International Journal of User-System Interaction 13(3) 2020, 149-168

Continuum – a virtual musical instrument

Diana Andronache, Andrei Bulai, Dorin-Mircea Popovici

Ovidius University of Constanta
124 Mamaia Bd, 900527, Constanta Romania
diana.andronache7@gmail.com; bulai.andrei10@gmail.com; dmpopovici@univ-ovidius.ro

Abstract. The aim of this paper is to present a possible development in the area of virtual musical instruments. Our solution consists of creating a version of a Laser Harp, a virtual musical instrument that uses laser beams to replace the strings of a real harp. The best current version of Laser Harp uses a Head-Mounted Display (HMD) with controllers that must be held by the user as they play the instrument. This is a common occurrence when it comes to virtual musical instruments. This research paper presents a way to allow users to play the virtual harp with their bare hands, without the need for controllers. In what follows, we will present the steps that were taken in building such an instrument that we named Continuum and the results that we achieved during the process.

Keywords: Virtual Reality, Leap Motion, virtual musical instrument, Laser Harp, audio system.

DOI: 10.37789/ijusi.2020.13.3.3

1. Introduction

Art has always been a part of human life, therefore the artistic field is one of interest for humanity, in particular, the branch of art related to music. The music field, nowadays, is a very complex one. From various singing techniques to complex musical theories to more and more interesting instruments, this field has developed at a fast pace.

The purpose of this paper is to contribute to the evolution of this field, more precisely it analyzes the evolution and development of virtual musical instruments. As the world evolved, musical instruments (like most things) tended to virtualize. Therefore, we can assume that the virtual version of a musical instrument is an evolved version of it. Possible advantages of virtual instruments compared to classic ones (depending on one instrument) are compactness, portability, the need for less physical effort to be used.

In the following sections, we will present the details of the raised problem,

the steps that were taken in the development of an application to solve this problem and the results.

2. Research contribution

This paper is the result of research on the field of virtual musical instruments. Most virtual musical instruments have a common problem, which, for some instruments, is quite serious. This problem is the representation of the user's hands in the virtual world, more precisely its rigidity.

For instruments that do not require finger mobility, such as drums, this is no longer a (big) impediment. But for instruments such as guitar, piano, or the like, for which it is important for the instrumentalist to have a fluid movement of the fingers, it is a big problem that we tried to find a solution to.

In cases like this, people have limited themselves to the movement of the hand, without being able to move all the fingers or to a method of playing the instrument without the need to move the fingers (such as pressing buttons). The question that arises is: "Is there a method that allows the user to have additional flexibility of movement in their fingers, and not the palm exclusively?"

To solve this problem, we chose to focus on a single musical instrument (the application can be extended to any similar instrument, which has the same difficulty of mobility), that is the Laser Harp as presented by Goran Ferenc (2014).

3. State of the art

The Musical field, as we mentioned before, is a very developed one. This work focuses on virtual musical instruments. They are too numerous to be listed, most of which are inspired by a classical instrument, which has a real physical body and produces sounds. From better-known instruments such as piano, violin, guitar, drums, to lesser-known instruments such as xylophone, viola, panpipes, triangle, bagpipes, there are various virtual representations of them, both in 2D and 3D. Some representations are relatively faithful to physical instruments, others have various changes that, in most cases, are related to the visual part, not the functional part. Examples of musical virtual applications are described by Roberto De Prisco (2016), Panagiotis

Tzevelekos (2008), Dionysios Marinos (2011), Zhimin Ren (2012), Zhimin Ren (2015) and Raul Altosaar (2019).

Some of the most popular 2D virtual instruments are applications for piano, guitar, drums, sound station, violin, or other more or less known instruments, that can be used on mobile or desktop. They vary greatly in number and scope. A simple search on Google Play (2021) for android applications that comprise musical instruments provides over 200 examples.

In terms of 3D, the applications are not as numerous, but they are in considerable numbers. Two of the most popular applications that contain multiple 3D instruments are Studio Virtual Reality (VR) Instrument provided by SkyWatcherVR (2019), and Exa - The Infinite Instrument provided by Aesthetic Interactive (2020). Each of these two VR applications is composed of an environment in which there are several musical instruments such as guitar, piano, drums, etc. The difference between them is the layout of the instruments: the appearance of those presented in Studio VR resembles that of real instruments, while the instruments in Exa are made up of various simple geometric shapes. The virtual environment is shareable: several people can connect simultaneously to play as a band. To use these virtual environments, VR equipment consisting of HMD with controllers is required.

Other fairly well-known VR applications are Tribe XR (2018), Electronauts provided by Survios (2018) and SynthVR provided by developer 42tones (2020). These applications are different music editing studios. They Require the same equipment mentioned above (HMD and controllers) to function and moreover, they require premade soundtracks that will be edited inside the virtual environment.

Considering that our problem only applies to 3D instruments, we will focus on them and, in particular, on how they receive the signals that trigger the production of sounds. Most instruments (mentioned above and in general) use a VR setup that contains an HMD and two controllers. Both controllers make it possible to track the hands and transmit the corresponding signals related to the position of the hands. This is the rigid version of the user's hand representation because it is related only to the position of the hand in space. A good example is Aesthetic Interactive (2020). There are also some representations such as SkyWatcherVR (2019), that are a little closer to the real-life concept, as they monitor the position of the fingers with 3 to 5 buttons (being limited to only two finger positions: in the air or pressing the button), but they are also quite rigid.

As Leap Motion (2014) reveal, there are some musical instruments (physical or virtual) that use a Leap Motion Controller as an extension of the instrument, for various purposes. Most of these instruments use the controller to trigger actions (with the help of certain hand gestures) associated with sound stations. We believe that this Controller can be used effectively to solve our problem as well.

As mentioned in the previous chapter, we choose to build a musical instrument similar to the Laser Harp. The Laser Harp is a musical instrument similar to a classic harp that has laser beams as a replacement for nylon or metal strings. Its functionality is also different from a class harp: instead of pinching the strings to produce certain sounds, it is necessary to interrupt the laser wave by blocking it with the hand, to send a signal to the software system, which will produce a sound. These Laser Harps are of several types depending on the number and colour of the strings, the appearance of the frame and the signal transmission system that generates sounds corresponding to musical notes. There are also more modern versions of a Laser Harp available in the form of desktop applications such as the one developed by Rob Wilmot (2018), mobile applications such as the Laser Harp Lite developed by Timegalore (2014), or even a VR Laser Harp application developed by Niki Odolphie (2016). The latter is available on Steam (2016) for $ 1.99 and has a design similar to Continuum, our virtual instrument. The difference between the two is both in appearance and in the signal transmission system (the VR Laser Harp uses the controllers that provide the rigid method of moving the user in the virtual environment (aforementioned), while the Continuum uses Leap Motion Controller that allows free movement of fingers).

4. Proposed solution

To solve the problem of rigidity and at the same time obtaining an experience as faithful as possible to reality for the user experience, we implemented a virtual musical instrument inspired by the Laser Harp, which we called "Continuum" (the application can be extended to similar instruments, that are dependent on the natural mobility of the hands). We built a virtual model based on a representation from Goran Ferenc (2014), Magun I (2006) and Ling Long (2013)), which has the element that solves the mentioned problem.

4.1. Application design

Our proposed model has a 3D representation that can be interacted within a VR environment. The structure and functionality of this application have been established during the application design stage as per paper of Bogdan Crenguta (2008).

Structure

We chose that the structure of the application would be as simple as possible in order to emphasize functionality. It is composed of three main elements: the application universe, the musical instrument and the user's representation, as shown in the diagram below (Figure 1).

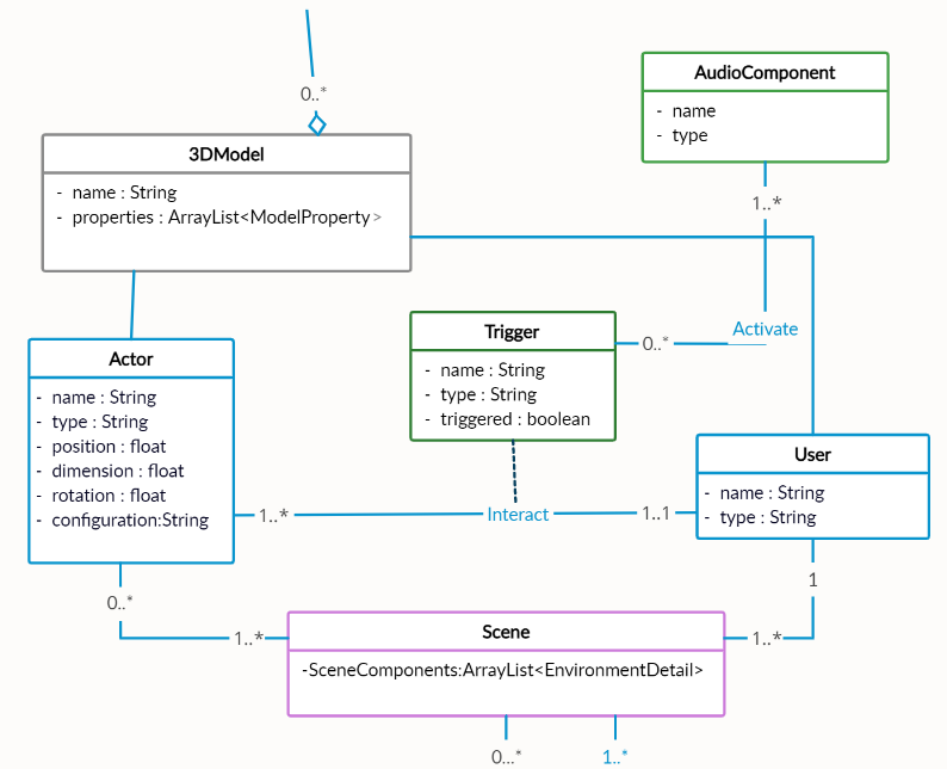

Figure 1. Class diagram that shows the application structure

The table below (Table 1) describes the three components that make up the application and the most important elements associated with them.

Table 4 – The components of the application

Component	Description
Universe	The most voluminous component. Represents the general framework of the application, which includes all other components of the application.
Scene	Represents a certain level (map) of the application. The universe can contain several Scenes. In the case of the presented application, there is only one scene (the one in which the virtual musical instrument is modeled). It contains the Environment, Actors and a user.
Ambient	Represents the framework that establishes the general aspect of the Scene. The environment consists of Environment Details.
Environment Detail	Represents a component of the Environment, which contributes to a certain part of its appearance. Example: AtmosphericFog, Landscape, LightSource, etc.
3DModel	Represents a modeling of a 3D object that can be imported into the application.
Model Property	Represents a feature of a 3D model. Examples: NoVertexes, Color, etc.
Actor	Represents a 3DModel located in the Scene, with which the user can interact.
User	Represents a 3DModel (like Actors) placed in the Scene, which can interact with Actors (the user is different from Actors). The user is assigned the characteristics of the real person who controls him (position and rotation given by the motion tracking sensors).
Trigger	Represents a condition when an event is triggered.
Audio Component	Attached to the event Trigger. It is used to generate the sounds of the Continuum instrument.

Functionality

The selection of possible user actions is showcased in Figure 2. Once the

application has booted, the virtual world is loaded and the user is presented with a main menu that allows them to:

- change a number of settings (the volume of sounds, for instance);
- open the application (the stage, where they can interact with the virtual instrument);
- exit the application.

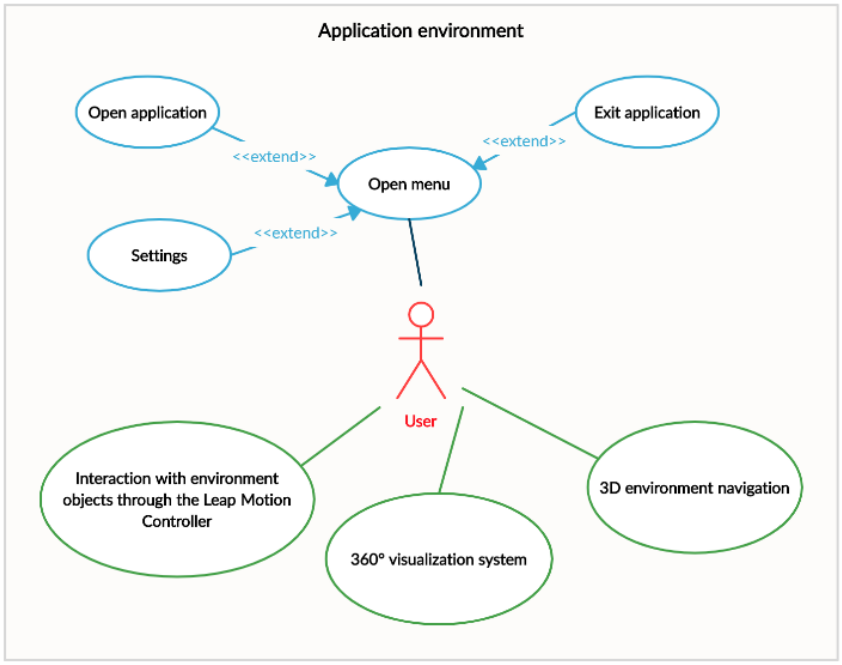

Figure 2. Actions that the user can perform

Also, within the application, the user has the options that are specific to any VR application (except the Leap Motion technology we added afterwards):

• The possibility of interaction with various elements from the scene, called actors. In the case of the application presented in this paper, the interaction will be performed using Leap Motion technology and triggers;

• Ability to view in 360 degrees, using the VR visual system (from HMD);

• Navigation in 3D modelled space, using the tracking system built into HMD and Leap Motion.

The sequence diagram in Figure 3 shows the order in which the events attached to a trigger occur. This order reproduces the logic functionality of the application, i.e. the interaction between the user and the virtual environment. The whole process starts from the user, who activates one of the triggers by touching it with his hands (using controllers/Leap Motion hand tracking). From the trigger, the data attached to it is transmitted to the component that manages the application (Application Manager) and to the virtual environment. The application manager analyzes the situation and forwards to the User Interface (UI) information about what it should generate as a user interface (depending on the gesture made by the user or on the position of the hand), to the virtual environment information about what changes have occurred and to the scene information on what events should run.

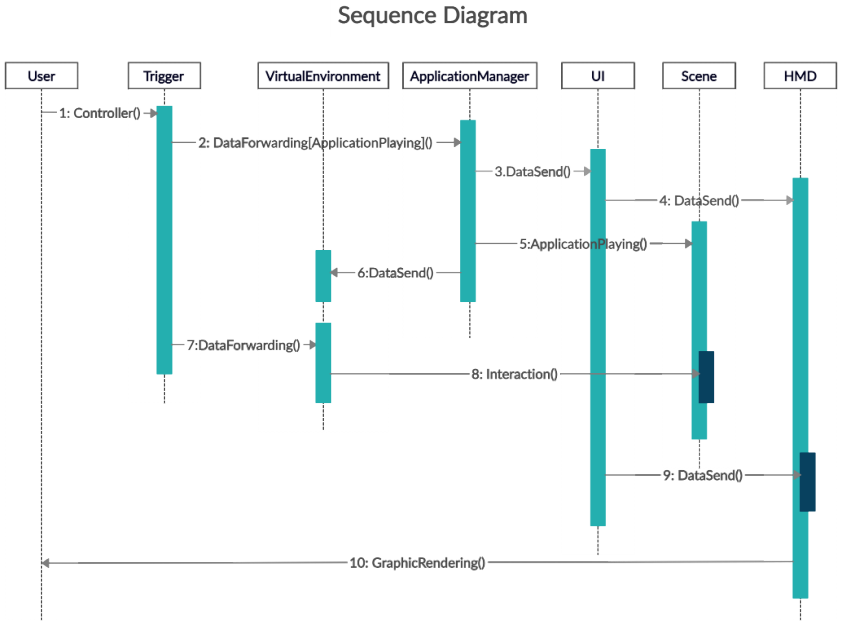

Figure 3. Trigger activation

The scene also receives information from the virtual environment related to the interaction. The UI forwards to the HMD the previously received information regarding the interface that it will have to display. Based on the received information, the HMD (through its visualization system) displays to

the user the graphical rendering of the virtual environment with all the changes produced.

The technologies used in the application

Hardware components used:

- Oculus Rift S provided by Facebook Technologies (2019);
- Leap Motion Controller provided by Leap Motion (2013);
- Personal Computer (PC).

Software components used:

- Oculus App version 16.0 (SDK – software development kit), the last version released for Oculus Rift S provided by Facebook Technologies (2020);
- Leap Motion Orion 4.0.0 (SDK) provided by Leap Motion (2018);
- Unreal Engine 4.11 provided by Epic Games (2016);
- VisualStudio 2019 provided by Microsoft (2019).

We chose an older version of Unreal Engine than the most recently released (Unreal Engine 4.25), because newer versions are unstable in terms of the plugins needed to establish the connection between the Leap Motion Controller and the editing engine. For the rest of the components, the area of flexibility is quite large. The reason for using them is the simple fact that those are the latest versions available at the moment.

4.2. Implementation

Based on the aspects presented in the previous section and information about how to properly build a virtual application provided by Andrei Sherstyuk (2009), the way to implement the application is further detailed. From the programming techniques used, to the final results and elements that make up the application framework, all these have their own level of importance in creating a framework, seemingly simple, but with a high degree of complexity in terms of functionality.

For the implementation of the application, we used for the most part the Blueprints (Visual Scripting system included in Unreal Engine). This is a complex and complete system of scripts, based on the concept of using an interface composed of nodes, which render the structure and mode of operation of the elements that make up the application developed with their help.

Like many script-based languages, Blueprints help define OO (object-

oriented) classes and objects. The programming language used to generate those classes (and their associated objects) is C ++.

The flexibility offered by this system makes it possible for programmers to use it to the same extent that it can be used by designers, because designers have access to tools and concepts that are normally only available to programmers, and programmers can create a basic system that can be further developed by designers as described by Unreal Engine (2019).

For the implemented concepts we used, mainly, the predefined classes within the previously mentioned system, making changes in the code only where necessary. As for the modelling elements, we used the editor and editing tools provided in Unreal Engine.

The scene

The scene represents the space in which the user is, surrounded by various actors. It is a fraction of the universe of the application.

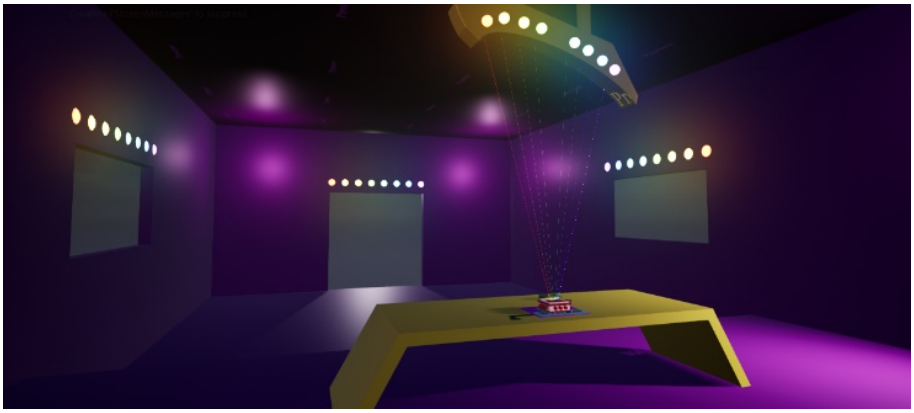

Figure **4**. Application scene with the virtual musical instrument Continuum

In this case (Figure 4), the universe of the application consists of a single scene, which in turn is composed of a user (only one person can enter the virtual environment of the application at a time), a main actor (Continuum), secondary actors (for example walls and table) and the elements that make up the environment, such as:

- atmosphere (Atmospheric Fog);
- global light source;
- Point Light (emitted from a single point);
- Sky Sphere;

• Sky Light.

Materials and textures

The material represents the component that renders the appearance of the 3D object, giving it a realistic look (or not, if this is the goal) as described by Plowman Justin (2016).

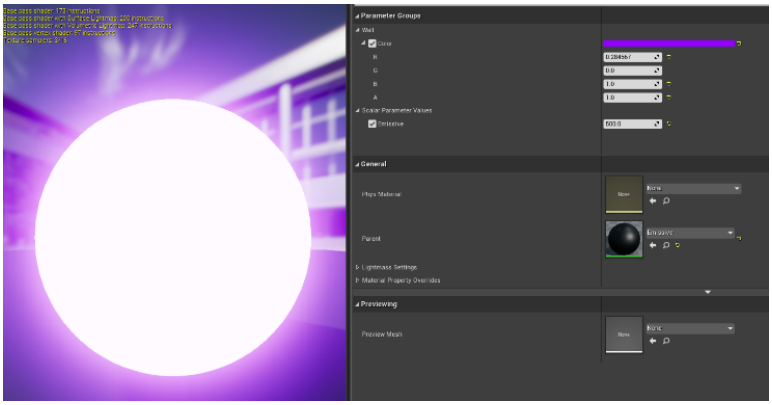

Figure **5**. Emissive material for LED actors

In the component of the light-emitting diode (LED) spheres (Figure 5), that we used to give an extra graphic volume to the scene, we made an emissive material, using instances of it for each element in the scene.

In order to use the same emissive property of the LEDs within each actor, we parameterized both the colour register and the value of its intensity.

Parameterizing certain values exponentially reduces the time required to create materials, because a single material is sufficient for all objects (no need to create new material for each object).

Actors

The 3D representation of the musical instrument (so-called Continuum) is the main actor, placed on the stage. The minimalist design of the stage, which contains few elements that make up the setting, puts emphasis on the musical instrument, which is built in a more detailed manner.

Unreal Engine offers the possibility to populate the scene with certain predefined actors for the convenience of designers. We chose not to use these default settings because they did not fit the needs of our application. Thus, all the objects placed in the scene were modelled from scratch.

Objects, materials and instances were created for Continuum and the rest of the actors, each one having its specific properties, in order to be imported in a newer version of Unreal Engine, or in another editor, for later versions.

Modelling procedures

There are many ways to model a three-dimensional object. Among the main modelling techniques that were used for modeling the application framework are:

• polygonal modelling - involves the application of basic operations, such as scaling, rotation, translation, on simple primitives, in order to obtain complex models;

• NURBS modelling (Non-Uniform Rational Basis Spline) - is used when the nature of the object has a high curvature and detail complexity;

• parametric modelling - this can be an alternative to NURBS modelling when the editor used does not have this technique implemented; it is done by twisting, pulling and applying boolean operations to an object;

• patch modelling - this is used to deform a terrain, a canvas, a plan or, as the name implies, a patch; it involves the use of Bézier curves after which a group of vertexes is arranged;

• polygonal subdivision modelling - this technique involves dividing all the faces, or only certain faces, into smaller faces, in order to achieve a fine transition between the angles formed by the initial polygons of a model.

The user (the Leap Motion component)

The user can interact with the rest of the actors present in the scene, can walk and can visualize around or the whole context of the scene, as described in Figure 2 and by William R (2003).

In the first stage of development, we implemented the user's representation using the controllers from the Oculus Rift setup, that came with a predefined 3D model. This model is often used with similar virtual instruments, including the VR Laser Harp. Its mobility, as mentioned before, is reduced to moving only 3 fingers or group of fingers in a predefined way. Basically, it only tracks the orientation and direction of the hand without the capacity to do the same for the fingers and thus simulating the movement realistically.

In the second stage, the representation of the user in the virtual environment is achieved through the 3D model of floating hands which is presented in Figure 6.

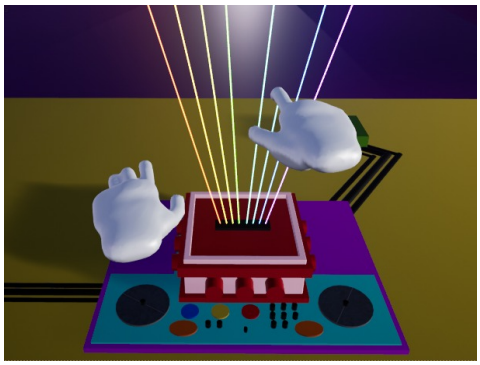

Figure **6**. User representation in the virtual environment.

This 3D model is named LeapFloatingHands and is obtained with the help of a plugin embedded in the Unreal Engine. The plugin needs to be enabled from the plugins tab as shown in Figure 7, and the default pawn changed into the LeapFloatingHands character.

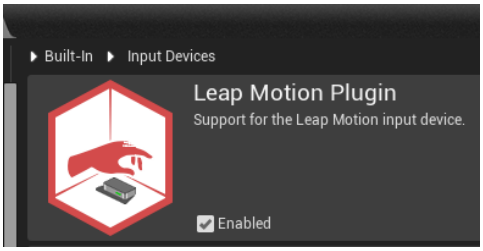

Figure **7**. Enable the Leap Motion plugin.

We choose not to make any changes to the code (blueprints) of the LeapFloatingHands because the predefined version was sufficient for our needs. When the Leap Motion plugin is enabled, the option to overwrite the code is available as described by Getnamo (2018). That means we have access to the events listed in Figure 8, that are triggered by the hand's motion and that make the mapping of the real hands and the virtual 3D model.

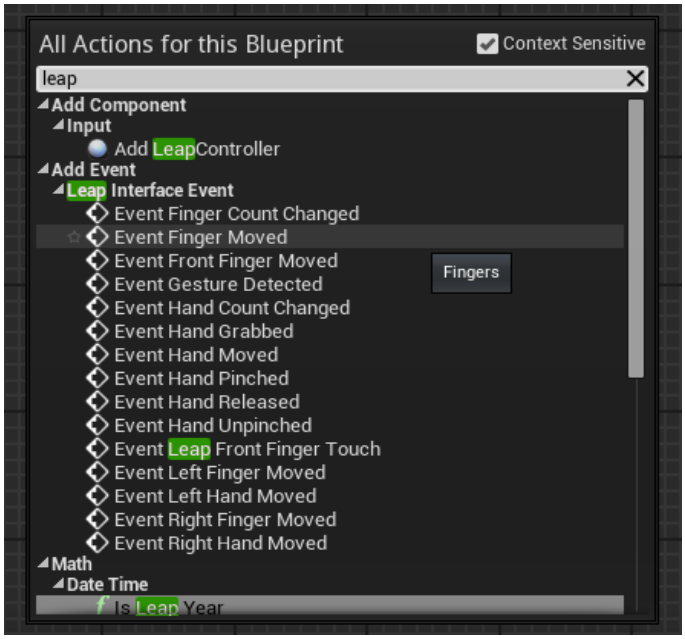

Figure **8**. Leap Motion predefined events.

These events are very important for our goal (achieving a free movement of the fingers) because they help to recognize when a single finger is moving. When performing an instrument such as a guitar (for example), if the player is right-handed, he needs to be able to move all fingers of his left hand except the thumb and all fingers of his right hand except the little finger (depending on the technique that the musician uses to perform), each finger generating a different action than the rest.

The floating hands are controlled through the Leap Motion Controller which is connected to the computer with the Leap Motion SDK. The Controller has integrated two infrared cameras, with a resolution of 640 × 240 pixels, for detecting the position and orientation of bones and joints.

Graphical interface (UI – user interface)

Through the visualization system incorporated in the HMD system, the user can see the content of the modelled universe within the application. It has a stereoscopic image rendering capability, peripheral vision and allows 360 ° rotation.

The user interface is friendly, given that relatively simple objects have

been modelled, containing shades of colour and shading effects that are pleasing to the eye, their brightness is adjusted so as not to disturb, and the scenario is indoors, which renders a plus comfort in the VR experience, through the physical stability it suggests.

Also worth mentioning is the menu created, which helps the user to exit the application safely, to continue using the musical instrument and to set the properties of some parameters that adjust the volume of the sounds generated by it.

Figure 9 shows the menu options, which correspond to the actions mentioned in the use case diagram (Figure 2). These allow the user, by pressing buttons, to enter the stage corresponding to the label on the button.

When scrolling the button on the slider, with the help of several events, the user can change the volume of the musical note played by the instrument strings and adjust the SFX component (sound effects) which provides the sound effect from the interaction of objects in the scene.

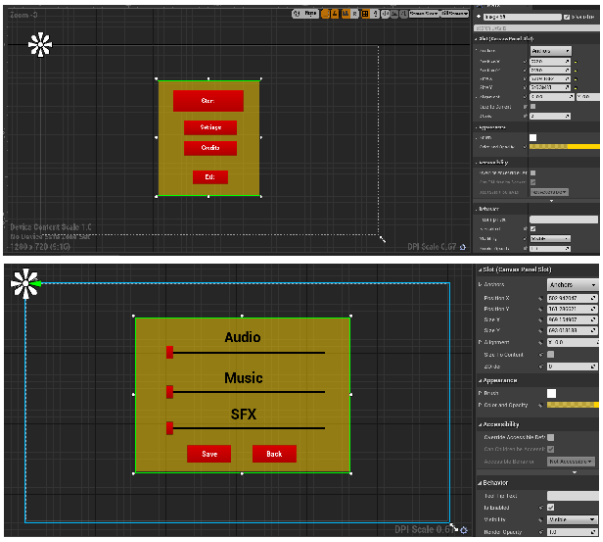

Figure **9**. Application menu in the design phase.

The menu was created using Blueprints technology and the visual editor from Unreal Engine, which is similar to the one from Visual Studio, by the drag-and-drop method. They automatically generate the C ++ code corresponding to the objects in the interface and the events created.

Audio system

The sounds generated by the instrument are musical notes. These notes also correspond to musical intervals. The best known musical interval is the octave, whose name comes from the number of musical notes that fall into that interval: eight notes.

For simplicity, we chose to build the instrument eight-stringed, corresponding to an octave, and as musical notes associated with the strings, we chose the main octave, due to its popularity. It contains the notes A, B, C, D, E, F, G and an extra C (which in fact represents the beginning of the secondary octave), unaltered, i.e. without sharp or flat. These choices make the virtual environment one easy to learn into as described by Julien Saunier (2016).

The sounds, as mentioned before, are distributed to the strings and their playback is activated when the trigger corresponding to the string is triggered by touching its hitbox with any of the hands.

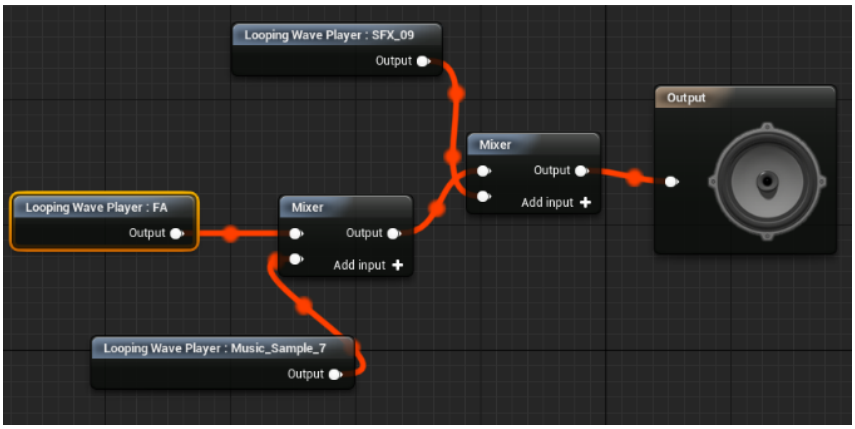

Figure **10**. Audio component of a string.

The figure above (Figure 10) shows the construction of the sound attached to a string, using Blueprints. We used three Wave Looping nodes that contain references to the following basic sounds:

- musical note F;
- the stamp of the Continuum instrument;
- the sound effect of touching the string.

4.3. Testing and user interaction

In the end, we made two versions of the application: one that uses the controllers that need to be held in order to track the user's hands position and one that uses the Leap Motion Controller to track the user's hands and fingers position and orientation. The purpose was to use both versions and to compare them.

The use of both versions is very simple by setting HMD and connect the Leap Motion Controller to it only for the second version (both physically and through the software) and then run the VR content, i.e. the developed application.

The interface created is simple and user-friendly. As the test team consisted of 15 people, including 2 people over the age of 40 and two under the age of 12.

People who initially tested the application were asked to answer the question: "Which of the two versions is more natural and more accurate in performing musical techniques?" 3 out of 15 people claimed that they did not find differences between the two, and the remaining 12 people claimed that the first version, the one with controllers, is more rigid and inaccurate than the version with Leap Motion.

The following test was performed by 15 professional musicians or amateur musicians. To the same question, everyone answered that the second version is better, some of them claiming that it is much easier to use and that they can interpret more complex compositions.

In the testing phase, we also monitored the time that each subject spent inside the virtual environment, for both versions. They were told to stay connected as long as they wanted. The average time per person spent in the virtual application was with 6.7 minutes greater for the version with the Leap Motion. That reveals the fact that the second version of our application (the one that uses the Leap Motion technology) is more entertaining than the first one (the one that uses the controllers).

We also measure the accuracy of the two versions by asking the testers to report each time they hit a string unintentionally. For the rigid version of the user's hands, the results were by 20% less accurate. Therefore, the Leap Motion version won our tests.

5. Conclusion

In this paper, we managed to complete a fully functional application, which contains a virtual musical instrument, called "Continuum". If we take into consideration the feedback from the people who tested the instrument, we can confirm that we achieved the goal that was mentioned at the beginning of the paper: we implemented a method that allows the user to move their fingers freely in the VR representation. This achievement involves a small step towards the evolution of the field of research related to virtual musical instruments.

Given that the equipment and technologies used are of the latest generation, the instrument can be imported and updated later in newer versions. It also implies that the experience offered (the virtual one) is one of superior quality.

The limitations of this application are the lack of tactile sensation when interacting with the musical instrument and the absence of the instrument's weight. Also, if the setup of the Leap Motion Controller is not proper (the Controller should be mounted on the HMD) the problem of its range arises, as it only has an area of about 80 cm.

Further research in this domain could be finding a method to add more ways of interaction with the virtual environment, such as including tactile sensation by adding haptic sensors.

References

Aesthetic Interactive. *Exa – The Infinite Instrument*. 1 2020. https://store.steampowered.com/app/606920/EXA_The_Infinite_Instrument/

Andrei Sherstyuk, Dale Vincent, and Anton Treskunov. 2009. *Towards Virtual Reality games*. In Proceedings of the 8th International Conference on Virtual Reality Continuum and its Applications in Industry (VRCAI '09). Association for Computing Machinery, New York, NY, USA, 315–316. DOI:https://doi.org/10.1145/1670252.1670322

Bogdan Crenguta. Concern-Oriented and Ontology-Based Design Approach of Software Architectures. 10 2008. DOI: 10.1109/SYNASC.2008.6.

Dionysios Marinos, Björn Wöldecke, Chris Geiger, and Tobias Schwirten. 2011. *Design of a touchless multipoint musical interface in a virtual studio environment*. In Proceedings of the 8th International Conference on Advances in Computer Entertainment Technology (ACE '11). Association for Computing Machinery, New York, NY, USA, Article 33, 1–7. DOI:https://doi.org/10.1145/2071423.2071464

Epic Games. *Unreal Engine.* 3 2016. https://www.unrealengine.com/en-US/download

Facebook Technologies. *Oculus Rift S.* 5 2019. https://www.oculus.com/rift-s/

Facebook Technologies. *Oculus App.* 4 2020. https://www.oculus.com/setup/

Getnamo. *UE4-LeapMotionPlugin.* 6 2018. https://github.com/getnamo/leap-ue4

Google Play. *Musical instruments.* 2021. https://play.google.com/store/search?q=musical+instrument&c=apps

Goran Ferenc and Jelena Popovi´c-Bozovic. *An infinite beam laser harp with external MIDI I/O functionality*. 11 2014, DOI: 10.1109/TELFOR.2014.7034545.

Julien Saunier, Mukesh Barange, Bernard Blandin, Ronan Querrec, and Joanna Taoum. 2016. *Designing adaptable virtual reality learning environments*. In Proceedings of the 2016 Virtual Reality International Conference (VRIC '16). Association for Computing Machinery, New York, NY, USA, Article 5, 1–4. DOI:https://doi.org/10.1145/2927929.2927937

Leap Motion. *Leap Motion Controller.* 2013. https://www.ultraleap.com/product/leap-motion-controller/#pricingandlicensing

Leap Motion. *Leap Motion Orion.* 6 2018. https://developer.leapmotion.com/releases/leap-motion-orion-400

Leap Motion. *4 Ways that Motion Control is Transforming Electronic Music.* 6 2014. https://blog.leapmotion.com/4-ways-motion-control-transforming-electronic-music/

Ling Long and Zhaoren Lu. *A design of electronic laser harp based on scm*. Advanced Materials Research, 765-767, 04 2013, ISBN: 978-94-91216-42-8 - ISBN: 1951-6851, DOI: 10.2991/icsem.2013.39.

Magun I., Guerrero F., and Jimenez E.J. *Gestural Control Using a Laser Harp*. 03 2006. ISBN: 0-7695-2505-9. DOI: 10.1109/CONIELECOMP.2006.32.

Microsoft. *Visual Studio 2019*. 4 2019. https://visualstudio.microsoft.com/vs/

Niki Odolphie. *VR Laser Harp.* 11 2016. https://www.wearvr.com/apps/vr-laser-harp

Panagiotis Tzevelekos, Anastasia Georgaki, and Georgios Kouroupetroglou. 2008. *HERON: a zournas digital virtual musical instrument*. In Proceedings of the 3rd international conference on Digital Interactive Media in Entertainment and Arts (DIMEA '08). Association for Computing Machinery, New York, NY, USA, 352–359. DOI:https://doi.org/10.1145/1413634.1413698

Plowman Justin. *3D Game Design with Unreal Engine 4 and Blender*. Packt Publishing Ltd, 6 2016.

Raul Altosaar, Adam Tindale, and Judith Doyle. 2019. *Physically Colliding with Music: Full-body Interactions with an Audio-only Virtual Reality Interface*. In Proceedings of the Thirteenth International Conference on Tangible, Embedded, and Embodied Interaction (TEI '19). Association for Computing Machinery, New York, NY, USA, 553–557. DOI:https://doi.org/10.1145/3294109.3301256

Roberto De Prisco, Delfina Malandrino, Gianluca Zaccagnino, and Rocco Zaccagnino. 2016. *Natural User Interfaces to Support and Enhance Real-Time Music Performance*. In

Proceedings of the International Working Conference on Advanced Visual Interfaces (AVI '16). Association for Computing Machinery, New York, NY, USA, 204–211. DOI:https://doi.org/10.1145/2909132.2909249

SkyWatcherVR. *Instrument Studio VR.* 12 2019. https://www.oculus.com/experiences/rift/1746672708729216/

Steam. *VR Laser Harp.* 11 2016. https://store.steampowered.com/app/543280/VR_Laser_Harp/

Survios. *Electronauts.* 8 2018. https://survios.com/electronauts/

Tribe XR. *Tribe XR.* 2018. https://www.tribexr.com/

Unreal Engine. *Blueprints Visual Scripting*. 2019. https://docs.unrealengine.com/en-US/Engine/Blueprints/index.html.

William R. Cockayne. 2003. *Virtual reality. Encyclopedia of Computer Science*. John Wiley and Sons Ltd., GBR, 1835–1839.

Zhimin Ren and Ming C. Lin. 2015. *Interactive virtual percussion instruments on mobile devices*. In Proceedings of the 21st ACM Symposium on Virtual Reality Software and Technology (VRST '15). Association for Computing Machinery, New York, NY, USA, 79–83. DOI:https://doi.org/10.1145/2821592.2821616

Zhimin Ren, Ravish Mehra, Jason Coposky, and Ming Lin. 2012. *Designing virtual instruments with touch-enabled interface*. In CHI '12 Extended Abstracts on Human Factors in Computing Systems (CHI EA '12). Association for Computing Machinery, New York, NY, USA, 433–436. DOI:https://doi.org/10.1145/2212776.2212820

42tones. *SinthVR*. 6 2020. https://42tones.itch.io/synthvr.