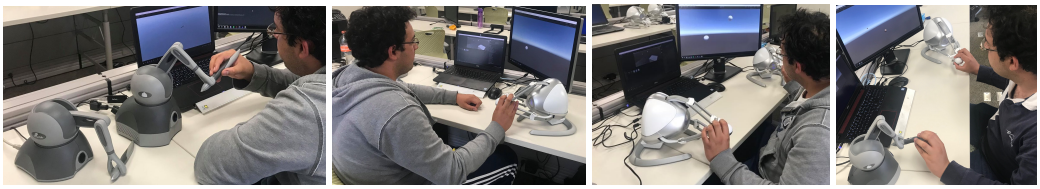
4. HaDIU Scenarios

To show the ease of creating visuo-haptic interactive simulations based on architecture, the following cases were implemented. They were created using Unity personal edition version 2018.1.1f1 These scenarios are described in the following subsections.

4.1. Haptic Connection and Environmental Effect

This simulation describes a simple scenario where the user moves the haptic device and its visual representation is shown in the virtual scene. In Figure 8a,b, a user is exploring the scene with a Geomagic Touch and a Novint Falcon haptic respectively. A test to check if multiple devices can operate at the same time was performed. The first test was the use of two Falcon haptic devices to navigate the environment (Figure 8c). The last test was the manipulation of one Falcon and one Touch haptic devices and performing the same task (Figure 8d). Finally, it has to be noted that the haptic devices' buttons were used to change some visualizations and enable a gravity effect. If the first button of the haptic devices' is pressed, the user will feel a simulated weight of a one kilogram. On the other hand, if the user presses the rest of the haptic devices' buttons, the texture of the IHIP will change.

The next scenario that was considered for testing was the addition of an environmental effect in the scene. In this scene, haptic devices are pulled to a desired point in the virtual scene. Additionally, a dampening factor can be added to reduce bounciness during the reposition process of the haptic device.

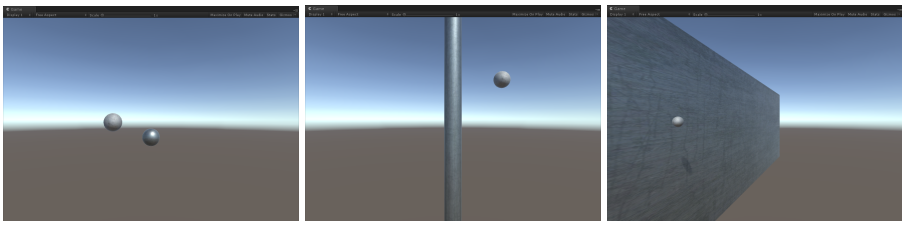


(**a**) (**b**) (**c**) (**d**)

Figure 8. User working on the haptic connection scenario. (**a**) User is using a Geomagic Touch, (**b**) User is using a Novint Falcon, (**c**) User is using two Novint Falcon, and (**d**) User is using a Novint Falcon and a Geomagic Touch.

4.2. Point Charge

This scenario enables the learner to feel the electric force between two point charges that has quadratic attenuation as the function of the distance of the charges (Figure 9a). Charge Q_1 is fixed at the origin of the coordinate system at location $[0, 0, 0]$, and charge Q_2 is controlled by the user, which is connected to the haptic cursor. The values of both charges can be chosen using Unity's editor. If the user chooses values that have different signs, he will feel the attraction force generated. On the other hand, if he chooses values that have equal signs, he will feel the repulsive force generated. The actual force is described by the Coulomb's law.



(**a**) (**b**) (**c**)

Figure 9. (**a**) Point charge, (**b**) line charge, and (**c**) plane charge scenarios.

4.3. Line Charge

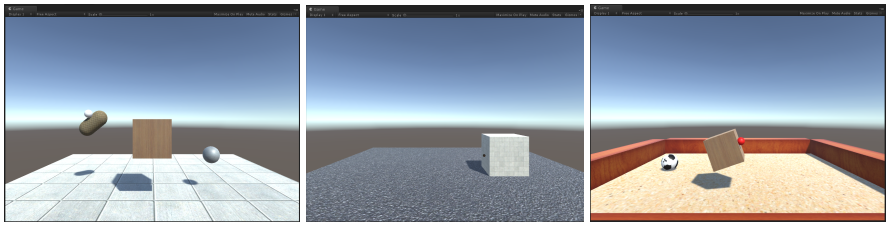
In this scene, shown in Figure 9b, the system displays a long straight-line charge fixed along the y-axis and a user-controlled mobile test point charge Q_2. The values for the linear charge density and λ of the line charge can be set and the user can select the point charge value. Both values can be set using Unity's editor. The forces that the user feels are calculated according to the sign of the values selected for the point charge and the straight-line charge.

4.4. Plane Charge

This simulation let users sense the force generated by the interaction of a large plane of charge fixed at the yz plane and a user-controlled mobile test point charge Q_2 (Figure 9c). The exerted force does not depend on the distance and is constant, because the plane is infinite. The value for the surface charge density σ, of the plane charge can be chosen by the learner. The point charge value can also be set as in previous scenarios.

4.5. Shape Exploration

Creating interaction forces that would arise when contacting stiff objects is considered a prototypical haptic task [25]. Therefore, after testing connectivity and force effects on the haptic device, a simple scene which allows touching different geometries and feeling an opposing force during contact was implemented (Figure 10a).



(**a**) (**b**) (**c**)

Figure 10. (**a**) Shape exploration, (**b**) friction forces, and (**c**) shape manipulation scenarios.

The HIP may penetrate any virtual object if the haptic devices' physical constraints are exceeded. However, the IHIP will display the contact point in the virtual scene. To display the opposing force experienced during contact with stiff objects, a force-rendering algorithm was implemented (Algorithm 1). In this algorithm, K represents the object's stiffness, Col_{pos} is the virtual object's contact collision position, and Hap_{pos} is the haptic device's position. In other words, this algorithm uses information on how far the HIP has penetrated the object to the compute interaction force.

Algorithm 1 Force feedback.

1: **if** HIP.isColliding **and** HIP.isInsideObject **then**
2: $F_{feedback} \leftarrow K * (Col_{pos} - Hap_{pos})$
3: **end if**

4.6. Friction Forces

In this simulation, two rigid bodies, a block object and a plane object are used (Figure 10b). Users can feel the opposing force during contact with virtual objects. The block's rotation was constrained and it only was allowed to move on the x-axis and z-axis during contact with the HIP. The physics related to the movement of objects is based on the friction force and the collision parameters, such as speed. Friction force is the opposing force coming into play when two bodies are in contact with each other. Therefore, the objects will move whenever the user exerts a force that is greater than

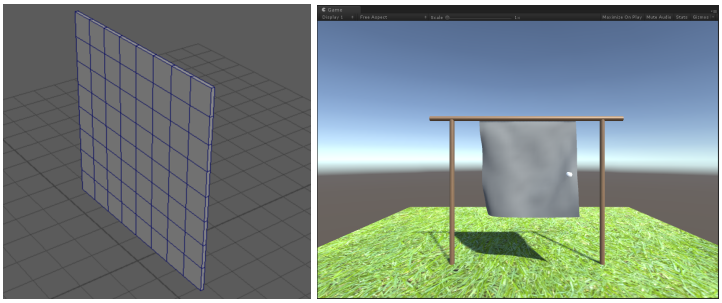
the static friction force. Once it is in motion, the object will continue to move on if the user applies a force greater than the dynamic friction force.

4.7. Shape Manipulation

An extension of the previous scenario, the shape manipulation was created. This scene allows touching different geometries and handling rendered virtual objects (Figure 10c). Users can feel the opposing force during contact with virtual objects. However, in this simulation, the rotation was not restrained to let users interact with the virtual objects freely. Additionally, if the user is grabbing a virtual object, he will feel the weight of the virtual object according to gravity constant (9.8 m/s^2). Moreover, if the virtual objects are held by the haptic device and released, the objects will experience a freefall animation until they collide with the floor. Finally, in this simulation, Unity is used to simulate some basic interactions between any object being manipulated and its surroundings.

4.8. Cloth Interaction

The last scenario was developed to implement deformable object interactions with haptic devices. Developers could use Unity's plane object; however, the backside of the plane is normally not rendered. Therefore, textures will not be seen from its backside. Consequently, a cloth model was created as a flat, rectangular cube made in Autodesk Maya (Autodesk, Inc., CA, USA). This cube was subdivided into 64 segments (Figure 11a).



(**a**) (**b**)

Figure 11. (**a**) Cloth model created using Maya software and (**b**) Cloth interaction scenario.

The challenge of this scenario was to recreate proper interactions between the cloth and the haptic device. The scenario was created as a sheet mounted on a clothing rack (Figure 11b). Then, a cloth component has to be added to the model. This component enables developers to constrain some parameters of its behavior. This feature was used to attach the cloth model to the upper pole. Finally, for collision detection between the haptic device and the cloth model, a mesh collider was added to the model and the attributes in the cloth component called "Capsule collider" and "Sphere collider" were used. The latter indicates the colliders that will affect the cloth instead of the colliders on the cloth. This attribute is used to perform collisions handling between the haptic devices and the cloth model.

Moreover, to enable haptic devices to grab the cloth model, a collision detection algorithm was programmed to detect interactions between the vertexes and the HIP. First, the HIP C# script calculates the nearest vertex to the haptic device object. This is performed by using the K-d-tree technique, which is a space partitioning algorithm. K-d-tree divides the scene into two subsets that do not overlap. Any point in the space can then be identified to lie in exactly one of the regions. The pseudo-code for the K-d-tree technique can be seen Algorithm 2.

Algorithm 2 K-d-tree algorithm.

```
if cloth.vertexSize is 1 then
    return nearest vertex.
else
    Split the scene into two subsets with a line through the median x-coordinate of the vertexes.
    if HIP.position is on the left side then
        Check left cloth model's vertex to find the nearest vertex according to HIP's position and radius.
        return nearest vertex.
    else
        Check right cloth model's vertex to find the nearest vertex according to HIP's position and radius.
        return nearest vertex.
    end if
end if
```

After finding the nearest vertex to the HIP, according to the Pythagorean Theorem, one can get the distance between the HIP and the vertexes. In 3D, this theorem can be imagined as a cuboid (Figure 12). First, the triangle ABC is used to obtain the distance between AC. Next, this value is used to solve triangle ACF to find distance AF. Finally, if the distance between the point and the center of the circle is less than the radius of the circle, they are colliding. Once the collision detection is obtained, the force feedback calculation from the shape manipulation scenario can be applied.

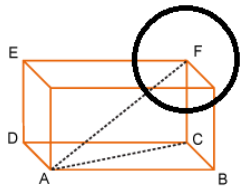


Figure 12. Pythagoras' theorem for 3D cases.

5. Experimental Study

To test the hypotheses raised in Section 1, another developer built the following additional scenarios based on our proposal.

5.1. Additional Scenarios

One additional developer was asked to test the environment. The new developer already knew how to program in C# and had some experience with Unity game engine. Thus, the learning curve was related only to the use of the haptic plugin. The functionalities were explained in a 1-hour talk and each scenario was introduced with a brief description of the underlying maths and the desired user interface with handwritten mock ups. To ensure that the development feasibility was related to the haptic plugin use and not to the understanding of the math equations or multiple dependencies, all the additional scenarios had to be stand-alone deployments related to high-school or freshman undergraduate physics courses. The selected environments to develop were: Hooke's Law, inclined plane, buoyancy, and torque.

5.1.1. Hooke's Law

Hooke's Law was tested with different scenarios with one or more springs. The easiest scenario had an unmovable wall, one handle and only one spring connected to both, as can be seen in Figure 13a. The user interface only allowed to change the spring constant, k. The haptic-device had to be able

to "touch" everything within the virtual environment and also be able to push and pull the spring by grabbing the handle. The objective was to reproduce the force generated by the spring with the selected parameter to understand the relationship between force and spring constant. Once this first environment was created two variations were introduced, where a new spring could be added to the mix, in parallel or in series, with the first one (Figure 13b). Moreover, each spring was independent of the other, thus each one could have a different constant, and the user should be able to touch both. The stand-alone deployment included the three variations (single spring, two springs in series, and two springs in parallel), and the end user could choose any of them within the same application.

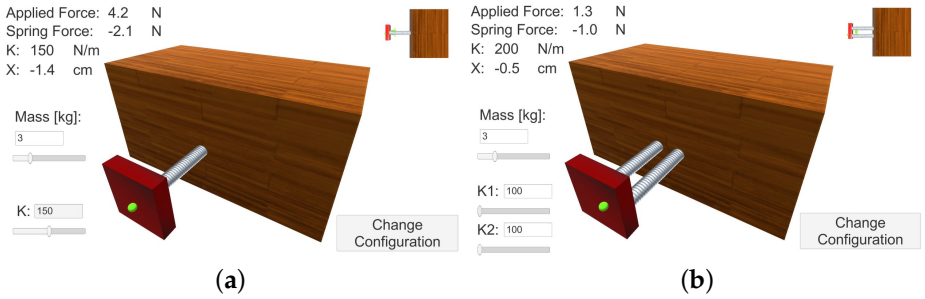


Figure 13. Hooke's Law scenarios: (**a**) simple spring and (**b**) double springs.

5.1.2. Inclined Plane

The inclined plane was tested with simple environment of two objects: a ramp and a movable object (Figure 14). To make an easier interaction the object was a crate with texture that allowed the user to grab it and move it; and the ramp had railings that limited the crate's movement in the depth axis. All objects including the railings were touchable objects. This environment represented a greater challenge than the Hooke's Law as it included a multiple number of parameters to customize while keeping the force computations in real time. The objective was to reproduce the force that the user would need to move the crate through the ramp while changing different parameters.

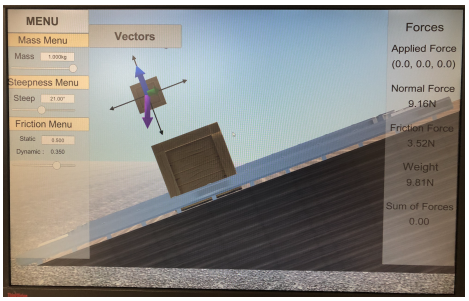


Figure 14. Inclined plane scenario.

5.1.3. Buoyancy

The objective was to simulate the movement and the force applied to an object (a cube) submerged in a liquid while the user pushes the object inside the liquid in a recipient (Figure 15). The physical parameters of the object that could be changed were its size, mass, and density, and for the liquid its density. The option to select some typical materials for the object and/or the liquid, instead of entering their corresponding densities, was also provided. Therefore, the interaction between liquid and object had a 2nd order differential equation behavior that had to be modeled not only for visual representation but for the haptic-device interaction too. Aside this, the main challenge was focused in the inherent vibrations from a 2nd order behavior plus the haptic perception in real time computation. The forces had to be checked in a real time basis to prevent the haptic device from overreacting to the forces generated that could hurt the user or damage the haptic device.

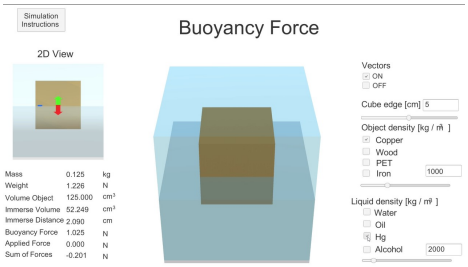


Figure 15. Buoyancy scenario.

5.1.4. Torque

The last scenario revolved about a different challenge than mathematical complexity, it involved the use of two haptic devices within the same environment applying forces to the same object at the same time. In this scenario the only object in the environment was a seesaw and its fulcrum (Figure 16). Each device could interact only with a section (right or left) of the seesaw and could be able to push the section downwards. The objective was to understand the equilibrium of torques in a rotatory object with more than one input force. The biggest challenge of this environment was the haptic devices synchronization due to the need of a minimum refresh rate to allow a real feeling when touching the object.

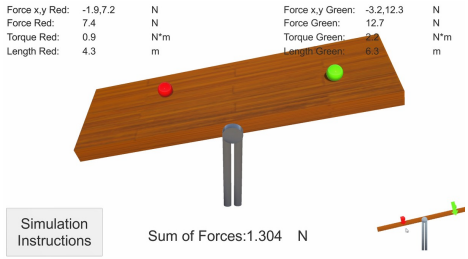


Figure 16. Equilibrium scenario.

5.2. Previous Simulators Developed Using Traditional Frameworks

To perform a comparison, visuo-haptic scenarios created using traditional frameworks (OpenGL, Chai3D, and C++) at the Cyber Learning and Data Science Laboratory of Tecnologico de Monterrey Mexico City Campus were taken into consideration [7,14]. Figure 17 shows examples of these applications. These simulators have been used to measure learning gains of students.

5.3. Unity Forces

The last stage of this study was to develop a haptic based friction simulator, Unity Forces. Taking as basis the scenarios described in Sections 4.6 and 4.7, and the simulator developed by Yuksel et al., the simulator uses three rigid bodies: three block objects and a plane object. Different textures were applied to each block, and in accordance with the needs of the experimental study, the blocks were constrained and it only was allowed to move on the x-axis during contact with the HIP. However, if the block is held by the haptic device and released, the y-axis is allowed to allow freefall animation and interaction, as in the shape manipulation scenario. Text elements are used to show the physical parameters of the simulation, as position and forces in the environment. These values are updated in real time; therefore, users can use the UI to locate themselves in the virtual environment and identify the force values that they are experience. Check-boxes are used to enable the visualization of each block and its mass. The drop-down is used to change the texture of the plane object and change its μ_s and μ_k parameters, according to the chosen texture. In accordance to the needs of the experimental study, sandpaper, cardboard, and fabric textures were used. Finally, the button element was implemented to reset the position of the virtual block to its original position.

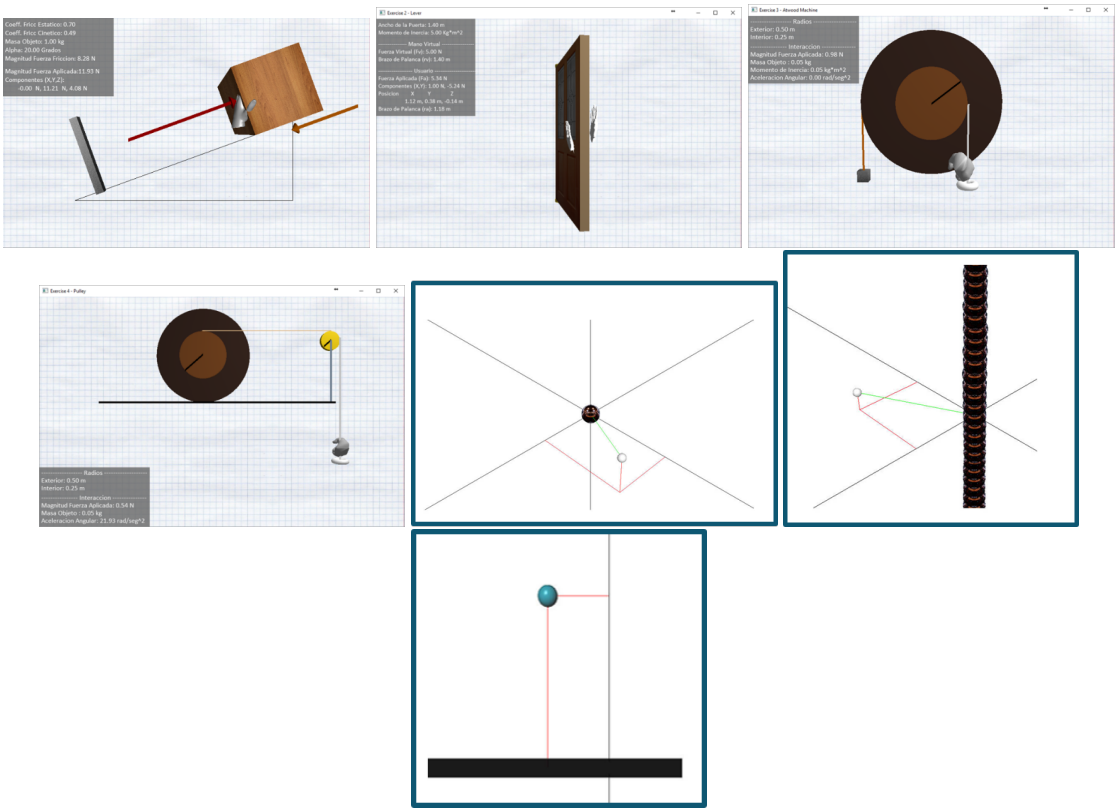


Figure 17. Previous developed simulators using traditional frameworks [7,14].

5.3.1. Users' Perception

During the January–May 2018 term, 16 undergraduate engineering students of Tecnológico de Monterrey, Mexico City Campus, who were enrolled in a first semester classic physics course, were selected. Participants had similar previous backgrounds in physics and academic conditions before starting the comparison study. First, all participants used a similar visuo-haptic training environment called HapStatics, developed by Yuksel et al. (Figure 2). They explored the simulator and performed two tests. After using HapStatics, participants were asked to test Unity Forces and perform the same tests.

In the first test, participants were asked to move the virtual blocks and feel the opposing force generated according to the friction force of each surface. They were not able to move the block until the static friction force was surpassed by the applied force of the user. Once the participants applied a force greater than the static friction force, the virtual block was able to move and the opposing force was the kinetic friction force (Figure 18).

In second test, participants had to grab the virtual block and feel its simulated weight. The user felt the weight of the virtual object (Figure 19a). In addition, they were asked to release the block in mid air and see how the virtual block gravity fell (Figure 19b).

Finally, all participants were asked to answer a perception questionnaire at the end of the experimental study. This questionnaire was focused on measuring students' experience while working with Unity Forces, taking as basis their previous experience during the use of HapStatics. Moreover, participants were asked to choose which scenario they liked the most based on visualization and forced perception. The survey was based on a yes/no format (questions Q1 to Q2 and Q4 to Q6); however, question Q3 is based on a five-step Likert scale. Lastly, in Q7, students were asked to give any free comments on their experience in working with Unity Forces. The applied survey can be seen in Table 1.

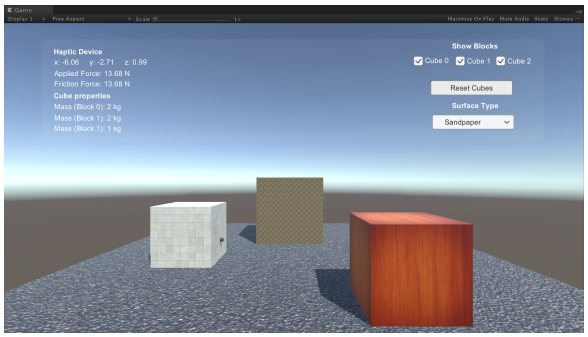
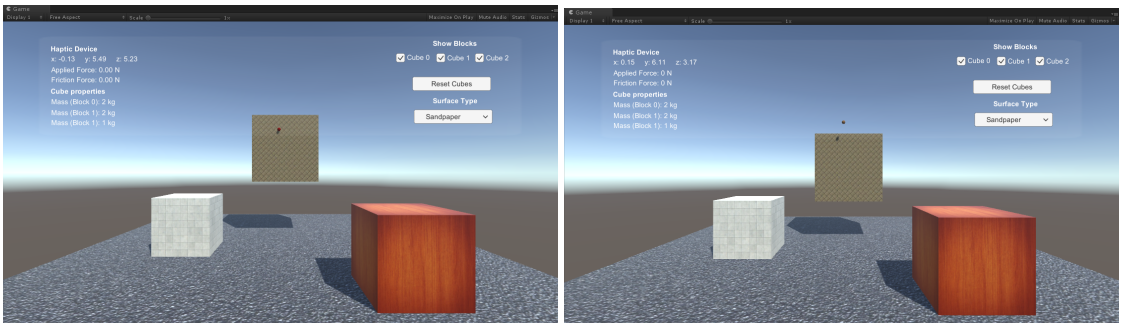


Figure 18. Unity Forces simulator: User is moving the second virtual block to the left.



(**a**) (**b**)

Figure 19. (**a**) User grabbing the first virtual block and experiencing its weight, and (**b**) User releases the first virtual block and it experiences freefall.

Table 1. Perception questionnaire applied to participants after testing Unity Forces.

Question	Statement
Q1	Do you like Unity Forces?
Q2	Do you like its graphic visualization?
Q3	In comparison with HapStatics, how was *Unity Forces*'s force feedback? (1 = worse to 5 = better)
Q4	Do you feel that both simulators display similar physics behavior?
Q5	Which of the simulators has better force feedback?
Q6	Which of the simulators has better graphic visualization?
Q7	What do you think about *Unity Forces* and what would you improve?

6. Results and Discussion

In this section, a summary and analysis of the results obtained in the study are provided.

6.1. Performance

Performance tests were applied to all the developed scenarios. The metrics to be evaluated were the average percentage of CPU consumption and GPU (VRAM) utilization in each simulation. Moreover, the steps considered to determine the development time necessary to create interactive visuo-haptic simulators were the following: the learning time it took to understand the simulation framework, the implementation of the simulation in terms of graphic visualizations, and the force feedback calibration. Tests were performed on a Dell Inspiron i75592512BLK, Intel Quad Core i7-6700HQ 2.6 GHz Processor, NVIDIA GeForce GTX 960M 4GB GDDR5, 16 GB DDR3L SDRAM, Windows 10 64-bit operating system.

Each scenario was tested using one to six haptic devices, excluding the equilibrium of forces due to its nature. Results can be seen in Tables 2 and 3. Results show that CPU and GPU utilization were linear according to the type of simulation and number of haptic devices used.

Table 2. Results obtained in the CPU utilization test using one to six haptic devices (HDs).

Scenario	One hd	Two hd	Three hd	Four hd	Five hd	Six hd
Haptic Connectivity	10%	11%	11%	12%	13%	14%
Environmental Effect	10%	11%	11%	12%	13%	14%
Point Charge	10%	11%	11%	12%	13%	14%
Line Charge	10%	11%	11%	12%	13%	14%
Plane Charge	10%	11%	11%	12%	13%	14%
Shape Exploration	10%	11%	12%	13%	14%	15%
Friction Forces	13%	13%	14%	15%	16%	16%
Shape Manipulation	14%	15%	15%	16%	16%	17%
Cloth Interaction	14%	15%	15%	16%	16%	17%
Hooke's Law	13%	13%	14%	15%	16%	16%
Inclined plane	13%	13%	14%	15%	16%	16%
Buoyancy	14%	15%	15%	16%	16%	17%
Unity Forces	13%	13%	14%	15%	16%	16%

Table 3. Results obtained in the GPU utilization test using one to six haptic devices (hd).

Scenario	One hd	Two hd	Three hd	Four hd	Five hd	Six hd
Haptic Connectivity	13%	20%	20%	20%	20%	20%
Environmental Effect	13%	20%	20%	20%	20%	20%
Point Charge	13%	20%	20%	20%	20%	20%
Line Charge	13%	20%	20%	20%	20%	20%
Plane Charge	13%	20%	20%	20%	20%	20%
Shape Exploration	21%	21%	22%	22%	22%	22%
Friction Forces	22%	22%	23%	23%	23%	23%
Shape Manipulation	23%	23%	24%	24%	24%	24%
Cloth Interaction	23%	23%	24%	24%	24%	24%
Hooke's Law	23%	23%	24%	24%	24%	24%
Inclined plane	22%	22%	23%	23%	23%	23%
Buoyancy	23%	23%	24%	24%	24%	24%
Unity Forces	22%	22%	23%	23%	23%	23%

Results state that HaDIU's performance was optimal and interactive applications created using the proposed architecture provide fluent visualizations. Additionally, CPU utilization proved that even though six haptic devices were connected in the environment, force feedback generation and graphic rendering were displayed properly, and no discrepancies or performance consumption were present during their connectivity. These results prove the fact that proper multiple haptic devices connectivity in game engines was achieved by using HaDIU.

6.2. Development Times

Using previous development times from simulators, described in Section 5.2, a comparison was made to evaluate the development times between typical frameworks and HaDIU. It has to be noted that all the scenarios described in Section 5.1 were set in this order to give the necessary tools to the developers to get used to the plugin and its requirements. The developers were able to finish all the scenarios within a week with no more than a few extra sessions for punctual doubts of the plugin usage. Table 4 shows the development times of the simulators. It has to be noted that the Haptic Connectivity scenario was not considered due to it does not involve interactions with virtual objects nor force feedback calculation. Results state that using HaDIU, developers could create visuo-haptic interactive applications faster than using traditional techniques.

Table 4. Development time comparison table for previous application developed by the authors and the scenarios developed using HaDIU.

Scenario	Framework	Development Times	Calibration	Total
Block on a rough incline	Chai3D + OpenGL	30 h	20 h	50 h
Rotating Door	Chai3D + OpenGL	38 h	29 h	67 h
Double Atwood's machine	Chai3D + OpenGL	43 h	28 h	71 h
Rolling reel	Chai3D + OpenGL	45 h	29 h	74 h
Point Charge	Chai3D + OpenGL	25 h	16 h	41 h
Line Charge	Chai3D + OpenGL	27 h	18 h	45 h
Plane Charge	Chai3D + OpenGL	29 h	19 h	48 h
Average		33.85 h	22.71 h	56.57 h
Environmental Effect	*HaDIU*	6 h	10 h	16 h
Point Charge	*HaDIU*	8 h	16 h	24 h
Line Charge	*HaDIU*	8 h	12 h	20 h
Plane Charge	*HaDIU*	8 h	12 h	20 h
Shape Exploration	*HaDIU*	4 h	8 h	12 h
Friction Forces	*HaDIU*	7 h	10 h	17 h
Shape Manipulation	*HaDIU*	8 h	12 h	20 h
Cloth Interaction	*HaDIU*	16 h	16 h	32 h
Hooke's Law	*HaDIU*	14 h	15 h	29 h
Inclined plane	*HaDIU*	10 h	14 h	24 h
Buoyancy	*HaDIU*	16 h	18 h	34 h
Torque	*HaDIU*	13 h	17 h	30 h
Unity Forces	*HaDIU*	12 h	14 h	26 h
Average		10 h	13.38 h	23.38 h

6.3. Perception Questionnaire

An experimental study to validate that the usage of this approach provides better visualizations was performed. Users who were enrolled in a first semester classic physics course were selected to test Unity Forces and the simulator HapStatics [11]. Students were first asked to apply a force to the block in order to move it on a rough flat surface and feel the strength of the corresponding static and friction forces. Then, they were asked to hold and move up and down the block to feel its weight. At the end of the testing phase, participants were asked to answer a perception questionnaire. Results state that force feedback and graphic visualization were better in Unity Forces. Consequently, the results of these tests indicate that using the proposed approach provides: (1) a faster development of interactive visuo-haptic simulators as compared to traditional techniques, and (2) a better graphic visualization of the physical scenario.

Results from questions Q1 to Q6 of the perception questionnaires are shown in Table 5. Results state that in general, most participants considered that they liked Unity Forces, its visualization of objects was adequate, its visual and tactile realism were appropriate, and that HapStatics and Unity Forces were similar in their deployment of physics. Nevertheless, Table 5 clearly shows that participants thought that Unity Forces was more realistic, intuitive and easier to handle than HapStatics. The steps of Q3's Likert scale are defined as: 1 = worse, 3 = equal, 5 = better, where 2 and 4 are the intermediate sensations between worse to equal and equal to better, respectively. Results from this question indicate that Unity Forces's force feedback was better than the one generated by HapStatics. This could be generated due to our approach's force feedback calculation, based on penetration depth. Current ways to generate force feedback in simulators are by using built-in functions from haptic rendering frameworks. The results from this question could state that by modeling properly the interaction forces that are present in the simulation, according to the established physics laws, could provide better results than using built-in haptic rendering algorithms. Finally, Q7 asked students to provide their opinion about Unity Forces and what they would improve. From the first part of the question, the comments summarized that they liked it better, they like the visualizations, and they found it easier to interact with the objects in the simulation. On the second part of Q7, most of the participants' comments focused on: improving space location in order to properly apply the force,

increasing the size of the virtual block to improve interactivity, and improving the camera angle to locate the place where the force was applied on the block.

Table 5. Results of the perception questionnaire applied to participants after testing Unity Forces (N = 16).

Question	Statement	Yes	No
Q1	Do you like Unity Forces?	94%	6%
Q2	Do you like its graphic visualization?	94%	6%
Q4	Do you feel that both simulators display similar physics behavior?	87%	13%

Question	Statement	1	2	3	4	5
Q3	In comparison with HapStatics, how was Unity Forces's force feedback?	0	2	6	2	6

Question	Statement	Unity Forces	HapStatics
Q5	Which of the simulators has better force feedback?	81%	19%
Q6	Which of the simulators has better graphic visualization?	81%	19%

6.4. Complementary Test

To test the visuo-haptic environments developed using Unity and HaDIU, a preliminary study was conducted with 51 undergraduate students enrolled in the Electricity and Magnetism and the Electromagnetic Fields courses. The participants used the point, line, and plane charge simulators described in Sections 4.2–4.4. Results are reported in [26]. In this study, the average grade of the Post-test was higher than that of the pre-test for both the Electricity and Magnetism and the Electromagnetic Fields course participants, and for the former, the average learning gain was statistically significant. This result is encouraging to include visuo-haptic simulators for other Physics topics. The perceptions of students on the use of the simulators was very positive. After this activity, students were more motivated to engage in class discussions and asked more interesting questions. They also talked very positively about the haptic activity to peers enrolled in other groups.

6.5. Hypothesis Test

For the hypotheses demonstration, the development times and the user study of Unity Forces were considered to evaluate the proposed framework.

6.5.1. Development Times

For this study, the two samples of Table 4 were considered. $M1$ will correspond to the scenarios developed using Chaid3D, and $M2$ includes all the simulators developed using HaDIU. Therefore, the hypotheses are:

h_0 : Development Times $M1 \leq$ Development Times $M2$

h_a : Development Times $M1 >$ Development Times $M2$

Due to the nature of the samples, non-normal distributed data, its is necessary to perform a non-parametric statistical test for few observations. A Mann–Whitney U (MWU) test was selected to perform the comparison. This study used the *wilcox.test* function from R's stats package with the parameter paired to be false to have the MWU test behavior. Results are shown in Table 6. It is observed that in $M2$ there was a significant shift to the left when compare to $M1$. Therefore, through the result obtained, there was statistical lower difference in the development times of $M2$.

Table 6. Mann–Whitney U non-parametric statistical test applied to the development times sample.

Wilcoxon Rank Sum Test With Continuity Correction
data: DTM1 and DTM2
W = 91 p-value = 0.0001767
alternative hypothesis: true location shift is greater than 0

6.5.2. User Study: Unity Forces

Due to the nature of the perception questionnaires questions, it was necessary to perform a binomial test to Q1, Q2, Q4, Q5, and Q6 and a *t*-test to Q3. The binomial test is used when an experiment has two possible outcomes. On the other hand, a Student's *t*-test was used due to the samples in this study were less than 25. For the binomial test, the hypotheses were:

$$h_0 : p = 0.5$$
$$h_a : p > 0.5$$

In Q1, h_0 means that there was no clear gain using Unity Forces. Consequently, when h_0 is rejected, we assume that h_a is valid. This means that there was statistical evidence of a gain when using Unity Forces. Similar interpretations can be applied to Q2, Q3, Q4, Q5, Q6. This study used the *binom.test* function from R's stats package. Due to it being expected that the gain was positive ($p > 0.5$), a binomial one-tailed test was performed. Results are shown in Table 7.

Table 7. *p*-value results obtained for each perception questionnaire question.

Question	*p*-Value
Q1	0.0002
Q2	0.0002
Q4	0.0020
Q5	0.0106
Q6	0.0106

As observed in Table 7, in all cases h_0 was rejected at the significance level of 0.05. This is statistical evidences that: students liked to use Unity Forces (Q1), they liked the graphic visualizations (Q2), they perceived both simulators as similar (Q4), they thought that the Unity Forces simulator had a better feedback of forces (Q5), and Unity Forces had better visualization (Q6) than HapStatics. It has to be noted that for Q4, it was expected that users perceived that the behavior of both simulators was similar. This question was conceived to validate the physics behavior and visualizations of Unity Forces. On the other hand, Q5 and Q6 were used to contrast both simulators. Lastly, for Q3, the hypotheses were:

$$h_0 : \mu = 3$$
$$h_a : \mu \neq 3$$

This means that in the Likert-scale, a value of three indicates that users perceived both simulators to be equal or similar. Consequently, it is expected that the users find a clear difference between them. This study used a one-tailed *t*-test from the same R package to calculate if the difference was positive and Unity Forces' force feedback response was better than HapStatics. The obtained *p*-value was 0.008817. This value rejected the null hypothesis.

As a conclusion, through the results obtained, it was demonstrated that the proposed approach allows the researchers create visuo-haptic applications faster and with higher visual realism than using typical techniques. Such statements allows the validation of the hypothesis described in Section 1 and encourages the use of the proposed methodology.

7. Conclusions and Future Work

visuo-haptic interactive solutions have been proposed to develop and to enhance students' ability to learn by adding tactile or kinesthetic sensations to visual simulators. A common way to develop a visuo-haptic simulation considers the use of a graphic and physics-based engine orchestrated with a haptic rendering framework. Nevertheless, new powerful simulation tools such as game engines, provide also the opportunity of incorporating a higher level of visual realism to the simulations.

Therefore, in this study an architecture to use game engines to create visuo-haptic interactive simulators is presented and tested. The incorporation of game engines capabilities allowed authors to facilitate the design and deployment of e-learning laboratories. Moreover, in this study a new successful technique to interconnect multiple haptic devices, even from different manufacturers, to the Unity platform is also described. In this context, a plugin named HaDIU was developed in this work in order to enable interconnectivity and functionality of multiple devices, as well as the generation of haptic rendering in game engines. While Unity is used to perform some of the physics calculations and graphic visualization, HaDIU is in charge of connecting haptic devices, sending their data to Unity, calculating interaction forces, and deploying force feedback on them.

The proposed novel architecture successfully demonstrates the connectivity of multiple devices in game engines and, at the same time, that the performance of the application is not affected by the number of connected devices. The performance depends on the inherent physics of the rendered scene. A factory design pattern that facilitates the development of complex environments that make use of haptic feedback is described in this work. This is due to the instantiation of haptic interaction and ideal haptic interaction points' virtual objects. This behavior is kept during the use of multiple haptic devices, and allows developers to create more permissive interactive environments by letting them decide what type and how many devices they will use. The results obtained in this study state that our approach is functional, it enables the use of multiple haptic devices, and it incorporates game engines as visuo-haptic simulation frameworks.

Several simulations were designed and implemented in this research to verify the operability of haptic devices, where each scenario was properly modelled with different haptic objectives. An experimental study was performed to validate that this approach provides better visualizations than an existing single purpose application. Engineering students enrolled in an introductory physics course compared the Unity Forces visuo-haptic simulator developed with the proposed architecture of this work with a similar visuo-haptic simulator developed by the standard procedure. By means of a perception questionnaire, students clearly stated that Unity Forces had both a better force feedback and a better graphic visualization.

A study of CPU utilization showed that even when six haptic devices were connected simultaneously in the environment, force feedback generation and graphic rendering were displayed properly, and no discrepancies on performance consumption were present during their connectivity. These results proved that proper multiple haptic devices connectivity in game engines is achieved by using HaDIU. A comparison of development times to create visuo-haptic interactive simulators was also performed. The results clearly showed that developers can design visuo-haptic simulators faster when using HaDIU than using traditional techniques. Therefore, as future work, we are designing new visuo-haptic scenarios using game engines and the proposed architecture to take advantage of this reduction of development times.

- Addition of new force feedback calculation routines. This improvement would facilitate the development of different scenarios and reduce the time required to develop them.
- Addition of spatial aid. This feature will help users navigate properly and not get lost in the virtual 3D environment, and it will offer intuitive, friendly and accurate interactions.
- Implementation of more case studies on education and training. By using our approach, developers could create cutting-edge learning and training environments and researchers could focus on top priority research areas' issues, such as improving learning outcomes and measuring interactivity.
- Create the HaDIU package to upload it into Unity's Asset Store. This will help other researchers use the proposed approach to develop visuo-haptic interactive simulators.

Author Contributions: Individual contributions are the following; Conceptualization, D.E.-C. and J.N.; data curation, R.A.C.-O. and A.G.-N.; funding acquisition, J.N.; investigation, D.E.-C. and J.N.; methodology, D.E.-C. and J.N.; project administration, J.N.; software, D.E.-C. and R.A.C.-O.; supervision, J.N.; validation, L.N. and V.R.-R.; writing—original draft, D.E.-C.; writing—review and editing, J.N. All authors have read and agreed to the published version of the manuscript.

Funding: This study was funded by a scholarship provided by Tecnologico de Monterrey to graduate student A01170737—David Escobar-Castillejos, a national scholarship granted by the Consejo Nacional de Ciencia y Tecnologia (CONACYT) to study graduate programs in institutions enrolled in the Padron Nacional de Posgrados de Calidad (PNPC) to CVU 559247—David Escobar-Castillejos, and the Science Department of the School of Engineering and Science of Tecnologico de Monterrey.

Acknowledgments: We would like to thank Vicerrectoría de Investigación y Posgrado, the Research Group of Product Innovation, the Cyber Learning and Data Science Laboratory, and the Science Department of the School of Engineering and Science of Tecnologico de Monterrey.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Gavish, N.; Gutierrez, T.; Webel, S.; Rodriguez, J.; Peveri, M.; Bockholt, U.; Tecchia, F. Evaluating virtual reality and augmented reality training for industrial maintenance and assembly tasks. *Interact. Learn. Environ.* **2015**, *23*, 778–798. [CrossRef]
2. Basdogan, C.; Srinivasan, M.A. Haptic Rendering in Virtual Environments. In *Handbook of Virtual Environments: Design, Implementation, and Applications*; Duffy, A., Ed.; Lawrence Erlbaum Associates, Publishers: Mahwah, NJ, USA, 2002; pp. 117–134.
3. Saddik, A.E. The Potential of Haptics Technologies. *IEEE Instrum. Meas. Mag.* **2007**, *10*, 10–17. [CrossRef]
4. Coles, T.R.; Meglan, D.; John, N.W. The Role of Haptics in Medical Training Simulators: A Survey of the State of the Art. *IEEE Trans. Haptics* **2011**, *4*, 51–66. [CrossRef] [PubMed]
5. Escobar-Castillejos, D.; Noguez, J.; Neri, L.; Magana, A.; Benes, B. A Review of Simulators with Haptic Devices for Medical Training. *J. Med. Syst.* **2016**, *40*, 1–22. [CrossRef] [PubMed]
6. Verdaasdonk, E.G.G.; Dankelman, J.; Schijven, M.P.; Lange, J.F.; Wentink, M.; Stassen, L.P.S. Serious gaming and voluntary laparoscopic skills training: A multicenter study. *Minim. Invasive Ther. Allied Technol.* **2009**, *18*, 232–238. [CrossRef] [PubMed]
7. Neri, L.; Shaikh, U.A.; Escobar-Castillejos, D.; Magana, A.J.; Noguez, J.; Benes, B. Improving the learning of physics concepts by using haptic devices. In Proceedings of the 2015 IEEE Frontiers in Education Conference (FIE), El Paso, TX, USA, 21–24 October 2015; pp. 1–7. [CrossRef]
8. Shaikh, U.A.S.; Magana, A.J.; Neri, L.; Escobar-Castillejos, D.; Noguez, J.; Benes, B. Undergraduate students' conceptual interpretation and perceptions of haptic-enabled learning experiences. *Int. J. Educ. Technol. Higher Educ.* **2017**, *14*, 1–21. [CrossRef]
9. Huang, H.M.; Liaw, S.S.; Lai, C.M. Exploring learner acceptance of the use of virtual reality in medical education: a case study of desktop and projection-based display systems. *Interact. Learn. Environ.* **2016**, *24*, 3–19. [CrossRef]
10. Hassani, K.; Nahvi, A.; Ahmadi, A. Design and implementation of an intelligent virtual environment for improving speaking and listening skills. *Interact. Learn. Environ.* **2016**, *24*, 252–271. [CrossRef]
11. Yuksel, T.; Walsh, Y.; Krs, V.; Benes, B.; Ngambeki, I.B.; Berger, E.J.; Magana, A.J. Exploration of affordances of visuo-haptic simulations to learn the concept of friction. In Proceedings of the 2017 IEEE Frontiers in Education Conference (FIE), Indianapolis, IN, USA, 18–21 October 2017; pp. 1–9. [CrossRef]
12. Han, I.; Black, J.B. Incorporating haptic feedback in simulation for learning physics. *Comput. Educ.* **2011**, *57*, 2281–2290. [CrossRef]
13. Hamza-Lup, F.G.; Baird, W.H. Feel the Static and Kinetic Friction. In *Haptics: Perception, Devices, Mobility, and Communication*; Isokoski, P., Springare, J., Eds.; Springer: Berlin/Heidelberg, Germany, 2012; pp. 181–192.
14. Neri, L.; Noguez, J.; Robledo-Rella, V.; Escobar-Castillejos, D.; Gonzalez-Nucamendi, A. Teaching Classical Mechanics Concepts using Visuo-haptic Simulators. *Educ.l Technol. Soc.* **2018**, *21*, 85–97.

15. Host, G.E.; Schonborn, K.J.; Lundin Palmerius, K.E. A Case-Based Study of Students' Visuohaptic Experiences of Electric Fields around Molecules: Shaping the Development of Virtual Nanoscience Learning Environments. *Educ. Res. Int.* **2013**, *2013*, 1–11. [CrossRef]
16. Gaudina, M.; Zappi, V.; Bellanti, E.; Vercelli, G. eLaparo4D: A Step Towards a Physical Training Space for Virtual Video Laparoscopic Surgery. In Proceedings of the 2013 Seventh International Conference on Complex, Intelligent, and Software Intensive Systems, Taichung, Taiwan, 3–5 July 2013; pp. 611–616. [CrossRef]
17. Pavaloiu, I.B.; Sandu, S.A.; Grigorescu, S.D.; Dragoi, G. Inexpensive dentistry training using virtual reality tools. In Proceedings of the 10th International Technology, Education and Development Conference, Valencia, Spain, 7–9 March 2016; pp. 279–285. [CrossRef]
18. Poyade, M. Motor Skill Training Using Virtual Reality and Haptic Interaction: A Case Study in Industrial Maintenance. Ph.D. Thesis, University of Malaga, Malaga, Spain, 2013.
19. Huang, C.; Sumit, K.; Ladak, H.M. Virtual Reality Simulator for Training in Myringotomy with Tube Placement. *J. Med. Biol. Eng.* **2016**, *36*, 214–225. [CrossRef]
20. Huang, G.; Metaxas, D.; Govindaraj, M. Feel the "Fabric": An Audio-haptic Interface. In *Symposium on Computer Animation*; Breen, D., Lin, M., Eds.; The Eurographics Association: Geneva, Switzerland, 2003; pp. 52–61. [CrossRef]
21. Nguyen, M.; Melaisi, M.; Cowan, B.; Uribe Quevedo, A.J.; Kapralos, B. Low-end haptic devices for knee bone drilling in a serious game. *World J. Sci. Technol. Sustain. Dev.* **2017**, *14*, 241–253. [CrossRef]
22. Bogert, K. Falconunity. 2018. Available online: https://github.com/kbogert/falconunity (accessed on 5 April 2020).
23. Berney, S.; Haddad, R.; Hauck, R.; Gradl, G. Improving spatial abilities with a 3D immersive environment: A pilot study. In Proceedings of the 16th Biennial EARLI Conference for Research on Learning and Instruction, Limassol, Cyprus, 25–29 August 2015; pp. 1–4.
24. Andaluz, V.H.; Chicaiza, F.A.; Gallardo, C.; Quevedo, W.X.; Varela, J.; Sánchez, J.S.; Arteaga, O. Unity3D-MatLab Simulator in Real Time for Robotics Applications. In *Augmented Reality, Virtual Reality, and Computer Graphics*; De Paolis, L.T., Mongelli, A., Eds.; Springer International Publishing: Cham, Switzerland, 2016; pp. 246–263.
25. Adams, R.J.; Hannaford, B. Stable haptic interaction with virtual environments. *IEEE Trans. Robot. Autom.* **1999**, *15*, 465–474. [CrossRef]
26. García-Castelán, R.; Neri, L.; Noguez, J.; Robledo-Rella, V.; Nucamendi, A.G. Understanding the interaction of charge distributions using visuo-haptic simulators. In Proceedings of the Memorias CIIE, Monterrey, Mexico, 16–18 December 2019; pp. 799–807.