# ANNALS

## OF THE

# UNIVERSITY OF CRAIOVA

Series: AUTOMATION, COMPUTERS,
ELECTRONICS and MECHATRONICS

Vol. 15 (42), No. 1, 2018

ISSN 1841-0626

**ANNALS OF THE UNIVERSITY OF CRAIOVA**
**Series: AUTOMATION, COMPUTERS, ELECTRONICS AND MECHATRONICS**
**Vol. 15 (42), No. 1, 2018 ISSN 1841-0626**

***Note*: The “Automation, Computers, Electronics and Mechatronics Series”** emerged from **“Electrical Engineering Series” (ISSN 1223-530X)** in 2004.

ANNALS OF THE UNIVERSITY OF CRAIOVA
Series: Automation, Computers, Electronics and Mechatronics, Vol. 15 (42), No. 1, 2018

# Design of Three-Dimensional Museum-Like Environment by using Virtual Reality

Andrei O. Dragomirescu, Florin Stinga

*Automation and Electronics Department, University of Craiova*
*Craiova, Romania (e-mail: andreid93@gmail.com, florin@automation.ucv.ro)*

**Abstract:** This paper showcases the production steps undertaken in the creation process of a three-dimensional, virtual museum-like environment, accomplished using the cross-platform Unity 3D and the aid of a third-party mixture of media production and editing tools. The consequent goal of the process is creating a commercial application available to users operating on a Microsoft Windows environment or (and) a mobile device running a version of the Android operating system.

***Keywords:*** Virtual museum environment, Unity 3D, Virtual Reality, Windows application, Android application

## 1. INTRODUCTION

This paper covers the main stages of creating a museum-like environment embedded as application used by any human user on a Windows or Android operating system platform. Over the last years, the field of virtual world design is a new area of research without a strong theoretical or methodological foundation (Jakobsson, (2018)).

Virtual reality uses a set a technique that artificially creates immersive sensory experience of physical presence in places in the real or imagined world, allows the interaction between the user and that world (Ferrari and Medici, (2017)). This technique has been applied in many areas of science and visual communication, such as restoration of cultural relics, indoor decoration design, landscape design and so on, allowing access from different places and times, even that the original sites are temporal or geographical inaccessible to the human users (Ferrari and Medici, (2017), Yubin and Yufen, (2014), Higgins et al., (1996), Maietti et al.(2017)).

The showcased application is one depicting a medieval settlement, structured in such fashion to allow the final user to traverse the medium in a free manner, immersing himself in a museum-like, three-dimensional virtual representation with the extended possibility of virtually guiding said user. The application developed implies the mixed usage of several applications, used in media creation and editing processes, whose results can and are introduced in the main engine at every step in the development process.

This work was supported by a grant of the Romanian National Authority for Scientific Research and Innovation, CCCDI–UEFISCDI, project number 78 BM.

Unity 3D is the main engine used to produce the virtual environment, the graphical and programmable medium, in order to embed the externally developed assets, and the final rendering tool used in the building process (Unity 3D, (2018)). Despite the fact that Unity is an integrated environment capable of offering all the necessary tools required to produce inside the engine itself, several other software third-packages are utilized in the presented application in this article, with the sole goal of creating a virtual medium as realistically close as possible. The current graphical capabilities of the computational hardware in today’s world makes such objectives quite reachable and not that difficult to implement and run natively on several platforms, whether they seem to be a bulky tower personal computer, or a small smartphone tucked in one of your pockets. The complementary software packages used in the process of development are as follows: 3DS Max (3DS Max, (2018)), Ableton Live (Ableton Live, (2018)), Adobe Photoshop (Adobe Photoshop (2018)), FMOD Studio (FMod Studio, (2018)), SpeedTree (SpeedTree, (2018)).

An important aspect that requires mentioning, is that the programming, although implementable in various ways and languages in the Unity Engine, is done extensively in C# (C Sharp), due to its versatile way of approaching object-oriented programming and the various advantages related to the integration with the .NET Framework and its application programming interface.

This article is structured as following: In Section 2 both the medium development and the control solution utilized in the traversal of the virtual environment is summarily, for both Windows and Android operating systems, showcasing the slight variations in the input and response of the developed system-based applications, Section 3 describes the process of creation and implementation of the layers and assets used for the virtual immersion of the user in the virtual medium, in Section 4 the integration of all the previously developed elements into a built and executable, end-user application, is introduced, and Section 5 presents conclusions from the presented studies.

## 2. VIRTUAL MEDIUM

Both the medium's various terrain and architectural representations and its control scheme used in the traversal of said medium, are structured in the Unity project hierarchy as "Assets". These assets can be regarded as components and game objects, and can range from the three dimensional medium itself, all the way to the architectural elements and the virtual foliage representations. The terrain component can quite easily be instantiated as a sterile three-dimensional plain, devoid of any other elements that will be added at future steps in the process. Unity has an integrated set of tools that will allow an in-depth shaping of the terrain, to its utmost finest detail, offering the possibility of creating quite a realistically (or if desired, a surreal) representation of landforms. A quick visual example of the afore mentioned can be observed in the Figures 1 and 2, captured in order to showcase the intermediary and final results of the creation, shaping and texturing of the terrain-environmental elements.

Unity offers the capability of using a mixture of textured graphical elements, from both inside the included Unity packages and external sources. Several such terrain and architectural textures are introduced into the project and used after several subsequent editing steps in Adobe Photoshop, as it shown in Figure 3. The control solution implemented, just as the previous elements, is structured as a game object. More specifically, this object implies an implementation of a parent-child like relationship between two assets. The two assets are categorized as:

FPC – The First-Person Controller: In this scenario, the parent of the group, responsible for the scripting aspects of the control solution and the process of control regarding the collisions and behavioural interactions with other elements present in the virtual medium.

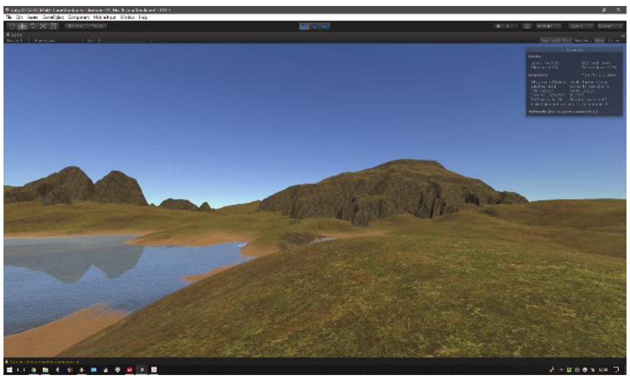
Fig.1: Running in - Engine Intermediary Terrain Preview

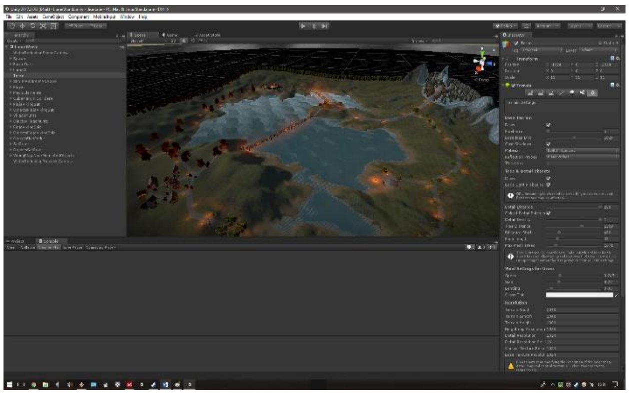
Fig. 2: Final reference of the virtual medium in Unity

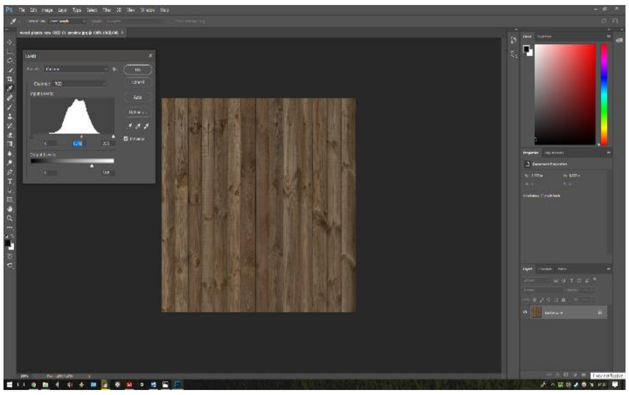
Fig. 3: Graphical elements editing done in Photoshop

Camera – The child, positioned at the "eye-level" of the control object, used as a point of view for the end user.

In Unity, programming and modification of the scripting elements is achieved mainly in the Microsoft development environment, Visual Studio Basic (Visual Studio, (2018)), bundled together with the initial installation of the Unity client (C# code extract) (Dragomirescu, (2018)):

```
//...
if (m_CharacterController.velocity.sqrMagnitude > 0 && (m_Input.x != 0 || m_Input.y != 0)){
m_StepCycle+=(m_CharacterController.velocity.magnitude+(speed*(m_IsWalking ? 1f : m_RunstepLenghten)))*Time.fixedDeltaTime;}
if (!(m_StepCycle > m_NextStep)){
return;}
m_NextStep = m_StepCycle + m_StepInterval;
PlayFootStepAudio();}
private void PlayFootStepAudio(){
if (!m_CharacterController.isGrounded){
return;}
//...
if(m_CharacterController.velocity.magnitude>0&&m_CharacterController.isGrounded){
```

*m_Camera.transform.localPosition=m_HeadBob.DoHeadBob (m_CharacterController.velocity.magnitude + (speed*(m_IsWalking ? 1f : m_RunstepLenghten)));*

*newCameraPosition = m_Camera.transform.localPosition;*

*newCameraPosition.y = m_Camera.transform.localPosition.y - m_JumpBob.Offset();}*

*//…*

Due to the variations of the input offered by the Windows platform, which implies the usage of keyboard-mouse combination or other controllers, and the compact devices running an Android operating system (like smartphones and tablets), which use a touch input, the control solution needs to possess a platform specific set of instructions, that will execute only on the specific platform the application is built for. The previously mentioned, programmable component present in the FPC, can be carefully modified to the extent of allowing and executing platform specific code in order to satisfy the needs for multi-platform application development (C# code extract) (Dragomirescu, (2018)):

*//...*

*if(!FindObjectOfType<EventSystem>()){*

*GameObject obj = new GameObject("EventSystem");*

*obj.AddComponent<EventSystem>();*

*obj.AddComponent<StandaloneInputModule>().forceModuleActive = true;}*

*//…*

*namespace UnityStandardAssets.CrossPlatformInput.PlatformSpecific*

*{public class StandaloneInput : VirtualInput{*

*public override float GetAxis(string name, bool raw){*

*return raw ? Input.GetAxisRaw(name) : Input.GetAxis(name);}*

*public override bool GetButton(string name){*

*return Input.GetButton(name);}*

*public override bool GetButtonDown(string name){*

*return Input.GetButtonDown(name);}*

*public override bool GetButtonUp(string name){*

*return Input.GetButtonUp(name);}*

*//…*

The Android version of the developed application has an additional graphical layer, of visual toggles, that implements equivalent controllable functions available on the keyboard and mouse version of the application.

Combining the two developed assets and packaging them in an executable fashion at this point in the development process, will allow the developer, artists, programmers and testers, to have a general idea regarding the possibilities and limits of the following developmental steps, and assure the quality of the previous and future steps that must be implemented.

## 3. IMMERSION

Unity allows its developers to both develop many re-usable assets inside the engine itself, or import externally developed assets, from its own Store, the Unity Asset Store or elsewhere. Game objects that are developed to be easily re-used are referred to as Prefabs (Prefabricated Objects). The prefabs are essentially templates of game objects that can be easily instantiated and quickly modified in other to create variations of the previously developed assets. Immersion is an experience in one moment in time involves a lack of awareness of time, a loss of awareness of the real world, involvement and a sense of being in the task environment (Jennete et. al, (2008)). An immersive world, even a virtual one, requires variation in its implementation of visually related elements. A quite repeatable visual pattern of the same graphical elements can detract from the experience and lower the quality desired by the developer's previously set objectives. The prefab concept can be quite the helpful solution to tackle such an issue.

Prefabricated elements can vary wildly from a complexity's standpoint in the developed application, but in itself the process of creating and modifying such an element remains fundamentally the same, whether implemented piece by piece inside Unity or in a third-party software (e.g. 3DS Max) and then imported in Unity. The Figures 4-7 shows the steps followed for one such asset inside the Unity engine.

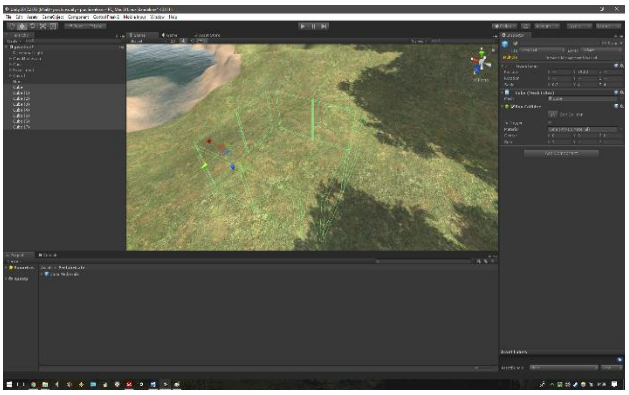
Fig 4: Prefab Step 1 – Creating the sub - components

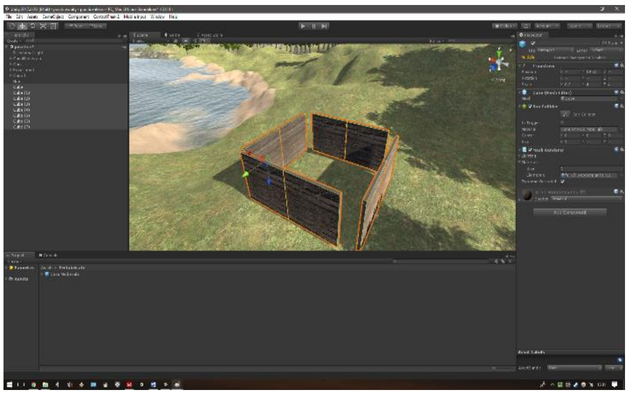
Fig 5:  Prefab Step 2 – Attaching graphical meshes

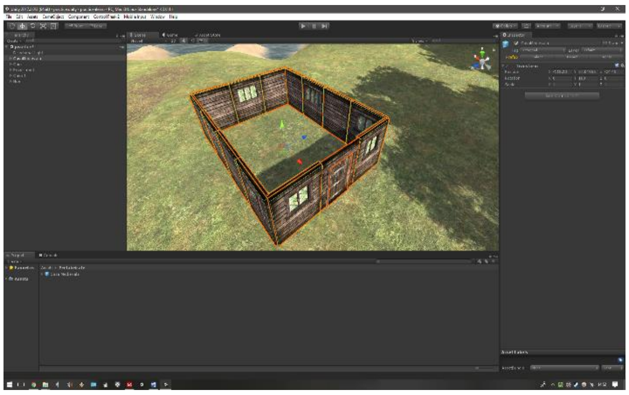

Fig 6: Prefab Step 3 – Arranging of resulted components

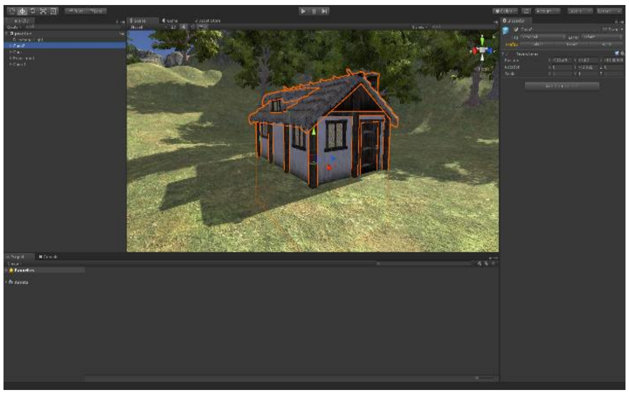

Fig 7: Prefab Step 4 – Grouping components as a Prefab

The usage of the prefabricated elements can be taken to new levels, when its implementation is brought to a smaller, detail-oriented application on objects used to populate the environment, in order to create a museum-like visual medium.

In the current application, the goal was to simulate a historical medieval setting, so the result is modelled to appear like an architecturally realistic setting that could have theoretically been populated at one point in the past as presented in Figures 8 and 9.

To further increase the quality and immersion of the virtual world, several other operations such as adding elements of foliage and the implementation of auditory elements are introduced.

Auditory elements are introduced into the Unity project with the combined help of the following software packages: Ableton Live 9 and FMOD Studio (see Figures 11 and 12).

Foliage is implemented through SpeedTree (see Figure 10). In itself the process of creating the foliage elements is quite easy to grasp and implement. The structure of each element is usually an inter-linked list of generated constructive elements, textured appropriately to resemble a realistic (or if desired, a surreal), equivalent of a real-life foliage element. The composition and editing of the used auditory elements are done entirely inside the digital audio workstation, Live 9, and subsequently imported into FMOD Studio as high-fidelity, sequenced audio files.

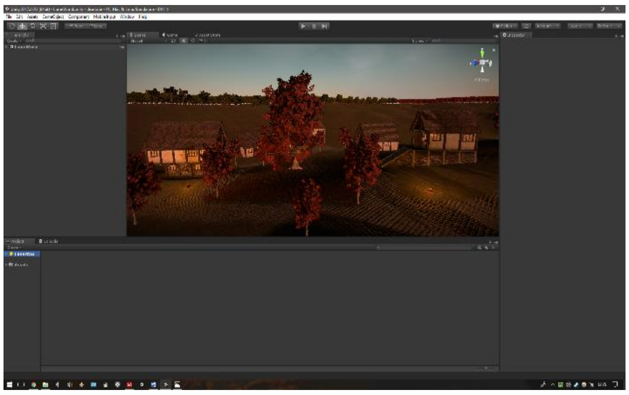

Fig. 8: Virtual Medium – Outdoors Medieval Setting

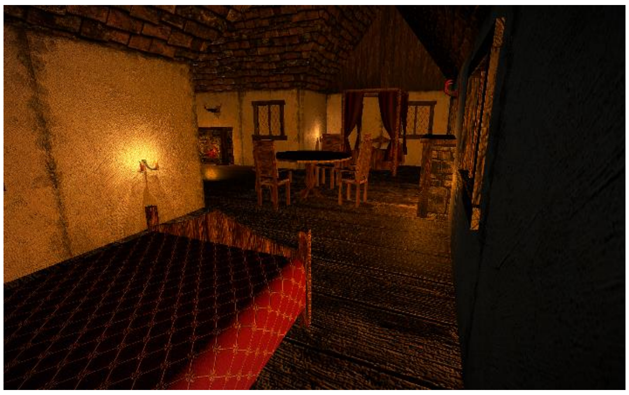

Fig. 9: Virtual Medium – Indoors Medieval Setting

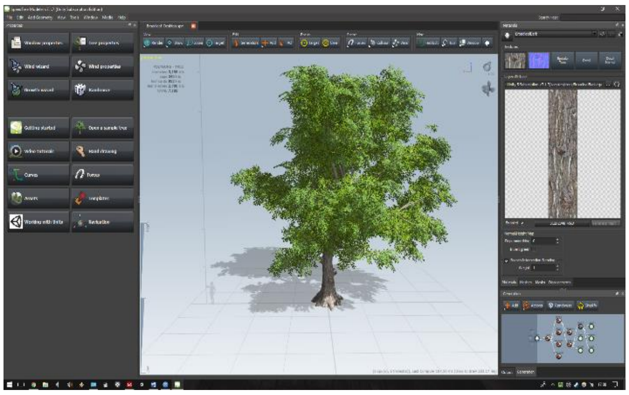

Fig. 10: Foliage elements done in SpeedTree

Then, these files are programmed via C# scripts inside the Unity project itself. FMOD acts as an intermediary, which allows high quality conditioned playback of the audio files, according to the rules and sets of instructions set in the afferent Unity project.

## 4. BUILDING

In the Unity environment, the final step that needs to be undertaken in order to finalize the project and obtain a structured installer file (in the case of an Android build), or an executable file (in the case of a Windows build).

Building options vary from platform to platform. Each and every build implements some lesser or higher quality form of texture compression, based on Unity texture compression algorithms. The lack of applied compression is also an available option, should this be the desired outcome. In Figure 13 is presented the Build Menu inside Unity Engine.

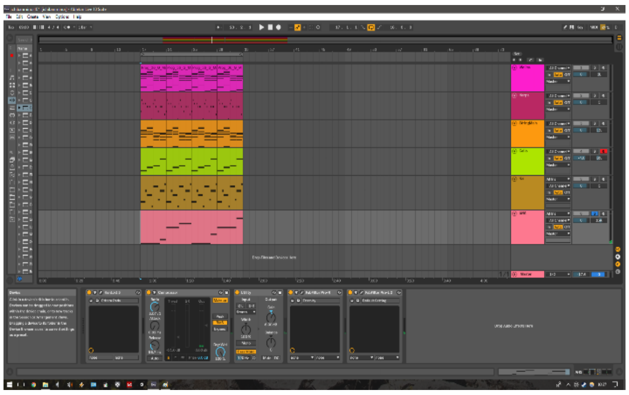

Fig 11: The Ableton Live environment – the musical score composition and the signal processing done on one of its generated tracks.

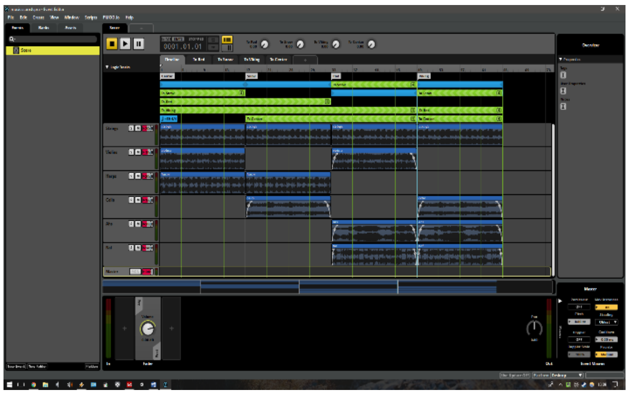

Fig 12: FMOD Interface – High fidelity audio files are structured and sequenced according to the desired programmable structure

Whilst the building process of a Windows version is quite straightforward, requiring only the selection of the Build option, the Android built will require an automated conversion of the used assets, using the Android SDK (Software Development Kit), alongside its afferent app-developing software package, Android Studio.

The engine offers building for many platforms, not just Windows/Android. Unity allows building for commercial hardware platforms such as iOS, Mac, Linux, Sony PlayStation, Xbox One, Oculus Rift, Steam VR and even TV's powered by the Android and Apple operating systems alongside many other more.

The developed application for the Android operating system can be installed and ran not just smartphones but on Android TV's just as effectively, but this implementation would most likely require an additional set of control instructions, specific to the TV operating system environment.

As previously mentioned the application described is built for both the Windows and Android operating systems.

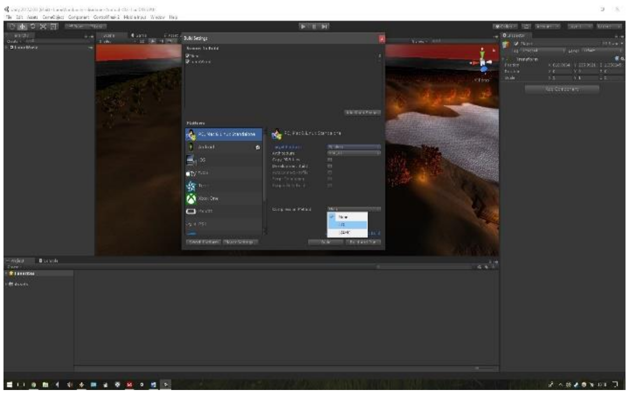

Fig 13: The Build Menu inside Unity Engine

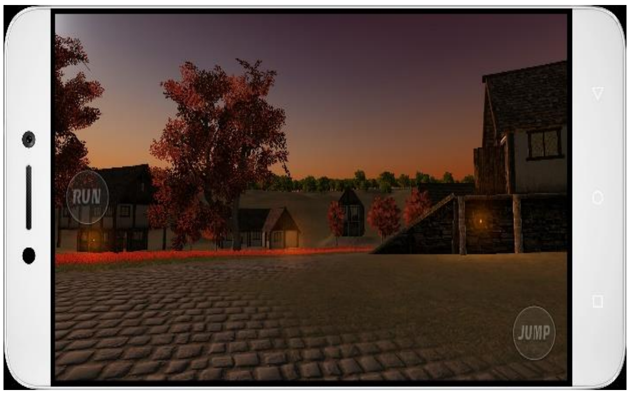

Fig. 14: The final Android interface implemented on a mobile device

In Figure 14 is presented the final Android interface of the developed application.

While the Windows version is not concerned with trivial aspects such as texture compression due to higher powered systems that usually run Windows as an operating system, the Android version has applied on itself a form of light compression that will maintain a very close high-fidelity look, just as its Windows equivalent. This is introduced due to subjective observations, regarding the implementation of a more optimized high-frame per second functionality on mobile devices and may be detrimental to users whom value the visual aspects more than an optimal performance and a lower battery consumption on their mobile Android devices.

## 5. CONCLUSIONS

In this paper were presented the main stages of creating a medieval settlement, structured in such fashion to allow the final user to traverse the medium in a free manner, immersing himself in a museum-like, three-dimensional virtual representation, embedded as an application usable by any human user on a Windows or Android operating system platforms.

Virtual reality is a technique used in presentation of a heritage application allowing access from different times, even that the original sites are temporal inaccessible to the human users.

Despite the rather complicated approach used in the development of this application, using Unity 3D and several third-party software packages, a virtual medium as realistically close as possible is created.

## 6. REFERENCES

3DS Max, Autodesk Inc., available on-line at https://www.autodesk.com/products/3ds-max/overview, 2018.

Ableton Live, Ableton AG, available on-line at https://www.ableton.com/en, 2018.

Adobe Photoshop, Adobe Inc., available on-line at https://www.adobe.com/ro/products/photoshop.html, 2018.

Dragomirescu A., Aplicație in Unity 3D, Graduated Thesis, *Faculty of Automation, Computers and Electronics, University of Craiova*, 2018.

Ferrari F. and Medici M., The Virtual Experience for Cultural Heritage: Methods and Tools Comparison for Geguti Palace in Kutaisi, Georgia, *Proceedings of International and Interdisciplinary Conference IMMAGINI?* Brixen, Italy, 2017.

FMod Studio, Firelight Technologies Pty Ltd., available on-line at https://www.fmod.com, 2018.

Higgins T., Main P., Lang J., Imaging the Past: Electronic Imaging and Comouter Graphics in Museums and Archaeology, *British Museum, UK*, 1996.

Jakobsson M., A virtual realist primer to virtual world design, available on-line at http://www8.informatik.umu.se/~mjson/files/primer.pdf, 2018.

Jennete C., Cox A., Cairns P., Dhoparee A., Epps A., Tijs T. and Walton A., Measuring and Defining the Experience of Immersion in Games, *International Journal of Human-Computer Studies* 66(9):641-661, 2008.

Maietti F., Di Giuliom R., Balzani M., Piaia E., Medici M., Ferrari F., Digital Memory and Integrated Data Capturing: Innovations for an Inclusive Cultural Heritage in Europe through 3D Semantic Modelling, In Mixed Reality and Gamification for Cultural Heritage, *Springer, Berlin*, pp. 225-244, 2017.

SpeedTree, IDV Inc., available on-line at https://store.speedtree.com, 2018.

Unity Technologies, Unity 3D, available on-line at https://unity3d.com, 2018.

Visual Studio, Microsoft Corporation, available on-line at https://visualstudio.microsoft.com, 2018.

Yubin L. and Yufen F., The virtual reality tehnique of lanscape arhitecture reconstruction, *BioTechnology Journal*, 10(11), pp. 5226-5233, 2014.

ANNALS OF THE UNIVERSITY OF CRAIOVA
Series: Automation, Computers, Electronics and Mechatronics, Vol. 15 (42), No. 1, 2018

# Automotive Application for Collision Avoidance

Florina Luminița Besnea*, Ștefan-Irinel Cismaru**

** Department of Mechatronics and Robotics, University of Craiova
Craiova, Romania (e-mail: besnea.florina@yahoo.com)
** Department of Automatic Control and Electronics, University of Craiova
Craiova, Romania (e-mail:cisstefan@gmail.com)*

**Abstract:** Car transport is crucial for society and has grown in terms of both number and complexity of the vehicles, their interaction and the traffic situation. The emergence of the technology focuses mainly on reducing human effort in every field. This paper proposes the development of an automated, easy-to-integrate and cost-effective electronic system model that will help reduce collisions. The developed system can be considered as a step towards minimization of mental and physical efforts made by the driver to control the vehicle safely. Collision avoidance is a research area that has become more relevant in recent years as the vehicles have been improved by adding driving assistant systems.

*Keywords:* automotive, car, electronics, sensorics, collision.

## 1. INTRODUCTION

Advanced driver assistance systems are developed to automate, adapt and improve vehicle systems for safety and a better driving manner. The automotive safety system's journey begins in 1901 with the Oldsmobile Curved-Dash, which offered a foot brake by direct pressure of a rail on the transmission shaft.

The vehicle has been used as a means of transport over 100 years. A few decades ago, electronic technologies and sensors were introduced to vehicles as intelligent steering assistance systems. This kind of system helps drivers in guidance for safety and comfort. Smart systems offer great potential for future mobility. Several sources indicate that the benefits of intelligent driver assistance systems are significant. The sensors monitor the driving circumstances and detect dangerous events, filling the sense of vision, distance and orientation of the human being.

A large part of the total vehicle accidents is due to the lack or decrease in the concentration of the drivers during the operation of the vehicles. Some drivers tend to handle distracting activities such as radio tuning, eating, talking to passengers or, the most common, making and taking phone calls. Other drivers find it difficult to focus only on driving, for example because of fatigue and health problems. Elderly drivers may experience difficulties in personal mobility, making it more difficult to constantly monitor the vehicle's perimeter. They can also develop other conditions that have a negative impact on their ability to focus on the road.

Failure to identify a vehicle on the side of the car, as known as the blind-spot, is still a cause of the increasing number of accidents. For some drivers, the simple solution is to place an additional side mirror. However, this is not the best solution because these additional side mirrors do not give an accurate picture of the actual or estimated distance to an object or other vehicle.

Today's vehicles are in a much better position from the point of view of safety, but unfortunately today drivers have at their disposal different devices that distract them when handling a car, causing more and more challenges in terms of avoiding possible collisions in traffic.

## 2. RELATED STUDIES

The main trends of today are oriented towards the effort for efficient transport combined with managing the high demands of society in terms of transport safety. Official accident statistics show that the number of vehicles and accidents has increased over the last 50 years. Automotive safety systems and applications play an important role, with the predisposition for modern cars to become 5% safer each year.

Road safety statistics for 2016, published by the European Commission, show a 2% decrease in the number of deaths registered in the EU last year. In 2017, 25,500 people were killed in road accidents in the EU, 600 fewer than in 2015 and 6,000 fewer than in 2010. Other 135,000 people suffered serious injuries, according to Commission estimates. This statistic shows the increase in population awareness for automotive applications and systems that are based on detecting or preventing an accident.

However, Romania is on the penultimate place in terms of the number of people killed in road accidents per one million inhabitants. Thus, Romania recorded 97 deaths per million, being overtaken only by Bulgaria, with 99 deaths per million.

From the annual publication of the National Institute of Statistics on road traffic accidents involving persons under the influence of alcohol in 2017, it appears that in Romania there have been 3801 accidents that have resulted in death or injury to one or more persons. Of this number, the number of accidents involving directly car drivers was 1688.

The study shows that the most frequent road accidents involving vehicles are produced in rural areas in 55%, in urban areas 37%, and on motorways they cover an 8% pricing.

## 3. COLLISION OVERVIEW

Collision is an event in which two or more bodies exert their forces on each other within a relatively short period of time. A traffic collision is called a collision with a motor vehicle collision (MVC) and occurs when a vehicle collides with another vehicle, pedestrian, animal, roadside remnants or other stationary objects such as a tree or a pillar (Jonas Jansson 2005). These collisions often result in injuries, deaths, and material damage.

A number of factors contribute to the collision risk, including vehicle design, operating speed, road design, road environment and driver qualification. Worldwide, collisions with cars lead to death and disability, as well as financial costs for both the company and the people involved in such attacks.

### *2.1 Collision stages*

The pre-impact period is called an ante-collision stage where there is no contact between cars. It is delineated in time from the initial moment of impact and the moment of triggering the imminent danger, the main objective being to assess the possibilities of avoiding the accident.

The collision stage occurs at the time of first contact between the vehicles and the moment of their separation. In this phase of collision the deformation energy is obtained from the kinetic energy, the contact between the vehicles having two different phases: compression and restitution.

The post-collision stage is triggered when the cars are separated and lasts until they are stopped.

The blind-spot is the area around the car that cannot be directly observed by the driver during driving.

### *2.2 The blind-spot*

Blind-spots are found in several categories of vehicles: cars, ships and aircraft. Correctly adjusting the mirrors and using technical solutions can eliminate or attenuate the blind-spot.

It has been statistically proven that many accidents occur when changing the lane when an overtaking manoeuvre is initiated, and on the relevant lane there is already another overtaken vehicle.

Checking with rear-view mirrors without checking a blind-spot is not enough. It should be noted that if a vehicle is located at a very narrow distance on the left or right, it may be in the mirror area that suffers from lack of visibility. Blind-spot checking lasts a few fractions of a second, and the prolonged viewing of the angle at this point is dangerous due to the rapid change in traffic. In order to avoid such a dangerous situation, certain collision detection and prevention systems have been developed in the dead corner.

### *2.3 The side collision*

The blind-spot monitoring system continuously scans both sides of the vehicle with ultrasonic sensors and warns the driver when another vehicle is in the detection area. The scanned area is always larger than the blind-spot area itself. The detection area starts at 40 cm from the vehicle at the level of the rear-view mirror.

The side collision detection sensors operate on the same principle as the front collision with the indication that the measures applied in case of detection are not as severe.

## 4. SYSTEM DEVELOPMENT

Each ultrasonic sensor is interfaced with Arduino to determine the distance to the obstacles.

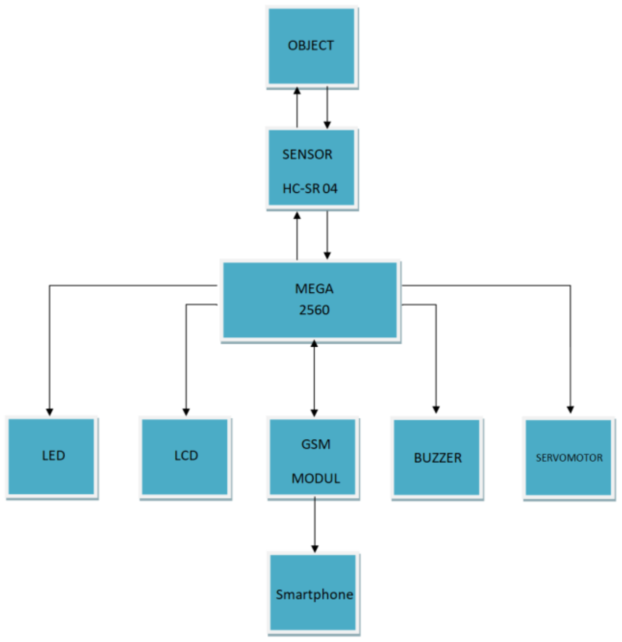

Fig.1 System block diagram

Circuit design, simulation, and block diagram are used to explain the processes involved during the project to achieve the objectives. The block diagram completely covered the process, from the ultrasonic sensor to the system.

The block diagram shows how an object is detected by the Arduino Mega 2560 by means of ultrasonic sensors and the actions taken by it are based on the detected