DOI: 10.5604/01.3001.0010.5212

A METHOD FOR VISUALIZATION OF 3D MOTION DATA USING A MOBILE DEVICE

Michał Oseńko[1], Jakub Smołka[2], Maria Skublewska-Paszkowska[2], Edyta Łukasik[2]
[1]Lublin University of Technology, Electrical Engineering and Computer Science Faculty, [2]Lublin University of Technology, Electrical Engineering and Computer Science Faculty, Institute of Computer Science

Abstract. *3D motion visualization is a topic closely connected with motion capture. Motion capture data come in many file formats, one of which is C3D. The following paper presents an innovative method of motion visualization in virtual reality using Google Cardboard. Motion data are read from C3D files. The paper presents introduction to the technology used, analysis of existing solutions, presentation of the proposed method, its implementation and results.*

Keywords: virtual reality, visualization, motion capture

METODA WIZUALIZACJI DANYCH RUCHU 3D WYKORZYSTUJĄCA URZĄDZENIE MOBILNE

Streszczenie. *Wizualizacja ruchu 3D jest zagadnieniem silnie związanym z przechwytywaniem ruchu. Dane wyjściowe przechwytywania posiadają wiele formatów, jednym z nich jest format C3D. Niniejszy artykuł przedstawia innowacyjną metodę wizualizacji danych ruchu, uzyskanych z plików C3D, w wirtualnej rzeczywistości, z wykorzystaniem Google Cardboard. Wiązać się to będzie z wprowadzeniem w użyte technologie, przeprowadzeniem analizy istniejących rozwiązań i przedstawieniem projektu, implementacji nowej metody oraz jej wyniku.*

Słowa kluczowe: wirtualna rzeczywistość, wizualizacja, akwizycja ruchu

Introduction

Virtual reality (VR) put simply is a simulation of the real world, based on tricking the senses of the user, who is placed in a state of a different reality. It involves presenting the user with three-dimensional graphics, where the display is dependent on his/her movements. This property is responsible for the sense of reality of experience [17]. Of the currently widely available technologies, it is virtual reality that best corresponds to the expectations posed by the problem of developing a method of visualization. It allows for an appropriate representation of 3D relationships [12]. One of the implementations of virtual reality is Google Cardboard [14]. It is a platform, as well as the name of goggles. The principle of operation of this technology is simple – the user just inserts a smartphone with an application that uses Google Cardboard into the goggles (made of cardboard or plastic) to experience virtual reality [9]. Visualization of motion is largely connected with the technology of motion capture, which is the digital recording of motion [5].

This article presents a new method for visualizing three-dimensional motion data on a smartphone, using the Google Cardboard technology. It is important to point out that the data read are stored in the C3D format. C3D (or Coordinate 3D) is a binary file format used in the recording of 3D motion, biomechanics, movement analysis laboratories, as well as animation [5].

1. Review of existing solutions

A thorough search has been conducted for methods of visualizing motion data using a mobile device, the Google Cardboard technology and C3D file format. No solutions have been found that meet the above criteria, but some meet them partially.

These solutions include:

- Analysis of gait in a virtual environment (GADV/VE) [1],
- Capturing movement and activity tracking by using a grid of smartphone sensors [14],
- Viewfinder [3],
- Mokka [2],
- C3D Motion Capture Viewer [15].

Some of them, the most important ones from the point of view of the article's topic, will be described below. Most attention is given to ways of data visualization and navigation within the application.

1.1. GAD/VE

The Gait analysis in a virtual environment system, as the only one among reviewed systems, includes the element of virtual reality. It is the prototype tool for the diagnostics of walking. In the assumption of the authors, it was supposed to be easy to use and intuitive for health professionals who do not have experience with such systems. The system accepts files in the C3D (Coordinate 3D) format as the input. Labels obtained from C3D files are used for the appropriate presentation of individual elements of the object [1].

The user interface design includes a toolbar (located at the bottom) and an animation panel (Fig. 1).

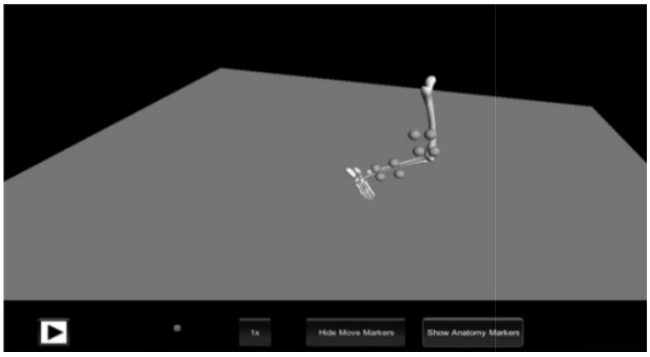

Fig. 1. User interface of the GAD/VE solution [1]

The different parts of the interface have been arranged in such a way that the animation panel takes up the most space. The toolbar consists of the following elements:

- play/pause button,
- keys for scrolling forwards and backwards,
- a button for choosing the playback speed,
- a button showing or hiding the marker labels,
- a button for showing markers' trajectories.

An important advantage of the GADV/VE is the possibility to use the user interface in three different ways:

- using remote manipulation of 3D data in a virtual reality environment, controlled by using gestures (motion capture), or mouse. Such a solution requires a projector or 3D TV, a fast computer, a head tracker with glasses, and additionally a wireless mouse or a motion capture system.
- using a stereoscopic display with 3D glasses, mouse and keyboard,
- using a standard monitor, mouse and keyboard for control [1].

The most effective way of using GADV/VE is to use the first method mentioned above. This would allow for appropriate preservation of the spatial relationships between markers, as well as very intuitive control of the object through body movement. This solution, however, requires very expensive and specialized equipment, which is difficult to obtain.

1.2. C3D Motion Capture Viewer

The C3D Motion Capture Viewer is a desktop, cross-platform application whose main purpose is to visualize the motion data from the binary file format C3D. It has an open source code, released under an MIT license in 2013 by Justin Stoecker [16]. Its implementation was made using the Java language and the OpenGL library. The user interface uses the Swing graphics library. The internal structure of the project is based on three packages:

- data structures (model),
- user interface (ui),
- data view (view).

It should be noted that the author of the C3D Motion Capture Viewer application developed his own mechanism for reading and writing C3D files. Figure 2 below presents a screenshot of the application.

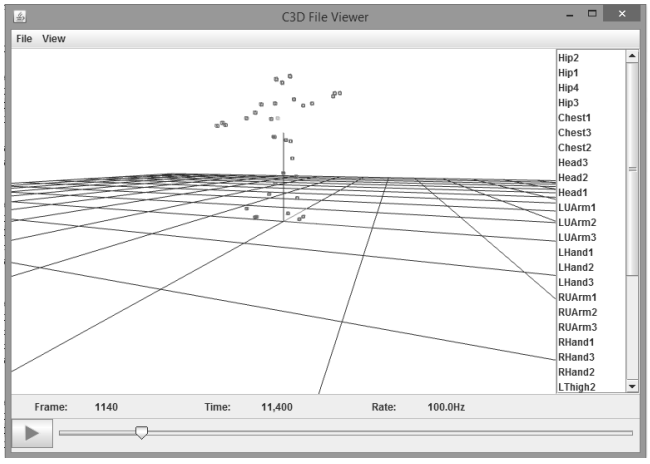

Fig. 2. User interface of the C3D Motion Capture Viewer application

The user interface consists of the following components:

- toolbar (at the top),
- animation panel (in the middle),
- marker label panel (right),
- animation control panel (bottom).

The animation control panel contains information labels (current frame number, current time in the recording, sampling frequency), the play/pause button, and the time scrollbar. In addition to the functions available directly from the user interface, the user has at his/her disposal features associated with the animation viewport. These features are controlled by using the keyboard and mouse. The A-D keys allow for viewport rotation in the vertical plane and the W-S keys in the horizontal plane. Moreover, the object can be scaled with the mouse wheel.

1.3. Mokka

Mokka (Motion kinematic & kinetic analyzer) is a multi-platform desktop application for biomechanical analysis of motion data. It allows one to read and write C3D files and many other formats. It is based on the VTK, BTKCore and Qt libraries [2]. The layout of the graphical interface is similar to that of the C3D Motion Capture Viewer presented earlier. At the top is the toolbar, on the right side the marker label panel, in the middle the play panel, and at the bottom the time scrollbar. It is the most robust application of those described so far. It allows for motion data interpretation in three ways: (1) using a three-dimensional view, (2) using a two-dimensional graph of the coordinate data or the so-called analog data, (3) using video view. It also features console logger for viewing messages coming from Mokka (Fig. 3) [2]. In addition to the standard options for visualization (which applications listed above have), it has many additional features. These, among other things, include: an extensive configuration abilities (size, marker color, marker visibility), ability of splitting visualization area into multiple parts, recording trimming (at the beginning as well as at the end). Viewport control for visualizing motion using a three-dimensional view is built around mouse actions [2].

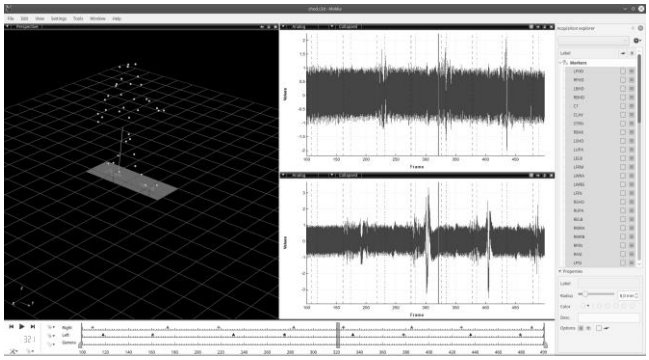

Fig. 3. User interface of the Mokka application

2. The new method of visualization

Analysis of the available solutions allowed us to gather important insights related to the motion visualization:

- the most important features of the application for visualizing motion data in C3D files are:
  - starting/pausing playback,
  - frame scrolling,
  - object rotation,
  - object scaling,
  - marker name presentation.
- for visualization of motion on a smartphone it is necessary to choose the appropriate methods of control and navigation in applications,
- the most reliable method of visualization is the one that clearly shows the spatial relationship between the markers,
- the cost of the solution must be acceptable to the average customer.

The method of visualizing motion data in C3D files utilizing Google Cardboard is a new and innovative method. The choice of the Google Cardboard technology has contributed to a shift in thinking about the user's interaction with the application. This technology provides only two ways of interaction: (1) clicking the button (magnet) on the goggles, (2) head movement tracking. As a result, it was necessary to introduce an additional peripheral device, a mouse, to get more ways of controlling the application. The mouse will be used for (1) file selection, (2) playback control, and (3) object scaling. The design of the proposed method in terms of functionality assumes fulfilment of the objectives stated above.

2.1. Presentation of the proposed method

The developed method of visualizing data was implemented as an application for the Android operating system. The application will be referred to as "C3D Cardboard Viewer" in the remainder of the paper in order to simplify the description. All screenshots bellow were taken from the application discussed. The equipment used for the presentation included: an LG Nexus 4 smartphone running the Android operating system (version 5), Weebo3D goggles (Fig. 4) and an A4TECH BT-630n Bluetooth Mouse. The visualized movement was loaded from an OptiTrack_2007.c3d file. All sample C3D files used for testing the visualization method were downloaded from [29].

Fig. 4. The weboo3d goggles [7]

The C3D Cardboard Viewer application, due to the lack of direct interaction with the screen, does not have the standard touch controlled file selection screen. The choice of a specific file is controlled by the mouse. The files must be located in the "C3D Cardboard Viewer" directory located on the phone's external storage. In absence of files to read the user will see the message "No files. Copy the .c3d files to the "C3D Cardboard Viewer"" (Fig. 5).

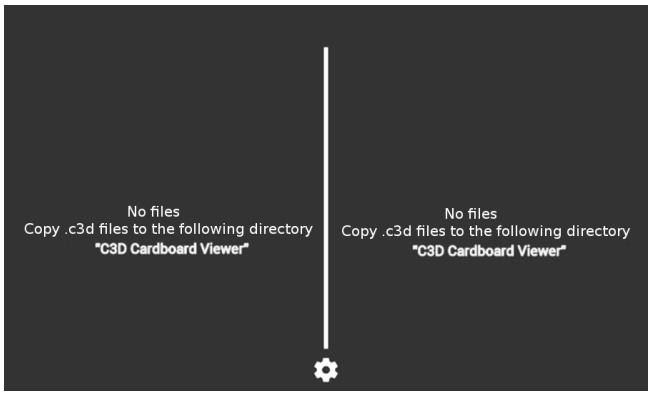

Fig. 5. C3D Cardboard Viewer – the first screen of the application is installed

After placing C3D files in the "C3D Cardboard Viewer" directory, the user can re-open the C3D Cardboard Viewer application to see that this time there is a list of C3D file names to choose from along with the message "File list loaded" (Fig. 6). In the case of loading a large number of files, an additional message may initially appear (Fig. 7). It is telling the user that the application is analyzing files in the directory. Loaded file names are displayed in the form of a vertical list where the name of currently selected file has blue font while the others use white font. To browse files the user turns the mouse wheel, and to load the selected file into the smartphone's memory either the goggles' button or the left mouse button.

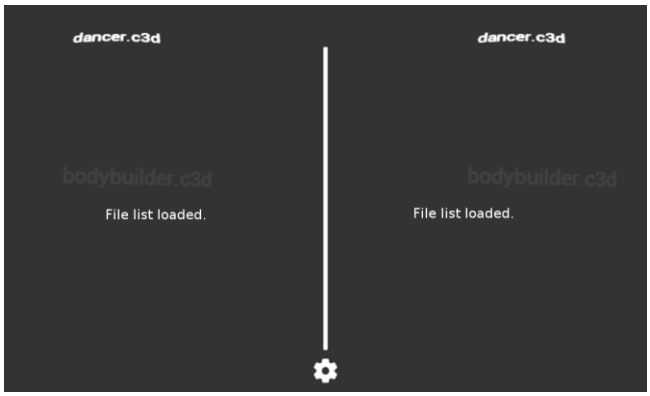

Fig. 6. C3D Cardboard Viewer – activity with file selection

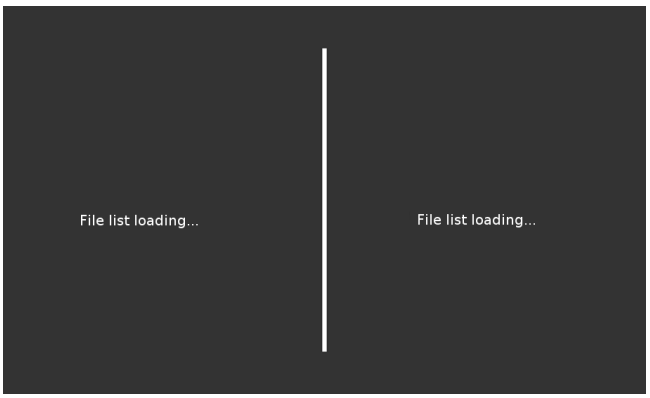

Fig. 7. C3D Cardboard Viewer – file list loading

After triggering the action of file loading, the screen displays "Loading data..." message (Fig. 8).

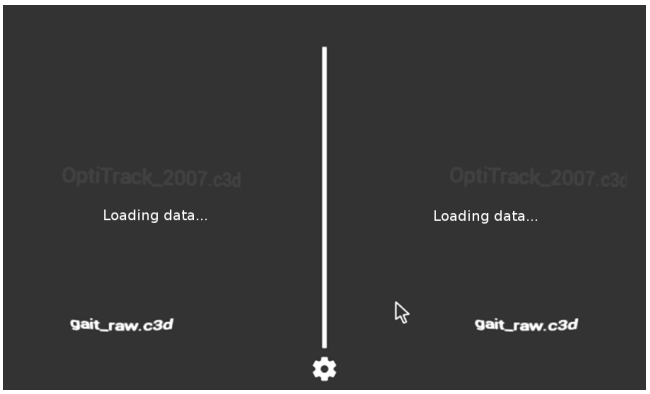

Fig. 8. C3D Cardboard Viewer – loading the file

Then the activity of motion visualization is opened with the markers displayed in the form of red dots (Fig. 9).

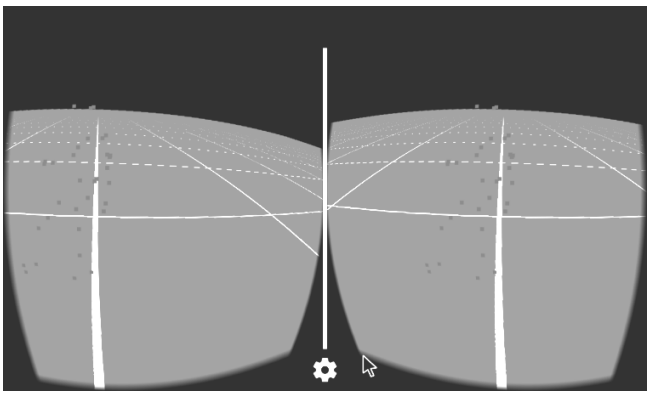

Fig. 9. C3D Cardboard Viewer – visualization activity; object displayed

As previously mentioned, this activity has two modes: (1) playback, which is the default mode of the visualization activity, and (2) the rotation and scaling mode. In playback mode, the movement of the user's head is associated with a rotation about the Y axis. This simulates the experience of the real world – a situation in which a person rotates around his/her own axis. This allows for better representation of spatial relationships between markers. In playback mode, the user has at his/her disposal five different options:

- starting the playback – by clicking the left mouse button,
- pausing the playback (only when it is in progress) – by clicking the left mouse button,
- stopping the playback (which also resets the current time) – by clicking the mouse wheel,
- scrolling visualization forwards – by turning the wheel forwards (up),
- scrolling visualization backwards – by turning the wheel backwards (down).

Loaded recording is played back by the C3D Cardboard Viewer application with the framerate read from the C3D file parameters.

Switching between playback and rotation/scaling modes is triggered by clicking the right mouse button. The user is informed about the currently selected mode by messages (e.g. "Mode: rotation, scaling"). Entering rotation/scaling mode is only possible when the playback is stopped. Unlike in playback mode, in scaling/rotation mode movement of the user's head in horizontal plane causes view point rotation around the object. This allows for looking at the object from all sides. Rotation of the object in relation to the state presented in Figure 9 is shown in Figure 10 (arrows indicate the direction of rotation).

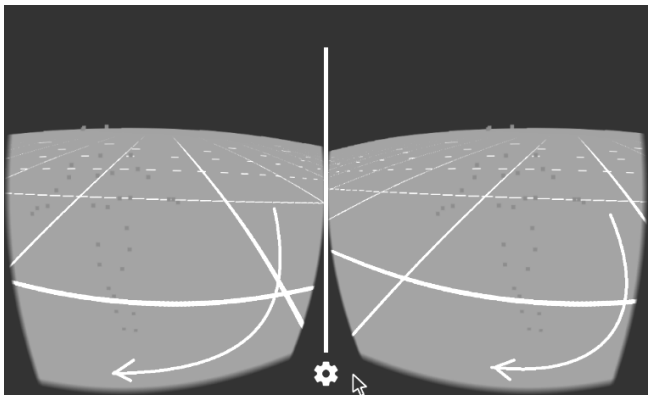

Fig. 10. C3D Cardboard Viewer – rotation around the object

Apart from rotation the user can scale the object (enlarging shown in Fig. 12 and reducing in size shown in Fig. 11). Scaling is triggered by scrolling the mouse wheel.

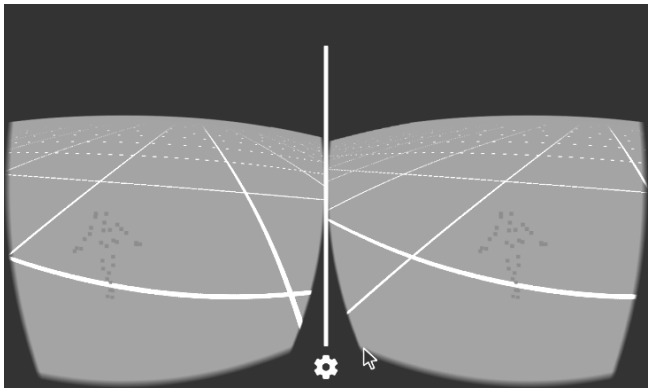

Fig. 11. C3D Cardboard Viewer – decreasing the object's size

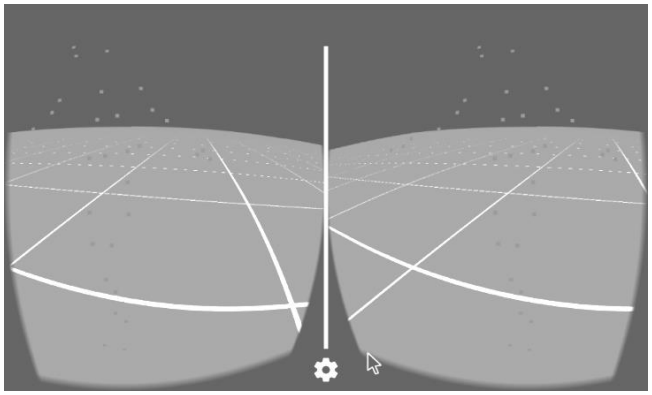

Fig. 12. C3D Cardboard Viewer – increasing the object's *size*

In order for the rotation mode to produce correct results the user needs to place the object in the center of his/her field of view prior to switching from playback to rotation/scaling mode. The correctness of scaling is dependent on the data obtained directly from the C3D file, namely the coordinates. In a situation where the coordinates constitute an object far away from the center of the world's coordinates, it may happen that from the user's perspective the object, instead of undergoing scaling, will move parallel to one of the axes. This follows from the definition of the scaling operation.

All of the above-presented manipulations performed on the object's coordinates, i.e. rotation and scaling, are not reset when the user switches again to playback mode. This way, one can watch playback of the object's motion from different directions and with a different scale factors. Return to playback mode is triggered by clicking the right mouse button.

2.2. Implementation of the proposed method

To implement the proposed method of visualization the following technologies were used:

- Android SDK,
- Cardboard for Google Android SDK (version for use with OpenGL ES 2),
- Gradle (for managing the project and its dependencies),
- Android Studio IDE.

After setting up the environment and collecting the necessary libraries, the C3DGoogleCardboardViewer project was created. Then the project structure was set up (Fig. 13).

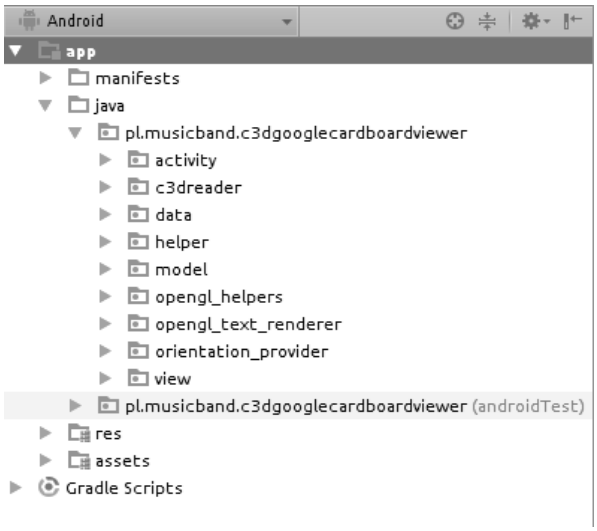

Fig. 13. C3DGoogleCardboardViewer – project tree with the view of individual packages

The primary package pl.musicband.c3dgooglecardboardviewer was divided into subpackages with respect to origin of the classes and their purpose. It consists of:

- activity – the most important part of the application, containing main activity classes: C3DFileBrowserActivity and C3DViewerActivity,
- c3dreader – this package contains classes for loading the C3D files into memory (taken from the C3D Motion Capture Viewer project [35]),
- data – the package containing the enumerations that define playback states, application modes, global constants, as well as the coordinates used for rendering the floor (taken from [10]),
- helper – this package contains auxiliary classes used for searching for files in the C3D format, rotating the camera around the object, reading the data from the C3D file, unit conversions, as well as for C3D file playback at a specified framerate (FPS),
- model – the package containing authors' own data structures:
  - PlayerBuffer – frame buffer model,
  - SimpleAngles3f – 3D angle,
  - SimpleVector3f – 3D vector,

- opengl_helpers – this package contains helper classes for use with OpenGL ES 2 (taken from [3]),
- opengl_text_renderer – this package contains classes for displaying text in OpenGL ES 2, used in the file selection activity (taken from [10]),
- orientation_provider – this package contains classes for reading the orientation sensors (taken from [13]),
- view – the package contains modified CardboardOverlayViewer class used for displaying text in activities such as CardboardActivity (taken from [10]).

Figure 14 shows component layout of the C3DViewerActivity.

activity_main.xml ×

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/ui_layout"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >

    <com.google.vrtoolkit.cardboard.CardboardView
        android:id="@+id/cardboard_view"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:layout_alignParentTop="true"
        android:layout_alignParentLeft="true" />

    <pl.musicband.c3dgooglecardboardviewer.view.CardboardOverlayView
        android:id="@+id/overlay"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true" />

</RelativeLayout>
```

Fig. 14. View layout in the C3DviewerActivity

A layout with the same structure was also used in the C3DFileBrowserActivity for file selection. Its main component is RelativeLayout which contains two views – CardboardView and CardboardOverlayView. CardboardView extends the GLSurfaceView class known from OpenGL ES implementation in Android. It renders the contents in a dual-channel way, with division into left and right eye channels. It is also responsible for distortion correction necessary for achieving the 3D effect [8]. CardboardOverlayView was already mentioned earlier, it is used for displaying messages on top of CardboardView.

Activities C3DFileBrowserActivity and C3DViewerActivity inherit from CardboardActivity – a base activity for Google Cardboard applications [8]. They implement the CardboardView.StereoRenderer interface. This interface is responsible for passing data required for rendering. It consists of the following methods: onSurfaceCreated(), onSurfaceChanged(), onNewFrame(), onDrawEye() and onRendererShutdown(). Most important of them are: onNewFrame() and onDrawEye(). The onNewFrame() method is invoked each time a new frame is rendered. A camera matrix is set in it this method. It is used for computing the view matrix. It is used for rotation around the object in rotation mode. As mentioned before the application has two modes:

- playback mode, in which the rotation of the head is changing the user's field of view (as in the real world),
- rotation and scaling mode, in which the rotation of the head rotates the point of view around the object (rotation around the Y axis).

There were at least two methods of realization of rotating the object:

- the camera (the user) rotates around the entire object – that is the camera moves on an arc of a circle,
- visualized object rotates around its own axis.

The first method was chosen for simplicity. The second method, in addition to object rotation, would require also the floor rotation. The implementation of the chosen solution consists in representing the rotation of the camera by an angle α around the object as a translation by x_p, z_p as well as camera's rotation around its axis by $-\alpha$ angle (Fig. 15). The assumption was introduced that before a rotation the camera is directly in front of the object.

Knowing (1) the distance between the camera and the object and (2) the rotation angle (obtained from the smartphone's motion sensor) allows for computing the translation. For this purpose trigonometric functions of an acute angle in a right triangle were used. Each quarter of the circle (marked with the Roman numerals in Figure 15) was considered separately. The formulas used for calculating the x_p, z_p values are shown in Table 1.

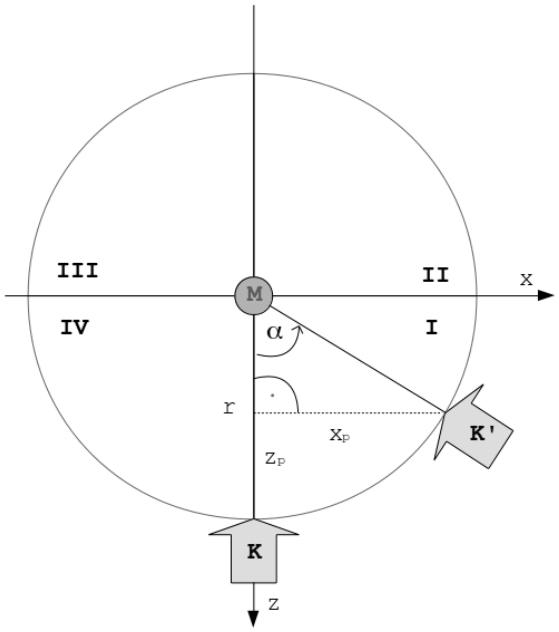

Fig. 15. Modelling the camera's (K) rotation around the object (M)

Table 1. Equations used for object rotation

quarter	Equations
I (0°-90°]	$x_p = r \cdot \sin\alpha$ $z_p = -r \cdot (1 - \cos\alpha)$
II (90°-180°]	$x_p = r \cdot \cos\alpha$ $z_p = -r \cdot (1 + \sin\alpha)$
III (180°-270°]	$x_p = -r \cdot \sin\alpha$ $z_p = -r \cdot (1 + \cos\alpha)$
IV (270°-360°]	$x_p = -r \cdot \cos\alpha$ $z_p = -r \cdot (1 - \sin\alpha)$

The OnDrawEye() method is called twice when a frame is rendered – separately for each eye. It is responsible for (1) setting the view and projection matrices (through modification of its eye parameter) and (2) rendering the object consisting of the markers and the floor. The aforementioned matrices play an important role in the entire rendering process. They allow for the separation of consecutive transforms. Their use is dictated by the convention adopted by OpenGL.

3. Summary

The assumptions associated with improved spatial vision using virtual reality have been met. The review existing solutions proves that the proposed method of 3D motion data visualization using Google Cardboard is innovative. It introduces a new quality and new possibilities in this field. It allows anyone with a smartphone, Google Cardboard compatible goggles and wireless mouse to use it. It provides basic functions such as reproduction of the object's motion, fast forward/rewinding, rotation around the object or its scaling. Navigation and control of the application mainly uses a mouse. The user needs to learn this new control method. However, this should not cause significant problems, since most of the methods of navigation and viewport control are very intuitive and were created on the basis of typical user's experience.

The method presented contains all the main functionalities related to the visualization of 3D motion. In the future it can be extended with the following features:

- shifting the object along the individual axes,
- spontaneous navigation to the location of the object (object tracking),
- visualization of bones in cases when the object is a human or an animal,
- improved layout of the file selection activity.

Adding new features would introduce more navigation and control options. Handling them in an appropriate and intuitive way would require using a peripheral device more powerful than a mouse. Such a device could be a game pad.

Bibliography

[1] Alfalah S.F.M., Chan W., Khan S., Falah J., Alfalah T., Harrison D.K., Charissis V.: Gait analysis data visualisation in virtual environment (GADV/VE), Science and Information Conference (SAI), 2014, 743–751.
[2] Barrea A., Armand S.: Biomechanical ToolKit, Open-source framework to visualize and process biomechanical data, Computer Methods and Programs in Biomedicine, Tom 114, Problem 1, 80–87.
[3] Brothaler K.: OpenGL ES 2 for Android, A Quick-Start Guide, The Pragmatic Bookshelf 2013.
[4] Martinsson J., Trost R.: Implementation of motion capture support in smartphones, Department of Computer Science and Engineering, Chalmers University of Technology, Göteborg, Sweden, 2010.
[5] Menache A.: Understanding motion capture for computer animation – second edition, Elsevier, 2011.
[6] Pascu T., White M., Patoli Z.: Motion capture and activity tracking using smartphone-driven body sensor networks, Third International Conference on Innovative Computing Technology (INTECH), 2013, 456–462.
[7] Domański J.: Weebo3d – drewniane VR do smartfonów, http://www.strefagracza.net/testujemy/peryferia/522-weebo3d-drewniane-vr-do-smartfonow.html [21.05.2016].
[8] Google: Cardboard – Getting started for Android SDK, https://developers.google.com/cardboard/android/get-started#build_the_sample_app [13.04.2016].
[9] Google: Describe of platform Google Cardboard, https://developers.google.com/cardboard [13.04.2016].
[10] Google: Sample program source Google Cardboard – „Treasure Hunt", https://github.com/googlesamples/cardboard-java [10.08.2015].
[11] Kodzhabashev A.: Rendering text in OpenGL on Android, https://github.com/d3alek/Texample2 [13.03.2016].
[12] Olszewski P.: Wszystko o wirtualnej rzeczywistości, Komputer Świat, http://www.komputerswiat.pl/centrum-wiedzy-konsumenta/gaming/wszystko-o-wirtualnej-rzeczywistosci.aspx [21.05.2016].
[13] Pacha A.: Sensor fusion demo for Android, https://github.com/apacha/sensor-fusion-demo [13.01.2016].
[14] Skałba Ł.: Google Cardboard, czyli wirtualna rzeczywistość za grosze – test i recenzja, http://lukaszskalba.komorkomania.pl/30256,google-cardboard-rzeczywistosc [20.05.2016].
[15] Stoecker J.: Projects overview, http://web.cs.miami.edu/home/jstoecker/projects/ [13.04.2016].
[16] Stoecker J.: Źródła programu „C3D Motion Viewer", https://github.com/jstoecker/c3dviewer [10.08.2015].
[17] Virtual Reality Society: What is virtual reality?, http://www.vrs.org.uk/virtual-reality/what-is-virtual-reality.html [13.04.2016].

B.Eng. Michał Oseńko
e-mail: michalosenko@gmail.com

Student of the second year of the Master of Engineering in Computer Science program, specializing in Web Applications, at the Lublin University of Technology, where he received a B.Eng. degree. His student activity includes software creation for the Android platform and developing programming skills in Java.

Ph.D. Jakub Smołka
e-mail: jakub.smolka@pollub.pl

Research worker at the Institute of Computer Science, Faculty of Electrical Engineering and Computer Science at the Lublin University of Technology. Earned his master's there, and his doctoral degree at the Silesian University of Technology. His research activity is in the area of motion data processing, mobile device applications, digital image processing and image compression.

Ph.D. Maria Skublewska-Paszkowska
e-mail: maria.paszkowska@pollub.pl

Member of the Institute of Computer Science, Faculty of Electrical Engineering and Computer Science at the Lublin University of Technology. Received her master's there and her doctoral degree at the Silesian University of Technology. Her research activities include: motion acquisition methods, 3D motion data analysis, wavelet transformations, picture quality measurement and its applications in adaptive image compression.

Ph.D. Edyta Łukasik
e-mail: e.lukasik@pollub.pl

Graduated from mathematics program at the University of Maria Skłodowska-Curie in Lublin. Received her Ph.D. from the Faculty of Mathematics, Physics and Computer Science there in 2007. Since 1998 member of the academic staff of the Lublin University of Technology. During 1998-2007 worked as an assistant, and since May 2007 as a research fellow. Her research interests are mainly algorithms, programming languages, data structures, numerical and optimization methods and mobile applications.

otrzymano/received: 13.10.2016 *przyjęto do druku/accepted: 14.08.2017*