The Nottingham Trent University
School of Science and Technology

Virtual room scene with the Wii Remote/Motion Plus

Archontia Anna Tsapanidou

2009

Project Report in part fulfilment of the requirements for the degree of MSc Computer Games Systems

i. Abstract

This project explores the use of the Nintendo Wii Remote and its recently released Motion Plus extension in a virtual environment. The Wii Remote and Motion Plus have been used to simulate manipulation and placement of objects into a virtual room scene. The environment has been created with the OGRE 3D rendering engine and the ODE physics engine. The integration of the controlling device and its extension into OGRE 3D and its sub systems (CEGUI) has been successful.

The results of the project show that there is great potential into the use of the controller with its extension in virtual environments and many different forms of intuitive interaction can be explored.

ii. Acknowledgements

I would like to thank my supervisor, Dr. Richard Cant, for his time, valuable guidance and enlightening ideas throughout this project.

A big thank you also goes to my family for their support and encouragement.

iii. Table of Contents

iv. List of Figures and Tables

1. Introduction

Virtual Reality and the realistic simulation of scenes taken from the real world have always been challenging topics and have evolved through the years with the emergence of new software and hardware technologies and the improvement of the existing ones. The combination of faster and more powerful computers and the ever-growing number of software engines and tools has led to the development of impressive systems that simulate the laws governing the real world quite convincingly and provide an immersive experience.

One important part of such virtual scene systems is the interface through which a user is capable of interacting with these computer-synthesised worlds. Apart from the actual software interface, the choice of a suitable hardware input device is crucial. Conventional input devices, such as mice and keyboards, could be proven to be inadequate for interaction and navigation in three dimensional environments, since they are limited by their two degrees of freedom. Many approaches have been taken, utilising devices characterised by more degrees of freedom, such as gloves and 6 DOF mice, to enhance usability.

An aspect of great interest is the fact that systems of such nature find wide applications, ranging from pure entertainment purposes, for instance in Computer Games, to educational and training purposes, such as training medical personnel in operations and various examinations.

Bearing the former in mind, the search and exploitation of low cost and widely available 6 DOF devices would be beneficial, as it would not only enable more possibilities for research but also, the distribution of the results of research (virtual scene/simulation software) into a wider end-user audience. After the release of the Wii game platform by Nintendo in 2006, the Wii Remote (Wii's primary controller) has been recognised as a device of great potential, since it possesses the desired attributes, stated above. Many different projects have been developed and publicised, rendering the Wii Remote a popular researcher/developer choice, for applications not limited to the Wii platform. In addition, the recent release of the Motion Plus extension for the Wii Remote, delivering more precision in motion sensing, has raised the interest in the device even more.

1.1 Project Aim

The aim of this project is to integrate the Nintendo Wii Remote and its extension Motion Plus into the interface of a virtual room scene, populated by objects. With the aid of this input device, the user will be able to grab, manipulate and place the objects anywhere in the room. The scene itself incorporates simulation of physics. The virtual environment is to be created with the aid of the graphics engine OGRE 3D and the physics engine ODE. The system built is to be compatible with the virtual object list used in the Untidy room project[1] by Cant and Langensiepen (2009).

1.2 Report Overview

The rest of the report is divided into seven Chapters:

[1] "Methods for Automated Object Placement in Virtual Scenes"

Chapter 2 (Background Information) covers essential information about the Nintendo Wii Remote, libraries through which the connection with a Bluetooth enabled PC is possible, the OGRE 3D graphics engine, the ODE physics engine and the OGREODE wrapper for OGRE. Furthermore, a selection of projects that use the Wii Remote are presented and their interaction model is discussed. Last but not the least, a brief description of the Untidy Room project is given.

Chapter 3 (System Requirements) identifies the requirements of the system that has been developed for this project.

Chapter 4 (System Design) is concerned with the design of the system, in terms of its user interface and its break-down into distinct software components.

Chapter 5 (Implementation) is concerned with the development of the system and its various components, according to the requirements.

Chapter 6 (Results and Discussion) presents and discusses the results of this project and clarifies decisions made during the implementation process.

Chapter 7 (Conclusion and Future Work) summarises the project, discusses any possible future development or different approaches to the problem and concludes the report.

2. Background Information

This Chapter illustrates the background literature and research behind the design and development of this project.

2.1 The Nintendo Wii Remote

With the release of the Wii console in the end of 2006, Nintendo brought into the public a new type of controlling device, the Wii Remote or Wiimote. The Wiimote has been an innovation over the existing game console controllers, in the sense that it offers a new means of user interaction with the virtual worlds and interfaces of each game application. This wireless controller is capable of motion sensing via an accelerometer and optical sensors and enables the interaction and manipulation of objects through motion (Wikipedia, 2009a).

Figure 2.1 The Nintendo Wii Remote (Armitage, n.d.)

Figure 2.2 The Sensor Bar (LED beacon) (Mann, 2009)

2.1.1 Hardware Specification

Nintendo has never officially released the specifications of the Wii Remote. However, individuals have examined/reverse-engineered the inner workings of the device and documented its technical features in detail, so that custom applications targeted for platforms other than the Wii could be developed (Lee,

2008), (Bernardes et al., 2009). This information is available at no cost in websites, such as www.wiili.org and www.wiibrew.org (Lee, 2008).

The Wii Remote is battery powered and its main components are an infrared camera tracker, an accelerometer, a series of twelve buttons, a vibration motor for tactile feedback, four individually addressable blue LEDs, a speaker for auditory feedback, an internal flash memory component and an expansion port (Lee, 2008).

Motion sensing is succeeded through the 3-axis linear accelerometer ADXL330, which has a +/-3g sensitivity range, a sampling rate of 100Hz and an 8bit resolution per axis (Lee, 2008). The coordinate system defined is right-handed, with the positive x-axis to the left and the positive z-axis upwards, when the device is held horizontally facing upwards (Bernardes et al., 2009).

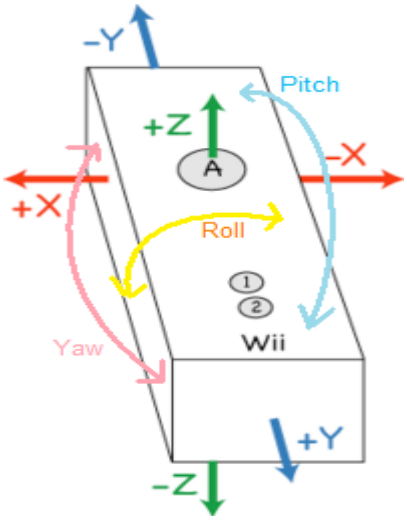


Figure 2.3 The Nintendo Wii Remote Axes (WiiLi Project, n.d.)

The infrared camera, located at the front end of the Wii Remote, has a resolution of 1024x768 pixels, a 100Hz refresh rate and a 45 degree field of view (Lee, 2008). It is capable of tracking up to four simultaneous IR light beacons within the device's field of view (Lee, 2008), (Bernardes et al., 2009). Nintendo provides a "sensor bar" accommodating two such beacons (Bernardes et al., 2009) bundled with the Wii console. The name of the bar could be said to be misleading, since it is the Wiimote – with the aid of this IR camera – that tracks/senses the location of the bar, in order to position itself in front of the display more accurately and not vice versa (Lee, 2008).

The Wiimote is a Bluetooth device, thus it can communicate with any Bluetooth enabled PC (Lee, 2008).

2.1.2 Wii Remote Spatial Information Reports

As a physical object, the Wiimote has 6 Degrees of Freedom (DOF) (WiiLi Project, n.d.). It can move linearly towards the directions of the three axes, that is, perform linear translations, but also it can rotate by angles (three rotation angles – pitch on the x-axis, roll on the y-axis and yaw on the z-axis) (WiiLi Project, n.d.), as seen in Figure 2.3.

While in motion, the Wiimote transmits values in arbitrary units representing the acceleration imparted on it (WiiLi Project, n.d.). The magnitude of the acceleration reported back from the controller is equal to g (gravitational acceleration), when it is at rest on a flat surface (Guo and Sharlin, 2008). If the controller is performing freefall, the value is then close to 0 (Guo and Sharlin, 2008). According to Guo and Sharlin (2008), the above implies that a relatively

accurate measurement of the pitch and roll angles is feasible, only when it is quite still.

In order to obtain the yaw, pitch and roll values, the data originating from the infrared camera has to be utilised. According to Lee (2008, p. 41), "the [...] camera sees the two groups and provides a method of laser-pointer-style input. The software can transform the x,y coordinate dot pairs to provide an averaged x,y coordinate pair, a rotation and a distance. The x,y and rotation values correspond to the controller's yaw, pitch and roll, respectively, and the distance is estimated using the known physical separation of the two IR groups (beacons) and the camera's fixed field of view".

2.1.3 Motion Sensing Limitations

No matter how appealing the motion sensing capability of the Wii Remote is, several limitations have been reported by researchers and developers. After undertaking their project involving the use of the controller for movement of 3D objects in a 3D scene, Cochard and Pham (n.d.) have reported that they faced various problems, mostly concerning the nature of the sensor outputs. Some examples have been the noise in the accelerometer data, due to the physical implementation of the device, limited accuracy, constant change of the initial accelerometer values, need of mathematical derivation of the third dimension, since IR camera tracking produces two dimensional output, actual amounts of raw data values that need to be filtered and manipulated further mathematically, in order to be useful and more (Cochard and Pham, n.d).

2.1.4 The Motion Plus Extension

In June 2009, Nintendo released an extension for the Wii Remote controller, under the name Motion Plus. Motion Plus promises greater accuracy in motion sensing. The data derived by it, in conjunction with the data obtained by the accelerometer and the infra red camera (tracking the sensor bar) can allow better tracking of the position and orientation of the controller in space. Thus, with Motion Plus, a direct mapping of the physical motion into a virtual scene is possible (Nintendo, 2008).

According to Wiibrew (2009a), the Motion Plus contains the InvenSense IDG-600 dual-axis gyroscope (angular rate sensor), for the pitch and roll and the EPSON TOYOCOM X3500W single-axis gyroscope, for the yaw. It reports the current angular velocity of the Wii Remote in all axes. This allows the orientation information to be decoupled from acceleration information, making it possible to track both independently.

Figure 2.4 The Nintendo Motion Plus extension for the Wii Remote (Acres, 2009)

Finally, the extension is equipped with a pass-through port, so that other extensions can be connected to the main controller, without having to disconnect Motion Plus.

2.2 Wii Remote Connectivity Libraries

As mentioned before, the Wii Remote has never been intended to be used in platforms other than the Nintendo Wii console. Therefore, several freeware helper libraries and APIs have been developed and delivered by both individuals and teams to allow the implementation of Wii Remote enabled pieces of software. These APIs can be used with a variety of languages and graphical environments. Some examples include Brian Peek's Managed Library WiimoteLib for the Windows platform, supporting the Visual Basic and C# languages (Peek, 2007), the native C++ Wiimote library WiiYourself! (Gl.tter, 2008) and Wiiuse by Michael Laforest, written in C (Laforest, n.d.). These APIs provide similar functionality, when it comes to connectivity with the Wii Remote. However, the level of processing of the raw data extracted from the device varies.

Due to the experimental nature of these attempts, it is not possible to take full advantage of the functionality of the device. For instance, in this stage, the mechanism that enables simultaneous use of both of the Motion Plus and any other extension connected through its pass-through port is unknown (Wiibrew, 2009a). Moreover, the reports obtained by the Wii Remote indicate that there is a mechanism for the self-calibration of the Nunchuck and Motion Plus extensions, but there is not enough information on how this is accomplished, as with the case of multiple extensions (Wiibrew, 2009a).

For the purposes of this project, WiiYourself! has been chosen primarily, since it has been the first to include support for the Motion Plus extension.

2.2.1 WiiYourself! API

WiiYourself! was initially developed as a C++ port of Brian Peek's library, but it has been rewritten and extended (Gl.tter, 2008). The library supports the use of multiple Wii Remote controllers and a number of peripherals, such as the Nunchuck, the classic controller, the Balance board and the Motion Plus. It provides functions to retrieve all crucial data, such as the accelerometer values, button presses and releases, battery level and visibility of the IR beacons (dots), from the Wii Remote. Other capabilities of the API include the estimation of the controller orientation (unfortunately not including the yaw), the setup of the LEDs, the use of the haptic feature (Rumble) and use of the built-in speaker (Gl.tter, 2008).

In terms of connectivity, WiiYourself! is able to auto-detect the type of the HID report returned from the device, in order to support all Bluetooth stacks and it can detect connection loss and breaks. Data can be retrieved by polling the device or through callbacks. Other features of the API include extensive debug output, always valuable to the developer, and threaded design, for multitasking (Gl.tter, 2008).

2.2.2 Wiiuse API

Wiiuse API is licensed under the GNU GPLv3 and GNU LGPLv3 for non-commercial use. It supports multiple Wii Remote controllers, extensions such as the Nunchuck and the classic controller, but not the Motion Plus. Unlike WiiYourself!, it runs on a single thread and it is compatible with a number of Bluetooth stacks Microsoft, BlueSoleil, WidComm and Toshiba for the Windows OS) (Laforest, n.d.).

At first, this library seems less flexible than WiiYourself!. Nevertheless, it allows more accurate estimation of the orientation and position of the controller in space. The data concerned with the IR beacon tracking are treated internally

and in a more advanced way; the number of visible dots is taken into account and the results are averaged according to that number. The yaw of the controller is then calculated with the aid of the IR data. The Wii Remote proximity to the Sensor bar (IR beacon) is provided by the library itself, so that further calculations by the developer are not necessary, in order to obtain the third dimension. Also, the option to specify the aspect ratio of the monitor used (e.g. 4:3, 16:9) is available, so that virtual screen is set and the values of the x,y coordinate pair are obtained in this virtual screen resolution (Laforest, n.d.). The same pair values are simply normalised in WiiYourself!.

Wiiuse has been chosen to be used as a companion to WiiYourself! in this project, due to its more advanced treatment of the raw values extracted by the controller.

2.3 Human Interaction in a Virtual Room

When designing an interface for human interaction, there is a primary issue that has to be taken into account and this is the ease of use of such an interface. The input method used to interact with the virtual room and its assets must provide an intuitive way of manipulation and placement of these objects in parts of the room by the user. Research work to address this issue falls into four categories of techniques; the DOF (Degrees of Freedom) reduction technique, the pseudo-physical technique, the semantic technique and the input device and manipulation technique (Xu, Steward and Fiume, 2002).

The first three techniques are input device independent or do not specify a particular input device type. In brief, they approach the problem by imposing rules in object manipulation and placement, such as the reduction of the degrees of movement freedom to which the user is exposed to, the semi-automatic placement of objects by allowing the user to drag them into a relative position and let the pseudo-physics engine do the exact placement and the exploitation of geometry of objects and the scene with the use of "binding areas" and object labels, where a placement is possible only if the labels of the objects are compatible (Xu, Steward and Fiume, 2002). On one hand, these techniques could simplify interaction to a degree. On the other hand, they would either not take full advantage of the Wii Remote and Motion Plus or not allow a more accurate simulation of object placement.

The fourth technique is input device dependent. Works that take this approach argue that higher DOF input devices are more suitable than limited DOF devices. Higher DOF devices, in the likes of 3 and 6 DOF mice, 6 DOF joysticks, 6 DOF gloves are more intuitive to use than 2 DOF devices, such as the standard mouse and keyboard, in the sense that they are a direct mapping metaphor. The user can grab, move and rotate a 3D object, as they would in the real world (Xu, Steward and Fiume, 2002). Xu, Steward and Fiume (2002) claim that although the higher DOF input devices have the advantage of adding a natural feeling to the user experience, they are expensive and not universally accessible. However, with the introduction of the Wii Remote to the public, this has been surpassed, since this input device is inexpensive and widely available, as already discussed in this report.

2.3.1 Higher DOF Input Device Interaction Issues in a 3D scene

One problem that arises when interacting with a 3D virtual room is how to select objects for manipulation. The most common technique to tackle this is raycasting. Bowman and Hodges (1997) define raycasting as “the use of a virtual light ray to grab an object, with the ray’s direction specified by the user’s

hand" (the object is grabbed if it intersects with this ray). However, they report that raycasting has the drawback that it does not provide a method of controlling the object's distance from the user. Moreover, there are difficulties rotating an object in place, except around the axis of the ray (Bowman and Hodges 1997).

Another mechanism suggested by the same authors is named "Go-go". It is more suitable for an immersive Virtual Reality environment, but it could be appropriate for a virtual room 3D scene on a 2D display device (monitor). The virtual hand movement coincides directly with the physical hand movement, while the user's hand moves in a defined local area and becomes faster, when the user moves their hand beyond this area. With this technique, the user can grab distant objects in the scene, the same way they would reach for a physical object in the distance. However, this approach is limited, because it is subject to a function that takes the user's arm length into account (Bowman and Hodges, 1997). The "stretch go-go" technique overcomes this limitation, by taking into account only the physical hand movement (Bowman and Hodges, 1997).

A problem that is not obvious from the beginning is related to object rotations with such a device. Poupyrev, Weghorst and Fels (2000) discuss that most researchers and commercial developers tend to use isomorphic (one to one) mapping between the input device and the virtual object. However, this type of mapping would render full rotation of an object impossible or straining for the user, due to the anatomical constraints of the human hand; a human hand cannot rotate by an angle of 360 degrees. For this reason, they propose the use of non-isomorphic mapping, which could deviate from strict realism, but could allow easier manipulation.

2.4 Projects Featuring the Wii Remote

The Wii Remote has been utilised in various projects that differ in nature and scope. According their interaction model, these projects could fall into separate groups. Of course, this grouping is not strict; the criteria are based on the degree of similarity amongst the models.

One approach taken is to use the Wii Remote as a motion tracking device (camera). In this case, the controller is stationary tracking one or more moving Sensor bars/IR beacons. Lee (2008) has developed a series of applications of this kind, from head tracking for desktop VR displays, to finger/object tracking and interactive whiteboards/tablet displays. Lee exploits all possibilities with not only custom software, but custom hardware, such as glasses with LEDs for head tracking, as well. In their paper titled "Optical Tracking Using Commodity Hardware", Hay, Newman and Harle (2008) discuss a "method for using Nintendo Wii controllers as a stereo vision system to perform 3D tracking or motion capture in real time" (Hay, Newman and Harle, 2008). Similarly, Bradsaw and Ng (2008) have developed a system to track Conductors' hand movements using multiple Wii Remotes (Bradsaw and Ng 2008).

Another interaction paradigm involves the use of the controller as a Tangible User Interface for operating and controlling robots and robotic arms. In this case, the orientation of the Wii Remote is the vital part; robot motion is mapped to the pitch, roll and yaw of the input device. Examples of this include Guo and Sharlin's (2008) attempts to use a TUI , in order to control a small zoomorphic robot and the operation of two heavy robot grappler arms by the Australian-based heavy machinery manufacturer Transmin, by replacing the joystick controls with Wii Remotes (Funk, 2009).

The topic of using the Wii Remote as a input device for 3D worlds and Virtual Reality applications is quite common. In this subject area, the controller is utilised either as a pointing pen or as a 1:1 object mapping metaphor (in most applications the two types of interaction are concurrent). Cochard and Pham

(n.d.) have developed a simple 3D scene filled with geometric shape objects, where the user can move and place these objects in the scene. The project is based on natural gesture recognition and raycasting, it uses OpenGL and the WiiYourself! API and its purpose is to evaluate the usability of such an interface (Cochard and Pham, n.d.). In a similar fashion, Gallo, De Pietro and Marra (2008) take advantage of the controller to create a more intuitive interface for volumetric medical data manipulation. The Wii Remote is used both as a pointing device and as a 1:1 metaphor (Gallo, De Pietro and Marra, 2008).

Manipulation of basic geometric objects in a 3D environment has been the subject of another project by Bernardes et al. (2009). Here, the aim of the project was to integrate the Wii Remote with enJine, an educational 3D game engine in Java (Bernardes et al., 2009).

Most of the above instances incorporate gesture recognition. In this way, motion patterns – rather than absolute positions of the Wii Remote – are taken into account. Finally it has to be noted that at the time of writing of this report, no publicised project utilising Motion Plus has been found.

2.5 Tools to Create the Virtual Room Environment

In this section, information about the building tools of this project is presented.

2.5.1 The OGRE 3D Graphics Engine

The Object-Oriented Graphics Rendering Engine or OGRE 3D is a flexible, cross-platform open source C++ library for the development of real-time 3D graphics applications. It is distributed under the GNU Lesser General Public License and the supported platforms are Microsoft Windows, Linux and Mac OS X (Junker, 2006, pp. 1-5).

According to OGRE Team (2000), a wide range of libraries, tools and assets generated by other applications are supported by this 3D engine. Specifically, it supports:

- Direct3D and OpenGL, abstracting their low level detail to make development easier,
- pixel and vertex shaders, either written in an assembler or in HLSL, GLSL, DirectX9 etc,
- loading and manipulation of textures in various formats,
- files with defined meshes,
- animation,
- particle systems,
- maskable scene querying system and raycasting,
- several shadowing techniques,
- various plug-ins/add-ons extending its functionality (adding overlay GUIs, I/O, etc.)

and many more (OGRE Team, 2000), (Junker, 2006, pp. 1-5). OGRE 3D is scene-oriented and it allows quick, flexible and easy to manage 3D scene generation, without being tied to any specific scene type. It handles render state management, spatial culling and transparency automatically reducing development time (OGRE Team, 2000).

OGRE is not oriented for the development of a particular type of application. It provides a generic 3D framework to create games, demos, simulations and even business applications, for commercial (as long as the source code is

distributed together with the product), educational and personal purposes (OGRE Team, 2000). Due to this fact, there is no built-in physics engine and collision detection library. However, integration of other outside-OGRE libraries, such as ODE (Open Dynamics Engine) for physics and collision detection, and plug-ins is possible (OGRE Team, 2000). Every developer is free to choose and use whichever solution they believe is appropriate for the needs of their system.

Another key feature of this 3D engine is the vast amount of documentation available. OGRE Team (2000) states that their API is extensively and thoroughly documented. Example sources and demos, numerous tutorials, how-to guides, as well as support through the website Forum, are all available. This fact allows the quick setup and full development of any type of 3D application.

2.5.1.1 The OGRE 3D Architecture Basics

OGRE 3D is based on an Object Oriented design. The base of all classes of the API is the Root class. The Root object holds references to a series of Manager classes, which – as the name implies – are responsible for managing the resources used in the lifetime of the application (such as textures, 3D models/meshes, materials etc.), the scenes rendered, the overlay elements, the particle systems and more. Moreover, the Root object is independent of the implementation of the rendering API in use (DirectX or OpenGL), due to the existence of the RenderSystem abstract class (Ogre Wiki, 2008). This gives us the advantage to be able to choose between the two, when the OGRE application starts, without having to include any specifics in the source code.

In an OGRE application, the class that is used more than anything else is the SceneManager, since it is responsible for keeping track of everything displayed onto the screen (from cameras and lights to planes) (Ogre Wiki, 2009a). Every renderable object (mesh) is an Entity; objects like cameras, lights, planes cannot be entities (Ogre Wiki, 2009b). In order to render an Entity or make an object of the other types visible into the scene, a SceneNode has to be created and this object has to be attached to the node. Nodes can be considered to be placeholders that contain information about the position of everything in the screen. When something needs to be moved, what is actually translated is node that holds it.

When the OGRE application starts, the Root object is created the rendering system and window are created, the resources are loaded and initialised and then the scene is created, so that the rendering loop can start (Ogre Wiki, 2009d). From this point and onward, two functions of another class will be called every time a frame has started and ended. This class is the FrameListener. The bodies these functions are responsible for the updates in each frame. Moreover, the rendering room continues, until one of them returns false (Ogre Wiki, 2009c).

2.5.2 The ODE Physics Engine

The Open Dynamics Engine or ODE is an open source C++ library for the simulation of the dynamics of articulated rigid bodies, such as vehicles, moving objects in virtual scenes and characters with moving limbs (Smith, 2006 pp. 1-2).

ODE features its own collision detection system (however it allows use of other collision detectors, if necessary), a stable first order integrator for simulating the dynamics of the world through time, a contact solver based on the Dantzig LCP solver and uses an approximation of the Coloumb friction model for friction. For the simulation of motion, a Lagrange multiplier velocity based

model is used. However, the engine does not provide a visualisation model and it is up to the developer to choose a rendering system (Smith, 2006 pp. 1-2).

ODE supports a variety of geometries for collision. These primitives are spheres, boxes, capped cylinders, planes, rays, tri-meshes and in the latest versions convex hulls. Moreover, it supports many different joint types, such as ball and socket, hinge, slider, fixed, angular motor, linear motor and universal joints (Smith, 2006 pp. 1-2).

2.5.2.1 ODE Simulation Concepts

The first basic concept in the simulation is the rigid body. A rigid body is characterised by constant and volatile properties. Constant properties are its mass, its centre of mass and its inertia matrix, which represents the mass distribution around the centre of mass (Smith, 2006 p. 5). Properties that change over time are the position, the linear and angular velocities and the orientation (Smith, 2006 p. 5).

The second basic concept is that of the joint. A joint is a constraint that keeps two rigid bodies connected together, no matter what forces are applied to them, in the same way a hinge works in real life. Every time a step in the simulation is taken, constraint forces are calculated and the joints apply these forces, so that the connected objects move without the joint being broken (Smith, 2006 p. 6). To prevent breaks, ODE uses an Error Reduction Parameter (ERP). Additionally, it is possible to determine the nature of the constraint forces. Constrains can be hard or soft. In the first case, they cannot be violated under any circumstances, e.g. the one object colliding into another object must not penetrate the latter. In the second, they can be violated to allow simulation of softer materials. The nature of the constraints is set by the Constraint Force Mixing (CFM) parameter (Smith, 2006 pp. 7-8).

Apart from this type of joint that is predetermined, joints are created at the moment of collision. These joints are created on the points of contact (contact joints) and are given information about friction at the contact surface, softness and bounciness of the surface and so on, in order to simulate the behaviour of the colliding objects correctly. Then, they are quickly destroyed, in order for the simulation to progress to the next step (Smith, 2006 p. 10).

2.5.2.2 Lifetime of a Simulation in ODE

The simulation begins by creating a dynamic world and the bodies to inhibit this world, together with their state (position etc). Then, the joints are created, attached to the bodies and their parameters are set. The collision world and collision geometry objects are created, together with a special group of joints to hold the contact joints (Smith, 2006 pp. 10-11).

At this point, the simulation loop is ready to begin. In every step, forces are applied to the bodies and the joint parameters are adjusted accordingly. The collision detection function is called, a contact joint is created for each point that collides and it is added to the contact joint group. Then, the simulation steps to the next state and all joints in the contact joint group are removed (Smith, 2006 pp. 10-11).

Finally, when the simulation is over, the dynamic world and the collision world are destroyed (Smith, 2006 pp. 10-11). It has to be noted that the simulation time step does not coincide with the frame rate of the rendering system, when the dynamic world is visualised. Several steps can be taken inside a single frame rendered per second. The step size is user defined, but caution

has to be taken, since larger time steps can reduce stability and accuracy (Smith, 2006, p75).

2.5.3 The OGREODE Wrapper

OGREODE is an open source add-on that allows seamless integration of the ODE physics engine in the OGRE 3D environment. This C++ API is targeted for use in Windows and Linux.

The OGREODE wrapper is responsible of:

- the handling of the conversion between the data structures used by the two libraries,
- the synchronisation of the positions and orientations of the geometries in the ODE dynamic world to the OGRE objects (entities),
- the ODE dynamic world stepping with a variety of methods, automatic or manual, with the StepHandler class.

2.6 A Few Words About the Untidy Room Application

A matter that influenced many design and implementation decisions of this project is the desired compatibility with the Untidy Room application. The Untidy Room application has been developed by Cant and Langensiepen (2009), in order to experiment with automatic placement of objects into a virtual scene using heuristic methods. It is written in C++, uses the ODE physics engine and OpenGL as the rendering system.

The primitives used for the ODE simulation in the virtual room are groups of convex shapes, i.e. one object consists of more than one convex hull. Some of the hulls in the group do not actually represent the geometry of the object utilised for the collision detection; these two types of entities are the widgets and containers of the object, without which the automatic placement would not be possible. The geometries are loaded into the system from appropriate files.

Each object to appear in the scene is located in a list of objects that contains the unique name of the object, the name of the geometry file, the name of the 3D mesh file (the 3D model), the archetype name of the object (e.g. Mug, if the object is a mug) and other properties used in the automatic placement mechanism. When the application runs this list is loaded from a file, so that the room can be filled with items.

Interface-wise, the application allows manual placement and manipulation of the on-screen objects. The input is performed through the mouse and keyboard. Due to the fact that the application has grown significantly over the years, the user interface has become convoluted, rendering the program difficult to use.

3. System Requirements

The requirements describe the system developed for this project, in terms of functionality.

F1. The user should be able to interact with the application with the aid of the Wii Remote and the Motion Plus extension.

F2. In the case that the Motion Plus extension is not present or does not function, the application should be able to cope with the single controller alone.

F3. The user should be able to insert objects into the Virtual Room scene.

F4. The user should be able to interact with the objects inhibiting the Virtual Room environment in two ways.

- F4.a The first method of interaction should involve manipulation of the objects directly. With the aid of the controller, they should be able to change orientation and position of the objects in the scene.

- F4.b The second method of interaction should involve indirect manipulation of the objects. The user should be able to push objects already in the scene, with the aid of a Wii Remote avatar, i.e. a virtual representation of the controller in the scene.

F5. The interface of the application should be intuitive and simple to use.

F6. The virtual environment and object interaction should follow a physical model (simulation of physical forces).

F7. The application should be compatible with the Untidy Room application.

- F7.a It should be able to import the list of objects from the Untidy Room library file.

- F7.b It should be able to import the object geometry description files.

4. System Design

There are two important parts in the application developed for the project, where design decisions have to be made, before the implementation process. The first part is the user interface through which the user is to interact with the environment and its assets. The second part is the actual software system and the building blocks that it is comprised of.

4.1 User Interface Design

When it comes to the user interface design of a virtual 3D environment application, the aspects that have to be tackled is the navigation in the environment and the interaction with the elements placed in it. Moreover, decisions have to be taken on how to implement additional functionality, in a way that it will not obscure the view or the make the primary tasks (here, navigation and placement) difficult to perform.

4.1.1 Navigation

The Virtual Room is a free roaming environment without up and down constraints in movement. The user is able to move forward, backward, left and right and also to look in every direction. Since the Wii Remote provides a Directional Pad (D Pad), that is a set of four buttons, one for each direction, this type of motion has been mapped to the D Pad.

Changing where the user looks, i.e. changing the camera view, is less straightforward than moving. There are three possible design solutions to tackle this problem:

- using the Wii Remote
- using the analogue stick of a Nunchuck extension
- using the mouse

The first solution would involve defining four regions in the screen, as seen in Figure 4.1 below. Once the Wii Remote enters these regions the camera would turn towards the corresponding direction. The major drawback with this is that it would be difficult to place and object near these regions, since entering them will cause the view to change.

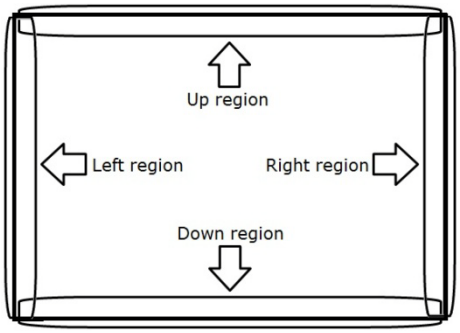


Figure 4.1 Screen regions

The second solution is the most optimal and intuitive, since many computer games use it, therefore it would not be an alien concept to the user. Moreover, it would eliminate the need of additional input devices, such as the mouse.

However, as discussed in Chapter 2 Section 2, simultaneous utilisation of the Motion Plus and other extensions has not been yet achieved. It could be possible to make a distinction of when the Motion Plus is present or not, but this would result in the creation of two separate mechanisms in the same interface and would lead in confusion.

The third option does not take advantage of the Wii Remote and could be seen as an emergency solution. The camera control is performed by the mouse, as in the majority of 3D environment applications. This approach has been chosen to be taken for the prototype system of this project.

4.1.2 Interaction

Interaction in the Virtual Room application involves object positioning and manipulation. The most intuitive and realistic model of interaction is to use isomorphic (one to one) mapping of the orientation of an object to the Wii Remote controller. Despite Poupyrev, Weghorst and Fels' (2000) note of warning about isomorphic mapping (discussed in Chapter 2 Section 3.1), it is believed that serious ergonomic problems will not arise, when the Wii Remote is the input device. The controller is not attached on the user's hand, as is the case with a GlovePIE, so that a 360 degree rotation of the hand would be impossible. Furthermore, it is light, slim and rectangular shaped, fitting in one's hand; rotating the controller around its axes and maintaining its position is pretty easy.

The model for the first mode of interaction (manipulation and positioning) is described below:

- The user moves a helper cursor with the aid of the Wii Remote (the controller is used here as a pointing device).
- The object under the cursor is selected by pressing a controller button and manipulation mode is entered.
- The selected object will be highlighted by making its bounding box visible. The bounding box will be made invisible, once the object has been deselected.
- Once in the manipulation mode, the object can be rotated around its axis and moved in space, in every direction (up, down, left, right, towards the front, towards the back). Motion in the z-axis (depth) is constrained, so that the object will not come too close to the camera and obscure the view or too far from the camera.
- In order to make positioning more accurate, the user has the choice to lock and unlock the orientation, while the object is being dragged across the room. Apart from accuracy, this choice has been made when another scenario has been taken into account. In the case that the controller becomes perpendicular to the floor and loses visibility of the Sensor bar, moving the cursor and the object in space is impossible.
- The user can place the object in a desired position by pressing the same button again.
- The objects can be locked in position by a button press, if desired, so that they do not get knocked over accidentally.

The model for the second mode is rather simpler. Once entering this mode, the cursor becomes a Wii Remote avatar, which is a rigid body as well. The user can move the avatar in every direction in space and push objects. Selection/ deselection and spawning of objects into the room are deactivated. Switching between the two modes is performed through controller button presses.

4.1.3 Additional Functionality and On Screen Information

The term additional functionality refers to the spawning of object into the room. The need of a mechanism for spawning can be satisfied by enhancing the application with a Graphical User Interface (GUI). The selection of the object to enter the room will be performed through a scrollable list. It will be possible to hide the list or drag it to another position on screen, so that the field of view is clear. The list will be invisible when entering the second mode of interaction, since its presence bears no meaning in this mode. Of course the cursor mentioned above is a GUI element, as well.

As described in the requirement list in Chapter 3, the application will be able to function with or without a Motion Plus extension. Moreover, there might be the case that the extension is present, but for some reason not functioning. The user should always be informed about the status of the extension. For this, an overlay text element will be present at one corner of the screen[2].

4.2 Software System Design

After an examination of the requirements of the application to be build for this project, the key components of the system have been identified. The main components are the graphic system, the physical simulation system, the GUI system, the input system, the system to parse the geometry and library of objects files and the representation of the objects in the world.

As expected, the graphic system is responsible for the visualisation of every event taking place in the virtual world and the world itself on screen. It creates the 3D environment, displays the GUI elements and the informative text and updates the contents of the screen, so that the changes occurred in the physically simulated world in each simulation step match with the display.

The physics system is – again as expected – responsible for the simulation of the physical laws governing the virtual world. It takes care of the collision detection, handles the behaviour of the objects in the world and simulates the effect of the forces applied on them.

The GUI system creates all the necessary elements for the additional functionality, determines their appearance and handles the events tied to it, such as button presses, selection of listed items and the cursor motion.

The input system handles the communication between the application and the input device. Every Wii Remote button press, change of position, orientation and acceleration values, as well as, change in the visibility status of the IR beacon is fed through this into the overall system, to make user interaction with the virtual world feasible. Moreover, it handles the communication status of the input device, ensuring the overall system will respond to changes, such as broken connections, reconnections and connection of extensions on the Wii Remote.

Without the parsing system, it would be impossible to load and handle the data stored in external files into the overall system. With its aid, the imported data is fed to the appropriate fields of the components of the application, so that the objects can be created and the GUI list of objects can be populated.

The objects of the virtual world are characterised by several attributes. These attributes are a unique name to distinguish them from others, a 3D model (mesh) to represent their renderable appearance, a body and mass to represent them in the ODE defined world and a geometry description to enable collision detection in this world. In addition, they are – renderable – Entities in the OGRE

[2] Overlay text can show more information, such as the status of an object (locked/unlocked etc) and can be used for debugging purposes as well.

3D created graphical world and they are tied to a SceneNode, so that they can be transformed and translated in the world.

4.2.1 System architecture

Now, there is the question of how all these components fit into the OGRE 3D engine architecture. A diagram describing the system architecture can be seen below.

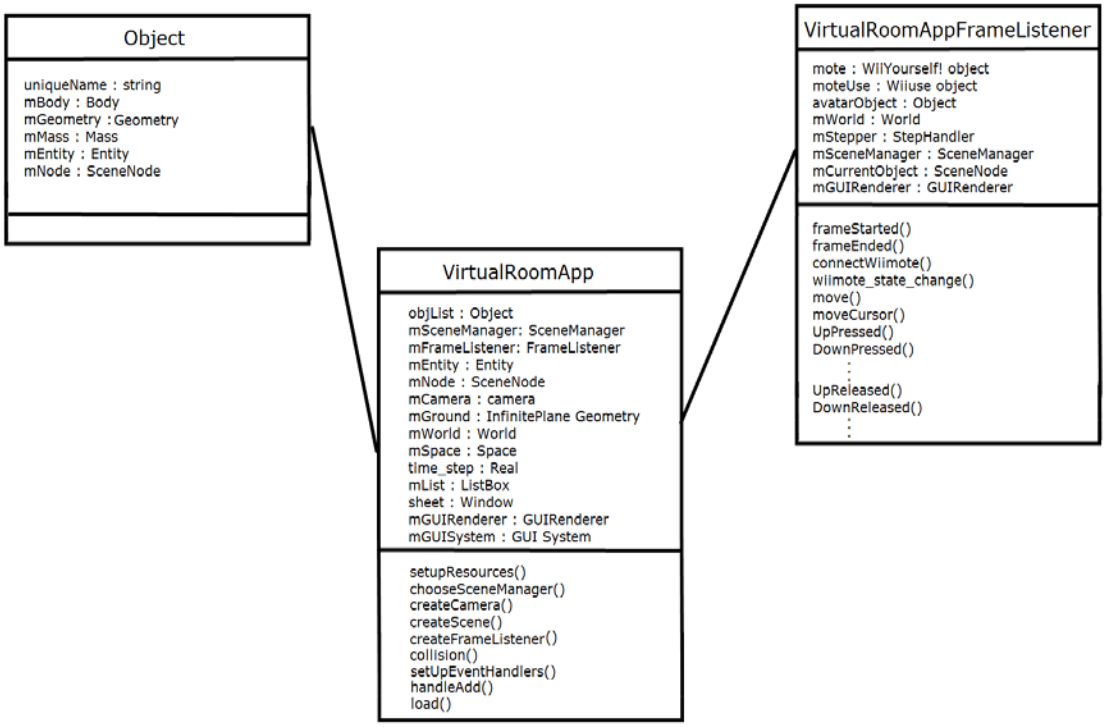


Figure 4.2 System architecture

The main system component is the VirtualRoomApp. Initialisation of every component (GUI, physics world, graphics world, list of object, loading) and collision detection happens here. The VirtualRoomApp FrameListener is part of the graphics system, handling updates per frame. There resides the input system, with its series of connection and input device event related functions, as well as the physical world progression and step handling.

5. Implementation

This chapter discusses the implementation process of the development of a prototype system for this project, together with the issues faced during this process.

5.1 The Graphics System

The OGRE 3D distribution comes with an abstract Example Framework with a basic set of functions and setup to allow rapid development of a basic OGRE application. These functions can be extended, so that one can add preferred functionality to their application. This Framework has been chosen as the starting basis for the prototype.

VirtualRoomApp, which is derived from the corresponding Example Framework class, has a series of functions to initialise the rendering system. The functions chooseSceneManager, createCamera, createViewports and createFrameListener set up the scene manager, the camera, the viewport and the frame listener for the application. The paths for the application resources (such as the 3D meshes) are set by the function setupResources. Moreover, lighting for the scene and a static geometry (a geometry that is not intended to be movable throughout execution) representing the floor are set in the function createScene.

Each time a frame starts or ends the event is passed to the VirtualRoomFrameListener object and the frameStarted and frameEnded, so that updating of the contents of the screen is performed. NB: This can be seen as the main loop of the application, since frameStarted and frameEnded decide on whether or not the rendering is to continue.

The informative text element is created with the aid of the public domain class TextRenderer, provided by Ogre Wiki (2009f). It is used in both VirtualRoomApp and VirtualRoomFrameListener classes.

5.2 The Physics System

The OGREODE world and collision space are initialised in the createScene function of the VirtualRoomApp class. The gravity vector magnitude has been set to the value of -9.8, with direction facing downwards to the y axis (the floor of the virtual room).

5.2.1 Stepping

OGREODE provides various automatic stepping mechanisms, through the StepHandler class. However, they do not seem to have any effect or work properly. Therefore, the stepping of the world has been handled manually. The step is set to a fixed value of 0.001. The frame rate is also fixed to 60 frames per second. If rendering frame rate drops under 60 fps, several steps are taken per frame. The StepHandler is set in the createScene function and step progression occurs in the frameStarted function of the VirtualRoomFrameListener.

5.2.2 OGREODE Simulation Settings and Collision Detection

As mentioned in Chapter 4 Section 2.1, the collision detection is handled in the VirtualRoomApp class. In order to behave as a collision listener, this class inherits functionality from the OGREODE CollisionListener class and collisions are managed by the function collision.

The Constraint Force Mixing (CFM) and Error Reduction Parameter (ERP) have been given the default values. The friction has been set to a value of 8, which allows object sliding, without toppling over and the Contact Correction Velocity has been set to 1, which prevents objects from falling into each other. Also, the bounciness of the surface on the contact points has been set to 0.1, resulting in a satisfactory effect.

5.2.3 Geometry

The Untidy Room application uses convex hulls to describe the geometry of the objects for the collision detection, so this type of geometry has been initially considered to be used.

Convex geometry is computationally inexpensive and allows the representation of an object by several convex hulls, for handling of concave geometries. Nevertheless, it has only recently been added to list of primitives that ODE supports and it is extremely under-documented. It has been proven that it is unsupported in OGREODE, since the library provides a dummy constructor. The constructor has been rewritten to allow support for this project, but the library cannot produce debug geometries.

Moreover, Untidy Room uses a custom format to define convex geometry and it is not compatible with ODE/OGREODE, as it is. There have been attempts to create files in a format supported by ODE's convex collider. At first, Blender was used to create geometries from the 3D meshes of the objects, but the given scripts failed to produce correct convex shapes. Then, the Untidy Room itself was used to create appropriate files, which did not result in success. Sample files can be found in Appendix D.

For these reasons, for testing purposes, trimesh geometry was utilised instead. Trimesh geometry has also only recently been supported by ODE, but unlike convex geometry, it is well documented. Performance-wise, it is relatively stable and slower than convex geometry and produces a large number of contacts on collision. OGREODE supports the trimesh primitive and there is no need of extra code from the developer.

5.2.4 Mass

Apart from the geometry of an object, mass is a property that determines the result of a collision. The mass of a rigid body can be calculated automatically by ODE for the trimesh case. It has been observed that this method fails randomly. Untidy Room calculates this property for its objects, so it has been used to export the mass, the centre of mass and the inertia tensor for each object. The files produced are loaded into the VirtualRoom application and this approach seems to work reliably.

5.3 The GUI System

OGRE 3D is delivered with the CEGUI add-on. CEGUI is a free library used to create GUIs quickly and simply. The library uses XML files to define the appearance/layout of the GUI. Moreover, a graphical editor (CELayoutEditor) to

create layouts is available. CEGUI has no rendering or mouse and keyboard event listening capability; OGRE provides a renderer for it and input events have to be injected manually into it (Ogre Wiki, 2009e).

The GUI for the application is initialised in the VirtualRoomApp, in the createScene function. A renderer is defined and the layout is loaded. Then a Listbox item and the cursor are created. The Listbox is populated by ListboxTextItem objects, initialised with the unique names of each object loaded from the library of objects. To add functionality to the Add button of the Listbox widget, an event listener is set up to catch button clicks. Each time the button event occurs, the event is handled by the function handleAdd. This function creates and places a new object into the scene, by retrieving its details by the list of objects and then removes it from this list, since it is possible to have only one object instance in the scene.

CEGUI does not provide native support for input by the Wii Remote. The controller input is injected into the CEGUI system, as if it were a standard mouse device. Wii remote button clicks are handled as left mouse button presses. For the cursor movement, instead of injecting mouse coordinates into CEGUI, x,y position values retrieved from the Wii Mote are passed into the appropriate function.

5.4 The Input System

The input system resides in the VirtualRoomFrameListener class. As mentioned in a previous Chapter, two Wii Remote libraries are used in this application, so the controller is represented by two different objects (class members mote for WiiYourself! and moteUse for Wiiuse).

Connection, status and data retrieval if achieved through the mote object. The connection of the input device to the application is initiated in the VirtualRoomFrameListener constructor. The connect function polls, until one controller is detected and then initialises the two Wii Remote objects. For the mote object, a report type is defined indicating what type of data is to be retrieved from the controller (e.g. IR, button, acceleration).

Changes in the state of the controller (buttons pressed or released, Motion Plus connected, connection lost) are handled by a callback function indirectly. When a change is detected, the callback function on_state_change changes the values of boolean flags corresponding to the nature of the change. The function triggers polls the values of these flags in the frameEnded function and then fires the appropriate function to respond to the change. The only events handled directly are the loss of connection and the Motion Plus extension detection. In the first case, the callback function will respond by trying to reconnect to the controller again. In the second case, it will enable the extension and change the report type, in order to retrieve values from the extension.

The object moteUse is only used for the handling of the IR and orientation values. These values are injected from mote to moteUse in the body of the frameEnded function. IR data is utilised to move the CEGUI cursor (function moveCursor) and to move objects in the z axis. Motion of the camera is performed in function move, according to the button status of the Wii Remote.

5.5 Objects, Library and Parsing System

When it comes to object representation and the library of objects, in order to ensure that Virtual Room is compatible with Untidy Room, the appropriate classes (Container_Library_Data, Library_Object, LibraryEntry, ObjectLibrary, Container_Instance, Object_Instance and the Ontology system) have been lifted

from the latter and modifications have been made, where required. Not all of them are vital for the Virtual Room application, but their addition allows further development of the system to incorporate automatic placement of objects.

For the objects, Object_Instance has been modified to include attributes, such as a World, a Body, a Geometry, a Mass, an Entity and a SceneNode and the constructor parameter list has been modified to accept OGREODE world objects. A unique name and archetype are already provided as attributes. The ObjectLibrary class has been extended with an additional function, FindAll, in order to be able to retrieve an object from the list of object only by unique name. This functionality was necessary for the handling of the selection of Listbox items.

Parsing of the library of objects occurs in these classes. The library file is loaded as it is, with no modifications of the source code. Since, the application extracts mass information from files, the Object_Instance constructor has been altered to open, parse and close the files. For the second mode of interaction a "wii avatar" record has been added to the library file. In the case of this object, the mass is set manually. The Ontology world and the object library are loaded in the createScene function of the VirtualRoomApp class.

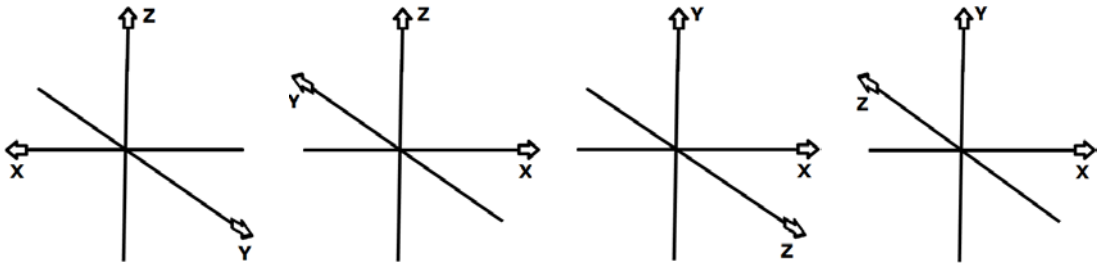
5.6 Wii Remote Input Treatment and Behaviour

Up to this point, the focus of the discussion has been on the functionality of the components of the prototype system with references to the Wii Remote. This section clarifies how the Wii Remote data has been treated, in order to be utilised by the system.

5.6.1 Wii Remote Coordinate System Transformation

The Wii Remote being a physical object in space has a coordinate system that is different from the 3D world coordinate system. The positive x axis direction is towards the left and the positive z axis direction is upwards (Earth gravitational vector) and the y positive axis is pointing forward to the direction of the viewer. When stationary, the acceleration values extracted with the aid of the WiiYourself! library are a normalised vector pointing up, due to gravitational acceleration g.

In order to use the controller correctly into the virtual world, its coordinate system has to be transformed into the world coordinate system. This is achieved through a series of axis rotations performed inside the frameEnded function. The rotations can be seen in Figure 5.1 below.



Wii coordinate system in space | Rotate by 180 degrees on z axis | Rotate by 90 degrees on x axis | Negate z axis – World coordinate system

Figure 5.1 Wii Remote to World coordinate system transformation

5.6.2 Orientation

The estimation of the orientation of the Wii Remote in space is performed in the frameEnded function. This is achieved by comparing the orientation of the controller axis (VirtualRoomFrameListener member variable Orient) against the axis defined by the gravitational normal (member variable ref).

When the Motion Plus is present, it is possible to retrieve more accurate acceleration readings. Its values can be used to rotate or update the last known acceleration-based orientation estimate.

Yaw is only possible when the Motion Plus and IR are present, since it is a rotation along the gravitational axis. In this application only the Motion Plus is used for yaw estimations.

5.6.3 Adding more to the simulation

Apart from estimations of the orientation, the data retrieved from the controller can be exploited to add more realism to the scene. During the design process, it was realised that there are two approaches to the implementation of the second mode of interaction (i.e. the Wii Remote as a tool to push objects). The first approach involves retrieving the acceleration of the Wii Remote on contact points when collision occurs and applying the corresponding force on the object hit. The second approach, which takes advantage of the Motion Plus extension, involves direct movement mapping and applying the accelerations the controller met in real life to the simulation. The former has the drawback that, if one wishes to perform a vertical hit, the controller will have to be vertical and it will lose visibility of the IR beacon. The latter, on one hand, surpasses this problem. On the other hand, it will result in the drifting of the object controlled, due to sensor noise and other factors. Bearing this in mind, the second approach has been abandoned.

On object impact, a Wii Remote acceleration-based force (F = m*a) is applied at the contact point allowing hitting objects in both modes of interaction (in VirtualRoomApp collision function). In order to avoid applying an extra force of gravity on the object hit, the gravity is eliminated from the acceleration vector. When an object is released, the acceleration and Motion Plus angular velocity values are set to the object allowing realistic throwing (VirtualRoomFrameListener function B_Pressed).

5.6.4 Calibration and Filtering

The gyroscope sensors of the Motion Plus have rest values that vary from sensor to sensor. As such, without calibration of these values, the extension reports continuous rotations that cause significant drift (Wiibrew, 2009a). A very simple method is to set the rest values to the current values on a button press, when the device is stationary. WiiYourself! did not provide calibration, so this functionality was implemented and added to it.

In order to smooth the motions of the objects, a low-pass filter has to be utilised to filter out noise and erratic movement. This is implemented as a first order Infinite Impulse Response (IIR) filter. A detailed explanation of this is beyond the scope of this document. In this simple form, the filter is given by the formula (Wikipedia, 2009b), (Adafruit, 2009):

$$y[n] = x[n-1] + \text{alpha} * (x[n] - x[n-1])$$

An alpha value of 0.1 gives 10% smoothing of the signal.

5.6.5 Raytracing

Raytracing with the OGRE 3D ray scene queries was used for two purposes in the Virtual Room application; performing selection of objects and object depth movement in the scene.

When the Wii Remote B button is pressed, while an object is not held, a ray is created with IIR treated IR values of the Wii Remote. Then, a ray scene query is executed, in order to find what lies in the end of the ray (on x,y). If the result is not a static geometry, but a movable object, then this object is selected.

When an object has been selected and is dragged across the room, a different query is set, in order to allow forward and backward movement of the object. This query involves z value estimation from the motion of the controller in space. The z value is estimated with the aid of the Wiiuse library and then filtered (in frameEnded). In short, the value is calculated by taking into account the distance of the controller from the IR beacon.

On selection, the z coordinate values of the object and the Wii Remote are saved. A plane facing the camera and containing the position of the object is created and the object is displaced by the relative displacement of the Wii Remote z from its original value. This results in the ability to drag the object in front of the camera in every direction.

5.6.6 Behaviour Tweaks

The options to lock the orientation and lock the position of an object are nothing but tweaks in the behaviour of the Wii Remote. At the moment that the user locks the orientation of the manipulated object by a controller button press, the current orientation is saved. The object is updated using this constant value, until it is unlocked. If an object is locked in position, the ray scene query performing object selection treats it as if it was a static geometry and discards it as not selectable. NB: The lock position functionality has been commented out in this version of the code and is not provided at the moment.

6. Results and Discussion

This chapter discusses the implementation results and various issues that affected development of the prototype system.

6.1 OGREODE

One of the major setbacks during development was the use of the OGREODE wrapper. This project has been orphaned for a long time, so that support of recent versions of OGRE 3D and ODE has been achieved through third party patches. Several features are not implemented or are under-implemented, as in the case of the convex geometry and the step handlers. The distribution comes with demos, from which only one does not fail on the library compiled with the recent versions of OGRE and ODE. Moreover, it causes numerous exceptions to the main OGRE 3D application that uses it.

Apart from these, the documentation of the API varies from limited to none. dOxygen has been used to compile some useful information by its headers and source code. The only supplied documentation comes in the form of tutorials on the project web site and it has been discovered that it provides erroneous information about certain issues, such as collision detection. The example code snippet provided rejects contacts between bodies that are joined. Since contacts are joints, this results in only one contact in use per pair of objects making objects fall through each other.

6.2 Wii Remote and Motion Plus

The Wii Remote has been successfully integrated into the interface of the Virtual Room application. Motion Plus support fulfils its role of tracking orientation, but it is used as a secondary device and its function to complement the Wii Remote could be improved. For instance, the yaw is not tracked accurately, as the extension is only taken into account when the current orientation cannot be obtained from the accelerometers, due to Wii Remote movement.

Calibration of the Motion Plus extension is as mentioned quite simplistic. This has the disadvantage that noise or accidental rotations might make the value inaccurate, so more than one try might be needed. However, the success rate during development was high. A better solution would an averaging of several samples.

Using the Infinite Impulse Response filtering proved to be highly beneficial. Filtering treating the various values extracted from the WiiYourself!, such as the IR values, has significantly improved the usability of the interface.

6.3 WiiYourself!

WiiYourself! at the moment provides experimental support for the Motion Plus extension, thus it is bound to be problematic. One of the first problems detected was the fact that Motion Plus connection was failing. It is suspected that the problem is caused by a threading issue in the library. There has been a workaround to surpass this, by manually setting the report type in the user thread, directly after enabling the extension. Now, the connection is successful most of the times, but it is not possible to start the application with Motion Plus

connected onto the Wii Remote. Another bug in the library found during development is that the button parsing code did not mask out unused bits of the incoming stream of data, causing the button callbacks to be triggered continuously, even if the status of the buttons has not been changed. Applying the appropriate mask inside the library solved the problem.

6.4 Other issues

Virtual Room is partially compatible with the Untidy Room application at the moment. The reason behind this was the inability to convert the geometry description files from the Untidy format to the format supported by ODE and the OGREODE problematic support for the convex geometry primitive. However, it provides the framework to accommodate the objects and Libraries and in the case that a suitable geometry format is finally found, adding functionality to parse it is not considered to be a difficult task.

When it comes to the physical simulation of the Virtual Room world, it can be said that it is fairly stable. The behaviour of objects is quite acceptable, that is there are no interpenetrations during collisions. With the current OGREODE setup, the simulation world is not as realistic as desired, because of the arbitrarily chosen scaling factors.

7. Conclusion and Future Work

This project has been an attempt to utilise the Nintendo Wii Remote and its latest extension, the Motion Plus, as a tool of interaction in a virtual dynamic world. After the design and development process of the proposed system prototype, it has been realised that the Wii Remote, together with Motion Plus can offer much to the interface design of such virtual worlds. Although imperfect, it can create paradigms of interaction that can be considered to be highly intuitive and add up to the over user experience of a virtual world. In addition, experimenting with the real world accelerations as inputs has shown that the device can be exploited quite extensively, in order to add more realism to such a scene.

The project has also given the opportunity to comprehend a variety of issues more thoroughly. These issues include the capabilities and as well as the limitations of the Wii Remote, as a piece of hardware, the process of integrating a physics engine into a rendering system for the a simulation of a dynamic world. Many ideas have been formulated during the entirety of the process, concerning not only the improvements of this specific application, but also the creation of new ones exploring the potential that the Wii Remote offers in different ways.

7.1 Future Work

The application built for this project is an experimental prototype at the moment. There are many ways that the prototype can be enriched or improved.

One major step would be to migrate from ODE/OGREODE to another physics engine with more capabilities, such as Bullet. Another issue that has to be tackled is the correction of the scaling factors of forces and the time stepping, so that the simulation is more realistic.

As explained in the results, the Motion Plus was used as a companion to the Wii Remote to compensate for accuracy limitations. An interesting approach could be to track full motion of the controller in space and to minimise problems, such as drifting. Moreover, it would be desirable to eliminate the need of a mouse to perform certain actions into the scene, thus enabling full interface interaction with the Wii Remote.

Interface-wise, there could be a few changes made to the method that objects enter the scene. Instead of using a Listbox with text entries, a graphical inventory of objects could be used, as seen in many computer games throughout the years.

References

ACRES T., 2009. Impressions: Wii Motion Plus. *SystemLink* [Online blog], 15 June. Available at: http://systemlinkblog.blogspot.com/2009_06_01_archive.html [Accessed 24 September 2009].

ADAFRUIT, 2009. *Wii MotionPlus protocol...Let us hack...* [Online]. Available at: http://forums.adafruit.com/viewtopic.php?f=8&t=11594&start=0 [Accessed 24 September 2009].

ARMITAGE M., n.d. *Nintendo "Controlling" the Evolution* [Online]. Available at: http://www.mywii.com.au/GuideDetail.aspx?id=250 [Accessed 24 September 2009].

BERNARDES, J., et al., 2009. Integrating the Wii controller with enJine: 3D interfaces extending the frontiers of a didactic game engine. *Comput.Entertain.*, 7 (1), 1-19.

BOWMAN, D.A., and HODGES, L.F., 1997. An evaluation of techniques for grabbing and manipulating remote objects in immersive virtual environments. *In*: SI3D '97: *Proceedings of the 1997 symposium on Interactive 3D graphics.* Providence, Rhode Island, United States, New York, NY, USA: ACM, pp. 35-38.

BRADSHAW, D., and NG, K., 2008. Tracking conductors hand movements using multiple Wiimotes. *In*: 2008 *International Conference on Automated Solutions for Cross Media Content and Multi-Channel Distribution*, 17-19 Nov, 2008. Piscataway, NJ, USA: IEEE, pp. 93-99.

CANT, R.J., and LANGENSIEPEN, C.S., 2009. Methods for Automated Object Placement in Virtual Scenes. *In: UKSIM '09: Proceedings of the UKSim 2009: 11th International Conference on Computer Modelling and Simulation,* Washington, DC, USA: IEEE Computer Society, pp. 431-436

COCHARD, D. and PHAM, K., n.d. *Navigation in a 3D environment: Using the Wiimote to map natural gestures* [Online]. Available at: video.davidcochard.com/projects/CS6456/Cochard_Pham_report.pdf [Accessed 24 September 2009].

FUNK, J., 2009. *Wii Remote Controls Giant Robot Arm* [Online]. The Escapist. Available at: http://www.escapistmagazine.com/news/view/93072-Wii-Remote-Controls-Giant-Robot-Arm [Accessed 24 September 2009].

GALLO, L., DE PIETRO, G. and MARRA, I., 2008. 3D interaction with volumetric medical data: experiencing the Wiimote. *In: Ambi-Sys '08: Proceedings of the 1st international conference on Ambient media and systems, Quebec, Canada,* Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering; ICST, Brussels, Belgium, Belgium: ICST, pp. 1-6.

GL.TTER, 2008. *WIIYOURSELF! – A native C++Wiimote library by gl.tter* [Online]. Available at: http://wiiyourself.gl.tter.org/ [Accessed 24 September 2009].

GUO, C., and SHARLIN, E., 2008. Exploring the use of tangible user interfaces for human-robot interaction: A comparative study. *In*: *26th Annual CHI*

Conference on Human Factors in Computing Systems, CHI 2008, 5-10 April, 2008. Florence, Italy: Association for Computing Machinery, pp. 121-130.

HAY, S., NEWMAN, J., and HARLE, R., 2008. Optical tracking using commodity hardware. *In*: *7th IEEE International Symposium on Mixed and Augmented Reality 2008*, 15-18 September, 2008. Piscataway, NJ, USA: IEEE, pp. 159-60.

JUNKER G., 2006. *Pro OGRE 3D Programming*. Berkeley, CA: APress.

LAFOREST M., n.d. *Wiiuse* [Online]. Available at: http://www.wiiuse.net/ [Accessed 24 September 2009].

LEE, J.C., 2008. Hacking the Nintendo Wii remote. *IEEE Pervasive Computing*, 7 (3), 39-45.

MANN P., 2009. *Motion-sensing game controllers explained* [Online]. Hexus. Available at: http://gaming.hexus.net/content/item.php?item=19458&page=2 [Accessed 24 September 2009].

NINTENDO, 2008. *Introducing Wii Motion Plus, Nintendo's upcoming accessory for the revolutionary Wii Remote* [Online]. Available at: http://www.nintendo.com/whatsnew/detail/eMMuRj_N6vntHPDycCJAKWhEO9zBvyPH [Accessed 24 September 2009].

OGRE TEAM, 2000. *OGRE* [Online]. Available at: http://www.OGRE3d.org/ [Accessed 24 September 2009].

OGRE WIKI, 2008. *Documentation Architecture* [Online]. Available at: http://www.ogre3d.org/wiki/index.php/Documentation_Architecture [Accessed 24 September 2009].

OGRE WIKI, 2009a. *Basic Tutorial 1* [Online]. Available at: http://www.ogre3d.org/wiki/index.php/Basic_Tutorial_1 [Accessed 24 September 2009].

OGRE WIKI, 2009b. *Basic Tutorial 3* [Online]. Available at: http://www.ogre3d.org/wiki/index.php/Basic_Tutorial_3 [Accessed 24 September 2009].

OGRE WIKI, 2009c. *Basic Tutorial 4* [Online]. Available at: http://www.ogre3d.org/wiki/index.php/Basic_Tutorial_4 [Accessed 24 September 2009].

OGRE WIKI, 2009d. *Basic Tutorial 6* [Online]. Available at: http://www.ogre3d.org/wiki/index.php/Basic_Tutorial_6 [Accessed 24 September 2009].

OGRE WIKI, 2009e. *Basic Tutorial 7* [Online]. Available at: http://www.ogre3d.org/wiki/index.php/Basic_Tutorial_7 [Accessed 24 September 2009].

OGRE WIKI, 2009f. *Simple Text Output* [Online]. Available at: http://www.ogre3d.org/wiki/index.php/Simple_Text_Output [Accessed 24 September 2009].

POUPYREV, I., WEGHORST, S. and FELS, S., 2000. Non-isomorphic 3D rotational techniques. *In*: CHI '00: *Proceedings of the SIGCHI conference on Human*

factors in computing systems. The Hague, The Netherlands, New York, NY, USA: ACM, pp. 540-547.

PEEK B., 2007. Managed Library for Nintendo's Wiimote. *Coding4Fun*. [Online blog], 14 March. Available at: http://blogs.msdn.com/coding4fun/archive/2007/03/14/1879033.aspx [Accessed 24 September 2009].

SMITH R., 2006. *Open Dynamics Engine v0.5 User Guide* [Online]. Available at: http://www.ode.org/ode-latest-userguide.pdf [Accessed 24 September 2009].

WIIBREW, 2009a. *Wiimote/extension controllers* [Online]. Available at: http://wiibrew.org/wiki/Motion_Plus [Accessed 24 September 2009].

WIILI PROJECT, n.d. *WiiLi Wiki* [Online]. Available at: http://www.wiili.com/index.php/Motion_analysis [Accessed 24 September 2009].

WIKIPEDIA, 2009a. *Wii Remote* [Online]. Available at: http://en.wikipedia.org/wiki/Wii_Remote [Accessed 24 September 2009].

WIKIPEDIA, 2009b. *Infinite Impulse Response* [Online]. http://en.wikipedia.org/wiki/Infinite_impulse_response Available at: [Accessed 24 September 2009].

XU, K., STEWART, J., and FIUME, E., 2002. Constraint-based automatic placement for scene composition. *In: Graphic Interface 2002*, 27-29 May, 2002. Calgary, Alta., Canada: Canadian Information Processing Society, pp. 25-34.

Bibliography

BLUNDELL B.G., 2008. *An Introduction to Computer Graphics and Creative 3-D Environments*. London: Springer.

HEARN, D. and BAKER M.P., 1997. *Computer Graphics C Version*. 2nd ed. New Jersey: Prentice-Hall.

JUNKER G., 2006. *Pro OGRE 3D Programming*. Berkeley, CA: APress.

SMITH R., 2006. *Open Dynamics Engine v0.5 User Guide* [Online]. Available at: http://www.ode.org/ode-latest-userguide.pdf [Accessed 24 September 2009].

WIIBREW, 2009a. *Wiimote/extension controllers* [Online]. Available at: http://wiibrew.org/wiki/Motion_Plus [Accessed 24 September 2009].

WIIBREW, 2009b. *Wiimote* [Online]. Available at: http://wiibrew.org/wiki/Wiimote [Accessed 24 September 2009].

A. Appendix A – User Manual

1. Wii Remote Connection Instructions

The connection steps many vary in different Bluetooth stacks and drivers. Sometimes the procedure has to be repeated more than once for a successful pairing of the Wii Remote with the Bluetooth enabled PC, because the device exits discoverable mode before the pairing has been achieved.

- To enter discoverable mode, keep buttons 1 and 2 pressed, so that the LEDs of the Wii Remote flash.
- Add a new device in Bluetooth devices, while the controller is in discoverable mode. It will appear as a standard HID device named RVL-CNT-01.
- If asked, choose pairing without a PIN.

2. Required equipment for full functionality

- A Nintendo Wii Remote
- A Sensor bar
- A Motion Plus extension (optional)

3. How to run

- Make sure that the Motion Plus extension is not connected to the Wii Remote before execution.
- On execution, an OGRE 3D requester will appear, in order to set preferences, such as the rendering system (OpenGL or DirectX), anti-aliasing level, full screen mode, resolution and more.
- When the scene is initialised and Wii Remote is not present or not paired with Windows properly, the application waits until a device becomes available. The application sets the second, third and fourth LED of the device on.
- After rendering starts, the Motion Plus can be plugged in. The status of the Motion Plus availability is displayed in the top left corner of the screen. Sometimes, the Motion Plus is not detected on first attempt.
- It is advisable to calibrate the device at rest, when the Motion Plus is enabled, to ensure greater accuracy.

4. Application Controls

D Pad Up	Move forward
D Pad Down	Move backward
D Pad Left	Move to the left
D Pad Right	Move the right
A	For use in the GUI (Select item in the list/Press Add button/Drag list box/Minimise list box (double A on the list title bar)
B	(1st mode only) Select/Delesect object. Press once to select object in scene, move the object, press again to drop item
1	1st mode of interaction
2	2nd mode of interaction
+	(Lock/unlock selected object's position)
-	Lock/unlock selected object's orientation

Home	Calibrate Motion Plus at rest. If objects seem to drift, recalibrate.
Mouse	Free camera look

B. Appendix B – Screenshots

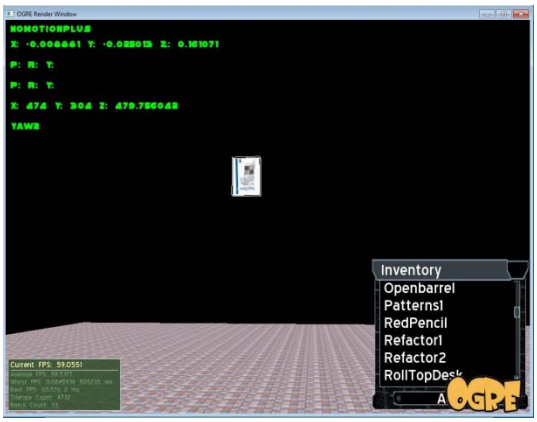


Figure B.1 Mode 1

Figure B.2 Mode 2

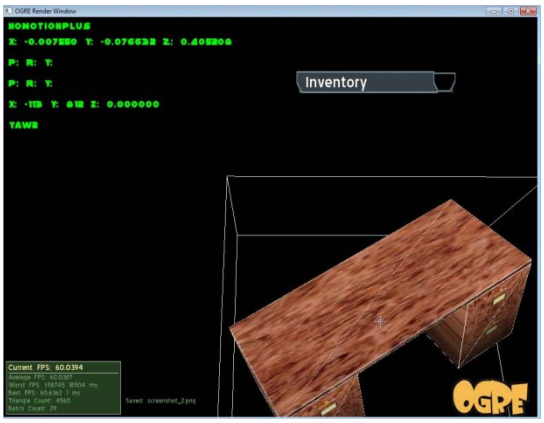


Figure B.3 Rotating an object

Figure B.4 Placement

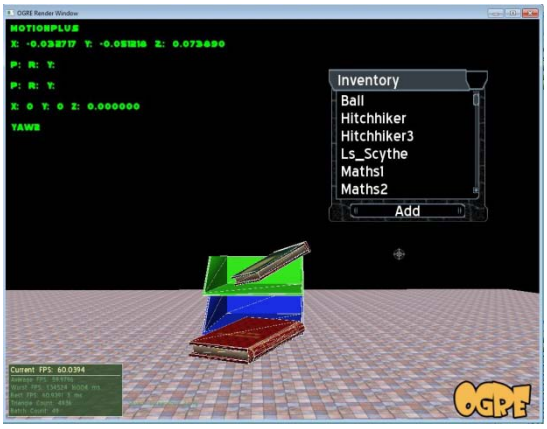


Figure B.5 Placement

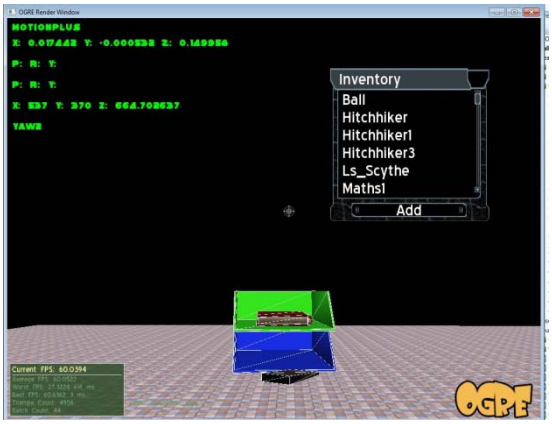


Figure B.6 Placement

C. Appendix C – Untidy room application sample input files

```
Woodenbucket 125 1 size medium Woodenbucket.txt Woodenbucket.mesh Bucket
workChair 401 2 size medium age modern simplechairplus.txt simplechair.mesh Chair
SquareWasteBin 601 2 colour grey age modern basicwastebinplus.txt basicwastebin.mesh Wastebin
paper1 602 1 colour white creased_paperplus.txt creased_paper.mesh Paper
paper3 604 1 colour white scroll.txt scroll.mesh Paper
markerpen 512 2 colour blue size medium bluemarkerplus.txt bluemarker.mesh Pen
redmarkerpen 5122 2 colour red size medium bluemarkerplus.txt bluemarker.mesh Pen
greenmarkerpen 5123 2 colour green size medium bluemarkerplus.txt bluemarker.mesh Pen
yellowmarkerpen 5124 2 colour yellow size medium bluemarkerplus.txt bluemarker.mesh Pen
blackmarkerpen 5125 2 colour black size medium bluemarkerplus.txt bluemarker.mesh Pen
purplemarkerpen 5126 2 colour purple size medium bluemarkerplus.txt bluemarker.mesh Pen
stapler 514 1 size medium basicstaplerplus.txt basicstapler.mesh Stapler
Desktidy2 513 1 size medium desktidy2plus.txt desktidy2.mesh Desktidy
Hitchhiker 101 2 size medium content science basicbook1plus.txt basicbook1.mesh Book
pencilEraser 511 1 size medium basiceraserplus.txt basiceraser.mesh Eraser
SIGGRAPH 504 2 size medium content science basicmagplus.txt basicmag.mesh Magazine
SIGGRAPH2 5042 2 size medium content science basicmagplus.txt basicmag.mesh Magazine
SIGGRAPH3 5043 2 size medium content science basicmagplus.txt basicmag.mesh Magazine
SIGGRAPH4 5044 2 size medium content science basicmagplus.txt basicmag.mesh Magazine
SIGGRAPH5 5045 2 size medium content science basicmagplus.txt basicmag.mesh Magazine
SIGGRAPH6 5046 2 size medium content science basicmagplus.txt basicmag.mesh Magazine
SIGGRAPH7 5047 2 size medium content science basicmagplus.txt basicmag.mesh Magazine
SIGGRAPH8 5048 2 size medium content science basicmagplus.txt basicmag.mesh Magazine
RedPencil 508 1 colour red redPencilplus.txt redPencil.mesh Pencil
BriefHistory6 102 2 size medium content science basicbookplus.txt basicbook.mesh Book
Sword2hand 111 1 size medium sword1.txt sword1.mesh Sword
shield 112 1 size medium shield.txt shield.mesh Shield
Ls_Scythe 120 1 size medium Ls_Scythe.txt Ls_Scythe.mesh Scythe
Openbarrel 121 1 size medium openbarrelcontainer.txt openbarrelcontainer.mesh Barrel
RollTopDesk 200 2 size medium age victorian basicdeskplus.txt basicdesk.mesh Desk
OldBookshelf 300 2 size medium age victorian basicbookshelfplus.txt basicbookshelf.mesh Bookshelf
mydesklamp 510 2 size medium age modern basicdesklampplus.txt BasicDeskLamp.mesh Lamp
paper2 603 1 colour white crumpled_paperplus.txt crumpled_paper.mesh CrumpledPaper
Ball 604 1 colour white crumpled_paperplus.txt crumpled_paper.mesh Toy
ThysYule 501 2 colour blue content earlymusic cdcasenohingeplus.txt cdcasenohinge.mesh CDCase
Hitchhiker1 101 2 size medium content science basicbook2plus.txt basicbook2.mesh Book
Vionnet1 103 2 size large content fashion basicbook2plus.txt basicbook7.mesh Book
Vionnet2 1032 2 size large content fashion basicbook2plus.txt basicbook7.mesh Book
Vionnet3 1033 2 size large content fashion basicbook2plus.txt basicbook7.mesh Book
Vionnet4 1034 2 size large content fashion basicbook2plus.txt basicbook7.mesh Book
Vionnet5 1035 2 size large content fashion basicbook2plus.txt basicbook7.mesh Book
Vionnet6 1036 2 size large content fashion basicbook2plus.txt basicbook7.mesh Book
Refactor1 104 2 size medium content science basicbook2plus.txt basicbook4.mesh Book
Refactor2 104 2 size medium content science basicbook2plus.txt basicbook4.mesh Book
Maths1 105 2 size medium content science basicbook2plus.txt basicbook6.mesh Book
Maths2 105 2 size medium content science basicbook2plus.txt basicbook6.mesh Book
HP 513 1 size medium basiccalculatorplus.txt basiccalculator.mesh Calculator
Patterns1 106 2 size large content science basicbook2plus.txt basicbook5.mesh Book
Patterns2 106 2 size large content science basicbook2plus.txt basicbook5.mesh Book
Hitchhiker3 101 2 size medium content science basicbook1plus.txt basicbook1.mesh Book
Mymug 507 2 size medium colour red Basicmugplus.txt Basicmug.mesh Mug
Mybluemug 5071 2 size medium colour blue Basicmugplus.txt Basicmug.mesh Mug
wii 800 1 size small WiiAvatar.txt WiiAvatar.mesh Avatar
```

Figure D.1 Library sample file