

Blindstation : a Game Platform Adapted to Visually Impaired Children

Sébastien Sablé and Dominique Archambault
INSERM U483 / INOVA - Université Pierre et Marie Curie
9, quai Saint Bernard, 75,252 Paris cedex 05, France
Sebastien.Sable@snv.jussieu.fr

Keywords: *visually impaired, multimedia, multi modality*

Abstract: The TiM project intends to develop and to adapt computer games for visually impaired children. A game platform, the blindstation, was developed to adapt existing content or create some new games. It provides a set of Python functions to describe those games in an abstract way, independent from their representation. The platform can then render the game in a multi-modal way using the screen, keyboard, mouse and joystick, but also using some specific devices like a Braille terminal, 3D sound, a tactile board or a speech synthesizer. The rendering is done according to an XML style sheet which describes the available resources. It can be customized depending on the available devices but also on the user's choices and disabilities. Several games have already been developed in different types (action, adventure, exploration...) and are currently tested in schools specialized in visually impaired children.

1. The TiM Project

The game engine presented in this paper is part of the TiM (Tactile Interactive Multimedia) project [TiM2000]. The purpose of this project is to offer multimedia computer games intended for young children who are blind or severely visually impaired. TiM is also addressing visually impaired children with additional impairments in form of slight to moderate degree of learning or cognitive disability and/or a physical impairment [TiM2001]. These games are planned to be used by the children in an autonomous way, without assistance of a sighted person.

To reach the needs of these children, the games have to be adaptable to their specific needs and to all the specific devices they use, each corresponding to a specific modality: tactile boards, Braille displays, speech synthesizers, as well as standard devices like keyboards and sound devices or adjustable screen settings (scalable font prints, customizable colors and contrast...).

Additionally, the specificity of modalities used by visually impaired children often makes it necessary to modify the scenario of interaction. For instance, a lot of games are based upon the global vision of the layout and the visual memory.

TiM plans to handle all the aspects necessary to ensure the development of high quality games by developing tools that allow the adaptation of existing commercial computer games ensuring that educational and play contents are incorporated; It has also developed specific contents meeting the particular needs of a young blind or partially sighted child. TiM will also include complete material about adaptability, methodology and guidelines.

2. Blindstation overview

2.1. Game Engine Principle

In order to simplify game creation and adaptation in the TiM project, a game engine or platform, named Blindstation, was developed. To easily understand the concept of a platform, people could think about the existing hardware game platforms, like Nintendo or PlayStation.

While the TiM platform is purely made of software and runs on a standard PC, it has the same features: it provides to developers some high level functionalities to interpret a TiM game script with its resources, to process the data and to drive all the devices that are

connected to the player's computer. Thus simplifying game design.

2.2. Adaptability Specificities

The mainstream computer games are usually designed especially to be used through a standard multi modal interface (graphical display, mouse and speakers). So the approach for games development is based on visual conception: interaction objects of the game are generally represented in the design software as pictures which are attached to some parts of the screen.

In order to be able to design games that are independent from the modalities and can use specific devices, the approach chosen by the TiM project is to use a modality-independent model [Arch2001]. This is the reason why in the TiM platform, the game are described in 2 distinct parts: On the first hand the game scenario is written in a script using some abstract high level components. On the other hand, the resources (audio samples, music, texts, pictures, animations...) that can be used to render the different components are specified in a structured style sheet.

It is only during the execution of the game, that the platform, using the script and the style sheet will render the interface in a multi modal way according to the environment and the needs of the player.

3. Game Script

```
from blindstation import *
class Intro(Scene):
    def __init__(self, game):
        Scene.__init__(self, game, "test_scene")
        self.label = Widget(self, "label")
        self.button = Widget(self, "button")
        self.button.validate_callback = self.callback1
        self.listener = Listener()
        self.source = Source("source")
        self.source.source_attach = self.source_callback
        self.source.queue("thunder")
        self.source.queue("bird")
        self.source.play()

    def callback1(self, name):
        print "callback at validation of button Widget"
        self.next = Menu

    def source_callback(self, name):
        print "callback at the end of the sound"
```

A simple example of a script using the Blindstation

The scenario of a TiM game scenario is described in a game script written in the Python [Python] computer high level language and using the components provided by the Blindstation. This script is interpreted by the Python interpreter and uses functionalities of the platform through the "platform API". An API is an abbreviation of Application Programming Interface, a set of routines, protocols, and tools for building software applications.

As described above, the API is organized in different components completely independent of the game representation and providing functionalities to the game designer in order to simplify game design.

3.1. Standard Components

Some components are used to describe the game flow and can be found in most traditional game platform. For example, the Scene, just like in movies, is a way to divide a game in many small coherent sequences; the Game component represents the context in which the

whole game is running and is responsible for initializing many low level aspects of the platform (like loading the many libraries, initializing the display, the sound, the braille display...) and dealing with scenes changes.

The interaction in the platform depends mainly on events being generated. Each time something happen in the game (for example when a user interacts with a component), an event is emitted. Generally events trigger some callbacks defined in the game script. Some simple components like the Timer make it possible to trigger some events at some given time.

3.2. User Interaction Components

Those components are used to interact with the player. Since the interface of the game has to be adaptable, those components are very versatile. For instance, there is a Widget component. Depending on the context, this component can be rendered on the screen at a specific position with some text and pictures, but it can also be accessed in Braille from a Braille device or read thanks to a speech synthesis or activated with a tactile board cell.

Other components include a Navigator which makes it possible to “navigate” in a virtual environment with a joystick for example, a Listener indicating the player position or spacialized sounds.

Components are generally associated to a resource name that will be used by the platform to recognize which resources are available in order to render this component.

3.3. Resources Style Sheet

```
<blindstation>
<scene name='test_scene'>
  <color r='255' g='255' b='255' a='255' />
  [...]
  <widget name='button'>
    <text lang='us'>Validate for Menu</text>
    <text lang='fr'>Valider pour le Menu</text>
    <zone x='50' y='50' l='500' h='50' />
    <shortcut key='SPACE' />
    <color r='0' v='255' b='0' a='128' />
  </widget>
  <sound name='thunder'>
    <file lang='fr' name="data/thunder.wav" />
    <file lang='en' name="data/thunder.wav" />
  </sound>
  [...]
</scene>
</blindstation>
```

The Style Sheet corresponding to the previous script

As seen before, the scenario of TiM games is described in a script that is completely independent of the final representation. Since the games have to be adapted to different multi modalities, the resources available to represent a given component (like sounds, text, pictures) have to be stored in a structured way. An Extensible Mark-up Language (XML) document called the Style Sheet is used.

This idea is based on the same principle that can be found in web pages. In web documents, the structure of the text is described in an HTML file which contains only the structure of the text (a title, a section, an item of a list). Then another file called Cascading Style Sheets (CSS) is responsible for the final aspect of the document [WAI]. Our style sheet also contains information in order to deal with many languages.

4. Platform internals

The Python language is particularly well suited for the needs of this project since it is high

level and interpreted. However some power consuming parts were optimized by using C code wrapped using SWIG [SWIG].

The platform is light enough to run on a multimedia computer with a low configuration, a sound card, loudspeakers and a CD-ROM drive. The platform is portable and can run on systems with Windows 95 and upper or with GNU/Linux and possibly MacOS.

4.1. Game flow

At initialization, the platform will detect, if technically possible, which devices are available on the computer and initialize them. Devices can also be specified in a configuration file.

Then the platform has to deal with scene changes and to listen to all the possible events. An event is an information that propagates itself to trigger some functionalities of components and is generally generated when a user does an action. Depending on the nature of the event, it can be treated internally for performance issues.

4.2. Resources management

One of the operations that consumes a lot of memory is to load pictures, sounds, video, fonts and other resources. The platform uses a cache mechanism to load resources only when needed and to ensure that only one instance is loaded at any time. In order to guaranty good performances while playing, the platform also does use streaming when necessary, for example when playing big sound files.

4.3. Adaptative representation

The Blindstation is responsible for the rendering of the components described in the game script using the resources style sheet. Many parameters are responsible for the final representation: available devices, user profile, data collected about the user during the game, user's preferences stored in a configuration file... On top of that, multilingual features have been integrated so that the same game should be played by people speaking different languages. Using all those pieces of information, the platform is able to select the best possible representation for the user interface.

4.4. External libraries

To improve extendibility, the project uses some external libraries for specific things: Libbraille [Libbraille], which was created for the TiM project, is responsible for all the low level interactions with braille displays. Libspeech [Libspeech] drives the speech synthesis while libboard deals with tactile boards.

Other libraries developed externally are used, like OpenAL [OpenAL] for interactive, primarily spatialized audio. It can play the sounds in the player's headphones or a 5.1 surround system of speakers. Finally SDL [SDL], a cross-platform multimedia library, is used to provide fast access to the graphic frame-buffer, audio devices, keyboards and joysticks.

5. Current Status And Further Works

5.1. Adapted games

The first version of the Blindstation was developed and several games have been designed and are currently evaluated with children :

- *Reader rabbit's: Toddler* is an adaptation of a mainstream discovery game designed for very young children. It features many small puzzles based upon sound and tactile recognition, for children who cannot read
- *MudSplat* is an arcade like game where the player defeats mud throwing monsters by squirting water at them. It is mostly based on sound and navigation systems.
- *Tim's journey* is an exploration game. The player solves a quest, by exploring in real

- time a surround sound environment and listening to stories
- *X-tune* is a musical construction game where the player can sequence and play with different sound environments
- *Magic dictation* is an educational game for learning reading and writing; it is particularly well suited to children learning Braille

5.2. Game builder

A TiM authoring software is planned that will facilitate the adaptation or development of games by providing a friendly drag and drop interface to completely create the game. This tool is destined to be used by non-computer literates: professionals working in special schools or rehabilitation centers, resource centers, educators, teachers, parents...

5.3. Evaluation

The game are evaluated in different sites across 3 countries including special schools for blind children, rehabilitation centers for multi-handicapped blind children, ordinary schools which receive blind children and a parents association. The studies are carried out by a team of specialists from various disciplines: educators, teachers, ergonoms, psychologists and therapists. The feedback to the software developers and game content designers will improve the games and the representation of components in the platform.

Acknowledgements

The TiM project is funded by the European Commission, on the program IST 2000 (FP5 - IST - Systems and Services for the Citizen/Persons with special needs) under the reference IST-2000-25298. The contents of this paper is the sole responsibility of the authors and in no way represents the views of the European Commission or its services.

References

[TiM2000]: Archambault, D. and al, TIM : Tactile Interactive Multimedia computer games for visually impaired children., , <http://www.snv.jussieu.fr/inova/tim>

[TiM2001]: Archambault D., Burger D., and Sablé S., The TiM Project: Tactile Interactive Multimedia computer games for blind and visually impaired children, in Assistive Technology -- Added Value to the Quality of Life, In: Marincek C., Bühler C., Knops H., and Andrich R. (eds.), Proceedings of the AAATE'01 Conference ,2001

[Arch2001]: Archambault, D. and Burger, D, From Multimodality to Multimodalities: the need for independent models., in Constantine Stephanidis (ed.) Proceedings of the UAHCI'01 conference - Universal Access in Human-Computer Interaction (joint with 9th International Conference on Human-Computer Interaction) – Towards an Information Society for All. ,2001

[Python]: , Python, an interpreted, interactive, object-oriented programming language, , <http://python.org>

[WAI]: Chisholm, W., Vanderheiden, G. and Jacobs, Web Content Accessibility Guideline 1.0, Web Accessibility Initiative, , <http://www.w3.org/TR/WCAG10/>

[SWIG]: , Simplified Wrapper And Interface Generator, , <http://www.swig.org>

[Libbraille]: Sablé S. and Archambault D., Libbraille: a Portable Library to Easily Access Braille Displays, in Proceedings of the 8th International Conference ICCHP 2002 - Computer Helping People with Special Needs ,2002

[Libspeech]: , , <http://libspeech.sourceforge.net>

[OpenAL]: , Open Audio Library, , <http://www.openal.org>

[SDL]: , Simple Direct Layer, , <http://libsdl.org>