ABSTRACT
TeaTracks is an audio sequencer for the visually impaired currently under development. It can be controlled throughout by a standard keyboard and every action spawns an auditory feedback. Such feedback is either a spoken reply by a text-to-speech engine or an auditory icon [1]. During the first phase of development, which result we are going to present, we examined different types of auditory display strategies.

Thanks to the UK's funding body for technology in disability, TechDis and the Higher Education Assistive Technology (HEAT) scheme, we have been able to conceptualise and create a working prototype in SuperCollider (SC)[2] which can already be used by blind or visually impaired students and which we will demonstrate.

1. INTRODUCTION
The idea to create an auditory user interface for a music and sound editor arose out of the need to look after a visually impaired student at Anglia Ruskin University. Although a few commercial audio sequencers have been scripted to enhance their accessibility by key commands and text-to-speech engines, we decided to start more or less from scratch in order to implement a true auditory display and make all features accessible.

Even in well scripted sequencers there are features left, which are unreachable for the visually impaired and hence unusable for them; a large and increasing amount of functionality is based on graphical user interfaces, such as graphs and curve editors, which are only intuitively usable by sighted users. In addition a pure text-to-speech approach, which we find in such programs when using commercial screen readers, can slow down the working process. Spoken text simply takes longer than perceiving an icon, so we are also considering and implementing specially designed earcons [3]. In short, we are attempting to create a true auditory display from the very foundations of the design of audio sequencing software.

2. DESIGN CONSIDERATIONS
Central points in the design are to investigate the practicalities of providing auditory feedback as keyboard controlled interaction to a non-sighted user. In order to realise a working and useable prototype in the time allowed by our initial research grant, we focused on basic operations, while keeping the implementation as open as possible to be able to expand it later on. At the current stage the following functionality is available: soundfiles can be imported into a pool; imported soundfiles can be placed on certain tracks; such objects can then be cut, pasted and moved on a timeline; a mix can be bounced to disk.

2.1. Auditory display
In order to communicate information in a short amount of time, the need to provide sounds or auditory icons [1] was obvious from the start. To use a visual analogy: following a time cursor on the screen is often more efficient than reading the time on display.

The challenge is that the software works upon the sequencing of audio files and therefore feedback sounds have to be differentiated from sound being edited. This is not trivial in contemporary music making considering that any given sound can become a musical element. One particular feature is that a sound file can represent itself upon playback. This provides ample scope for exploring the idea of mapping sound elements without having to rely entirely on metaphor or symbolism. An aspect during the development of this software is to keep the implementation of the auditory feedback as open as possible in order to give users the possibility to alter the sounds to their taste and needs, i.e. giving users the choice between Text-to-Speech feedback or auditory icons for each individual action. A custom auditory
display is advantageous because it does not require additional screen reading software, allowing greater flexibility. The experience for visually impaired users can be made much more immediate and customised.

Although interesting work has been done in this general area, there is comparatively little done on music software auditory displays: from the early research of William Gaver, including the definition of auditory icons, on the ‘sonic finder’ in the late 80s and work such as that of Elizabeth Mynatt et al [4] on ‘audio aura’, through the developments of earcons, and later spearcons in the work of Bruce Walker et al, up to recent work such as that of Tony Stockman on sonification of data for blind users or Rochesso et al on continuous auditory feedback.

2.2. Keyboard navigation

For the most common operations, like copy/paste, we settled on the widely used keyboard shortcuts, like 'control - c' and 'control - v'. Other shortcuts are assigned by using virtual layers, which are enabled by pressing a modifier or by tabbing to another layer. While the first method is employed for more often used commands, during working on the timeline for instance, the latter one is used when entering another level of the program, for instance selecting a soundfile.

On the timeline for instance the control modifier always relates to a soundfile object. The cursor keys left and right move the selected soundfile object, while control is being pressed. The same keys move the time cursor without modifier.

3. IMPLEMENTATION

TeaTracks has been developed in SuperCollider on Mac OS X. The choice was made, because of SC's object oriented approach and the wide range of possible audio manipulations, which will be of great use, in a later stage of the project. One of the authors had also previously implemented a direct binding to Apple's speech engine, which we could thus control directly from SC. So in fact the program is a library in SC, which has been put together to a stand alone application. The integration in SC also opens up possibilities to actually use some of its other features in the future. It could be for instance easily combined with other tools, which were made for visually impaired in SC. A basic visual interface has been realised in order to track down bugs more easily and to help (sighted) assisting persons to see what is going on in the application.

3.1. Flexible integration

In technical terms the auditory feedback was implemented according to a Model-View-Controller (MVC) scheme [5], which is used in object-oriented user interface development. Model refers to the data object, which corresponds its current state via a controller object to the view object, which in our case is the auditory display object. This approach is flexible and opens up the possibility to switch between different kinds of displays. One might for instance need an icon and a textual description in a learning phase of the software and could later on switch to hear the icon only. So also on the programmatic side auditory feedback takes over the role of a view.

All keyboard shortcuts are defined separately, so they can be changed to fit the needs and taste of the user. The keyboard responders have been defined as extensions of each functional class that they control and not in a view or graphical interface. This approach has been chosen to keep the whole program as independent from any graphical representation as possible.

Figure 2 shows the current implementation. The user interacts through a view and a keycode responder with the data, which is then displayed through a controller. Auditory feedback is provided by a static text-to-speech engine and a static audio server. Each text or sound icon can be disabled. Removing the controller would disable all sonic feedback.

3.2. SuperCollider library

As said the prototype has been implemented as a library in SuperCollider. The sequencer or multitracker is set up as a tree. A main class, called 'MultiTracker', holds a certain amount of tracks. Each track holds an undefined amount of soundfile-objects, the data object, which contains information, such as start time, duration and a reference to a soundfile object. Each of these building blocks has its own player, which is separated from the data object. This simple, but effective setup, also opens the door to various customised kinds of playing and arranging methods, which are not available on standard software packages. Algorithmic playback or arranging tools could be implemented in the future. In principal the sequencer can hold any kind of SC data, including code-snippets, which means that other features can be added easily. The library can be also used as a SC extension directly.
4. FINAL NOTE

The demonstration will present the solutions we arrived at when trying to implement an auditory display and show the state of the current development with a few examples reflecting the working flow of the application.

5. REFERENCES


