

# 1 Introduction

Users with limited physical control or motor difficulties are unable to usefully access a computer via the default mouse and keyboard devices. The current generation of On Screen Keyboard (OSK) is still in a primitive state in comparison with other assistive technologies such as screen readers, screen magnifiers and voice input software. The possibilities for improving the efficiency of OSK usage are vast, and yet the community is drastically underserved.

This proposal for improving switch access to Firefox on Windows is clearly seen as one step towards the greater goal of providing transparent alternative input access to computers. This will be achieved by creating a cross platform (Linux, Mac, Windows) OSK with features such as direct 'in application' selection, flexible input, scriptability and attractive visuals.

We have selected Windows as that is the platform that the target users are currently using. Firefox has the dual advantages of giving standards based access to the rich world of web services and is also an persuasive example of the quality of Open Source Software 'products'.

## 2 Background

### *Types of Users*

To make a broad categorization users may be:

- Adults who experience an accident or event such as a stroke
- People with congenital physical disabilities, especially children, who often have multiple disabilities, including learning difficulties or limited education, and often need to learn to use a switch through specialist educational programs.

### *Input Methods*

The current generation of software works with [a variety of hardware devices](#), but unfortunately users are separated into two broad categories.

- Users that can point to any area of the screen (e.g. via a head mouse, eye tracking or a joystick). These users can use an OSK reasonably well perhaps with the screen enlarged or special tremor reducing hardware or drivers. They can usually activate features of an application directly in the user interface. However, for text input this is still much slower than typing.
- Switch access users (who can perform a single action such as pressing a button). These users must operate software through a so-called “scanning interface”. In this context, scanning refers to a moving highlight and then selecting the highlighted item. There are several variations in the exact method of scanning and selection including manual or automatic movement of the highlight. Scanning is extremely tedious, for example when an option is missed, the user must cycle back to the start. There are several optimisations such as arranging the order of items according to use or grouping items to allow cause scanning before finer selection within the group.

Each switch user has unique control requirements and often relies on specialist clinical technical support to provide, select and configure their equipment. In the UK equipment tends to be expensive with funding and provision via educational or clinical public services,

charities or private funding.

There is an [online demo of various scanning modes](#) (needs windows flash plugin) and the Ace Centre have a comprehensive [introduction to Switch Access](#).

A problem with the current separation of user types is that users who cannot point, but can perform more than a single action, must deal with an oversimplified user experience. Even if they are capable of using more than one switch, the OSK may not make use of it to improve input efficiency.

## ***User Tasks***

- Command and control. This is where the user selects from known options in a UI
- Text input. A common tasks, where users can enter arbitrary text.
- Basic verbal communication. Users in both above groups often rely on alternative input technologies not only for computer access, but also to enable basic verbal communication with others (called AAC, as used by Stephen Hawking).

## ***Application Support***

Switch access to applications is provided either via specific support in the application itself (Clicker), drivers that emulate other input devices (Sensory Software driver) or via a general purpose intermediate scanning overlay program (GOK, SAW or The Grid). In all cases operation is relatively tedious due to the limited gestures available to users and exacerbated by frequent difficulties with accuracy and consistency.

This grant aims to allow access to any accessible desktop application, with a strong focus on usability with Firefox.

## ***Today's State of the Art***

The current market is focused on Windows and proprietary solutions in manufactures, provision and support. Two good FOSS general switch scanning programs exist -- GOK on Linux and SAW on Windows. The community appear to be unaware of GOK and its advanced features designed by experienced developers and adaptive technology trainers.

The [GOK](#) maintainers feel it now needs to be refactored / rearchitected, have a visual refresh and should become cross platform. They have also expressed a desire to use Python. The lack of Windows and Mac support further limits its adoption. One thing that GOK has shown, however, is the possibility of presenting switch users options gleaned from the user interface, rather than a simple keyboard where difficult keystroke combinations must be simulated for each action. GOK also has predictive text entry. After extensive discussions with the authors of GOK, we are choosing not to extend GOK, because of the clear need for refactoring the codebase for flexibility and new features. However, the group believes it is achievable and desirable to reuse parts of GOK as wrapper Python libraries.

[SAW](#) has been developed for switch users by staff at the Ace Centre who have extensive clinical experience. It depends on Microsoft Visual C and MFC. It has many advanced features including graphics. Although SAW is open source, for this project we are choosing not to extend SAW due to limitations with the MFC base. First, it is not cross platform; second, MFC is judged to be too brittle and difficult to work with and third it's current design may not lend itself to easy refactoring.

[The Grid](#) is a very popular proprietary program from Sensory Software and offers many features that focus on it being a communications package.

Another interesting case is a Firefox extension called the [Hawking Toolbar](#). While the Hawking toolbar does not allow for text input (and thus is not a complete solution), it shows the value of in-application scanning. That is, the actual scanning bar does not highlight options in a separate keyboard taking up valuable screen real estate. Instead, the options in the Firefox UI itself are highlighted. This is a much more direct and natural way of selecting program features and browsing the web. The Hawking Toolbar is a Firefox Extension whereas a long term goal of this proposal is to create a solution that will work with any program. However some of the Hawking Toolbar code may be usable.

[Drake Music Project's E-Scape](#) music composition and performance program is worth a mention as a program that makes complex operations possible via alternative input devices using options built directly into the GUI.

For text input, we can learn a lot from Dasher. Dasher is a cross-platform assistive technology. Although Dasher is something best [seen demonstrated](#) rather than explained, it shows the benefit of creative input methods and word prediction. Dasher allows users to enter letters by steering using only 'left' and 'right' controls. As letters are selected, the probability of the next letter affects the size of the region which grows to meet the mouse pointer. Dasher users have been able to type 25 words per minute. On Linux Dasher also allows control of applications by selecting menu items.

[OnBoard](#), formerly SOK, is a new light weight OSK from Ubuntu without switch functionality, though there are plans to add it. It has attractive visuals created using Cairo. Although its feature set is still very primitive and the agenda seems rather different, we are currently considering the offer of the developers to collaborate.

We will also take as input the [WWAAC accessible browser](#) project , which identified many of the requirements of an accessible browser.

### 3 Solutions

The proposal is to develop a new cross-platform alternative input control centre. The software will be written with long range goals in mind, but aim toward short term deliverables. In the first round of development, the focus will be specifically on Firefox on the Windows platform. As much as possible, control of the software will occur in the user interface itself, providing transparent control of the browsing experience without obtrusive OSKs being present on screen. OSKs will used were appropriate. This is seen as the important first step in improving OSK users experience of using GUIs.

The long term goal, perhaps via several rounds of grant funding, is to develop a flexible cross platform access system, that enables each user to reach maximum potential for their various User Tasks (as stated above). The input capabilities of each user will be matched to software features via a 'capability palette' that allows gestures made using various input technologies to be used to control programs according to each users preferences. Other technologies for control and text input such as Dasher and Voice dictation will also be integrated in order to provide the most efficient and transparent experience.

It is hoped that this project will promote awareness of alternative input amongst developers and will encourage much needed innovation in the switch access arena.

#### ***Firefox as the first phase***

Firefox has the capability to be a platform of choice for switch users. It's support of and commitment to standards and accessibility provide a firm foundation on which to build solutions allowing switch users to access existing websites and on-line services. The rapid increase in Web 2.0 style features

such as web apps combined with the upcoming Accessible DHTML in Firefox will make even more functionality accessible. Firefox's extensibility via XUL offers another dimension for creating powerful services. Finally, as open source, Firefox is low cost for users and service providers. It is also attractive to third party developers who can freely use it as a strong foundation for their offerings. Switch users should be enabled to benefit from all this functionality and have much to gain from transparent access to the web.

Making Firefox as transparent as possible for switch users will meet the the needs of adults and children. It also provides a way to introduce FOSS to those locked into Windows and thus offers them choice. Even more choice will be offered by the many websites and services made accessible. It will significantly empower a group of users who are currently poorly served by IT.

Firefox is created with the Mozilla Platform which is a leading-edge, cross-platform, extensible framework for creating modern applications. The Mozilla platform is attractive fro implementing this project and supporting all the needs of OSK moving forward into the foreseeable future. This is quite separate from using Firefox as the target application in the first phase.

## ***Implementation strategies***

There are two currently proposed possibilities for implementation.

The favorite currently proposed implementation is to create a XULRunner application which builds the keyboard widget using SVG and XBL. Where possible, scripting will be based in Python. Code from compatibly-licensed OSK's may be reused as may that from the Hawking Keyboard. This combination of technologies was inspired by the [XULRunner Clock](#). It allows for a scaleable, skinnable interface with an extremely attractive look and feel. Semi-transparent overlays are among the possibilities. This would create an extremely flexible cross-platform system that would be a joy to extend, and have a great chance of attracting additional contributors. The XBL widget abstractions would make it easy to define new keyboard layouts.

The second possibility is to work with the developers of OnBoard and extend that. The obvious advantage is that there would be more developers right away. Unfortunately, it does not currently provide a declarative system for designing keyboards as XBL would.

## **4 Work Plan**

- Research GOK / onBoard / SAW and proprietary scanning switch overlay programs to determine the best of current practices. Also review techniques for in application switch selection such as the Hawking Keyboard and E-Scape . Make a decision on collaboration with the onBoard developers vs. XULRunner/XBL/Python implementation.
- Research physical and software driver switch interfaces to PCs and decide on initial level of support (probably via USB joystick simulation with switches connected as fire buttons. Keys 1,2,3 and 4).
- Work with clinical organizations and users to ensure user requirements are accurately met. OATSoft.org will be used as a public discussion forum as many practitioners meet there.
- Implement component that provides switch access to Firefox under Windows. This will provide OSK and in-app selection for important control, text input and browsing.
  - XULRunner process (with Windows Installer).
  - XBL, XUL and SVG for UI and visuals
  - Python, via pyXPCOM for behavior

- Hook USB switch / Joystick events, using pyGame binding of SDL.
- Drive Firefox via MSAA or synthetic key/mouse events or direct menu control
- Patch base technologies if required.
- Refactor GOK code and use sections converted to Python 'C' modules.
- Research innovative new methods to improve the user experience, which may not be developed until a later grant. See Future Possibilities below.
- Graphics design will be needed, possibly from Mozilla community member Ken Saunders (A.K.A mouse runner)

## 5 Success Criteria

Success will be determined by the availability of access to all critical Firefox functions via the on screen keyboard developed, either via pointer-only or switch access. The selection of program features should be available directly from within the UI. Although success will be determined on the Windows operating system, a successful completion of this grant will also result in an extensible platform that can be ported without extreme difficulty to Mac and Linux, and is reasonably easy to extend for future improvements (see Future Possibilities below). Portability will be achieved by using inherently portable technologies such as XUL and following good design practices such as abstraction layers.

## 6 Future Phases

The following ideas represent areas of future development, subject to further funding becoming available.

**Cross platform:** This has the potential to become a central accessibility feature of Linux+Gnome as well as Windows and Mac. Future versions should run on all platforms.

**Extended applications:** The keyboard 'keys' could show standard symbols sets as used for AAC devices. Speech synthesis and recorded audio 'samples' will also be needed. The [WWAAC](#) web browser project, investigated many features required for a highly accessible browser and some could be added to this project.

**Hardware communication devices (VOCAs):** Many users use dedicated hardware AAC devices to communicate with others using switch input and speech synthesis. Embedded Linux with this program will provide the basis for a low cost alternative system.

**User Capability Palette:** One way we can really improve on today's experience for users with physical disabilities is to do the most we can with all of each user's capabilities. A capability palette could allow the matching of user capabilities to program features. For example, features could include "Go back", "Skip five", "Next input modality", etc. Input capabilities might include "user utterance", "foot switch", "brain switch action", etc.

**Enhance Text Input:** The concept of mouse gestures could be extended to joysticks, head mice and eye trackers, allowing faster text input and feature selection. This needs practical investigation, but could provide a huge improvement in the speed of access. Another useful addition would be the inclusion of word prediction, thus speeding up user input a great deal, even for switch users.

**Customisation:** The on screen keyboard will be able to drive applications using a variety of means in order to provide control and content creation. It will be visually appealing and usable out of the box, yet offer creative possibilities for skinning and personalisation.

**Community Resources:** It will be possible to develop online repositories of keyboard layouts, skins and community features such as forums. This could provide a focal point for user discussion and innovation.

**Open Standard for keyboard layouts:** The definition format for keyboards could be promoted as a standard to be adopted by other programs such as The Grid. This will give users more choice.

**Reusable Components:** It may be possible to factor out components that will be of use to other projects. For example scanning modes and text prediction.