Journal of Neuroscience Methods xxx (2008) xxx–xxx

Contents lists available at ScienceDirect

Journal of Neuroscience Methods

journal homepage: www.elsevier.com/locate/jneumeth

APEX 3: a multi-purpose test platform for auditory psychophysical experiments

Tom Francart*, Astrid van Wieringen, Jan Wouters

ExpORL, Department of Neurosciences, Katholieke Universiteit Leuven, O & N 2, Herestraat 49 Bus 721, B-3000 Leuven, Belgium

ARTICLE INFO

Article history:
Received 28 November 2007
Received in revised form 16 April 2008
Accepted 18 April 2008

Keywords:
Auditory psychophysics
Behavioral experiments
Software
APEX
Research platform

ABSTRACT

APEX 3 is a software test platform for auditory behavioral experiments. It provides a generic means of setting up experiments without any programming. The supported output devices include sound cards and cochlear implants from Cochlear Corporation and Advanced Bionics Corporation. Many psychophysical procedures are provided and there is an interface to add custom procedures. Plug-in interfaces are provided for data filters and external controllers. APEX 3 is supported under Linux and Windows and is available free of charge.

1. Introduction

In general, behavioral experiments (e.g. psychophysical experiments or speech perception tests) are controlled by a computer. In most cases custom software is created for each new experiment. However, behavioral experiments have many parts in common. Appropriate stimuli are created and presented to a subject via a transducer, the subject responds via an interface to a computer and the results are stored for analysis. Developing software to perform a specific behavioral experiment is a tedious process that takes a lot of time programming and even more time evaluating all possible response scenarios and eliminating all possible programming errors. Moreover, everything that different experiments have in common has to be programmed and tested again for each different experiment. Consequently, in most cases only researchers with advanced programming skills can set up experiments, whereas there is a strong need for psychophysical testing done by psychoacousticians, audiologists, clinicians, speech scientists, etc., who may have less-advanced programming skills.

A versatile research platform has been developed at ExpORL (Laneau et al., 2005; Geurts and Wouters., 2000) to perform auditory psychophysical and speech perception experiments, either with acoustical stimulation or electrical stimulation via a cochlear implant. Over the years it has evolved from a limited program that could only perform certain specific experiments with electrical stimulation using a cochlear implant of the Laura type (Geurts and Wouters., 2000) to a version that included acoustical stimulation and more extensive procedures (Laneau et al., 2005), to finally a versatile experimental platform (APEX 3) that allows most auditory behavioral experiments to be performed without any programming, for acoustic stimulation, direct electric stimulation via a cochlear implant or any combination. In this paper, the novelty of APEX 3 will be discussed. While there are many software packages on the market for visual psychophysics, to our knowledge there are no publicly available packages that are specifically suited for auditory behavioral experiments and that allow many different auditory experiments to be performed.

The idea behind APEX 3 is that one should be able to set up an experiment quickly without any programming knowledge. APEX 3 is a generic platform with abstract interfaces to the computer monitor, input devices such as keyboard and mouse, and output devices such as sound cards or interfaces to cochlear implants, such that the user can use any of the interfaces without programming any device-specific details. While APEX 3 was mainly developed for research purposes, it is used for rehabilitation and diagnostic purposes too.

APEX 3 is a complete redesign of the previous version of APEX. It builds on the knowledge and stability we gathered during many years of experience with the previous versions of our platform (Laneau et al., 2005; Geurts and Wouters., 2000). The previous versions of APEX have been used in many studies worldwide, as shown by the citations of both APEX papers (Laneau et al., 2005; Geurts and Wouters., 2000). APEX 3 incorporates all features of version 2

* Corresponding author. Tel: +32 16 33 04 76; fax: +32 16 33 04 86.
E-mail addresses: Tom.Francart@med.kuleuven.be (T. Francart), Astrid.Vanwieringen@med.kuleuven.be (A. van Wieringen), Jan.Wouters@med.kuleuven.be (J. Wouters).

0165-0270/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.jneumeth.2008.04.020

(Laneau et al., 2005) and many more. It has already been used at ExpORL for several years and by different international partners. New in APEX 3 is that experiments are now defined in the well-known eXtensible Markup Language (XML) format,[1] allowing for a structured experiment definition in a generic format. A Matlab toolbox (the APEX Matlab Toolbox (AMT)) is distributed together with APEX 3 to ease the automatic generation of experiment files and analysis of results files. Note that a valid Matlab license is required to use the AMT.

To register, the form at http://www.kuleuven.be/exporl/apex should be filled out. After registration APEX 3 can be downloaded and used free of charge. The hardware requirements are limited to a personal computer running the Linux or Windows operating system and the necessary output devices. The main features of APEX 3 are given in the following list. Features already available in the previous versions of APEX are marked with asterisk (*).

- No programming is required to set up an experiment*.
- Multiple platforms are supported, including MS Windows and Linux.
- Multiple output devices are supported, including sound cards, an interface to cochlear implants from Cochlear Corporation and an interface to cochlear implants from Advanced Bionics Corporation. The supported devices can be used in any combination, allowing, for example, for synchronized simultaneous stimulation via a cochlear implant in both ears (bilateral electrical stimulation) or simultaneous stimulation via a cochlear implant in one ear and acoustical stimulation in the other (bimodal stimulation).
- Several psychophysical procedures are readily available and custom procedures can easily be added (plug-in procedure).
- A results file is saved after each experiment. It includes the score, the subject's responses, response times, calibrated parameter values and much more.
- Visual feedback can be given after each trial*.
- There is a special animated interface for testing (young) children*.
- There is a Matlab toolbox for experiment file creation and advanced result file analysis.
- Custom signal processing filters can be added (plug-in filter).
- Custom interfaces to external controllers can be added (plug-in controller).
- There is a graphical user interface (GUI) for calibration of parameters.

Included with the APEX 3 software package are the following:

- The APEX 3 binaries (the program itself).
- The APEX 3 schema, containing the constraints on the structure of an experiment and documentation for each element.
- The AMT, for generating experiment files and analyzing result files.
- The APEX 3 user manual.
- The APEX 3 reference manual, containing an exhaustive description of all possible elements in an experiment file.
- Example experiment files.
- Example plug-in procedures, plug-in filters and plug-in controllers.

In the following sections we will first describe the general concepts on which APEX 3 is based. In the next section (design), we show how these concepts are translated to APEX 3 implementation blocks (modules). Then the plug-in mechanism is detailed and subsequently it is shown how an experiment can be defined using an XML file. Then the general workflow when deploying APEX 3 is shown and finally some examples are given of APEX 3 in use. We will clearly distinguish between the concepts and terminology (Section 2), and the actual software implementation (the modules, Section 3).

[1] The complete XML specification can be found at http://www.w3.org/TR/xml11/.

2. Concepts

The design of APEX 3 is based on a few basic concepts that are common to all psychophysical experiments. In what follows, we define the following concepts: device, controller, datablock, stimulus, filter, screen, response, trial, procedure, experiment, result, ID and parameter. In Section 3 we will show how every concept relates to an APEX 3 module.

Device is a system connected to the computer that can be controlled by APEX 3. Devices can send signals to a transducer. Examples are sound cards and interfaces to cochlear implants. Devices can have settings (parameters) that can be controlled by APEX 3.

Controller is a system connected to the computer that does not accept signals but has parameters that can be controlled by APEX 3. An example is a programmable attenuator that controls the gain of an amplifier.

Datablock is an abstraction of a basic block of data that can be processed by APEX 3 and can be sent to the appropriate device. In the case of a sound card, the datablock would be an audio signal in the form of a series of samples that is commonly stored on disk as a so-called wave file.

Stimulus is a unit of stimulation that can be presented to the subject, to which the subject has to respond. In the simplest case it consists of a single datablock that can be sent to a single device, more generally it can consist of any combination of datablocks that can be sent to any number of devices, simultaneously or sequentially.

Filter is a data processor that runs inside APEX 3 and that accepts a block of data, e.g., a certain number of samples from an audio file, and returns a processed version of the block of data. An example is an amplifier that multiplies each sample of the given block of data by a certain value.

Screen is a GUI that is used by the subject to respond to the stimulus that was presented.

Response is the response of the test subject. It could for example be the button that was clicked or the text that was entered via the screen.

Trial is a combination of a screen that is shown to the subject, a stimulus that is presented via devices and a response of the subject. Note that while a trial contains a stimulus, it is not the same as a stimulus.

Procedure controls the flow of an experiment. The procedure determines the next screen to be shown and the next stimulus to be presented. Generally a procedure will make use of a list of predefined trials. The general working of a procedure is shown in Fig. 1.

Experiment consists of a combination of procedures and the definition of all modules that are necessary to conduct those procedures.

Result is associated with an experiment and contains information on every trial that occurred.

ID is a name given to a module defined in an experiment. It is unique for an experiment. If, for example, a device is defined, it is given an ID by which it can be referred to elsewhere in the experiment.

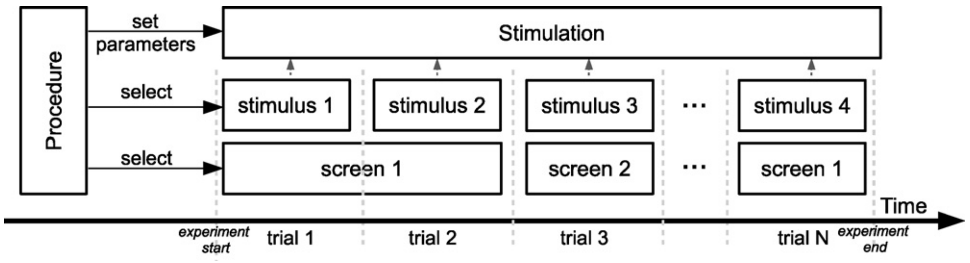

Fig. 1. Overview of the general working of procedure. Procedure presents a trial by selecting a stimulus to be sent to the stimulus output logic and a screen to be shown.

Parameter is a property of a module (e.g. a device or a filter) that is given an ID. A filter that amplifies a signal could, for example, have a parameter with ID gain that is the gain of the amplifier in dB. The value of a parameter can be either a number or text.

Parameter is one of the most important concepts of APEX 3. There are two types of parameters: fixed parameters and variable parameters. A fixed parameter is a property of a stimulus. It cannot be changed by APEX 3 at runtime and is defined when the experiment file is created. It can be used by the procedure to select a stimulus from a list, it can be shown on the screen or it can be used as a piece of information when analyzing results.

A variable parameter is a property of a module and its value can be changed at runtime. In general, a module can both have variable parameters and set variable parameters of other modules. Examples of modules that can have variable parameters (to be set by another module) are Filter, Controller and Device. Examples of modules that can set variable parameters are AdaptiveProcedure, Device, Calibrator and Screen (more information in Section 3). If a stimulus description contains a variable parameter, the parameter will be set by Device just before the stimulus is presented.

3. Design

Internally, APEX 3 consists of several modules that correspond to the concepts defined in Section 2. APEX 3 is written entirely in the C++ language[2] and makes extensive use of the Qt library[3]. C++ is an object oriented programming language, and as is usually done in such languages, every module has a base class from which several children (implementations) are derived. For example there is a generic Device module from which the WavDevice module and the L34Device (cochlear implant) module are derived for output via a sound card and output via the Cochlear Corporation Nucleus Implant Communicator (NIC) interface, respectively.

In the following sections, a number of modules are described briefly and some of the current implementations are listed. Fig. 2 gives a graphical overview of some APEX 3 modules. This list of modules is not exhaustive, but is provided to illustrate general principles. Also, since APEX 3 is designed to be easily extended by the developers and third parties (by the use of plug-ins), an ever increasing number of modules may be available in the future. The standard set of modules is described fully and exhaustively in the documentation that accompanies the software.

[2] The C++ standard is defined in ISO/IEC 14882:1998 and can be found on http://www.open-std.org/jtc1/sc22/wg21/.

[3] Qt is a programming library created by TrollTech and available from http://trolltech.com/products/qt/.

3.1. ApexControl

ApexControl is automatically loaded when APEX 3 is started. It takes care of loading all other modules and controlling the general flow of an experiment. ApexControl performs several actions (1) at the start of an experiment, (2) during an experiment and (3) at the end of an experiment. For example it will (1) prompt the user for an experiment to be loaded, (2) ask Procedure to present the next trial and (3) ask ResultSink to save the results.

3.2. Procedure

Procedure determines which stimulus is to be played next and which screen is to be shown. The general working of procedure is illustrated in Fig. 1.

Fig. 2 shows more details of Procedure. A procedure definition consists of a configuration part and a list of trials. Each trial contains references to a stimulus, a screen and an answer.

Currently, the following implementations of Procedure are present in APEX 3: ConstantProcedure, AdaptiveProcedure, TrainingProcedure, PluginProcedure and MultiProcedure.

To select the next trial, ConstantProcedure selects a trial from the trial list. It can choose a random trial from the trial list every time or present the trials in the order in which they were defined in the trial list. It completes the experiment after every trial has been presented a certain number of times. Technically, ConstantProcedure is the simplest procedure implemented in APEX 3. Typically a percent correct score is calculated from the results, or a psychometric function is fitted to the results.

AdaptiveProcedure is the implementation of an adaptive procedure. It works in the same way as ConstantProcedure, but instead of selecting a random trial it can select a trial or a stimulus based on a parameter that is changed according to the subject's last response. If the response is correct, the task is made more difficult and if the response is incorrect, the task is made easier according to a certain strategy. AdaptiveProcedure can adapt either a variable parameter or a fixed parameter. In the case of a variable parameter, the parameter will be set just before the stimulus is presented (in Fig. 1 this is indicated by the "set parameters" arrow). In the case of a fixed parameter, the stimulus with the fixed parameter closest to the desired value is selected from the user defined list of stimuli. Generally, in psychophysics other types of response strategies using the adaptive procedure exist (Leek., 2001). They can be implemented in APEX 3 using PluginProcedure (see below).

TrainingProcedure does the opposite of ConstantProcedure: it selects the next trial by comparing the subject's last answer to the possible answers defined in the different trials and selecting the one that corresponds. It can, for example, be used to make a training experiment to allow the subject to listen to the stimulus corresponding to each button.

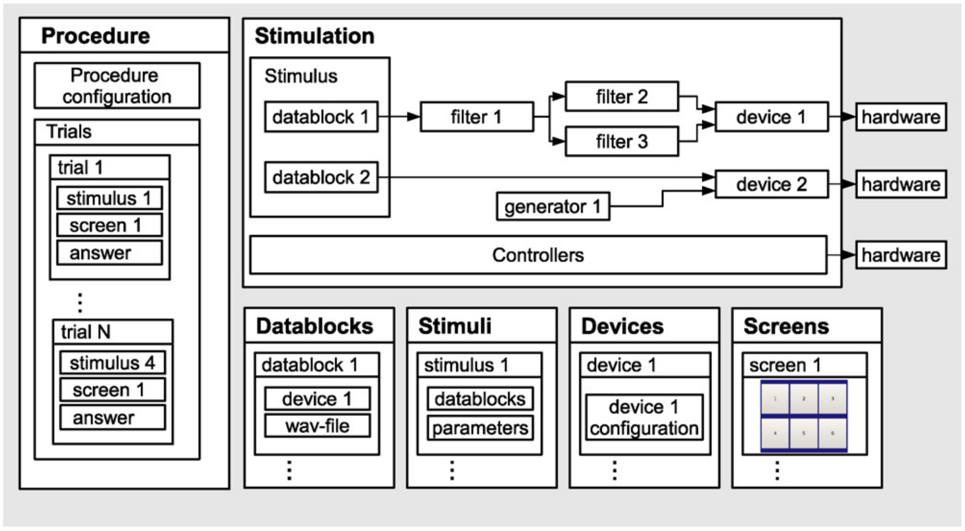

Fig. 2. Overview of several APEX 3 modules. The stimulation box is not an APEX 3 module, but groups all stimulation-related modules. The four bottom right boxes do not show a complete description of datablocks, stimuli, devices and screens, but serve to guide the eye and indicate that the corresponding modules are defined.

PluginProcedure allows a custom procedure to be defined using ECMAScript. More details are given in Section 4.

MultiProcedure is not a procedure itself, but it is a wrapper for multiple member procedures of the 4 types above. It allows procedures to be interleaved, either by selecting a random procedure for the next trial or by selecting all member procedures sequentially.

3.3. *Device*

Device can perform the following actions: load a stimulus, set a parameter and start the output. It generally loads data from disk and sends it to a transducer. It can have several parameters that control certain aspects of the device. For example, a sound card can have an output gain parameter.

In Fig. 2 the devices are shown at the right hand side of the *stimulation* box. It is clear that they accept data originating from datablocks or filters and send data to external hardware.

Currently, the following devices are implemented in APEX 3: WavDevice, L34Device and ClarionDevice.

WavDevice is an interface to sound cards, for acoustical stimulation. Any sound card supported by the operating system can be used. The following sound drivers are supported: Portaudio v19[4], ASIO[5] (Windows only), and Jack[6] (Linux only). The ASIO and Jack drivers allow APEX 3 to be used together with real-time signal processing software on the same sound card.

L34Device is an interface to the Nucleus implant communicator (NIC) interface version 2, provided by Cochlear Corporation, for direct electrical stimulation to a cochlear implant. Via the NIC interface, an L34 or a Freedom processor can be controlled to stream arbitrary pulse sequences to the cochlear implant.

ClarionDevice is an interface to the Bionic Ear Data Collection System (BEDCS) software version 1.16 and higher, provided by Advanced Bionics Corporation. It allows the presentation of arbitrary pulse sequences to the CII or HiRes90K cochlear implants.

3.4. *Controller*

Controllers are used to control devices or software outside APEX 3. They can be considered the same as Devices, with the restriction that they do not load data. Therefore the main properties of controllers are parameters. In Fig. 2, the controllers can be found at the bottom of the stimulation box.

Currently, APEX 3 contains the following controllers: PA5, an interface to the TDT PA5 programmable attenuator,[7] Mixer, an interface to the sound card mixer provided by the operating system, and PluginController, which allows custom controllers to be implemented by third parties. More information on plug-ins is given in Section 4.

3.5. *Screen*

The Screen module allows the user to define an arbitrary GUI for subject responses by combining a number of predefined building blocks. The building blocks can be divided into two groups. Elements are the actual controls shown on the screen and Layouts specify the way the elements are arranged on the screen.

The main layout types are GridLayout and ArcLayout. GridLayout arranges elements in a regular grid and ArcLayout arranges elements in a (semi-)circle. ArcLayout can be used for localization experiments, as illustrated in Section 7.5.

The main Elements are those commonly found in GUIs: Button, Label, Textbox, Spinbox and Picture. A special element is Flash, it allows a FLASH[8] movie or animation to be shown instead of a static image. In this way a test can be adapted to the interest of young children and reinforcement can be given after each trial (Laneau et al., 2005). ParameterLabel and ParameterList can be used to show the current value of a parameter on the screen.

[4] Portaudio is a free, cross platform, open-source, audio I/O library. http://www.portaudio.com/.

[5] ASIO (Audio Stream Input/Output) is an audio transfer protocol developed by Steinberg Media Technologies GmbH.

[6] JACK is a low-latency audio server, written for POSIX conform operating systems. http://jackaudio.org/.

[7] http://www.tdt.com/products/PA5.htm.

[8] http://www.macromedia.com/software/flash/about/Macromedia is currently a division of Adobe Systems Inc.

If required, the appearance of all screen elements can be completely customized by the use of style sheets.[9] A style sheet can be specified for the whole of APEX 3, for a certain Screen or per element. Examples of properties that can be changed by the use of style sheets are the color, font or position of an element.

3.6. *ResultSink*

After each trial, ResultSink queries all other modules for information to be stored in a results file. When Procedure has finished, it prompts the subject for a file name and saves the results accordingly. Results are stored in the XML format. While it is very well possible to read and interpret the XML results file, in many cases only a small part of the data presented in this file is required to interpret the results. For example, when evaluating the results of an adaptive procedure, one is primarily interested in the staircase and not always in the subject response times. To filter out unwanted information, ResultSink performs an XSL transform[10] on the results to extract the information that is required by the experimenter. The results after XSL transformation can be saved to the results file and can also be shown on screen. Even when performing an XSL transformation, the original XML results file is kept and can be consulted if further information is required.

3.7. *Calibrator*

Calibrator provides a GUI for calibrating parameters and saving and applying calibration results. Commonly a parameter such as output gain is calibrated to achieve the desired stimulation level. Any stimulus defined in the experiment file can be used as a calibration stimulus.

3.8. *Filters*

Filters are used to process data before sending it to a Device. In Fig. 2 filters can be found in the stimulation box, in between datablocks and devices. Examples of filters that are currently implemented are Amplifier, for amplifying or attenuating sound data, and PluginFilter, an interface for implementing custom filters. More information on plug-in filters can be found in Section 4.3.

A special kind of filter is a generator, a filter without input channels. Examples of generators that are currently implemented are SineGenerator, NoiseGenerator and DataLoopGenerator. The first two generate respectively sine waves and white noise. DataLoopGenerator loops a given datablock infinitely.

For each filter or generator it can be specified whether it should keep on running in between trials (while the user is responding) or not.

3.9. *Connections*

If many Datablocks, Filters and Devices are defined, it may not be straightforward for APEX 3 how to connect them. Therefore connections can be defined. Any Datablock can be connected to any Filter or Device and any Filter can be connected to any other Filter or Device. In Fig. 2 the arrows between datablocks, filters, generator and devices signify connections. By defining connections, a connection graph is created, which can also be shown graphically by APEX 3 for verification purposes. Fig. 3 shows the connections for the example experiment of Section 5.1.

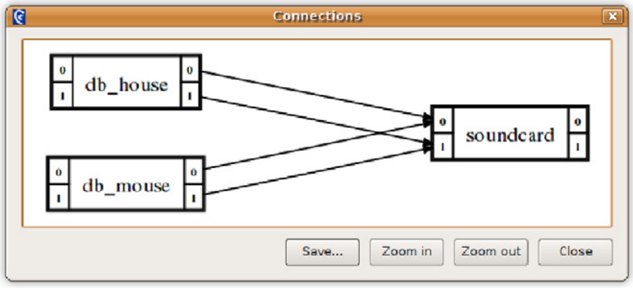

Fig. 3. Connection graph of the simple example, as generated by APEX 3. In this case each datablock has two channels (left and right) that are connected to the two channels of the sound card. The left and right channels are indicated by the numbers 0 and 1, respectively.

4. Extending APEX 3

While APEX 3 can be used for multiple purposes, it is specifically aimed at auditory research. As research inherently requires "special" and "new" features, it is possible for anyone to extend APEX 3 for their own purposes. Currently APEX 3 can be extended in three different ways, using PluginProcedure, PluginController and PluginFilter.

4.1. *PluginProcedure*

When a plug-in procedure is specified in the experiment file, the user must refer to a script file on disk. In the script file, the user must implement a few functions such as NextTrial, which determines the next screen to be shown and the next stimulus to be played.

The script file is to be written in the ECMAScript language, as defined in the ISO/IEC 16262 standard.[11] ECMAScript was based on the relatively simple JavaScript language that is used for programming dynamic web pages. Several examples of plug-in procedures are bundled with APEX 3.

While writing such scripts requires some programming, a user need only program the relevant parts of a very specific experiment and not bother with routines that are common to all behavioral experiments, such as output devices, the GUI and saving of results. Programming a simple procedure in ECMAScript typically requires only a few tens of lines of programming code.

4.2. *PluginController*

PluginController allows a user to let APEX 3 control an external device or other software program. As most device manufacturers provide an interface to their devices in the C or C++ language, PluginControllers have to be written in C++. For this purpose the Qt Plug-in mechanism is used and several examples of controllers are provided. Writing a PluginController does not require the user to be familiar with the entire C++ language, but only requires limited knowledge to understand the PluginController examples that are provided and eventual examples from the device manufacturer.

[9] The specification of cascading style sheets (CSS) and more information can be found at http://www.w3.org/Style/CSS/.

[10] XSL transforms are standardized by the W3C consortium and the specification is available at http://www.w3.org/TR/xslt.

[11] http://www.ecma-international.org/publications/standards/Ecma-262.htm.

4.3. *PluginFilter*

As the name suggests, a PluginFilter acts like the built-in APEX 3 filters. Just like PluginControllers, PluginFilters have to be written in the C++ language. A PluginFilter is essentially a callback function that is called every time a block of data has to be processed. If implementing a custom algorithm in C or C++ is too bothersome or difficult, a user can alternatively use a different language, such as Matlab or another script language. This option requires that (1) the script language can be called from C or C++, and (2) it is possible to convert between C/C++ data types and the script language's data types.

5. Defining an experiment

Previous versions of APEX used a custom text format to define experiments. The format was as simple as possible to enable the creation of experiment files without much technical background knowledge. While APEX 3 of course still has the same aim, it is clear that given the large number of possible experiment configurations, a simple text format does not suffice. Therefore, the XML format was chosen for defining experiments. To ease the transition, APEX 3 can convert an APEX 2 experiment file to a file in the new XML format.

Advantages of the XML format are that it is human readable, i.e., it can be viewed and interpreted using any text editor, and that it can easily be parsed by existing and freely available parsers.[12] Moreover, many tools exist for editing, transforming or otherwise processing XML files.

Next to adhering to the general XML format, APEX 3 experiment files have a fixed structure that is enforced by an XML Schema[13] file. This file specifies where elements should occur and in addition contains documentation on every element in English. A good XML editor, such as OxygenXML[14] and many others, can use the APEX 3 schema file to check whether an experiment file is valid, to suggest, while typing, what element is to be defined next in the file and to show appropriate documentation per element of the experiment file that is being edited.

In what follows we will describe a very simple APEX 3 experiment file step by step. Note that the order of our descriptions does not correspond to the order of the elements in the experiment file. We will only describe the elements that are necessary to understand the general structure of the file. For more details we refer to the APEX 3 user manual and reference manual, both distributed together with APEX 3. The example is an experiment that will show two buttons on the screen with text "house" and "mouse". When started, it will play either a wave file sounding like "house" or a wave file sounding like "mouse". The subject has to click on the button corresponding to the perceived sound. In speech science, this is called a minimal pair.

An XML file consists of a series of *elements*. Every element can have content. There are two types of content: *simple* content, for example a string or a number, and *complex* content: other elements. An element can also have attributes: extra properties of the element that can be set. Elements are started by their name surrounded by < and > and ended by their name surrounded by </ and >. In the following example, element <a> is started on line 1 and ended on line 7. Element <a> contains *complex* content: the elements <b> and <c>. Element <b> contains *simple* content: the numerical value 1. Element <c> again contains *complex* content: the elements <c1> and <c2>. Element <c1> has an attribute named attrib1 with value 15. Element <c2> on line 5 shows the special syntax for specifying an empty element. This is equivalent to < c2 ></c2 >.

```
<a>                                  1
  <b>1</b>                           2
  <c>                                3
    <c1 attrib1="15"> </c1>          4
    <c2/>                            5
  </c>                               6
</a>                                 7
```

As APEX 3 experiment files are in the XML format, the general syntax is the same as in the previous example, but of course the structure is more complex and there are restrictions as to which element can occur where (as enforced by the APEX 3 schema).

5.1. *A simple example experiment*

In what follows we will describe each of the main elements in the experiment XML file separately. Together they define the entire experiment. First we define a device to interface with our sound card.

```
<devices>
  <device id="soundcard"
    xsi:type="apex:wavDeviceType">
    <driver>portaudio</driver>
    <card>default</card>
    <channels>2</channels>
    <gain>0</gain>
    <samplerate>44100</samplerate>
  </device>
</devices>
```

All devices defined in the experiment file are grouped in the element `<devices>`. As there is only one device in this file, there is only one `<device>` element. The attribute id is set to `soundcard`. As an ID is unique for an entire experiment file, we can use it later on to refer to this device. The `xsi:type="apex:wavDeviceType"` attribute tells APEX 3 that we are dealing with a sound card. The `device` element contains several other elements that set various parameters of the sound card. The number of output channels to be used is 2, the output gain is 0 dB and the sample rate is 44 100 Hz. Information on all available parameters can be found in the APEX 3 reference manual.

Next we define two datablocks as follows:

```
<datablocks>
  <uri_prefix>../stimuli</uri_prefix>
  <datablock id="db_house" >
    <device>soundcard</device>
    <uri>house.wav</uri>
  </datablock>

  <datablock id="db_mouse" >
    <device>soundcard</device>
    <uri>mouse.wav</uri>
  </datablock>
</datablocks>
```

[12] APEX 3 uses the Xerces-c parser for parsing XML files. http://xerces.apache.org/xerces-c/.
[13] The XML Schema specifications are available at http://www.w3.org/XML/Schema/.
[14] OxygenXML (http://oxygenxml.com/) has all necessary features for working with APEX 3 experiment files. It is a commercial program, but a free license can be obtained by non-profit organisations that work in the domains of ecology, human aid and renewable energy sources.

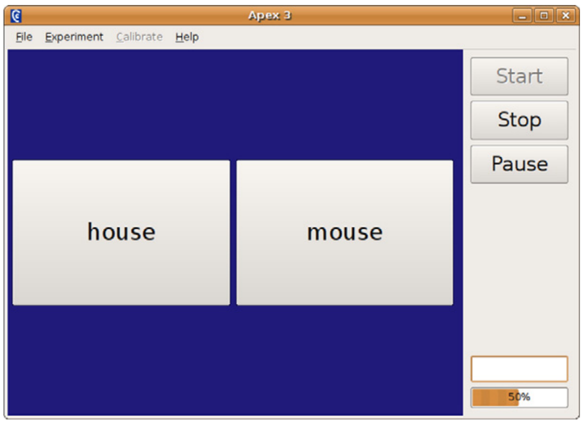

Fig. 4. Screen of the example experiment.

All datablock definitions are grouped in the element <datablocks>. In this case two datablocks are defined. They each get an ID that is unique for the experiment file and that allows us to refer to them later on. For each datablock, <device> refers to the ID of the device that will play the datablock and <uri>[15] contains the name of the file from which to read the data. The number of channels in the file is automatically determined by APEX 3. Here we refer to the ID soundcard that was defined in the <devices> element.

We now have one device with ID soundcard and two datablocks with ID db_house and db_mouse. As no specific connections are defined for this experiment, APEX 3 automatically connects all datablocks to the device. Fig. 3 shows the connection graph in this case, as generated by APEX 3.

Next we define two stimuli.

```
<stimuli>
  <fixed_parameters/>
  <stimulus id="stim_house">
    <datablocks>
      <datablock id="db_house"/>
    </datablocks>
    <variableParameters/>
    <fixedParameters/>
  </stimulus>

  <stimulus id="stim_mouse">
    <datablocks>
      <datablock id="db_mouse"/>
    </datablocks>
    <variableParameters/>
    <fixedParameters/>
  </stimulus>
</stimuli>
```

In this very simple example, each stimulus again gets an ID and refers to one datablock. We now have one device, two datablocks and two stimuli. All stimulation-related specifications are now defined. We proceed by defining a screen.

```
<screens>
  <screen id="screen1">
    <gridLayout height="1" width="2">
      <button row="1" col="1" id="btn_house">
        <text>house</text>
      </button>

      <button row="1" col="2" id="btn_mouse">
        <text>mouse</text>
      </button>
    </gridLayout>

    <buttongroup id="buttongroup">
      <button id="btn_house"/>
      <button id="btn_mouse"/>
    </buttongroup>
    <default_answer_element>
      buttongroup
    </default_answer_element>
  </screen>
</screens>
```

The <screens> element can contain several <screen> elements. In this case there is only one screen and it contains a GridLayout with a single row and two columns. In the GridLayout, there are two buttons with ID btn_house and btn_mouse. On each button a piece of text is shown, in this case “house” and “mouse”.

The remaining element in <screen> groups the buttons into a ButtonGroup. The resulting screen is shown in Fig. 4. For more information on ButtonGroup we refer to the APEX 3 reference manual.

Finally we define the Procedure that will control the flow of the experiment.

```
<procedure
   xsi:type="apex:constantProcedureType">
  <parameters>
    <presentations>2</presentations>
    <order>sequential</order>
  </parameters>

 <trials>
    <trial id="trial1">
      <answer>btn_house</answer>
      <screen id="screen1"/>
      <stimulus id="stim_house"/>
    </trial>

   <trial id="trial2">
      <answer>btn_mouse</answer>
      <screen id="screen1"/>
      <stimulus id="stim_mouse"/>
    </trial>
  </trials>
</procedure>
```

The <procedure> element contains two other elements: <parameters> and <trials> and the attribute xsi:type="apex:constantProcedureType" indicates that we

[15] Uniform resource identifiers (URI) are defined in RFC 3986. In its simplest form, an URI can be a file name.

use a ConstantProcedure. In <parameters> the behavior of the procedure is defined. In this example we specify that each trial has to be presented twice and that the trials are to be presented in the order as specified in the <trials> element (sequentially).

The <trials> element contains several individual <trial> elements that specify a trial. After selecting the next trial to be presented, the Procedure will show the specified screen and send the specified stimulus to the correct devices. After the subject's response, it will check whether the response corresponds to the given answer and decide on the next trial to be presented. For example if the subject clicked on the button with text "house", the procedure will compare the ID of this button (btn_house) with the content of <answer>.

This simple example illustrates that no programming at all is required to define an experiment and that the syntax is straightforward and easy to learn, especially when using the examples that are provided with APEX 3.

5.2. Writing experiment files

For complicated experiments with many stimuli, an experiment file can become rather long and tedious to write manually. There are several solutions to this problem. APEX 3 comes with many examples and most probably one will find an example that can be adjusted to the specific requirements of the experiment. Also, several XML editors can parse the APEX 3 schema file and suggest the element to be defined next and give documentation on the current element in the experiment file.

A more efficient solution is to use the APEX Matlab Toolbox (AMT). This toolbox is a collection of Matlab files that generate parts of APEX 3 experiment files. One can use the different functions in the AMT to generate an entire experiment file or one can create a template and fill in the missing parts using the Matlab toolbox. Take, for example, the simple experiment from Section 5.1. If we would like to adapt this experiment to present 50 different words instead of only 2, we could take the original experiment with two different words and replace the <trials>, <datablocks> and <stimuli> parts by special markers, e.g., \$\$trials\$\$, \$\$datablocks\$\$ and \$\$stimuli\$\$. The AMT contains a function that recognizes these markers and replaces them by given pieces of text. An experiment file with such markers is called a *template*.

The AMT also contains functions like a3trial, a3datablock and a3stimulus that generate the corresponding elements in XML format. We could therefore create a loop in Matlab that is executed 50 times and generates the correct trial, datablock and stimulus elements and afterwards have the AMT replace the markers in our template. A typical Matlab function for generating an experiment file using the latter mechanism requires a few tens of lines of code, in contrast to the thousands of lines of code that would be required to write and debug the same experiment entirely in Matlab.

6. Workflow

In this section, we show the typical workflow of setting up, conducting and analyzing an experiment using APEX 3. The workflow is illustrated in Fig. 5.

Experiment design determines the goals and methods of the experiment.

Experiment file creation determines how the methods can be implemented as an APEX 3 experiment by describing them in terms of the basic APEX 3 concepts. If necessary one of many examples can be consulted.

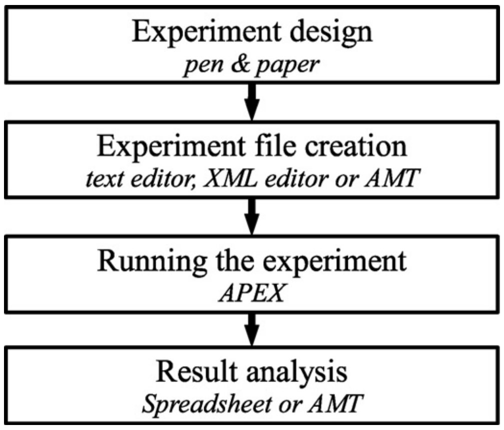

Fig. 5. Workflow conducting an experiment using APEX 3. AMT is the APEX 3 Matlab Toolbox.

Running the experiment APEX 3 can be used for unattended experiments, where the subjects can respond using a computer mouse, keyboard or touch screen, but also for attended experiments where the experimenter controls the computer. In the latter case, APEX 3 can be configured to show some properties of the current stimulus on screen.

Results analysis For each run of the experiment, a results file is available in XML and, if requested, an XSL transformed version. It is possible to either analyze the results manually by pasting them into a spreadsheet or statistical analysis software, or automatically by using the APEX Matlab Toolbox (AMT) to read the results files and perform advanced analyses.

7. Examples

In this section we give a few examples where APEX 3 can be used. This list is nowhere near exhaustive, as APEX 3 is designed to be able to perform any psychophysical experiment.

7.1. Gap detection using a three-alternative forced choice paradigm with a cochlear implant

In our gap detection experiment the method of constant stimuli was used. The subject will, in every trial, hear three different sounds (three so-called intervals). One of the sounds has a small gap in it. The subject has to respond whether the sound with the gap was in the first, second or third interval.

As we want to present the sounds directly to the cochlear implant of our subject, we use the L34Device as a Device to control a cochlear implant from Cochlear Corporation. We need two data files on disk: one containing the sound without gap (NoGap) and one containing the sound with gap (Gap). While our datablocks refer to wave files in the case of a sound card, they now refer to so-called qic files, that can be streamed directly to the cochlear implant and can be created by the Nucleus Matlab Toolbox provided by Cochlear Corporation.

To create the experiment file, we can start from the example in Section 5.1. First we replace the datablocks by two datablocks that refer to our Gap and NoGap file. Then we replace the stimuli by two stimuli that refer to our Gap and NoGap datablocks and we replace the device by an L34Device. We also change the screen to show three buttons instead of two. Finally we change the procedure to reflect our experimental design. This is done as follows:

```
<procedure
  xsi:type="apex:constantProcedureType">
  <parameters>
    <presentations>10</presentations>
    <skip>0</skip>
    <order>sequential</order>
    <choices>3</choices>
  </parameters>

  <trials>
    <trial id="trial1" >
      <screen id="screen1" />
      <stimulus id="stimulusGap" />
      <standard id="stimulusNoGap"/>
    </trial>
  </trials>
</procedure>
```

For experiments where several stimuli are presented during a single trial and the subject is expected to recognize the stimulus that is different in a certain way, multiple stimuli have to be defined per trial. The stimulus that is different is defined using <stimulus> and the other stimuli using <standard>.

<choices> contains the number of stimuli presented to the subject per trial. In this example the number of choices is three, which means that the stimulus defined using <stimulus> will be presented once and the stimulus defined using <standard> will be presented twice.

Note that while we used the L34Device (not shown in the XML listing) to control the cochlear implant directly, the experiment setup is nearly identical for acoustic stimulation.

7.2. *Adaptive determination of the speech reception threshold*

APEX 3 can be used to determine a subject's speech reception threshold (SRT) for a certain speech material in noise. The SRT is defined as the signal-to-noise ratio (SNR) at which the subject's performance is at 50% correct. We will use an adaptive procedure to determine the SRT. In this example the first speech token (sentence or word) is presented at a low SNR and is repeated at increasingly higher SNRs until the answer is correct. Thereafter the SNR is decreased using a certain step size when the response is correct and increased when the answer is incorrect. Our setup is attended, meaning that the subject has to answer orally and that the experimenter controls the computer running APEX 3. Any speech material can be used. As an example we will use the LIST sentences with the accompanying speech-weighted noise (van Wieringen and Wouters, 2008) which consists of 35 lists of 10 sentences.

Again we start from the example in Section 5.1. We create a datablock for each of the 10 sentences with ID db-sentenceN with N the number of the sentence and one extra datablock for the file with speech weighted noise with ID noisedata.

We want the noise file to be repeated continuously. Therefore we create a dataloop generator as follows:

```
<filter xsi:type="apex:dataloop"                 1
      id="noisegen">                             2
  <device>soundcard</device>                     3
  <channels>1</channels>                         4
  <continuous>true</continuous>                  5
  <datablock>noisedata</datablock>               6
  <basegain>0</basegain>                         7
  <gain id="noisegain">0</gain>                  8
</filter>                                        9
```

The generator has ID noisegen, it will use datablock with ID noisedata and it will play during the entire experiment, even while the user is responding (line 5). To vary the SNR, in this example we will vary the amplitude of the noise. We will therefore vary gain of our dataloop generator. On line 8 the gain element has an extra id attribute, which results in the gain of our generator being declared as a parameter that can be modified during the experiment by other APEX 3 modules. In order to change the gain of the dataloop generator, an adaptive procedure is defined. Note that in this case, the level of the noise varies with the SNR and the level of the speech is held constant. The opposite can be achieved by using an amplifier to adapt the level of the speech.

```
<procedure                                         1
   xsi:type="apex:adaptiveProcedureType">          2
<parameters>                                       3
  <presentations>1</presentations>                 4
    <skip>0</skip>                                 5
    <order>sequential</order>                      6
    <nUp>1</nUp>                                   7
    <nDown>1</nDown>                               8
    <adapt_parameter>                              9
      noisegain                                    10
    </adapt_parameter>                             11
    <start_value>-12</start_value>                 12
    <larger_is_easier>                             13
      true                                         14
    </larger_is_easier>                            15
    <repeat_first_until_correct>                   16
      true                                         17
    </repeat_first_until_correct>                  18
    <stepsizes>                                    19
      <stepsize begin="0" size="2"/>               20
    </stepsizes>                                   21
</parameters>                                      22
```

```
<trials>                                           23
  <trial id="trial_sentence1">                     24
    <answer>correct</answer>                       25
    <screen id="screen"/>                          26
    <stimulus id="stimulus_sentence1"/>            27
  </trial>                                         28
  <trial id="trial_sentence2">                     29
    <answer>correct</answer>                       30
    <screen id="screen"/>                          31
    <stimulus id="stimulus_sentence2"/>            32
  </trial>                                         33
etc...                                             34
</trials>                                          35
</procedure>                                       36
```

On line 10 the parameter to be adapted is set to the gain of our dataloop generator by referring to its ID. On lines 7 and 8, the adaptive procedure is defined as a 1up/1down procedure and on line 14 larger values of the parameter are defined to be easier for the subject to respond. The elements <repeat_first_until_correct> and <stepsizes> on lines 16–21 are described in detail in the APEX 3 user manual.

7.3. *Evaluation of a signal processing algorithm with an adaptive SRT procedure*

Imagine we want to do an SRT test as shown in Section 7.2, but now not only present the stimulus to the subject but first run it through a custom noise suppression signal processing algorithm. In this case we would develop a PluginFilter for our algorithm using the C or C++ language. When a sound signal is played back, APEX 3 splits it in fixed-size blocks of samples and sends each block to the PluginFilter, which can process it. After processing, the resulting blocks are sent to the next Filter or to the output Device.

7.4. *Bimodal stimulation (acoustical and electrical)*

In this example, we will use different devices together. We will not create an entire experiment but just create a stimulus that presents an acoustical sinusoid and an electrical pulse train sequentially.

In the <devices> element we now have two devices, a WavDevice with ID soundcard and an L34Device with ID l34:

```
<devices>
  <master>soundcard</master>
  <device id="soundcard" xsi:type="apex:wavDeviceType">
    <channels>2</channels>
    <gain>0</gain>
    <samplerate>44100</samplerate>
  </device>
  <device id="l34" xsi:type="apex:L34DeviceType">
     <device_id>1</device_id>
     <implant>cic4</implant>
     <trigger>in</trigger>
     <volume>100</volume>
     <defaultmap> ... </defaultmap>
  </device>
</devices>
```

The <master> element indicates that the sound card should be started last. The defaultmap for the L34 is not shown here and for a description of the other L34 parameters we refer to the APEX 3 reference manual.

We create two datablocks: one refers to sinusoid.wav and the other to pulsetrain.qic and to their corresponding devices. Our stimulus is now defined as follows:

```
<stimulus id="stimulus_bimodal">             1
  <datablocks>                               2
    <sequential>                             3
      <datablock id="db_sinusoid"/>          4
      <datablock id="db_pulsetrain"/>        5
    </sequential>                            6
  </datablocks>                              7
</stimulus>                                  8
```

As the datablocks are inside a <sequential> element, first the acoustical sinusoid will be played and immediately thereafter the electrical pulse train will be sent to the subject's cochlear implant. This type of stimulus could for example be used for a pitch matching task or a loudness balancing task with a subject with both an acoustic hearing aid and a cochlear implant. Note that simultaneous bimodal stimulation could be achieved by replacing <sequential> by <simultaneous> on line 3.

7.5. *Localization of sounds*

In a localization experiment, typically the subject is seated in the middle of an arc of *N* speakers. A stimulus is presented from one of the speakers and the subject's task is to indicate this speaker.

Again starting from the simple example in Section 5.1, we only need to modify the <devices>, <screens> and <connections> elements.

If the sound card has a sufficient number of output channels to control all the speakers, we only have to change the <channels> element in the <device> element to value *N*. If not, multiple sound cards can be used together.

The screen has to be changed to show a semi-circle of *N* buttons instead of a grid of two buttons. Therefore <gridLayout> is changed to <arcLayout> and the necessary buttons are added. For $N = 9$, the result would look like Fig. 6.

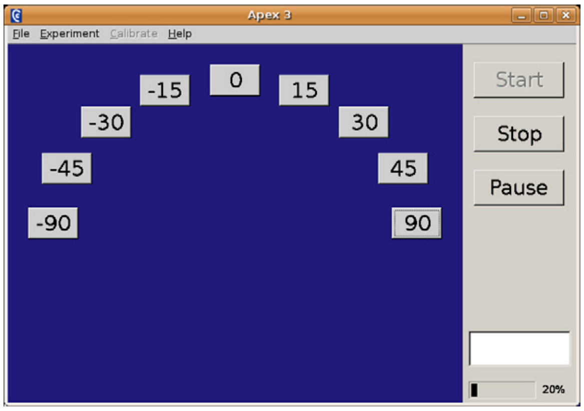

Fig. 6. Example of an arcLayout with $N = 9$ buttons.

8. Conclusions

APEX 3 is a versatile program for conducting psychoacoustic behavioral experiments. The most commonly used psychophysical procedures are implemented and APEX 3 can easily be extended with custom procedures. It can control three different output devices: (1) sound cards, (2) streaming and sending pulse sequences to cochlear implants of Cochlear Corporation and (3) sending pulse sequences to cochlear implants of Advanced Bionics Corporation. In addition, custom signal processing algorithms and controllers can be plugged into the APEX 3 framework.

To ease the generation of experiment files and the analysis of results, a Matlab toolbox is provided.

APEX 3 is freely available for anyone after registration. Documentation and many examples are distributed with the software.

Acknowledgments

The authors thank Stijn Verstraeten, Michael Hofmann and Dominique Devriese for their help programming APEX 3. We acknowledge the help of Lieselot Van Deun and Astrid van Wieringen in writing the manual and by reporting problems with previous versions. We also thank all other beta-testers for their valuable comments and bug reports.

Dieter Beaven, Colin Irwin and Clemens Zweekhorst from Cochlear Corporation have been a great help implementing the NIC interface and Olivier Macherey assisted with the implementation of the BEDCS interface.

During the development of APEX 3, the first author was partly funded by the IWT (Institute for the Promotion of Innovation by Science and Technology in Flanders) project 050445.

References

Geurts L, Wouters J. A concept for a research tool for experiments with cochlear implant users. J Acoust Soc Am 2000;108(6):2949–56.

Laneau J, Boets B, Moonen M, van Wieringen A, Wouters J. A flexible auditory research platform using acoustic or electric stimuli for adults and young children. J Neurosci Meth 2005;42:131–6.

Leek MR. Adaptive procedures in psychophysical research. Percept Psychophys 2001;63(8):1279–92.

van Wieringen A, Wouters J. LIST en LINT: sentences and numbers for quantifying speech understanding in severely impaired listeners for Flanders and The Netherlands. Int J Audol; in press.