IST-2003-511592 STP

MICOLE
Multimodal collaboration environment for inclusion of visually impaired children

Specific targeted research project

Information society technologies

Deliverable D19 – Specialised User Software Toolkits

Due date of deliverable: 28.02.2007
Actual submission date: 02.03.2007
Revision date: 15.10.2007
Version 1.1

Start date of project: 1.9.2004 Duration: 36 months

Name of the partner responsible for the deliverable: Reachin Technologies (RIT)

Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)		
Dissemination Level		
PU	Public	**X**
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Authors:

Ida Olofsson, RIT
Roope Raisamo, UTA

Table of Contents

1 Introduction

As a part of the dissemination activities in the MICOLE project a Software Toolkit has been developed and the package and is now being distributed. The main goal of creating the Software Toolkit has been to create a package that can help developers to build applications that can be used by both sighted and visually impaired users. The Software Toolkit contains tools to speed up the development of new multi-modal 3D applications.

Reachin Technologies task has been to create this Software Toolkit and handle its dissemination. This document describes the contents of the Software Toolkit and how it packaged and delivered to the users.

One intention of the Software Toolkit is to provide tools for developers to create haptic applications that with a small effort can be adapted to be used by visually impaired and to bridge the gap between sighted and visually impaired users.

2 Specialised User Software Toolkits

The toolkit contains the following tools developed within the scope of the MICOLE project:

- A customized haptic menu for visually impaired.
- A library with predefined force effects to simplify future development efforts.
- 3D audio package for locating objects and paths in three dimensions.

The haptic menu enables construction of user interfaces which are suitable for both sighted and visually impaired. The predefined force effects are suitable when developing applications for visually impaired. The force effects can be used either to give information to the user or to allow new interaction possibilities.

The 3D audio package enables the user to develop applications with sound sources in three dimensions when aiding the user in advanced navigation and guidance tasks. Combined with haptic feedback and 2D/3D graphics, 3D audio enables the developer to create multi-modal applications with more realism and immersion. Combining these three components of the Software Toolkit greatly simplifies the development of applications which can be used by both sighted and visually impaired users.

2.1 3D Audio package

The audio package has been developed within the scope of the MICOLE project to simplify the development process of 3D audio in virtual environments. For visually impaired users the audio and haptic feedback are two very important information channels when using computer applications. The features in the audio package have been designed to support a broad range of possible usages.

The 3D audio package contains features for creating audio sources and placing them in a three dimensional environment. The contents of a 3D audio source can be loaded from a file, but there are also features for creating mathematically calculated sounds such as sine waves, pulse waves and saw tooth waves.

Each virtual scene contains a listener. The listener represents the position of the user in the scene and the 3D property of a sound is always based on the position of the listener. The listener can be positioned anywhere in the scene and thus enabling functionality for so called “ears-in-hand“ where the user can position the listener, for example, at the device position and when the user moves the device in the scene he or she can listen at different positions.

The implementation of the sound has been done with OpenAL. At first the FMod sound library was considered and although it has more features and more complex sound calculations than OpenAL, the cost of FMod was too high for general use. The decision was discussed among all partners and we agreed on using OpenAL.

The implementation of the sound package has two classes that take care of the actual rendering of the sound, ALSoundManager and ALSound. The ALSoundManager has functionality for initialising OpenAL and managing the sound sources. It also takes care of handling the Listener in the scene. ALSound is the base class for an OpenAL sound source. It is managing and creating sound buffers.

The respective sound functionality has also been given VRML and python interfaces so that the user can easily use the 3D audio features in a python script or position the sound sources and the listener in a VRML file. The actual VRML and python interfaces for the 3D sound package are implemented in the Reachin API.

By combining the features in the audio package together with haptics the developer can enhance the feedback to the user by for example letting the device velocity, device position or contact forces control the audio feedback.

2.2 ForceModel library

The ForceModel library contains ready to use ForceModels which can be used to improve and enhance haptic interaction.

A ForceModel is a basic haptic effect that controls the device directly without geometrical objects with surfaces. The ForceModels can be seen as building blocks for haptic interaction and navigation and they give the developer additional possibilities to communicate with the user through the sense of touch. One example of such an effect is vibration. The ForceModels can be combined with other ForceModels or surfaces to create more advanced haptic behaviour.

The haptic effects can either have a position in the world (global) like a magnet which attracts and holds the device to a specified location in the world. It can also be a space-free effect (local) which affects the device independently of the position and rotation of the device. One example of such an effect is viscosity, which simulates a small weight attached to a spring in the tip of the device.

2.2.1 Animated ForceModels

The “tape recorder” metaphor allows the ForceModels to be started, stopped and animated over time to create very interesting haptic behaviour. All ForceModels can in addition of having infinite duration also be specified to play over a specified number of seconds. They can also be looped over this duration with a specified idle duration where there is no force output, which creates a pulsing behaviour in the ForceModel. One other possibility is to increase or decrease the magnitude of the ForceModel over a specified time, and ease in or ease out the force output.

The time control is a very powerful feature to create applications that communicate solely with haptic feedback and audio. For example, one-second vibration in the device could indicate to a visually impaired user that another user is requesting contact. It can also be used to give information about where in the scene a specific object is placed by creating a magnetic force which draws the user towards the object.

2.2.2 Included ForceModels

Below is a short description of the ForceModels that are currently in the Software Toolkit. All of these can be animated over time, as described above, and combined.

Impulse	(local) gives a constant force to the device and pushes it in the specified direction.
SineWave	(local) gives a sinusoidal force output to the device with the specified direction, amplitude and frequency
SawWave	(local) gives a saw wave force output with the same parameters as a SineWave.
SquareWave	(local) gives a square wave force output with the same parameters as a SineWave.
Explosion	(global) a sinus wave which has a position and an effect distance in the world. The sinusoidal will decrease with the distance to the explosion centre.
Magnet	(global) attract the device to a position in the world with a specified effect distance. The effect is strong near the centre and weak at the perimeters.
RepellingMagnet	(global) repels the device from a position in the world to the specified effect distance.
Spring	(global) attracts the device to a position with a spring force that is weaker near the centre and increases in strength with the distance from the spring origin.
Viscosity	(local) simulates that a small mass is attached to the tip of the device with a spring.

3 Haptic menu

The goal of this section is to specify how a haptic menu for visually impaired and sighted users could function, feel and look.

3.1 Target Group

The target group of the menu system is both visually impaired and fully sighted users, which can have a varying age and degree of experience with 3D haptic applications.

Since the target group is diverse the design process will focus on creating a menu for visually impaired which can also be used for sighted users.

3.2 Requirements

Navigation and selection in the menu system should be possible with only haptic and audio feedback, and as a result large emphasis should be made on distinct haptic properties to activate, navigate and select commands. Clear audio feedback for exploration should be implemented. The menu system should facilitate at least eight commands, but should be flexible in structure if less than eight commands is needed.

Since the intention of the menu is to be used by both novice and experienced users, it is important to facilitate intuitiveness and control for beginners and at the same time provide fast and accurate selection for more experienced users.

The implementation of the menu has been done in C++ with support for VRML and Python.

3.3 3D menus – Theoretical Background

Since there is not much literature on 3D haptic menus for both visually impaired and sighted users, this theoretical section is focusing on sighted users and haptic interaction in general. However, many principles for sighted users can be applicable to visually impaired users with additional audio feedback.

3.3.1 Placement of Menus and Occlusion

One challenge in incorporating WIMP (Window, Icon, Menu, Pointing device) components in a 3D environment is that it converts a 2D task to a 3D task. This can make interaction more difficult since the user has to locate a floating 2D menu in a 3D space. If stereoscopic viewing is not used it becomes difficult to judge the correct depth of the menu if no additional depth cues are given. This transforms a 1 DOF task, choosing from a list, into a 6 DOF task, hence increasing the likelihood for errors and user frustration.

Instead of displaying the menus inside the virtual environment at all times it could be more efficient and intuitive to display menu items only when the user wishes to interact with them. One solution to the problem is to make the menu appear directly under the current proxy position when a user activates the menu. The proxy is the graphical representation of the input device, both in two and three dimensions. The menu should be hidden again after a selection has been made. This approach solves the depth problem for both sighted and visually impaired users.

3.3.2 Pie Menus for mouse input

A 2D pie menu has a circular design where menu options are arranged around the centre in radial sectors, where each sector corresponds directly to a command. The menu appears directly under the cursor which saves the distance needed to first move the cursor to a menu bar before activating the menu. A selection is made by pressing and holding the mouse button, dragging the cursor in a direction into a radial sector and releasing the mouse button.

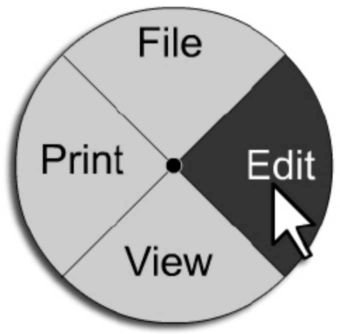


Figure 1 - *A two dimensional pie menu with four commands that pops up under the mouse pointer when activated.*

The philosophy behind pie menus are that directions and menu layout will get coded into the muscle memory with practice and hence reduce visual attention needed to make a menu selection, which is very desirable from a visually impaired user's perspective. This transition to rapid expert behaviour is not possible with linear menus where visual attention is needed even for experts. Pie menus have been shown to be measurably faster than traditional linear menus because of the large selection regions and the short and consistent movements needed to make a selection. There are different menu configurations, but it is common that pie menus have four, eight or twelve sectors for commands [1].

One version of a haptic pie menu with eight directions has been implemented and evaluated by Komerska and Ware [2]. The menu appears under the proxy when a user wishes to interact with it. They conclude that pie menus are up to 25% faster than their linear equivalents, and that pie menus are a very good alternative for selecting commands in 3D haptic applications.

Based on findings from the literature it was concluded that pie menus have the potential of being a very promising alternative technique for applications used by sighted users. It is also likely that they will be a very good alternative for visually impaired users with additional distinct audio and haptic feedback.

3.3.3 Conceptual Design, Activation and Selection

The menu can have between three and eight choices. Based on design principles from 2D pie menus it was concluded that it is not feasible with more than eight commands in one pie menu. The menu is invisible and non accessible from the working volume when not needed. The menu pops up at the current proxy position when activated.

Selection should be made with the button of the device to enforce control and reduce unnecessary static muscle strain like other selection techniques could produce.

3.3.4 Haptic Feedback

The proxy should be locked by a weak magnetic force to the menu plane when the menu is activated. A geometrical wall acts border constraint around the haptic part of the menu (selection shape). The geometrical wall constraint will help both sighted and visually impaired users to not go outside the menu and for the visually impaired users they can follow the border around the menu to explore what different choices can be made. The proxy can move unrestrictedly in the 2D haptic plane that is bounded by the selection shape. The corners of the geometrical border constraint correspond to a menu alternative.

In each corner, a localized haptic effect (magnet) is placed to indicate that a selection can be made. The magnet combined with a surface creates a haptic dimple in the surface which further indicates that a selection is possible. The radius of the magnet is approximately the size of the proxy. The effect is weak and indicating so that it does not disturb the user during e.g. his or her exploration of the menu.

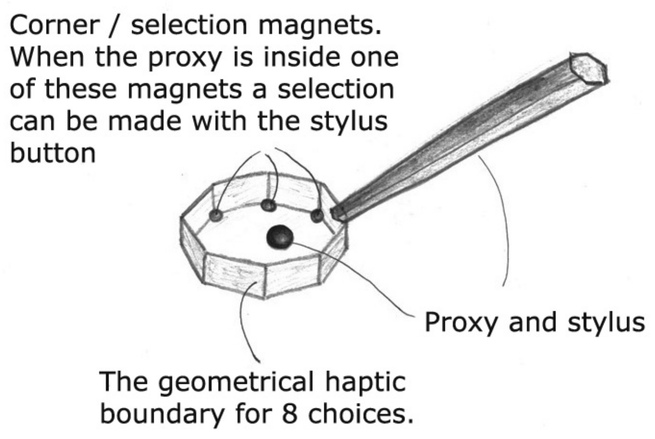

Corner / selection magnets. When the proxy is inside one of these magnets a selection can be made with the stylus button
Proxy and stylus
The geometrical haptic boundary for 8 choices.

Figure 2 *- The positioning of the magnets and the geometrical boundary of the menu.*

3.3.5 Audio Feedback

When the proxy is in a corner, (inside a corner magnet before a selection has been made) a sound file that is related to the menu option is played, preferably a pre-recorded voice that reads the menu choice. This helps the visually impaired user to explore the menu options by following the border of the menu and to listen to the options. When the user pulls away from the corner the sound will stop playing as soon as the device is outside the range of the selection magnet.

All audio feedback, such as the different sound that are played, should be designed and customized by the developer of the application to suite the needs of the application.

3.3.6 Visual Design

The form of the menu should visualise the number of choices that can be made. For example, a menu with three choices will have the form of a triangle and a menu with four choices will have the form of a square (see figure 3). The haptic representation of the menu shape will be of the same form as the graphical representation. This way a visually impaired user can explore the menu and get a quick overview of the number of choices that can be made. A sighted user can just look at the menu and will not need to explore it in the same way as a visually impaired user.

The menu labels should be separate from the haptic selection area to be able to make the geometry of the menu smaller and thus reduce the movements of the device when exploring the menu and make selections. This will reduce the muscle strain imposed on the user when using the menu.

The menu should have customizable visual attributes such as color, transparency and size of both the menu and the text for the different choices. If the developer changes the size of the actual menu geometry the haptic representation will also change size.

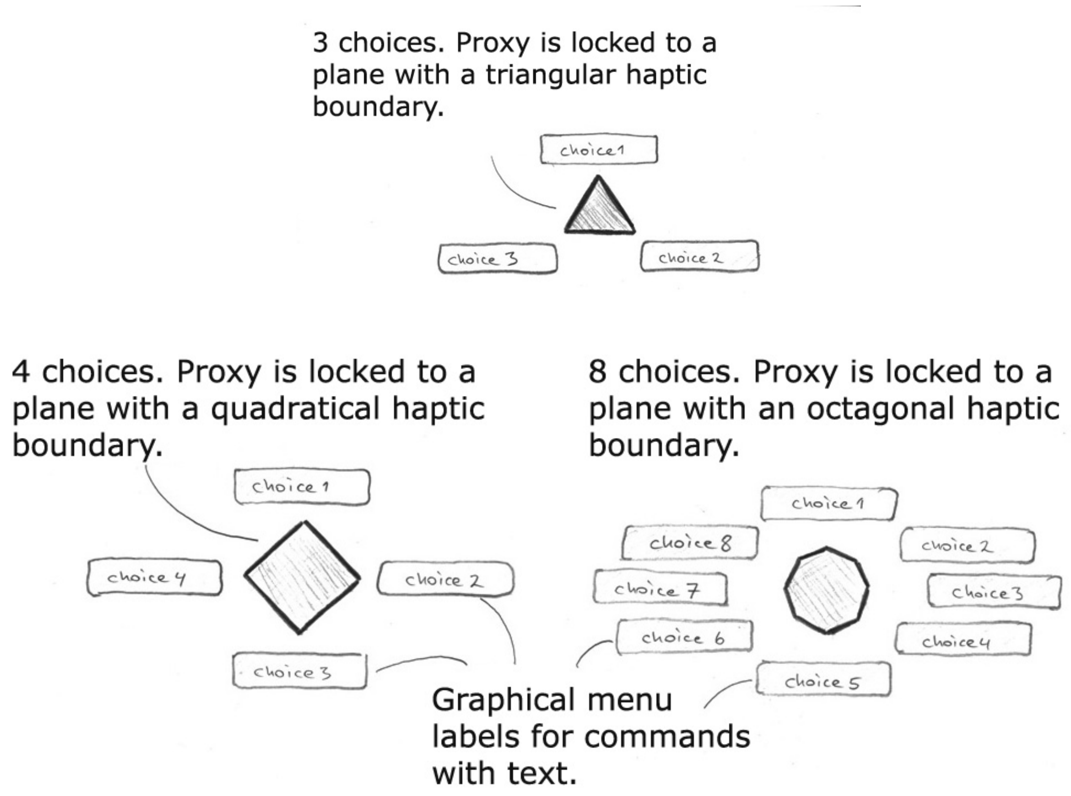
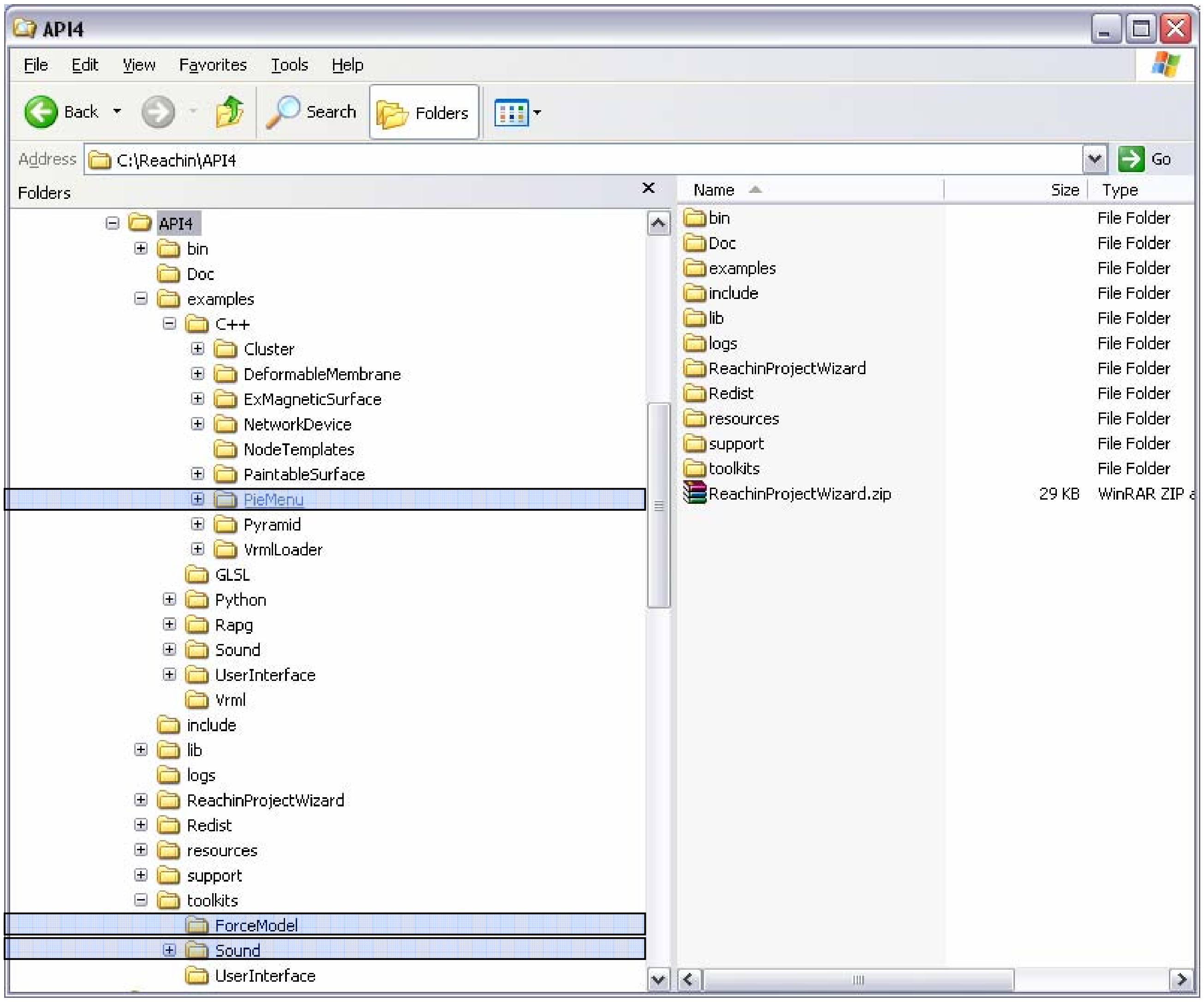

3 choices. Proxy is locked to a plane with a triangular haptic boundary.
choice1
choice 2
choice 3
4 choices. Proxy is locked to a plane with a quadratical haptic boundary.
choice 1
choice 2
choice 3
choice 4
8 choices. Proxy is locked to a plane with an octagonal haptic boundary.
choice 1
choice 2
choice3
choice4
choice 5
choice 6
choice 7
choice 8
Graphical menu labels for commands with text.

Figure 3 - *The graphical design of the menu. The shape of the menu will also be felt haptically.*

4 Dissemination and Packaging

The three parts of the Software Toolkit are bundled together with Reachin API and distributed together with Reachin API 4.2. The packages are built into the Reachin API installation package and contain only binaries and header files together with documentation and examples of how to use the components. One exception is the haptic menu where the source code will be supplied together with the API to let the developer modify the menu for their own needs.

Examples of how to use the parts of toolkit are supplied in the three different languages supported: C++, Python and VRML. The installer will install everything that is needed so that the user can get started quickly by just installing the toolkit and start running and modifying the examples. The documentation will consist of help files in digital format so that the user can look up the information he or she needs when it is needed.



C:\Reachin\API4

```
⊟ API4
  ⊞ bin
    Doc
  ⊟ examples
    ⊟ C++
      ⊞ Cluster
      ⊞ DeformableMembrane
      ⊞ ExMagneticSurface
      ⊞ NetworkDevice
        NodeTemplates
      ⊞ PaintableSurface
      ⊞ PieMenu
      ⊞ Pyramid
      ⊞ VrmlLoader
      GLSL
    ⊞ Python
    ⊞ Rapg
    ⊞ Sound
    ⊞ UserInterface
      Vrml
    include
  ⊞ lib
    logs
  ⊞ ReachinProjectWizard
  ⊞ Redist
  ⊞ resources
  ⊞ support
  ⊟ toolkits
      ForceModel
    ⊞ Sound
      UserInterface
```

Name	Size	Type
bin		File Folder
Doc		File Folder
examples		File Folder
include		File Folder
lib		File Folder
logs		File Folder
ReachinProjectWizard		File Folder
Redist		File Folder
resources		File Folder
support		File Folder
toolkits		File Folder
ReachinProjectWizard.zip	29 KB	WinRAR ZIP

***Figure 4** – The File Structure*

The functionality of the Software Toolkit is finished. The packaging and documentation is done and the three components are distributed together with Reachin API.

The target user groups of the Software Toolkit are predominantly researchers and developers within the research area of multi-modal applications who seek to combine haptics, graphics and audio output for both sighted and visually impaired users. In this case the target buyers are most likely to be the same as the target users since the haptic market is still mostly populated by researchers. Reachin's experience from this market is that it is difficult to sell development software to users that have not initiated the first contact by themselves hence traditional sales and marketing efforts are not very useful. The Software Toolkit requires a certain level of programming skills and a general knowledge about 3D programming.

The Software Toolkit will be distributed to Reachin's existing and new customers together with the Reachin API. This will be done through Reachin's reseller network. Users who buy a new copy of the API will get the Software Toolkit together with the Reachin API product box.

Reachin will have the overall all responsibility to keep and maintain the code for the Software Toolkit. Future development of the Software Toolkit will mostly be based on user requests but initially Reachin focuses on the further development of the predefined ForceModels.

5 References

[1] Shengdong Zhao and Ravin Balakrishnan. Simple vs. compound mark hierarchical marking menus. Proceedings of the 17th annual ACM symposium on User interface software and technology, pages 33–42, 2004.

[2] Rick Komerska and Colin Ware. A study of haptic linear and pie menus in a 3d fish tank vr environment. 12th International Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems (HAPTICS'04), pages 224–231, 2004.