University of Crete

Computer Science Department

PUPIL: Pervasive **U**I develo**P**ment for the amb**I**ent c**L**assroom

by

MARIA KOROZI

MASTER'S THESIS

Heraklion, September 2010

University Of Crete
Computer Science Department

PUPIL

Pervasive **U**I develo**P**ment for the amb**I**ent c**L**assroom

by
MARIA KOROZI

A thesis submitted in partial fulfillment of the
requirements for the degree of

Master of Science

Author: _____
Korozi Maria, Department Of Computer Science

Board of enquiry:

Supervisor _____
Constantine Stephanidis, Professor

Member _____
Anthony Savidis, Associate Professor

Member _____
Panagiotis Tsakalides, Professor

Approved by: _____
Panos Trahanias, Professor
Chairman of the Graduate Studies Committee

Heraklion, September 2010

Abstract

The question of how computers and Internet can improve the efficiency and effectiveness of education has been in the foreground for the past decade. ICTs are acknowledged to bring great potential in the domain of education, especially when appropriately used to expand the access to information, raise the educational quality and transform it into an active process directly connected to the real life. In particular, Ambient Intelligence technologies can be directly linked with education and especially the notion of Smart Classroom.

Smart Classrooms, in general, aim to enhance the conventional classroom with sophisticated technologies to support the learning process. The numerous existing approaches address the technological augmentation of the classroom, but do not pay enough attention to the actual student needs or the designers, which are called to build complex “smart” educational applications.

The PUPIL system, reported in this thesis, in collaboration with the ClassMATE infrastructure, aims to lay the foundation towards the realization of the AmI Classroom and assist the students throughout their journey to knowledge. In particular, the PUPIL system has been developed following a user centered design process (UCD), and puts to practice a set of tools and guidelines to promote the design of usable educational applications that can migrate among various technologically augmented classroom artifacts. The designers are equipped with a library of widgets that can be appropriately transformed to achieve optimal display on the available artifact. This library does not contain only common UI elements, but also incorporates mini interfaces that are frequently used by educational applications as ready-to-use modules. Therefore, the designers’ work is minimized, since numerous design decisions have been made a priori.

Furthermore, considering the diversity of characteristics among the classroom artifacts, the PUPIL system aims to support the students through their everyday educational activities by introducing appropriate workspaces (Window Managers) to host the applications and facilitate studying.

This thesis discusses the adopted design process, the derived design guidelines for Ambient Intelligence classroom applications, the developed widget library and Windows Managers, and reports a case study concerning the development and evaluation of a series of applications implemented using the developed tools.

Περίληψη

Την τελευταία δεκαετία το ενδιαφέρον για τον τρόπο που οι υπολογιστές και το διαδίκτυο μπορούν να επηρεάσουν την αποδοτικότητα και αποτελεσματικότητα της εκπαίδευσης είναι αρκετά έντονο. Οι τεχνολογίες πληροφοριών είναι αποδεδειγμένο ότι μπορούν να προσφέρουν νέες δυνατότητες στον τομέα της εκπαίδευσης, ιδιαίτερα δε όταν χρησιμοποιούνται κατάλληλα για να επεκτείνουν την πρόσβαση σε πληροφορίες και να βελτιώσουν την ποιότητα της εκπαίδευσης. Συγκεκριμένα, οι τεχνολογίες διάχυτης νοημοσύνης είναι άρρηκτα συνδεδεμένες με την εκπαίδευση στα πλαίσια της εφαρμογής τους για την υλοποίηση της "έξυπνης" τάξης.

Ο όρος "έξυπνη" τάξη δεν είναι καινούριος. Αναφέρεται σε ένα εκπαιδευτικό περιβάλλον κατάλληλα ενισχυμένο και τεχνολογικά εξοπλισμένο στοχεύοντας στην διευκόλυνση και υποστήριξη μαθητών και καθηγητών. Η "συμβατική" τάξη μεταμορφώνεται και εφοδιάζεται με την τελευταία λέξη της τεχνολογίας όπως αλληλεπιδραστικούς πίνακες, οθόνες αφής και φορητές συσκευές έτσι ώστε οι εκπαιδευτικές δραστηριότητες να επεκταθούν και να υποβοηθηθούν κατάλληλα.

Ωστόσο, παρά το γεγονός ότι η εκπαίδευση έχει στην διάθεσή της τις πιο εξεζητημένες τεχνολογίες, οι ήδη υπάρχουσες προσεγγίσεις περιορίζονται σε αυτοματισμούς που διευκολύνουν την διεξαγωγή της διδασκαλίας, την ηλεκτρονική καταγραφή των διαλέξεων του καθηγητή, την ανταλλαγή πληροφοριών μεταξύ συμμαθητών και την υποστήριξη απομακρυσμένων μαθητών. Είναι λοιπόν απαραίτητες πιο συστηματικές προσεγγίσεις που λαμβάνουν υπόψη τις ιδιαίτερες ανάγκες των μαθητών, καθώς και τις προκλήσεις που καλούνται να αντιμετωπίσουν οι σχεδιαστές "έξυπνων" εκπαιδευτικών εφαρμογών για ένα τέτοιο περιβάλλον.

Το σύστημα PUPIL που αναπτύχτηκε στο πλαίσιο αυτής της εργασίας, βασιζόμενο στην υποδομή που προσφέρει το ClassMATE, προσπαθεί να θέσει τα θεμέλια για την υλοποίηση μιας "έξυπνης" τάξης που έχει στο επίκεντρο τον μαθητή. Πιο συγκεκριμένα, το PUPIL αναπτύχθηκε ακολουθώντας μια ανθρωποκεντρική διαδικασία σχεδίασης με στόχο την δημιουργία εργαλείων και την εκμαίευση κανόνων που προάγουν την σχεδίαση εύχρηστων εκπαιδευτικών εφαρμογών, κατάλληλων για όλες τις "έξυπνες" συσκευές της τάξης. Για την επίτευξη αυτού του στόχου, οι σχεδιαστές εξοπλίζονται με μια βιβλιοθήκη από αλληλεπιδραστικά αντικείμενα διεπαφών, που μπορούν να προσαρμοστούν κατάλληλα για να επιτύχουν την βέλτιστη εμφάνισή τους σε οποιαδήποτε "έξυπνη" συσκευή. Η βιβλιοθήκη αυτή δεν αποτελείται μόνο από βασικά αντικείμενα, αλλά ενσωματώνει μικρο-

διεπαφές (συχνά χρησιμοποιούμενες από εκπαιδευτικές εφαρμογές) σε συμπαγή μορφή. που μπορούν να χρησιμοποιηθούν απευθείας από τους σχεδιαστές. Με τον τρόπο αυτό το έργο των σχεδιαστών διευκολύνεται, καθώς βασικές σχεδιαστικές αποφάσεις έχουν ήδη ληφθεί εκ των προτέρων.

Επιπλέον, το PUPIL γνωρίζοντας την ποικιλομορφία χαρακτηριστικών που διακρίνει τις "έξυπνες" συσκευές της τάξης, διευκολύνει τους μαθητές που τις χρησιμοποιούν κατά την διάρκεια των εκπαιδευτικών τους δραστηριοτήτων. Για αυτό τον λόγο, αναπτύχθηκαν κατάλληλοι "χώροι εργασίας" (Window Managers), οι οποίοι φιλοξενούν τις εκπαιδευτικές εφαρμογές και διευκολύνουν την χρήση τους από τους μαθητές.

Αυτή η εργασία παρουσιάζει την διαδικασία σχεδίασης που ακολουθήθηκε και καταγράφει τις οδηγίες σχεδίασης "έξυπνων" εκπαιδευτικών εφαρμογών που προέκυψαν, περιγράφει την βιβλιοθήκη που δημιουργήθηκε για την υποστήριξη των σχεδιαστών καθώς και τους "χώρους εργασίας" που απευθύνονται στους μαθητές. Τέλος, τεκμηριώνει την χρησιμότητα του PUPIL, μέσω μιας πειραματικής μελέτης που περιλαμβάνει την ανάπτυξη και αξιολόγηση διάφορων εκπαιδευτικών εφαρμογών.

Ευχαριστίες (Acknowledgements)

Αρχικά θα ήθελα να ευχαριστήσω τον επόπτη της μεταπτυχιακής μου εργασίας, κ. Κωνσταντίνο Στεφανίδη, για την συνεχή καθοδήγηση και υποστήριξη που μου προσέφερε τα τελευταία τέσσερα χρόνια στο πλαίσιο της συνεργασίας μου με το Εργαστήριο Αλληλεπίδρασης Ανθρώπου-Υπολογιστή, του Ινστιτούτου Πληροφορικής του Ιδρύματος Τεχνολογίας και Έρευνας. Η συνεργασία αυτή με βοήθησε να εξελιχθώ και να αποκτήσω γερά θεμέλια στον συγκεκριμένο τομέα.

Στη συνέχεια θα ήθελα να ευχαριστήσω θερμά τη Μαργαρίτα Αντόνα, τη Σταυρούλα Ντοά και τον Γιώργο Μαργέτη για την πολύτιμη βοήθεια και συμβουλές τους κατά τη διάρκεια της σχεδίασης, ανάπτυξης και συγγραφής της συγκεκριμένης εργασίας. Δεν ξεχνώ όμως και την αμέριστη συμπαράσταση που μου προσέφεραν δημιουργώντας ένα άριστο περιβάλλον συνεργασίας για το οποίο δεν μπορώ παρά να είμαι ευγνώμων.

Επιπλέον, θα ήθελα να ευχαριστήσω τους φίλους μου Έφη, Γιάννη Λ. , Χρύσα και Γιάννη Δ., που μου συμπαραστάθηκαν σε όλους του τομείς κατά την ολοκλήρωση της μεταπτυχιακής μου εργασίας. Ξεχωριστά όμως θα ήθελα να ευχαριστήσω τον Στέριο, ο οποίος εκτός από τις σημαντικές συμβουλές που μου προσέφερε στα πλαίσια της παρούσας εργασίας βρισκόταν δίπλα μου όποτε τον χρειαζόμουν.

Τέλος το μεγαλύτερο ευχαριστώ το οφείλω στους γονείς μου Κώστα και Δήμητρα καθώς και τον αδερφό μου Γιάννη, οι οποίοι παρά την απόσταση που μας χώριζε, με στήριζαν και με εμψύχωναν καθημερινά. Η αγάπη τους και η κατανόηση τους με βοήθησαν κατά τη διάρκεια των σπουδών μου και ελπίζω αυτή η εργασία να αποτελεί μια μικρή «ανταμοιβή» για τις θυσίες που έκαναν για μένα.

To my parents Costas and Demetra

and

my brother John

Table of Contents

List of figures

List of tables

1 Introduction

1.1 Interactive technology in the classroom

In the recent past there has been a growing interest in how computers and the Internet can improve the efficiency and effectiveness of education at all levels. Information and communication technologies (ICTs) are acknowledged as potentially powerful tools for educational change and reform. When used appropriately, different ICTs are said to help expand access to information, strengthen the relevance of education to the increasingly digital workplace, and raise educational quality by, among others, helping make teaching and learning an engaging, active process connected to real life [46].

The students benefit particularly from the use of ICTs in education, since the access to educational information is unlimited, the learning environment is enriched, active learning and collaboration is promoted, and motivation to learn is enhanced [1] In the recent past, learning with the use of ICT was strongly related to concepts such as distance learning [5], educational games [13], intelligent tutoring systems and e-learning applications [7].

The notion of smart classrooms has become prevalent in the past decade [48]. Smart classroom is used as an umbrella term, implicating that classroom activities are enhanced/augmented with the use of pervasive and mobile computing, sensor networks, artificial intelligence, robotics, multimedia computing, middleware and agent-based software [10]. Following the rationale of augmented technology in the educational environments, new means of interaction - such as interactive whiteboards, touch screens and tablet PCs - have gained popularity and have become a major tool in the educational process, allowing more natural interaction. Smart classrooms, for example, may support one or more of the following capabilities: video and audio capturing in classroom [43], automatic environment adaptation according to the context of use, such as lowering the lights for a presentation [11], lecture capturing enhanced with the instructor's annotations, or information sharing between class members.

1.2 Ambient Intelligence for Education

The AmI Programme of ICS-FORTH envisions a smart classroom consisting of technologically enhanced artifacts purposed to serve teachers' and students' particular needs. These artifacts aim to replace the traditional student desks, teacher desks and classroom boards, in order to support the developmental processes of disseminating and receiving knowledge. All the classroom artifacts will operate in an ambient environment that understands and

adapts to the needs of each specific user (teacher or student). For instance, consider a teacher entering the classroom and sitting in his/her desk; the system, being aware of the course that will take place, reminds the teacher the exact point that the lecture had stopped the day before. Similarly, imagine that a student sitting on his/her desk is immediately informed about the homework exercises that must be submitted to the teacher. However, the AmI Classroom possibilities are not limited to simple automations; contrariwise, a list of supported features is presented below:

- Student facilities
  - Access to unlimited educational information
  - Dedicated storage area for student's material
  - Access to personal material and classroom applications at anytime through personal devices (PDA's, Netbooks)
  - Personalized content, help and study guidelines for each student
  - Records of student progress
  - Collaboration among classmates
  - Active participation of students in the teaching process
  - Automation of everyday tasks
- Teacher facilities
  - Lecture preparation assistant
  - Statistics of class progress
  - Real-time student monitoring
  - Automation of everyday tasks

The PUPIL and ClassMATE [26] systems aim to lay the foundations for the realization of the AmI Classroom. The collaboration of these systems realizes an augmented classroom as the one depicted in Figure 1, and they can be easily scaled to accommodate any envisioned AmI Classroom. The ClassMATE system offers the backbone infrastructure that enables context awareness, facilitates service resolution independently of the prevailing platform, offers a generic mechanism for heterogeneous devices manipulation, and encapsulates a sophisticated content discovery and personalized content delivery mechanism.

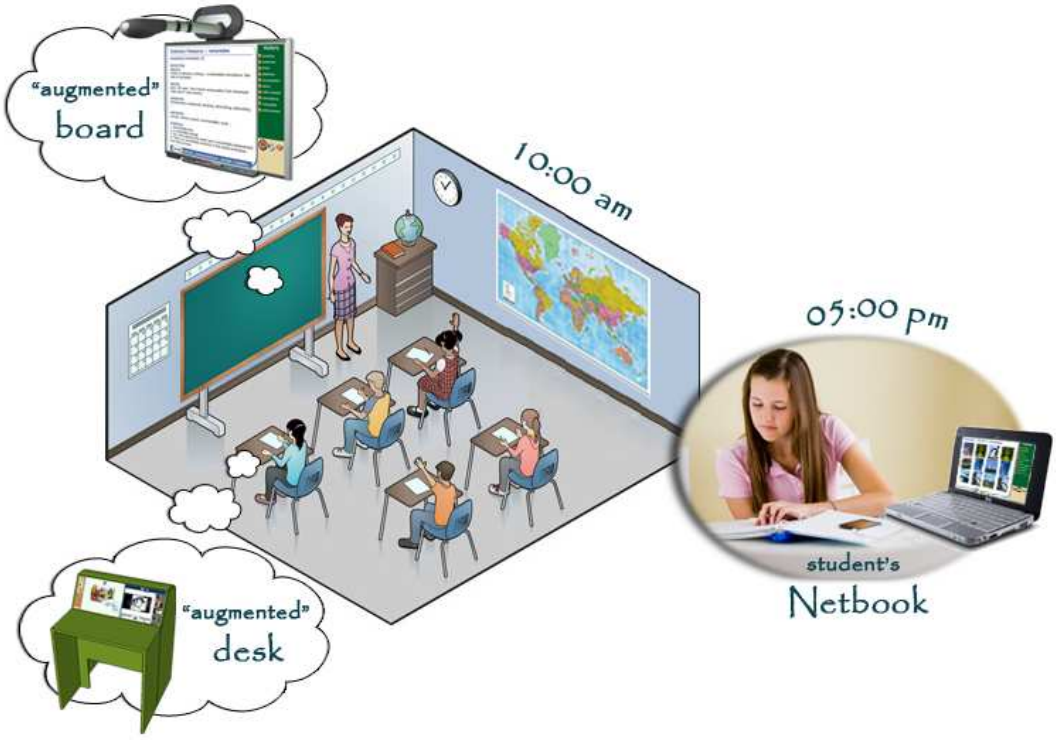


Figure 1: The augmented classroom realized by the PUPIL and ClassMATE systems

1.3 Objectives of the PUPIL system

The PUPIL system, presented in this thesis, aims to empower the AmI Classroom to successfully assist the students throughout their journey to knowledge. The current realization of the AmI classroom, as depicted in Figure 1, comprises an augmented board and augmented student desks, while the student's Netbook is supported for use either in the classroom environment or at home. The existence of various artifacts inside the classroom creates the need of educational applications that can be displayed appropriately on each one of them. For example, the student can choose to view an application privately in the desk or Netbook artifact, or share it with the entire class through the augmented board. However, application migration entails the risk that the application interface will not be suitable for all the available artifacts, due to different screen sizes, interaction styles and usage modalities. The PUPIL system aims to support application migration without compromising the visibility, legibility and usability of the included UI elements, greatly facilitating the development of AmI classroom applications which can adapt themselves to diverse interactive artifacts, form the very large (e.g., the classroom board), to the very small (e.g., potentially, a mobile phone). To this purpose, it equips the designers of educational applications with a library of widgets that can be appropriately transformed to achieve optimal display at the various artifacts. In addition, the widget library does not contain only

common UI elements (e.g., buttons, textboxes), but also incorporates mini interfaces (e.g., Image Displayer, Multiple Choice Question & Answers, etc.) that are frequently used by educational applications as ready-to-use modules.

Furthermore, considering the diversity of characteristics among the classroom artifacts, the PUPIL system introduces appropriate workspaces (Window Managers) to host the applications and facilitate the educational activities. An illustrative example is the Window Manager purposed for the student's desk, which allows parallel work with two applications (one next to the other), while the classroom board displays only one application to avoid distraction. Finally, a set of design guidelines that emerged through the design process of the PUPIL system is exposed to the designers of educational applications. The interfaces generated through the PUPIL system comply with these guidelines; however designers can use them independently to create usable educational applications.

The detailed goals of the PUPIL system are:

- Promote the design of usable educational applications
- Enable application migration among classroom artifacts
- Free designers from building the same interface for various artifacts
- Minimize artifact specific design decisions
- Enable reusability of common interface patterns
- Ensure scalability in terms of supported artifacts and library widgets
- Facilitate educational activities
- Equip each artifact with appropriate workspaces

1.4 Structure of this Thesis

Through the remaining chapters of this thesis it becomes clear how the PUPIL system achieves its goals.

Chapter 2 briefly presents previous work related to the PUPIL system. The presented systems are categorized into: (i) systems that focus on interface adaptation with respect to the user's or platform's characteristics, and (ii) systems that promote an augmented classroom environment and facilitate teaching and learning activities.

Chapter 3 provides a detailed overview of the requirements elicitation process that informed the design analysis of the PUPIL system. In particular, the steps towards a User Centered Design process, such as literature review, scenarios and prototypes development,

and children interviews, are thoroughly analyzed. This chapter concludes with a set of design guidelines for educational applications that emerged from the conducted requirements analysis.

Chapter 4 presents the widget library offered to the designers, highlighting its benefits for the development of AmI classroom applications. Both simple and complex widgets are described in details, while the mechanisms that enable widget transformations are thoroughly inspected. Furthermore, this chapter presents the mechanisms built to support the various Window Managers that host educational applications and describes the characteristics of the ones currently.

Chapter 5 provides an overview of a number of applications developed through the PUPIL system as a case study validating the overall approach as well as its implementation in the PUPIL system.

Chapter 6 describes the heuristic evaluation process on the developed applications. The results of that evaluation have analyzed to collect useful feedback and eliminate any severe usability problems in the widgets provided by PUPIL's library.

Finally, Chapter 7 summarizes the conclusions of the thesis and outlines potential enhancements for future work.

2 Related Work

This section briefly presents previous work related to the PUPIL system. The presented systems are categorized into: (i) systems that focus on interface adaptation with respect to the user's or platform's characteristics, and (ii) systems that promote an augmented classroom environment and facilitate teaching and learning activities.

2.1 Toolkits for User Interface Adaptation

User interface automatic adaptation has been recognized in recent years as a technical solution for supporting accessibility and enhancing usability of interactive applications and services. A methodological framework for the development of user interfaces capable of automatic adaptation is the Unified User interface Development Methodology [41]. Unified User Interfaces embed knowledge of the relevant characteristics of users (e.g., age, abilities, computer expertise), interaction platforms and the environment of use, decision making capabilities, as well as libraries of alternative interaction objects which can be activated and deactivated as the need arises during the adaptation process. Adaptation can take place at the beginning of interaction based on static user and context characteristics (adaptability), or at run-time based on characteristics which are subject to change during interaction (adaptivity). Various tools have been developed in order to support user interface adaptation within the Unified user interface development framework [44]. A latest development is the provision of interaction toolkits which directly embed adaptation capabilities in the available interaction widgets, so as to support the easy development of user interfaces which adapt themselves at the level (e.g., size, colour, fonts, etc.).

This section reviews toolkits that support interface adaptation. The JMorph, Eager and CMF^{4}ALL systems, developed based on the Unified User Interface methodology, address accessibility issues and provide libraries of self-adaptive widgets that can be used as-is by designers to build interfaces that can transform according to the end-user's needs. On the other hand, the last three examined systems provide mechanisms to support the migration of user interfaces to various platforms without compromising interaction.

2.1.1 JMorph

The JMorph adaptable widget library [27] is a toolkit that inherently supports the adaptation of user interface components. It contains a set of adaptation-aware widgets designed to satisfy the needs of various target devices –Swing-based components for PC, AWT-based components for Windows Mobile devices. Adaptation is completely transparent to developers, who can use the widgets as "conventional" UI building blocks.

The JMorph library provides the necessary mechanisms to support alternative look and feels either for the entire environment (i.e., skins) or for individual applications. The implemented adaptations are meant to address the interaction needs of older users, and follow specific guidelines, which have been encoded into decision-making rules in the DMSL language [42]. The Decision Making Specification Language (DMSL) is a rule-based language specifically designed and implemented for supporting the specification of adaptations. DMSL supports the effective implementation of decision-making, and has been purposefully elaborated to be easier for designers to directly assimilate and deploy, in comparison to programming-based approaches using logic-based or imperative-oriented programming languages.

This approach is targeted to novice developers of adaptable user interfaces, and relieves them from the task of re-implementing or modifying their applications to integrate adaptation-related functionality.

The developed widgets are built in a modular way that facilitates their further evolution, by offering the necessary mechanism to support new features addition and modifications. Therefore, more experienced developers can use their own adaptation rules to modify the adaptation behavior of the interactive widgets.

2.1.2 EAGER

As opposed to the JMorph approach, which concentrates on desktop applications, EAGER [14] is a toolkit built to support WUI (Web User Interface) adaptation and facilitate the design of web interfaces that can adapt to the diversity of the target user population. In particular, EAGER integrates a Design repository of:

- alternative primitive UI elements with enriched attributes (e.g., buttons, links, radios, etc.)
- alternative structural page elements (e.g., page templates, headers, footers, containers, etc.)
- fundamental abstract interaction dialogues in multiple alternative styles (e.g., navigation, file uploaders, paging, text entry).

The EAGER Designs Repository contains implemented and ready-to-use alternative elements satisfying the requirements posed by specific user and context parameters values. These alternative styles were designed following a User Centered Design approach combined with iterative prototyping and guidelines compliance. Additionally, EAGER design alternatives provide a suitable approach to personalized accessibility.

2.1.3 CMF4ALL

Concluding with the first category of related systems, CMF4ALL [28] is a framework that facilitates the universal design and development of usable and accessible CMSs (Content Management Systems), by hiding from designers and developers any user related information, such as accessibility needs. Thus, designers do not have to be experts or be aware of accessibility and user interface design issues, while developers do not have to alter their ordinary development tasks in order to integrate user centric features in their work.

CMF4ALL defines a Widget Library of the most commonly used widgets used in CMS applications, where each widget encapsulates all the necessary complexity to support adaptation according to the user's requirements and the context of use. The realization and adaptation of the CMF4ALL widgets is performed one step before the finalized web page is rendered to the user's browser.

In the application development life cycle, a user interface should be initially designed by the designer as a non-interactive mockup (electronic or not) and then be implemented by the developers. This process demands both the designers' and the developers' effort in order to achieve the transformation of UI mockups into functional components of the web application. CMF4ALL improves the aforementioned process by allowing designers to produce ready-to-deploy UIs, while developers only have to integrate the essential business logic.

This is achieved by determining UI designs in a specific XML format, while the actual UI instances are automatically generated at runtime by an accessibility-aware multi-platform translation engine [25], a core component of the CMF4ALL architecture.

2.1.4 TERESA

The TERESA tool [32] supports the design and development of nomadic applications, providing general solutions that can be tailored to specific cases. Through TERESA designers can either specify the appearance of common UI elements for the supported platforms or even modify some general design assumptions. This tool supports transformations in a top-down manner, providing the possibility of obtaining more concrete descriptions starting with abstract representations. Upon designer's specifications the TERESA tool can generate appropriate XHTML files for each of the supported platforms.

2.1.5 Flexible Interface Migration

The work of R. Bandelloni and F. Paterno in [4] provides a mechanism for facilitating interaction with an application in a multi-platform environment while freely moving from

one device to another. Users are able to change device and continue their interaction from the same point. The platform-aware runtime migration process of Web applications is managed through a service which takes into account the application's runtime state and adapts the application interface to the features of the target platforms. It is optimized for applications developed through a model-based, multiple-level approach. The intelligence of the adaptive interfaces resides in the migration server, which adapts data collected at runtime from their original format to the format best fitting the features of the target platform. Finally, the proposed platform supports through the same infrastructure partial migration and synergistic access, through keeping a part of the user interface on one device and moving the remaining part to another with different characteristics.

2.1.6 Voyager

The Voyager development framework [40] supports the implementation of ambient dialogues, i.e., dynamically distributed user Interfaces, which exploit, on-the-fly, the wireless devices available at a given point in time. The primary motivation of Voyager is based on the vision that the future computing platforms will not constitute monolithic ''all-power-in-one'' devices, but will likely support open interconnectivity, enabling users to combine the facilities offered by distinct devices on-the-fly. Physically distributed devices may be either wearable or available within the ambient infrastructure (either stationary or mobile), and may be connected via a wireless communication link for easier deployment. Operationally, each such device will play a specific role by exposing different processing capabilities or functions, such as character display, pointing, graphics, audio playback, speech synthesis, storage, network access, etc. From the hardware point of view, such devices may be wrist watches, earphones, public displays, home appliances, office equipment, car electronics, sunglasses, ATMs, etc. The Voyager implementation focuses on device discovery and registry architecture, device-embedded software implementation, ambient dialogue style and corresponding software toolkit development, and a method for dynamic interface adaptation, ensuring dialogue state persistence.

2.2 Classroom Augmentation

This section discusses systems that aim to augment the classroom environment by assisting the teaching process, recording lecture activity, enabling collaboration among students, supporting remote student telepresence, and encouraging active student's participation and contribution in the learning process.

The system reported in [11] aims to improve the teaching experience by offering automated device control and creating a suitable environment for each detected situation. For example, when an instructor logs on to the classroom computer, the system infers that a computer-based lecture will be given, automatically turns off the lights, lowers the screen, turns on the projector, and switches the projector to computer input. However, it is emphasized that manual override functionality should be provided in order to permit modifications on behalf of the instructor. Additionally, the system promotes a wired classroom capable of recording a digital version of any presentation, including audio and video, as well as the instructor's slides and notes written during the lecture. Finally, a presenter-tracking algorithm was developed in order to follow the instructor's movements and improve the quality of video capture.

Similarly to [11], Y. Shi et al. in [43] try to facilitate the lecturers by providing alternatives for command execution that do not require being in direct contact with the computer. In this approach, the teacher can use a traditional laser pen to indicate and circle a point of interest, while a camera detects the user's intention and sends a mouse click message to the computer. However, this way of clicking turned out to be quite difficult so the speech recognition technique was also adopted. This technique allows the teacher to dictate commands such as "Jump to Next Page" or "Go ahead", to the slide presentation program. The system also aims to support remote students in a more efficient way than conventional real-time teaching systems do. A "Smart Cameraman" interprets a set of actions (e.g., teacher on the board, teacher showing a model, remote student speaking, etc.) and focuses on the objects/people of general interest.

In order to endorse students' collaboration and active participation in the learning process H. Breuer et al. [6] developed a platform to support the programming of distributed applications running on different platforms. In their approach, applications running on different platforms and sharing an object will have automatically the status of the object synchronized. An interactive whiteboard application is offered, named "DeepBoard", that implements a gesture-based interaction paradigm and a hierarchical semantic to store and retrieve data, created on the fly, by free-hand writing. In order to enable several pen-tablets to interact simultaneously within an application program, an intermediate software layer that receives and interprets the packets generated by the pen-tablets was added to the DeepBoard. These packets generate mouse events that are sent to the modified DeepBoard application, which can process them simultaneously. Finally, collaboration is promoted

through MCSketcher, a system that enables face-to-face collaborative design based on sketching on handheld devices. MCSketcher may be used by students inside or outside the classroom to exchange ideas through sketches on empty sheets or over a recently taken photograph of the object being worked on.

Collaboration among students is also supported in the Smart Classroom proposed by S. Yau et al. [49]. In the Smart Classroom, each student has a situation-aware PDA. Students' PDAs dynamically form mobile ad hoc networks for group meetings. Each PDA monitors its situation (locations of PDAs, noise, light, and mobility) and uses situation to trigger communication activity among the students and the instructor for group discussion and automatic distribution of presentation materials. Middleware can effectively address situation-awareness and ad hoc group communication for pervasive computing, by providing development and runtime support to the application software.

2.3 Discussion

The JMorph, EAGER and CMF4ALL toolkits focus on the diversity of users and contexts of use, and provide libraries of self-adaptive widgets to address the needs of the end-users. The PUPIL system share some similar aspects with these systems, as it aims to support the diversity of artifacts that are available to students, and it equips designers with a library of self-transforming widgets that ensures optimal display of applications when traveling among the classroom artifacts. However, the offered library does not contain only common UI elements (e.g., buttons, textboxes), but also incorporates mini interfaces (e.g., Image Displayer, Multiple Choice Question & Answers, etc.) that are frequently used by educational applications as ready-to-use modules. Consequently, the designer's work is significantly simplified, as numerous design decisions made a priori can be easily reused.

The Voyager system on the other hand offers a language for the designers to exploit in order to build applications that can migrate among various everyday devices (e.g., wristwatch, mobile phone, etc.). The logic of translating the designers' specifications to appropriate UI elements is encapsulated in the target devices. The resulting interfaces consist of widgets supported by device-specific UI libraries. However, the target platforms of this approach do not support sophisticated user interfaces and "appealing" graphics that are essential when designing for students.

As far as the TERESA tool is concerned, the benefits to the designers are that they can easily filter the content that will be displayed to the various platforms and can modify the appearance of specific UI elements. However, the designer is responsible for determining

the exact layout of an application for each available device. On the contrary, using PUPIL the designer builds the application having in mind an abstract artifact, and during runtime the appropriate transformations take place to achieve optimal display, without burdening the designer with complicated decisions.

The approach of R. Bandelloni and F. Paterno in [4] promotes the synergistic access among different devices and the state awareness of the applications, whereas PUPIL in collaboration with ClassMATE [26] supports application migration where the user can continue with his/her work on the same UI.

Finally, the classroom adaptation approaches envisioned the “smart” classroom as an environment that assists the instructor and enables students’ remote access, collaboration and active participation during the lectures. However, the PUPIL system approaches this concept from the students’ perspective and aims to facilitate the developmental process of learning. The student has in his quiver a collection of workspaces tailored to the needs of the available artifacts (i.e., netbook, desk, board) and a library of course-related applications that facilitate studying and transform boring school “chores” into pleasant activities.

3 Requirements Elicitation towards the Pervasive Classroom UI

3.1 User-Centered Design

The PUPIL system targets the realization of a technology-augmented classroom that supports the students throughout their journey to knowledge. Since PUPIL's philosophy evolves around the students' benefits, it was of foremost importance to concentrate on their needs, wants and limitations. In order to efficiently address these user requirements the students should be actively involved in the design process. Consequently, User-Centered Design (UCD) was selected as the most appropriate design approach.

Many definitions of User-Centered Design exist, mainly focusing on the user and emphasizing that the user's perspective should be incorporated in all design stages. It is an approach to design that concentrates on information about the people who will finally use the product, trying to meet their demands and expectations. Donald Norman describes UCD as "a philosophy based on the needs and interests of the user, with an emphasis on making products usable and understandable" (Norman 2002, p.188).

The User-Centered Design process is described in the ISO-standard 13407 [22], which provides guidance on human-centered design activities throughout the life cycle of interactive computer-based systems and illustrates an iterative development cycle consisting of four steps, as depicted in Figure 2:

- **Understand and specify the context of use**. It is important to understand and identify the details of the context of use in order to guide early design decisions. This specification should include aspects such as the characteristics of the intended users, the tasks the users will perform, the environment (physical or ambient) in which the users will use the system, etc.
- **Specify user and organisational requirements**. The documentation of user and organizational requirements is of outmost importance to ensure that the final product will meet their needs.
- **Produce designs and prototypes of plausible solutions**. The potential design solutions should be based on established state-of-the-art practices, the experience and knowledge of the participants and the results of the context of use analysis. The design could be improved by allowing representative users to perform tasks (or simulated tasks) on the prototypes and gathering useful feedback. This process should be repeated until the design objectives are met.

- **Carry out user-based assessment**. This is an essential step, which assesses whether user and organisational objectives have been met and provides feedback to improve design.

ISO 13407 has been revised in ISO 9241-210:2010 [23], which reflects recent changes and advances, and highlights six main principles for human-centered design:

- the design is based upon an explicit understanding of users, tasks and environments
- users are involved throughout design and development
- the design is driven and refined by user-centered evaluation
- the process is iterative
- the design addresses the whole user experience
- the design team includes multidisciplinary skills and perspectives

In conclusion, user centered-design is not simply conducting usability studies or talking to users; it emphasizes the active involvement of users and requires studies to understand users, as well as to drive and evaluate the design and the final system.

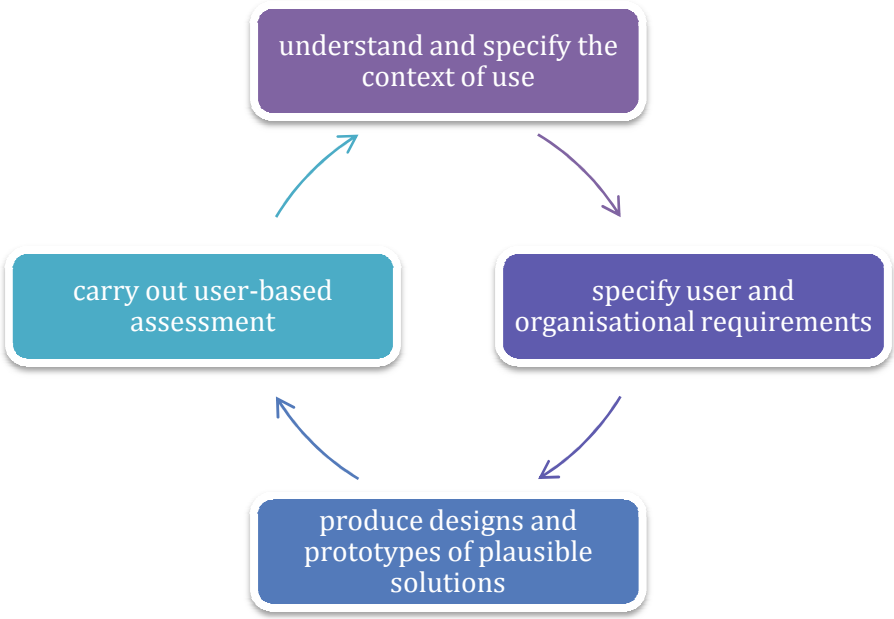


Figure 2: The UCD iterative development cycle

The process that was followed for the design of the PUPIL system was the UCD approach as represented in Figure 2, updated so as to follow all the human-centered design principles defined by the more recent ISO 9241-210:2010. Under this perspective, an iterative design

process was followed, driven by user-centered evaluation, consisting of the following steps Figure 3:

- The users' characteristics, their tasks and the environment were analysed through literature review and brainstorming sessions[1] with a team of experts, which included members of various backgrounds, skills and perspectives. In more detail, the team comprised participants of both genders (3 male, 4 female) and with a variety of professional roles (two designers, two usability experts, two developers, and a university professor).
- Based on the above specifications, detailed scenarios where elaborated, in order to be used as tools for the next step, which entailed the involvement of real users through interviews in order to specify their requirements.
- In order to visualize plausible solutions, a set of prototypes were created and evaluated by children during a formative evaluation experiment.
- The results of this evaluation were taken into consideration during the development process in order to better meet the end users' needs.
- Once the development phase had advanced and interactive prototypes had been created, a heuristic evaluation was conducted, aiming to eliminate serious usability problems before proceeding to a user testing.
- The PUPIL system was improved according to the heuristic evaluation results.
- A user-based evaluation of the improved system is planned to be the next step of this work.

[1] Brainstorming is a group creativity technique, bringing together a set of experts to inspire each other in the creative, idea generation phase of the problem solving process [37],[13].

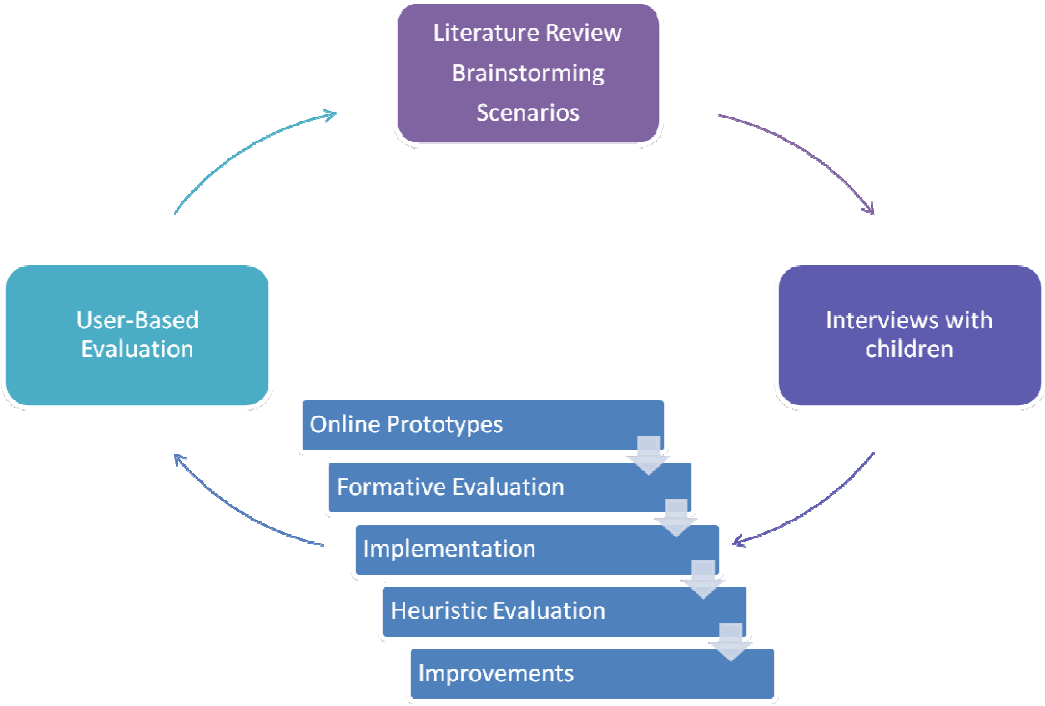


Figure 3: PUPIL's iterative design process

3.1.1 Understand and specify the context of use

The context of use specification can be realized by (i) understanding the user characteristics (i.e. the users' skills, needs, fears, etc.), (ii) by defining the tasks the users will perform and (iii) by analysing the environment in which the system will operate.

3.1.1.1 Characteristics of the users

The end users of the PUPIL system are students that attend grades higher than the fifth grade of the elementary school. Thus, the examined ages range between 11 and 18 years. Generally, in HCI design children should be regarded as a different user population with its own culture and norms [21]. Designing for students of these ages is quite challenging, as their needs and preferences differ from those of adults.

Jean Piaget [38], a Swiss Psychologist, divided children's cognitive development in a series of four stages:

- Sensori-motor (birth to 2 years)
- Preoperational (ages 2 to 7)
- Concrete operational (ages 7 to 11)
- Formal operational stage (ages 11 and up)

More recent studies [8] acknowledge that each child develops differently, and point out that these stages might be only theoretical. However, Piaget's study introduces a generic model of children characteristics, and shows that children do not just lack knowledge and skills, but also fundamentally experience and understand the world differently than adults.

The PUPIL system focuses on students that have reached the formal operational stage. During this stage, children develop the ability to think about abstract concepts and acquire skills such as logical thought, deductive reasoning, and systematic planning. Thus, at this stage the children's thinking process is similar to that of adults, while their tastes and interests remain different. Consequently, a student that has reached the formal operational stage is able to apprehend – if not already familiar– abstract concepts (e.g., gestures, folders etc.) supported by computer systems. This is also a consequence of the fact that today's children are exposed from a very early age to a wide range of technologies, including multimedia systems, electronic toys and games, and communication devices [3].

According to Nielsen Norman Group studies [29] that address teenagers, teens have much less research experience than adults, and therefore have more difficulty combing through and making sense of complex information. Furthermore, teens tend to have less patience and give up faster than adults, thus they like getting answers quickly and dislike complicated interactions. Finally, as teenagers are particularly sensitive about their age, they prefer a clean and modest design, with visually meaningful icons and age-appropriate instructions easy to comprehend and remember [20], over a childish one.

Druin et al. [16] have found that children when using technology want to be in control, share, show and use technologies with others, and have at their disposal expressive tools. They pay attention to various aspects of technology. For example, whether it is "cool" or not, easy to learn, appealing, and if multimedia is available in order to ensure a multisensory experience. As far as the use of multimedia is concerned, Said's studies with children aged from 9 to 14 years old indicated that children become more engaged with a multimedia application when immediate feedback is provided, as well as when the environment allows them to be in control and set their own goals [39].

Socializing is another characteristic of children at this age; however, it is not limited to talking to classmates, but includes activities like sharing interesting resources with friends or collaborating to solve an exercise. Kaplan et al. carried out a contextual inquiry regarding the reading practices of teenagers (10-14 years old), according to which even though children may be co-present in the same space, they would like to be supported with tools to share

their experiences with their friends and teachers. Furthermore, according to the same research, although children would like to have fun and be entertained, they recognize the limits of pure play and the value of learning and studying [24].

3.1.1.2 User Tasks

School aims to educate students, while any student wishes to acquire knowledge through interesting activities and complete school assignments pleasantly. The PUPIL system addresses the students' needs by transforming boring tasks to "student-friendly" activities, thus supporting them to achieve their goals and motivations.

Several studies have taken place since the early 1960 regarding student's goals and motives as drives for learning. The theory of "achievement motivation" was first developed in the early 1960s [12] by Atkinson and McClelland, claiming that achievement results from an emotional conflict between striving for success and avoiding failure. Furthermore, students' academic goals include learning and performance goals. Learning goals refer to one's aspiration to increase their understanding and skills [31], while performance goals involve outperforming others as a means to demonstrate their ability [12]. However, apart from the academic goals, students' social goals have also a profound effect on learning [47]. Such social goals include being appreciated by peers or teachers, becoming a productive member of society or honoring one's family. Having in mind students' goals, a list of tasks was devised that the students should be able to perform in the class and therefore should be supported by the PUPIL system. These include:

- solve exercises of various types (e.g., multiple choice, fill in the gaps, etc.) and write essays
- get assistance on exercises
- submit exercises that have been completed either during the lesson or at home
- retrieve additional resources about something interesting (e.g., an image displayed on the book) or about an assignment dictated by the teacher
- have access to assistive applications (e.g., calculator, dictionary, etc.)
- have access to multimedia
- maintain a personal area with access to
    - board's history, so that material written on the board during school-hours is always available
    - homework's history
    - history of exercises solved in the classroom

    - electronic material (e.g. course book, lecture notes provided by the teacher, etc.)
- collaborate with classmates to complete a task.

A scenario-based approach was adopted to further elaborate the characteristics of the aforementioned student tasks. Scenarios are narrative descriptions of interactive processes, including user and system actions and dialogue [3]. They are really useful both at early stages of the design process, providing realistic examples of how users carry out their tasks in a specified context, and for subsequent usability testing.

The detailed scenarios elaborated for PUPIL, presented in Appendix A - Scenarios used in Interviews, describe activities concerning an English course. These scenarios led to the identification of a set of software applications, including the student's personal area, the clipboard, which is available only on the desk artifacts and allows students to temporarily save material, the exercises in electronic form with the supported hints, a dictionary application and an application for viewing course related multimedia.

3.1.1.3 Environment

The environment's context of use can be defined by studying both the physical and the technical aspects of the end user's whereabouts. On the one hand, the classroom's physical environment is typically a small room (usually covering an area of 40-60 m^2) with an average of 22 students, while the students per class may vary from 30 or more to nearly half that number [35]. Due to these spatial characteristics and the students' impatient behavior, it is clear that in such a noisy environment students' attention cannot be taken as granted.

On the other hand, in terms of technical characteristics, a typical classroom consists of the students' desks and a board placed at the centre of the room. Moving towards a technologically enriched classroom, new computer-equipped artifacts could replace the class board and students' desks. The PUPIL system supports two types of desk artifacts, the AmIDesk and the SmartDesk. The AmIDesk artifact is a vision-based touch-enabled device [2], while the SmartDesk incorporates a typical touch screen. Similarly, two types of board artifacts are supported, the AmIBoard, which technologically resembles the AmIDesk, and the SmartBoard, which is a touch sensitive interactive whiteboard. In addition to these class-only artifacts, a portable artifact is supported which can be used both inside the classroom and at home, the student's netbook. The PUPIL system takes into consideration the special characteristics of these artifacts in order to optimally support the students and adapt he interaction accordingly.

3.1.2 Specify user requirements

As the developed scenarios of use describe in details a series of activities concerning the English course, it was of outmost importance to view them through the students' perspective. Thus, a series of interviews with children were conducted, in order to specify the user requirements.

Interviews are a technique frequently used in HCI, where users are questioned about their needs and requirements about a specific system [30]. There are three types of interviews (i) the unstructured interviews, where the interviewer intends to gather information concerning the user's experience and their expectations of the system, (ii) the semi-structured interviews, where the interviewer follows a predefined set of questions yet allowing new questions to be brought up during the interview, and (iii) the structured interviews where the interviewer has a specific list of questions to guide and direct the interview.

The semi-structured interview technique was followed for the design of the PUPIL system and five children of ages between 11 and 16 were interviewed individually. At the beginning of the interview, the children were briefed about the characteristics and abilities of the examined system and then the interviewer guided them through the scenario. During that process the children were asked questions concerning different aspects of the English course activities. Some of these questions were predefined (Appendix B - Questions used in Interviews) in order to highlight specific features of the system, while other questions were brought up during the interview according to the reactions of the children.

The children's opinions about the system during the interviews ranged from positive to enthusiastic. They were excited about the applications and the fact that they could share with the entire classroom interesting resources. Furthermore, the fact that they could work from their netbook appealed to them, as two of the children said: "studying would be much easier". Finally, the students raised some interesting issues, which are listed below:

1. They asked whether they could have access to educational and cooperative games
2. They wondered whether they could save interesting content (e.g., images) for access from home.
3. They asked if the multimedia application would be accessible during other courses. They thought that it would be useful in the Geography course too.

4. They asked if the teacher would deactivate the hints during a test, or when the student is examined. They suggested that the student should have the flexibility to turn them on/off.
5. They wondered if they could share anything they want and at any time with the entire classroom.
6. They asked if they could send an interesting resource (e.g., image) to a classmate.
7. They found the multiple-choice exercise interesting, and asked whether other types of exercises would be supported.
8. They suggested an organizer application, accessible both from their desk and netbook, to remind them their homework.
9. They asked for an application where they could save difficult words for further study.

To address the children's comments and suggestions, the initial list of User Tasks was enhanced to include the "be entertained" task, as the children like to play educational and cooperative games both at school and at home. Furthermore, they affected architectural decisions [26] concerning educational material storage (i.e., what, when, where) and action authorization (e.g., send resource to board during reading, ask for hint during test, etc.). Finally, they pointed out the importance of assistive applications, such as SchoolOrganizer, which displays homework or predefined exams, and MyVocabulary, which stores a list of words specified by the student for further studying.

3.1.3 Produce designs and prototypes of plausible solutions

3.1.3.1 Prototypes

At this stage of the design process, as the context of use and user requirements analysis created a pretty clear picture about the students' needs and requirements, a set of prototypes were created based on the scenarios built during the first step of this cycle and the feedback received from the interviews with the students.

A prototype is defined as a concrete representation of part or all of an interactive system. It is similar in concept to building a foam core model or other physical mock-up of hardware. It is also similar in concept to having a paper mock-up of a document to show the approximate size, shape, binding, page layout, etc.

In HCI, prototyping is primarily a design activity, although software engineering is used to ensure that software prototypes evolve into technically sound working systems and scientific methods are followed to study the effectiveness and acceptance of particular designs. Its goal is to provide an early opportunity to observe something about the nature of the final product, evaluating ideas and weighing alternatives before committing to one of them. Two forms of prototyping representation can be distinguished: offline and online.

The offline prototypes (also called paper prototypes) include paper sketches, illustrated storyboards, cardboard mock-ups and videos. Their advantages are that they are quick and inexpensive to be created usually in the early stages of design, with tools that everyone knows how to use, and are thrown away when they have served their purpose. Moreover, despite they only expose some of the final functionality they may encourage more suggestions because they seem more changeable. However, one significant disadvantage is that the paper versions may lack “face validity” to users and not be taken seriously enough to bring valuable results.

The online prototypes (also called software prototypes) include computer animations, interactive video presentations, programs written with scripting languages, and applications developed with interface builders. The cost of producing online prototypes is usually higher and may require skilled programmers to implement advanced interaction techniques, however they are usually more effective in the later stages of design, when the basic design strategy has been decided. Their main advantages are that users can manipulate them online, cover more tasks/functions and look and feel more like the final product. However, they are more expensive and time-consuming to build, as they require knowledge of the prototyping tool.

Based on the above, online prototyping was preferred as the prototypes would be used for evaluation with children, which usually pay particular attention to appearance. Initially, a set of non-interactive prototypes was created, simulating the aspects of the PUPIL system that were thoroughly described in the scenarios. These prototypes were used as a means to stimulate discussions and fine-tune design and usability issues. However, in order to facilitate user testing, these prototypes were converted to non-functional interactive Flash applications, that could be easily shown to the children.

The following figures present prototypes simulating an exercise and a dictionary application, as they would be displayed on the various artifacts included in the classroom.

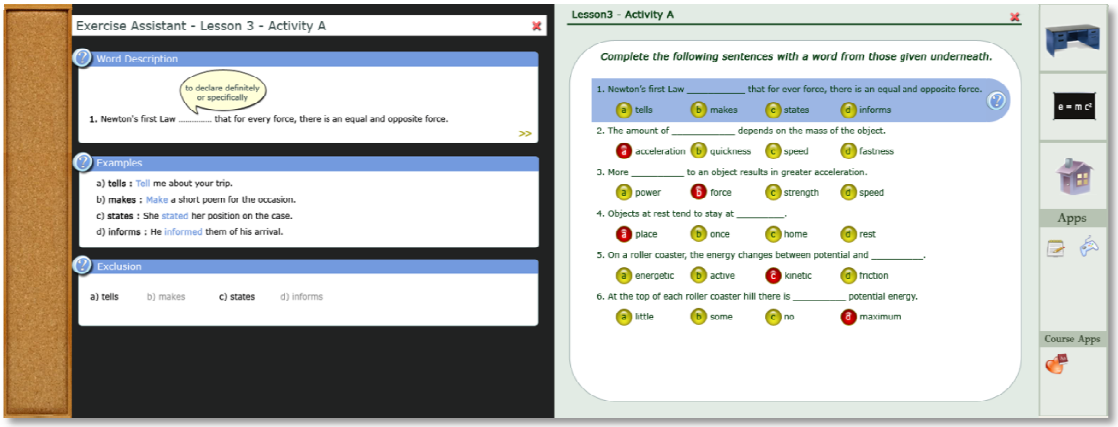


Figure 4: The AmIDesk Window Manager presenting a Multiple-Choice Exercise and relevant hints

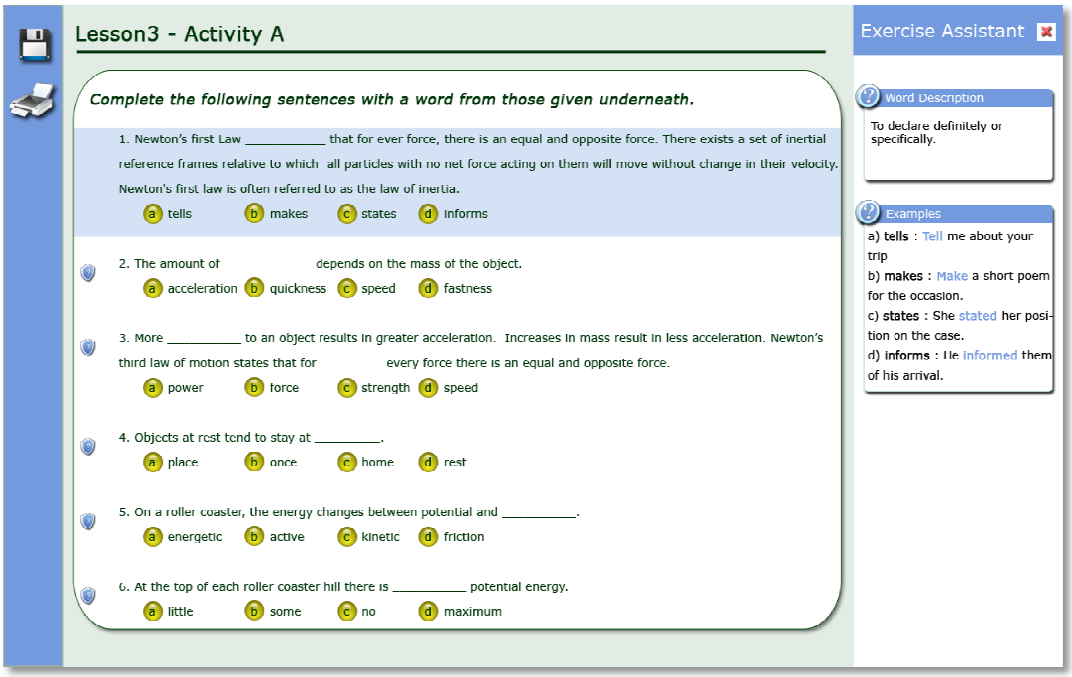


Figure 5: The Netbook Window Manager presenting a Multiple-Choice Exercise and relevant hints

Figure 6: A Dictionary application displayed on the AmIDesk artifact

Figure 7: A Dictionary application displayed on the SmartBoard artifact

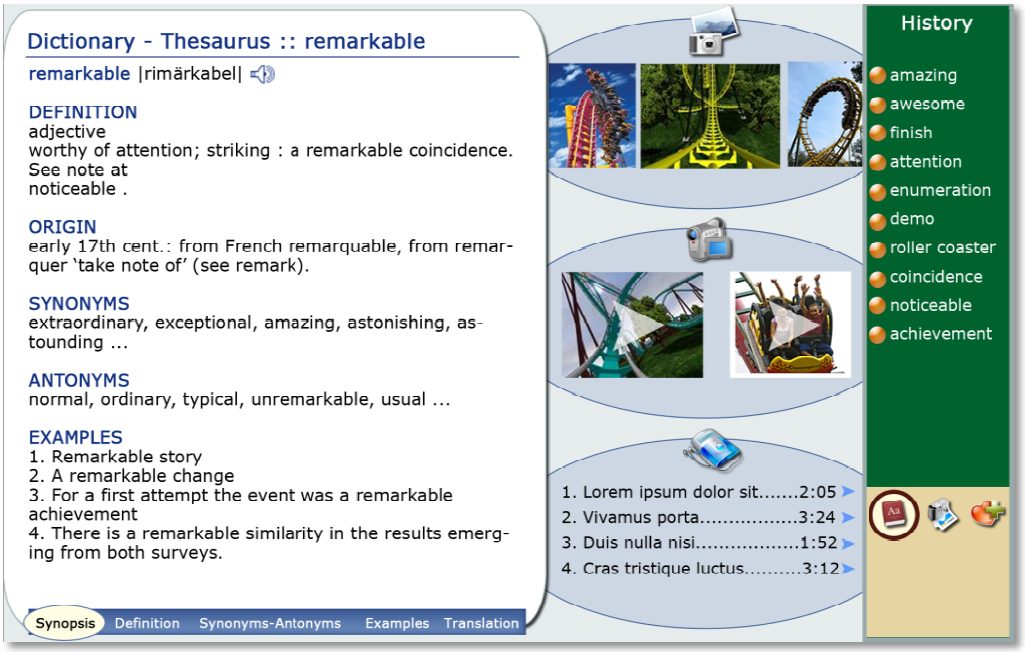


Figure 8: A Dictionary application displayed on the AmIBoard artifact

These high-fidelity prototypes not only define the basic layout of each application, but also integrate details about the preferred colors, fonts-styles and controls. An evaluation based on this kind of prototypes would provide useful feedback on numerous aspects of the applications, allowing the final product to better meet the end users' needs.

3.1.3.2 Formative Evaluation

As soon as the prototyping process was completed, a formative evaluation experiment was conducted in order to collect students' opinions about the functionality and applications supported by the AmIDesk artifact. Consequently, the initial scenarios (Appendix A - Scenarios used in Interviews) were modified to include activities performed only on the student's desk; this new set of scenarios (Appendix

C - Scenarios used in Formative Evaluation) was augmented with tasks concerning the MyVocabulary and Hangman applications, as indicated by the interview's findings. Details on Personal Area features, as well as exercise submission, were also included.

Having the informed consent of their parents, five young students, of ages between 11 and 16, were invited to participate in the evaluation process. All the students were familiar with computer systems and touch interfaces, as 3 out of 5 owned a mobile phone with touch capabilities.

Prior to the evaluation experiment, the lab was cleared from unnecessary items, in order to avoid distraction, while some appealing posters where added trying to make the environment child-friendly [19]. The idea of video-recording the process in order to observe the students was discarded, as the children tend to "freeze" or "perform" when they see a camera in the room [15]. The preferred alternative was a set-up where two note-takers observed and noted down the children reactions and comments, while an interactor initiated the discussion, asked questions about the activity and encouraged the students when necessary.

The evaluation process took place individually for each student and was scheduled to last 45 to 50 minutes, as children of these ages become tired after an hour of intense computer use [19]. At the beginning of each session, the interactor reassured the children that the evaluation process was about testing the software and not their knowledge or capabilities, and made a brief introduction about the system under evaluation. Then the students followed a predefined simple scenario, guided by the interactor, while during that process a series of questions were asked in order for the children to express their opinions about various aspects of the system. Finally, questionnaires (Appendix D - Questionnaires used in Formative Evaluation), formulated in an informal style to appeal to children, were handed to the students. A total of 12 questions used a Likert scale from 1 (sure!) to 5 (no way!) while two questions concentrated on what the children liked or disliked the most about the system. Once children filled-in the questionnaire, they were debriefed and asked for any additional comments, suggestions, as well as their overall impressions.

Next, both the questionnaires and the notes taken were analysed and further studied. As Figure 9 indicates, answers were classified into categories, and for each category the average and the standard deviation were calculated. Lower scores in the chart correspond to

positive and higher scores to negative answers, as the Likert scale ranged from 1 (best feature rate) to 5 (worst feature rate).

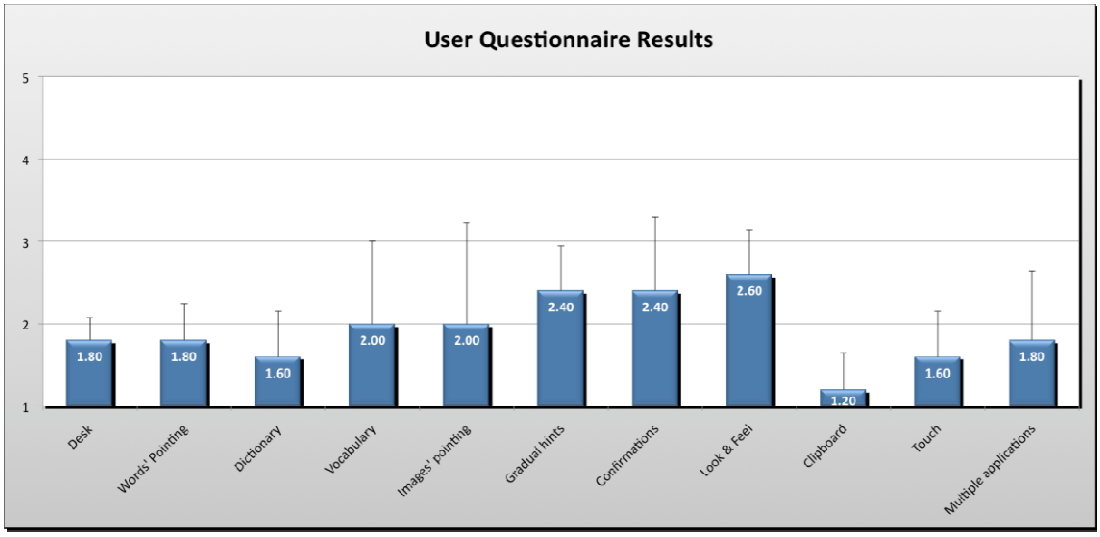


Figure 9: User questionnaire results

The results of this formative evaluation were encouraging, as the students' reactions and comments were mostly positive. They appreciated the educational support that the PUPIL system aims to provide, as well as the potential for better organization of work between the classroom and the home environment and collaboration with teachers and other students. Furthermore, it should be mentioned that although some of the participants are not technology enthusiasts, they all suggested that an augmented desk would definitely improve learning and that they would like to have the provided facilities available from home as well.

The preferred applications were the Personal Area, followed by the Dictionary, the educational games and the Exercise Assistant. The AmIDesk specific features that the children enjoyed most were the Clipboard and touch interaction. Furthermore, the idea of electronically submitting an assignment was quite appealing too. The dislikes of the children concerned the fonts and colors of the applications; some of them even proposed their favorite colors, raising the issue of personalization in terms of appearance. Figure 10 and Figure 11 indicate the features that were most and less favored respectively, according to the participants' answers in the last two questions. For example, the feature of the personal area was mentioned by four out of six students, and therefore it was chosen by 66,67% of the participants. It is interesting to mention that very few things were highlighted in the last questions as features that were less liked.

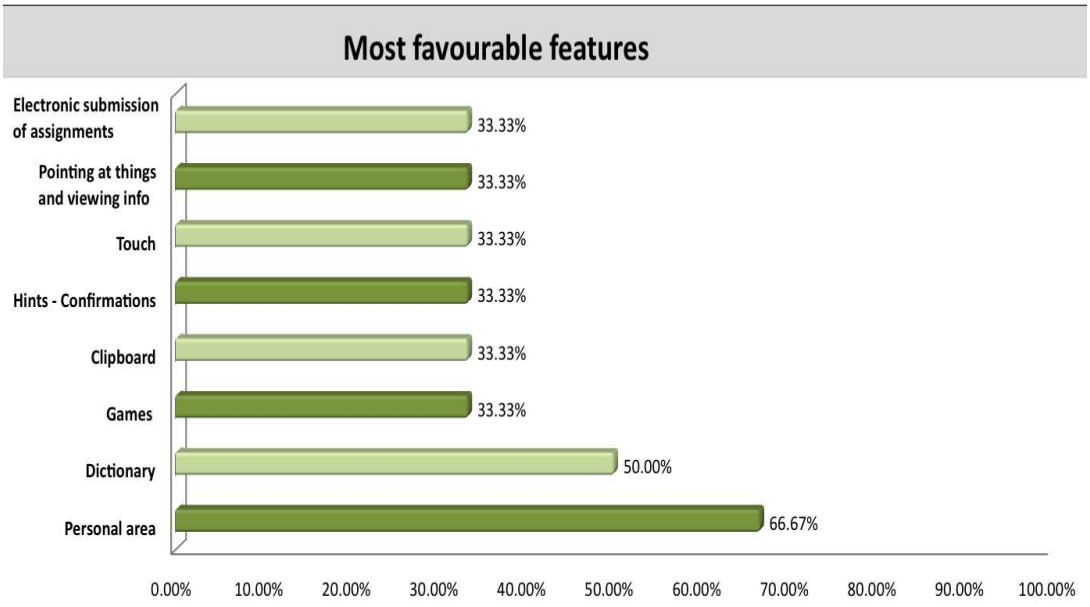


Figure 10: Most favourable features as indicated by the formative evaluation

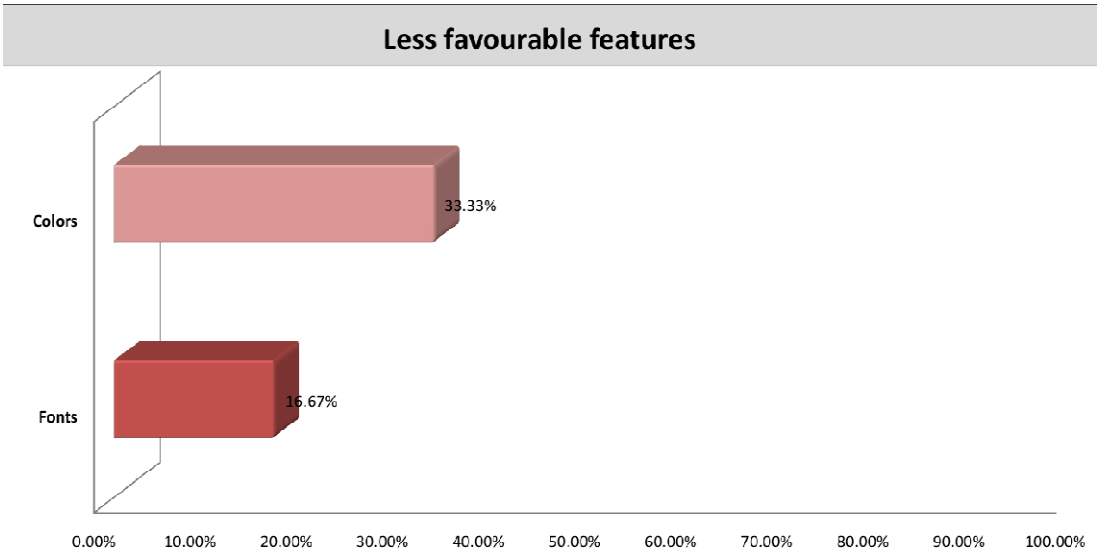


Figure 11: Less favourable features as indicated by the formative evaluation

Additional students' suggestions regarding the functionality of the applications are summarized below:

- A recycle bin at the bottom of the Clipboard would be useful in order to remove all the items it contains.
- The importance of teacher's control over the Exercise Assistant (Hint application) was raised by most of the students. They suggested that the teacher should be able

to activate and de-active the application, and view a real-time report of the hints requested by each student individually.

- The importance of educational and entertainment games was pointed out, however it was also mentioned that teachers should be aware of the games played by each student.
- A request for a grammar application displaying grammar rules, verb tables, etc., was expressed by two of the students, as it would be a valuable tool for language (foreign or native) learning courses.

Through the requirements elicitation process the needs of the students became clearer. The characteristics of children as users of interactive technologies for learning were not revealed only through literature review, but also through personal contact with real users during the interviews and the formative evaluation. The requirements elicitation findings contributed into the emergence of several UI design guidelines (3.2) that were followed throughout the PUPIL system development and that can be used by designers of educational applications. Furthermore, the application suite that was identified through scenarios, interviews and prototypes was the starting point for discovering common interface patterns that could be incarnated into compound widgets for designers to exploit. Finally, useful feedback was received on how the classroom artifacts should manage the applications (e.g., the desk artifact allows the student to work in parallel with to applications, one next to the other, the board artifact displays only one application to avoid distraction, etc.) in order to facilitate studying and active participation during the lectures.

3.2 UI Design Guidelines for the AmI classroom

This section presents the design guidelines elaborated on the basis of the results of the process described in section 3.1. Besides having being integrated in the PUPIL system, these guidelines can be of value to other designers addressing pupils' needs in the technological classroom, especially within the paradigm of Ambient Intelligence environments.

G01. ***Provide feedback to indicate successful interaction with a UI element.*** The user must be notified that his action triggered some operation and that a result will be visible shortly, independently of the operation's outcome (success or failure).

G02. ***On mouse-enabled artifacts controls must be always visible.*** On touch-enabled artifacts some actions are triggered indirectly by gestures. As a result, the related controls may be hidden and become visible on demand. On the contrary, on mouse-enabled artifacts the same controls must always be visible, as there is no other way to

perform these actions. For example, on the AmIDesk artifact the zoom operation is performed by the “zoom in” and “zoom out” (pinch) gestures, while on the Netbook artifact the existence of “zoom in” and “zoom out” buttons is essential to trigger those actions.

G03. ***On the AmIBoard and SmartBoard artifacts the interactive controls must be placed lower to the screen.*** The interactive controls of an application, when the last is launched on a board artifact, must be reachable by all students regardless of their height. For example, a horizontal menu that exists on the top of the content must be relocated at the bottom.

G04. ***Use gestures that imitate real life behavior.*** The users memorize easily a gesture that resembles an everyday interaction like panning on a photo to move it, thus the use of well-established gestures is encouraged.

G05. ***Consistent gesture behavior must be maintained.*** When a specific gesture is used by two PUPIL applications, they should respond in a similar manner.

G06. ***Properly spaced input elements.*** In order to eliminate erroneous interaction on the touch-enabled artifacts, the input elements must be properly spaced.

G07. ***Buttons should declare their state.*** It should be clearly visible when a button is idle, inactive, selected and about to perform an action in order to provide adequate feedback to the user actions.

G08. ***Appropriate button size.*** A button displayed on a touch-enabled artifact must be sufficiently large to be easily pointed and selected by a user’s forefinger. Literature review [45] has shown that a button‘s size on a touch screen must be equal to or bigger than 35*35 pixels.

G09. ***Appropriate font sizing.*** When displayed at long-distance artifacts, textual elements must have adequately large font sizes in order to be clearly visible and legible.

G10. ***Appropriate Image/Icon sizing.*** When displayed at long-distance artifacts, visual elements must be adequately large in order to be clearly visible.

G11. ***Take advantage of each artifact’s spatial characteristics when launching additional information.*** Use the available space to facilitate the exploitation of the additional information by placing it near the related content, without distracting the user.

G12. ***Take advantage of each artifact's spatial characteristics when displaying expandable content boxes.*** In artifacts that accommodate displays with high-resolution the content boxes should be expanded taking advantage of the free space. On the contrary, in smaller resolutions they should remain collapsed, leaving the user to decide whether to expand them or not.

4 The PUPIL System

The PUPIL system aims to support the artifacts listed in Table 1, purposed for the technologically enhanced classroom, by facilitating the development of auto-transforming educational applications and providing the appropriate software modules to host them. The application development process is simplified, thanks to a collection of UI components, which automatically transform to achieve optimal display for the current artifact. Using that collection, the designer's work is minimized, since the creation of a single design is sufficient to generate the appropriate alternatives for each artifact at runtime. On top of that, an appropriate working environment for each artifact is offered by the PUPIL system through five individual Window Managers (one for each artifact).

Artifact	Characteristics	Resolution	Screen Diagonal	Intended Use
AmIDesk	Vision-based touch-enabled device	1600x600	27''	student's desk
SmartDesk	Touch screen device	1440x900	19''	student's desk
AmIBoard	Vision-based touch-enabled device	1920x1200	81''	classroom board
SmartBoard	Touch sensitive interactive whiteboard	1024x768	77''	classroom board
Netbook	Common Netbook	1024x600	10.1''	both classroom and home

Table 1: Artifacts supported by the PUPIL system

In the following chapters the PUPIL UI Collection and the offered Window Managers are described in details.

4.1 PUPIL Widgets

4.1.1 PUPIL's UI Collection

Pervasive UIs are the cornerstone of the PUPIL system user experience. The PUPIL Widgets Collection, a collection of "pervasive" widgets, realizes such UIs. The collection's key feature is that every contained widget can transform itself, for optimal display, according to the artifact's characteristics. For that to be achieved, every PUPIL Widget encapsulates a set of UI alternatives and indirectly exposes them at design time through a single component, the Designer Control.

The PUPIL Widgets are divided into two categories: (i) simple widgets and (ii) complex widgets. The simple widgets extend native WPF UI Elements (WPF Textblock, WPF Image, etc.) and incorporate the transformation mechanism. The complex widgets, on the other hand, are the realization of common interface patterns recognized among numerous learning applications (Image Displayer, Multiple Choice Question & Answers, etc.) into ready-to-use widgets. The complex widgets' design process included brainstorming sessions, prototypes and heuristic evaluation, so optimal display and usability is ensured when traveling among artifacts. Therefore, using these widgets the designer has at his disposal a set of usable user interfaces based on validated design decisions. Finally, the developers are also benefited by the use of complex widgets since they are fully functional building blocks that encapsulate their business logic.

The Designer Control is a WPF UserControl that exposes a number of dependency properties, allowing the designer to customize several display attributes for that particular widget. Nevertheless, the designer has restricted access to some attributes to ensure widget's usability. For instance, the foreground color of a complex radio button (interactive control and label) might be altered, but the label must be always placed on the right hand side to facilitate the scanning process.

Following the Factory Method design pattern [17], each Designer Control implements the createProduct method of icsWidgetFactoryIface interface. That method selects and creates the appropriate PUPIL Widget alternative, according to the current context of use (AmIDesk, Netbook, SmartBoard, etc.).

The Factory Method pattern is an object-oriented design pattern to implement the concept of factories. Like other creational patterns, it deals with the problem of creating objects (products) without specifying the exact class of object that will be created. The factory method design pattern handles this problem by defining a separate method for creating the objects, which subclasses can then override to specify the derived type of product that will be created.

Finally, each PUPIL Widget is responsible for transforming itself (through the createProduct method) in order to better fit the available space. Nevertheless, a designer might decide to place two or more widgets in a container that forces them to be displayed in the same line. A native WPF widget would just show itself as is, retaining the risk of exceeding the screen boundaries. On the contrary, a PUPIL Widget can be scaled down, to a permitted limit, in

order to compose a proper layout. The scale method is defined by the Scalable interface, which is implemented by every PUPIL Widget alternative.

Occasionally a student's choice might reduce the available space of an application (i.e. the "application history" Sidebar appears). At that point some of the PUPIL Widgets might be scaled down. The Reset method of the Scalable interface allows the widgets to obtain their original size, as soon as the available space is restored.

4.1.2 Sizing Methodology

In the augmented classroom environment where an application might migrate from the desk to the board artifact, it is really important to ensure that upon migration, all the application UI elements are clearly visible and legible from the entire class. Taking into consideration that designers build their application interfaces having in mind an abstract display, a methodology was introduced and adopted throughout the widget library development process to address the sizing issue.

This methodology included empirical experiments, where experts evaluated various widgets in various classroom artifacts, in order to determine the minimum sizes of each widget when displayed on every artifact. These minimum values are employed through Formula B, by the developed widgets at runtime, in combination with designers' size-related choices, to proportionally scale them for optimal display.

Formula A

MinS = MIN (AmIDesk MS, SmartDesk MS, AmIBoard MS, SmartBoard MS, Netbook MS)

Formula B

WidgetSize = RoundUp((AMS * DS) / MinS)

MS: minimum size
DS: widget size as specified by the designer
AMS: current artifacts minimum size

The aforementioned formulas are used by several widgets, and concrete examples will be presented in the following sections.

4.1.3 Simple Widgets

4.1.3.1 PUPIL Textblock

The PUPIL Textblock widget ensures that every textual element is appropriately translated when displayed in a PUPIL artifact. The visibility and legibility of a Textblock could be easily compromised by improper font sizes when traveling among artifacts with various screen resolutions and different physical places (i.e., student's desk, classroom board) in the classroom. For instance, the students sitting in the back of the class would face difficulties trying to read a 12point text displayed in the AmIBoard. To address this issue, the PUPIL Textblock ensures that its contents' font size will never be less than the minimum font size of the current artifact as depicted in Table 2.

Artifact	Min Font Size
Netbook	12pt
SmartDesk	12pt
AmIDesk	16pt
SmartBoard	20pt
AmIBoard	30pt

Table 2: Minimum font sizes for the supported classroom artifacts

Consider a designer declaring a 14point PUPIL Textblock; the font size of this widget will remain the same at the Netbook and SmartDesk artifacts as it complies with the minimum size specification. On the contrary, when displayed on the AmIDesk the font size value will increase to 19 as calculated through Formula B. The above process ensures that every font size will be proportionally adjusted to each specific artifact's needs.

Another feature of the PUPIL Textblock widget is that it automatically wraps when the available space is limited, liberating the designer from dealing with space limitations. Subsequently, when displayed at a larger screen, the textual element can take advantage of the horizontal space, while when moving to a smaller one, it is assured that it will never be out of bounds, as it will expand vertically.

4.1.3.2 PUPIL Button, PUPIL ToggleButton and PUPIL ImageButton

The PUPIL Button widget contains the logic that controls the dimensions of the displayed button. When a touch-enabled artifact launches an application, the size of the contained buttons must be adequately large in order to avoid inaccurate user inputs. Consequently, all buttons that do not conform to the size requirements are scaled proportionally. The role of this transformation is twofold, as on the one hand it ensures optimal performance, and on the other hand it increases the student's confidence that his actions will have the desired

results. The PUPIL Button transformation conforms to the G8 guideline and takes into consideration the proposed minimum dimensions. The PUPIL ToggleButton and PUPIL ImageButton widgets behave similarly to the PUPIL Button when displayed on a touch-enabled artifact.

The PUPIL Button and PUPIL ToggleButton share another common characteristic. Using the same methodology as the PUPIL Textblock, they are able to adjust their font size, ensuring optimal visibility.

4.1.3.3 PUPIL Image and PUPIL Icon

The PUPIL Image widget is used by applications that intend to transmit knowledge through visual information. For example, the Hint application might use a PUPIL Image in order to describe the correct answer of a multiple-choice sentence. Consequently, this widget needs to be adequately large in order to ensure its visibility, especially when displayed on the AmIBoard and SmartBoard artifacts. So the alternatives created for the PUPIL Image ensure that their dimensions will at least cover the minimum threshold as presented in Table 3.

Artifact	Min Width/Height Size
Netbook	100px
SmartDesk	100px
AmIDesk	100px
SmartBoard	250px
AmIBoard	250px

Table 3: Minimum image width and height values for the supported classroom artifacts

The PUPIL Icon widget, on the other hand, is used to display explanatory information. For instance, consider a “tick” icon next to a multiple-choice sentence. This shows that the sentence was filled by the correct answer. The “tick” icon does not need to be thoroughly examined while it conveys its meaning through its pictorial resemblance to a simple physical object or a commonly used mark. Therefore the acceptable dimensions of a PUPIL Icon are allowed to be much lesser than those suggested for the PUPIL Image (Table 4).

Artifact	Min Width/Height Size
Netbook	25px
SmartDesk	25px
AmIDesk	25px
SmartBoard	70px
AmIBoard	70px

Table 4: Minimum icon width and height values for the supported classroom artifacts

4.1.3.4 PUPIL ScrollViewer

The PUPIL ScrollViewer widget enables content to be displayed in a smaller area than its actual size. When the content of the PUPIL ScrollViewer is not entirely visible, scrollbars are displayed that allow the user to scroll (up/down, left/right) in order to view the entire content. A typical scrollbar consists of a bar (or thumb) that can be dragged along a trough (or track) as well as two arrows on either end; however the PUPIL ScrollViewer introduces three alternatives that contain appropriately modified scrollbars.

The first alternative used for the Netbook artifact, adopts the default style of scrollbars. The second alternative that is displayed on the SmartBoard artifact, modifies the vertical scrollbars so as to contain both arrows at the bottom of the track. This ensures that younger children will be able to scroll up. Finally, the third alternative displayed on every touch-enabled artifact (i.e., AmIDesk, SmartDesk and AmIBoard), contains scrollbars with no buttons. The scrollbars are really thin in order to allow more space for the content, however, as soon as a user start scrolling, the width of the scrollbar increases to ensure that the user's finger will not miss the track.

4.1.3.5 PUPIL Slider

A Slider is a control that allows the users to select a value from a range of values. The default slider depicted in Figure 12, contains a thumb that the user can drag to the desired value. However, this might be problematic in the touch-enabled artifacts, as the user's finger will never be precise enough to drag the thumb over a specific point of the track. For that reason the PUPIL Slider, generates an alternative (Figure 13) for the AmIDesk, SmartDesk and AmIBoard artifacts that includes two buttons to move the slider thumb to the next/previous step.

Figure 12: The PUPIL Slider as displayed on the Netbook artifact

Figure 13: The PUPIL Slider as displayed on the AmIDesk, SmartDesk and AmIBoard artifacts

4.1.4 Complex Widgets

4.1.4.1 PUPIL Menu

The PUPIL Menu consists of menu items whose actions are defined at runtime, enabling intra-application navigation. The PUPIL mechanisms are responsible for the automatic generation of the application menu (LayoutBase), however the designer can customize its appearance through an external XML file. The visual attributes that can be parameterized and their valid value ranges are described in the following tables.

Menu Item Attributes	Examples of Valid Values
Idle, Focused and Pressed Foreground	Color (White, Black, #FFFFFF, etc.) or Image
Idle, Focused and Pressed Background	Color (White, Black, #FFFFFF, etc.) or Image
Idle, Focused and Pressed Font Size	12, 14, 18, etc.
Idle, Focused and Pressed Font Weight	Normal, Bold
Idle, Focused and Pressed Font Family	Arial, Verdana, etc.
Margin	5, 10, 15, etc.

Table 5: Menu item attributes that the designer can specify through an external XML file

Menu Attributes	Examples of Valid Values
Background Color	White, Black, #FFFFFF, etc.
Orientation	Vertical, Horizontal
Menu Item Style	Text, Icon, Icon before Text, Icon after Text

Table 6: Menu attributes that the designer can specify through an external XML file

These attributes allow composing a fully customized menu, but the PUPIL system ensures optimal display by applying runtime transformation for the contained simple widgets. For instance, the designer can set the font size of a menu item, and the system will follow a validation process (similar to the one described for the PUPIL Textblock widget) to assure that the menu will be clearly visible regardless of the artifact's characteristics, thus the designer-defined font size will increase to achieve optimal display in AmIBoard's large screen.

4.1.4.2 PUPIL Tabs and PUPIL ExpandableTabs

The PUPIL system offers two tab widgets: (i) the PUPIL Tabs and (ii) the PUPIL ExpandableTabs. Their differences concern only visual characteristics, as the second widget was introduced to address space limitations. The PUPIL Tabs widget when displayed resembles a common tab menu, while PUPIL ExpandableTabs widget (Figure 14) looks like a common button that toggles the visibility of a contained tab item list.

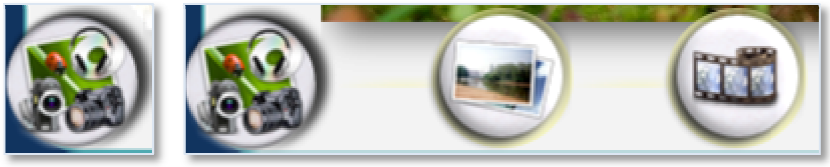


Figure 14: The PUPIL ExpandableTabs as displayed on the SmatDesk artifact

Similar to the menu, the tabs of an application and their actions are generated at runtime by the PUPIL system. When composing an application, the designer can define the type of the generated tabs, as well as their position. For instance, an application could contain either a PUPIL Tabs widget placed vertically next to the content, or PUPIL ExpandableTabs placed at the bottom left corner of the application in order to provide more horizontal space for the content. As in the case of the PUPIL Menu, an XML file is used for the visual customization of these widgets. Through this file the designer can modify the attributes presented in the following tables.

Tab Item Attributes	Examples of Valid Values
Idle, Focused and Pressed Background	Color (White, Black, #FFFFFF, etc.) or Image
Margin	5, 10, 15, etc.

Table 7: Tab item attributes that the designer can specify through an external XML file

Tab Attributes	Examples of Valid Values
Orientation	Vertical, Horizontal

Table 8: Tab attributes that the designer can specify through an external XML file

4.1.4.3 PUPIL Paging

A paging mechanism is essential to the majority of the classroom applications. Therefore, the PUPIL system introduces the PUPIL Paging widget. The alternatives generated by this widget ensure usability and optimal performance of paging.

The Netbook and SmartBoard alternatives incorporate a pair of previous-next buttons and a drop down menu for page selection. This combination allows students to either browse

through the pages one by one, using the previous/next buttons, or to jump from one page to another, using the drop down menu.

Figure 15: The Paging alternative for the Netbook and SmartBoard artifacts

When the number of pages is limited (10 pages or less) the generated alternatives for the touch-enabled artifacts (AmIDesk, SmartDesk and AmIBoard) support scrollable pages. The user (teacher or student) can slide a finger over the visible page, following a bottom-up/top-down direction to scroll down/up respectively. The number of pages is indicated by a set of interactive rectangles. When a rectangle is selected the pages start scrolling, and stops when the respective page is revealed. The currently visible page is represented by a solid rectangle.

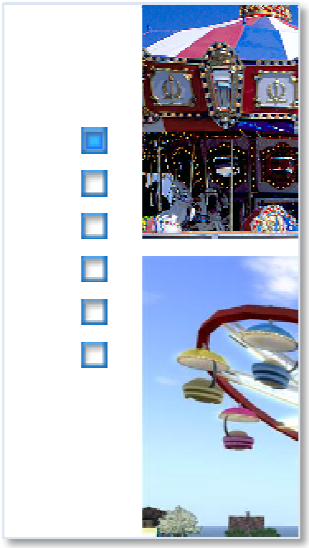


Figure 16: Paging alternative for touch-enabled artifacts when available pages are less than 10

In case the number of pages exceeds the 10-page threshold, the PUPIL Paging alternatives display a pair of previous-next buttons and a set of buttons labeled by each page's number. Through the page buttons the student can jump directly to the desired page. If the page buttons do not fit in the available space, then a Google-like approach is adopted, where only the appropriate subset is displayed.

Figure 17: Paging alternative for touch-enabled artifacts when available pages are more than 10

4.1.4.4 PUPIL ZoomableImage

The PUPIL ZoomableImage widget extends the PUPIL Image by adding zoom and pan functionality. Two alternatives exist: the touch-enabled and the mouse-enabled PUPIL ZoomableImage. The first one targets touch-enabled artifacts (i.e., AmIDesk, SmartDesk, AmIBoard) and offers intuitive interaction (zoom-in and pan gestures) taking advantage of their touch capabilities, while the latter targets mouse-enabled artifacts (i.e., SmartBoard, Netbook) and provides a set of explicit controls (a slider and a pair of zoom-in/out buttons) to facilitate interaction.

The **zoom-in** gesture is performed when two fingers slightly separated are placed on the screen and then start to slide further apart, while the **zoom-out** gesture is initiated when the fingers slide closer together.

The **pan** gesture is performed when a finger is placed on the screen and starts to move horizontally, vertically, or both.

The **mouse-directed panning** occurs when a mouse click is performed on the image followed by a mouse move (i.e. horizontal, vertical or diagonal).

4.1.4.5 PUPIL Video

This PUPIL Video widget is used by applications that require video playback and integrates a set of controls responsible for media playback (play, stop, pause), volume and position. These controls will be appropriately transformed taking into consideration the current artifact features. For example, the slider that controls media volume must have a sufficiently wide track when displayed on a touch-enable artifact (i.e., AmIDesk). This ensures that the user's finger will not miss the track while trying to increase/decrease the volume. Finally, to ensure usability, the PUPIL Video widget defines its minimum size appropriately for each artifact.

4.1.4.6 PUPIL ImageViewer

Pictures and illustrations generally enhance learners' performance [9], so an ImageViewer widget was introduced for use by numerous applications. For instance, consider a Dictionary application for the English course. Besides describing a word, providing synonyms and giving

examples of use, the Dictionary could be enhanced by an ImageViewer in order to facilitate the student's better understanding of the presented word. After all, "a picture is worth a thousand words".

The ImageViewer widget, offered by the PUPIL UI collection, supports the presentation of numerous pictures, which the student can browse, maximize, zoom or pan. These actions are supported through variant controls according to each artifact's characteristics. The Designer Control of this widget allows the designer to define the percentage of the horizontal and vertical space which the displayed alternative will occupy, as well as the size of its thumbnails. These parameters will be appropriately interpreted by the respective alternative. Consider the following example, if the thumbnail width and height set by the designer are too small for the AmIBoard artifact, the clear view of students sitting at the back of the classroom might be compromised, so the generated alternative is responsible to ensure that the thumbnail size is appropriate for long distance viewing.

The AmIDesk and SmartDesk alternatives of the ImageViewer widget are the same. Taking advantage of the touch interaction supported by these artifacts, the most interesting approach of browsing through the pictures was sliding. The student can slide his finger, from right to left and vice versa, over the pictures in order to see the next or previous one. When selected, a picture can be maximized (PUPIL ZoomableImage) and then by using the common zoom in and zoom out gestures the student can examine it thoroughly. Finally, the maximized picture can be closed, in order to continue browsing. The interactive control that performs that action is large enough for a student's finger not miss selecting it.

Figure 18: ImageViewer thumbnails and maximized picture as displayed on the AmIDesk

The AmIBoard alternative differentiates from the alternatives developed for the desk artifacts, as it addresses to the entire classroom and not a single student. Thus, it would be of outmost importance to display more and larger picture thumbnails, enabling all the

students to study an adequate number of pictures at once. Furthermore, a paging (PUPIL Paging) mechanism is offered to support a large number of thumbnails.

The maximize process is identical to the one supported in the desk alternatives. However, counter to the "close" interactive control used on the AmIDesk and SmartDesk artifacts, a gesture is chosen to close the maximized picture. To accomplish that action a user must slide his finger over a designated area, following a right to left direction. In order to be easily accessible by younger students, that area is placed at the bottom of the ImageViewer widget.

The SmartBoard and Netbook alternatives have the same characteristics. They both display a grid of picture thumbnails; while their only difference is that the first requires a larger minimum thumbnail width and height than the second one. The contained PUPIL ZoomableImage and PUPIL Paging widgets automatically enable their alternatives to handle appropriately the zoom, pan and paging operations.

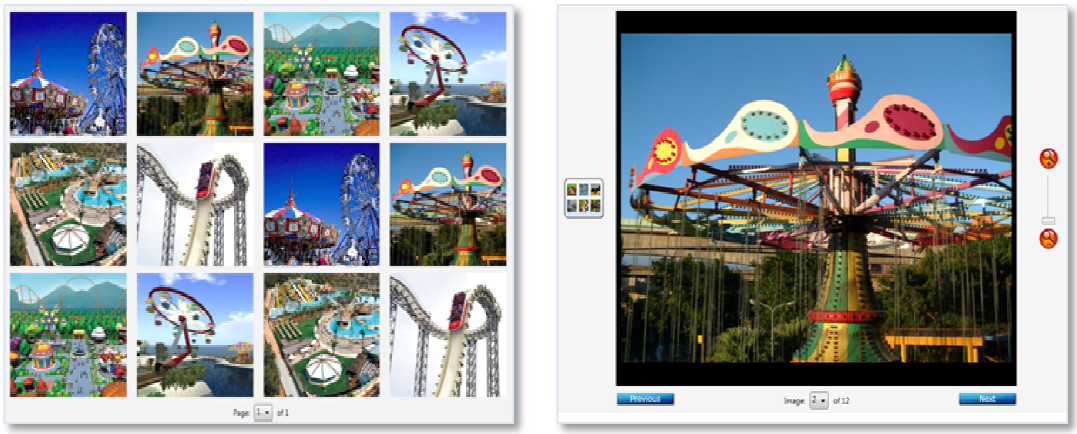


Figure 19: ImageViewer displaying grid of thumbnails and maximized picture on the SmartBoard

4.1.4.7 PUPIL VideoViewer

Another interesting widget offered by the PUPIL system, providing access to a collection of video elements, is the PUPIL VideoViewer. It holds a set of PUPIL Video widgets available for playback on demand. Consequently, a student can browse a number of video thumbnails and select a specific one to watch, although the browsing process differentiates among the various artifacts.

When displayed on the desk artifacts (Figure 20), the composed elements are a placeholder for PUPIL Video widgets and a vertical list of video thumbnails. The student can slide a finger up or down that list in order to browse all the available videos and by selecting a thumbnail

a PUPIL Video widget is launched playing the respective video. On the contrary, when displayed on the board and Netbook artifacts, the PUPIL Video Viewer offers a grid of video thumbnails from which the student can select one to view. The grid representation is used for the board artifacts, as it offers a larger number of visible thumbnails and allows all the students to look at more pictures at once. The same representation is more suitable for the Netbook artifact too, enabling the user to browse easily through the video thumbnails using the mouse.

Similarly to the PUPIL ImageViewer, the PUPIL VideoViewer allows the definition of thumbnail dimensions as well as the desired space percentage that the widget will occupy.

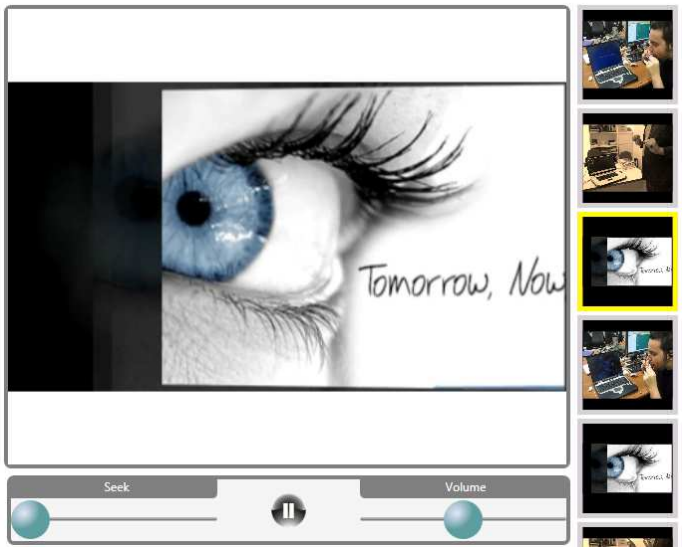


Figure 20: The VideoViewer as displayed on the desk artifacts

4.1.4.8 PUPIL MultipleChoiceElement

This widget was created to assist the development of multiple-choice exercises, which is a very frequent type of exercise. The PUPIL MultipleChoiceElement consists of a PUPIL Textblock, which displays the sentence that needs to be completed, and a number of PUPIL ToggleButtons accompanied by PUPILTextblocks, which compose the set of possible answers. The scalability of this widget is based on the fact that the number of possible answers is not predefined, but it is dynamically determined.

Another feature of the PUPIL MultipleChoiceElement is that it contains a “hint” button responsible for displaying helpful information that could facilitate the student completing the respective sentence. Consequently, when the “hint” button is selected, the sentence is highlighted and the Hint application is launched.

When displayed on the various artifacts its components, the PUPIL Textblocks and PUPIL ToggleButtons included in the PUPIL MultipleChoiceElement translate themselves for optimal display, however this is not the only transformation enforced to this widget. On the majority of artifacts, the possible answers are displayed the one next to the other. However, when a multiple-choice exercise is launched at the AmIDesk, half the answers are displayed in the first row and the rest in the second one. This transformation is necessary due to the smaller horizontal space offered by the AmIDesk artifact for each application.

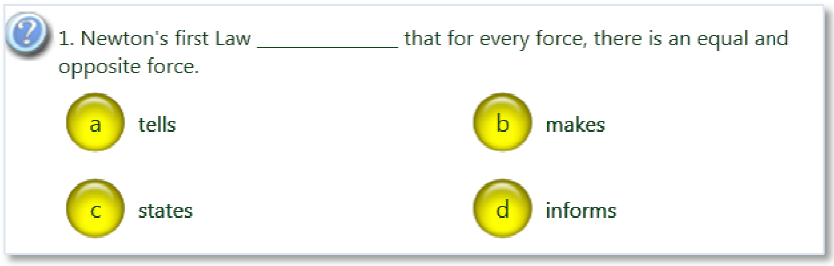


Figure 21: The MultipleChoiceElement as displayed on the AmIDesk artifact

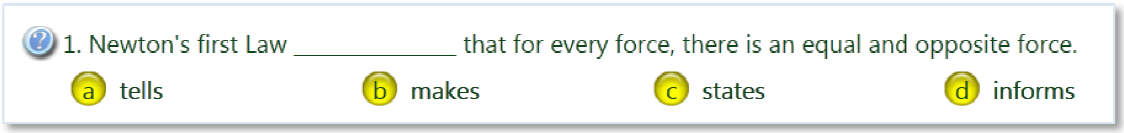


Figure 22: The MultipleChoiceElement as displayed on the SmartBoard artifact

4.2 Window Managers

The PUPIL system supports the following artifacts: AmIDesk, SmartDesk, AmIBoard, SmartBoard and Netbook. Each of these artifacts is equipped with a Window Manager in order to create an appropriate workspace for the students and facilitate the learning process.

A window manager is the system software that controls the placement and appearance of windows within a windowing system in a graphical user interface. Most window managers are designed to help provide a desktop environment. Each PUPIL Window Manager has unique characteristics, as it is purposed for a specific classroom artifact (i.e., student's desk, classroom board, etc.).

The following sections describe the skeleton of a typical PUPIL application, the mechanisms that facilitate the intra-application navigation, and finally the characteristics of the supported Window Managers.

4.2.1 Core Modules

4.2.1.1 Application Definition

The Extensible Application Markup Language (XAML) is used to build a PUPIL application. XAML is a declarative XML-based language created by Microsoft, which is used to initialize structured values and objects. XAML elements map directly to Common Language Runtime object instances, while XAML attributes map to Common Language Runtime properties and events on those objects. XAML files can be created and edited with visual design tools such as Microsoft Expression Blend and Microsoft Visual Studio, which makes the designer's work easier.

A generic application layout contains, apart from the main content, a list of tab items for switching among the application's main content categories and menu items, thus facilitating navigation through a category's subcategories. In order to conform to that specification, any application launched by the Window Managers should follow a specific structure consisting of three PUPIL-specific XAML elements: (i) PUPILApplication, (ii) PUPILApplet and (iii) PUPILAppletComponent. A designer uses the aforementioned elements to build the appropriate hierarchy and create the Application Tree.

PUPILApplication Element

The root element of the Application Tree hierarchy is the PUPILApplication Element, which defines a single application (e.g., the Multimedia application). A number of properties exposed by the PUPILApplication Element support its customization according to the designer's needs. These properties concern (i) the application Title (text, foreground and background color), (ii) the Tabs style (expandable, vertical, horizontal) and finally (iii) the XML file that describes the menu appearance as aforementioned in the PUPIL Menu section.

PUPILApplet Element

The PUPILApplication Element can contain more than one PUPILApplet children. This element describes the content categories of an application; an example is the Image Displayer and Video Displayer components of the Multimedia application. During the launching process of an application, the PUPILApplet is translated to a Tab item. However, if only one PUPILApplet exists, the Tab item remains hidden as it is redundant.

The customizable properties of the PUPILApplet Element are the Icon and Text of the generated Tab item plus the Background Color of the subcomponent (i.e., the Image Displayer might have a different background color than the Video Displayer).

PUPILAppletComponent Element

One or more PUPILAppletComponent Elements defined inside a PUPILApplet, describe the subcategories of that specific element. As an example consider the Image Displayer category of the Multimedia application and the subcategories Book Images and Web Images. The translation of the PUPILAppletComponents to menu items –during the launching process– ensures a straightforward navigation. The designer can configure the menu item icon, the selected menu item icon and the menu item text through the respective properties. As some artifacts may support text-only or icon-only menu items, it is necessary for the designer to fill in these values.

The designer is not limited in using only PUPIL Widgets, as the children of the PUPILAppletComponent Elements might be a combination of PUPIL Widgets simple or complex and WPF native UI elements.

The following example describes the Application Tree of a Multimedia application that can host images from two different sources and play videos. For that to be achieved, two PUPILApplets are defined to host the Image Displayer and Video Displayer modules respectively. Since a new tab item will be created for each one of these two modules, the designer has to populate the icon property of each PUPILApplet with representative icons.

Moving deeper in the hierarchy, two PUPILAppletComponents are defined under the PUPILApplet entry that signifies the Image Displayer. These PUPILAppletComponents consist of the “Book Images” and “Wikipedia Images” subcategories that will be populated with images from the two different sources. Similarly, a single PUPILAppletComponent is defined under the PUPILApplet entry that denotes the Video Displayer category; such PUPILAppletComponent component constitutes the single subcategory that will host the video player.

In addition to defining the application’s structure, the designer in order to facilitate the user’s navigation through the automatically generated menu, customizes the PUPILAppletComponent properties (text, icon) appropriately.

```
1   <PUPILApplication Title="Multimedia Application" TitleFgColor="White">
2       <PUPILApplet Text="Image Displayer" Icon="ImgVwr.png">
3           <PUPILAppletComponent Text="Book Images" Icon="bookImg.png">
4               //PUPIL Widgets and/or WPF UI elements
5           </PUPILAppletComponent >
6           <PUPILAppletComponent Text="Web Images" Icon="webImg.png">
7               // PUPIL Widgets and/or WPF UI elements
8           </PUPILAppletComponent>
9       </PUPILApplet>
10      <PUPILApplet Text="Video Displayer" Icon="VdoVwr.png">
11          <PUPILAppletComponent Text="Videos" Icon="videos.png">
12              // PUPIL Widgets and/or WPF UI elements
13          </PUPILAppletComponent>
14      </PUPILApplet>
15  </PUPILApplication>
```

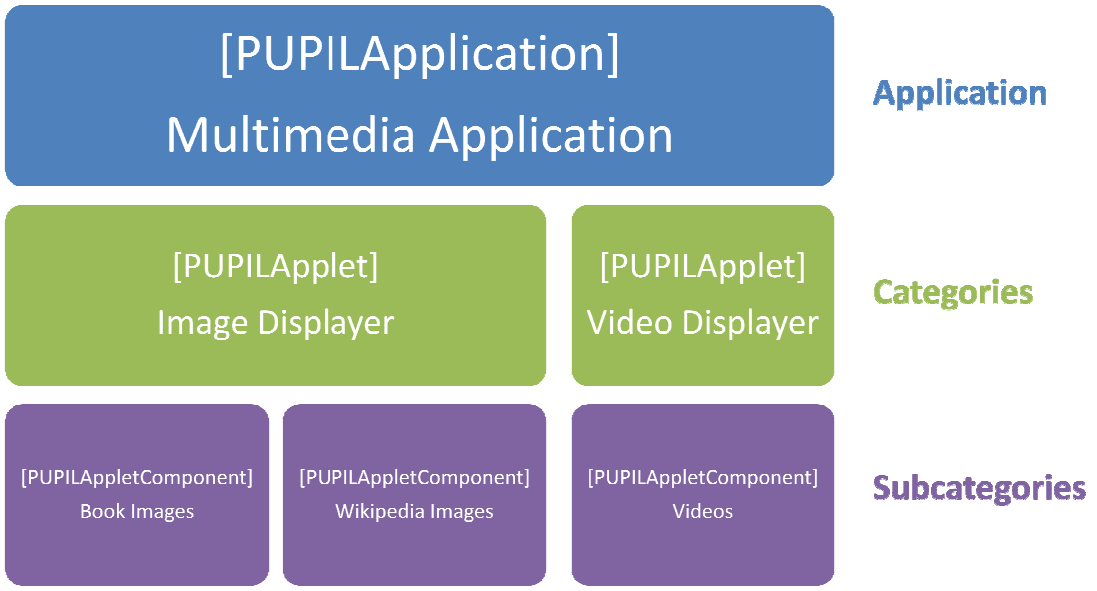


Figure 23: Graphical representation of the Application Tree of the Multimedia Application

Designer tools: group and breakLine

In order to take advantage of all available space that an application might exploit (depending on the artifact), the WPF WrapPanel container was selected as the default container to host all the widgets specified by the designer.

In more details, the WrapPanel is used to position child elements in sequential position from left to right, breaking content to the next line when it reaches the edge of its parent container. So according to this behavior, two widgets displayed next to each other in a big screen, would probably be rearranged in a smaller screen sending the second widget to the next line.

Nevertheless, a system that only supports UIs where all the contained UI elements wrap in the available space does not provide an optimal solution. The designers should be given more flexibility when building the UIs, without compromising application's usability and appealing appearance when traveling among artifacts. To address that issue (i) group and (ii) breakLine designer elements were introduced.

The group element enables the application designer to denote that all the contained widgets must always be placed at the same line. However, forcing the display of a number of widgets in a single line entails the risk of hiding some of them due to space limitations. In such case, since the designer's directive was single line occupancy, the strategy is to scale down the contained widgets to fit the designated area without hiding any of them.

On the contrary, the use of the breakLine element forces the following widget to be displayed at the next line.

4.2.1.2 LayoutBase

Any application layout can be customized at design time through the available options (i.e., menu orientation, tab orientation and style, etc.); however, device characteristics may lead to its modification at run time. For instance, consider a designer who creates an application having in mind the AmIDesk artifact. The designer selects the horizontal version of the menu and places it above the main content. The same application, if launched on the SmartBoard, will have a different layout. The menu will maintain its horizontal orientation but it will be placed below the main content to be reachable by younger students.

The variety of placement alternatives for every main UI component (menu, tabs, main content and application previews) results in more than one layouts for each artifact (AmIDesk, Netbook, SmartBoard, etc.). Nevertheless, the process of parsing any Application Tree and composing the layout is always the same. To encapsulate the common logic, the LayoutBase class was defined as the base class from which every layout will derive.

LayoutBase composes the application layout based on the Application Tree defined by the designer. The PUPILApplication is scanned first for any PUPILApplet elements that will

generate the application's tab items and menus (one menu for each tab item). Then, every PUPILApplet is scanned for PUPILAppletComponents that populate the respective menu. The actions of the generated tab and menu items are defined so that each tab item activates a menu and each menu item displays the appropriate content. At that point, since the controls (i.e., tabs and menus) are successfully generated, the intra-application navigation routes are complete. The main content population follows the same iterative process, and is accomplished by iterating and appropriately translating the children of a PUPILAppletComponent.

When a menu item is selected, the respective PUPILAppletComponent is parsed to display its content. Likewise, when a tab item is selected, the same process is followed for the first menu item of that tab. A local cache is used to store the UI elements of any previously selected menu item, optimizing the above process.

A PUPILAppletComponent may contain any combination of PUPIL Widgets and native WPF UI elements. To offer an optimal layout the following strategy is followed. Any retrieved element is placed in a WPF Wrap Panel to ensure that if it exceeds the space limits it will move to the next line. When encountering a breakLine element a new Wrap Panel is created below the previous one, to host the subsequent elements. However, whenever a group element is retrieved, after adding to the Wrap Panel, it must be checked whether its children can be displayed at the same line as is, or need to be scaled down.

In case that the available application space is reduced at runtime (a sidebar appears), all the PUPIL Widgets that do not fit the remaining space must be scaled down. On the contrary, any WPF UI elements that do not have the appropriate size are displayed as is compromising the layout.

To complete the layout, the created tabs, menus and content are positioned at the appropriate placeholders, and the background color, background image and application title are set according to the Application tree properties.

4.2.1.3 WindowManagerBase

WindowManagerBase is a base class from which any Window Manager derives. Its main objective is to provide a fundamental set of functions that each Window Manager may override. That set not only includes common application management method that all derived Windows Managers must implement (i.e., launch, terminate and focus), but also artifact specific methods, which are implemented by the respective artifacts. For instance,

the features of application slide or pin to the Clipboard, are only offered by the AmIDesk and the SmartDesk artifacts, thus only the relevant Window Managers should implement the respective methods.

4.2.1.4 Intra Application Navigation (Command Handler Pattern)

The Intra-Application navigation is a two-tier structure. The first layer consists of the application's applets encoded as tab items, while the second layer consists of the application's applet components encoded as menu items. An indicative example of that structure, as extracted from the Multimedia application, is the following. The Multimedia application's tabs items are "Image Displayer "and "Video Displayer". The menu items for the "Image Viewer" tab item could be the "Book Images" and "Wikipedia Images", while the "Video Displayer" could have a single menu item "Videos". As implied by the aforementioned example every application contains a single Tab list, though there are as many Menus as tab items. Each tab item activates its Menu and each menu item is responsible to display its content.

As already mentioned the tab and menu items are generated automatically during the application launching, so the actions of the navigation items are application specific and defined by the Application Tree structure.

When two objects communicate, often one object is sending a command to the other object to perform a particular function. The most common way to accomplish this is for the first object (the "issuer") to hold a reference to the second (the "recipient"). The issuer executes a specific method on the recipient to send the command [17]. In the case of PUPIL though, a more sophisticated solution was required due to the following reasons:

- The issuer (tab/menu item) of the event is not directly aware of the recipient that will handle it
- Flexibility would be compromised if the delegate was hardcoded (e.g., when trying to replace the delegate invoked by the tab/menu item when clicked or add a new one).

Since a dynamically set callback mechanism would be more appropriate, the Command pattern was followed. The Command pattern is a design pattern that encapsulates the concept of the command into an object. The issuer holds a reference to the command object rather than to the recipient and sends the command to the command object by

executing a specific method on it. The command object is then responsible for dispatching the command to a specific recipient to perform the desired action.

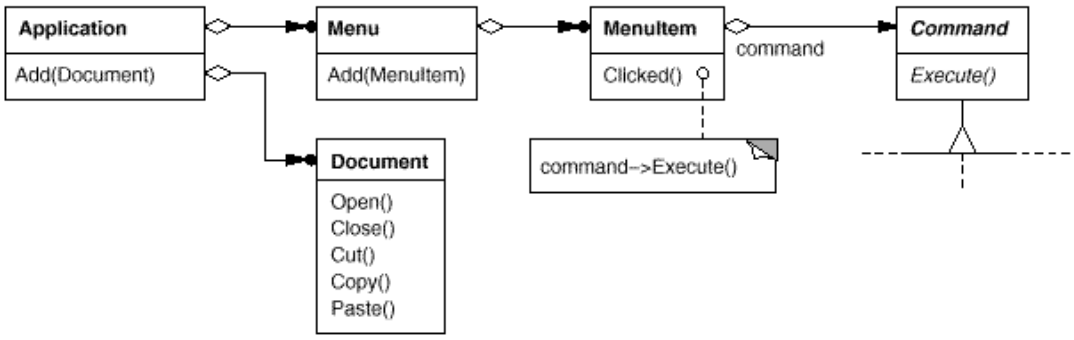


Figure 24: The Command design Pattern

4.2.2 AmIDesk and SmartDesk Window Managers

4.2.2.1 Overview

The AmIDesk and SmartDesk Window Managers have similar characteristics, since they are both proposed for the student's classroom desk. Their key feature is to display two applications simultaneously to enable student's parallel work. In addition to parallel display, these Window Managers support working area organization by offering application sliding from left to right and vice versa. Sliding –named from that point forward as "browse" – or "transfer" commands execution (i.e., migrate an application instance to the SmartBoard) can be achieved through a special-purposed Pie Menu. The Clipboard and Toolbar utilities offered by both Window Managers will be described in more details in the next section.

Since both the AmIDesk and SmartDesk artifacts offer widescreen resolutions, any application that needs to display additional information such as history or help, can take advantage of the extra horizontal space and launch a new application next to it with the relevant content.

The available widescreen resolutions, 1600x600 for the AmIDesk and 1440x900 for the SmartDesk, resulted to the following Window Manager layouts.

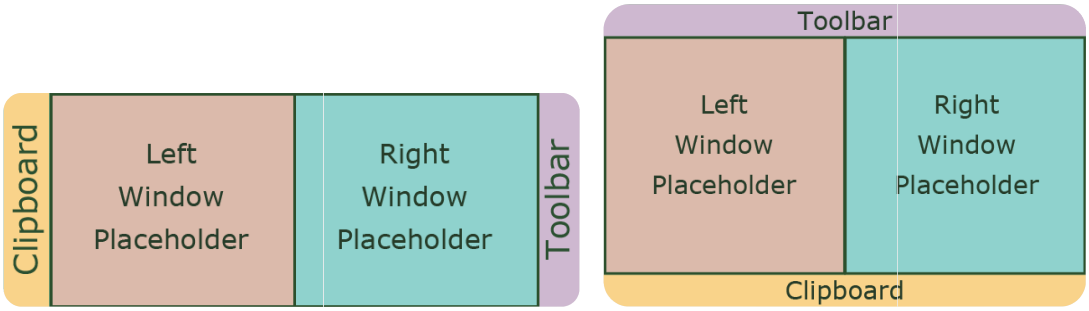
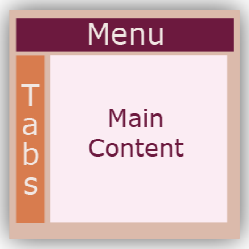


Figure 25: The Window Manager layouts for the AmIDesk and SmartDesk artifacts

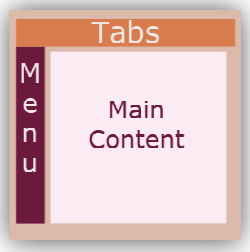
4.2.2.2 Application Layout

The application layout alternatives for both the AmIDesk and SmartDesk artifacts are:

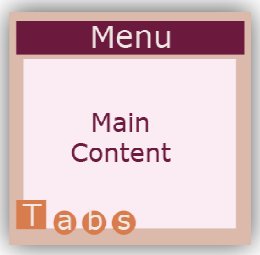
- Horizontal Menu
- Vertical Tabs
- Main Content



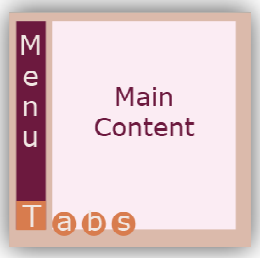
- Vertical Menu
- Horizontal Tabs
- Main Content



- Horizontal Menu
- Expandable Tabs
- Main Content



- Vertical Menu
- Expandable Tabs
- Main Content



4.2.2.3 Toolbar

The Toolbar is a specific area offered by the AmIDesk and SmartDesk Window Managers where useful shortcuts are displayed. The Class Board and Personal Area shortcuts are always visible, while the lower area of the Toolbar is populated with course related applications. For instance, during the English class the Dictionary shortcut is displayed on the Toolbar.

For each course, an XML file describes all the related applications that need to be added to the Toolbar. The provided data are: the shortcut's icon, the absolute path to the binary file that contains it (dll library) and the application's name, so by parsing the appropriate XML file the Toolbar gets populated with the application shortcuts.

When a shortcut is selected, the respective application must be launched. As a result, the launch method of the Window Manager is executed taking the respective application as a parameter.

4.2.2.4 Clipboard

The Clipboard is a utility that holds instances of applications temporarily - e.g., one school hour- and disposes its contents before the next class (the student will be asked whether to save the pinned application instances in his personal area [26]). As soon as an application is pinned to the Clipboard, its state is stored and the student can restore it by selecting the respective application thumbnail.

A student can use the Clipboard to pin an application instance in order to access it later. The Clipboard permits browsing the pinned thumbnails through scrolling. When a thumbnail is pressed, it disappears from the Clipboard and the relevant application is either launched if suspended, or brought to front displaying the data recalled through the pinned instance.

Consider the following example. A student is browsing through some pictures using the Multimedia application and finds an interesting picture of a roller coaster. He pins it to the Clipboard and continues browsing. After a while he decides to take another look to the roller coaster image, so he selects the pinned thumbnail from the Clipboard. The Multimedia application is then brought to foreground displaying the roller coaster image.

The Clipboard items can be removed by striking through them (right to left). For mass removal, an “Unpin All” button is provided to facilitate the students.

Figure 26: The Clipboard of the AmIDesk Window Manager

4.2.2.5 Pie Menu

The Pie Menu is a round menu that appears when the student touches the screen at the same point continuously for more than 400 milliseconds, displaying a number of useful options. The concept was to give the student the opportunity to reorganize the launched applications so as to facilitate studying and simplify the processes of submitting an exercise and display something to the entire classroom. So the menu includes “browse” commands such as sliding applications from left to right and vice-versa in order to assist the student browsing of the launched applications, as well as “transfer” commands such sending application instances to the Clipboard, the class board and the teacher. When the Pie Menu is activated the student can either execute a command by sliding his finger over the desired option, or cancel his action and deactivate the popup menu by lifting his finger from the screen.

The original idea was that the student could just move his finger left, right, up and down to execute a predefined action. However, the drawback of that concept was that those gestures would be probably used by the applications too, and the system would not be able to distinguish whether the student intended to interact with the application or perform a browse/transfer gesture. For instance, consider a simple paint application and a student trying to slide the application window from left to right; the system would be confused whether to perform the sliding or draw a line following the student’s finger.

To overcome this difficulty, two alternatives were considered. The first one was to define specific areas on the application windows where which the student could perform the browse/transfer gestures. However, this technique would result to significant decrement of valuable content space. The second alternative, which was finally adopted, was to reserve one gesture (not commonly used by the majority of applications) in order to activate a menu including all the browse/transfer commands.

The selection of a round type of menu aims to minimize student effort and eliminate the possibility of errors. Once the Pie Menu is activated, the student can immediately make his selection by just sliding his finger over one menu item. Furthermore, the possibility of error is reduced due to the appropriate size of menu items.

In more details, the Pie Menu options are: slide left, slide right, send to clipboard, send to class board, send to teacher. As soon the menu is activated, only one sliding option is displayed according to its position, so if the menu is launched at the left part of the screen the “slide right” is visible. These two options appear at the left and right of the Pie Menu

respectively, fulfilling the natural mapping design principle. The same principle is followed by the “send to class board” option, which was placed at the top center of the round menu to take advantage of the physical board’s placement in a classroom. As soon as an application instance is sent to the class board, the same application is launched at the AmIBoard or the SmartBoard artifact, displaying the same data for all the students to see. Finally, the “send to teacher” option is visible only when the selected application is an exercise that needs to be submitted to the teacher.

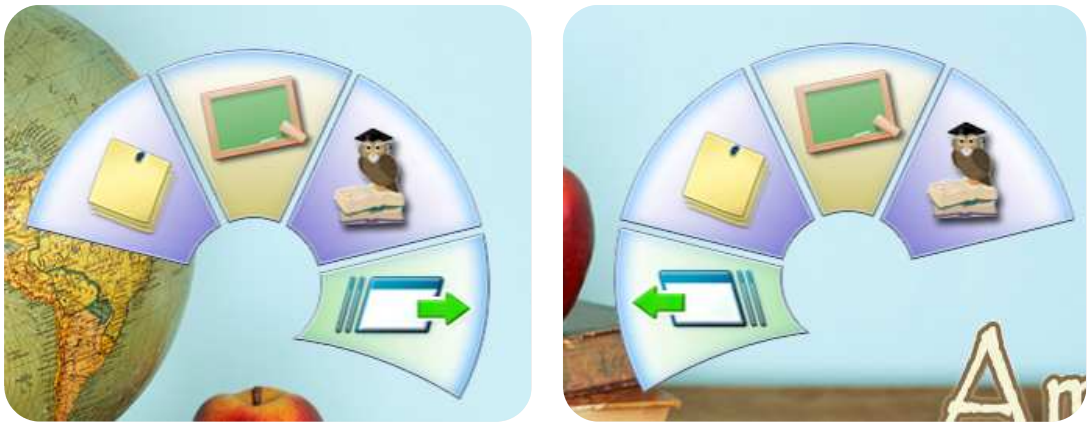


Figure 27: The Pie Menu of the AmIDesk Window Manager

4.2.2.6 Application Switcher

The Application Switcher is a utility activated when the student touches a specific area on the screen (down right corner for the AmIDesk and up right corner for the SmartDesk).

It displays thumbnails previewing the launched applications using a grid layout, and the student can select one or two applications to come to front. When an application thumbnail is touched, the Application Switcher collapses and the respective application(s) come to front. When active, the Application Switcher can be closed by touching anywhere else on the screen.

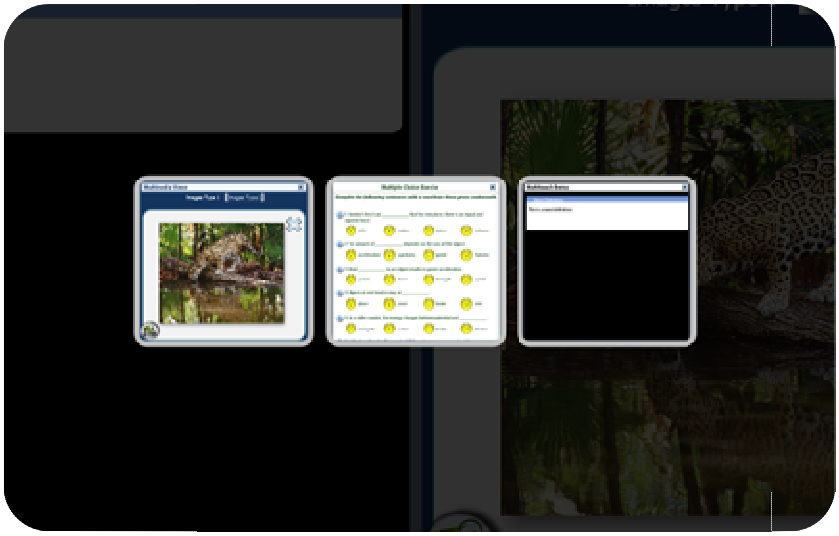


Figure 28: Snapshot of the SmartDesk Application Switcher when three applications are launched

4.2.2.7 Placement Decision Making Component

The fact that the AmIDesk and SmartDesk Window Managers support the presentation of two applications simultaneously raised the need for a Placement Decision Making Component. This component decides the placement (left or right) of each newly launched application according to the following algorithm:

1. If an available spot exists, launch the new application there
2. If the new application provides additional information related to an already launched application, place it at the opposite position
3. Else
   - 3.1. If there is a most recently used application, launch the new one at the opposite position
   - 3.2. Else, launch the new application at the opposite position of the last launched application

The second step of the aforementioned algorithm ensures that an application and its additional information will be launched next to each other. Furthermore, as step 3.1 depicts, the algorithm prioritizes the most recently used application by not placing the new one on top of it, preserving the student's working area on focus.

4.2.2.8 Glass Input Handler

The Glass Input Handler is a transparent mechanism that supports the "browse" and "transfer" commands of the AmIDesk and SmartDesk Window Managers. The concept of this mechanism was based on Java's Glass Pane, which is like a sheet of glass over all the UI elements intercepting input events for the Root Pane [36]. Similarly, the Glass Input Handler

is handling the touch events transparently and either propagates them to the respective applications or launches the Pie Menu.

Before explaining the internal mechanics of the Glass Input Handler in more details, the event types generated by the touch hardware/software and handled by the system should be defined. Both a touch enabled device (SmartDesk) and a vision enabled touch system (AmIDesk) generate the same types and sequences of events.

The three discrete types of touch events are TouchDown, TouchUp and TouchMove. TouchDown denotes that a user's hand contacted the touch-sensitive screen while TouchUp denotes the loss of contact. The TouchMove event is raised every time the user is sliding a finger across the screen without losing contact.

The aforementioned touch event types may seem that they only concern single touch interaction. However, any multi touch interaction can be reduced to a sequence of single touch events. Thus, the valid touch interactions that a student can execute are:

Momentarily touch: When the student touches the screen momentarily, a TouchDown event is raised followed by a TouchUp.
Continuous touch: When the student holds down his finger for some time and then lifts it, the generated events are a TouchDown, a sequence of TouchMove events (near to the TouchDown point) and then a TouchUp.
Drag: When the student touches the screen, drags his finger across it and then lifts it, the generated events are a TouchDown, a sequence of TouchMove events following the finger track and finally a TouchUp.

When a TouchDown event is raised, it must be checked whether the student was aiming an application or intending to activate the Pie Menu. Due to the layout and size of the Pie Menu, a specific area is defined in which the user can activate it, ensuring that it will be painted inside the screen boundaries. So every TouchDown event outside that area is immediately propagated to the first UI element of the respective application, listening to this kind of events.

As already mentioned, for the Pie Menu to be activated the student should hold a single finger at the same point for more than 400 milliseconds. A single TouchDown event inside the designated area indicates that the student may intend to activate the Pie Menu. If this event is followed by a TouchUp, then both events are being propagated since this event sequence constitutes a Momentarily touch. Similarly, if the initial TouchDown is followed by

new TouchDown events, then they are all propagated since Multi Touch interaction is indicated. On the other hand, if the TouchDown is followed by a sequence of TouchMove events distant enough from the initial touch point, then all the events including the initial TouchDown are propagated since a simple Drag interaction is indicated. On the contrary, if the sequence of TouchMove events is confined near the point that the TouchDown event occurred then if a time interval of 400 milliseconds is reached the Pie Menu is activated. The algorithm used by the Glass Input Handler and the respective STN diagram are presented bellow:

- If a TouchDown event occurs outside the designated area, propagate it
- Else if a TouchDown event occurs, keep it on hold
  - If a TouchUp event occurs
    - If the Popup Menu is activated, hide it
    - Else
      - If exists, propagate the on hold TouchDown
      - Propagate the current TouchUp
  - If another TouchDown event occurs
    - If exists, propagate the on hold TouchDown
    - Propagate the current TouchDown
    - Propagate anything from this point on until every active touch is handled
  - If a TouchMove event occurs
    - If the move point is distant enough from the initial TouchDown point
      - If exists, propagate the on hold TouchDown
      - Propagate the current TouchMove
      - Propagate any event from this point on until every active touch is handled
    - Else
      - If the time interval reaches 400 milliseconds
        - Activate the Pie Menu

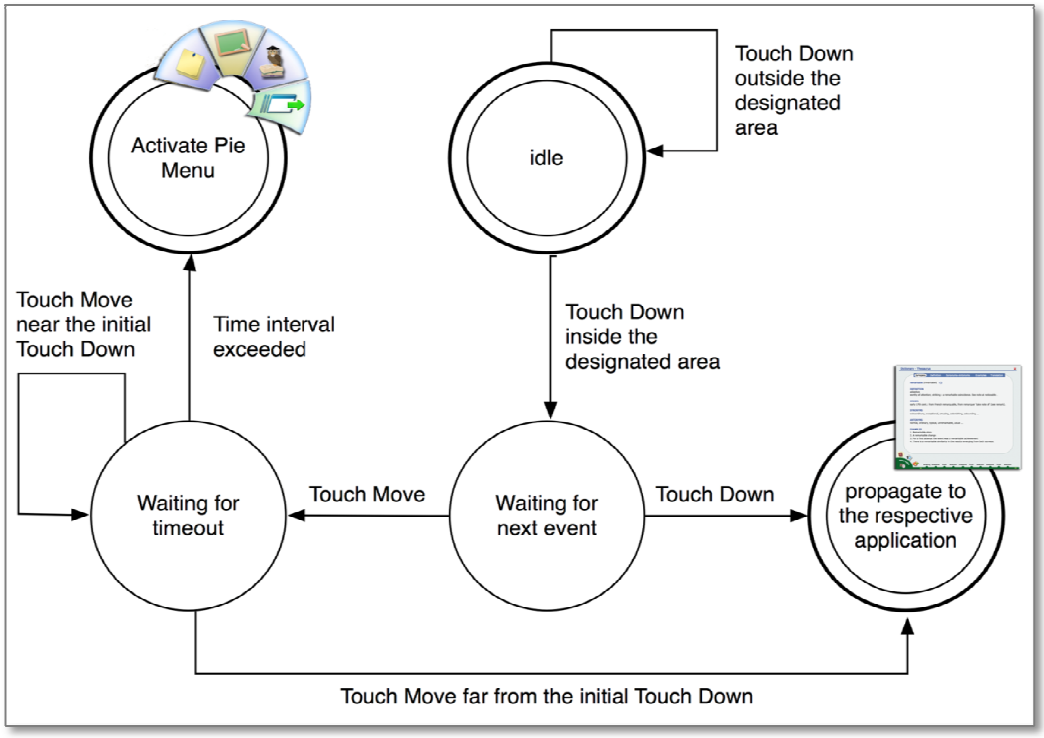


Figure 29: STN diagram that represents the rationale of Pie Menu activation

4.2.3 Netbook Window Manager

The Netbook Window Manager includes a Toolbar, an application placeholder and a Sidebar (a vertical expandable area for displaying additional information). Only a single application that fills the entire available space can be visible at any time, while the student can access all the launched applications and choose one to bring to the foreground through an Application Switcher mechanism similar to the one used by the AmIDesk and SmartDesk. The Netbook Window Manager layouts are depicted in the following figures.

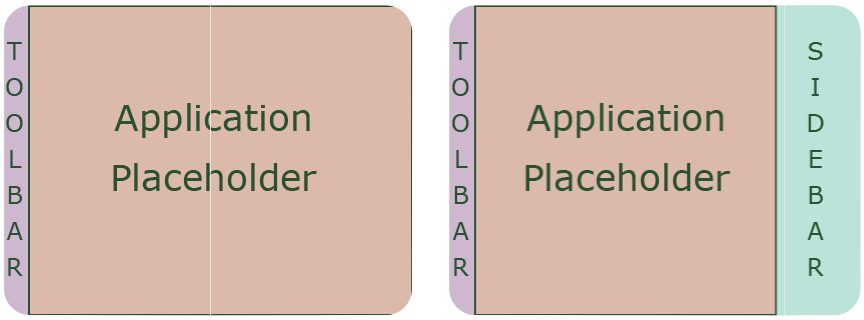
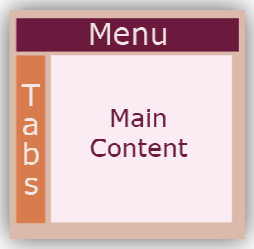


Figure 30: The Netbook Window Manager layouts

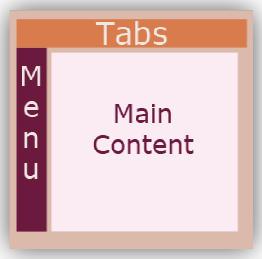
4.2.3.1 Application Layout

The application layout alternatives for Netbook artifact are the following:

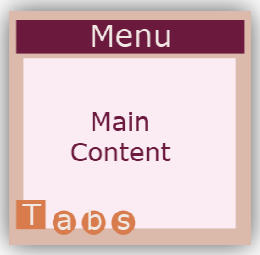
- Horizontal Menu
- Vertical Tabs
- Main Content



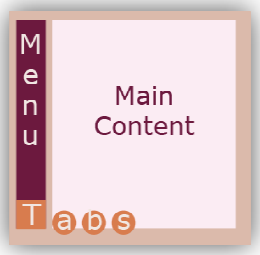
- Vertical Menu
- Horizontal Tabs
- Main Content



- Horizontal Menu
- Expandable Tabs
- Main Content



- Vertical Menu
- Expandable Tabs
- Main Content



4.2.3.2 Toolbar

The Toolbar is a specific vertical area displayed on the left of the application placeholder, which holds a list of useful shortcuts.

The first shortcut opens the student's Personal Area. The student's Netbook "works" outside the AmI Classroom environment and independently from the course hours. Consequently, a student wishing to launch the English Dictionary application must browse through his Personal Area, find the English Course Area, choose to view the English course applications and finally select the Dictionary. As soon as a user selects an application from a specific course area (i.e., English Area) a shortcut to this area appears on the Toolbar.

The next shortcut displays the Application Switcher from which the student can select an already launched application by clicking on the respective thumbnail.

At the bottom of the Toolbar there are a "Save" and a "Print" option. The "Save" option is used to save the instance of the visible application to the student's Personal Area. Consider the following example. The student completes a multiple-choice exercise, which was his homework for the English course, and saves it to his Personal Area in order to access it the next day from his desk (i.e., AmIDesk, SmartDesk). The "Print" option uses a local printer to print the displayed data.

4.2.3.3 Sidebar

Since only a single application can be visible at any time, a Sidebar that emerges at the right of the application and covers 25% of the screen width is offered to display additional information (i.e., help, history, etc.).

The Sidebar becomes visible only when the displayed application needs to present additional data. When the Sidebar is activated, the application shrinks in order to make space for the sidebar to be displayed. When the sidebar collapses, on the other hand, the application restores its original size by taking over the space previously used by the Sidebar.

4.2.4 SmartBoard and AmIBoard Window Managers

The SmartBoard and AmIBoard Window Managers are intended for a board artifact. They both provide the Toolbar and Sidebar features, while the AmIBoard Window Manager additionally supports a layout that integrates application-related previews to utilize the extra space offered by its higher resolution (1920x1200). The SmartBoard and AmIBoard Window Managers' layouts are depicted in the following figures.

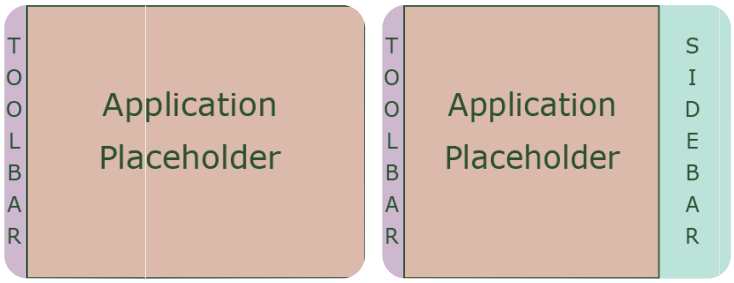


Figure 31: Available layouts for both SmartBoard and AmIBoard Window Managers

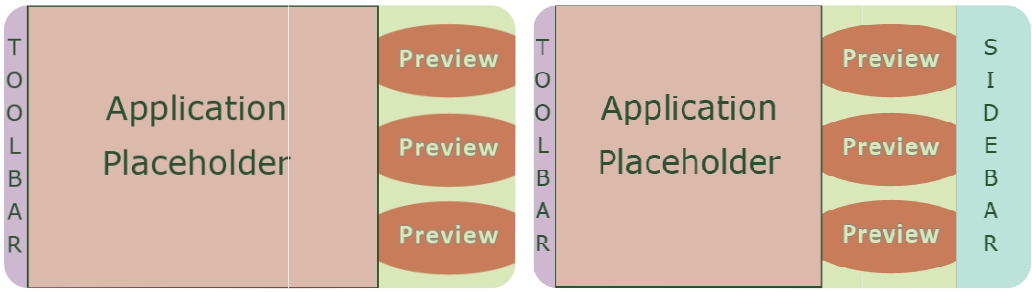


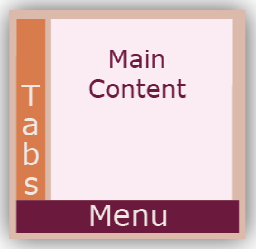
Figure 32: Available layouts for the AmIBoard Window Manager

4.2.4.1 Application Layout

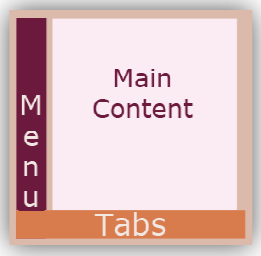
Since the board artifact is seen by all students, it is important to ensure that all interactive controls are positioned appropriately without excluding the younger learners. For that to be achieved, the available application layouts for this artifact contain proper placeholders for

the Tabs and Menus. Furthermore, as depicted in the Figures below, when a Tab or Menu list is placed vertically next to the content of an application, it must anchor at the bottom of the designated placeholder.

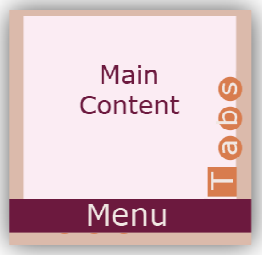
- Horizontal Menu
- Vertical Tabs
- Main Content



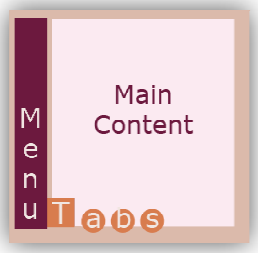
- Vertical Menu
- Horizontal Tabs
- Main Content



- Horizontal Menu
- Expandable Tabs
- Main Content



- Vertical Menu
- Expandable Tabs
- Main Content



4.2.4.2 Application Launcher

As a board artifact is aware of the current course, it was necessary to determine an appropriate placeholder for displaying course-related application shortcuts. The concept of placing these shortcuts on the Toolbar area was discarded, as it was quite likely that some of them might become unreachable for younger learners. For that reason, a new module was introduced, the Application Launcher, which displays a grid of application shortcuts following a bottom-up placement algorithm. When activated, this module covers the area designated for a PUPIL application.

4.2.4.3 Toolbar

The Toolbar items of a board artifact are the Application Switcher shortcut, the Application Launcher shortcut, and the application close control. The Board, as opposed to the Netbook artifact, does not display information concerning an individual student, thus the Personal Area shortcut (included in the Netbook's Toolbar) was redundant. Taking into consideration the Toolbar's vertical placement, its contained items anchor at the bottom to be reachable by all students.

4.2.4.4 Previews

The idea of previews originated during the design of the enhanced Dictionary application. This application has the following categories and subcategories:

- Dictionary/thesaurus

- Definition
- Synonyms/Antonyms
- Examples of use

- Multimedia
  - Images
  - Videos
  - Sounds

In order to take advantage of the AmIBoard extra space, the concept was to display in parallel to the word's definition, snapshots of related pictures and videos. The designer can easily employ this feature by selecting the appropriate preview-enhanced layout that gives users a glimpse through the application's categories/subcategories. Figure 33 presents such a layout for the enhanced Dictionary Application. While the students observe images related to the word "remarkable", the preview placeholders may display the definition of the word and/or some relevant videos and audio. When the student or teacher points to one of the preview areas the respective category/subcategory is displayed.

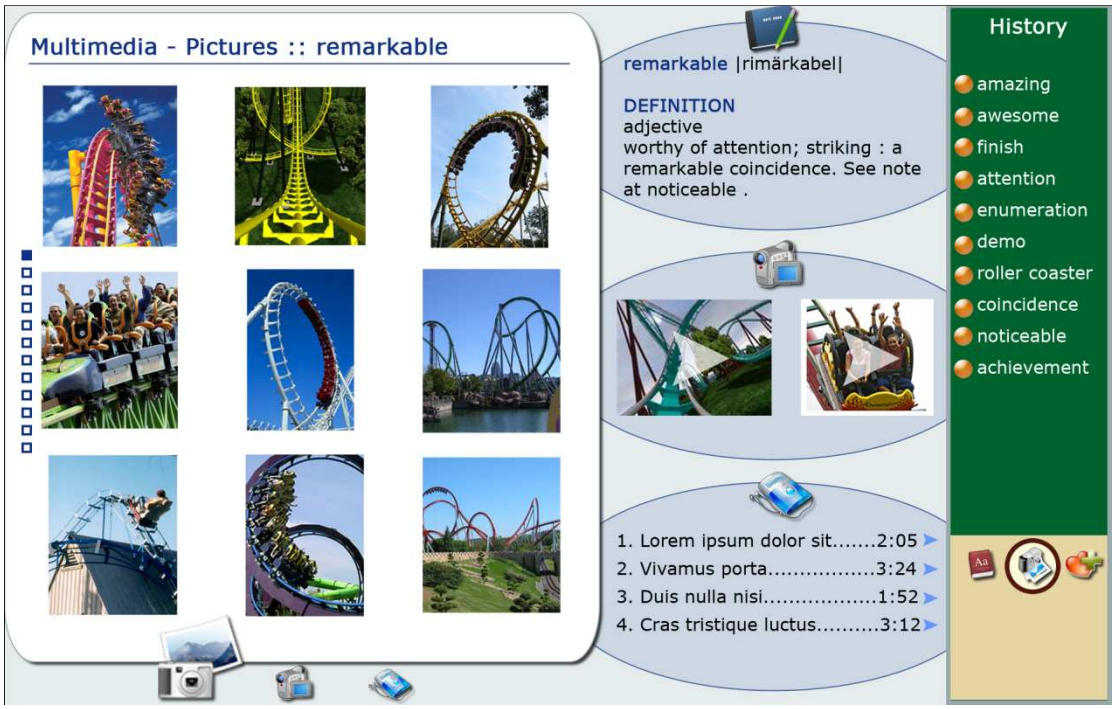


Figure 33: The Multimedia Category of the enhanced Dictionary Application presented on the AmIBoard

5 Developed applications

The PUPIL Widget Library was used to build some educational applications in order to evaluate the library itself and the developed Window Managers.

5.1 ClassBook Application

The ClassBook Application is the electronic version of the currently open page of the physical book. The images and exercises displayed on any page are selectable, and when selected the appropriate applications are launched to display relevant content (e.g., the Multimedia Application is launched if an image is selected or the Multiple-Choice Exercise if an exercise is selected).

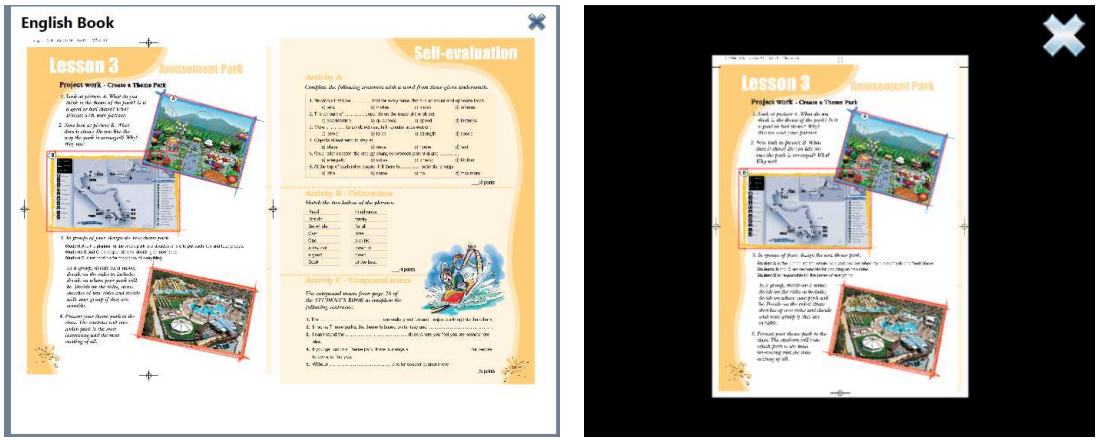


Figure 34: Snapshots of the ClassBook Application as displayed on the ClassBook Application

5.2 Multimedia Application

The Multimedia Application as implied by its name displays multimedia content. The student can choose to view images or videos about a preselected topic. For example, the student can select an image displayed on the ClassBook Application to view relevant multimedia.

Figure 35: Snapshots of the Multimedia Application as displayed on the SmartDesk artifact

5.3 Multiple-Choice Exercise Application

This application is the online representation of a multiple-choice exercise. Through this application the student can solve the exercise electronically by just selecting the one of the possible answers. A hint button is offered next to each sentence, in case the student requires help to find the correct answer. As soon as the hint button is selected, the Hints application is launched offering appropriate information.

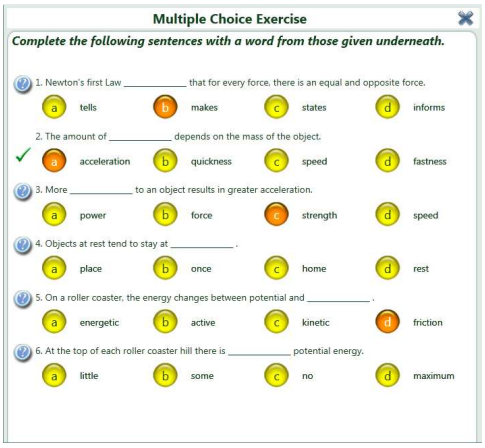


Figure 36: A Multiple-Choice Exercise Application as depicted on the SmartDesk artifact

5.4 Hints Application

The Hints Application is launched only when the student explicitly asks for help about a specific exercise. It supports three kinds of hints that are presented to the student gradually in order to assist the development of critical thinking skills. If the first hint is insufficient for the student to solve the exercise he can select to view the second one and similarly the third one.

Figure 37: The Hints application as displayed on the SmartDesk artifact

6 Heuristic Evaluation

As the development phase had advanced, a heuristic evaluation was conducted in order to eliminate serious usability errors of the SmartDesk Window Manager and the developed applications. Heuristic evaluation is the most popular of the usability inspection methods and is carried out as a systematic inspection of a user interface design for usability [33]. The goal of heuristic evaluation is to find the usability problems in the design so that they can be attended to as part of an iterative design process. The process involves having a small set of evaluators examine the interface and judge its compliance with recognized usability principles (the "heuristics"). In general, heuristic evaluation should involve more than a single evaluator, since experience has shown that different people find different usability problems. According to Nielsen [34], the best approach is to involve three to five evaluators, since larger numbers do not provide much additional information.

During, heuristic evaluation each individual evaluator inspects the interface alone and compares it with a list of recognized usability principles (the "heuristics") described in Table 9. As soon as all the evaluators have completed the aforementioned process the discovered usability errors are aggregated in a list with references to those usability principles that were violated by the design in each case in the opinion of the evaluators. Finally, each evaluator is called to provide severity ratings independently of the other evaluators. The severity ratings range from zero to four (Table 10) and are used to indicate how serious each problem is and how important is to fix it.

	Heuristics	Interpretation
1	**Visibility of system status**	The system should always keep users informed about what is going on, through appropriate feedback within reasonable time
2	**Match between system and the real world**	The system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order
3	**User control and freedom**	Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support undo and redo
4	**Consistency and standards**	Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions
5	**Error prevention**	Even better than good error messages is a careful design, which prevents a problem from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action
6	**Recognition rather than recall**	Minimize the user's memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of

		the system should be visible or easily retrievable whenever appropriate
7	**Flexibility and efficiency of use**	Accelerators -- unseen by the novice user -- may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions
8	**Aesthetic and minimalist design**	Dialogues should not contain information, which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility
9	**Help users recognize, diagnose, and recover from errors**	Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution
10	**Help and documentation**	Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large

Table 9: The usability heuristics used by the evaluators

Severity Rating	Interpretation
0	I don't agree that this is a usability problem at all
1	Cosmetic problem only: need not be fixed unless extra time is available on project
2	Minor usability problem: fixing this should be given low priority
3	Major usability problem: important to fix, so should be given high priority
4	Usability catastrophe: imperative to fix this before product can be released

Table 10: Severity Ratings

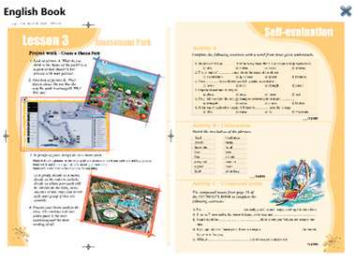
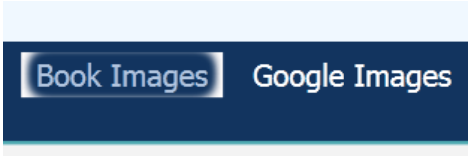
The evaluation of the SmartDesk and the developed applications was performed by five evaluators, which followed the aforementioned process. The usability issues discovered by the five evaluators where noted down by an observer, who was monitoring the evaluation progress. The complete list of usability issues along with the violated rules and average values of severity ratings is presented in Table 11.

The most severe issues regarding the PUPIL's UI collection were:

- the lack of indication of the available images in the PUPIL ImageViewer widget
- the absence of the thumb on the PUPIL Scrollviewer widget
- the absence of actually "filling the gap" functionality in the PUPIL MultipleChoiceElement

As far as the SmartDesk Window Manager is concerned, four important issues were discovered:

- Pie Menu should not be activated when no applications are launched
- Application Switcher should not be activated when no applications are launched
- Applications should allow maximization
- The application slide should be performed by direct gestures

Book Viewer Application	
One specific book page can be opened if the user touches an inactive area. The same interaction pattern should be supported for closing the book page and returning to the two-page layout. **Rules violated:** Consistency and standards, Flexibility and efficiency of use **Severity Rating:** 1,9	
When two active elements intersect, it is not clear which will be activated when their intersection is touched. **Rules violated:** Recognition rather than recall **Severity Rating:** 1,3	
Multimedia Application	
The main menu appearance does not make clear to the user which is the active option and which one is the currently selected (Book Images / Google Images). **Rules violated:** Visibility of the system status, Recognition rather than recall **Severity Rating:** 2,6	
The expandable tab icon is not intuitive, indicating both its contents as well as its functionality. Furthermore, it is very cluttered.	

Rules violated: Recognition rather than recall

Severity Rating: 3

Once the expandable tab menu has opened, there is no indication for the currently selected tab item.

Rules violated: Visibility of the system status

Severity Rating: 2,9

When the multimedia application is initiated from the main menu (and not from the book application), there are no contents to be displayed. A search field for providing a keyword is required.

Rules violated: Visibility of system status, User control and freedom

Severity Rating: 3,1

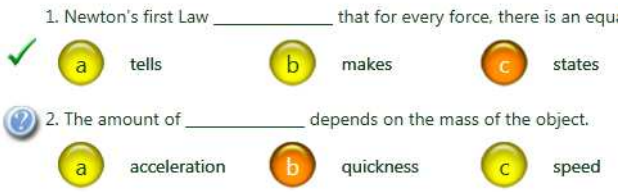
An indication for the number of the available results should be provided.

Rules violated: Visibility of system status

Severity Rating: 2,6

Exercise Application

When students request confirmation for the validity of their answers in a given exercise, feedback should be provided not only for the correct answers but for the erroneous as well.



Rules violated: Visibility of system status

Severity Rating: 3,6

For a given row of answers it is not clear to which questions they correspond. Sequential questions could be differentiated with the use of different shades of a light background colour.

Rules violated: Error prevention

Severity Rating: 1

When an answer has been selected in a given “fill-in the gap” question, the selected option could not only be highlighted but also filled in the corresponding gap.

Rules violated: Visibility of system status

Severity Rating: 1,5

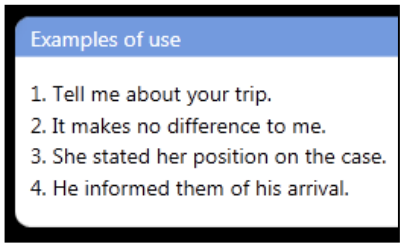
Hints Application

The wording of the first hint (Word Definition) is not appropriate, in order to help users understand that the provided definition refers to the word he/she is looking for. It could be rephrased as: “You are looking for a word that means:”

Rules violated: Error prevention, Match between system and the real world

Severity Rating: 3

The wording and the formatting of the second hint (Examples of Use) is not appropriate in order to help users understand that one example is provided for each one of the available options. The hint could be rephrased as: “Examples of use for the available choices”. Furthermore, the examples should be numbered with “a., … d.” instead of “1., … 4.”. Finally the words for which the examples are being provided should be marked in bold.



Rules violated: Error prevention, Match between system and the real world

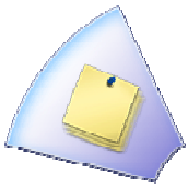
Severity Rating: 2,6

The icon for asking the second hint is not intuitive, especially for children. As a result students might accidentally ask for help even if they do not want to. It should be replaced by text.

Rules violated: Error prevention, Match between system and the real world

Severity Rating: 2,8

Application Switcher
When no applications are active and the application switcher is initiated, the screen becomes blank (since no application thumbnails are available to display). In this case, the application switcher should not be activated at all. Rules violated: Error prevention **Severity Rating:** 3,4

Pie Menu	
The icon used for the interactive pie menu area which can be used to move a window left or right is not intuitive. **Rules violated:** Match between the system and the real world **Severity Rating:** 1,6	
The icon used for the interactive pie menu area which can be used to pin an application on the clipboard is not intuitive **Rules violated:** Match between the system and the real world **Severity Rating:** 0,6	
The icon used for the interactive pie menu area which can be used to send something to the teacher is not intuitive. **Rules violated:** Match between the	

system and the real world

Severity Rating: 1,2

When no applications are open and the user touches the desktop for a prolonged time, the pie menu is activated. However, in this case it is out of context as it has no usefulness and might confuse users.

Rules violated: Error prevention

Severity Rating: 3,6

Clipboard

When an application pinned to the clipboard is selected, it is restored for the user to view and it is automatically removed from the clipboard. Applications should remain pinned to the clipboard until the user decides otherwise.

Rules violated: User control and freedom

Severity Rating: 2

The “Unpin all” button is not at all intuitive and does not inform users of the action that it will perform. Especially since it performs a crucial action (i.e. it deletes all the clipboard contents) it needs a more intuitive icon.

Rules violated: Error prevention, Match between the system and the real world

Severity Rating: 3,2

General

In general, direct gestures should be supported whenever possible. For example, windows could be moved left or right, by just dragging them towards the desired direction and not by selecting an icon from a menu.

Rules violated: Match between the

system and the real world, Flexibility and efficiency of use **Severity Rating:** 2,4
Application windows could also allow users to maximize them. For example, if users could maximize the book application, the two-page layout that is used would be more readable. **Rules violated:** Flexibility and efficiency of use, User control and freedom **Severity Rating:** 2,3
Scrollbars should always indicate the current scrolling area in comparison to the overall contents. **Rules violated:** Visibility of system status **Severity Rating:** 2,8

Table 11: The complete list of usability issues discovered during the heuristic evaluation process

The chart presented in Figure 38 illustrates the heuristic evaluation results, ordered according to their severity. The most severe usability issues that were ranked higher than "major" usability problems (>3) are marked in red color. These findings are the most imperative to fix. The issues ranked higher than "minor" (>2) and which are represented in the figure by the orange coloured bars are next in turn to be fixed. Issues marked in blue color range from "just aesthetic" problems (severity rating 1) to "minor" problems (severity rating 2) and will be fixed before the next system version becomes available. Finally, the issue rated lower than 1, which is marked with green color, is currently under discussion, as most of the evaluators disagreed that it was a usability problem at all and rated it with 0.

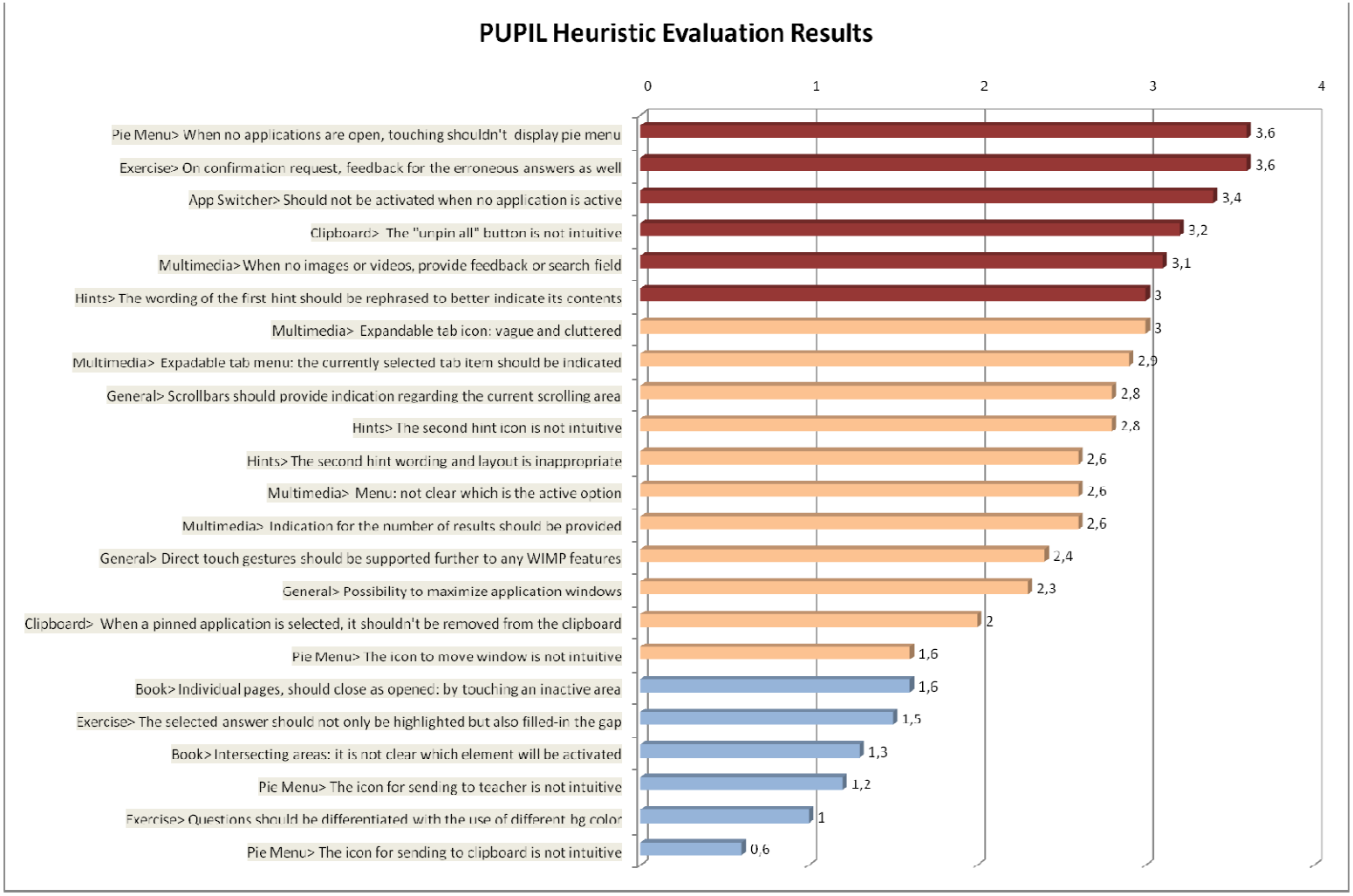


Figure 38: Heuristic evaluation results

7 Conclusions and Future Work

7.1 Summary

The main objective of this thesis was to lay the foundation towards the realization of the envisioned AmI Classroom by supporting the development of user interfaces for the various artifacts which may comprise the classroom technological environment. This was achieved through the PUPIL system, which (i) empowers the designers of educational applications and (ii) offers appropriate workspaces to support students' work.

Using PUPIL, designers can easily create applications that can be optimally displayed at the student's desk, student's Netbook or the classroom board. This requires minimum effort on their behalf, and only a single interface implementation is needed, since the PUPIL widgets are able to transform appropriately according to the characteristics of the available artifact. Furthermore, the complex widgets also provided by PUPIL promote reusability of common interface patterns and minimize the designers' work, since numerous design decisions have been made a priori.

In order to support the students through the everyday educational activities, each artifact is equipped with an appropriate workspace (Window Manager). The Window Managers for the artifacts taken into account were created keeping in mind the special characteristics of each classroom artifact, as well as the user's goals of use.

Both Widgets and Window Managers were developed following a user centered design process (UCD). Through such process, the user needs and goals became clearer and the feedback received through the formative and heuristic evaluations led to significant improvements. The conducted design phase also led to the elaboration of a set of design guidelines for AmI classroom applications, which have been instantiated in the realization of the PUPIL widgets, but can also be used independently.

Finally, the PUPIL system and its available widgets have been extensively used in the development of a set of AmI classroom applications which aim to validate the overall approach, as well as its added value for designers and developers of educational applications.

In summary, the outcomes of the presented work include:

- the PUPIL widget library for AmI classroom applications, addressing varioys types of interactive desks and borads, as well as standard student netbooks.

- the Window Managers for each of the addressed classroom artifacts, especially designed to exploit at best the physical characteristics and usage modalities of each device

- the developed educational applications for the AmI Classroom

- the results of the conducted requirements analysis and evaluation processes, including the derived design guidelines for AmI classroom applications

7.2 Conclusion

In combination with the ClassMATE system, which realizes the overall AmI classroom infrastructure, the PUPIL system supports scenarios such as the following:

- the student points an image to the electronic version of the book and the Multimedia application is launched to display relevant content
- the student send an interesting image to the augmented board in order to discuss it with the entire class
- the student decides to solve an exercise electronically, points the exercise to the electronic version of the book and the electronic version of the exercise is launched
- the student asks a hint for a specific exercise and the hint application is launched offering appropriate information.

Overall, it can be claimed that this work constitutes a significant first step towards supporting the extensive use of AmI technologies in the context of the classroom and of the educational process more in general, by (i) informing the design of applications which follow appropriate design guidelines and (ii) facilitating the development of such applications through hiding the complexity deriving from their use on various artifacts and devices.

7.3 Future work

Hereafter, additional steps should be taken to fully support the initial concept.

The next step of this work would be to conduct a full-scale user-based evaluation of the currently developed system, in order to acquire additional useful feedback from end users. The evaluation experiment should include all the supported artifacts, as it is important to record and analyze the performance of real users while following a predefined set of tasks at each one of them. The results of this evaluation would lead to further improvements of the Widgets library, the Window Managers and the educational applications in order to better meet the students' needs.

Furthermore, the PUPIL UI collection needs to be enriched by both simple and complex widgets in order to fully-support the development of educational applications. It is important to extend the range of addressed subject matters to other courses apart from the English language, in order to identify more common interface patterns that could be realized into complex widgets. An example of this would be a diagram presenter that could be used by applications intended for the Mathematics and Physics courses.

In addition to supporting the student's Netbook, the PUPIL system should take into consideration the increasing number of students that use smart phones on a daily basis and include these artifacts to the AmI Classroom. Towards this end, a new Window Manager – appropriate for handheld devices – needs to be developed and the respective representation logic needs to be incorporated in the existing widgets.

In order to assist the teacher, the PUPIL system could also be enhanced to support the teacher's augmented desk. The goals of a teacher are completely different from those of a student; thus, it is important to follow a requirements elicitation process similar to the one described at Chapter 3 in order to fully understand the teacher's needs. Furthermore, it is of outmost important to determine the tasks that a teacher might perform in the AmI Classroom in order to create the appropriate widgets to support assisting applications.

Finally, an idea to empower the PUPIL system is to integrate the PUPIL widgets into design tools such as Microsoft Expression Blend or Microsoft Visual Studio. That way the designers could easily build the application interface by simply drag and dropping widgets instead of memorizing and typing XAML code, thus enabling them to develop AmI classroom applications without any modification of their current working practices.

Bibliography

[1] Abrami, P., Bernard, R., Wade, C., Schmid, R., Borokhovski, E., & Tamim, R. (2008). A Review of E-learning in Canada: A Rough Sketch of the Evidence, Gaps and Promising Directions.

[2] Antona, M., Margetis, G., Ntoa, S., Leonidis, A., Korozi, M., Paparoulis, G., et al. (2010). Ambient Intelligence in the classroom: an augmented school desk. *Applied Human Factors and Ergonomics.*

[3] Antona, M., Ntoa, S., Adami, I., & Stephanidis, C. (2009). User Requirements Elicitation for Universal Access. In C. Stephanidis, *The Universal Access Handbook.*

[4] Bandelloni, R., & Paterno, F. (2004). Flexible Interface Migration. *Proceedings of the 9th international conference on Intelligent user interfaces.*

[5] Bates, A. (2005). *Technology, E-learning and Distance Education.*

[6] Breuer, H., Baloian, N., Sousa, C., & Matsumoto, M. (2007). Interaction Design Patterns for Classroom Environments.

[7] Brooks, C., Greer, J., Melis, E., & Ullrich, C. (2006). Combining ITS and eLearning Technologies: Opportunities and Challenges.

[8] Bruckman, A., & Bandlow, A. (2002). HCI for Kids.

[9] Carney, R. N., & Levin, J. R. (n.d.). Pictorial Illustrations Still Improve Students' Learning from Text. *Educational Psychology Review* .

[10] Cook, D. J., & Das, S. K. (2007). How smart are our environments? An updated look at the state of the art.

[11] Cooperstock, J. (2001). Classroom of the Future: Enhancing Education through Augmented reallity. *Proc. Conf. Human-Computer Interaction (HCI Int'l 2001)*, (pp. 688–692).

[12] Covington, M. V. (2000). Goal theory, motivation, and school achievement: an integrative review. *Annual Review of Psycology*, (pp. 171-200).

[13] Cross, N. (1989). Engineering design methods. J. Wiley, 1989, 3 pp. 37 – 38.

[14] Doulgeraki, C., Partarakis, N., Mourouzis, A., Antona, M., & Stephanidis, C. (2007). Towards Unified Web-based User Interface.

[15] Druin, A. (1999). Cooperative inquiry: developing new technologies for children with children. *Proceedings of the SIGCHI conference on Human factors in computing systems: the CHI is the limit*, (pp. 592-599).

[16] Druin, A., Bederson, B., Boltman, A., Miura, A., Knotts-Callahan, D., & Platt, M. (1999). Children as Our Technology Design Partners. In A. Druin, *The Design of Children's Technology* (pp. 55-67). Morgan Kaufmann Publishers.

[17] Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). *Design patterns : elements of reusable object-oriented software.*

[18] Gros, B. (2007). Digital Games in Education: The Design of Games-Based Learning Environments. *Journal of Research on Technology in Education* , 23–38.

[19] Hanna, L., Risden, K., & Alexander, K. (1997). Guidelines for usability testing with children. *Interactions* .

[20] Hanna, L., Risden, K., Czerwinski, M., & Alexander, K. (1999). The Role of Usability Research in Designing Children's Computer Products. In A. Druin, *The Design of Children's Technology* (pp. 17-19). Morgan Kaufmann Publishers.

[21] Heller, S. (1998). The meaning of children in culture becomes a focal point for scholars. *The Chronicle of Higher Education* , A14-A16.

[22] *ISO 13407:1999. Human-centred design processes for interactive systems.* (1999). [Motion Picture].

[23] *ISO 9241-210:2010. Ergonomics of human-system interaction -- Part 210: Human-centred design for interactive systems.* (2010). [Motion Picture].

[24] Kaplan, N., Chisik, Y., Knudtzon, K., Kulkarni, R., Moulthrop, S., Summers, K., et al. (2004). Supporting sociable literacy in the international children's digital library. *In Proceedings of the 3rd International Conference on Interaction Design and Children (IDC'04)* (pp. 89-96). ACM Press.

[25] Korozi, M., Leonidis, A., Margetis, G., & Stephanidis, C. (2009). MAID: A Multiplatform Accessible Interface Design Framework. *Proceedings of the 13th International Conference on Human-Computer Interaction (HCI International 2009).* San Diego,CA, USA.

[26] Leonidis, A. (2010). ClassMATE - Classroom Multiplatform Augmented Technology Environment. Heraklion: Computer Science Departement - University of Crete.

[27] Leonidis, A., Antona, M., & Stephanidis, C. (2010). Rapid Prototyping of Adaptable User Interfaces. .

[28] Leonidis, A., Korozi, M., Margetis, G., & Stephanidis, C. (2009). A Content Management Framework for All. *AAATE.*

[29] Loranger, H., & Nielsen, J. (2007). Usability Guidelines for Creating Compelling Websites for Teens.

[30] Macaulay, L. A. (1996). *Requirements Engineering. .* New York: Springer-Verlag.

[31] Midgley, C., Kaplan, A., Middleton, M., & Maehr, M. L. The development and validation of scales assessing students' achievement goal orientations. *Contemporary, Educational Psychology*, (pp. 113-131).

[32] Mori, G., Paternò, F., & Santoro, C. (2003). Tool Support for Designing Nomadic Applications. *International Conference on Intelligent User Interfaces.*

[33] Nielsen, J. (2010). *http://www.useit.com/papers/heuristic/*.

[34] Nielsen, J. (2010). *http://www.useit.com/papers/heuristic/heuristic_evaluation.html*.

[35] OECD. (2010). *Organisation For Economic Co-Operation And Development*. Retrieved from

http://www.oecd.org/document/52/0,3343,en_2649_39263238_45897844_1_1_1_1,00.html

[36] Oracle. (n.d.). *The Java Tutorials - How to Use Root Panes*. Retrieved from http://download.oracle.com/javase/tutorial/uiswing/components/rootpane.html

[37] Osborn, A. F. (1963). Applied Imagination, Scribeners and Sons, New York.

[38] Piaget, J. (1970). Science of Education and the Psychology of the Child. New York: Orion Press.

[39] Said, N. (2004). An Engaging Multimedia Design Model. *Procceding of the 2004 conference on Interaction design and children: building a community*, (pp. 169-172).

[40] Savidis, A., & Stephanidis, C. (2005). Distributed Interface Bits: Dynamic Dialogue Composition from Ambient Computing Resources. *Personal and Ubiquitous Computing*, (pp. 142-168).

[41] Savidis, A., & Stephanidis, C. (2004). Unified User Interface Development: Software Engineering of Universally Accessible Interactions. *Universal Access in the Information Society*, (pp. 165-193).

[42] Savidis, A., Antona, M., & Stephanidis, C. (2005). A Decision-Making Specification Language for Verifiable User-Interface Adaptation Logic. *International Journal of Software Engineering and Knowledge Engineering* , 1063-1094.

[43] Shi, Y., & Xie, E. A. (2003). The smart classroom: Merging technologies for seamless tele-education. *IEEE Pervasive Computing Magazine.*

[44] Stephanidis, C., Antona, M., Savidis, A., Partarakis, N., Doulgeraki, C., & Leonidis, A. (2010). Design for All: Computer Assisted Design of User Interface Adaptation. In *Handbook of Human Factors and Ergonomics.*

[45] Sun, X., Plocher, T., & Weina, Q. (2007). An empirical study on the smallest comfortable button/icon size on touch screen.

[46] Tinio, V. L. (2003). *ICT in Education.*

[47] Urdan, T. C., & Maehr, M. L. (1995). Beyond a two-goal theory of motivation and achievement: A case for social goals. *Review of Educational Research* , 213-243.

[48] Xu P., Han. G. (2009). Towards Intelligent Interaction in Classroom, In: Universal Access in Human-Computer Interaction.

[49] Yau, S. S., Gupta, S., & Karim, F. (2003). Smart Classroom: Enhancing Collaborative Learning Using Pervasive Computing Technology.

APPENDICES

A - Scenarios used in Interviews

Exercise solving on Desk Artifact

After teaching the necessary theory, Mrs. Brown asks her students to solve a gap filling exercise.

The exercise is automatically displayed on the class board, as well as on each student's desk. In order to fill in the gap, each student should just select the appropriate button by touching the screen, or drag and drop the answer to the gap.

After some time, the teacher's system indicates that several students and in fact a percentage of about 43% do not know what to fill in the first gap. Mrs. Brown directly instructs the system to display a hint for the first gap on the class board. A total number of three hints can be requested for each gap, whereas hints are gradually provided by the system, either individually to each student or collectively on the class board.

The first hint for the first gap is displayed on the class board, next to the exercise main window, as a sidebar. Mrs. Brown explains orally that the word that needs to be filled in means to "declare specifically" and provides some examples. The hint remains active until another hint is activated or until the system infers that it is no longer usable (e.g. if all students have answered the exercise), or until the sidebar is explicitly closed.

In the meantime, Helen has completed the first two gaps, by tapping on the desired answer, and is currently in the third one. She asks for a hint to be displayed individually on her desk screen, by double-tapping the gap that should be filled-in (Hints can be activated by double-tapping either on the actual course book or on the desk screen). The question for which the hint is requested is highlighted and a question mark appears next to it.

At the same time, John has answered the first two questions and remains inactive for quite some time. He selects an answer for the third gap, but soon he changes his mind. He swipes the completed answer with his finger, in order to clear the field and continues thinking what to answer to the third question. The system, which is monitoring his performance, asks him if he would like some help, by highlighting the question(s) for which it thinks that the student needs help.

John answers affirmatively and the system provides a hint, by describing the word that should be filled in. The hint application opens in a different window than the exercise application next to it.

The word to fill-in still seems unclear to John, so he selects to see one more hint. This time the system displays examples of the suggested words, as they are used in sentences.

Still John can't decide the word to use, so he asks for one more hint. The last hint is displayed below the other two and excludes two of the four options, allowing the student to select between two possible options.

Exercise solving on Board Artifact

After some time, the teacher's system indicates that more than half of the students in the classroom have provided some answers to all the gaps, while only 38% of them have solved the exercise with a success rate higher than 75%. Given these facts, and that the bell is about to ring, Mrs. Brown asks the students to solve the exercise on the class board. Several students raise their hands and Mrs. Brown asks Joan to come and provide the solution.

Joan is an average-sized girl and therefore she can't reach the top of the class board. As soon as she reaches the board, the interactive content automatic scrolls down in order to be reachable.

Joan fills the first gap, by touching the letter that corresponds to the correct answer.

Mrs. Brown discusses the answers with the class, and when she is about to finish the bell rings for a break. She sums up, gives instructions for their forthcoming homework and dismisses the class.

Exercise solving on Netbook Artifact

Peter is at home being sick and later this afternoon he logs in to his personal area, in order to see the class activities during the day and make sure that he keeps up. The exercise is displayed on his screen.

He starts solving it, and selects an option for the first gap. Soon he realizes that he is not certain about it and after placing the mouse over the completed word, he selects the "delete" button that appears.

Before continuing he decides to request some help, so he clicks on the hint icon next to the question for which he needs a hint. The hint application will provide hints gradually, in a sidebar next to the main window content.

Dictionary use on Desk Artifact

The teacher, Mrs. Brown gets in the class and asks her students to read a passage from the current book page, referring to a theme park in the UK, in order to discuss it later. While reading, Helen comes across the word remarkable. She is not certain about the meaning of this word, so she decides to look for it in the dictionary.

She points at the word with her finger, and the dictionary application appears on her desk's screen, presenting initially a short definition of the word, its pronunciation, a button allowing the student to hear the pronunciation, some synonyms and a few examples of use.

In addition, the dictionary offers options for extended descriptions, grammar information, complete list of synonyms, antonyms and several examples. In order to view some examples of the word remarkable, Helen selects the option "Examples" from the top menu bar.

Students can also select to navigate through the history of the words they have searched in the dictionary (previous /next words). History is displayed as a bar at the bottom of the application window, allowing horizontal scrolling to move to previous and next words, which are not currently displayed.

Helen selects the multimedia button, from the main application menu located towards the lower end of the dictionary application window, and a list with images related to the word "remarkable" appears. Through the multimedia submenu, located near the top of the application window, students can select to view related videos or hear related audios. Students can navigate in the available content either by selecting the next/previous page controls, which appear after double-tapping, or by dragging their finger to the right or to the left, indicating the horizontal scrolling direction.

Helen selects to view an image that seems interesting, by just touching it. A large version of the image is displayed, which can be zoomed in or out by diagonally dragging two fingers either away or towards the center of the image, and rotated by performing the rotation movement with two fingers on the screen. Furthermore, zooming and rotation functionality is available through UI controls, which appear after double-tapping on the screen.

Mrs. Brown notifies the class that they still have 5 minutes in order to complete reading and Helen returns to the text in order to read the last couple of sentences…

Dictionary use on Board Artifact

The teacher, Mrs. Brown gets in the class and asks her students to read a passage from the current book page, referring to a theme park in the UK, in order to discuss it later. While reading, John comes across the word remarkable. He has no idea what it means, so he asks Mrs. Brown.

The word remarkable has been recently taught, so Mrs. Brown decides to ask the class. Only a few students raise their hands to explain the word, and therefore she initiates the dictionary application in the SmartBoard.

The dictionary application appears on the SmartBoard, initially presenting a search field. Mrs. Brown asks John to come up and type in the word. John does so, and after correctly typing the word through the on-screen keyboard, a short definition of the word is displayed, including also its pronunciation, a button allowing students to hear the pronunciation, some synonyms and a few examples of use.

Most of the available board space is occupied by the application's main content. The main application menu (including the options: dictionary, multimedia) is displayed in the bottom right corner, while the history bar becomes always visible on the top right area. Finally, the current submenu is placed near the bottom of the screen, in order to be reachable by younger students as well.

In addition, the dictionary offers options for extended descriptions, grammar information, complete list of synonyms, antonyms and several examples. In order to view some examples of the word remarkable, John selects the option "Examples" from the submenu bar, located at the bottom of the screen, as instructed by his teacher.

Mrs. Brown asks her students if the meaning of the word is now clear.

Helen raises her hand and asks the teacher if the word is the one they were taught last week when talking about theme parks. Mrs. Brown confirms this remark, and selects the multimedia menu item, in order to view the related images and see if theme park images are included.

A list with several rows of images related to the word “remarkable” appears. Through the multimedia submenu, located near the lower end of the application window, students can select to view related videos or hear related audios. Students can navigate in the available content through vertical scrolling, by dragging their finger to the top or to the bottom, indicating the vertical scrolling direction.

Mrs. Brown selects to display an image of a roller coaster that was also included in students’ books, by just touching it. A large version of the image is displayed, which can be zoomed in or out and rotated by using the appropriate controls that are displayed next to the image.

Mrs. Brown now asks the class to continue reading the text in order to discuss it later…

Multimedia View on Desk Artifact

Mrs. Brown asks the class why they think young people like roller coaster rides and asks them to provide audio/visual material in order to support their claims.

John is really intrigued by the photo and since he’s never been on a roller coaster, he is really eager to seek for related material. He points at the photo with his finger and as a result a collection of related photos, videos and audio appears in the screen of his desk.

John navigates in the suggested content by dragging his finger on the desk’s screen. First of all, he selects to see a picture showing a close shot of a child riding the roller coaster. The child seems quite scared but he would like to zoom in the picture for a closer look. In order to do so, he touches the photo with two fingers and moves them away diagonally.

The photo is zoomed in and since the child’s face although scared seems quite funny, John decides to save the photo in his personal student area (See also scenario X, for students’ records). He touches the photo and drags it to the “Home” icon of the desk main menu bar in the right of his screen.

The material is automatically saved in his personal student area, under the category “English>Multimedia”.

Then, he closes the picture and returns to the suggested multimedia content screen. A video there seems also interesting; John selects it and touches the play control.

He thinks that this video can answer the question and therefore he saves it to the Clipboard area, in order to use it later to support his claims. (Clipboard is a vertical area where students can temporarily hold any information they are interested in to review it later).

John raises his hand and suggests that the video he has found could answer the question Mrs. Brown has asked.

The teacher allows him to show the video on the class board and John drags the video from the Clipboard to the board icon located in the main toolbar (which will appear in the rightmost part of the screen). The video is transferred to the board, John selects the play control and now everyone can watch it.

B - Questions used in Interviews

Student Information
Name: _____ **Surname:** _____ **Gender:** ☐ Male ☐ Female **Age:** _____ **Computer Familiarity:** ☐ low ☐ moderate ☐ high

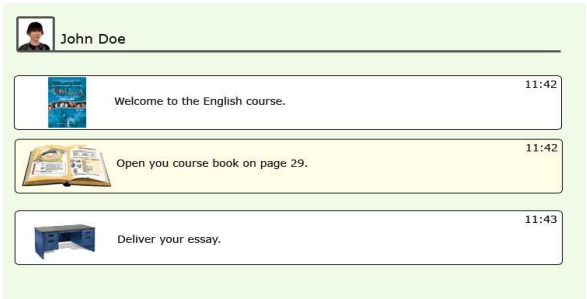
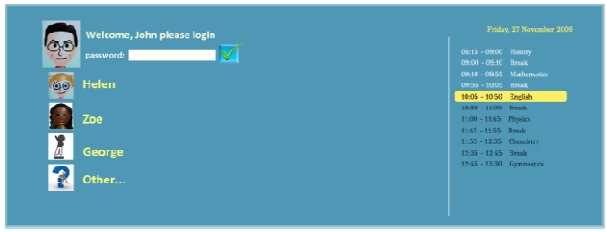
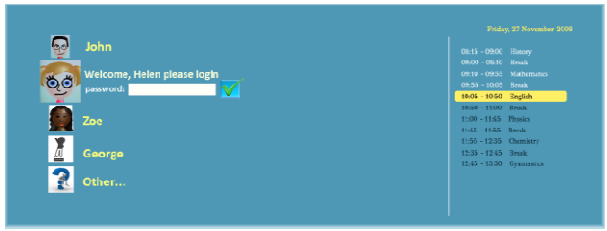
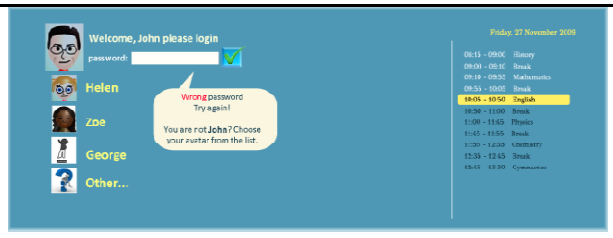
Questions

1. Do you think that a Hint application is useful for the English course exercises?
    a. **Positive answer:** In what types of exercises you would like to get assistance?
    b. **Negative answer:** Find out how students prefer to get help and why they do not like the Hint application
2. What do you think about the system suggesting help if you remain idle?
    a. **In favor:** In what ways would you like the system to suggest help?
    b. **Against:** Investigate intrusiveness of the suggested feature
3. What do you think about the three types of hints provided by the Hint application?
4. Would you prefer all the hints to be displayed at once?
5. Do you think that it is useful that applications adjust to the student's height?
6. What would you like to do with an exercise solved in your netbook?
    a. Send it to your personal area?
    b. Print it?
    c. Submit to the teacher?
    d. Other?
7. Do you think that the Dictionary application is useful for the English course?
8. Do you like that the Dictionary is enhanced with multimedia?
9. Do you think that a History feature would be useful for the Dictionary?
10. What do you think about scaling the pictures with your fingers, using the common zoom in and zoom out gestures?
    a. **Opposed:** What do you feel about touch interaction in general?
11. Do you think that a Multimedia application is useful?
12. What do you think about sharing a video/image with the entire classroom?
    a. **In favor:** Is there anything else (relative to the course) in particular that you would like to share with rest of your classmates?
13. Is there anything in particular that has not been taken into account in the scenarios,

you would like to perform during an English course?

14. Is there anything in particular that has not been taken into account in the scenarios, you would like to perform while studying English at home?

C - Scenarios used in Formative Evaluation

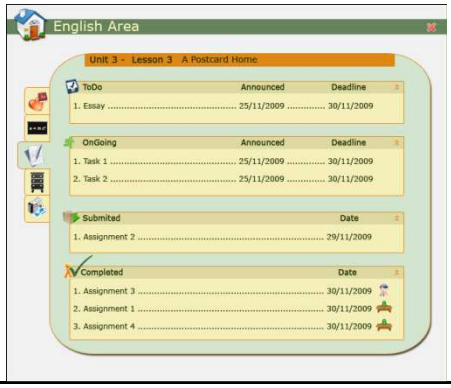
Plot	Screenshots
On Tuesday morning John enters the classroom, leaves his jacket on the hanger and moves towards his desk.	
He sits on his chair and the front screen of his desk welcomes him and asks him to identify himself, by placing his personal school calendar / notebook on his desk.	
John does so, and the current course welcome screen appears on the screen, welcoming him to the English course and prompting him to deliver any homework that was assigned to the class and open his student book at page 29.	
In case a student has forgotten his personal notebook, and since each student usually sits at specific desks*, the password login screen initially includes the possible students' avatars and their names. John selects his avatar and he types his personal password in the related text field.	 
If the password is correct, the current course welcome screen appears on the screen. If the password is wrong, the screen indicates that the password is not correct.	

Each student that logs in to his/her desk is reminded that there is homework pending to deliver. Today's homework for the English course is a 200 words essay.

John has brought his homework on his USB memory stick. He takes his stick from the school bag and places it in his desk's slot. The smart desk automatically recognizes the stick contents located under the mySchool folder and classifies them according to their type and thematic area. Furthermore, it recognizes that some of the files should be submitted for today's homework. Therefore, the desk asks John if it can proceed with submitting the files. John answers affirmatively and the files are submitted to the teacher.

George, on the other hand, has stored his homework in his personal student area. As in John's case, when he logs in the system, his assignment is delivered to the teacher.

He opens his personal area in order to confirm that the assignment has been delivered. Since the current course is English the personal area automatically opens in the English course section. He goes to the assignments' tab and under the current thematic area he sees the assignment marked as submitted. Assignments under a thematic area are classified as: New (they have been initiated by the teacher for the students to fill-in), Ongoing (Students have edited them but they have not been submitted automatically or manually yet), Submitted (homework automatically submitted or class work manually submitted) and Completed (in-class exercises that have been completed or homework that has been corrected and returned by the teacher).

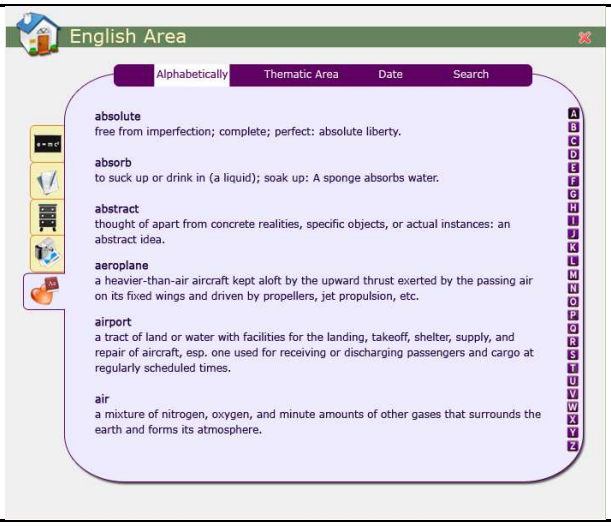


The teacher, Mrs. Brown gets in the class and asks her students to read a passage from the current book page, referring to a theme park in the UK, in order to discuss it later. While reading, Helen comes across the word terrifying. She is not certain about the meaning of this word, so she decides to look for it in the thesaurus.

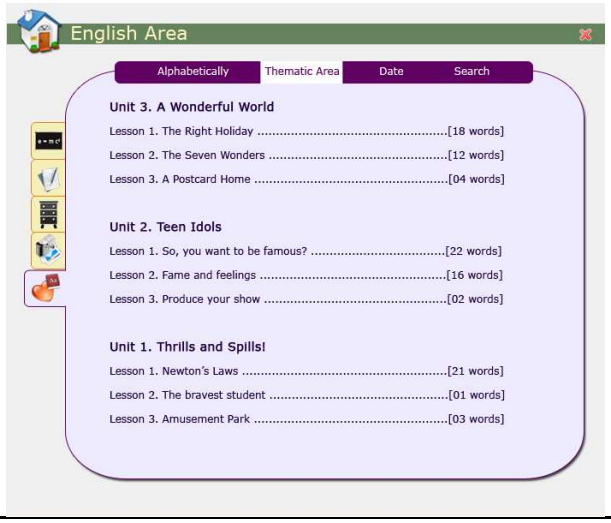
She points at the word with her finger, and the dictionary application appears on her desk's screen, presenting initially a short definition of the word, its pronunciation, a button allowing the student to hear the pronunciation, some synonyms and a few examples of use in a sentence.

In addition, the dictionary offers options for extended descriptions, grammar information, complete list of synonyms, antonyms and several examples. It also allows students to add a word to their personal vocabulary or to view related audio/visual material. Finally, students can select to navigate through the history of the words they have searched in the dictionary (previous /next words).

Helen takes a look and by touching the heart-shaped icon she adds the word to her MyVocabulary application in order to also read it later from home. Her MyVocabulary application can be accessed not only from the school desk, but also from home or the internet café through her personal student area.

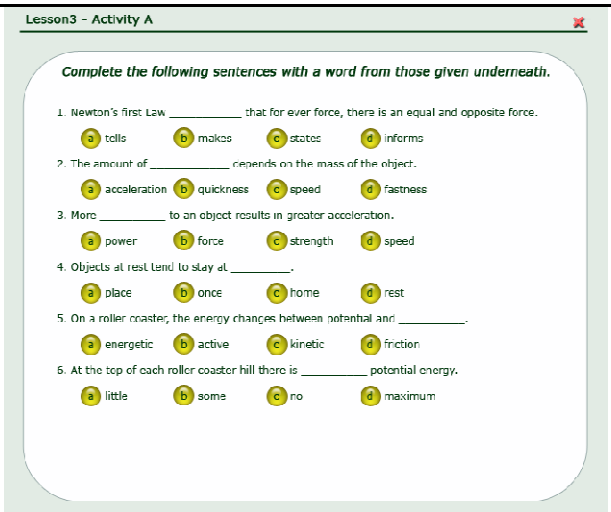


When a word is added to the MyVocabulary application, all the relevant information is added (definition, examples, multimedia, etc.). When the MyVocabulary application is launched all the student's saved words can be browsed alphabetically or by thematic area, according to each student's preferences. Sorting can be inferred by the system or explicitly stated by the student. A word may be found in more than one thematic area. MyVocabulary also provides search facilities and delete – reorganize facilities.



After completing the reading task, Mrs. Brown asks her students to solve a related exercise, in order to see if they have understood the passage. In order to do so, she asks them to turn to page 35, and solve the first exercise, i.e., to fill the gaps in six given sentences, using a word from a set of possible answers provided underneath each sentence.

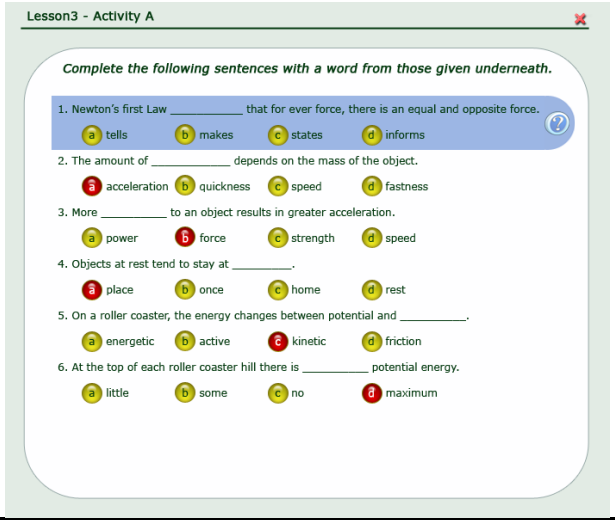
On the desk's screen the exercise is automatically displayed. John will solve the exercise electronically, and so he puts his book away.



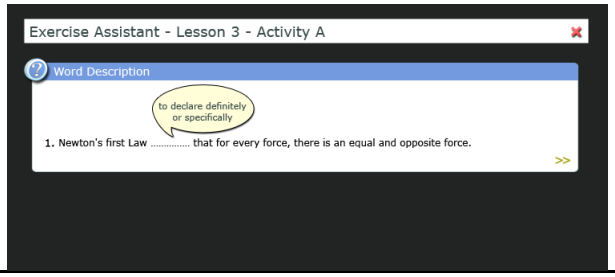
He leaves the first sentence blank at first, and moves on the second one. He touches one of the possible answers and drags it on the gap to be filled in.

He completes all the sentences but the first one. He has been inactive for some time, so the system which is monitoring his performance asks him if he would like some help.

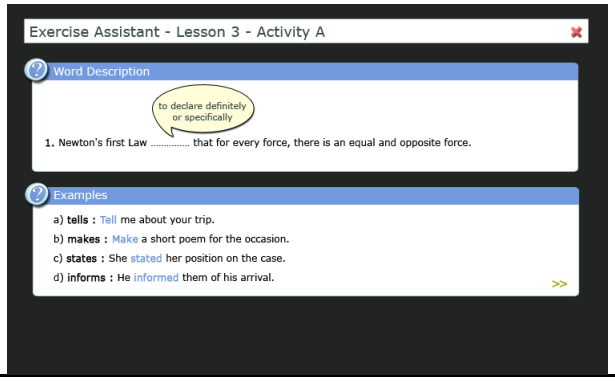
Students can also ask for a hint by themselves, by double-tapping the gap that should be filled-in (either on their actual course book or on the desk screen).



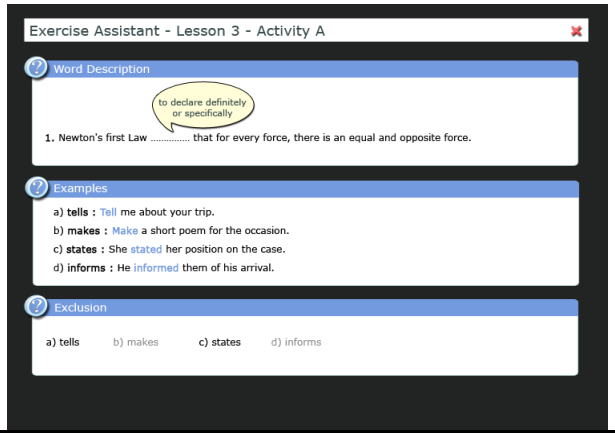
John answers affirmatively and the system provides a hint, by describing the word that should be filled in (*all the hints should be provided according to the context of use*).



The word to fill-in still seems unclear to John, so he selects to see one more hint. This time the system displays examples of the suggested words, as they are used in sentences.

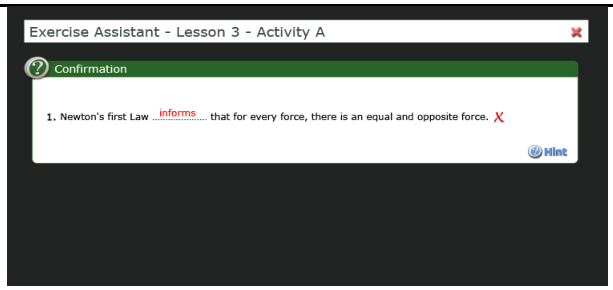


Still John can't decide the word to use, so he asks for one more hint. The last hint is displayed below the other two and excludes two of the four options, allowing the student to select between two possible options.



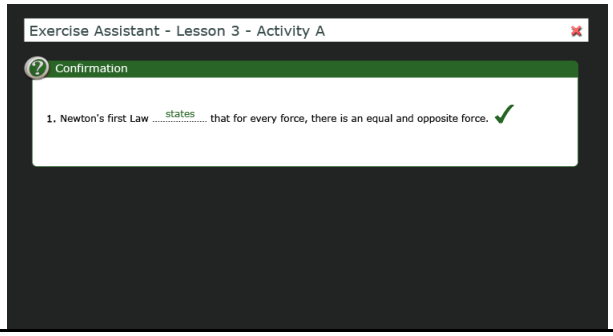
In John's student record the system records that he has used a hint for the specific task, but John knows now how to answer, and completes the exercise. (Hints can be provided gradually). The AmI classroom system records students' activity and keeps logs of the hints that have been requested. Furthermore, it creates a report for the teacher.

In the meantime, Helen has selected to do the exercise on her actual course book. After completing all the gaps, Helen would like to see if she has answered correctly. She touches the filled gap that she would like to confirm and draws a "tick" with her finger (multiple selection of gaps to confirm is also possible). This time, she asks for confirmation for the first sentence. The system indicates that it has been erroneously filled-in and Helen directly asks for a hint.



After getting the first hint, she changes her answer and checks the response again. The confirmation application indicates that this time it has been completed correctly.

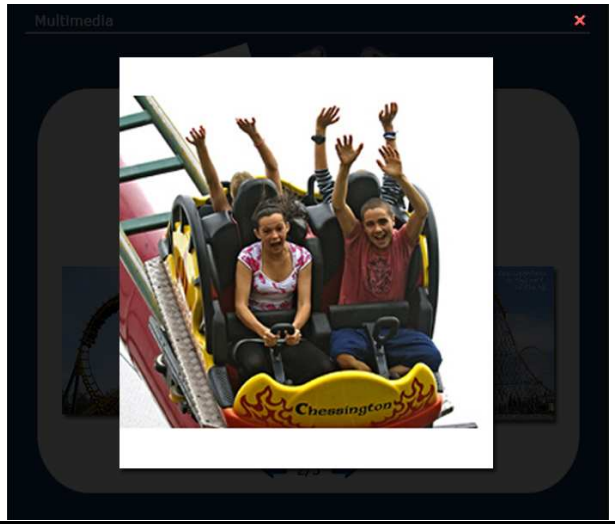
If Helen has left any gaps empty then this is not identified as an error by the confirmation application; errors are indicated separately from blanks.



Once students complete the exercise, Mrs. Brown asks the class why they think young people like roller coaster rides and asks them to provide audio/visual material in order to support their claims.

John is really intrigued by the photo and since he's never been on a roller coaster, he is really eager to seek for related material. He points at the photo with his finger and as a result a collection of related photos, videos and audio appears on the screen of his desk.

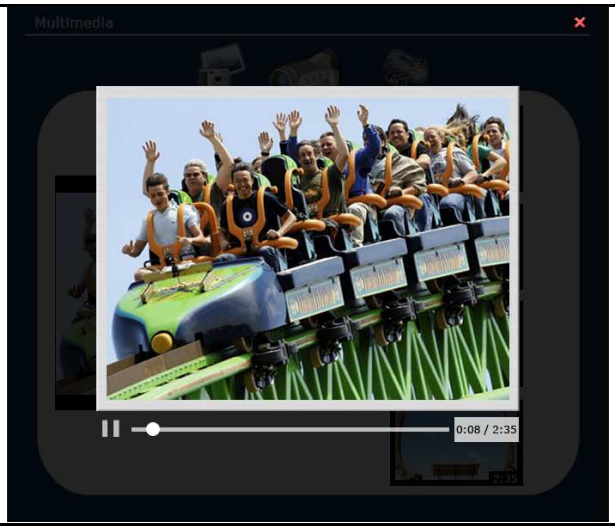
John navigates in the suggested content by dragging his finger on the desk's screen. First of all, he selects to see a picture showing a close shot of a child riding the roller coaster. The child seems quite scared but he would like to zoom in the picture for a closer look. In order to do so, he touches the photo with two fingers and moves them away diagonally.



The photo is zoomed in and since the child's face although scared seems quite funny, John decides to save the photo in his personal student area. He touches the photo and drags it to the "Home" icon of the desk main menu bar on the right of his screen.

The material is automatically saved in his personal student area, under the category "English> Multimedia> Thematic Area".

Then, he closes the picture and returns to the suggested multimedia content screen. A video there seems also interesting; John selects it and touches the play control.



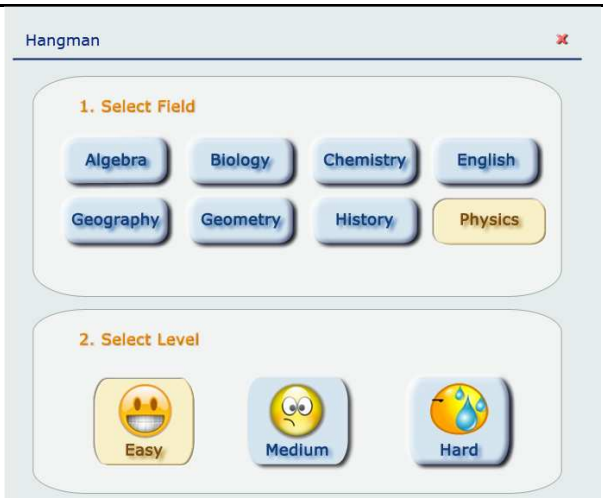
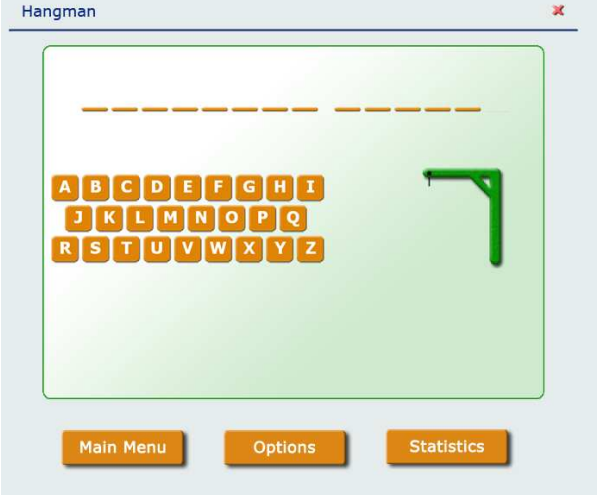
John thinks that this video can answer the question and therefore he saves it to the Clipboard area, in order to use it later to support his claims. (The Clipboard *is a vertical area where students can temporarily hold any information they are interested in to review it later*). He keeps looking however in case he finds something more relevant.

Mrs. Brown asks the kids about the results of their research and John raises his hand and suggests that the video he has found could answer the question of Mrs. Brown.

The teacher allows him to show the video on the class board, and John drags the video from the Clipboard to the board icon located in the main. The video is transferred to the board, John selects the play control and now everyone can watch it.

The course is over, the bell rings and Mrs Brown greets her students and leaves the class. She did not assign them any homework for the next class, so everyone is very happy.

Most of the kids leave the class, however some of them decide to remain inside since it is unusually cold outside today. Helen decides to launch the games application. She selects the “Hangman” game and the welcome screen appears, prompting her to select a thematic area and a difficulty level to play.

	
She selects easy physics, and the first physics-related words appear. She has to fill-in the letters from two words the first consisting of eight letters (magnetic) and the second one of five (field), having ten guesses.	

D - Questionnaires used in Formative Evaluation

System Evaluation Questionnaire

Instructions for answering the questionnaire

Through this questionnaire, we would like to ask you to let us know your opinion regarding the system that was presented to you. Your answers will help us find out what you liked most, as well as what needs to be changed in the system.

This questionnaire includes 12 claims, to which you should indicate whether you agree or not, by circling the answer that best describes your feelings. You may also add extra comments wherever you like to explain your answers or provide additional remarks.

The last two questions ask you to explain what you liked most and what you disliked most about the system.

Thank you for helping us through your participation!

1. **Generally I like the idea of having a "smart" desk.**

Sure! Yes Maybe No No way!

Comments: _____

_____

_____

2. **A "smart" desk would assist most of the students through the learning process.**

Sure! Yes Maybe No No way!

Comments: _____

_____

_____

3. **Pointing a word on my book and then seeing relevant information on the "smart" desk's screen is really useful.**

Sure! Yes Maybe No No way!

Comments: _____

_____

_____

4. **I think that a dictionary application is necessary for studying English.**

Sure! Yes Maybe No No way!

Comments: _____

_____

_____

5. **For foreign language courses, having an application where I can store words for further study, available both at school and at home, is really helpful.**

Sure! Yes Maybe No No way!

Comments: _____

6. **I think that it is useful pointing an image on my book and then being able to see relevant material (images, videos, sounds) on the "smart" desk.**

Sure! Yes Maybe No No way!

Comments:

7. **I like the ability of gradually getting hints while solving an exercise.**

Sure! Yes Maybe No No way!

Comments:

8. **Asking for confirmation about the correctness of a specific exercise is a functionality that I would use.**

Sure! Yes Maybe No No way!

Comments:

9. **In general I liked the applications' appearance.**

Sure! Yes Maybe No No way!

Comments:

10. **I enjoyed the idea of a Clipboard where I could temporarily hold any information,**

in order to review it.

Sure! Yes Maybe No No way!

Comments: _____

_____

_____

11. **I enjoyed the idea of sending course related material to the board or to the teacher, by just dragging it with my finger on the respective icons, located at the right toolbar of the "smart" desk.**

Sure! Yes Maybe No No way!

Comments: _____

_____

_____

12. **In my opinion, it is useful to be able to view two applications at the same time (one next to the other) on the "smart" desk.**

Sure! Yes Maybe No No way!

Comments: _____

_____

_____

13. **The six things that I enjoyed the most were:**

_____

_____

_____

_____

_____

_____

_____

14. The six things that I enjoyed less were: