Potential of Low Cost Motion Sensors Compared to Programming Environments

Juraj Mihaľov, Emília Pietriková, Anton Baláž, Branislav Madoš, Norbert Ádám

Department of Computers and Informatics, Faculty of Electrical Engineering and Informatics, Technical University of Košice, Letná 9, 04200 Košice, Slovakia

Email: {juraj.mihalov, emilia.pietrikova, anton.balaz, branislav.mados, norbert.adam}@tuke.sk

Abstract: The article investigates systems, which represent a modern and popular approach to Virtual Reality and controlling systems. We would like to focus on low-cost motion sensors used in applications which are oriented on object tracking and gesture recognition. There are various types of sensors. Some of them measure the infrared light reflected from the opposing surface, previously emitted by the device in to gather information about any movement in the observed environment. Another way how to recognize not only a moving object present in the environment, but also its gestures and further characteristics of the movement is to use the Kinect. Therefore, we included Kinect also in our research. There is also a sensoric device called Leap Motion, which is specially developed to analyze gestures of human hands and track their motion with very high accuracy. We will provide pros and cons of every mentioned type of sensors or sensoric devices. Our aim is to summarize specific characteristics of mentioned devices to evaluate their ability to be beneficial in the recently very intensively expanding IoT sector. Considering new trends, we decided to focus on low cost sensors in to make our research more relevant also for small businesses and start-ups whose initiative leads to further development of sensoric soloutions and involving them in IoT. We decided to include also Myo Armband. It uses eight electromyography sensors, combined with a gyroscope and an accelerometer to sense electrical activity produced by the muscles of the forearm. Of the multiple programming environments available, we decided to compare and evaluate three programming engines most frequently used for programming applications processing sensoric data. For gaming purposes, the Unreal and Unity 3D engines are the most frequent. For robotics, medicine or for industrial purposes usually LabVIEW is the best choice. In this, we compare the aforementioned three programming environments using different algorithms, utilizing the three motion controllers, and we discuss their (dis)advantages and programming perspectives.

Keywords: Leap Motion; LabVIEW; Myo Armband; Unity 3D; Unreal Engine

1 Introduction

As a motion controller, Kinect provides an intuitive way of controlling a computer, eliminating the need of a keyboard, a mouse or other input devices. It can track up to six people simultaneously and adjust its microphone field, so it can recognize the talking person. This low-cost device has changed the meaning of computer control. However, from our point of view, the main disadvantage of Kinect is its field of view. This can be partly solved by using a Leap Motion device, which – similarly to Kinect – employs reflected infrared light. Though it is not able to track the entire body, only the hands, it can recognize gestures and moves at a high-level, including real-time reactions. A combination of Kinect with Leap Motion could upgrade the sensed area and allow high-precision hand tracking, close to the sensor [1]. Since a direct view of the sensors (using the infrared spectrum) is required, they mainly recognize gestures. That is why current research focuses on the ability to track the user, to allow him/her to turn back or sideways to the sensors and have gesture recognition still sufficiently accurate. In addition to infrared light measurement, another possibility is to use electromyographic sensors, gyroscopes or accelerometers affixed directly to the human body [2]. An example of such a peripheral device, utilizing the aforementioned technologies, is the Myo Armband. From our point of view, combining low-cost sensors, such as Myo Armband, Kinect and Leap Motion, can result in tracking users to a distance of up to 4 meters and reliably recognize gestures even in case of poor visibility. Moreover, such a combination may lead to integration with CAVE systems (Cave Automatic Virtual Environment), representing a fully immersive virtual reality system [3]. To exploit their full potential, one has to evaluate the accuracy of the sensors in the available programming environments (including sensor control features). CAVE consists of several subsystems, in fact separate agents, working in parallel with a huge amount of data [4]. Currently, the market offers several possibilities. To select the optimal movement recognition technology and user's gestures is as important as to choose corresponding development environment. It is important to consider also its reliability, availability of plugins and updates. Our first two choices were the Unreal and Unity engines, both mostly used in computer games and entertainment applications. These softwares are ideal for households and for private purposes; however, they are not sufficient for industrial purposes. Therefore, our third choice was LabVIEW, a programming environment preferred in the industry and robotics [5]. In this paper, we evaluate these three programming engines using small applications created to control three motion sensors, focusing on their effectiveness as well as their user-friendliness [6].

2 Motion Sensors

As already stated, current research is aimed at creating a sensor network, consisting of several various sensors employing infrared light. Notable examples of infrared depth sensors include Kinect, DUO3D, Intel Realsense, Leap Motion and, in case of outdoor environment analysis, Fotonic [7]. Of these, we selected Kinect and Leap Motion, being low-cost devices and, as stated in [8] and [9], their data may be combined in a complementary way. From Kinect data, we generate 3D images of the target, while Leap Motion data allow accurate tracking of hands and fingers. Authors of [10] discuss a combination of Kinect and Leap Motion, testing their accuracy and reliability in ASL (American Sign Language) gesture recognition. From user view, we focus on research on sensors that are radiating infrared light spectrum. It is important to examine the reliability and accuracy of current motion sensors in order to increase their quality. There are projects like DMF (Deformable Model Fitting), an algorithm for 3D face recognition using Kinect to classify mimics with maximum probability, or the Microsoft Avatar Kinect tool, imitating the user, including his/her facial expressions. Motion sensors like Kinect has a high potential of utilization in virtual reality, environment analysis, and – using a combination of sensors and state-of-the-art technology – even in health-care. It is used by therapists to help people with various physical disabilities who tend to lack enthusiasm for physical exercise. Games and other physical activities based on Kinect motivate patients to move more and have fun while doing exercises. Another well-known use is Adora[1], an operation assistant controlled by voice and hand gestures, enabling surgeons to access patient data. The Virtualrehab project uses Kinect or Leap Motion to track and capture movements of patients and allows them to play games. Patients can do physical exercises by playing games using Leap Motion – to train hands – or Kinect – to train their entire body. Such games focus on balance, coordination and posture of the patient and they use customized rehabilitation programs to treat physical health problems. Kinteract software, utilizing motion-based games in the rehabilitation process, is described in [11]. The added value resides in providing a motion sensor server that supports a growing array of motion sensors and merges their data into a single protocol. Authors interconnect Kinect, Leap Motion and Orbotix Sphero devices[2]. In this combination of sensors, Orbotix Sphero is a hand-held durable" robotic ball", with a diameter less than 12cm and weight of 200g, communicating with devices compatible with iOS, Android and Windows. Sphero is a low-cost device to be controlled by other devices; however, it can also be used as a controller due to the powerful integrated IMU (Inertial Measurement Unit). Sphero can be used as a motion sensor thanks to its gyroscope and

[1] Adora homepage, http://adora-med.com/.

[2] Sphero homepage, http://www.sphero.com/.

accelerometer sensors. It is programmable within Lightning Lab, which allows programming Sphero with a compatible device.

2.1 Infrared Motion Sensors

Infrared technology is utilized within the Leap Motion sensor to track one or both hands in a fast and accurate fashion. It usually monitors space from the top of a desk, in a range of 25 mm and 600 mm. As a great advantage, it may be attached to virtual reality headsets, most often Oculus Rift or HTC Vive. The device recognizes simple gestures, hand movements and their location at 200 Hz. In order to get the relative location of the tracking point, Leap Motion utilizes frames, subsequently placing this point into the Cartesian coordinate system. It contains three infrared LEDs and two infrared cameras. When sensing the environment, it immediately sends coordinate data via USB and calculates the framed scene [12]. Since Leap Motion was released with an SDK (Software Development Kit), it is possible to merge it with the fields of view of other motion controllers, such as the Kinect – in case of the latter, it is up to 150×120°.

2.2 Sensor Combinations in Gaming Controllers

Kinect is a well-known motion controller created as a combination of several sensors in a single device. Microsoft unexpectedly stopped producing Kinect 2018 25th October and sold Apple's patent[3]. Apple has already integrated it into the iPhone X. The modified Kinect in this smartphone is placed in the upper ramp and uses it on FaceID. FaceID emits 30,000 infrared beams and monitors their return time to measure depth and recognize the user[4]. Kinect was produced in two versions: Kinect 360 and Kinect v2. Authors of [13] directed their attention to these sensors, confirming that Kinect v2 is more accurate and has better parameters in comparison with the first-generation sensor (see Table 1). Authors of [14] described an experiment leading to the conclusion that the main technical advantage of v2 over v1 was that it provided better resolution. This was achieved through distance measurement performed for each pixel of the captured depth maps, allowing more accurate detection of small objects and better color images [15]. In general, Kinect is a combination of an infrared depth camera, a color camera and a microphone array of four microphones. Thanks to the microphone array, Kinect can recognize the talking person even if there are more users in front of it and it can be controlled by voice as well. Kinect can sense up to six users together and it is able to recognize 26 joints per person.

[3] Microsoft kills Kinect, Stop manufacturing it, http://www.theverge.com/.

[4] Kinect is officially dead. Really. Offcially. It’s dead, http://www.polygon.com/.

Table 1
Comparison of Kinect sensors

	Kinect 360	**Kinect v2**
RGB camera (pixels)/(Hz)	1280×1024/15	1920×1080/15 or 30
Depth camera (pixels)	640×480	512×424
Min. depth (m)	0.8	0.5
Max. depth (m)	4	4
Tilt motor	Yes	No
Horizontal Field of View (°)	57	70
Vertical Field of View	43	60
Defined skeleton joints	20	26
Full tracked skeletons	2	6
USB version	2.0	3.0

Originally, Kinect was produced as part of the Xbox game console; however, following a massive demand from users, Microsoft developed Kinect for Windows [16]. Similarly, other controllers like Playstation Move or Nintendo Wii were produced, mostly to control games and other devices. Readers interested in virtual reality are advised to check the very innovative sensor combinations available in the Oculus Rift and Oculus Touch devices. Authors of [17] published an interactive virtual museum, combining Oculus Rift and Kinect, allowing hand gestures. In case of Oculus Touch, the user must hold a pair of controllers and push their buttons. In [18], Kinect and Nintendo Wii were compared, focusing on their possibilities for home use, as a rehabilitation tool. Participants tested the devices for ten weeks during their rehabilitation process and finally inclined to use Kinect for their future rehabilitation.

2.3 Wearable Motion Sensors

Sensors using infrared light to monitor the users ‘moves and gestures need direct visibility [19]. On the other hand, Myo Armband monitors forearm muscles with the help of eight electromyographs: the user puts this device on his/her dominant arm, similarly to a bracelet, right below the elbow. The muscle sensor cooperates with a nine-axis inertial measure unit, a three-axis accelerometer and a gyroscopic sensor. Then, on the skin surface, the device measures the EMG signal known as MUAP (Motor Unit Action Potential), created by multiple small muscle strings. Myo Armband monitors the skin surface at a frequency of 200 Hz [20], while the IMU works at 50 Hz. Myo Armband does not provide RAW EMG data since it only offers the user classified output. Data are processed by algorithms within Myo Armband. The classified output of Myo Armband shows which gesture corresponds to the sensed signal. Currently, the set of available gestures includes wave in/out, spread fingers, fist and hand in relaxed position. RAW data are available through the Myo Data Capture application. To its advantage, Myo

Armband is compatible with almost every platform, including MS Windows, iOS or Android from version 4.3. However, it can connect only to low energy Bluetooth 4.1, using a unique USB dongle. Currently, the official release does not provide support for Linux, though Myo community developers released PyoConnect to access Myo Armband devices. In [21], it was experimentally combined with Kinect. The authors claim that the connection was established without any issues. Myoware, similarly to Myo Armband, is a wearable sensor measuring EMG muscle signals [22]. This sensor has neither an accelerometer nor a gyroscope. However, it provides both standard and RAW EMG output. The lack of a case or cover makes it an attractive sensor; it provides various connection options, such as the Cable Shield for more cables, the Proto Shield for a prototype board, the Power Shield for batteries and the Mighty Master Shield. The Master Shield contains LEDs indicating the intensity of muscle workout. A great advantage of Myoware is that it can be easily combined with other sensors. Thus, it is possible to create special “costumes” reacting to muscle activity or it is possible to create a unique prosthesis. Moreover, Myoware is an Arduino-compatible sensor, providing many opportunities, e.g. to sense any muscle of the human body. On the other hand, Myoware needs a permanent connection via cable. The aim of the current research is to accomplish an accurate combination of low-cost sensors, requiring wireless as well as case-covered sensors. Therefore, we used Myo Armband to experimentally evaluate the programming engines described in this study.

2.4 Comparison of Selected Motion Sensors

In [23], the authors tested these three sensors in 250 applications and they analysed 15 common gestures. The main disadvantage of Myo Armband, compared to Leap Motion is that while Myo Armband enables a limited set of gestures, resulting in a more consistent use of gestures across applications. Leap Motion enables a wider range of gestures and hence it provides greater variability of gestures. Gestures involving fingers and hand movements are less commonly used in Microsoft Kinect application due to its current hand-tracking limitations. The experiment was not focused on testing security matters (e.g. intrusion detection) of the respective sensors [24]. We tested these sensors in our experiment [25] to evaluate the recognition effectiveness of gestures: pointing, waving, hand rotation, fist gesture and fist rotation. The results of the experiment are available in Figure 1, displayed by gestures and sensors. Figure 1 also shows the percentage of recognition accuracy for all sensors. The following sections discuss achieved results, recorded also in Figure 1. The graph includes percentage results on efficiency with five gestures performed by three sensors. In pointing measurements we tried to aim with the cursor in the application at the desired place and monitored whether by our hand movement the cursor reached the intended destination. We start with Leap Motion. We went through different applications whether it was a desktop cursor controller or a game in which we tried to select the desired button in the game menu. We aimed with our index finger directly above the Leap Motion sensor. The results were very pleasurable, the cursor almost each time stopped at the place we expected. With Myo Armband we had to move all our hand to point somewhere and often it was just too slow. Although the cursor hit the mark on 90% precision with Leap Motion, 79% with Myo Armband and 80% with Micorosoft Kinect. Similar results were achieved with Kinect and Myo Armband. One must use the whole arm to move the cursor, but it is meant to be so as Kinect scans the whole body recognizing the body joints. Nevertheless, aiming was uncouth and needed much effort same as with Armband. Accurately sensor in this category was without the doubt the Leap Motion with its 90% accuracy. Waving was different for each of the three devices. For Leap Motion we performed a motion with a hand very similar to petting an object. Leap Motion took the challenge very well, our hand gesture actions were followed by the expected reactions. Since Myo Armband scans the muscles, it does not recognize the movement of the gestures, rather their starting and ending position. These two gestures may seem dull, but they work very well for their purpose. For Kinect, we took the whole arm waving into account once watching the bone recognition viewing the body joints, then different crates punching activities with our arms. The results were ample, with no big deviations. We liked the performance of Kinect in waving category the most. Kinect and Leap Motion achived 95% success and Myo Armband only 90%. We took hand rotation movement as turning one's arm around the arm's own axis. For Leap Motion, It showed just a negligible error. With Myo Armband, the hand rotation movement was pretty nice. It noticed even the smallest changes of rotation and reacted accordingly. So Leap Motion and Myo Armband achieve 98% and 97% success. For Kinect there were only a few applications using hand rotation, so we focused at bone recognition with this gesture and watched whether the body joints of the arm are moving accordingly. With a maximal deviation 10%, they copied our movements. For this gesture, we decided to recommend Myo Armband. Fist gesture for fist recognition we decided not only to scan the process of making and ceasing the fist gesture but also holding the gesture and performing arm motion. It is one of the most used gestures within each of the three devices. Leap Motion offers very accurate finger detection. It sensed the closed fist or in other words the absence of the fingers very well, although sometimes it showed one or more fingers spread apart. For Myo Armband, motion recognition was not such a big problem, since the band tracks the muscles. The problem was if one was already too comfortable with a fist holding they sometimes accidentally released the grip for a moment. Armband calibration plays a big role here as we all have different sized arms. Kinect applied fist gesture mainly on holding things, similarly as Leap Motion, uses the grab gesture. Despite some finger leaks we selected the Leap Motion as the best device for fist gesture recognition. After a long software procedure and after 50 measurements for three devices, we finally step into the result evaluation. Changing the results into percentage values we made the final graphs as we state them in Figure 1. The results are pretty straightforward, Leap

Motion showed the best results. When waving is concerned the results were similar, we selected Kinect for a subjective reasons. Although Leap Motion had slightly better measured results in hand rotation, we selected Armband as the best due to its comfortability. In fist recognition both passive and active, Leap Motion provided the best results. As we can see each device is satisfactory in different matters. When taking the cursor controlling in front of the computer, the Leap Motion is the clear answer. But when controlling the cursor from distance, like it may be at some presentations, one should rather choose, Armband or Kinect. As waving has different purposes and different ways of performance through each device, we cannot tell which one is the best. When hand rotating the best choice is Myo Armband as it is very reliable and at the same time, one can just have their arm hanged next to their body. If we want to make fist recognition, again we watch the distance from the computer. Leap Motion seems as the best but only in close distances.

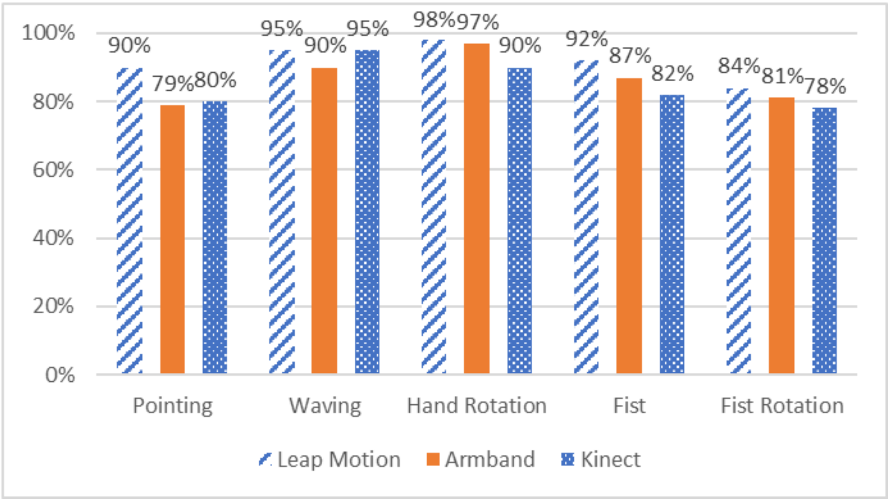


Figure 1
Summary of sensor efficiency

3 Unity3D Engine

In this study, we created simple programs interacting with three sensors: Kinect, Leap Motion, and Myo Armband. In general, we either created software using the official SDKs released with the respective sensors by their manufacturers (usually published with newer firmware versions) or we used game engines. Manufacturers and third-party developers produce drivers and plug-ins enabling creation of programs within popular game engines. Their benefits as well as their limitations are described in the following chapter. As far as game engines are concerned, there are three key factors to be considered by programmers[5]. The first one is usability – i.e. mainly its user friendliness. The second is functionality, defining the exact capabilities of the particular engine. The last key factor is price. In this case, we should take into account what platform the final solution utilizes. Unity3D Engine[6] and Unreal Engine[7] are game-based programming environments currently dominating the market, which includes other engines like Frostbite, CryEngine or Source. In the following chapters, we evaluate these two engines using simple algorithms, all operating with three motion controllers: Kinect, Leap Motion, and Myo Armband. Authors [26] developed via Unity in cooperation with Kinect and Arduino The Robot Engine. Computer tracks and simulate user's gestures via Kinect sensor, recognizes users voice commands and accordingly to the command performs a required action. A survey performed by TNW Deals[8] showed that almost half of game developers focus primarily on Unity3D. Contrary to C++, which is preferred by most of other engines, this engine works in C#, providing better scripting features, e.g. to create a game world, as well as advanced programming aspects [27]. Moreover, it provides UnityScript – its own scripting language similar to JavaScript – intended mainly for inexperienced programmers.

Unity3D is multiplatform; it supports 2D and 3D scenes, including virtual and augmented reality. The most popular games developed in this engine are Assassin's Creed, Deus Ex, Lara Croft etc. It offers thousands of free game assets. On the other hand, the free version of Unity3D does not provide Unity Profiler, which is in general intended for game optimizing. The price of the Unity Pro is $ 1500 for lifetime support or $ 75 for a month. Authors of [28] explained how they developed a game called Callory Battle AR. They developed two types of the game: one created without a 3D game engine and the second one with a free game engine from the Unity3D suite. They described their challenges with augmented reality issues as well. Authors of [29] used Unity3D to collect large amounts of customer data concerning their participation in games and they used clustering and visualization techniques. They described a prediction model based on the technology acceptance model to improve the sales performance of innovative products. Another use of Unity3D is described in [30], where the authors described how they developed a robotic platform to evaluate cooperative bilateral telerehabilitation approaches. The main goal was to evaluate the stability and performance of the force reflection strategy in the cooperative bilateral

[5] Gamesparks: Game Engine Analysis and Comparison, http://www.gamesparks.com/blog/game-engine-analysis-and-comparison/.

[6] Unity 3D homepage, https://Unity3D.com/.

[7] Unreal Engine homepage, https://www.unrealengine.com/.

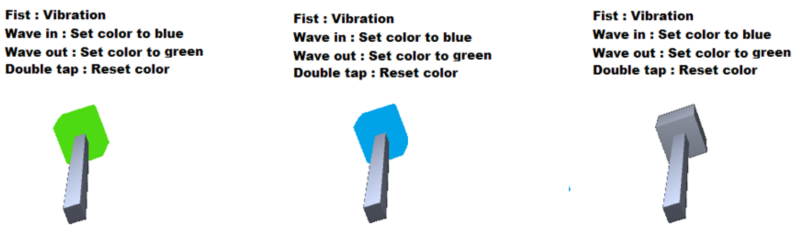
[8] TNW Deals: This engine is dominating the gaming industry right now, http://thenextweb.com/gaming/2016/03/24/engine-dominating-gaming-industry-right-now.

configuration and three robotic teleoperation techniques. In the following sections we describe the experience gained during the integration of three motion sensors with Unity3D. Step by step, we downloaded all drivers released by the sensor manufacturers and performed the correct installation of the employed sensors.

3.1 Cube Movement and Color Change using Myo Armband

Before creating a project for Myo Armband in Unity3D, it is important to import the package containing the SDK, available for download at the official website. Since every project using the engine requires the addition of a separate plug-in, this step is necessary as well. Then, an abstract camera – sensing moves from Myo Armband – and the Sun – representing natural light in the environment – appear on the screen. Our scenario involves a simple push of the right mouse button, resulting in the addition of a cube. Although writing a C# script is necessary (e.g. in Visual Studio), Unity3D compiles it by default and discovers the utilized objects.

In our case, we use an input script to move the cube by means of a motion controller, the Myo Armband – support for this is built into the system. The script has a simple behavior: following a fist gesture, the Myo Armband vibrates. The cube (visible on the screen) represents the end of a simple “hand” following the movements. Following a wave in/out gesture, the color of the cube changes from green to blue. The default color can be set by a double tap, as shown in Figure 2. The library of Myo Armband includes gestures like fist, wave in/out, spread fingers, or relax.



Fist : Vibration
Wave in : Set color to blue
Wave out : Set color to green
Double tap : Reset color

Fist : Vibration
Wave in : Set color to blue
Wave out : Set color to green
Double tap : Reset color

Fist : Vibration
Wave in : Set color to blue
Wave out : Set color to green
Double tap : Reset color

Figure 2
Myo Armband program in Unity3D

3.2 Cube Movement and Color Change through Kinect

Since the Kinect plug-in is not compatible with the newest version of Unity3D, our first issue was to find and install a compatible version of the engine. After connecting Kinect to a USB port, creating a new project in C# is simple. Again, the addition of the SDK is required, as well as importing the package and input plug-ins. An existing script can be edited in Visual Studio, containing two public void functions: start and update. The start function works only with a starting script – it can be used for setting parameters and values of variables. The update function is called every 10 milliseconds, as fast as Kinect senses the scene. After dealing with hand scanning and movement, we apply the data to the cube object again. In Figure 3 is shown application which copy users right hand movement. Box change color when user moves his hand to the left to red color, to the right to green color and when user hold his hand still, box is white.

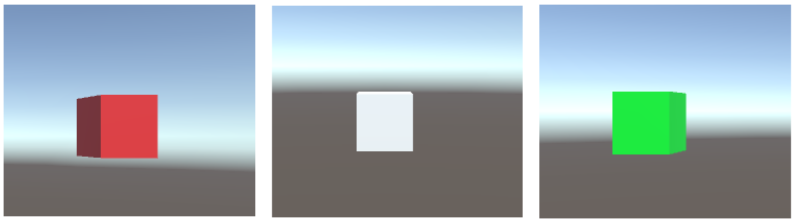


Figure 3
Kinect program in Unity 3D

3.3 Cube Movement and Color Change through Leap Motion

Similarly, to Myo Armband and Kinect, the official Leap Motion SDK contains a Unity3D plug-in, which we downloaded and imported into the compatible version of the engine and the project directory. However, here the engine encountered a problem since gestures have been removed from the library. Again, the script included two public void functions: start and update. The first one worked only with a starting script, which means it can be used for a controller which, however, Leap Motion does not contain. Therefore, we added a Controller class to call particular functions and switching the controller on/off. The update function operates every 10 milliseconds, as fast as Leap Motion senses a scene and turns it into a frame. Then, the script for hand scanning and movement is put into the cube object and set within graphic environment. The resulting script sets the cube color to green, if user moves his/her hand to the left. Movement to the right changes the cube color to red. It is important to create materials having different parameters, e.g. color or a material (see Figure 4).

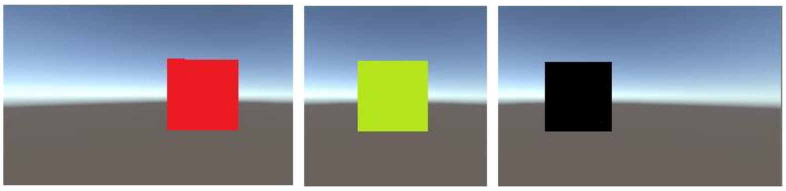


Figure 4
Leap Motion test in Unity 3D

3.4 Unity3D Engine Conclusion

We made three simple applications for motion sensors in Unity3D Engine and after that we can conclude Unity3D Engine is usable C# based programming environment for this purpose. We found huge support and libraries on the internet for Leap Motion and Myo Armband. Kinect has limited support of this environment and it was hardwork to find supported plugins. Work with C# and UnityScript is for programmer very intuitive and we can recommend this environment to work mainly with Myo Armband thanks to greater support and actual plugins.

4 Unreal Game Engine

As already stated, Unity3D covers almost half of the gaming industry software. On the other hand, there is a number of popular, high quality games created within another game engine: Unreal. Such games include Alone in The Dark or Tekken. A major difference of the two competing engines is programming language: Unity3D utilizes both C# and UnityScript (similar to JavaScript), while Unreal utilizes C++ or a graphic programming environment. This visual scripting environment utilizes components called Blueprints, claimed to be very user-friendly[9]. Authors of [31] described their experiment with six different popular game engines and compared them from many points of view, e.g. supported platforms, language support, physics engine or forward/backward compatibility of particular engines. They compared GPU and CPU usage of the games created using the respective engines. The results revealed that the Unreal engine was one of the most popular engines among visual programmers. Unreal Engine 4 has different pricing structure than Unity3D 4. It costs $ 19 per month plus 5% anytime you have earned some revenue.

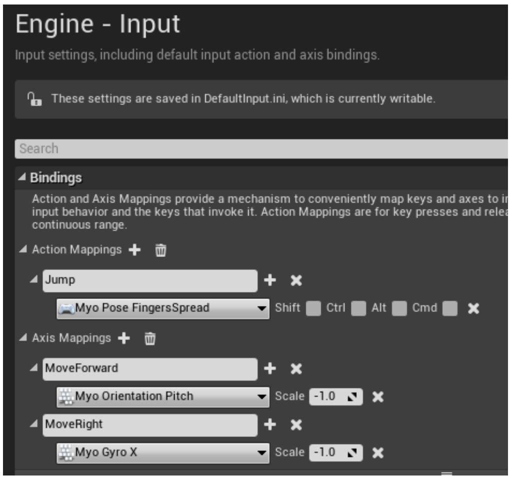
4.1 Orb Jump through Myo Armband

When creating a new project for Myo Armband within the Unreal engine, first it is necessary to import the binaries and the Kinect 4 Unreal plug-in[10]. Myo Armband can be used as a controller only if the input mapping is set. After performing these settings, we attached a character to a PhysicsBallBp blueprint. To receive Myo events, an interface is necessary. Then, MyoComponent was added to the Component. Figure 5 depicts our settings saved to DefaultInput. Next, MyoInterface was added, allowing the performance of poses like fingersSpread,

[9] Pluralsight homepage, https://www.pluralsight.com/.

[10] Myo Developer blog, http://developerblog.myo.com/.

doubleTap, unknown, rest and waves. Our choice was to use “fist” to make a jump, as shown in Figure 6. After we played the scene, it was possible to control the orb’s movements in the virtual world. By performing a fist gesture, it was possible to make the orb jump, as shown in Figure 7.



Engine - Input

Input settings, including default input action and axis bindings.

These settings are saved in DefaultInput.ini, which is currently writable.

Search

Bindings

Action and Axis Mappings provide a mechanism to conveniently map keys and axes to i
input behavior and the keys that invoke it. Action Mappings are for key presses and relea
continuous range.

Action Mappings

Jump

Myo Pose FingersSpread Shift Ctrl Alt Cmd

Axis Mappings

MoveForward

Myo Orientation Pitch Scale -1.0

MoveRight

Myo Gyro X Scale -1.0

Figure 5
Input setting for Myo Armband in Unreal

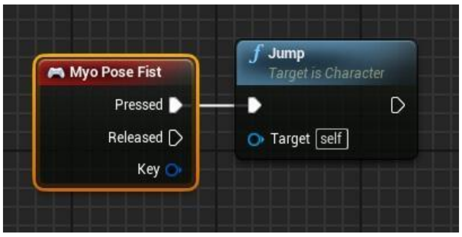


Figure 6
Blueprint for orb jump through Myo Armband

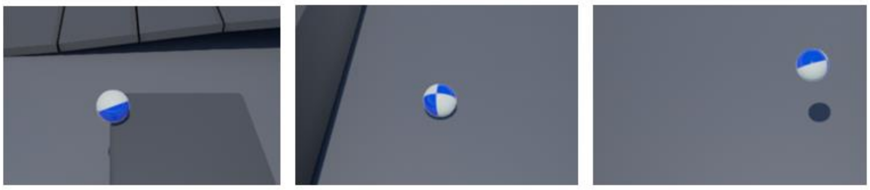


Figure 7
Orb jump controlled by Myo Armband

4.2 Orb Movement through Kinect

Again, the Kinect 4 Unreal plug-in must be imported. We selected a plug-in with API documentation, which is a primary interface to a component-based system[11]. This should be activated through the plug-in menu. Part of the plug-in is developed within the Introduction environment, where we were able to test functionality of Kinect in Unreal. Figure 8 depicts an image from IR and RGB Kinect cameras.

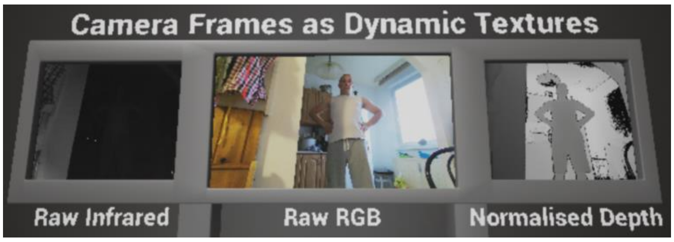


Figure 8
Kinect 4 Unreal Introduction test

Kinect tracking data is exposed via the Blueprint interface. Integration of Kinect with the Unreal engine requires the same steps as with Myo Armband or Leap Motion. First, the plug-in has to be downloaded into the root directory. The last necessary step is the configuration of the controller. In our experiment, we were able to sense our moving hands, as shown in Figure 9.

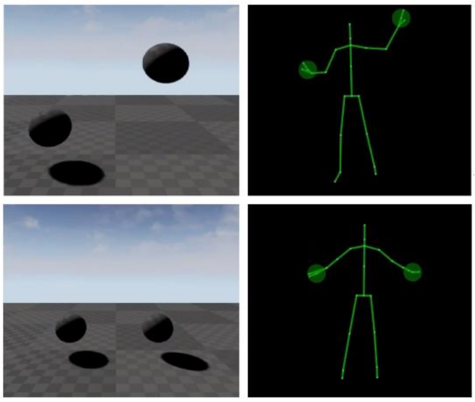


Figure 9
Kinect in Unreal Engine

[11] Opaque homepage, http://www.opaque.media/.

4.3 Object Movement through Leap Motion

Similarly to the other two controllers, binaries and plug-ins – downloadable from an official web page as part of the official plug-in Leap Motion for Unreal Engine[12]-were added. After the first steps, we could see an environment depicted in Figure 10. Convenience Rigged Characters are automatically included since version 0.9 has plug-in content. So, in order to use these characters, we changed our game mode to `LeapRiggedCharacter` or `LeapFloatingHandsCharacter` as a default mode. We modified our pass through character by changing collision preset to `PhysicsActor`. The `BasicBody_Physics Viewport` allows modification of collision shapes colliding with the world. Here, the `LeapBasicRiggedCharacter` was used, being the only one skeletal mesh available for a non-virtual reality environment. Using this character allowed us to modify any aspect of rigging. After a few steps, it was possible to test Leap Motion's features from many views, e.g. we tried to grab objects, move them from one place to another, or shake them in space. We made 50 tries and 45 of them were successful that means a 90% success rate.

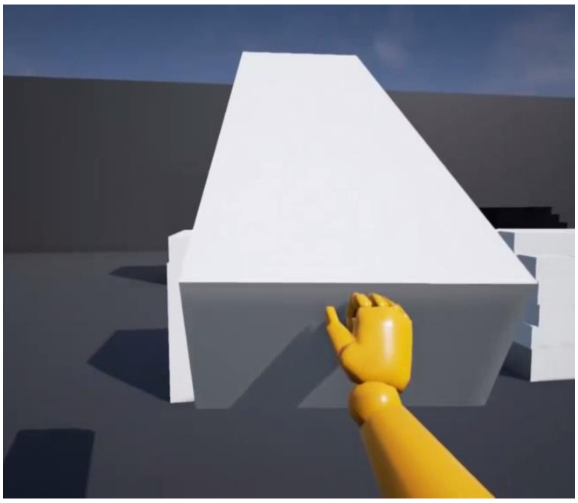


Figure 10
Leap Motion in Unreal Engine

4.4 Unreal Game Engine Conclusion

In Unreal Game Engine we prepared three environments for three different motion sensors. It is not programming as in Unity3D Engine. In Unreal Engine, work with

[12] Leap Motion Developer page, https://developer.LeapMotion.com/.

motion sensors is about setting sensors as controllers and just setting up the environment. Compared to Unity3D Engine environments that we tested we completed in 50% of the time thanks to using blueprints. Thanks to blueprints, actual plugins and better license terms we encourage using Unreal Game Engine against Unity 3D.

5 Low-Cost Sensors in LabVIEW

To create software for low-cost sensors like Kinect, Myo Armband or Leap Motion, there is a wide variety of possibilities. In addition to official SDKs or the already demonstrated game engines like Unity3D and Unreal, many other visual programming environments are available [32]. One of these is LabVIEW, produced by National Instruments, aimed at the development of engineering applications. It works with many hardware targets and programming languages, utilizing plug-ins or toolkits. In this environment, programmers can count on support for third-party devices as well as on many open-source toolkits and useful add-ons. The LabVIEW environment workspace contains two main parts – Front Panel and Block Diagram. The Block Diagram contains the graphical source code of the program, while the Front Panel shows graphical output of the running program. Objects are added using the Functions Pallete that automatically appears by right-clicking the block diagram workspace. Unlike Unity3D or Unreal, LabVIEW is not free. This is a great disadvantage in case of low-cost programming, even though it offers many possibilities. Nevertheless, we will demonstrate low-cost sensor utilization in LabVIEW as an alternative way of program development for industrial, or robotics applications, aimed mainly at small and medium enterprises [33] and for Smart Cities. The trial version and student license are free; however, the LabVIEW 2016 for Analysing and Signal Processing version costs more than 3500 €[13].

5.1 Myo Armband in LabVIEW

Authors of [34] described recognition of six elementary hand movements in LabVIEW by sensing RAW EMG data of Myo Armband, while authors of [35] used them to control wheelchair motion. At the National Instruments forum[14] and student competition, a student created an intuitively controlled prosthetic hand prototype, identifying and mimicking hand gestures. The cost of this prosthetic hand, utilizing Myo Armband with LabVIEW, was less than £ 300. National

[13] National Instruments homepage, http://www.ni.com/.

[14] National Instruments forum, http://forums.ni.com/t5/LabVIEW-Student-Design/Robotic-Hand-Control-Through-EMG-Classification/ta-p/3538008.

Instruments created a forum called the LabVIEW MakerHub[15] for developers working with the LabVIEW environment, with a lot of helpful third-party content. In order to interconnect Myo Armband with LabVIEW, we downloaded Myo UDP to LabVIEW – this enables LabVIEW to work with an UDP stream of data sensed by Myo Armband's EMG sensors, as shown in Figure 11.

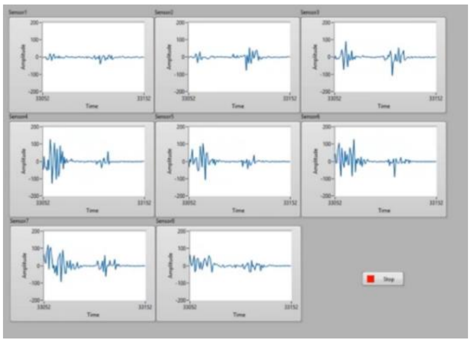


Figure 11
Front Panel for Myo Armband with the Functions Palette open in LabVIEW

5.2 Leap Motion in LabVIEW

The LabVIEW MakerHub interface for Leap Motion is a free open-source LabVIEW add-on, which makes it easy to use Leap Motion to track hand and fingertip positions with sub-millimeter accuracy, to get velocity and acceleration vectors and to recognize gestures like swipes, taps and circles. This interface made it possible to read all RAW data from Leap Motion and to use them in block diagrams. See Figure 12 and Figure 13 for an example of this – measurement of hand velocity.

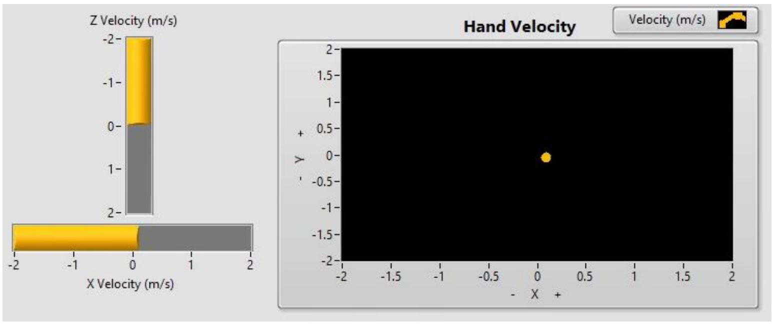


Figure 12
Front Panel for Myo Armband with the Functions Palette open in LabVIEW

15 LabVIEW MakerHub homepage, https://www.LabVIEWmakerhub.com/.

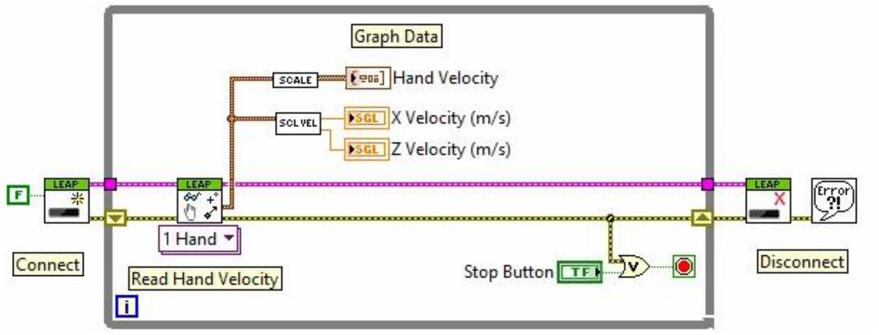


Figure 13
Block Diagram for Myo Armband with the Functions Palette open in LabVIEW

5.2 Kinect in LabVIEW

In order to utilize Kinect within LabVIEW, again, it is required to install the Kinect plug-in from MakerHub or just to use the built-in Kinect API. All Kinect possibilities can be utilized by developers to create block diagrams such as color, depth, skeletal or infrared video streams, as shown in Figure 14.

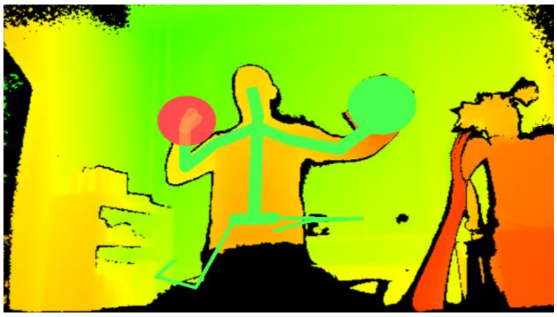


Figure 14
View on RGB-D map from LabView included Hand Gesture recognition

5.3 LabVIEW Conclusion

Work in Labview programming environment was different when compared with Unity3D and Unreal Game Engine mainly thanks to block diagram and front panel. Similar to Unreal Game Engine we spend only 50% of the time to prepare a fully functional application able to sense users movements and gestures. In Labview there are no options to prepare game environments and game applications but after our experience with Labview we decided to use it for our next experiments with motion sensors aimed toward industry and Smart Cities.

Conclusion

We presented three low-cost sensors, which we used in three different programming environments. Kinect and Leap Motion are infrared light sensing depth sensors, while Myo Armband is a sensor sensing EMG skin surface signals. These three sensors are fully compatible with both Unity3D and Unreal engines, mostly used for entertainment and gaming. The sensors are also compatible with the LabVIEW programming environment, mostly utilized for robotics and industry.

However, the free version of LabVIEW may not be used for our purposes of low-cost sensor combination, disqualifying it from such a challenge. However, in the paid (expensive) version, both its usage and software development are rather easy. We made an experiment showing how the three motion sensors work within the three programming environments. In Unity3D, we used C#, in Unreal we used the Blueprints workspace, while in LabVIEW we used the standard front panel and block diagram workspace. In these environments, working with the sensors was comfortable and highly accurate. However, to create a low-cost sensor combination, Unreal seems to be the best choice. Both Unity3D and Unreal Engine support virtual reality and they are compatible with 2D as well as 3D and 2D/3D worlds. Blueprints may be easily used with the Unreal engine, making this environment the most appropriate. The Unreal engine is the most user-friendly, with a wide range of capabilities and useful extensions. Even though it is said to be a game engine, it is a powerful programming environment. At the time we wrote this article, we thought Kinect would be ideal for our research needs.

We were considering whether Leap Motion and Myo Armband would be suitable for our research. We are focusing on sensors that are used by both, households and industry. They could be part of intelligent homes, serve as a controller for controlling non-contact computer games, and so on. They could be helpful in operations and save lives like Adora and Virtualrehab. Therefore, their reliability is important. From an industrial point of view, their consumption is also important. Various development environments are used to integrate these sensors in different areas. For playing games, is the best way to use Unity 3D and Unreal Engine, and is advisable to use LabView for sensor networks development or to implement is Smart City. After our experiments we can say that Kinect is suitable for controlling games and creating simple Smart Home soloutions. We do not consider Kinect suitable for Smart City and industrial purposes.

Acknowledgement

This work was supported by KEGA Agency of the Ministry of Education, Science, Research and Sport of the Slovak Republic under Grant No. 077TUKE-4/2015 „Promoting the interconnection of Computer and Software Engineering using the KPIkit“.

This work was supported by KEGA Agency of the Ministry of Education, Science, Research and Sport of the Slovak Republic under Grant No. 003TUKE-4/2017 Implementation of Modern Methods and Education Forms in the Area of Security of Information and Communication Technologies towards Requirements of Labour Market. This support is very gratefully acknowledged.

References

[1] Craig A. and Krishnan S., “Fusion of Leap Motion and Kinect Sensor for Impreved Field of View and Accuracy for VR Applications” In: 2016, Stanford EE26, [online] [quoted: 29.6.2018], Available at: http://stanford.edu/class/ee267/Spring2016/report_craig_krishnan.pdf

[2] Kainz, O. et al., “Low-cost Assistive Device for Hand Gesture Recognition using sEMG”, Proceedings of SPIE, Bellingham 2016, pp. 1-7

[3] Nan X., Zhang Z., Zhang N., Guo F., He Y., Guan L., “VDesing: Toward Image Segmentation and Composition in Cave Using Finger Interactions”, IN: 2013 IEEE China Summit & International Conference, DOI: 10.1109/ChinaSIP.2013.6625382, ISBN: 978-1-4799-1043-4

[4] Szabó C., Korečko Š., Sobota B., “Data Processing for Virtual Reality,” In: Advances in Robotics and Virtual Reality: Intelligent Systems Reference Library: Volume 26. - Berlin Heidelberg: Springer-Verlag, 2012, pp. 333-361, ISBN 978-3-642-23362-3 - ISSN 1868-4394

[5] Vokorokos L., Hartinger M., Ádám N., Chovancová E., Radušovský J., “Increasing Efficiency of the Sequential Algorithms Programs Execution Using CUDA,” In: SAMI 2014: IEEE 12th International Symposium on Applied Machine Intelligence and Informatics : Proceedings : January 23-25, 2014, Herl'any, Slovakia. - Danvers: IEEE Computer Society, 2014, pp. 281-284, ISBN 978-1-4799-3441-6

[6] Lor Johan P., “International and Comparative Librarianship: A Thematic Approach”, 2014, ISBN 13: 9783110268003

[7] Vokorokos L., Mihaľov J. and Leščišin Ľ., “Possibilities of Depth Cameras and Ultra Wide Band Sensor”, In: SAMI 2016, IEEE, 2016, pp. 57-61, DOI: 10.1109/SAMI.2016.7422982, ISBN 978-1-4673-8739-2

[8] Penelle B., Debeir O., “Multi-Sensor Data Fusion for Hand Tracking using Kinect and Leap Motion”, In: VRIC '14 Proceedings of the 2014 Virtual Reality International Conference, Article No. 22, DOI: 10.1145/2617841.2620710

[9] Kopják J, Kovács, “Event driven software modeling for combinational logic networks based control programs” In: Szakál Anikó (szerk.), Proceedings of the 16th IEEE Conference International Conference on Intelligent Engineering System 2012. Lisszabon, Portugália, 2012.06.11-2012.06.13. Lisszabon: Institute of Electrical and Electronics Engineers (IEEE), 2012, pp. 253-257, ISBN: 978-1-4673-2695-7

[10] Marin. G., Dominio F., Zanuttigh P., "Hand gesture recognition with Leap Motion and Kinect devices", IN: Image Processing (ICIP), 2014 IEEE International Conference, DOI: 10.1109/ICIP.2014.7025313

[11] Matos N., Santos A., Vasconcelos A.,“ Kinteract: a multi-sensor physical rehabilitation solution based on interactive games", IN: PervasiveHealth'14 Proceedings of th 8th International Conference on pervasive Computing Technologies for Healthcare, pp. 350-353, DOI: 10.4108/icst.pervasivehealth.2014.255325

[12] Weichert F., Bachmann D., Rudak B., Fisseler D., 2013, "Analysis of the accuracy and robustness of the Leap Motion Controller.", IN: Sensors 2013, DOI: 10.3390/s130506380, ISSN 1424-8220

[13] Pagliari D., and Pinto L.,"Calibration of Kinect for Xbox One and Comparison between the Two Generations of Microsoft Sensors ", Sensors 2015, 27569-27589, ISSN 1424-8220, 2015. DOI:10.3390/s151127569

[14] Lachat E., Macher H., Mittet M. A., Landes T., Grussenmeyer P., "First Experiences with KinectV2 Sensor for Close Range 3d Modelling. " In Proceedings of the Conference on 3D VirtualReconstruction and Visualization of Complex Architectures, Avila, Spain, 25-27 February 2015; pp. 93-100, DOI: 10.5194/isprsarchives-XL-5-W4-93-2015

[15] Chovanec M., Bíly J., Chovancová E., Radušovský J., "Algorithms for User Detection and Authentication based on Face Analysis", In: ICETA 2013: 11th IEEE International Conference on Emerging eLearning Technologies and Applications, October 24-25, 2013, Stary Smokovec. - Danvers: IEEE, 2013, pp. 63-66, ISBN 978-1-4799-2161-4

[16] Kainz, O. et al., "Detection of Persons and Height Esatimation in Video Sequence“, International Journal of Engineering Sciences & Research Technology. Vol. 5, No. 3, 2016, pp. 603-609

[17] Pinto J., Dais P., Eliseu S., Santos B. S.,“ Interactive configurable virtual environment with Kinect navigation and interaction", IN: Sciences and Technologies of Interaction 2015, Online: 30.3.2017

[18] Beaulieu-Boire L., Belzile-Lachapelle S., Blanchette A., Desmarais P.O., Lamontagne-Montminy L., Tremblay C., Corriveau H., Tousignant M., "Balance Rehabilitation using Xbox Kinect among an Elderly Population: A Pilot Study" IN: Novel Physiotherapies 2015, DOI: http://doi.org/10.4172/2165-7025.1000261

[19] Zug S. et al. "Are laser scanners replaceable bz Kinect sensors in robotic applications?" IN: IEEE International Symposium on Robotic and Sensors Environments, 2012, DOI: http://doi.org/10.1109/ROSE.2012.6402619

[20] Mulling T., and Sathiyanarayanan M., 2015, "Characteristics of Hand Gesture Navigation: a case study using a wearable device (MYO)", IN: Proc. of the 29th British Human Computer Interaction (HCI), pp. 283-284,

ACM, July 2015, DOI: http://doi.org/10.1145/2783446.2783612, ISBN: 978-1-4503-3643-7

[21] Kainz O., Jakab F., "Approach to Hand Tracking and Gesture Recognition Based on Depth-Sensing Cameras and EMG Monitoring", IN: Acta Informatica Pragensia 3, 2014, 104-112, DOI: 10.18267/j.aip.38

[22] Jobes S. K., Bernier J. M., Dryer S. L., Douglas E. D., "Arm Mounted Exoskeleton to Mechanically Assist Activities of Daily Living", IN: American society for engineering education 2016 Conference, Online 30.3.2017

[23] Cabreira A. T. and Hwang F., "An Analysis of Mid-air Gestures Used Across Three Platforms," in Proceedings of the 2015 British HCI Conference, ser. British HCI '15. ACM, 2015, pp. 257-258, doi: 10.1145/2783446.2783599

[24] Ennert M., Chovancová E., Dudláková Z., "Testing of IDS model using several intrusion detection tools," In: Journal of Applied Mathematics and Computational Mechanics. Vol. 14, No. 1 (2015), pp. 55-62, ISSN 2353-0588

[25] Vokorokos L., Mihaľov J., Chovancová E., "Motion Sensors: Gesticulation Efficiency Across Multiple Platform", 20th Jubilee IEEE International conference on Intelligent engineering systems, pp 293-298, 2016, doi: 10.1109/INES.2016.7555139

[26] Bartneck, C., Soucy, M., Fleuret, K., & Sandoval, E. B. (2015). "The Robot Engine - Making The Unity 3D Game Engine Work For HRI ". Proceedings of the IEEE International Symposium on Robot and Human Interactive Communication, Kobe pp. 431-437, DOI: 10.1109/ROMAN.2015.7333561

[27] Jakab, F. et al.: Rich Media Delivery, Computer Science and Technology Research Survey (CSTRS), Vol. 3, 2008, pp. 31-36

[28] Kim L. S., Suk H. J., Kang J. H., Jung J. M., Laine T. H., Westlin J., "Using Unity3D to Facilitate Mobile Augmented Reality Game Development", IN: 2014 IEEE World Forum on Internet of Things, DOI: 10.1109/WF-IoT.2014.6803110

[29] György Györök, "Embedded hybrid Controller with programmable Analog Circuit", In: IEEE 14th International Conference on Intelligent Systems. Gran Canaria, Spain, 05/05/2010-07/05/2010. Gran Canaria: pp. 1-4, Paper 59, ISBN: 978-4244-7651-0

[30] Felipe A. Pires, Wilian M. Santos, Kleber de O. Andrade, Glauco A. P. Caurin, Adriano A. G. Siqueira, "Robotic Platform for Telerehabilitation Studies based on Unity Game Engine", In: Serious Games and Applications for Health (SeGAH), 2014 IEEE 3rd International Conference, 14-16 May 2014, DOI: 10.1109/SeGAH.2014.7067094, ISBN: 978-1-4799-4823-9

[31] Mishra P., Shrawankar U., "Comparison between Famous Game Engines and Eminent Games", IN: International Journal of Interactive Multimedia and Artificial Intelligence, 2016, ISSN: 1989-1660, DOI: 10.9781/ijimai.2016.4113

[32] Madoš B., Hurtuk J., Čajkovský M., Kudra E., "Visual programming tool for computer with data flow computation control" In: Acta Electrotechnica et Informatica, year. 14, č. 4 (2014), pp. 27-30, ISSN 1335-8243

[33] Chovancová E., Vokorokos L., Chovanec M., "Cloud computing system for small and medium corporations," In: SAMI 2015. - Danvers: IEEE, 2015 S. 171-174. - ISBN 978-147998221-9

[34] Navarro J.C., León-Vargas F., Pérez J.B., "EMG-Based System for basic hand movement recognition", IN: Dyna, year 79, pp. 41-49, Medellin, 2012, ISSN 0012-7553

[35] Sathish S., Nithyakalyani K., Vinurajkumar S., Vijayalakshmi C., Sivaraman J., "Control of Robotic Wheel Chair using EMG Signals for Paralysed Persons", IN: Indian Journalof Sience and Technology, Vol. 9, January 2016, DOI: 10.17485/ijst/2016/v9i1/85726, ISSN: 0974-6846