Chapter 15
Introduction to Low-Cost Motion-Tracking for Virtual Rehabilitation

Sebastian Koenig, Aitor Ardanza, Camilo Cortes, Alessandro De Mauro and Belinda Lange

Abstract Low-cost motion sensors have seen tremendous increase in popularity in the past few years. Accelerometers, gyroscopes or cameras can be found in most available smart phones and gaming controllers. The Apple® iPhone, Nintendo® Wii™ and the PlayStatio® EyeToy™ are just a few examples where such technology is used to provide a more natural interaction for the user. Depth-sensing cameras by companies such as Microsoft, PrimeSense and Asus can enhance the user experience even further by enabling full-body interaction. This chapter will specifically discuss the use of the Microsoft® Kinect™ depth-sensing camera (Kinect) for rehabilitation of patients with motor disabilities. In addition, examples will be provided of how the Kinect can be used with off-the-shelf computer games or utilized in conjunction with modern game development tools such as the game engine Unity. The examples will outline concepts and required resources in order to enable the reader to use low-cost depth-sensing cameras for rehabilitation.

Keywords Video games · Middleware · Virtual Reality · Interactive technologies

S. Koenig (✉) · B. Lange
USC Institute for Creative Technologies, Los Angeles, CA, USA
e-mail: skoenig@ict.usc.edu

B. Lange
e-mail: lange@ict.usc.edu

A. Ardanza · A. De Mauro
Department of Vicomtech-IK4, eHealth and Biomedical Applications, San Sebastian, Spain
e-mail: aardanza@vicomtech.org

A. De Mauro
e-mail: ademauro@vicomtech.org

C. Cortes
Universidad EAFIT, Medellin, Colombia
e-mail: ccortes@vicomtech.org

J. L. Pons and D. Torricelli (eds.), *Emerging Therapies in Neurorehabilitation*, Biosystems & Biorobotics 4, DOI: 10.1007/978-3-642-38556-8_15,

15.1 Virtual Reality and Video Games for Rehabilitation

Virtual reality (VR) based rehabilitation of motor disorders is a very promising area of research and development. Virtual reality systems can range from high-cost platform systems and robotics to low-cost off-the-shelf video gaming technologies. There has been growing recognition of the potential value of VR and video game technology for creating a new generation of tools for advancing rehabilitation, training and exercise activities. Research in the area of VR and rehabilitation suggests that VR game-based technology can be used effectively to improve motor skill rehabilitation of a range of functional deficits (Boian et al. 2002; Chuang et al. 2002; Adamovich et al. 2004; Dvorkin et al. 2006; Fung et al. 2004; Fulk et al. 2005; Fung et al. 2006; Subramanian et al. 2007; Holden et al. 1999; Merians et al. 2002, 2009; You et al. 2005; Jack et al. 2001; Mirelman et al. 2009). Virtual Reality has also been integrated with robotics to provide motivation and feedback within rehabilitation with robotic devices that control and/or assist the user to move (De Mauro et al. 2011, 2012; Deutsch et al. 2004).

Part of the excitement in this area has grown from the well-known concept in motor learning that by providing a stimulating environment in which to practice repetitive, targeted movements with appropriate feedback can improve rehabilitation outcomes. Providing people with the opportunity to practice physical exercises within a digital game simulation has the potential to motivate the user to perform a higher number of repetitions in a more engaging environment (Rizzo and Kim 2005). Virtual reality systems demand focus and attention because the user is interacting within a situation that depends on their input. Game-based interactions can motivate the user to move and provide the user with a sense of achievement, even if they cannot perform that task in the 'real world'. Researchers have shown that the movements performed during VR rehabilitation can be similar to those used in traditional therapy, however, this is dependent on the specific VR system (Antonin et al. 2004).

The recent release of physically interactive video game systems has increased the interest and accessibility of the use of VR technologies within the rehabilitation setting. Some researchers have treated neurological impairments by implementing off-the-shelf game consoles, such as the Sony Playstation®2 EyeToy™ (Flynn et al. 2007; Rand et al. 2008) and Nintendo® Wii™ (Deutsch et al. 2008), with promising results. The underlying motion-sensing and 3D graphic technologies that are used in these commercial game systems allow the user to engage in entertaining motor games using gross body movements that are not bound by the limits of a mouse, joystick or game-pad interface. Yet whilst these systems have enjoyed wide adoption by millions of users and are generally stimulating and entertaining, clinicians and patients cannot easily alter the hard coded stimulus parameters of the game system as is needed to optimally rehabilitate precise motor skills in a controlled, clinically relevant way (Lange et al. 2009). In addition to the limited options for the systematic control of stimulus parameters needed to customize interaction challenges to the needs of the user, these video games provide

limited capacity for the recording of meaningful performance data (Lange et al. 2009, 2010). Therefore, while it is noted that these interactive video games are fun and motivating, such out-of-the-box systems do not consistently meet the clinical requirements for delivering precise motor interventions in a systematic fashion that can be customized to the needs of a target user group. However, the potential does exist that these systems can be creatively repurposed for useful rehabilitation purposes.

Many researchers are beginning to explore the potential of the Microsoft Kinect technology for rehabilitation. The solution to the challenge of using VR and video games in rehabilitation can be approached in two different ways: (1) the use of a middleware that allows tailored movements to be programmed so a patient can play an existing game with individualized movements or (2) the development of software specifically designed for the purpose of customized rehabilitation that is compatible for use with off-the-shelf interactive and video game hardware. This chapter will provide an overview of the tools that can be used to customize existing games, provide a tutorial to allow the reader to begin to explore the use of these tools, and provide a brief introduction to some of the tools that can be used to develop specific rehabilitation software.

15.2 Advances in Low-Cost Tracking Technology

Recent advances in video game technology have made available a large number of low-cost devices that can track the user's motion. These range in capability from handheld controllers that can be used for gesture-based control, such as the Nintendo® Wiimote™ and the Playstation® Move™, to cameras that use computer vision techniques to sense the user's body poses such as the Playstation® EyeToy™ and the Microsoft® Kinect™.

Depth-sensing cameras provide developers and clinicians with the most natural, but also the most flexible way to interact with rehabilitation applications. The user is not required to wear any markers, carry any additional devices or use specific limbs to interact with a system. Full-body tracking provides the opportunity to select any combination of gestures and limbs for interaction with a software application. Such flexibility can be used to tailor the user interaction towards the specific rehabilitation goals of the user (Lange et al. 2011a, b).

15.2.1 Depth-Sensing Frameworks

For the purpose of this chapter the Microsoft® Kinect™ has been used to demonstrate how depth-sensing cameras can be used with off-the-shelf computer games and customized applications within a game engine such as Unity. The Kinect was chosen because of its wide availability and the existing integration

within several development tools that facilitate its use in rehabilitation. There are multiple options for using the Kinect as an input device for games. Originally, the Kinect was released as a peripheral for the Xbox360 gaming console. However, Kinect-enabled games for the Xbox360 system do not give clinicians the freedom to choose gestures and movements that fit the therapeutic goals of the patient. Once the user input has been decided by the developer of the game, it cannot be adapted easily to fit the needs of users with disabilities. Alternatively, the Kinect can be used with a Windows PC through third-party software from OpenNI or the Kinect for Windows Software Developers Kit (SDK). Both options provide more freedom for leveraging the Kinect's tracking capabilities to control games for the purpose of rehabilitation and social reintegration of users with disabilities. The Kinect for Windows SDK enables developers to access the Kinect's data to enhance existing or newly developed games with gesture control and full-body tracking.

OpenNI™ (DotNetNuke Corporation) is an industry-led not-for-profit organization promoting the standardization of natural interaction devices. OpenNI provides an open-source framework for developers to leverage these devices for their own applications. The framework provides integration for the Microsoft® Kinect™ camera, PrimeSense camera and Asus Xtion Pro camera through an API for the sensor devices and an API for additional high-level middleware packages.

There are several other solutions for depth-tracking available that provide similar features to OpenNI's and Microsoft's development kits. Omek's Beckon SDK (Omek Interactive, Israel) is a comprehensive development suite that supports any commercially available 3D camera. Omek also offers a range of tools to enable developers to author gestures or integrate full-body tracking into the Unity game engine. SoftKinetic's iisu SDK (SoftKinetic, Belgium) offers a similar set of comprehensive 3D gesture recognition tools that support a range of different depth-sensing cameras. Lastly, PrimeSense offers a depth-sensing camera and sensing framework NITE which can be used with the OpenNI framework. For each of the available software suites it is important to consider which sensors are supported, how gestures are detected and implemented, which distances to the camera are leveraged (e.g. close-range tracking for hand-detection) and which licensing options are available.

15.2.2 Flexible Action and Articulated Skeleton Toolkit (FAAST)

FAAST (Institute for Creative Technologies, CA; Suma et al. 2011) is a middleware that facilitates the integration of full-body control with VR applications and video games using OpenNI-compliant depth sensors (Fig. 15.1). It interfaces directly with OpenNI/NITE or the Microsoft Kinect for Windows SDK to access pose information and perform additional high-level gesture recognition for generating events based on the user's movements.

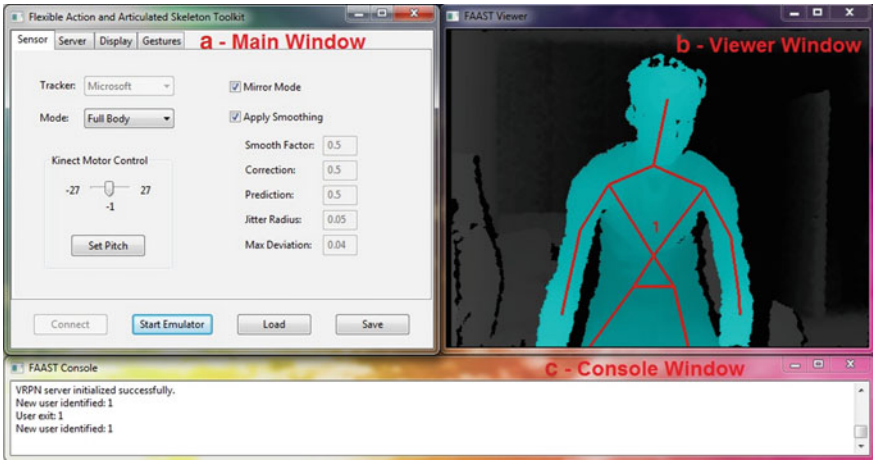

Fig. 15.1 FAAST user interface

FAAST considers two broad categories of information from the sensor: actions and articulated skeletons. Articulated skeletons consist of the positions and orientations for each joint in a human figure and are useful for VR and video game applications in allowing direct control of a virtual avatar through body movements. FAAST retrieves these skeleton joints from the OpenNI/NITE or Kinect for Windows SDK drivers and transmits them to the end-user application using the Virtual Reality Peripheral Network (VRPN), a popular software package in the VR community for interfacing with VR peripherals (Taylor et al. 2001). FAAST includes a custom VRPN server that streams each joint's skeleton data as a six degree-of-freedom tracker, allowing applications to interface with the sensor as they would with any other motion-tracking device.

FAAST enables these custom sensors to provide input to a wide range of applications. It can be used to emulate keyboard and mouse input for standalone PC applications as well as web-based games. Users can customize the key-bindings and sensitivity for triggered actions at run-time, thus providing flexible input that can easily be adjusted according to the individual user's preferences and therapeutic goals. Exemplarily, the same game can be played with different gestures and ranges of motion to allow players with different levels of ability and different therapy goals to play with or against each other. Figure 15.2 shows a player using FAAST to control an online game with different sets of gestures.

15.2.3 MiddleVR

MiddleVR (IminVR, France) is a middleware that facilitates the integration of virtual reality hardware in a wide range of applications. A standalone version of MiddleVR and a Unity implementation are offered for developers to enhance user

Fig. 15.2 Using FAAST to control a game (Suma et al. 2011)

interaction though VR hardware. MiddleVR can be utilized to display immersive content with complex projection solutions such as Head-Mounted Displays, Powerwalls, VR Walls, Workbenches, Holobenches, HoloStages, Caves and 3D Televisions. Active and passive stereoscopic displays are supported. MiddleVR is also able to integrate VR peripherals, enabling the user to interact with the virtual world in a more natural way. For example, 3D trackers, haptic devices, joysticks, 3D mice, depth-sensing cameras and several other devices can be used to interact with immersive virtual content. Most of these devices are supported through the VRPN library. However, it is also possible to use other drivers to integrate devices such as the Microsoft Kinect. This allows developers to use the VR peripherals to perform actions in the 3D application instead of using traditional input devices. For instance, the position and orientation of the viewport of a virtual scene can be modified according to the measurements obtained from a 3D tracker installed on the head of the user. This can be a meaningful interface when the users are expected to orient themselves in the environment or observe a virtual scene or object from different perspectives.

Alternatively, MiddleVR can be used to interact with virtual objects via alternative input devices such as a haptic glove. Users can perform different gestures to reach, pick up and release virtual items which can be a useful training scenario for motor rehabilitation. Integration of the Microsoft Kinect depth camera gives the developer the freedom to use any of the user's tracked joints to interact with a virtual environment.

MiddleVR is set up through a graphical configuration tool which allows the developer to select the supported hardware and decide how the hardware input is mapped to user actions. For example, the head position of the Microsoft Kinect can be mapped to the position of the scene's main camera, allowing the user to look around a virtual scene naturally. In addition to the configuration tool MiddleVR also provides a C# interface and a C++ API for direct integration in applications (Fig 15.3).

15.2.4 Unity

Unity (Unity Technologies, San Francisco, CA) is a development platform for games and general 3D content that allows the developer to publish applications for

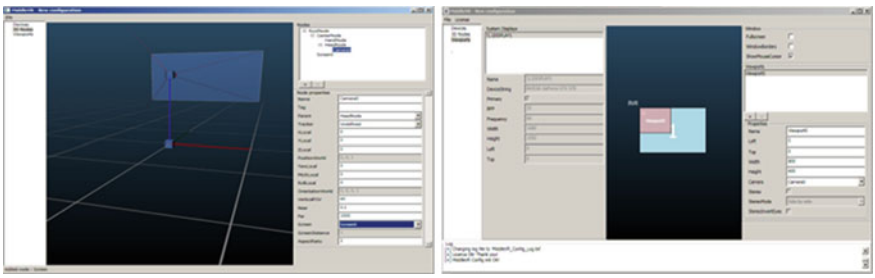

Fig. 15.3 MiddleVR interface

PC, Mac, mobile devices, web, Flash and gaming consoles (Wii, PS3, Xbox360). As a so-called game engine it integrates a wide range of tools that can be used to create interactive virtual environments. Unity's toolset encompasses the rendering of 3D models, animations, shading and lighting, input and output operations, user interfaces, physics simulations, audio, network integration, scripting of game logic and other features that are needed to develop games and interactive applications. There are many free and paid online tutorials available to explain each of the engine's features in detail. Of particular importance is Unity's ability to communicate with external applications via the use of plugins. This allows developers to import data from the Kinect via OpenNI or the Kinect for Windows SDK into Unity. The information can then be used within Unity to control virtual avatars, trigger events or allow the user to interact with a virtual environment. Further, external applications such as MiddleVR can act as a middleware to exchange data between Unity and a wide range of sensors or display solutions.

15.3 Tutorial

15.3.1 Using FAAST to Define Gestures that can be Used to Play Existing Games

The FAAST middleware can be used to define a range of gestures. These gestures can be assigned to key strokes, such as the up, down, left and right arrows or the space bar and 'w', 's', 'a' and 'd' keys. A FAAST keyboard emulation feature can be used so that the gestures assigned to the different keys can be used to play any existing game that uses those keys. This provides an opportunity for clinicians to assign customized gestures for their patients to play games that are engaging and low-cost (or free) for therapeutic exercise. The following section will provide a basic overview of how to define and save gestures.

15.3.1.1 FAAST User Guide

In order to use FAAST, a Microsoft Kinect camera needs to be installed and connected to the PC. Both versions of the Kinect sensor, the Xbox360 version and the Kinect for Windows version, are supported. Further, Microsoft's Kinect for Windows SDK or OpenNI/NITE drivers are needed to use FAAST. The driver packages can be downloaded and installed from the respective websites listed in the reference list (Microsoft Kinect for Windows, FAAST). Once these driver packages are installed, open FAAST by double clicking on the FAAST application icon in the FAAST folder.

FAAST consists of three different windows: main window (Fig. 15.1a), viewer window (Fig. 15.1b) and console window (Fig. 15.1c). The main window consists of four tabs: the sensor tab, the server tab, the display tab and the gesture tab. The sensor tab (Fig. 15.1a) allows the user to specify the installed drivers (Microsoft SDK or OpenNI), the tracking mode (full-body or seated), and sensor angle and provides several options to control tracking characteristics such as mirror mode and smoothing parameters. The server tab (Fig. 15.4a) provides choices for the tracked skeletons and the coordinate system in which the user's joints are being calculated. For any standard application in which only one user is tracked to control an application, no changes of these options are required. The display tab (Fig. 15.4b) allows the user to change the appearance of the FAAST viewer window by switching between RGB image and depth image for foreground and background of the tracked scene. It also provides options for changing the text size of the console window and moving all FAAST windows as well as saving the window configuration. The gesture tab (Fig. 15.4c) can be used to define gestures by providing sets of input conditions and specifying the output of each gesture.

To add a gesture, click on 'New gesture' in the gesture tab. A new window will open (Fig. 15.4d). Type a gesture name and define timings for input and output. Timeouts are used to limit the frequency at which gestures can be recognized (input timeout) or how often outputs are triggered when a gesture output is set to continuous looping (e.g. pressing a mouse button 10 times per second while the gesture is being maintained). Timeouts are displayed in seconds.

Once the new gesture is added, Input and Output conditions must be assigned to the gesture. Input relates to the gesture that must be performed and Output is the keyboard press, mouse-click or mouse movement that the gesture will be assigned

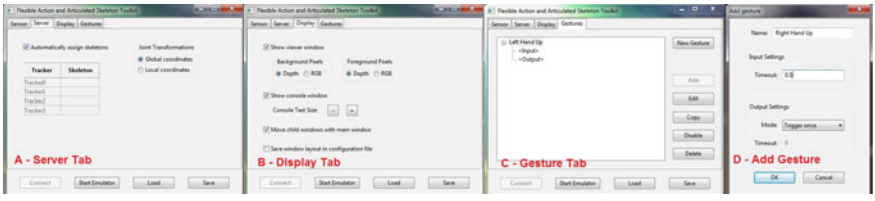

Fig. 15.4 FAAST interface: main window, viewer window and console window

to within the game or interaction. Input and Output conditions can be defined by clicking on each gesture's input and output node (Fig. 15.5a) in the gesture tab and selecting the "Add"-button on the right-hand side of the window (Fig. 15.5b). FAAST offers a wide variety of input constraints to define the user movement that is expected to trigger an output action. Selecting a constraint type opens a new window to add and define the parameters of a gesture (Fig. 15.5c). Exemplarily, body constraints can be customized to detect any leaning, turning or jumping movement in any direction and magnitude/distance. Multiple input and output nodes can be combined to create more complex gesture sequences which have to be satisfied before an action is triggered. Time constraints also provide the ability for temporal sequences of inputs and outputs for even greater customizability of the FAAST tool.

Gestures can be enabled and disabled separately to experiment with different combinations of gestures. This can be especially helpful when individual gestures are tested with patients with motor deficits. By sequentially combining different gestures, the complexity of the user interface can be increased gradually.

All settings can be saved to the local hard drive by selecting the "Save" button in the FAAST main window. Previously saved configurations can be loaded by clicking the "Load" button and selecting the saved configurations file (file ending.xml). Once all gestures have been defined and the Kinect has been successfully started ("Connect" button), pressing the "Start Emulator" button will enable gesture recognition. The FAAST console window will display whenever defined gestures are triggered and users enter or exit the scene. When a gesture is recognized the assigned output (e.g. keyboard or mouse button press) is triggered. The output is sent to the currently active application. As such, FAAST can run alongside standalone or web-based applications to provide the needed user input, effectively replacing keyboard or mouse inputs. The gesture recognition can be stopped by pressing the "Stop Emulator" button in the main window.

When defining gestures for existing applications the developer or clinician needs to consider the number of gestures that are required to control all aspects of

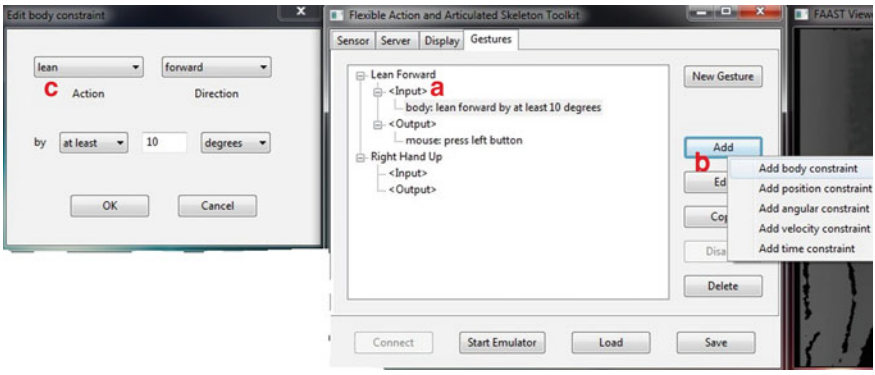

Fig. 15.5 FAAST interface: defining gestures

the game or application. Any application that requires a large number of keystrokes or mouse commands will place high demands on the user to remember and combine the assigned gestures. This problem becomes even more apparent with motor-impaired patients who only have limited range of motion or limited control over parts of their body. Furthermore, games that require precise timings of button presses or use mouse movements as primary control scheme are difficult to master with FAAST. Specific clinical considerations also include the goal of the interaction. The games and gestures must be chosen carefully so as to not encourage sequences of movements that could be inappropriate or unsafe for the patient.

15.3.2 Using MiddleVR to Interact Within a Unity3D Application

MiddleVR can be used to integrate VR input and display devices into existing game development workflows. Developers can utilize VR hardware to make their rehabilitation applications more engaging and customizable for the users. The following section will provide a basic overview of how to configure MiddleVR and a Microsoft Kinect within the game engine Unity in order to start developing customized rehabilitation applications.

15.3.2.1 MiddleVR User Guide

Working with MiddleVR requires several configuration steps that will be described in Sects. 15.3.2.1 and 15.3.2.2. For the purpose of this guide, MiddleVR's integration with the Unity game engine was used. The goal of this guide is to control an avatar within Unity by mapping simple cubes to the location of each of the Kinect's joints.

Firstly, MiddleVR needs to be configured to specify input and output devices that are used with the final application. The configuration is saved in a file (file format.vrx) which is then used in conjunction with the actual application that is being controlled by the user. The initial configuration process is performed in MiddleVR's main window. The window consists of the windows for "Devices", "3D Nodes", "Viewports" and "Cluster".

By default Middle VR has selected keyboard and mouse as input devices (Fig. 15.6). To add or delete any device we will use the "+" and "−" buttons. In the list of "3DTrackers": we have the option to select "Tracker Kinect (Microsoft SDK)".

After defining all needed input devices we need to map the device input to objects or actions in the application (Fig. 15.7). Through the 3D Nodes window the user can define scene objects (i.e. nodes) to be controlled by the input of any Kinect joint. Nodes are placed in a hierarchy with the "VRRootNode" (Fig. 15.7a)

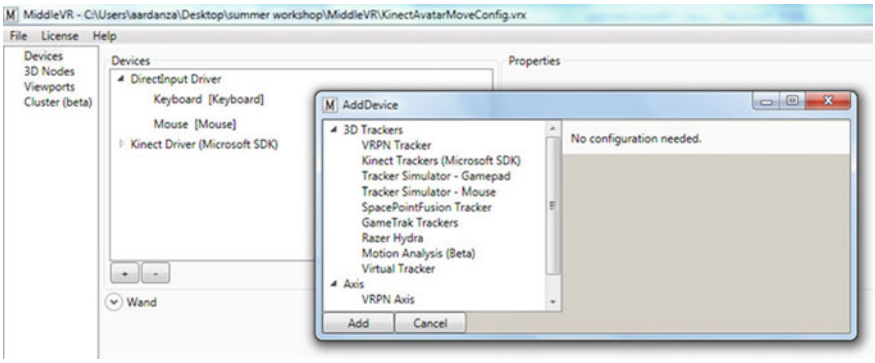

Fig. 15.6 MiddleVR: device selection interface

being the highest node in this hierarchy under which all other nodes are being arranged. While creating mappings between user input and nodes one needs to be aware that each created node will correspond to an object in the actual application.

For example, if the goal of the final application is to control the scene camera through the Kinect's head joint and to control two virtual hands by the Kinect's two hand joints, a total of five nodes are needed. The "VRRootNode" is at the very top of the hierarchy, a "Kinect0.RootNode" (Fig. 15.7b) contains all remaining nodes associated with the tracked skeleton of the active Kinect camera. Underneath the "Kinect0.RootNode" the three nodes for the user's head and both hands are placed. Each of these nodes needs to be assigned a Kinect joint that is controlling the node's position. The available Kinect joints can be selected in the dropdown list "Tracker" (Fig. 15.7c). The name of each created node should follow a consistent naming convention, because the node names and the objects

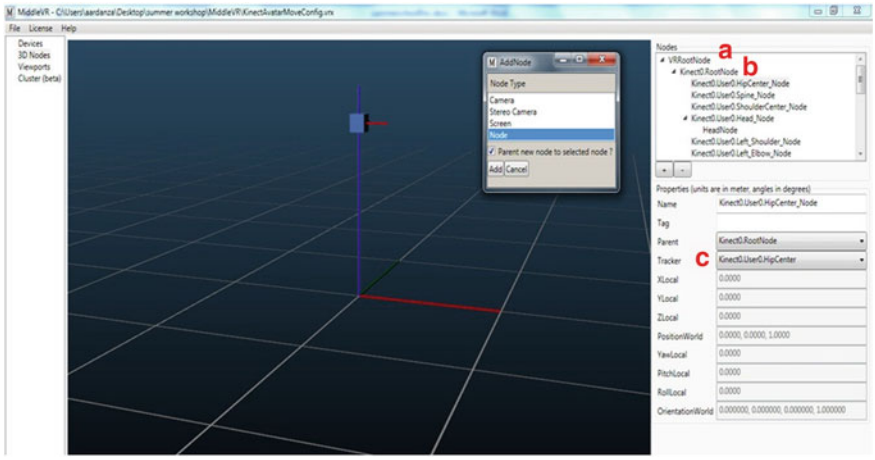

Fig. 15.7 MiddleVR: scene objects interface

being controlled by the Kinect within the game engine Unity need to match. For example, one of the "Kinect0.User0.Head_Node" in Fig. 15.8 is connected to the Kinect tracker "Kinect0.User0.Head". When the application is set up within the game engine Unity, the object being controlled by the Kinect's head joint needs to be named "Kinect0.User0.Head_Node".

MiddleVR also provides the option of controlling a camera to manipulate the player's view during the game. This can be achieved by creating a "camera" object instead of a node and linking it to the Kinect's head joint. However, for the purpose of this tutorial a default Unity camera will be used instead of a head-tracked viewpoint.

Once all nodes and trackers have been configured, a configuration file is saved to the local hard drive. This file contains the information of the nodes and trackers which needs to be accessed from the application within Unity.

15.3.2.2 Unity3D User Guide

In order to import MiddleVR's tracking information into Unity a downloadable Unity-package is available on MiddleVR's website (see reference list). After launching Unity and creating a new scene (select "File—New Scene"; Fig. 15.8a), the MiddleVR package needs to be imported into Unity. This can be achieved by clicking "Assets—Import Package—Custom Package".

- After selecting the previously downloaded Unity package (MiddleVR.unity-package) and importing the asset, a new folder named "MiddleVR" will be available in the Project View (Fig. 15.8b).

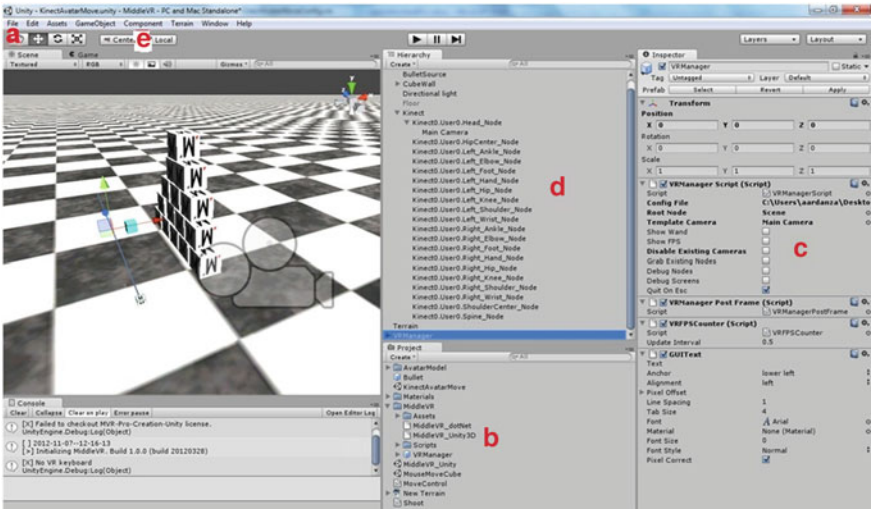

Fig. 15.8 Unity3D Main interface

- The folder contains a so-called prefab named "VRManager". The "VRManager" needs to be dragged and dropped into the Hierarchy View (Fig. 15.8d).
- When clicking on the "VRManager" in the Hierarchy View, the properties of the "VRManager" Script become available in the Inspector (Figs. 15.8c, 15.9).
- The "Config File" needs to be set to the location of the previously created MiddleVR configuration file (Fig. 15.9).
- The Root Node needs to be set to the game object that contains all of the Kinect-controlled objects. In this example the root node that contains all relevant objects is the game object "Kinect" in the Hierarchy View (Fig. 15.8d).
- In order to control all objects correctly, game objects for each of the previously created nodes are required. These game objects need to be placed underneath the root node and renamed to the names we assigned to the nodes in MiddleVR.
- For this example, simple cubes or spheres are sufficient to simulate an avatar (Select "Game Object—Create Other—Cube/Sphere" in Unity). In a complete game each of these nodes could be part of a more sophisticated rigged character that contains joints and limbs created in 3D modeling applications such as Blender, Autodesk Max or Maya.
- Figure 15.8d shows a list of 20 cubes that correspond to 20 nodes representing the whole skeleton that the Kinect is able to track. If the original configuration within MiddleVR contains more or less nodes (e.g. only head and two hands as previously described), the number of game objects in the hierarchy needs to be adjusted accordingly.
- If all objects and scripts are set up, the play button in the top center of the Unity application will attempt to run the application. If a Kinect is connected and the Microsoft Kinect for Windows SDK is installed, the created cubes will follow each joint of the tracked skeleton. The virtual avatar comes alive.
- In order to enable the avatar to interact with the virtual environment, a physics simulation can detect whenever the avatar collides with virtual objects. For this purpose physics components need to be added to each avatar game object. This can be achieved by selecting each avatar object and adding a box collider (or

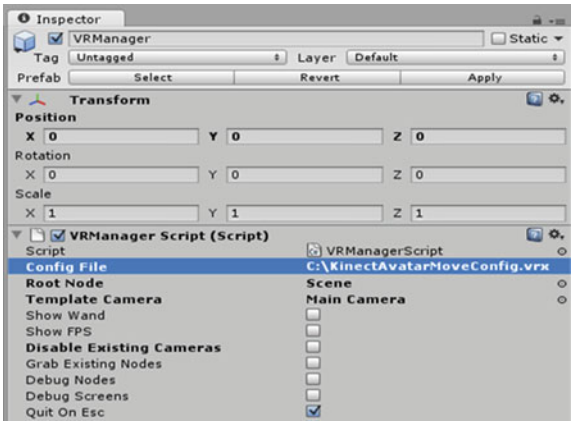

Fig. 15.9 Unity3D VR manager properties

sphere collider if spheres were selected as body parts, Fig. 15.8e "Component—Physics—Box Collider/Sphere Collider").
- The example in Fig. 15.8 shows a stack of boxes in the scene view. For collisions to take place, these boxes also need box colliders and a rigidbody. Rigidbody components (Fig. 15.8e "Component—Physics—Rigidbody") allow the object to be affected by gravity.
- Once all objects are set up properly, the avatar can interact with the stack of boxes by simply colliding with the placed objects.
- Be advised that attempting to punch or kick the stack of boxes can lead to injuries as the player can easily collide with real-world obstacles while interacting with the immersive virtual environment.

15.4 Conclusion

Many researchers are beginning to explore the potential of the Microsoft Kinect technology for rehabilitation. This chapter provided an overview of the tools that can be used to customize existing games or to develop games that use VR hardware. Two potential examples for the development of rehabilitation applications using the Kinect were outlined in this chapter: (1) the use of a middleware (FAAST) that allows tailored gestures to be programmed so a patient can play an existing game with individualized movements and (2) the use of MiddleVR middleware for the development of software specifically designed for customized rehabilitation. The use of FAAST (or similar middleware applications mentioned in this chapter) is perhaps more accessible and user-friendly for clinicians and non-programmers to practice and use with patients in the clinical setting. However, gestures and game choices must be given careful consideration in order to maintain rehabilitation goals and avoid the risks of frustration or injury to the patient. The development of specifically tailored rehabilitation applications using low-cost hardware such as the Kinect requires more technical and programming skills. While this chapter provided a brief introduction to some of the technical components, the development of game-based rehabilitation applications is an iterative process that requires the collaborative effort and involvement of clinicians, patients, designers, programmers and engineers.

References

Antonin V, Anatol F, Bradford MF, Mindy L (2004) Reaching in reality and virtual reality: a comparison of movement kinematics in healthy subjects and in adults with hemiparesis. J NeuroEng Rehabil 1:11

Adamovich SV, Merians AS, Bojan R, Tremaine M, Burdea GS, Recce M, Poizner H (2004) A virtual reality based exercise system for hand rehabilitation post–stroke: transfer to function.

Conference Proceedings-IEEE Engineering in Medicine and Biology Society, vol 7, pp 4936-4939
Boian R, Sharma A, Han C, Merians A, Burdea G, Adamovich S, Recce M, Tremaine M, Poizner H (2002) Virtual reality-based post-stroke hand rehabilitation. Stud in Health Technol Inform 85:64–70
Chuang TY, Huang WS, Chiang SC, Tsai YA, Doong JL, Cheng H (2002) A virtual reality-based system for hand function analysis. Comput Methods Programs Biomed 69(3):189–196
De Mauro A, Carrasco E, Oyarzun D, Ardanza A, Neto AF, Torricelli D, Pons JL, Gil A, Florez J (2011) Virtual reality system in conjunction with neurorobotics and neuroprosthetics for rehabilitation of motor disorders. In: Proceedings of Medicine Meets Virtual Reality 18: NextMed, vol 163, p 163
De Mauro A, Carrasco E, Oyarzun D, Ardanza A, Frizera-Neto A, Torricelli D, Pons JL, Agudo AG, Florez J (2012) Advanced hybrid technology for neurorehabilitation: The HYPER Project. Adv Robotics Virtual Real 26(1):89–108
Deutsch J, Paserchia C, Vecchione C, Mirelman A, Lewis J, Boian R, Burdea G (2004) Improved gait and elevation speed of individuals post-stroke after lower extremity training in virtual environments. J Neurol Phys Ther 28(4):185
Deutsch JE, Borbely M, Filler J, Huhn K, Guarrera-Bowlby P (2008) Use of a low–cost, commercially available gaming console (Wii) for rehabilitation of an adolescent with cerebral palsy. Phys Ther 88(10):1196–1207
Dvorkin AY, Shahar M, Weiss PL (2006) Reaching within video—capture virtual reality: using virtual reality as a motor control paradigm. Cyberpsychol Behav 9(2):133–136
Flexible Action and Articulated Skeleton Toolkit (FAAST) Institute for Creative Technologies, CA. Available: http://projects.ict.usc.edu/mxr/faast/. Accessed 14 Nov 2012
Flynn SM, Palma P, Bender A (2007) Feasibility of using the sony playstation 2 gaming platform for an individual poststroke: a case report. J Neurol Phys Ther 31(4):180–189
Fung J, Malouin F, McFadyen BJ, Comeau F, Lamontagne A, Chapdelaine S, Beaudoin C, Laurendeau D, Hughey L, Richards CL (2004) Locomotor rehabilitation in a complex virtual environment. Conference Proceedings-IEEE Engineering in Medicine and Biology Society, vol 7, pp 4859–4861
Fulk GD (2005) Locomotor training and virtual reality—based balance training for an individual with multiple sclerosis: a case report. J Neurol Phys Ther 29(1):34–42
Fung J, Richards CL, Malouin F, McFadyen BJ, Lamontagne A (2006) A treadmill and motion coupled virtual reality system for gait training post stroke. Cyberpsychol Behav 9(2):157–162
Holden M, Todorov E, Callahan J, Bizzi E (1999) Virtual environment training improves motor performance in two patients with stroke: case report. J Neurol Phys Ther 23(2):57
Jack D, Boian R, Merians AS, Tremaine M, Burdea GC, Adamovich SV, Recce M, Poizner H (2001) Virtual reality-enhanced stroke rehabilitation. IEEE Trans Neural Syst Rehabil Eng 9(3):308–318
Lange BS, Koenig S, Chang C, McConnell E, Suma E, Bolas M, Rizzo A (2012) Designing informed game-based rehabilitation tasks leveraging advances in virtual reality. Disabil Rehabil 34(22):1863–1870 (ePub ahead of print)
Lange B, Rizzo A, Chang C, Suma EA, Bolas M (2011) Markerless full body tacking: depth-sensing technology within virtual environments. Proceedings of the interservice/industry training, simulation, and education conference (I/ITSEC) 2011a, Orlando, FL
Lange B, Suma EA, Newman B, Phan T, Chang C, Rizzo A, Bolas MT (2011) Leveraging unencumbered full body control of animated virtual characters for game-based rehabilitation. In Proceedings of HCI (14), pp 243–252
Lange BS, Flynn SM, Chang K, Proffitt R, Rizzo A (2010) Development of an interactive game-based rehabilitation tool for dynamic balance training. Top Stroke Rehabil 17(5):345–352
Lange B, Flynn S, Rizzo A (2009) Initial usability assessment of off-the-shelf video game consoles for clinical game-based motor rehabilitation. Phys Ther Rev 14(5):355–363

Merians AS, Jack D, Boian R, Tremaine M, Burdea GC, Adamovich SV, Recce M, Poizner H (2002) Virtual reality–augmented rehabilitation for patients following stroke. Phys Ther 82(9):898–915

Merians AS, Tunik E, Adamovich SV (2009) Virtual reality to maximize function for hand and arm rehabilitation: exploration of neural mechanisms. Stud Health Technol Inform 145:109–125

Microsoft Kinect for Windows SDK Available: http://www.microsoft.com/en-us/kinectforwindows/develop/developer-downloads.aspx. Accessed 17 Nov 2012

MiddleVR < i'm in VR—The virtual reality company> Available: http://www.imin.fr/middlevr/. Accessed 14 Nov 2012

Mirelman A, Bonato P, Deutsch JE (2009) Effects of training with a robot-virtual reality system compared with a robot alone on the gait of individuals after stroke. Stroke 40(1):169–174

Omek Beckon SDK 3.0 Available: http://www.omekinteractive.com. Accessed 17 Nov 2012

OpenNI Available: http://openni.org/. Accessed 14 Nov 2012

Rand D, Kizony T, Weiss P (2008) The sony playstation eyetoy: low–cost virtual reality for use in rehabilitation. J Neurol Phys Ther 32(4):155–163

Rizzo AS, Kim GJ (2005) SWOT analysis of the field of virtual reality rehabilitation and therapy. Presence: Teleoperators Virtual Environ 14(2):119–146

SoftKinetic iisu SDK 3.5.1 Available: http://www.softkinetic.com. Accessed 17 Nov 2012

Subramanian S, Knaut LA, Beaudoin C, McFadyen BJ, Feldman AG, Levin MF (2007) Virtual reality environments for post-stroke arm rehabilitation. J NeuroEng Rehabil 4(1):20

Suma EA, Lange B, Rizzo A, Krum DM, Bolas MT (2011) FAAST: the flexible action and articulated skeleton toolkit. IEEE, Virtual Reality, pp 247–248

Taylor RM, Hudson TC, Seeger A, Weber H, Juliano J, Helser AT (2001) VRPN: a device-independent, network-transparent VR peripheral system. Proceedings of the ACM symposium on virtual reality software and technology, pp 55–61

Unity Technologies Available: http://unity3d.com/. Accessed 14 Nov 2012

You SH, Jang SH, Kim YH, Hallett M, Ahn SH, Kwon YH, Kim JH, Lee MY (2005) Virtual reality–induced cortical reorganization and associated locomotor recovery in chronic stroke an experimenter-blind randomized study. Stroke 36(6):1166–1171

Author Biographies

Sebastian Koenig is currently working as a Research Associate at the USC Institute for Creative Technologies and as Lead User Researcher at Red 5 Studios where he conducts user tests and experimental studies in cognitive psychology. Dr. Koenig also designs and develops virtual reality applications for the assessment and training of cognitive functions.

Aitor Ardanza Telecommunications engineer specialized in computer graphics, games and virtual reality. Currently he is a researcher at Vicomtech-IK4 in the area of 3D animation and interactive virtual environments. His main interests are in the design and development of virtual reality interfaces for a wide range of applications (such as rehabilitation and assistive technology) and in the interaction with motion capture systems.

Camilo Cortés is currently a Ph.D. student of the Engineering program at Universidad EAFIT, Medellin, Colombia. He is performing an ongoing research internship at Vicomtech, San Sebastian, Spain, as part of his Ph.D. thesis. His research interests are Medical Robotics, Mechatronics, Applied Computational Geometry and Optimization of Geometrical Setups.

Alessandro De Mauro is senior researcher at the eHealth and Biomedical Applications Department of Vicomtech-IK4 (San Sebastian, Spain) and member of the eHealth Group (Bioengineering Area) at Biodonostia Health Research Institute. His main research interests are related to virtual and augmented reality for medicine, real-time and physically based simulation and medical robotics.

Belinda Lange is a Research Scientist at the Institute for Creative Technologies and Research Assistant Professor in the School of Gerontology at the University of Southern California. She received her PhD and degree in Physiotherapy (Honors) from the University of South Australia and her Science Degree from Flinders University. Dr. Lange′s research interests include the use of interactive video game and virtual reality technologies for cognitive assessment, motor rehabilitation, exergaming, postoperative exercise, and virtual human character interactions. She was on the conference program committee for Meaningful Play conference in 2008, 2010 and 2012, on the organizing committee for the rehabilitation track of the Games for Health conference in 2010, 2011 and 2012, co-chaired the Presence 2009 conference and was the Workshop Chair for the International Virtual Rehabilitation conference in Zurich in 2011. She is also a co-founder of www.games4rehab.org, a non profit social network that brings together individuals with disabilities and those undergoing rehabilitation with researchers, clinicians and game industry professionals.