



KUNGLTEKNISKA HÖGSKOLAN

Royal Institute of Technology
Numerical Analysis and Computing Science

CID-86, KTH, Stockholm, Sweden 1999

Shared Spatial Desktop Development

Kristian Simsarian, Ben Bederson, Pär Hansson, Jean-Paul Hourcade,
Karl-Petter Åkesson, Gustav Taxén, Allison Druin, Victor Banyon, Steve Benford,
Danaë Stanton & Helen Neale



CID
Centre for
User Oriented IT Design

**Kristian Simsarian, Ben Bederson, Pär Hansson, Jean-Paul Hourcade,
Karl-Petter Åkesson, Gustav Taxén, Allison Druin, Victor Banyon, Steve
Benford, Danaë Stanton & Helen Neale**

Shared Spatial Desktop Development

Report number: CID-86

ISSN number: ISSN 1403-073X

Publication date: August 1999

Reports can be ordered from:

CID, Centre for User Oriented IT Design

Nada, Dept. Computing Science

KTH, Royal Institute of Technology

S-100 44 Stockholm, Sweden

telephone: + 46 8 790 91 00

fax: + 46 8 790 90 99

e-mail: cid@nada.kth.se

URL: <http://www.nada.kth.se/cid/>



D1.1: WP1 Deliverable: Shared Spatial Desktop Development

ABSTRACT

This document is part of the deliverable for Work Package 1 of the KidStory project. This document describes the progress of WP1: The shared Desktop. This work centred around building Single Display GroupWare (SDG) to promote ‘shoulder-to-shoulder’ collaborative storytelling for and with children (aged 5, and 7) in schools. Over the course of the first year, infrastructure has been researched, designed, and constructed that supports multiple input and interaction on a “single desktop display.” Two storytelling platforms have been developed, one built on the technique of using 2D+zooming, the other employing a 3D environment. The two platforms are designed to complement each other in their capabilities. We first describe the goals of the project, and follow that discussion with how techniques in the two platforms meet the first year goals, and then present the details of those platforms in separate chapters and conclude with a summary and starting points for the second year of work.

Document ID	WP1deliverable.doc
Type	Deliverable
Status	Complete full version
Version	1.0
Date	990806
Author(s)	Kristian Simsarian, Ben Bederson, Pär Hansson, Juan-Pablo Hourcade, Victor Bayon, Gustav Taxén and Karl-Peter Åkesson, Allison Druin, Steve Benford, Danaë Stanton, Helen Neale
Task	D1.1

Contents

1	INTRODUCTION: SHARED SPATIAL DESKTOP COMPUTER.....	5
1.1	PROJECT GOALS OVERALL	5
1.2	COLLABORATIVE STORYTELLING TOOLS	6
1.3	KIDSTORY WORKPLAN: WP1. DEVELOPMENT OF PLATFORMS AND STORYTELLING APPLICATIONS.....	6
1.3.1	<i>Tasks from WorkPlan</i>	<i>8</i>
1.4	SINGLE DISPLAY GROUPWARE AND SHOULDER-TO-SHOULDER COLLABORATION.....	9
1.4.1	<i>Why Single Display?.....</i>	<i>11</i>
1.4.2	<i>Tradeoffs In Single Display GroupWare.....</i>	<i>11</i>
1.5	TOOLS TO SUPPORT COLLABORATIVE STORYTELLING.....	13
1.5.1	<i>KidPad.....</i>	<i>14</i>
1.5.2	<i>Klump</i>	<i>14</i>
1.6	METHODS FOR ENCOURAGING COLLABORATION.....	14
1.7	PARTICIPATORY DESIGN AND TECHNICAL ITERATION	15
1.8	REFERENCES	16
2	TOOLS FOR COLLABORATIVE STORYTELLING.....	18
2.1	INTRODUCTION TO TOOLS FOR STORYTELLING	18
2.1.1	<i>New media tools without direct tradition</i>	<i>18</i>
2.2	SOME OF THE NATURE OF NARRATIVE	19
2.3	NARRATIVE COMPOSITION	20
2.4	TIME, SPACE AND CAUSALITY.....	22
2.4.1	<i>Causality.....</i>	<i>22</i>
2.4.2	<i>Time</i>	<i>22</i>
2.4.3	<i>Space</i>	<i>23</i>
2.5	BUILDING NARRATIVES FROM THE PERSPECTIVE OF IMPROVISATION THEATRE	24
2.5.1	<i>Offers, Accepting, and Blocking.....</i>	<i>26</i>
2.5.2	<i>Interface and offers.....</i>	<i>26</i>
2.5.3	<i>Recounting forms of Narration: narrator, narratee, implied reader and reader</i>	<i>27</i>
2.6	THE TOOLS FOR COLLABORATIVE EXPLORATION	27
2.7	PROJECT PLATFORM DEVELOPMENT: TWO APPROACHES, KIDPAD-KLUMP.....	28
2.7.1	<i>Hardware Implications of Two Platforms.....</i>	<i>28</i>
2.7.2	<i>Issues influencing technical design and specification.....</i>	<i>28</i>
2.7.3	<i>Toward Storytelling Objects and Reactive Spaces.....</i>	<i>29</i>
2.8	REFERENCES	29
3	KIDPAD - A 2D+ZOOMING COLLABORATIVE STORYTELLING TOOL.....	31
3.1	BACKGROUND.....	31
3.2	DESIGN	32
3.3	LOCAL TOOLS	32
3.3.1	<i>Collaborative Tools.....</i>	<i>34</i>
3.4	ZOOMABLE USER INTERFACES (ZUIs).....	35
3.5	SINGLE DISPLAY GROUPWARE (SDG)	37
3.5.1	<i>The Java MID Package</i>	<i>38</i>
3.5.2	<i>Related Work on Single Display GroupWare.....</i>	<i>39</i>
3.5.3	<i>MID Architecture.....</i>	<i>40</i>
3.5.4	<i>MID general purpose package</i>	<i>41</i>
3.5.5	<i>MID Event sources</i>	<i>41</i>
3.5.6	<i>MID mouse event source</i>	<i>41</i>

3.5.7	<i>Use of MID events vs. Java Events</i>	42
3.5.8	<i>Extending MID</i>	45
3.5.9	<i>MID Conclusion</i>	45
3.6	PARTICIPATORY DESIGN AND ITERATION	46
3.7	DESIGN SUGGESTIONS FROM ADULTS	48
3.8	ACTIVITY PATTERNS.....	49
3.8.1	<i>Contextual Inquiry Sessions of KidStory Project, Year 1</i>	49
3.8.2	<i>Artefacts</i>	51
3.9	REFERENCES	53
KLUMP – A 3D COLLABORATIVE STORYTELLING TOOL		55
4.1	THE KLUMP APPLICATION AS AN EXAMPLE OF A SHARED DESKTOP STORYTELLING TOOL	55
4.2	GROUNDING CONCEPTS	56
4.3	STORY EXISTENTS	57
4.3.1	<i>Shaping the Klump</i>	57
4.3.2	<i>Morphing</i>	59
4.3.3	<i>Texture Manipulation</i>	59
4.3.4	<i>Texture Selection</i>	59
4.3.5	<i>Moving the Texture</i>	60
4.3.6	<i>Combining Textures</i>	60
4.3.7	<i>Speckling</i>	61
4.3.8	<i>Rubber Stamping</i>	62
4.4	TECHNICAL ISSUES	62
4.5	GRAPHICS AND MECHANICS OF THE KLUMP.....	64
4.5.1	<i>Effective Distribution and Mapping of Textures</i>	64
4.6	INTERACTION SOUND.....	66
4.7	COLLABORATIVE DEVICE ISSUES	68
4.8	KLUDDING – A DRAWING TOOL AS A FORM OF MIDI LIGATURE	69
4.9	STRUCTURE AND EVENTS	70
4.9.1	<i>Theatre</i>	71
4.9.2	<i>Character Behaviours</i>	71
4.9.3	<i>Theatre Wheel</i>	73
4.9.4	<i>StorySphere</i>	74
4.10	PARTICIPATORY DESIGN AND ITERATION	76
4.11	CONCLUSION.....	77
4.12	REFERENCES	78
5	DESIGNING STORYTELLING TECHNOLOGIES TO ENCOURAGE COLLABORATION BETWEEN YOUNG CHILDREN	79
5.1	INTRODUCTION	79
5.2	THE INITIAL VERSIONS OF KIDPAD AND THE KLUMP.....	80
5.2.1	<i>KidPad</i>	80
5.2.2	<i>The Klump</i>	82
5.3	INTERFACES TO ENCOURAGE COLLABORATION	83
5.3.1	<i>Relationship to previous work on shared interfaces</i>	84
5.4	REDESIGNING KIDPAD AND THE KLUMP TO ENCOURAGE COLLABORATION	86
5.4.1	<i>Redesign of KidPad</i>	86
5.4.2	<i>Redesign of the Klump</i>	87
5.4.3	<i>Initial reflections on the revised interfaces</i>	88
5.5	SUMMARY AND FUTURE WORK.....	91

5.6	REFERENCES.....	91
6	CONCLUSION AND FUTURE DIRECTIONS.....	93
6.1	SUMMARY OF TOOL THEMES.....	93
6.2	TOWARD WP1.2	94
7	APPENDIX A –RESEARCH PROJECTS AND COMMERCIAL PRODUCTS RELATED TO KIDSTORY.....	95
7.1	INTRODUCTION	95
7.2	PROJECT REVIEW	95
7.3	PUPPET – I3.....	95
7.4	TODAY'S STORIES –I3	96
7.5	POGO –I3	96
7.6	THE VIRTUAL THEATRE PROJECT	97
7.7	THE NICE PROJECT.....	97
7.8	THE PLAYGROUND PROJECT –I3.....	98
7.9	COMMERCIAL PRODUCTS RELATED TO KIDSTORY	99
7.10	PUPPETTIME.....	99
7.11	ZOWIE	99
7.12	ORLY'S DRAW-A-STORY CD-ROM.....	100
7.13	LEGO MINDSTORMS.....	100
7.14	REFERENCES	101

1 Introduction: Shared Spatial Desktop Computer

This document serves as the deliverable description of workpackage 1.1 activities performed in year one. The document consists of six chapters. This first chapter reviews the overall technical goals of the project, and introduces the notion of Single Display GroupWare as well as the storytelling platforms used in the project. The second chapter explores the concept of storytelling tools more generally as well as details the concepts and definitions of storytelling as they pertain to this project. Chapter 3 describes the development work carried out with KidPad as a shared collaborative storytelling tool. Chapter 4 describes our progress with the Klump application as a storytelling tool. Chapter 5 explores the notion of *encouraging* collaboration in a single display GroupWare environment. Chapter 6 presents a summary and a review of proposed work going into the second year of development.

1.1 Project goals Overall

The KidStory project involves three phases of technological development executed over the three year course of the project. In each phase, a shared storytelling platform and associated applications are constructed for integration into a participatory design cycle carried out in the school environments. Each phase builds on previous phases and extends the interface further away from traditional computer hardware towards more kid-friendly and inherently collaborative forms of interaction. We first review these three phases and then concentrate on the year one activity, “Shared Spatial Desktop Computer”:

- **SHARED SPATIAL DESKTOP COMPUTER:** This is the first step beyond the current computer interface. Development of existing software and hardware platforms is extended to support shared access through multiple input devices. These developments enable simultaneous child-users to employ multiple input devices and thus share control over story creation on a single computer in the classroom. This phase builds upon previous experience with two kinds of interface approaches: 2-D zoomable interfaces and 3-D virtual environments.
- **SHARED STORYTELLING OBJECTS:** In this phase, we will extend our focus to include the creation of sharable “storytelling objects” to be used in the creation of stories. We consider both physical and virtual storytelling objects. Physical storytelling objects include tangible interface objects such the ‘stuffed toolbox,’ a physically manipulable and kid-friendly interface to a computer. Virtual storytelling objects include characters in a virtual environment that can be under computer control, and can be controlled directly by children as if they were puppets or can even be inhabited as self-representations for role-play.
- **SHARED AUGMENTED SPACE:** In the third and final phase, we extend our focus beyond shared objects to consider the physical space within which these objects are located. We will design and construct different kinds of room-size reactive space within which both physical and virtual storytelling objects can be located. Multiple children will be able to explore these physical spaces and interact with them through gesture, movement or other forms of “less-encumbered” interaction.

The main focus of this report is the development of the “Shared spatial desktop computer.” The goal of that development is to create Single Display GroupWare (SDG) which in turn can support Shoulder-to-Shoulder collaboration. One key element of this is the incorporation of an architecture for supporting multiple input devices on a single computer. Another key element is the development of sharable tools for storytelling. The development of the shared desktop has proceeded while keeping the needs of the other phases in focus. An example of where this is made explicit is in the development of the “multiple input device” (MID) architecture, which we expect to be extensible for work in the second and third phases of the project.

1.2 Collaborative Storytelling tools

One aim of the project is to introduce new dimensions to collaborative storytelling that could not be easily achieved with either conventional computers or with traditional modelling materials (e.g., paper drawings and models). In this sense, our goals are to augment current in-school practices rather than replace them. These dimensions for year one include:

- Experimentation with different forms of shared control, where one child controls some degrees of freedom of interaction and another child controls other degrees of freedom, and where they encouraged to collaborate in order to tell a story. Forms of this we call “subjective interaction.”
- Experimentation with new interaction paradigms, especially “zooming”. We argue that zooming offers users a visual way to connect information that includes context. When you move through stories you see where you've been or where you're going, as opposed to hopping from one picture to another as in a ‘slide show’ (as traditional multimedia/hyperlinking does).
- Experimentation with dimensionality of the visual representation in storytelling. For example, what affordances are there in storytelling tools in a 2-D+zooming environment, or in a 3-D environment.
- Experimentation with novel and traditional storytelling structures and environments that could not be created with traditional materials (e.g. large virtual spaces that embody notions of magic, physics and so-forth), and with new metaphors for creating and sharing stories.

Here it should be stressed that the intent of the introduction of these tools is not to replace traditional materials. Indeed, a key aspect of our work on storytelling objects and augmented spaces is to consider how the ubiquitous and tangible nature of traditional materials can be integrated with the power of computer animation and control.

1.3 KidStory Workplan: WP1. Development of platforms and storytelling applications

First we explore the details of the technical first year KidStory workpackage (WP 1.1) and use that as a basis for the description of first year accomplishments.

KidStory workpackage WP1.1 involves three partner sites carrying out activities on two platforms. The three partners are: The University of Nottingham (NOTT) in the UK; The Royal Institute of Technology (KTH) in Sweden; and the Swedish Institute of Computer Science (SICS), also in Sweden.

The first year technological development plan was:

Extending current desktop interfaces for sharing

This workpackage takes as a starting point access to a conventional computer. It then considers how such a computer can be extended to support sharing in creating and experiencing stories and support the notion of “shoulder-to-shoulder” collaboration. The work extends the existing software platforms that are available and makes creative use of multiple peripheral devices and considers how these can be configured for different kinds of sharing.

Task 1.1 develops a shared environment based on current desktop computer technology, which is extended with multiple user inputs and screens. General concepts include the development and incorporation of techniques to support multi-user collaboration including infrastructure and functionality development. An example of infrastructure development is extending platforms to enable co-located users to share a single desktop machine. An example of functionality is to incorporate support for subjective interaction and mechanisms that encourage collaboration.

SICS is the workpackage leader and the development in the technical workpackage has occurred within the three sites: SICS, KTH, and Nottingham. This workpackage builds on the past experience (technical and conceptual) of SICS/KTH in the Pad++ [Bederson95] and KidPad [Druin97] desktop environments; and SICS, KTH and Nottingham in collaborative virtual environments [Benford97, Greenhalgh 97, Hagsand93, Simsarian96, Simsarian97] and educational applications of virtual environments [Brown97, Neale99, Stanton98].

The following table, taken from the KidStory work-program, summarises the tasks defined for this workpackage. Included are task title, description, partners, start month, end month and resulting deliverables.

1.3.1 Tasks from WorkPlan

T 1.1	Extending existing desktop interfaces and user platforms for sharing	SICS (14 person months - responsible), KTH (6 p.m.), Nottingham (12 p.m.)	Start: 1	End: 12
Description: <p>Within this task, prototypes will be developed, and platforms extended, in close co-operation with children and practising educators. The prototypes will be built upon existing platforms (e.g. Pad++ and DIVE) and will then be carried into the subsequent workpackages (2 and 3) for integration within school environments and evaluation. The main concentration in task1.1 is to extend current computer interfaces to support co-operation through the use of multiple input devices and/or screen. For example, extending the KidPad 2D zooming interface or existing 3D virtual learning environments [Brown97] to support sharing by multiple co-located children.</p> <p>SICS and KTH will develop the KidPad platform into a shared desktop storytelling environment. This will include the extension of KidPad to support multiple simultaneous users, sharable creation tools as well as multiple input devices with extensions to functionality as appropriate from past experience and through participatory design. In addition, this will involve the porting of KidPad from Unix to Windows. The development will be in close contact with children and professionals as a co-operative design process.</p> <p>SICS will also extend its work with shared co-located input devices to spatial environments. Such devices will be made to support multiple simultaneous co-located virtual environment users. In addition, support for concepts in KidPad, such as state-preserving creation tools (such as for drawing and authoring) will be built into existing tools such as the DIVE multi-user virtual environment system. We see these are a further extension of the concept of 'subjective views' where there is a need to define the concept of 'subjective interaction.'</p> <p>Nottingham will extend current single-user interfaces to virtual environments, e.g., the use of a standard PC to access a virtual world, to consider how multiple participants might share a common storytelling environment. This will involve exploring the use of multiple and different input devices to control interaction with a shared virtual world as well as considering how kids navigate and interact with objects within the world.</p>				
Deliverable 1.1: Demonstrator of shared desktop for shared storytelling				

Over year one, the two applications, KidPad and Klump, have been developed and expanded to incorporate the primary notions of the “Shared Spatial Desktop Computer.” Through the method of Co-operative Inquiry, developers, researchers, children and teachers have been working in schools together as partners to develop the methods and mechanisms for supporting “shoulder-to-shoulder” co-operation through collaborative storytelling tools. Both Platforms, KidPad and Klump (KidPad is described in Chapter 3 and Klump is described in Chapter 4) have been extended to support multiple inputs and shared display. In addition, both platforms work on mainstream consumer operating systems, Microsoft Windows (both NT and 98). Extensions have been made to both storytelling platforms that allow for story object creation, story structure specification and for sound generation in the Klump application. Along with the notion of the shared spatial desktop computer comes the of subjective interaction and local tools, both of which are described in detail with other issues in the application chapters. Of primary importance in KidPad is the development of the MID architecture to support multiple input devices. Also of importance have been methods for the encouragement of shared user-focus and collaboration. That is to say mechanisms that encourage shoulder-to-shoulder partners to work together, but not require it (this is covered in Chapter 5).

In the next section we explore and define the notion of Single Display GroupWare further.

1.4 Single Display GroupWare and Shoulder-to-shoulder collaboration

Most computer applications written today are single user applications – they have no special support for multiple users. In contrast, GroupWare applications are group aware, they have a fundamental knowledge of multiple users. SDG is a subset of GroupWare that focuses on co-present collaboration: multiple users at the same time and place.

Traditional GroupWare systems create applications that are intended to be run on multiple workstations and can communicate with one another across a computer network. They either communicate in a distributed fashion where each database is synchronised, or with a single centralised server. Similar to a single user application (see Figure 1.1), a traditional GroupWare application provides both a single input channel and a single output channel for each user.

In contrast, SDG applications provide an input channel for each user through the use of a separate input device, but each must share the single output channel (see Figure 1.2). These are the qualities that give SDG applications their unique character: the combination of multiple independent input channels together with a single shared output channel. There have been traditional GroupWare systems which chose to use a shared user interface, or coupled navigation, but the conclusions were that doing so limited the functionality of the application for no apparent gain when the users were remote [cockburn96 , shu92].

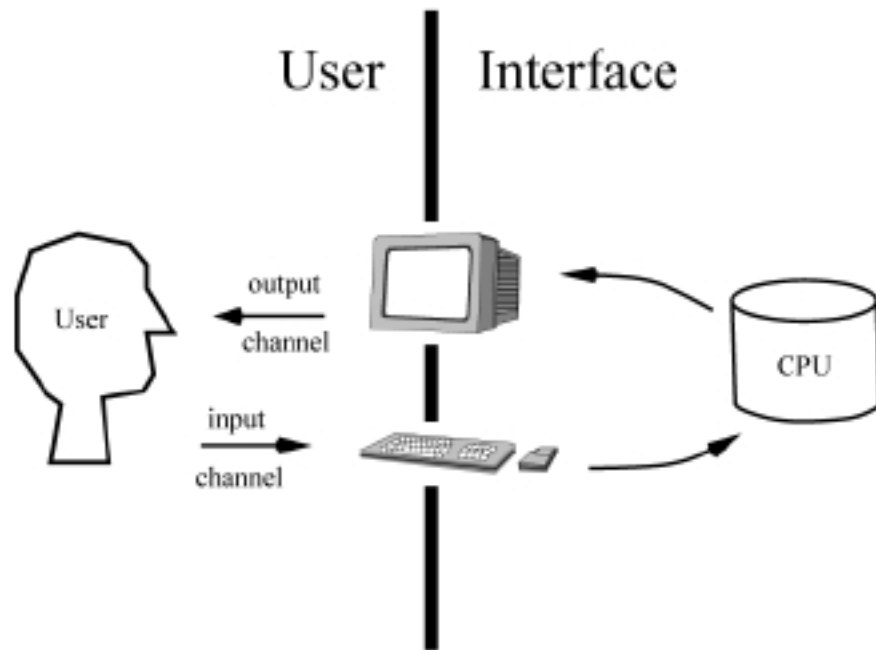


Figure 1.1: The User Interface for Single User Applications

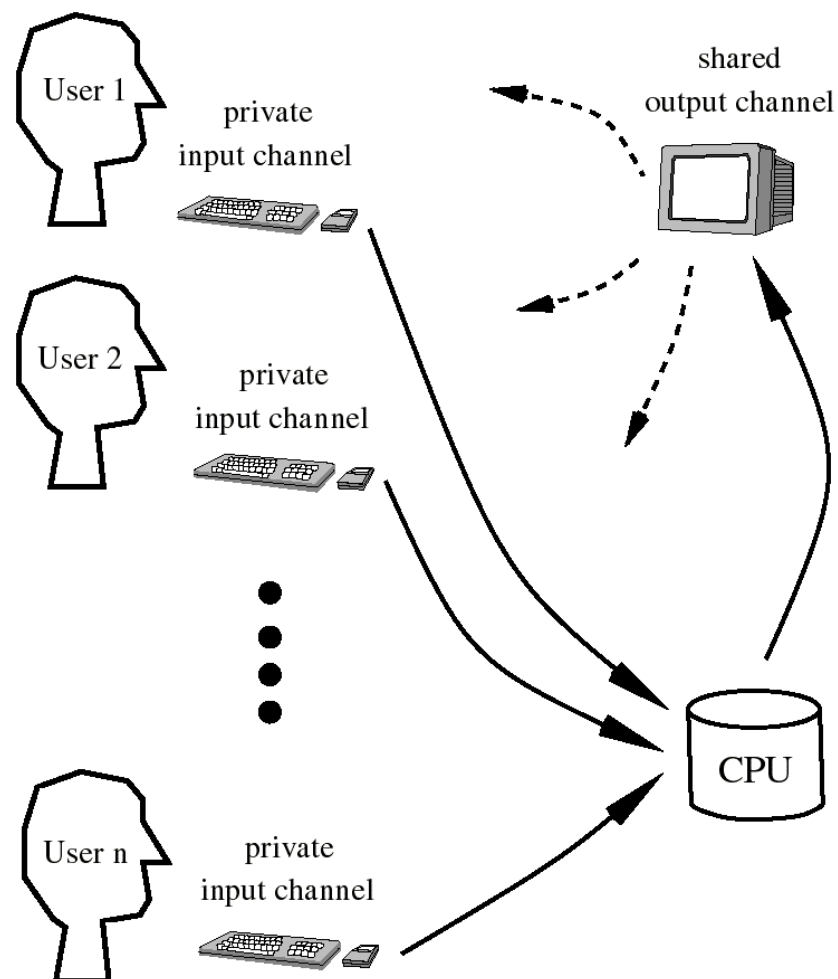


Figure 1.2: Single Display GroupWare

Some of the central differences between designing SDG and traditional GroupWare systems are:

Shared User Interface. Even though users have separate input devices, the user interface elements that are used to communicate with the computer (menus, palettes, buttons, etc.) must be designed to handle multiple simultaneous users.

Shared Feedback. The user interface elements used by the computer to communicate information to users (buttons, palettes, etc.) will likewise be shared by all users and must be capable of relaying information to all users simultaneously.

Coupled Navigation. Whenever one user navigates to a different part of the Model the other users will be affected. If the coupling is tight, then all users will navigate together when one navigates. If the coupling is loose, then other users may have part of their Views obscured by one user navigating to a different area of the Model.

1.4.1 Why Single Display?

We could have chosen to expand the scope of this model to include multiple output devices, and called it Co-Present GroupWare (CPG). The goal of this work, however, was to study the architectural concerns that arise while supporting multi-user collaboration around a single Personal Computer (PC). The overwhelming majority of current PC systems use a display as the main output channel by which to communicate with users. Some feedback is given using an audio channel, but almost never are touch, taste, or smell used [Buxton94, ishii97]. When users collaborate around a single computer, they consider themselves to be collaborating around the display and not the CPU, hard drive, or CD-ROM drive. For these reasons we chose the single shared display as the central metaphor for this new paradigm.

The single display metaphor is intended to connote several properties of applications that are designed for co-present use. Not only do such GroupWare systems have shared data, they also possess a shared UI and shared or coupled navigation. What constitutes a single display? If a single computer has multiple displays, does that mean it is not using SDG? What about full wall projection devices that use three projectors to create a single seamless display? What constitutes a display? A blind person may use a computer whose only feedback is sound, is SDG therefore not for blind people?

Co-Present GroupWare is a more general form of SDG, but since the majority of computers rely almost solely on a visual display for output, we decided that what we lost in generality, we gained back in concreteness. Therefore, we will not include examples which relax the strict conditions imposed by having multiple co-present users at a single display. For example, by using a two-monitor computer each user could be given their own UI, and the shared user interface restriction no longer applies. However, if the use of the second monitor is solely to provide extra physical screen space and not to provide an independent UI, then the conditions still apply and the system could still be considered SDG. Likewise the concept of SDG can be extended beyond purely visual interaction to other modalities as long as the “co-present” and “simultaneous use” conditions apply.

1.4.2 Tradeoffs In Single Display GroupWare

Current computer systems do little to encourage collaboration of multiple users. Single user systems provide only one explicit input channel for all users, so if multiple users attempt to collaborate using such a system it is up to the users to develop a sharing mechanism for utilising that channel. In contrast, SDG applications will have an inherent notion of multiple co-present users and will provide each user with an equivalent input channel. This could have an impact on many aspects of using computers together. Some possible benefits are:

Enabling collaboration that was previously inhibited by social barriers. For example, in many cultures there is often a reluctance to invade the personal space of another person. The personal space surrounding close friends is smaller than that surrounding co-workers and acquaintances, and the space surrounding strangers is the largest of the three [Hall68] . Due to these proximate effects, many people may be inhibited from attempting to share a computer when another person is sitting in front of it. By explicitly providing for a separate input channel, the personal space around the person may be decreased enough to allow another person to comfortably interact with the computer at the same time. Enabling types of interaction that require multiple users. Bricker has explored a number of collaborative interactions that require multiple simultaneous users at a single computer. The goal of her research was to create tools that would strengthen collaborative learning [Bricker98].

Enriching existing collaboration at a computer. For example, turn taking is often viewed as unnecessary and cumbersome [shu92]. Enabling multiple input devices will in some cases enable work to be done in parallel, making the collaboration both more efficient and more enjoyable in the eyes of the users [druin97, Stewart98]. Also, a number of studies have indicated the benefit of shoulder-to-shoulder collaboration due to the collaborators enhanced verbal and non-verbal communications [9, pp. 108–111, smith89].

Reducing or eliminating conflict when multiple users attempt to interact with a single application. Often it is difficult to create an appropriate sharing mechanism for the shared channels, or it is difficult to obey the mechanism created [Stewart98] . By providing separate channels, potential conflicts are pushed one step further away, encouraging peer-learning and peer-teaching. When existing single user technology is used in a collaborative learning setting, the competition between users to interact with the application can inhibit the learning benefits of collaboration [Stewart98] . By providing applications with multiple communication channels, it is possible to enrich learning by diminishing competition for access to the input channels [pappert96, p. 89].

Strengthening communication skills. Because strong willed users can no longer monopolise a task by merely controlling the input device, users may have to communicate more with each other to resolve conflicts. Further information regarding the philosophy of ‘Children as Collaborators’ can be found in WP2.

Along with the potential benefits of the new computer paradigm comes the potential for negative effects:

New conflicts and frustrations may arise between users when they attempt simultaneous incompatible actions. Working in parallel can be an advantage, but it could also be a disadvantage if

each user has conflicting agendas. One serious concern in this area is navigation. Since there is only a single shared output channel (the display), if one user decides to navigate elsewhere in the data space, it may negatively affect the other users. SDG applications must squeeze functionality into a very limited screen space, which may result in reduced functionality compared with similar single-user programs. Due to increased processing requirements, SDG applications might be slower than a single user version, or a traditional GroupWare version. Because successful SDG implementation depends on low-level operating system and windowing system issues, applications may not be very portable and might exist for only the most popular operating systems. Completing tasks might take more time, because it is no longer possible for a strong willed user to direct the collaboration by controlling the input device. Users may actually collaborate less. Because they can do work in parallel, they may set about completing their own tasks and never communicate with the other users. Some of these issues are addressed in the chapter in this deliverable on encouraging collaboration between shoulder-to-shoulder partners.

1.5 Tools to support collaborative storytelling

KidStory is constructing tools to support collaborative storytelling activities. Our design model is to invite children to be our design partners in an iterative design process that occurs within the school context. Research that informed the creation of the project demonstrated that children in groups tend to be collaborative even when ample equipment is available. However current consumer-style (e.g. PC) platforms do not readily support this. The KidStory response has been to augment the current hardware model with multiple input devices while at the same time, moving away from interaction in the desktop "fishbowl." The project acknowledges that spatial navigation (e.g. in virtual worlds) is problematic and instead offers solutions away from traditional interaction paradigms and toward real world interfaces, e.g. tangible artefacts. Planned solutions employ real-world interaction devices (plush toys) and real world display (reactive spaces, augmented environments) as an answer to spatial navigation and interaction problems.

The narrative focus in KidStory is to go beyond the scripted interactive book-style narrative and allow children to create shared stories and storytelling experiences together. We are building two storytelling platforms with a strategy that the two approaches complement one another, both in their approach to narrative and in their mode of human computer interaction. The first is a zoomable desktop drawing program with a tools-based interaction metaphor and a 'scene-based' narrative presentation (KidPad). The second is a 3D shared virtual environment which is based on gesture and mode interaction and is inspired by more improvisational forms of storytelling (Klump).

We believe the tools complement each other well. KidPad is based primarily on a drawing metaphor while Klump is based on a modelling metaphor. KidPad employs virtual crayons, erasers, and text to create narrative existents; it has zooming as a basic function to supply the narrative structure. Klump, on the other hand, has 3D objects as its base and employs other methods, e.g. Spherical storage objects, to supply structure. Although the platforms are developed rather independently, there is a synergy between their development. Both are co-ordinated by SICS and continuous discussions occur between developers and expected code-sharing is planned. The platforms provide differing affordances for the purpose of collaborative storytelling in the SDG

framework.

1.5.1 KidPad

The first platform, KidPad, provides a 2D zooming interface that enables child users to draw on a canvas, zoom in space, and link locations. The drawing tools, crayons and an eraser, enable the creation of the objects, settings, and characters of the stories. Another tool, the magic wand, enables different locations in space and scale to be linked, creating the story structure. Zooming is a fundamental part of the interface and data structure of the system, rather than simply a “feature.” The zooming and the ability to connect links between portions of the screen in multi-scale space are what makes KidPad an interesting tool for storytelling. The multiple input device architecture that KidPad employs enables children to collaboratively author stories as partners on the scale space drawing pad.

1.5.2 Klump

The second platform, Klump, is more experimental and is based on the DIVE (Distributed interactive virtual Environments) system. This system enables the creation of 3D objects within the context of collaborative virtual environments. We are working on methods and mechanisms that promote collaborative exploration and creative play and the creation of novel methods for providing time structuring within the 3D environment (e.g. cinematic, theatre, or other form of spatial and temporal linking). This includes 3D objects that provide intuitive and everyday affordances for story creation and retelling. Some of the storytelling inspirations for these objects and mechanism come from 'story quilts', puppetry, campfires, etc.

The 3D blob object, or Klump, is used as piece of mouldable media (clay) to be shaped, mainly by stretching and pushing, to form the objects (existents) of the narratives¹. The Klump object had its origins in the first discussions of KidStory as a project proposal (the first prototype in September 1997). From there it became the “blob” where it has been used in the eSCAPE project. As the Klump in KidStory it takes a role as an object for creating story existents, characters, settings. The moulding metaphor can be extended in WP 1.2 with tangible, squeezable interfaces for Klump manipulation.

The two systems are being developed in parallel and benefit from 'cross-pollination' and iterative development with our child and teacher partners in the schools. Both storytelling platforms are described in detail in later chapters. Chapter 3 describes KidPad and Chapter 4 describes the Klump.

1.6 Methods for encouraging Collaboration

We make a distinction between “enabling” collaboration and “encouraging” collaboration. Simply put, the interest is in developing mechanisms that encourage users to work together. As promoting

¹ The name Klump is something that works well in Swedish and English, the two school locations. It is intended to be a play on the words Clay lump – while also making explicit the metaphor of working with clay.

communication and collaboration are some of the main aims of the project, we are researching ways to promote this with the tools. The difference between encouraging and similar concepts such as enabling, or requiring collaboration, is that the child users, as free agents can choose to collaborate under the framework of tools that “encourage” collaboration. In the ideal case, these tools encourage child users to work together because there is a benefit to doing so. Even though they might be able to perform the same or similar functionality by themselves, there is a clear advantage of working together. Chapter 5 in this report explores this notion in more depth.

1.7 Participatory design and technical iteration

The work in WP1 and WP2 are closely intertwined. The work in the schools feeds back in a number of ways into the technical development. For more on these pathways and the overall relationship between the children, teachers, developers and researchers the reader is referred to the WP2 and WP3 deliverables. Where appropriate, specific recommendations of improvements and other changes (features, bugs, methods) that have come from the partnerships with the schools are mentioned in the KidPad and Klump chapters.

Over time the Participatory Design process adapts to the needs of the project. The project started slowly with technical work in the schools because the platforms were not available for schools use (both were partly ported from Unix to Windows based operating systems). In the second half of the year the process of working in the schools with the technology became more frequent. The pathways by which technologists, educational researchers, teachers and children communicate have adapted and changed as appropriate. For a further discussion of this methodology and philosophy, please see deliverable D2.1.

1.8 References

- [Bederson96] Bederson, B., Hollan, J., Perlin, K., Meyer, J., Bacon, D., and Furnas, G. 1996. Pad++: A Zoomable Graphical Sketchpad for Exploring Alternate Interface Physics: Advances in the Pad++ Zoomable Graphics Widget, *Journal of Visual Languages and Computing*, 7, pp. 3-31.
- [Benford97] Benford, S. D., Greenhalgh, C. M. and Lloyd, D., Crowded Collaborative Virtual Environments, *Proc. 1997 ACM Conference on Human Factors in Computing Systems (CHI'97)*, Atlanta, March 1997.
- [Bricker98] Bricker, L. J. (1998). Collaboratively Controlled Objects in Support of Collaboration. Doctoral dissertation, University of Washington, Seattle, Washington.
- [Brown97] Brown, 1997, Brown, D. J., Kerr, S. and Wilson, J. R., Virtual Environments in Special Needs Education, *Communications of the ACM*, 40 (8), 72-75, ACM Press, 1997.
- [Buxton94] Buxton, W. (1994). The Three Mirrors of Interaction: A Holistic Approach to User Interfaces. L. MacDonald, & J. Vince (eds.), *Interacting With Virtual Environments*. New York: Wiley.
- [Cockburn96] Cockburn, A., & Greenberg, S. (1996). Children's Collaboration Styles in a Newtonian Microworld. In *Proceedings of Extended Abstracts of Human Factors in Computing Systems (CHI 96)* ACM Press, pp. 181-182.
- [Druin97] Druin, A., Stewart, J., Proft, D., Bederson, B., Hollan, J. 1997. "KidPad: A Design Collaboration Between Children, Technologists, and Educators". *Proc of ACM CHI'97*, ACM Press, pp. 463-470
- [Greenhalgh98] Greenhalgh, C. M., and Benford, S. D., Supporting Rich and Dynamic Communication in Large Scale Collaborative Virtual Environments, *Presence*, MIT Press (in press 1998).
- [Hall68] Hall, E. (1968). *The Hidden Dimension*. Anchor.
- [Hagsand93] Carlsson, Christer and Hagsand, Olof, *DIVE-- A Platform for Multi-User Virtual Environments*, *Computers and Graphics*, 1993, vol. 17(6).
- [Ishii97] Ishii, H., & Ullmer, B. (1997). Tangible Bits: Towards Seamless Interfaces Between People, Bits and Atoms. In *Proceedings of Human Factors in Computing Systems (CHI 97)* ACM Press, pp. 234-241.
- [Neale99] Neale, H., Brown, D. J., Cobb, S. V. and Wilson, J. R., Structured Evaluation of Virtual Environments for Special Needs Education, *Presence: Teleoperators and Virtual Environments*, MIT Press., 8, 3, 264-282
- [Pappert96] Pappert, S. (1996). *The Connected Family: Bridging the Digital Generation Gap*. Longstreet Press.
- [Shu92] Shu, L., & Flowers, W. (1992). Groupware Experiences in Three-Dimensional Computer-Aided Design. In *Proceedings of Computer Supported Collaborative Work (CSCW 92)* ACM Press,

pp. 179-186.

[Simsarian96] Simsarian, K. et al. "Achieving Virtual Presence with a Semi-Autonomous Robot Through a Multi-Reality and Speech Control Interface," Virtual Environments and Scientific Visualization'96, M. Gobel, J. David, P. Slavik, J.J. van Wijk (eds), Springer Computer Science Series, 1996.

[Simsarian97] Simsarian, K. T. and Åkesson K.-P., "Windows on the World: An example of Augmented Virtuality", Montpellier 1997: Interfaces: Man-Machine Interaction.

[smith89] Smith O. Smith, R. B., O'Shea, T., O'Malley, C., Scanlon, E., & Taylor, J. (1989). Preliminary Experiments With a Distributed, Multi-Media, Problem Solving Environment. In Proceedings of First European Conference on Computer Supported Cooperative Work Slough, UK: Computer Sciences House, pp. 19-34.

[Stanton98] Stanton, D., Foreman, N., and Wilson, P. N. (1998) Uses of virtual reality in training: Developing the spatial skills of children with mobility impairments. In G. Riva, B. K. Wiederhold and E. Molinari *Virtual Environments in Clinical Psychology: scientific and technological challenges in advanced patient-therapist interaction*. IOS Press.

[Stewart98] Stewart, J., Raybourn, E., Bederson, B. B., & Druin, A. (1998). When Two Hands Are Better Than One: Enhancing Collaboration Using Single Display Groupware. In Proceedings of Extended Abstracts of Human Factors in Computing Systems (CHI 98) ACM Press, pp. 287-288.

2 Tools for Collaborative Storytelling

2.1 Introduction to Tools for Storytelling

A primary goal in providing computer-based tools to children in the school environment is to support the practice of “collaborative storytelling.” Simply defined collaborative storytelling is the creation of stories by individuals simultaneously working together with a shared notion and focus on the story being constructed. The configuration in which KidStory has set out to pursue this collaborative storytelling is within the framework of Single Display GroupWare (SDG) as outlined in the previous chapter. SDG employs multiple input devices to support “shoulder-to-shoulder” collaboration. In this chapter we explore the underlying concepts of “storytelling tools.” In the process we present some structural definitions of narrative, how these relate to the tools, the goals we have in providing these tools, the challenges, and point to the application solutions we have developed.

Included in the first year deliverable are two chapters on “narrative and storytelling.” The first is this D1.1 chapter, which focuses on the storytelling and narrative from the perspective of the tools. This chapter starts to look for answers to questions such as “what is the space of storytelling and what is the composition of tools for storytelling?” Thus this chapter takes a functional approach to narrative. The second chapter can be found in deliverable D2.1, Chapter 4, “Philosophy 2: Children as Storytellers.” That D2.1 chapter concentrates on the practice of children as storytellers. How children’s stories may be seen within the greater concept of “narrative,” what has been witnessed in the schools to date, and how we might begin to approach questions of narrative analysis. Both the work in this chapter and that in D2.1 are at a beginning stage. They offer few conclusions but instead demonstrate the beginning of an exploration of the role and meaning of collaborative narrative in the project.

2.1.1 New media tools without direct tradition

Building these tools without an attempt to understand the traditional academic, educational and entertainment traditions that have come before would be irresponsible. However, as we research the background for storytelling tools in new media, we realise that there is no great base of germane academic literature. There have been studies on new media tools for children (e.g. [druin96]), but as yet there is no great body of academic work on computer storytelling tools for children. While the field of storytelling in education has a historical tradition and is extensive (e.g. [mcewan95]), as well as studies of children’s stories [engel95]. The field of computer supported storytelling is nascent. CD-ROMs, for example, are only a decade old. Video games with narrative are but two decades old, and there is little academic analysis of computer storytelling. There are futuristic manifestos and technological visions[murray97] and we can draw on these reflections and the experiences of constructing video games and CD-ROMS and other similar storytelling research projects (see the appendix for a review). However, there have yet been any demonstrations of definitive examples in this field. The dominant model for new media storytelling, CD-ROMs, do offer examples of narrative in new media (Myst, Riven, Doom, Orly’s Draw-a-Story). Many of the

most successful examples however, are “games” and fall into a form of “interactive book” (a set, predefined, story space) and do not offer the user the ability to author. In addition, many non-game examples depend on a direct manipulation (e.g. “point and click”) interaction model that we partly seek to move beyond. Since no particular field suffices, we must then draw on a number of traditional academic fields and ‘break new ground.’

To see why these tools are new, consider what computer storytelling tools might encompass:

- Traditional narrative elements (found in narrative text studies);
- Visual elements (found in film studies);
- Interactive principles found in HCI;
- Collaboration enabling mechanisms, a focus of KidStory.

In looking for a foundation for building storytelling tools, we have thus begun to explore a number of fields. In particular these are narratology, film studies, educational storytelling, child-centred HCI, improvisation theatre, children’s storytelling and children’s collaboration. (For the last two of these the reader is referred to the other two deliverables D2.1 and D3.1.)

This chapter then is the beginning of a literal “coming to terms” with some of the important work that has come before. We want to develop a vocabulary for describing parts of the storytelling tools as well as lay out the functional space (of possibilities) that the tools may cover. For the reader that wants to move ahead to the prototypes developed in the project, much of this chapter can be skipped and later returned to; the chapter is not entirely necessary to understand the core functionalities described in the two tools chapters that follow (chapters 3 and 4).

2.2 Some of the nature of narrative

“any sequence of clauses with at least one temporal conjuncture is a narrative” - (William Labov)

A number of narrative researchers have tried to demonstrate the fundamental nature of narrative [bal97, martin86, genette90, berger96]. For example, storytelling may relate to long-term memory, it may be part of social understandings, it may be a key method of passing along social values and methods, and it may serve as a medium of collective memory. With a respectful nod to narratological analysis, we leave the analysis of stories generated in this project to later phases of the project as appropriate. In this document we concentrate on a structural analysis of narrative and how we can develop a vocabulary that is in turn useful for discussing the construction and use of various components of the software tools. In KidStory, stories are inherently collaborative because the focus is on the shared creation of stories by multiple individuals. For most of the work in the first year we have concentrated on pairs of children creating stories on a desktop computer extended with multiple “mouse” input devices. Thus the narrative focus is on collaborative storytelling. In this configuration it is quite common for children to swap roles, lead, follow where the distinction between authors and audience may be fluid. In supporting shoulder-to-shoulder storytelling within

the SDG framework, stories are then seen as collaborative constructions.²

In this next section, we lay down the structural definition of narrative which we use in order to have the vocabulary to discuss the various parts of the tools and their functions. It is not presented as *the* vocabulary, but does provide a narrative structural decomposition which is consistent with the functional decomposition of the tools we are building.

Having such a vocabulary is important. While most people do have a notion of what a “story” is when “they hear it,” for those that have not specifically studied narratives or narratology having a shared narrative vocabulary is rare. In addition to the vocabulary (the naming of parts of what makes a story), there is the structural definition, the elements that together form the whole of the narrative and experience. When going about the construction of “tools for storytelling” we quickly realised there was a need to lay down a common understanding of what a “story” is: what its vital elements are, and how we could think about its presentation and representation. This quickly leads into a discussion of “narrative composition” and an examination of the different axes of narrative. The field of narratology offers much work in this area that has fed this examination. Whether the particular model of narrative presented here is “correct” from the viewpoint of the field of narratology, is less important to KidStory than whether it offers a shared means of discussing the tools and their functionalities.

Given a vocabulary and some basic understanding of the space of “stories,” we can then return to the business of constructing tools that provide those functionalities. Here we are primarily interested in acquiring a basic understanding of the space of storytelling activities, in order to explore tools to support those activities especially within a collaborative context. We begin by looking at a model of narrative composition and then briefly explore notions of time, space, causality and the roles of author and audience.

2.3 Narrative composition

What is a narrative? This is a term that is used and interpreted widely. For the purposes of developing a common language to discuss how we can construct tools for storytelling, it is necessary to present a definition and give a taxonomy of narrative. One such definition of Narrative can be found in Seymour Chatman’s *Story and Discourse* [chatman78]. Chatman’s definition is certainly not the only attempt to describe narrative³, however this particular description has an intuitive appeal and most importantly offers a powerful vocabulary for discussing the various parts of a system suited to narrative construction.

² For more on this topic as practised and witnessed in schools, please see D2.1, specifically chapter 4, “Philosophy 2: Children as Storytellers.”

³ In fact there seems to be little consensus on the definition of narrative and it is not hard to find a definition that conflicts and is more restrictive than this one. Some might see Chatman’s treatment of narrative as encompassing far too broad a definition of narrative including non-traditional narrative forms. For example, some definitions of narrative would include only text based narratives and consider other forms, e.g. theatre, not a narrative form. Other definitions only consider stories told in the first person as true narratives (see [martin86] for a survey). It is the breadth of Chatman’s *Story and Discourse* definition that has been found to be a large part of its generality and attraction.

The work of KidStory is to develop computer-based collaborative storytelling tools. This narrative vocabulary then allows the different elements that make up a narrative to be separated out into their computer software tool equivalents. The vocabulary should also include definitions that acknowledge pre-literate and non-textual literacy, and forms of narrative that are non-verbal and non-traditional. While the term "stories" seems appropriate for discussing the artefacts produced by students with the tools, a more complex vocabulary must be understood and used to capture the roles of the tools of this project and as well as providing a comparative function in relation to previous projects. This treatment does not attempt to place this narrative definition within some greater academic themes in the narrative debate, e.g. the discourse on structuralist, poststructuralist theories (that discourse is beyond the scope intended here). Nor is any attempt made to try and deduce fundamental claims about the nature or purpose of narrative. Instead, this definition is used as a means of discussing the design of tools that enable the creation of what people generally agree to be narratives. These narrative artefacts which are the social produce of the narrative creation tools will here generally be called *stories* (plural).



Figure 2.1 Narrative Taxonomy adapted from [Chatman78]

Chatman's taxonomy of *narrative* is presented in the figure above. Essentially *narrative* is composed of two major components, the *story* and the *discourse*. The elements of *story* are the building blocks of the narrative, the *events* and the *existents*. The *events* consist of the *actions* and the *happenings* of a *story*. The *existents* consist of the *characters* and *settings* of a *story*. The *events* are the abstract set of plots, subplots, etc and their abstract relations that make the story appear to be a "story". One might understand *events* as the verbs of the *story* and intuitively, "the things that happen." Similarly, the *existents* can be understood as the nouns of the *story*, and intuitively "the people and places where things happen."

The *discourse*, on the other hand, is the unfolding of the story – the way it is presented, how it is related to the 'audience.' This consists of two components, the *structure* and the *manifestation*. The *structure* can principally be understood as the time structure, the order of how the *story events*, in combination with the *existents*, are presented. While a linear approach to storytelling might start at the earliest event and proceed to the last, a more complex telling could involve many time shifts, parallel scenes, etc. A quick reflection on the modern cinematic experience reveals that there are many common examples of flashback, end-first, background, fantasy, etc, scene ordering available in the *discourse* of popular films. The *manifestation* of a *narrative* is the form it takes when it is presented. The *manifestation* might be oral, balletic, cinematic, textual, video-game, multimedia

(CD-ROM) or virtual reality. Thus, the same *story* might have many tellings, or presentations, e.g. varying *discourses*. As well, a story may have many manifestations; examples might be a traditional oral fable that is presented as a ballet, a theatre play, a film, and a CD-ROM video game.

The first level narrative division of story and discourse describes a separation of the concepts of "what the story is about" and "how that story is told". The story exists independent of the audience.⁴ But the discourse, however, must involve an audience. It is hard to conceive of narrative existing at all without the discourse, as soon as becomes concrete it involves a discourse. Nor can the discourse exist without the story and be a narrative. Because a discourse involves an audience (e.g. a recipient), implied or explicit, the narrative, even of itself, can be seen to be a collaborative enterprise.

2.4 Time, Space and Causality

"We can consider narrative to be a chain of events in cause-effect relationship occurring in time and space." [Bordwell96]

2.4.1 Causality

Causation has been described as that element that makes a set of elements or statements a story. A clear or deducible cause-effect relation in the events between story elements, characters, settings, can be said to be a vital element that unmask the narrative from just a collection of statements. This statement is, of course, debatable. However as it seems to make good intuitive sense and has resonance with the Labovian definition of story given earlier we prefer to accept this as part of our definition of narrative. These cause-effect relationships happen over time and space. We introduce below a set of definitions that serve to describe this spatio-temporal narrative space. This collection of definitions are a synthesis of terminology from studies of text-based narrative[prince89] and film-based narrative[bordwell96].

2.4.2 Time

There are a number of different measure and representations of time and space in a given narrative. These differences vary for different media. For new media which encompasses elements of traditional media (text, picture, interaction) we synthesise a new description as appropriate. For example:

- Story time: This is the time internal to the story itself, the time that is experienced by the

⁴ Audience is meant here in an abstract sense. Sometimes the audience only listens and watches, other times the audience interacts, and sometimes the 'audience' may even author.

characters of the story.

- Discourse time: This is the time that is actually communicated in a story.
- Screen time: This is the time over which the story is recounted as the author intended, e.g. the length of a narrative.
- Playback time: This is time over which the story is actually experience by the recipient of the story.

Story time is thus a time away from both the author(s) and the audience. It is a time that is internal to the story itself and may be fictional. Discourse time is likely to be a subset of the story time. Screen time is the length of a recounted narrative (e.g. in analogy to the film length). Playback time is the time in which the narrative is experienced by a reader-recipient (note that this may differ than what the author intended). Time, further, has the qualities of order, duration, and frequency. Temporal order may not necessarily be linear from start to finish. Duration is the related to the length, or span of time represented, and frequency, includes repeated elements that might be revisited in a story.

These terms are related with the intention of communicating the different notions of time encountered in narrative as well as supplying a vocabulary for discussion the different temporal aspects of stories.

For example, the creation of structure with the storytelling tools often sets up the discourse time of the story. The discourse time of the story is usually the temporal sequence where the events of the story are related. The author often has a notion of how the story should be played back, this is the screen time. When the story is played back there may be a completely different time. Playback may differ in order, duration, frequency from the planed screen time. Reflecting on these differences opens up a debate on elements that should be author or recipient control, or some mix.

Is this important for children's narratives? Certainly many narratives the children create contain these elements. Our discussions of screen time and playback time have raised questions on how important playback and control of the story are to the project. Primarily we directed the development of the tools to support the creation of the discourse of the story. The story time is often something implied by the story itself (e.g. "A long time ago"). Examining the different aspects of time in a story also allows us, as storytelling tools developers, to think about what time aspects we should concentrate effort upon. If relating the stories to audiences becomes more important in later phases of the project, we will focus on providing supporting mechanisms for control of screen time and facilities for playback.

There are many other forms of time not addressed here, including some of the purely subjective notions of time that are mainly used in analysis, e.g. sensed time of experience, time remembered, frequency of time replayed. As yet, the analysis in KidStory has not advanced to a stage where these notions of time have become relevant. Elements of these notions of time may later play a role in the observation of collaborative behaviour and skills development.

2.4.3 Space

The full narrative includes elements that are both inside and outside the story itself. That is to say, that in addition to all the elements explicit in a story, there are other elements that are suggested to be part of a story. *Diegesis* is defined as the story world, be it fiction or non-fiction. It is that space that includes all those elements that are given or implied by the story. In addition to all the explicit people, places, things, etc in the story, these might also include what might reasonably be inferred about a story – e.g. the life of a character before the story started. Further there are these aspects of the diegesis and its representation:

1. Story Space – The space (characters, settings, events), explicit or implicit in the diegesis.
2. Discourse space – The space of elements represented in this particular instantiation of the story.
3. Screen space – The physical space of the screen (computer monitor, theatre stage, silver screen) that is used to 'stage' the narrative for the audience.

These terms form a simplified definition of elements of narrative. More complex concepts built upon these definitions have not been addressed and are mostly employed in the academic analysis of narrative. Another area we do not address is the interpretation of narrative⁵.

Note that the screen space is generally a subset of the discourse space which in turn is a subset of the story space. When we consider how we can supply tools for children to create the diegesis of their stories, we focus on the support for tools that create. For the most part, the focus on year one has been tools for the creation of the existents, and structure of the story. The tools for creation of the story existents and structure is treated as implemented in the KidPad and Klump chapters.

Topics that are yet to be addressed in year one are alternative manifestations. Such explorations of manifestations, e.g. displays, physical settings, manipulatives, that are part of the way the story is presented belong more to the activities of year two and year three. The issue of playback will then likely become a topic for exploration. As mentioned the focus up to now has been the collaborative creation of narratives with little emphasis on the playback. Year two in the project can be seen as further steps to explore discourse space and "screen" space as they are generalised to interactive and reactive storytelling spaces. This very topic leads into a brief discussion of narrative from the perspective of improvisational theatre.

2.5 Building Narratives from the perspective of Improvisation theatre

Even though there are many examples of storytelling in culture, the cultural traditions of collaborative storytelling are less clear. In western culture we can look to the modern practice of Improvisation theater, its cultural antecedent Commedia Dell'arte, as well as the variations of street theater, happenings, Mardi Gras. These are arts where the communicators of the stories are also the

⁵ Interpretation, e.g. the people, things, etc, as pre-processed by the author's cultural codes, knowledge, location or by the reader as well as the context when experienced is not addressed in the WP1 deliverable. Interpretation analysis both of the author's constructed narrative and especially of the reader's understanding is also not particularly relevant to the presentations of tools. Where such aspects of interpretation are relevant, a discussion is begun in the "Children as Storytellers" chapter in D2.1.

creative authors and the story is created 'in the moment.' In this section we begin to explore the notion of improvisation theatre and how it might help give an understanding of possible practice with the tools.

In improvisational theatre actors collaboratively create scenes and stories on stage, often with the barest of frameworks. We explore some of the notions of this form of storytelling and discuss its vocabulary as it relates to the storytelling tools in the project.

From a work-a-day improvisation theatre perspective, that of trying to build new scenes (stories) to perform on the spot, a fundamental understanding of narrative has to be gained. For "improv players" this sense has to become nature, part of the subconscious operating procedures of interacting with other actors and the audience. This understanding in the words of Jeff Wirth[wirth95] corresponds to developing a sense of environment and character, what he refers to as: *Character, Relationship, Object, Where*. This understanding clearly corresponds to the existents of Chatman's taxonomy. Wirth also refers to the components that make a story a story, *Beginning, Middle, End, and Tag*. A good social understanding of these concepts allows actors to spontaneously create collaborative stories, this understanding may be explicit or implicit.

An ending may provide a confirmation, or refutation of expectations, often an ending reminds the audience of elements of narrative that may have been forgotten and provides a cohesive 'wholeness' to the experience. A good (popular or satisfying) narrative often has a clear end, it does not leave elements 'hanging'; there is a logical sequence that provides a thread through the narrative (often forming the "point" of the story).

This theory of the popular notion of story can be related to what is often referred to as the "classical" or Aristotelian narrative[aristotle97]. In *Poetics*, Aristotle is acknowledged as one of the first to give an account of narrative. One key element of this account relates the ordering of events in a narrative (Setting, Relationship, Conflict, Increase of conflict, Resolution) to the nature of narrative (that it is of a mimetic nature, art as an imitation of real life). An Aristotelian Narrative, embodies this straightforward telling of story, a linear time progression and what has come to be standard, exposition, climax, denouement structure. This "classical" or "canonical" narrative and often implies a plot centred around cause and effect.

Although we have no "educational" policy of "story" outcomes, there may be one conveyed regardless. From adult examples of narrative both oral and using the tools with the children and our demonstrative guidance with the children when they are creating narratives, we are providing examples of "good" narratives (often of the classical form outlined above). This however is not enforced through the tools. There have been discussions about providing "narrative templates" in order to "get children started." These discussions, however, have been motivated by wanting to encourage children to explore story space, rather than the idea of providing "good" examples. Such templates would provide an example of how a story might be structured in the media/screen space. An approach to building a generalised storytelling practice might be to provide a number of different templates of different nature. Providing enough of these might lead the user to infer that story configuration is not set. As a nascent media with unknown possibilities it would be unnecessarily restrictive to limit story possibilities to the classical forms.

2.5.1 Offers, Accepting, and Blocking

In the theory of improvised theatre, people such as Keith Johnstone [johnstone89] describe the way improv players build stories collaboratively as a sequence of making *offers* and *accepting* those offers without *blocking* them.

An improvised scene is built around actors working together to generate offers, the other actors accepting those offers, working to establish more of the narrative (story and structure) and moving on to generate more offers which in turn will be accepted by the other players. One of the greatest improv errors is to block such an offer, to not accept. Blocking offers destroys the narrative, changes the context, removes the flow. As an example, an actor may offer “this mountain air is so fresh”, offering both a setting (mountains) and implying an activity (hiking, walking, something healthy as a goal). An actor reply of “what do you mean, it is smoky and dark in this theatre” would be a radical block. A block of not just the story (characters, place, event), but of the whole enterprise of creating environments of fantasy in the theatre. There is little doubt such a block would get a laugh from the audience, however it is at the expense of the other players, the performance and the constructed narrative. Such a block from a player would be seen as anti-social, eroding the trust of the collaborative experience. An exercise to work on these accepting skills is aptly named “yes and...” In this exercise actors respond to anything another actors says with a new offer following the words “yes, and....” This progressively adds elements to a constructed narrative.

2.5.2 Interface and offers

One reason these concepts are relevant is that it gives a language for describing a quality of interactive narrative construction, and what might be termed “an enabling interface.” Given this background, we can then ask, “how can our tools accept offers, and facilitate the acceptance of offers between collaborating users”? Further we might inquire how the tool might make offers (outlines for existents) and provide a framework for a story (events, structure)? That is to say, informally, how can we ensure the tools are in a “yes, and...” loop? Just as important is that the tools do not block offers made by the child user. Many of the stories produced this year have taken offers from previous sessions, other students stories, the tools, etc. For example, with the Klump, many students tell stories about monsters, gooey substances, “blobby” characters, aliens, and other supernatural phenomena. Informally, these sorts of topics occur more often with the klump than with KidPad (where often there are stories of houses, people, and family life). This analysis has yet to be done, but it is clear that the affordances of the tools, and the settings in which they are presented, do have some influence and make offers to what the children have created as stories.

The tool and its designers have a great deal of influence over the sorts of narrative components that are created. The end results depend on the morphology of creation tools, the allowable movements and gestures, the library of pre-defined “stencils”, etc, as well as external factors including the context of the tools setting. As mentioned, in practice we have seen children take examples given by the teachers in storytelling quite literally. Many elements of the storytelling end up in the artefacts, even when children have been asked to “create their own.” This of course is also an example of collaborative narrative building, where the children are accepting the “offers” of the teachers and building on them. The point here is that, as tool designers, we need to be aware of the power we wield as we define the interfaces and make offers unintentionally or not. Very concrete

tools may suggest very concrete ideas to children, while it is possible that more abstract tools might provide starting points, but not carry with them extra cultural and historical meaning that might ‘over-specify’ the content of the diegesis.

2.5.3 Recounting forms of Narration: narrator, narratee, implied reader and reader

Our work has of yet not looked deeply into the roles of the narrator, narratee, implied reader, and reader. These different roles, in our model of storytelling tend to be fluid. Given that most of the stories that are created are inherently collaborative constructions, the narrator, narratee, and reader may be constantly swapping and redefining their roles (see D2.1 chapter “Children as Storytellers”). Also, since it is often the creation of the story that is the exploratory learning experience, the notion of the reader-audience has not come into research focus as yet. For the most part, stories have been recounted once by one of the story creators, and the subsequent readers have been other members of the class or, research members of the KidStory team for the purposes of tool design evaluation.

However we have realised there is a need for story persistence and have built in functionality for saving and retrieving stories in KidPad and Klump. This realisation has come directly from interaction with the children in the schools who have asked to revisit stories previously created. We see this as a fruitful area of investigation and find some of the deeper issues worth investigating. One example is in providing a collective story memory by expanding the saving and retrieving function and allowing children to constantly expand their stories, or other’s stories. For example, a child might have one “pad” for telling stories, that contains all of their collaborative stories and perhaps those of the class. Using the zooming structure of KidPad it is possible to store these. However this issue of story-persistence will raise further performance issues to be investigated (e.g. the saving, rendering, and retrieving of very large story files).

2.6 The tools for collaborative exploration

“We believe that new learning experiences need to be developed that are supported with technologies that are as inherently collaborative as a box of crayons or a pile of blocks.” (Allison Druin –KidStory project program).

We now focus the presentation on an introduction to the storytelling platforms that have been extended and developed in the first year. The co-located single display collaborative use of consumer style (PC) machines is central to KidStory. The tools we build will enable a flexible, rich, basic, interaction toward the explorative creation of stories developed through collaboration. Currently available tools do not achieve this. The personal computers that are ubiquitous today are just that, personal. They are not built on a model that allows the basic sharing of on-screen activity. For example, it is a fact that trying to engineer solutions on WINTEL machines to enable the use of two simultaneous gesture input devices, e.g. mice, encounters fundamental design problems and conflicts with the lowest design level of the machine. This is because the producers of desktop machines have not yet acknowledged a need for this collaborative use of computers in, for example, single display GroupWare (SDG).

2.7 Project platform development: Two approaches, KidPad-Klump

The project started with two platforms, KidPad and Klump. KidPad was near school-ready from the start of the project as a single user prototype form before the project began. The Klump, on the other hand, has been viewed as the more experimental storytelling tool. With the Klump we have tried to explore some ideas and concepts that might lie outside of mainstream computer-based storytelling tools and in the process trying to discover ways of augmenting children's collaborative storytelling experiences.

2.7.1 Hardware Implications of Two Platforms

The two platforms have differing requirements for running. The KidPad platform is implemented in Java and runs well under both Windows 98 and Windows NT. However, because the multiple input device architecture is based on the USB architecture and this is not supported in NT, multiple users can only run KidPad on Windows98. KidPad runs well on high-end laptops outfitted with a USB hub and 2+ mouse input devices (4 have been tried, theoretical maximum is 256).

The Klump application relies on 3D model computations and textures and thus requires hardware accelerated 3D rendering. We have experimented with a selection of new laptops with graphics acceleration, but as yet have not discovered a portable solution. This has hampered, to some extent, our ability to run this application in the schools regularly. Thus, partly because of the extra hardware set-up, the Klump application has been tested and run in school sessions less often than KidPad. However, the Klump application was ready from the start of month five in the project with a two mouse solution, while KidPad was released with a two-mouse solution and general architecture in month nine. Thus, the Klump application gave us some initial feedback and information from the in-schools sessions and helped improve the overall scheme of building the general multiple input device architecture and especially the mechanisms and metaphors to support collaborative story creation.

2.7.2 Issues influencing technical design and specification

Portability has strong implications for technical design. We have been investigating methods for leaving equipment in the schools that can be easily set-up and configured as per teacher and school needs. We knew from the beginning that using generally available consumer platforms was important. The first 3 months of the project were spent ensuring that applications ran under the Windows platform. Long term use in schools requires either installing our software onto school computers or leaving computers in a manner enabling teachers to easily configure and operate. Another issue encouraging portability is theft. The school locations in Stockholm have been the subject of computer theft. This implies that a portable platform would be better as it could be moved away from highly visible locations and locked up. This is a practice the Swedish school already employs. Portable computers are connected to docking stations and locked up in a safe at night.

The KidPad platform, as stated, works well on such windows-based portable computers. The Klump application is not quite running at usable speed on laptops. Our hope is that hardware will

improve in the near future, making 3D graphics accelerated laptops more common. This is already happening, driven to a great extent by the 3D multi-user gaming community. Most major manufactures now offer some form of graphics acceleration in their portable computer lines. Currently, however, the Klump application requires a third-party 3D graphics accelerating extension card. Unfortunately not many docking stations support such cards, so it is currently PC boxes which meet the computational needs of running the Klump. In practice this means significantly more set-up and configuration for our schools sessions and long term set-ups in schools require special set-ups.

The Klump application currently runs under Windows NT. With the introduction of Windows 2000, we expect to be able to integrate the multiple input device (MID) solution from KidPad into the Klump application. This will greatly simplify some of the set-up and configuration time of the Klump during the school sessions with the children and teachers.

2.7.3 Toward Storytelling Objects and Reactive Spaces

As we progress in the project, the expectation is to move further away from traditional desktop solutions to methods and mechanisms that both relate more to children's notions of storytelling and to supporting children's interactions in the classroom. Towards this we are looking for solutions that build on the experiences and framework of the "shared spatial desktop computer" in different ways. Simple solutions involve the dressing up (e.g. fur and foam) of devices to become new types of input devices. More complex solutions involve the use of new physical manipulatives. These solutions might be based on the current MID architecture and USB (Chapter 3), or they may be non-technical objects with links to the computer-based storytelling platforms. Such new interfaces, and displays, will move away from simple one hand interactions toward body interactions and may involve the use of cameras. Displays also will move away from the desktop and may become smaller, e.g. small LCD displays, as well as larger, wall-sized displays big enough for group sharing. These steps will be carried out within the context of participatory design in the classrooms with children and teachers in an attempt to gain an understanding of how these tools can augment storytelling, collaboration and communication.

The progress of technical development in later years is away from the desktop computer and toward a form of real-space interaction. The project is to move away from the monitor box entirely. Thus, not much effort is spent on issues such as trying to understand navigation in the "fishbowl" (the desktop monitor in which you use a mouse to interact inside of it). The first year technical work has built a foundation for further exploration of moving away from the desktop.

2.8 References

[Aristotle97] Aristotle, and Heath, Malcolm (Translator), Aristotle's Poetics, Penguin, 1997.

[Bal97] Bal, Mieke, Narratology : Introduction to the Theory of Narrative, University of Toronto Press, 1997.

[Berger96] Berger, Arthur Asa, Narratives in Popular Culture, Media, and Everyday Life, Altamira Press, October 1996.

-
- [Bordwell96] Bordwell, David, *Film Art: An Introduction*, McGraw Hill College Div; August 1996.
- [Chatman78] Chatman, Seymour, *Story and Discourse : Narrative Structure in Fiction and Film*, Cornell University Press, 1978.
- [Druin96] Allison Druin, Cynthia Solomon, *Designing Multimedia Environments for Children*, John Wiley & Sons, 1996.
- [Engle95] Susan Engel, *The stories Children Tell, making sense of the Narratives of Childhood*, D.H. Freeman, 1995.
- [Genette90] Genette, Gerard, Jane E. Lewin (trans), *Narrative Discourse Revisited*, Cornell University Press, 1990.
- [Johnstone89] Johnstone, Keith, *Impro : Improvisation and the Theatre*, Routledge Theatre Arts, 1981.
- [Labov72] Labov, William, *Language in the inner city*. University of Pennsylvania Press, 1972.
- [Martin86] Wallace, Martin, *Recent Theories of Narrative*, Cornell Univ Press, April 1986.
- [McEwan95] McEwan, Hunter and Egan, Keiran (eds), *Narrative in Teaching, Learning, and Research (Critical Issues in Curriculum Series)*, Teachers College Press, 1995.
- [Murray97] Murray, Janet, *Hamlet on the Holodeck: The future of Narrative in Cyberspace*, Free Press, 1997.
- [Prince89] Prince, Gerald, *A Dictionary of Narratology*, Gerald Prince, Univ of Nebraska Press, 1989.
- [Wirth95] Wirth, Jeff, *Interactive Acting : Acting, Improvisation, and Interacting for Audience Participatory Theatre*, Fall Creek Press, 1994.

3 KidPad - A 2D+Zooming Collaborative Storytelling tool

3.1 Background

We started developing a predecessor to the current KidPad technology in 1995 at the University of New Mexico. In that effort (Druin et. al., 1997), we developed the basic approach we are following with the current design. The original KidPad system had these basic features that we have refined for the KidStory project:

- **Local Tools** - an alternative user interface approach which replaces pull-down menus and tool palettes.
- **Zoomable User Interfaces** - the basic "canvas" that the stories are created on are *zoomable*. This means that children can create stories that can be zoomed into for more detail, and the zooming can in fact become a fundamental part of the story.
- **Single Display GroupWare** - support for multiple children simultaneously using a single computer, each with their own mouse
- **Storytelling authoring software** - the basic idea that children can learn communication skills by creating and telling stories.
- **Children as design partners** - children will learn more, and the technology will be most appropriate if children are closely involved in the design of the technology.

For the KidStory project, we have refined each of these concepts with significant effort going into technology (with a complete re-implementation in Java) as well as design. This chapter discusses the current version of KidPad, and how each of these aspects has progressed during the first year of the project.



Figure 3.1: A very simple drawing with some "crayons"

3.2 Design

A basic goal of KidPad is to help children improve their communication skills through storytelling. Our aim is to focus on 5-7 year old children, and to use visual, cartoon-like stories to support an international group of children without much familiarity with computers. To this end, we developed three user interface technological approaches that dovetail in a way that results in a novel approach to computing. These three aspects of KidPad (local tools, zoomable user interfaces, and single display GroupWare) will be looked at in detail.

As is discussed in Deliverable 2, the basic approach to designing KidPad has been to work with children from the beginning, and develop and refine the technologies based on the children's comments, drawings, and other feedback. This use of children as design partners makes the process of the technical development unique, and has directly helped in making KidPad what it is. Many of the basic ideas, especially local tools and single display GroupWare, were a direct result of working with children. Many of the refinements in the technical implementations also come from working with the children. This will be discussed at the end of this chapter.

The KidPad approach to creating stories is based on a very simple cartoon-like model of stories. Because we are working with such young children (5 to 7 years old), we have tried to create a model of stories that is not too abstract. Stories are by definition elements that change over time in response to actions. However, creating an electronic story interface that changes over time can be confusing because there needs to be a mechanism to control the time within the story. Further, different elements of the story are visible depending on the story time.

We avoid this complexity within KidPad by creating all elements of the story on the KidPad canvas in a static fashion. We control the time within the story by changing the *viewpoint* onto the KidPad canvas. Much as a cartoon where the reader reads each pane sequentially, a KidPad story consists of story elements (similar to cartoon panes), and moves from one element to the next with spatial hyperlinks. KidPad also uses zooming to offer a spatial and intuitive mechanism for organising the story elements, and moving between them.

3.3 Local Tools

We initially started working with children using very traditional interfaces based on pull-down menus and tool palettes. We quickly discovered that while children could learn to use them, they did not appear natural or intuitive to the children. The basic issue is that these mechanisms are quite abstract. The menus were confusing because most of the information was hidden most of the time - and not only that, but the information was very textual, and the information was hidden in a very abstract manner.

Further, the tool palettes while fairly easy to use for adults, seemed difficult for children to master. After spending some time working with the children, we asked them how they wanted to think about their tools. They pointed to a box of crayons and showed how they can just dump the crayons on the table, and pick them up and use them as needed. We realised only at this point that tool palettes are an abstract concept. While we think of them as a palette of tools, they really are "mode switchers". Clicking on a tool with the mouse changes the mode the computer is in, and the mouse

then works differently.

Motivated by the children's box of crayons, we developed "local tools". KidPad has no pull-down menu, and no tool palettes. Instead, it starts up with a simple empty canvas with some crayons sitting on the canvas. The crayons are tools that move when the mouse is moved. When the child clicks on the mouse button, the crayon draws. There are actually several crayons on the canvas, and if the child clicks on another crayon, it is swapped with the one that they have been using (see Figure 3.1).

This approach extends to any number of tool types. The tools sit directly on the drawing canvas, and can be picked up by clicking on them. They can then be used by clicking again. This approach simplifies the notion of going to a special place to pick a mode - instead, the child understands to use a tool by simply picking it up just like they do in the physical world.

Local tools work fine for a small number of simple tools, but we had to develop some improvements to get this concept to work well as we developed a larger number of more complex tools. We developed the concept of a "tool box" to keep tools in. Each toolbox can be opened which takes the tools out of the box, and lines them up on the canvas next to the toolbox. Later, it can be closed which puts the tools away by animating them into the toolbox.

We created three toolboxes which store different types of tools. The crayon toolbox stores just different crayon colours. This one starts out open so children can start drawing immediately. The advanced toolbox stores all the other extra tools: a hand for navigating, an eraser, a text tool for writing, an arrow for selecting and moving things, a magic wand for creating hyperlinks, and a bulletin board for saving and recovering previously created stories (these will each be described in turn.) Finally, the third toolbox is just now under development, and is intended to contain "modelling" tools, which support animation, sculpting, and 3D figures.



Figure 3.2: Each of the three toolboxes opened to show the tools

The second toolbox (figure 3.2, middle row) has the following tools:

- **Eraser:** This is a very simple eraser which erases the entire object that the eraser is over. The children have asked for an eraser which will erase just part of an object, and that is something we will add.
- **Text Tool:** This tool lets the child use the keyboard to type simple text.

- **Selection Tool:** This tool picks up things that have already been created, and then the objects can be either moved around, or resized.
- **Magic Wand:** The magic wand is perhaps the most important tool for creating stories. It enables children to make links between different parts of the story. Children typically draw stories by creating different elements of the story in different places on the canvas - similar to panes of a cartoon. The magic wand creates a spatial hyperlink from any object to any place on the canvas. Then, later when the child wants to tell the story, she picks up the "Hand" tool which is used to follow the links.
- **Hand:** The hand is used as the primary story-telling tool. It lets the user follow links previously created with the magic wand. The hand can also be used to "pan" through the story. This is similar to scrolling in that dragging the hand on the canvas drags the story so that it follows this hand. In this way, the child can access more canvas space. For example, dragging the hand to the left moves the story to the left, opening up empty canvas space to the right.
- **Bulletin Board:** Stories can be saved with the bulletin board tool. This tool works differently than the others. Rather than picking it up like other tools, clicking on the bulletin board automatically zooms to a different place where pictures of all the previously saved stories are hanging on a wall. Clicking on any one of those stories loads that story in. Or, clicking on the "Save" button will save the current story onto the bulletin board. This mechanism allows children to manage their stories without having to create filenames, or worry about where the files are stored. The bulletin board is also used to print the current story.

The third toolbox (Figure 3.2 top) is just now under development. It is intended to contain "modelling" tools. These are tools that are used to create and modify more complex shapes. Currently, there is a single modelling tool called the "turn-alive" tool. Children use this tool to animate lines they have drawn with the crayons. It makes them undulate in a wave-like fashion. This can be used for many things - for instance, it can make smoke from a chimney appear to flow, leaves on a tree blow in the wind, or ears on a horse twitch. We anticipate adding many more kinds of animation, and controls over the animation parameters.

The children have consistently asked for easier drawing tools. One of the ways we are trying to answer this is by investigating alternative kinds of drawing tools. For instance, we may offer a special tool that takes a two-dimensional crayon drawing and makes it three-dimensional by extruding the shape. Alternatively, we are thinking about introducing sculpting tools that enable the child to start with big piece of stuff, and cut things away from it leaving the remainder as a creation.

3.3.1 Collaborative Tools

As described below in the section on Single Display GroupWare, KidPad supports several children working simultaneously. Each child uses one mouse, and each mouse controls one tool. So, if three children each have a crayon active, they can all draw simultaneously. This frequently results in simultaneous, but separate interaction. We wanted to encourage more direct collaboration, so added the notion of collaborative tools.

There is a special "copy" tool that makes a copy of any tool. Once there is more than one copy of a

particular type of tool, two or more children can simultaneously use that tool. We attempted to encourage collaboration by making it so that when the children use two tools of the same type at about the same time and place, something special happens. For instance, if two children start drawing with a crayon at about the same time, and near each other, then instead of getting two separate lines, it results in one thick line that gets drawn between the two crayons. In addition, the colours of the two crayons are mixed.

Collaborative tools are described in complete detail in chapter 5.

3.4 Zoomable User Interfaces (ZUIs)

A basic element of KidPad is that it supports 2D zooming. This means not only that the drawing canvas can be zoomed in and out so that things can be made larger or smaller, but also that you can draw with more detail as you zoom in. Children regularly use this feature to help tell stories. For instance, by zooming into a character's head, you can see what they are thinking. By zooming out, you can see an overview of the entire story. Figure 3.3 shows a sequence of views as we zoom into a simple story. We have found zooming to play three roles in KidPad:

- **Navigation:** At its simplest, zooming can be used as a simple navigation aid. To get to a part of the story that is fairly far away, the child can just zoom out to get an overview, and then zoom in to the part of the story in question.
- **Large Canvas:** Zooming provides a simple mechanism for accessing a large canvas. Traditional visual programs rely on either a single large linear document (such as word processors), or more commonly, individual pictures which are disconnected or perhaps related by hyperlinks. Zooming creates an organisational structure based on a single large canvas. Similar to a large mural, different parts of the story can be put in different places, and then spatial hyperlinks (created with the magic wand) connect the story elements. This approach has a very different feel than traditional interfaces, as the interface implicitly provides context - showing the relationship between each part of story. As one child put it, "Travelling on the internet is like travelling with your eyes closed. Zooming is like travelling with your eyes open."
- **Story-Telling Structure:** After children learn how zooming can be used to organise pieces of their story, and move through it - they typically quickly realise that zooming can be used as an intrinsic part of the story itself. As mentioned above, a typical use of zooming in this manner is to zoom into a character's head to see what he is thinking. We have also seen zooming used in this way for a number of narrative themes - zooming into the place where someone lives to see it in more detail, zooming into a house to see what is in the house, etc.



Figure 3.3: A sequence of views as we zoom into a simple story (from left to right, and then top to bottom)

While zooming makes a richer storytelling environment, it has its costs. Zooming adds complexity in two areas - navigation, and implementation. Navigation is difficult in KidPad primarily because standard interface hardware does not have good affordances for zooming in and out. Since we are currently using standard PC computers with just keyboards and mice, we must use the regular keyboard and mouse buttons for navigating through the zooming space.

We tried many different navigation mechanisms with the mouse and keyboard. Every mechanism we tried with the mouse was too difficult for young children to control. We first tried a three-button mouse where the middle button zoomed in and the right button zoomed out. The children had trouble managing the big mouse, and couldn't remember well which button did what. We then tried a two-button mouse where the right button zoomed in and out (depending on which way the mouse was moved), and that was even harder to use because children couldn't figure out which way to move the mouse to zoom the way they wanted. Finally, we gave up on the mouse, and used the keyboard to zoom. Now, pressing the "Page Up" key zooms in, and pressing the "Page Down" key zooms out. Also, pressing the arrow keys pans the view left, right, up, and down. This approach also solved a shared navigation problem. Since there is just one keyboard, only one child can use it at a time, and thus the scene can be zoomed in only a single direction at a time.

We built KidPad using Jazz, an open source Java package that supports general purpose zoomable interfaces (Bederson & McAlister, 1999). The University of Maryland group working on the KidStory project built Jazz for separate purposes (and under separate funding), and so the KidPad development effort was able to take good advantage of the close connection with the Jazz development team, and even to have modifications made to Jazz in order to support KidPad.

Jazz takes many of the structural elements common to 3D graphical systems, and creates a

scenegraph for 2D graphics. By using a basic hierarchical scenegraph model with cameras, Jazz is able to directly support a variety of common as well as forward-looking interface mechanisms. This includes hierarchical groups of objects with affine transforms (translation, scale, rotation and shear), layers, zooming, internal cameras (portals), lenses, semantic zooming, and multiple representations. KidPad takes advantage of many of these features.

3.5 Single Display GroupWare (SDG)

Communication, collaboration, and co-ordination are brought to many people's desktops thanks to GroupWare applications such as Lotus Notes and Microsoft Exchange, two of the leading commercial products in the field of Computer-Supported Co-operative Work (CSCW). They help people collaborate when they are not in the same place at the same time. We believe that computers should also help people collaborate when they are in the same place at the same time.

Single Display GroupWare (SDG) is a model for supporting co-present collaborative work. An SDG application makes it possible for co-present users to collaborate using a single computer and display through the use of multiple input devices (Stewart et. al., 1999).

SDG applications must deal with problems particular to their domain, and rules of interaction need to be considered. There is a range of issues that come up for builders of SDG applications. Interaction metaphors must be re-evaluated for multiple users. Simultaneous use conflicts need to be resolved. And, many applications maintain global state and use focus in a way that implicitly assume only a single user. These issues and others have been addressed elsewhere (Bier & Freeman, 1991; Myers et. al., 1998; Stewart et. al., 1999). In this paper, we focus on the technical issue of getting input from multiple devices on one computer with a consistent mechanism.

We built a general-purpose Java package called Multiple Input Devices (MID) to allow several children to simultaneously use KidPad (Figure 3.4). KidPad with MID allows several mice to be plugged into a single computer, and each mouse controls one KidPad tool. In this way, several children can create a story collaboratively. For instance, three children might be drawing while one is erasing. Children change tools by clicking on them with their own tool. It is even possible to grab a tool from another person as they are using it.

We only recently got the multiple mouse version of KidPad working, and so were not able to work with children in the school to get feedback on it. However, we have used it with some children in our lab. We found that children respond well to it, and have noticed how the co-present nature of the collaboration directly changes the way children work together. For instance, since the children are right next to each other, any one that regularly grabs the tools of another quickly gets noticed and responded to in a physical way!

We have made some preliminary informal observations with the current version of KidPad. We also have spent time with children in an earlier much simpler version of KidPad, and we have noticed that co-present use of KidPad appears to result in four kinds of collaborative behaviour. Most commonly, children rapidly switch regularly between these interaction styles.

- **Competitive:** Children react directly to the shared drawing surface by trying to control the

other's drawing. For instance, one children frequently get into "scribble wars" where one tries to draw over the other. Or, one may frequently erase the others work. This tends to be done in fun, and since the children are next to each other, typically doesn't last too long.

- **Collaborative:** Children work together to a common goal. For instance, one child draws the head while another draws the body of a character.
- **Separate:** Children ignore each other, each working on their own piece. Sometimes children even go so far as to draw a line on the screen where each child works on opposite sides of the line.
- **Peer-helping:** Children help explain a feature of the software to each other by demonstrating it with their own mouse. This works very nicely since the helper doesn't have to take the mouse away from the one being helped.

3.5.1 The Java MID Package

MID provides developers using Java the ability to write SDG applications with a powerful yet straightforward and cross-platform way of getting input from multiple devices. This section describes the MID architecture in detail.



Figure 3.4: Two children using two mice with one computer. They are running KidPad, an application we wrote that uses MID.

We chose Java because of platform independence. We want our SDG applications to run on multiple platforms. MID consists of a cross-platform Java layer, and a native platform-specific layer. The cross-platform layer consists of a general-purpose package and of sub-packages that have to be written for each device type (one sub-package per type). A native class must be implemented for each device on each platform. Applications using MID use just the cross-platform Java layer, and therefore do not have to be changed for use on different platforms.

MID currently supports just Universal Serial Bus (USB) mice on Windows 98. We picked this hardware/platform combination to start because of its wide availability and installed base, and because of the generality of USB devices. While Windows NT does not yet support USB devices, Microsoft has informed us that the next version of NT will, and we should be able to get input from

multiple mice there as with Windows 98. The only issue with the next version of NT is that a registry key will have to be set so that NT will not merge the input from multiple mice before it gets to MID.

We chose mice as the first device to support because they are not very expensive, and because a good percentage of Personal Computer users are familiar with how to use them. We chose the USB standard because it supports different device types, and can handle up to 256 devices simultaneously through a single USB port (via hubs). In addition, USB is growing in popularity which we expect will become the standard in the near future.

While MID is written largely in Java and is designed to be as cross-platform as possible, there is a fundamental trade-off to be made in the degree of portability vs. the feature set MID offers. Since Java already supports a truly cross-platform event model, we decided that our goal for MID is to enable applications to access all the input device capability possible while working in as cross-platform a manner as possible.

This trade-off has worked out to mean that while the applications that use MID do not have to be changed to run on different platforms, the underlying implementation of MID must be updated for different devices and different platforms. Furthermore, the very nature of accessing specialised input devices means that if those input devices are not physically available, then it is the application's responsibility to deal with that, and accommodate the user with whatever input devices are available. The good news, though, is that applications that get input from mice using MID will work even if the MID native code is unavailable. In this case, MID reverts to offering just standard Java events. We strongly encourage developers who plan to extend MID to offer this feature.

3.5.2 Related Work on Single Display GroupWare

Other people have also looked at the problem of getting input from multiple devices. Stewart solved the problem on X Windows using XInputExtension (Stewart et. al., 1998). His solution supported serial mice and tablets, and was designed for Tcl/Tk. MMM (Multi-Device, Multi-User, Multi-Editor) was an early SDG environment that also supported input from up to three mice (Bier & Freeman, 1991).

The Pebbles project has investigated the use of Personal Digital Assistants (PDAs) as input devices for SDG applications (Myers et. al., 1998). While PDAs as input devices are a fine choice for people that already have them or for some special situations, they are currently a prohibitive expense for most people that just want an input device to control a mouse cursor, being approximately 10 times the cost of a standard mouse. MID gets input from USB mice, which are considerably less expensive than PDAs. Mice may also have an advantage over PDAs when users, such as young children, don't have a lot of precision in their control of input devices.

Inkpen studied the effects of turn-taking protocols on children's learning in mouse-driven collaborative environments (Inkpen et. al., 1997). In her study, two mice were connected to a computer, but they were not used simultaneously. Bricker used Windows-based gaming input devices to investigate how multiple users collaborated on a single computer when each controlled a different aspect of the input (Bricker et. al., 1997).

Rekimoto's pick-and-drop (Rekimoto, 1997) allows for drag-and-drop between computers or PDAs through the use of special "pens". While his architecture supports a computer receiving input from multiple devices, the focus of his research was not on SDG. Pick-and-drop requires special hardware, computers that are connected by a network, and provides only two types of interaction: picking and dropping.

Some video game systems can get input from multiple devices and would be able to support limited SDG applications. These systems for the most part support only joysticks and are not particularly easy to program.

Finally, Buxton and others have investigated how one person can use both hands simultaneously to interact with a computer. This area also has all been implemented with custom code to receive input from one extra device, typically a tablet using a serial port (Buxton & Myers, 1986).

3.5.3 MID Architecture

In this section we describe the design of MID. Figure 3.5 gives a visual overview of its structure. MID consists of a general-purpose package that is a hub for all MID events, and device specific sub-packages (one for each device type) that are the sources of events.

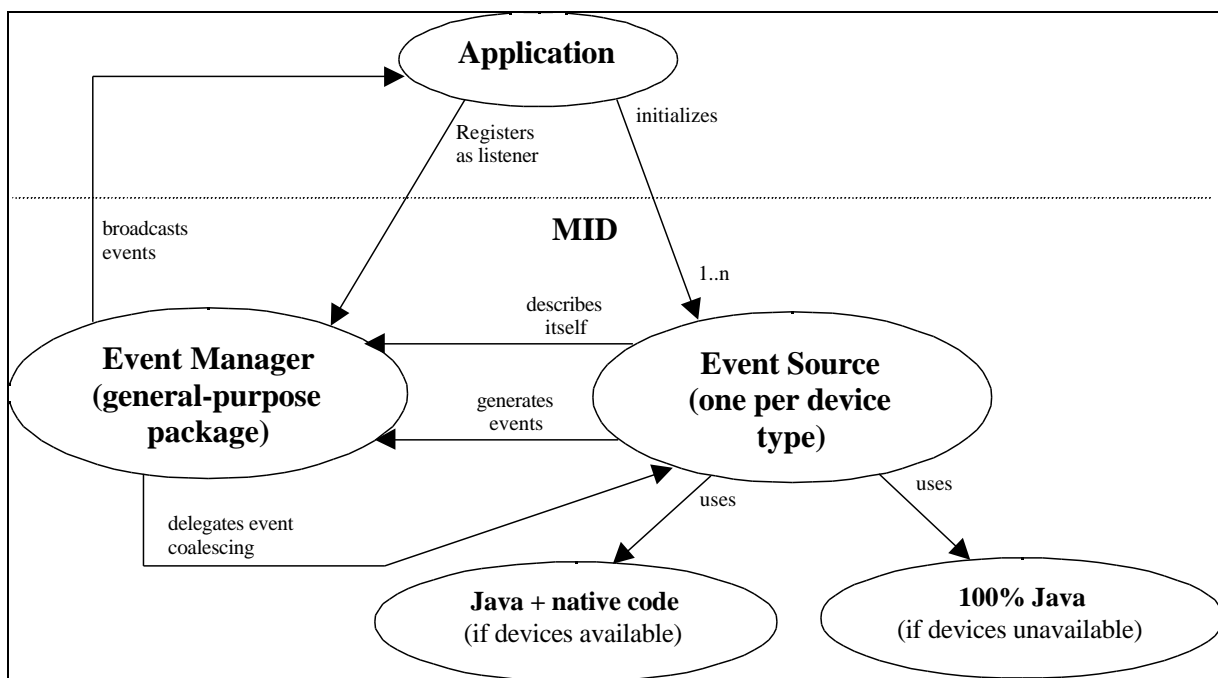


Figure 3.5: Architecture of MID. The application initialises the device-specific sub-packages and registers to listen to events with the general-purpose package. The device-specific sub-package uses a combination of Java and native code if devices are available and 100% Java code otherwise. The sub-package describes itself to the general-purpose package and tells it when events should be generated. The general purpose package broadcasts the events to the application. It also delegates event coalescing decisions to the device-specific sub-package.

3.5.4 MID general purpose package

This package handles the registration of listeners and the broadcasting of events. It has no specific knowledge of any of the device specific event sources and will therefore work with any event source written in the future. It is designed to make it as easy as possible for developers to add support for types of devices that Java doesn't support.

Event sources have to specify to the general-purpose package what type of events they support, what listener interfaces these events may be sent to, and what devices are available. Each device may generate multiple types of events, and there may be multiple devices for each source type. Each event type may be sent to multiple listener interfaces.

MID's general purpose package also supports the dynamic addition and removal of input devices by being a source of device change events. Objects that register to listen to this type of event are notified when a device is added or removed.

When broadcasting events, the general-purpose package posts them to the Java event queue. The Java event queue gives event sources the ability to coalesce events. The general-purpose package delegates this responsibility to the source that generated the event.

3.5.5 MID Event sources

Event sources are implemented with device specific sub-packages. They are responsible for providing all the needed device-specific support for the type of device they support. Their basic responsibility is to know when events should be generated. Event sources include event classes and their corresponding listener interface classes.

The event sources communicate with native code in order to know when to generate events. We have studied two different ways of communicating with native code. One way is to have the native code in a library and use the Java Native Interface to communicate with it. The other way is to use sockets for communication with an event server written in a separate process (could be Java or native code, and could even come from another machine).

The event sources are responsible for informing the general-purpose package of the event types and listener interfaces they support as well as the devices they have available. Sub-packages are also responsible for deciding when to coalesce events.

While it is not required, we recommend that all event sources revert to some kind of default behaviour if the devices they support are unavailable. This behaviour should be coded solely in Java. Applications that follow this recommendation will always work, even in some limited way when the native code is not available.

3.5.6 MID mouse event source

The MID mouse event source is the only sub-package that has been implemented so far for MID. It generates events from USB mice under Windows 98. It reverts to Java mouse events if multiple mice are not available. This ensures that applications that use this sub-package will always work.

The MID mouse event source communicates with a native code library through the Java Native Interface in order to know when to generate events. The native code uses the Microsoft DirectInput

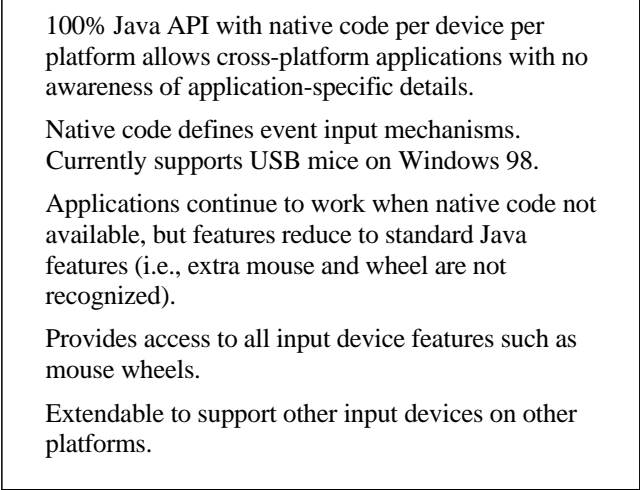
API to get input from USB mice.

MID mouse allows developers to set the location of mice, apply motion constraints, and the time span used for counting mouse clicks. Setting the location of mice is particularly useful for an application that has to define its own cursors and needs to give them an appropriate initial location on the screen. Constraining the motion of all mice and/or the motion of particular mice can be used to keep the cursor within the visible area of the window. Another application would be to divide the screen into regions where only one mouse would operate in each region. When MID is extended to support other types of devices, the corresponding sub-packages would also provide extra functionality specific to each type of device.

Another important feature of the MID mouse event source is that it enables applications to access all of the input buttons and movement axes on the particular mouse in use. Thus, a mouse with a wheel generates a third button event and motion in the z-axis.

When supporting a mouse event's access to the keyboard's control, shift, and alt modifier keys, this sub-package currently assumes that there is only one keyboard available. This access to the keyboard through mouse events demonstrates the low level at which the assumption of a single mouse and keyboard has been made. When MID supports multiple keyboards, we will have to deal with this differently.

The implementation of event coalescing in this sub-package coalesces move and drag events if they come from the same mouse. To review, Figure 3.6 summarises MID mouse's features.



- 100% Java API with native code per device per platform allows cross-platform applications with no awareness of application-specific details.
- Native code defines event input mechanisms. Currently supports USB mice on Windows 98.
- Applications continue to work when native code not available, but features reduce to standard Java features (i.e., extra mouse and wheel are not recognized).
- Provides access to all input device features such as mouse wheels.
- Extendable to support other input devices on other platforms.

Figure 3.6: Summary of MID mouse features.

3.5.7 Use of MID events vs. Java Events

To a programmer, using MID events is very similar to using Java events. MID extends the three types of Java classes associated with events that applications typically use: event listeners, event sources, and the events themselves. As an example, we will examine the differences between using Java mouse events and MID mouse events.

A class that receives Java mouse events has to implement the `MouseListener` and `MouseMotionListener` interfaces. It also has to register with a component (usually the visible

window where the events will occur) to get those events. It does so by calling the `addMouseListener()` and `addMouseMotionListener()` methods on the component. Finally, the class has to implement the methods specified in the interfaces. These methods will be called when mouse events occur, and a `MouseEvent` object describing the event will be passed to them. The top of Figure 3.7 shows sample code of a class that uses Java mouse events.

The bottom of Figure 3.7 shows sample code of a class that uses MID mouse events. The most noticeable difference between this code and the code that uses Java mouse events is two extra lines of code that get a MID source (`MIDMouseEventSource`) object (from the MID mouse event source) and a MID manager (`MIDEventManager`) object (from the general-purpose package).

Instead of registering with a component, the class has to register with the MID manager to listen to MID events. This is because the MID manager handles the broadcasting of all MID events. The method call used for registering to listen to events is the same for any kind of MID listener because the listener class is passed as a parameter.

```
// Java code using standard Java events to access mouse and mouse motion events
//
class MyClass implements MouseListener, MouseMotionListener {
    // Constructor adds mouse and mouse motion listeners
    public MyClass(Component myComponent) {
        . . .
        myComponent.addMouseListener(this);
        myComponent.addMouseMotionListener(this);
        . . .
    }

    // Mouse listener events
    public void mousePressed(MouseEvent e) {
        . . .
    }
    public void mouseReleased(MouseEvent e) {
        . . .
    }
    . . .

    // Mouse motion listener events
    public void mouseMoved(MouseEvent e) {
        . . .
    }
    public void mouseDragged(MouseEvent e) {
        . . .
    }
}
}
```

```
// Java code using MID events to access mouse and mouse motion events
// from multiple USB mice.
//
class MyClass implements MIDMouseListener, MIDMouseMotionListener {
    int numberOfMice; // Number of USB mice connected to system
    public MyClass(Component myComponent) {
        . . .
        MIDMouseEventSource midSource = MIDMouseEventSource.getInstance(myComponent);
        numberOfMice = midSource.getNumberOfMice();
        MIDEventManager midManager = MIDEventManager.getInstance();
        midManager.addListener(MIDMouseListener.class, this);
        midManager.addListener(MIDMouseMotionListener.class, this);
        . . .
    }

    // Mouse listener events
    public void mousePressed(MIDMouseEvent e) {
        int device = e.getDeviceID(); // Use mouse ID in application-specific manner
        . . .
    }
    public void mouseReleased(MIDMouseEvent e) {
        int device = e.getDeviceID(); // Use mouse ID in application-specific manner
        . . .
    }
    . . .

    // Mouse motion listener events
    public void mouseMoved(MIDMouseEvent e) {
        int device = e.getDeviceID(); // Use mouse ID in application-specific manner
        . . .
    }
    public void mouseDragged(MIDMouseEvent e) {
        int device = e.getDeviceID(); // Use mouse ID in application-specific manner
        . . .
    }
}
}
```

Figure 3.7: The top code fragment shows Java code that gets standard Java mouse and mouse motion events. The bottom code fragment shows Java code using MID events to access multiple mice.

MID offers the flexibility of registering to listen to all devices or to a particular device. This supports applications written in two styles. The first style dispatches events from all devices to each listener. It is up to the listener to determine which device generated the event by calling

`MIDEvent.getDeviceID()`, which returns the ID of the device that generated the event.

An alternative application style is to register a listener to receive events from a specific device. Then, an application would write one listener for each device, which would support de-coupling of the devices.

In addition, there is a call to `MIDMouseEventSource.getNumberOfMice()` that returns the number of mice currently available. This turns out to be necessary for most applications so they can build internal data structures and mechanisms to support the available input devices. If other types of devices were available through MID, the application would have to loop through the available devices to figure out which ones were mice. This could be easily accomplished through the MID manager. The only other difference in the code is that the names of the interfaces and the registration methods, and the name of the event have "MID" prepended.

3.5.8 Extending MID

MID can be extended in two ways: adding support for multiple mice on other platforms, and adding support for other devices.

Adding support for multiple mice on another platform consists of creating a native library that gets input from multiple mice and interfaces correctly with the MID mouse sub-package.

Adding support for other devices consists of adding device specific sub-packages for each type of device. These sub-packages would use native code in order to know when to generate events.

3.5.9 MID Conclusion

The most limiting aspect of MID is that it currently has only one implemented event source, and this source supports input from multiple mice only under Windows 98 when the mice are USB mice.

The mice have to be USB mice because of a limitation of Windows. Windows merges the input from the non-USB mouse with the input from USB mice into one event stream.

Another limitation of the MID mouse event source is that it takes over the system cursor and users cannot send mouse input to other applications unless they switch to other applications through the keyboard. This was done on purpose because the alternative is to have both mice control the single system cursor.

By generating events when notified by `DirectInput`, MID mouse suffers from minimal overhead. While we have not measured the time to generate MID events, we have compared it to the standard Java event code, and there are no noticeable differences that should affect performance. In addition, we have tested MID by using four mice simultaneously with our KidPad application running on a 266 MHz Pentium II laptop, and performance is fine.

As other input devices, such as tablets, become available for USB, we plan to add support for them. We also plan to make a fully functional MID mouse sub-package available to other platforms. Our first focus is likely to be UNIX systems that run X Windows.

In addition, MID provides the possibility of providing more application-specific input events. For

instance, an application could have a native voice-recognition system that generates events with the recognised voice data. Then, a cross-platform application could be written that accesses the voice data using MID with a mechanism consistent with other input devices. Thus, our hope is that MID becomes a uniform architecture to support multi-modal input.

We feel MID will make it easier to build SDG applications by removing the painful problem of getting input from multiple devices. Since MID's use is very similar to the use of Java events, developers with Java experience should find it easy to incorporate MID into their applications. We are also confident that MID provides a clean way of adding support for devices currently not supported by Java because of their type and/or number. While we think that there is room for improvement, we believe MID already is a step in the right direction.

3.6 Participatory Design and Iteration

We developed KidPad using the process of Co-operative Inquiry as described in detail in the Work Package 2 deliverables. To illustrate some of that process, we describe here a snapshot of some of the feedback we got from the children, and how we responded to that feedback in development of KidPad features.

Table 3.1 shows a frequency count of all the suggestions we received from children through analysing their journals (see Work Packages 2 + 3 deliverables for details on how this information was recorded and analysed.) The text below the tables then describes how we responded to these requests.

JOURNALS

Children's KIDPAD Design Suggestions Sweden+UK School Sessions-- Year 1

Autumn 98 Spring 99

7	7	easier tools to draw with
5	5	Animation
3	12	more colours
2	14	sound to tell stories
2	4	Internet/email-call someone
2	3	different crayon widths
2	3	Games
2	2	portable computer (walks or flies with you)
2	1	easier way to move around screen
2		wants to talk to the computer not write
1	10	draw straight lines
1	6	fill space with colour
1		easier way to move objects
1		secrets you can hide using zooming
1		magic wand should produce treasure/surprises
1		mix colour
	23	pre-drawn shapes/objects (e.g. triangles, eyes)
	10	additional media (TV, video, photos)
	9	letter/number icon
	8	stamps (e.g. KidPix)
	6	different input devices besides mouse
	5	green crayon
	5	an "undo" button
	4	a help button/person
	3	Dictionary
	2	Glitter
	2	easier to save
	2	Clock
	2	spell checker
	2	trash can
	1	surprise colour
	1	the computer to talk to the child
	1	more than 2 mice
	1	Eraser
	1	speech bubble
	1	screen saver
	1	rewind (backwards through links)
	1	record to play back stories
	1	make pictures invisible/reappear
35	159	Total Design Suggestions

Table 3.1. Frequencies of children's KidPad design ideas

3.7 Design Suggestions from Adults

Table 3.2 represents suggestions regarding KidPad offered by *adults* in their journals in the first year of the KidStory project. Adults offered significantly more KidPad design suggestions as the year progressed. In the autumn, adults offered 11 design suggestions in their journals. In the spring, adults offered 51 design suggestions in their journals.

JOURNALS			Adults' KIDPAD Design Suggestions
YEAR 1	Autumn	Spring	UK and Sweden—Year 1
6	4	2	multiple input devices
5	1	4	program should run faster
5	0	5	combine crayons, mixing colours
4	4	0	easier to delete/erase
3	0	3	other input devices besides mouse
3	0	3	more colours
3	0	3	tools that don't clump /get stuck on each other
2	0	2	screen refresher
2	0	2	turn alive tool
2	0	2	letters/text
2	0	2	templates for zooming
2	0	2	sound
2	0	2	playback
1	1	0	larger drawing pad
1	1	0	easier to use magic wand
1	0	1	more control over zooming
1	0	1	ability to make straight lines
1	0	1	function keys to define and use hyperlinks
1	0	1	thumbnails when saving
1	0	1	version of MID using Java
1	0	1	eraser
1	0	1	load a picture without having to save the current picture
1	0	1	fix Ctrl-Q (when you haven't drawn anything)
1	0	1	home key should be more exact
1	0	1	multiple tool boxes
1	0	1	x-ray box
1	0	1	ways to create secrets
1	0	1	tool "factory"
1	0	1	individualised tool boxes
1	0	1	undo feature
1	0	1	insert a story
1	0	1	library (e.g. Of shapes, textures)
1	0	1	video
1	0	1	levels of complexity for tool boxes
62	11	51	Total Adults' KidPad Design Suggestions

Table 3.2. Frequencies of adult's design suggestions for KidPad

3.8 Activity Patterns

Table 3.3 represents the activity patterns that we identified in the contextual inquiry sessions of our first year. Included are the frequencies that correspond to each activity pattern. In total, we identified 16 activity patterns and took note of 550 instances when those patterns were repeated. Activity patterns of greatest frequency include drawing and erasing, struggling for control of input device, storytelling and writing.

CI	Contextual Inquiry ACTIVITY PATTERNS
Year 1	UK and Sweden School Sessions—Year 1
147	Drawing
98	Struggling for control of input device
42	Erasing
39	Storytelling
35	Writing
34	Trying out features
26	Sharing control of input device
23	Difficulty selecting tools
21	Offers help
20	Seeks ownership
19	Seeks help
14	Practical co-operation
10	Linking/Zooming
9	Difficulty with drawing
8	Difficulty with linking (wand)
5	Difficulty with erasing
550	Total Activity Patterns

Table 3.3. Frequencies of children's activities when using KidPad

3.8.1 Contextual Inquiry Sessions of KidStory Project, Year 1

ROLES

Table 3.4 represents the roles that we identified in the contextual inquiry sessions of our first year. Included are the frequencies that correspond to each role. In total, we identified 11 roles and took note of 559 instances when those roles were performed. Roles of greatest frequency include children as artists, leaders, frustrated users, partners, and storytellers.

CI	Contextual Inquiry ROLES
Year 1	UK and Sweden School Sessions-- Year 1
170	Artist
97	Leader
60	Frustrated user
41	Partner
39	Storyteller
34	Writer
34	Explorer
24	Bored user

21	Helper
21	Learner
18	Owner
559	Total Roles

Table 3.4. Frequencies of ‘roles’

DESIGN SUGGESTIONS

Table 3.5 represents the design suggestions that we have identified in the contextual inquiry sessions of our first year. Included are the frequencies that correspond to each design suggestion. In total, we identified 14 design suggestions and took note of 288 instances when suggestions were made in those areas. Design suggestions of greatest frequency include multiple input devices, help options, easier to select tools, ways to fill colour, and ownership options.

CI	Contextual Inquiry Design Suggestions
Year 1	UK and Sweden School Sessions—Year 1
141	Multiple input devices
37	Help options
28	Easier to select tools
19	Ways to fill colour
17	Ownership options
13	Easier to erase
12	Easier to link (wand)
7	More colours
4	Easier to draw
4	Letter (Swedish “ä”)
2	Draw straight lines
2	Stamps
1	Undo button
1	Sound
288	Total Design Suggestions

Table 3.6. Frequencies of design suggestions

It is clear that all three forms of feedback, children's journals, adult journals, and CI notes all have similar kinds of suggestions, and enhance the other suggestions. To illustrate how we responded to the children's and adult's suggestions, we describe the iteration on a few design elements:

- **Easier tools to draw with:** This suggestion, as with many, needs some interpretation. The children often had trouble drawing at first with the crayons. However, it is not because the crayons themselves were hard to use, but rather the mechanisms for picking up and changing crayons were hard. In an attempt to offer some shortcuts to managing the tools, we first allowed double-clicking to “drop” a tool, replacing it with the hand. However, the children would often double-click accidentally, thus requiring them to pick the tool up again. Our second attempt led us to use the right button for dropping the tool. However, some children pressed the right button accidentally, resulting in the same problem. Finally, we eliminated this shortcut, and the children now just click on the hand tool when they want it. This makes the crayons substantially easier to use.

- **Animation:** Several children asked for ways to make things move. We started with the "turn-alive" tool described earlier in this chapter. It enables children to add some liveliness to objects, bringing the story alive.
- **More Colours:** This was a common request - especially later on in the Spring when many of the other problems were fixed so that children were able to concentrate more on the drawing they were creating. We responded to this request by adding more crayons (now there are six). However, we have already been told by the children that that is not enough. They would like the ability to mix colours, and change the crayon sharpness. So, we are currently working on that design idea.
- **Fill space:** As children used KidPad more, they began to ask for a way to fill space. They had to scribble a lot with a crayon in order to fill up an area. We first considered adding a traditional "paint" tool that fills in empty space. However, we were able to avoid adding an extra tool (and the associated clutter) by adding collaborative tools. We introduced collaborative crayons to create filled space. Now, picking up two crayons and drawing together fills the space between the crayons. This satisfies the children's desire of filling space, but not with the specific technique that they first suggested. Instead, it demonstrates a combination of children and adults working together to solve real problems in novel ways.

3.8.2 Artefacts

Detailed drawings made by the children who worked with KidPad are available in the deliverables of Work Package 2. Here in Figures 3.8-3.10 are a few screen snapshots of KidPad showing drawings typical of the kind of thing children have created.



Figure 3.8: A screen snapshot of an early version of KidPad

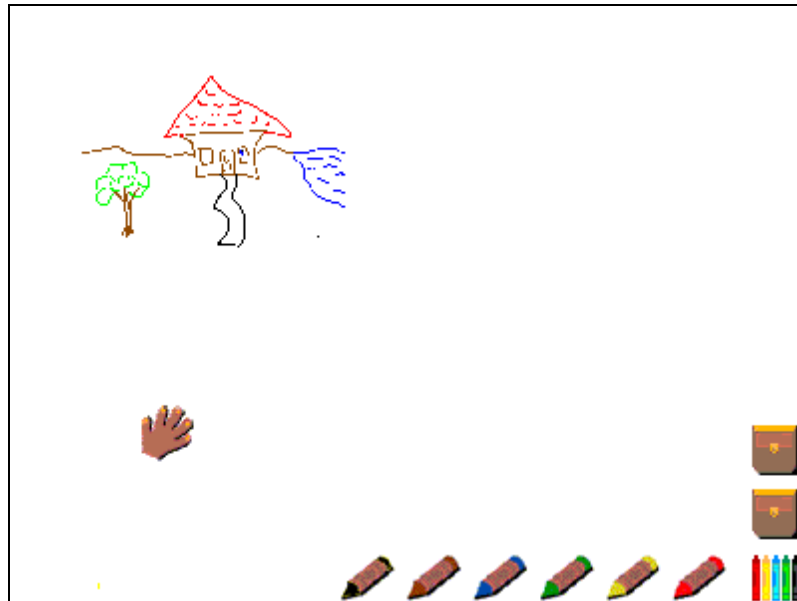


Figure 3.9: The current version of KidPad showing just the crayon toolbox open.

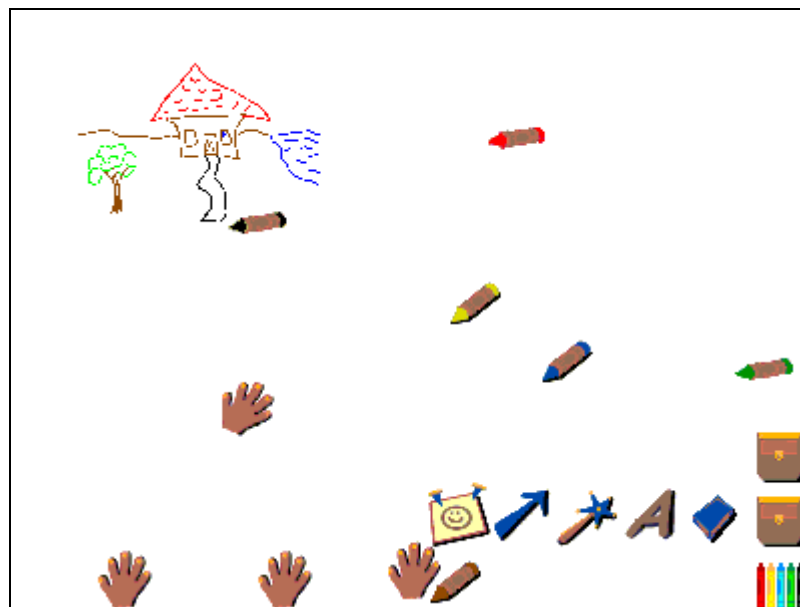


Figure 3.10: The current version of KidPad showing all the toolboxes open at once with four simultaneous users.

3.9 References

- Bederson, B. B., Hollan, J. D., Druin, A., Stewart, J., Rogers, D., & Proft, D. (1996). Local Tools: An Alternative to Tool Palettes. In *Proceedings of User Interface and Software Technology (UIST 96)* ACM Press, pp. 169-170.
- Bederson, B. B., Hollan, J. D., Perlin, K., Meyer, J., Bacon, D., & Furnas, G. W. (1996). Pad++: A Zoomable Graphical Sketchpad for Exploring Alternate Interface Physics. *Journal of Visual Languages and Computing*, 7, 3-31.
- Bederson, B. B., & McAlister, B. (1999). Jazz: An Extensible 2D+Zooming Graphics Toolkit in Java. In *Proceedings of User Interface and Software Technology (UIST 99)* ACM Press, (submitted).
- Bier, E. A., & Freeman, S. (1991). MMM: A User Interface Architecture for Shared Editors on a Single Screen. In *Proceedings of User Interface and Software Technology (UIST 91)* ACM Press, pp. 79-86.
- Bricker, L. J., Baker, M. J., & Tanimoto, S. L. (1997). Support for Cooperatively Controlled Objects in Multimedia Applications. In *Proceedings of Extended Abstracts of Human Factors in Computing Systems (CHI 97)* ACM Press, pp. 313-314.
- Buxton, W., & Myers, B. A. (1986). A Study in Two-Handed Input. In *Proceedings of Human Factors in Computing Systems (CHI 86)* ACM Press, pp. 321-326.
- Druin, A. (1999). *The Design of Children's Technology*. San Francisco, CA: Morgan Kaufmann.
- Druin, A., Stewart, J., Proft, D., Bederson, B. B., & Hollan, J. D. (1997). KidPad: A Design Collaboration Between Children, Technologists, and Educators. In *Proceedings of Human Factors in Computing Systems (CHI 97)* ACM Press, pp. 463-470.
- Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley.
- Inkpen, K., Booth, K. S., Klawe, M., & McGrenere, J. (1997). The Effect of Turn-Taking Protocols on Children's Learning in Mouse-Driven Collaborative Environments. In *Proceedings of Graphics Interface (GI 97)* Canadian Information Processing Society, pp. 138-145.
- Kurtenbach, G., Fitzmaurice, G., Baudel, T., & Buxton, W. (1997). The Design of a GUI Paradigm Based on Tablets, Two-Hands, and Transparency. In *Proceedings of Human Factors in Computing Systems (CHI 97)* ACM Press, pp. 35-42.
- Lee, S. K., Buxton, W., & Smith, K. C. (1985). A Multi-Touch Three Dimensional Touch-Sensitive Tablet. In *Proceedings of Human Factors in Computing Systems (CHI 85)* ACM Press, pp. 21-25.
- Myers, B. A., Stiel, H., & Gargiulo, R. (In Press). Collaboration Using Multiple PDAs Connected to a PC. In *Proceedings of Computer Supported Collaborative Work (CSCW 98)* ACM Press,
- Rekimoto, J. (1997). Pick-and-Drop: A Direct Manipulation Technique for Multiple Computer

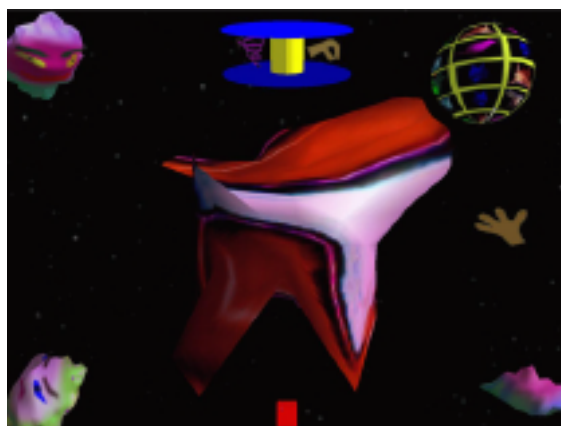
Environments. In Proceedings of User Interface and Software Technology (UIST 97) ACM Press, pp. 31-39.

Stewart, J. (1998). Single Display GroupWare. Doctoral dissertation, University of New Mexico, Albuquerque, NM.

Stewart, J., Bederson, B. B., & Druin, A. (1999). Single Display GroupWare: A Model for Co-Present Collaboration. In Proceedings of Human Factors in Computing Systems (CHI 99) ACM Press, p. (in press).

Stewart, J., Raybourn, E., Bederson, B. B., & Druin, A. (1998). When Two Hands Are Better Than One: Enhancing Collaboration Using Single Display GroupWare. In Proceedings of Extended Abstracts of Human Factors in Computing Systems (CHI 98) ACM Press, pp. 287-288

4 Klump – A 3D Collaborative Storytelling Tool



4.1 The Klump Application as an example of a Shared Desktop Storytelling Tool.

The Klump is the starting point for 3D storytelling tools developed in conjunction with the process of Co-operative Inquiry (see Deliverable 2.1). Starting with some existing technology that proved to be evocative in other contexts, the Klump is an application for conjuring up stories and collaboratively developing creative ideas. To provide a greater story structure, the Story Sphere concept is being developed as a placeholder for Klump shapes. Modelled Klumps can be frozen and placed into a sphere for storage and later retrieval for storytelling. The mechanisms of the Klump together with the Story Sphere offer a multi-media means of collaboratively creating, storing and retelling stories.

The Klump is a 3D graphical object with a physically-based spring model that gives it organic dynamic properties. In addition, the Klump is decorated in colourful abstract textures. The Klump also generates sounds that are directly related to its movements and user interaction. This combination of sound, movement, and textures has worked to make the Klump an engaging electronic artefact.

The Klump application is based upon earlier work done by SICS (within the ESPRIT I3 eSCAPE project). That work centred around an application called the Blob[Wallberg98]. The Blob is an interactive art piece, jointly developed by technicians, artists, and musicians in an attempt to make the technology meet the demands of art. The Blob was designed to be demonstrated on a large touch screen, where the user wears stereo shutter glasses and interacts with the sculpture's shape, texture, and sounds, by touching the screen. The enthusiasm and involvement the users expressed when interacting with the Blob is what the KidStory project wanted to capture and use in the context of creating story characters and free-form improvisational story telling. The Klump uses the same physics as the Blob, but is written for the Windows NT platform, for multiple simultaneous child users, and for more controlled modelling.

The Blob's origins are actually rooted in the KidStory project proposal writing. The first blob was

developed from an initial prototype of possible interaction object for children based on a participatory design session in Stockholm where children developed squeezable interfaces based on clay and balloons. It later became part of the eSCAPE WorkPlan. One intention of using the Klump in the KidStory project is that functionality is expected to easily extend to the tangible and spatial interfaces of the second and third years of KidStory.

The rest of this chapter will present the basic concepts, background and details of the of the Klump application work.

4.2 Grounding Concepts

The most salient aspects of the work with the Klump are the following:

- **Gestural modelling:** In interaction with the Klump, there is a strong notion of using “mouse gestures” to model both the shape of the Klump object as well as the colour of the textures, and the sound that is produced. What is meant by gestures is the movements of the mouse. In this way, users can pull out parts of the Klump surface as well as shape the sounds emitted from the Klump.
- **Subjective interaction:** Input device interaction among simultaneous users can differ. For example, while one user is moving or colouring the textured surface, another user can be modelling the Klump’s shape. Subjective interaction requires users being made aware of the mode of their input device when using it.
- **Multi-modal output and interaction:** The Klump employs different modal outputs. These are visual, including shape and texture, as well as aural, including modulated midi sounds. It is believed that it is (at least partly) this coupling of modalities with the dynamic behaviours that make the Klump an engaging focus of attention.
- **Storytelling 3D shapes and structures:** The Klump provides mechanisms for shaping and forming shapes that can be used for storytelling. The Klump’s organic movements, abstract textures, and sound tend to create an environment where child users are given ideas for stories. Interaction with the Klump tends to be engaging for many users. Through this interaction, ideas for stories tend to spring out. We call these “blind” offers (they are blind in the sense that, as yet, the application is not aware of story context). In this way, the Klump provides a mechanism for facilitating an improvisational storytelling between the children working with it.
- **Local Tools:** Following the experience with KidPad, the Klump application has adopted the Local Tools approach over a menu and palette approach common in many GUIs. However these Local Tools are extended into the 3D environment of the Klump environment. In this way, the cursors are 3D objects as well as mechanisms for selecting the 3D cursors.
- **Single Display GroupWare:** The Klump follows the SDG philosophy of collaborative work as outlined in the first chapter. This sharing of the display provides a platform for encouraging collaborative activities. The Klump also follows this Single Display philosophy by offering the Klump as a focal point for activity.

- **Children as design partners:** The Klump has been developed together with children and it is through this process, together with researchers that the concepts, metaphors and new functionalities have evolved.

The above concepts outline the basic concepts of the Klump and StorySphere storytelling application. What follows is a presentation of the details of the Klump functionality and work in developing the Theatre, TheatreWheel and StorySphere concepts which have lead us to our present stage of development. This is followed by a discussion of the participatory design process as it relates to the Klump and how this work is pointing the way toward the WP 1.2 (Storytelling objects) work.

4.3 Story Existents

In this section, the Klump will be presented as a tool to create existents (the characters and settings of a story). There are a number of ways to model and give the characters their looks and behaviours. The different tools will be explained, such as changing the Klump geometry, changing textures, and painting on textures. Furthermore, some technical issues related to the tools will be described. Also, the sound that is generated from interaction, which is believed to enhance the experience, is explained. As a last part of the Klump section, some collaborative device issues are discussed.

4.3.1 Shaping the Klump

At the heart of the Klump application is the mouldable clay or gel-like 3D object called the Klump. This virtual object is modelled as a mesh of point masses connected by springs, using discrete basic Newtonian physics.

Given an initial set of vertices (points on a sphere for example, see Figure 4.1), a large data structure is generated that holds all of the associated properties and parameters. Every vertex is assigned a mass and neighbouring vertices are connected by springs. The springs are modelled with parameters such as length and spring constant.

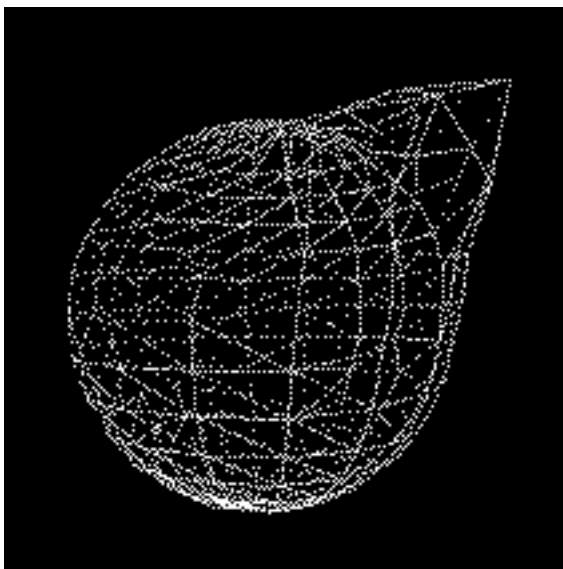
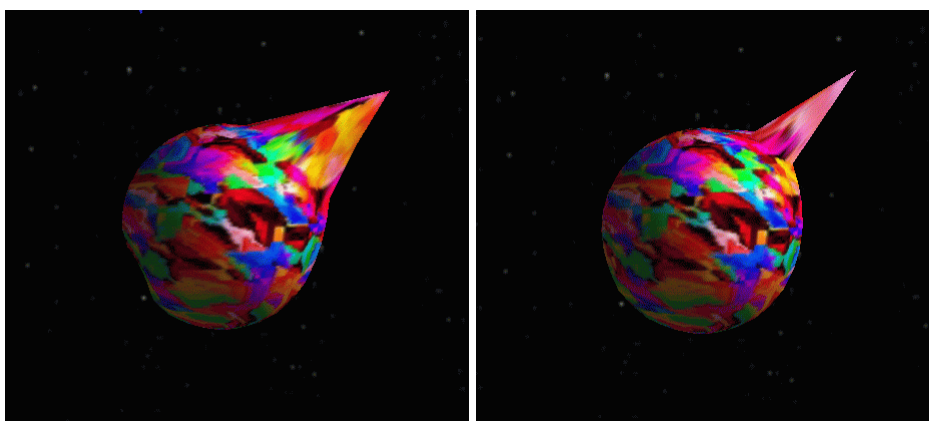


Figure 4.1: The Klump as a mesh

The user interacts with the Klump by pulling vertices, and thereby applying forces to move those vertices. The positions of the vertices are constantly recalculated.

Applying enough force to a group of vertices eventually stretches the lengths of the spring, and thus one can shape the object. Feedback from the child users (e.g. children tend to click more haphazardly and more often than adult users) has made us implement this so that only after pulling for longer than a certain amount of time (in the order of half a second), does the Klump actually stretch. The number of vertices that are affected by the pull force can be changed by selecting a 3D icon (see Figure 4.2).

**Figure 4.2: The different pull modes**

There is a trade-off between 'aliveness' and accurate shaping. If large spring constants are used between the point masses together with a powerful pulling force and little energy escaping the system, the response of the Klump is quick, powerful and rubber-like. Yet accurate modelling is difficult. If small spring constants, a more moderate pulling force, and some energy loss are used the Klump is more sluggish but more accurate modelling is possible. We have chosen the latter, not only to make accurate modelling possible but also to compensate for the child user's wilder mouse movements.

Simulated gravitational forces help keep the Klump in the centre of view at all times. This is carried out by recalculating the Klump's centre of gravity, and then applying corrective forces to keep it centred.

To give the Klump more alive, active, and engaging characteristics when it is not being interacted with, random forces are applied to random vertices at random points in time, to make the Klump jiggle. This random movement has proven to be an important properties to keep users engaged. This is done by noting the time at which each Klump interaction is initiated. After a certain amount of time with no user activity (approximately five seconds), small forces with random strengths are applied rapidly to random vertices, in the positive or negative direction of the surface normals. As soon as the user starts shaping the Klump again, this random activity ceases.

4.3.2 Morphing

A morphing algorithm (gradual change from one shape to another) has been introduced as a reset function for bringing the Klump back to its original shape. However, this feature can be extended to support morphing between different shapes, as a modelling tool. To morph, one needs to know the corresponding vertices of the two objects (the current Klump and the object into which it will be morphed). This is non-trivial because in general, nothing is known about the vertex correspondence between the two objects. That is, given a vertex in source object, it is unclear which vertex it corresponds to in the destination object. The algorithm used by the Klump application restricts the source and destination objects to have the same number of vertices. The pseudo-algorithm for matching the objects is:

```
load_object_to_morph_to
test_against_current_morphing_constraints (e.g. presently, number of vertices
                                           must match)
construct_list_matching_klump_vertices_to_corresponding_morph_object_ones {
    build_morph_obj_vertex_neighbor_array_by_going_through_vertex_index_array
    for number_of_existing_edges, starting at the first edge {
        match_vertices_on_both_sides_of_edge
        register_edges_between_the_current_edge_vertices_and_side_vertices
        put_the_newly_registered_edges_in_a_list_to_be_processed
    }
}
store_morph_spring_lengths_using_match_list
remove_morph_object
```

4.3.3 Texture Manipulation

Apart from shaping the Klump, the users can also manipulate the texture, the image that is wrapped onto the surface of the Klump. A number of different ideas have been discussed. Based on these discussions, we have introduced a number of different ways to manipulate the texture of the Klump, as described below. Currently, texture movement and change of texture image have been implemented. The other features will be introduced in future versions of the software.

4.3.4 Texture Selection

The users can change the texture by co-operatively, within a small time frame, selecting one of ten paint splats (some of which can be seen in Figure 4.3).

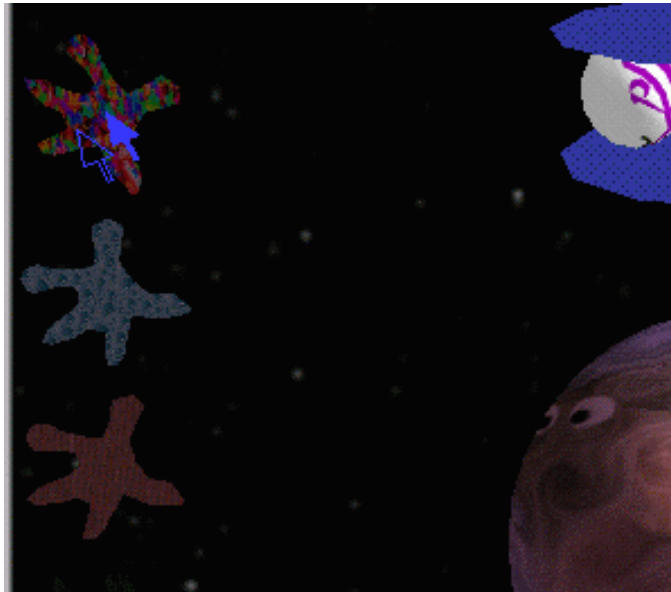


Figure 4.3: Selecting textures

The current set of textures includes both abstract graphical images and images of different facial expressions. To give feedback when a new texture has successfully been selected, the paint splat rotates for a short period of time.

4.3.5 Moving the Texture

The users can also choose to move the texture on the Klump, that is, slide it around to place a certain part of the texture onto a certain point of the object. When a mouse button is pressed, the texture will follow the mouse movement, in effect 'sliding' on the surface of the Klump (see Figure 4.4). The user can, for example, put the mouth under a nose she/he has modelled on one of the facial expression textures. Note that the effect of this is different than rotating the object itself. In appearance rotating the texture is analogous to rotating the “skin” (or clothes) of the Klump. The morphology of the object remains in constant transform, but the surface texture rotates about the object’s surface.



Figure 4.4: Moving the texture

4.3.6 Combining Textures

The current implementation for changing textures enforces collaboration, i.e. both users have to agree on which texture to select. To encourage rather than to enforce collaboration, a forthcoming

version of the software will allow users to combine textures. For example, given a set of facial components (mouths, eyes, and eyebrows), different facial expressions can be obtained by combining the components in different ways, as shown in Figure 4.5). By minimising the number of facial parts and expressions, the number of combinations can be kept at a minimum and all combinations can be predetermined and generated in advance. This gives a huge performance advantage since no image processing is needed to combine the images, i.e. the images are already pre-made. In a future version, arbitrary images could be allowed for any combination.

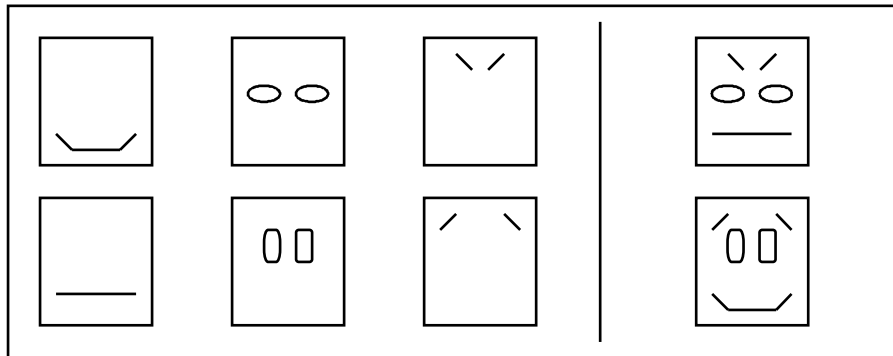


Figure 4.5: By combining different facial parts with different expressions, new faces can be produced.

4.3.7 Speckling

In the current version of the Klump tool, textures are selected from a predefined set of static images. To facilitate redesign of the images, we have implemented a prototype version of the software that allows the user to create 'speckles'. When pressing the mouse button, an image effect spreads in a spiral from the point where the cursor is situated as illustrated in Figure 4.6. The effect is analogous to a leaking pen. More can be done to expand this notion, e.g. leaking in a vector direction, oozing with differing viscosity, and incorporating the speckling itself into the story.

We have made a conscious choice of not providing standard drawing tools such as lines and polygons. The Klump application is exploring alternatives to 'direct manipulation, e.g. that are exhibited in paint programs. Thus, in some ways, the user interacts 'with' the Klump application. While the Klump does not attempt to be an 'agent', it does attempt to offer the child user a rich set of tools to conjure ideas and explore them. We want to provide the users with ways to change existing textures, creating new textures, in ways different than traditional 'creative' applications. One way of doing this is to allow the user to apply different effects onto the texture of the Klump.

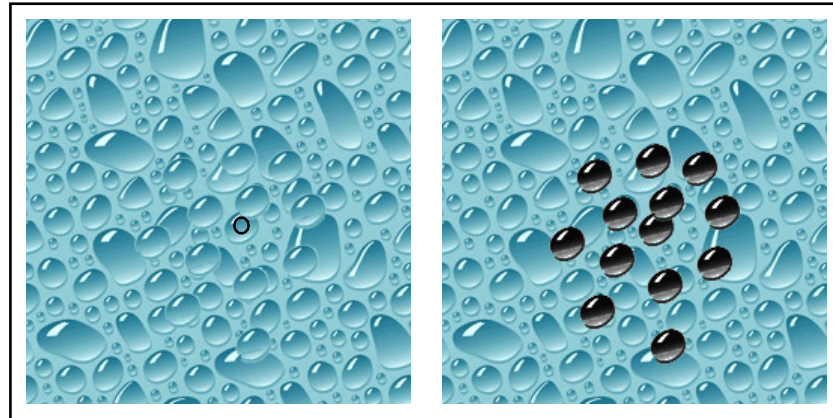


Figure 4.6: The speckling effect. Both images are the same but in the right one the effect is highlighted. The ring in the left image is the mouse interaction point.

4.3.8 Rubber Stamping

The rubber stamp tool is used to copy a user-specified region of the texture. To specify a region, one encircles it with a line, as shown in Figure 4.7. The region can then be 'stamped' onto other locations of the image. In multi-user mode, one user can define the image region while another user makes the copies. This tool will be available in an upcoming version of the software. It is inspired by programs like “KidPix” as well as through participatory design sessions where children have requested such “rubber stamping” feature.

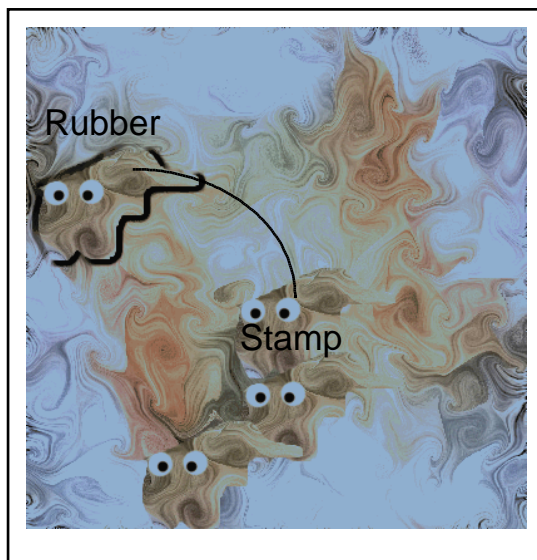


Figure 4.7: The rubber stamp tool. The rubber is highlighted to the left and with the stamp the user has drawn three copies of the rubber.

4.4 Technical Issues

Before delving into the collaborative device and technical issues, we will present some background information on DIVE, the platform on top of which the Klump software is written. We

will also give some brief information on previous work related to the Klump.

DIVE

The Klump software is based on the SICS Distributed Interactive Virtual Environment (DIVE) experimental platform for development of virtual environments, user interfaces and applications based on shared 3D synthetic environments[Hagsand97]. Dive is especially tuned to multi-user applications, where several networked participants interact and collaborate over an distance connected by the internet[Fahlén94, Benford95b]. Also built into DIVE, as well as part of the inspiration for it, is a notion and underlying model of “spatial awareness” [Benford93].

Dive is based on a peer-to-peer approach with no centralised server, where peers communicate via reliable and non-reliable multicast, based on IP multicast. Conceptually, the shared state can be seen as a memory shared over a network where a set of processes interact by concurrently accessing the memory. This model can be contrasted with other approaches such as server-based (hybrid or other) [Greehalgh95].

Consistency and concurrency control of common data (objects) is achieved by active replication and reliable multicast protocols. That is, objects are replicated at several nodes where the replica is kept consistent by continuously being updated. Update messages are sent by multicast so that all nodes perform the same sequence of updates.

The peer-to-peer approach without a centralised server means that as long as any peer is active within a world, the world along with its objects remains "alive". Since objects are fully replicated (not approximated) at other nodes, they are independent of any one process and can exist independently of the creator.

The dynamic behaviour of objects can be described by interpretative scripts in Dive/Tcl, which can be evaluated at any node where the object is replicated. A script is typically triggered by events in the system, such as user interaction signals, timers, collisions, etc.

Users navigate in 3D space and see, meet and collaborate with other users and applications in the environment. A participant in a Dive world is called an actor, and is either a human user or an automated application process. An actor is represented by a "body-icon" (or avatar), to facilitate the recognition and awareness of ongoing activities. This avatar represents the user's presence in the virtual environment[Benford95a]. The body-icon can be used as a template on which the actor's input devices are graphically modelled in 3D space.

A user ‘sees’ a world through a rendering application called a visualiser (see Figure 4.8). The visualiser renders a scene from the viewpoint of the actor's eye. Changing the position of the eye, or changing the "eye" to an another object, will change the viewpoint. A visualiser can be set up to accommodate a wide range of I/O devices such as an HMD, wands, datagloves, etc. Further, it reads the user's input devices and maps the physical actions taken by the user to logical actions in the Dive system. This includes navigation in 3D space, clicking on objects, grabbing objects, etc.

In a typical Dive world, a number of actors leave and enter worlds dynamically. Additionally, any number of application processes (applications) exist within a world. Such applications typically build their user interfaces by creating and introducing necessary graphical objects. Thereafter, they

"listen" to events in the world, so that when an event occurs, the application reacts according to some control logic.

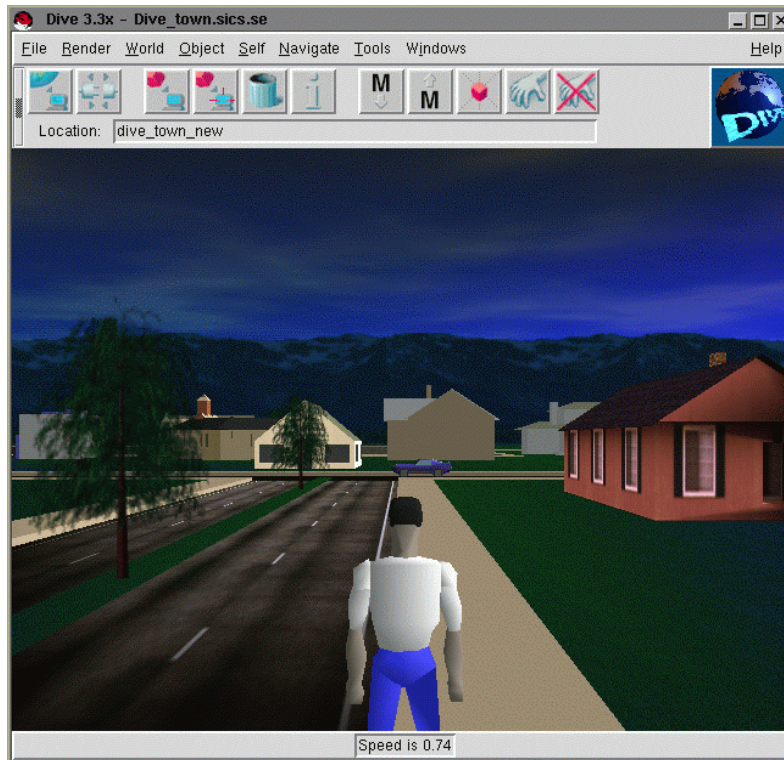


Figure 4.8: The DIVE visualiser

4.5 Graphics and Mechanics of the Klump

4.5.1 Effective Distribution and Mapping of Textures

As the Klump is a combination of several DIVE applications running simultaneously, changes to a texture must be distributed to all the connected peers. The test showed that real-time performance is only obtained for very small textures. Since the tools described above only change part of the texture image, it is only the affected regions that need to be distributed. We are implementing such an image distribution model in DIVE.

The current image distribution in DIVE is based on image streams, where each stream is associated with a unique name. This name can be used to tell an object to have a particular image stream as its texture. The stream does not necessarily have to be an actual stream of images, it is fully possible that it consists of only one image. By extending the current definition of images within DIVE we easily achieve partial distribution. The current image distribution is extended to also include images with an offset. A pair of x- and y-co-ordinates defines the offset. If the offset values are not equal to zero, then the image is supposed to be part of a larger image with the offset defining where within this larger image the sub-image is to be placed. If both offset values are equal to zero, the image is conceived as a whole image.

▪ Mapping Mouse Co-ordinates to Texture Co-ordinates

Most of the texture tools require that for a given 2D screen co-ordinate, a corresponding 2D texture co-ordinate located on the surface of the Klump is calculated. Since the Klump model consists of a set of 3D triangles with pre-defined 2D texture co-ordinates at each vertex, the texture co-ordinate can be obtained as follows:

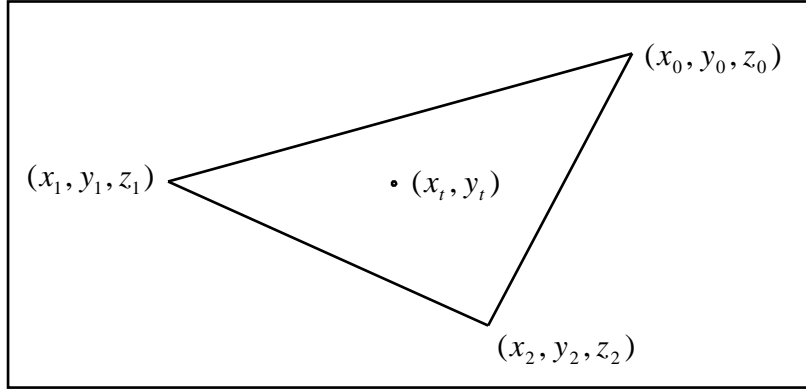


Figure 4.9: A Klump triangle with its co-ordinates for the different vertices.

Given the mouse cursor location (x_i, y_i) , a ray is traced from the virtual eye point through (x_j, y_j) on the virtual image plane, towards the Klump model triangle set. The intersection closest to the camera (if any) is noted. A given intersection point (x, y, z) on a triangle surface with vertices at (x_0, y_0, z_0) , (x_1, y_1, z_1) , and (x_2, y_2, z_2) , as shown in Figure 4.9, can be defined mathematically as

Equation 1

$$\begin{aligned} x &= x_0 + \alpha_1 t + \alpha_2 s \\ y &= y_0 + \beta_1 t + \beta_2 s \\ z &= z_0 + \gamma_1 t + \gamma_2 s \end{aligned}$$

Where s and t are the parameters for the surface and

Equation 2 and Equation 3

$$\begin{aligned} \alpha_1 &= x_1 - x_0 & \alpha_2 &= x_2 - x_0 \\ \beta_1 &= y_1 - y_0 & \beta_2 &= y_2 - y_0 \\ \gamma_1 &= z_1 - z_0 & \gamma_2 &= z_2 - z_0 \end{aligned}$$

By combining equations 1 and 2 we obtain the following expression

$$s = \frac{\alpha_1(y - y_0) - \beta_1(x - x_0)}{\alpha_1\beta_2 - \alpha_2\beta_1}$$

And by combining equations 1 and 3 we get

$$t = \frac{\alpha_2(z - z_0) - \gamma_2(x - x_0)}{\alpha_2\gamma_1 - \alpha_1\gamma_2}$$

In the texture image co-ordinate system the surface is only two dimensional and can also be described by a parametric form

$$\begin{aligned}x_t &= x_{t0} + k_x s \\y_t &= y_{t0} + k_y t\end{aligned}$$

Where x_{t0} and y_{t0} are the texture mapping co-ordinates associated with (x_0, y_0, z_0) . The notation is equal for the other vertices of the triangle, thus (x_{t1}, y_{t1}) maps to (x_1, y_1, z_1) and (x_{t2}, y_{t2}) maps to (x_2, y_2, z_2) . At (x_1, y_1, z_1) the parameters s and t are $s = 0$ and $t = 1$ which gives

$$k_y = y_{t1} - y_{t0}$$

And at (x_2, y_2, z_2) the parameters are $s = 1$ and $t = 0$ which gives

$$k_x = x_{t2} - x_{t0}$$

By combining the equations for x_t, k_x and s and in the same way combining y_t, k_y and t we get the following two transformation equations that transform a 3D interaction point into texture 2D co-ordinates

$$\begin{aligned}x_t &= x_{t0} + (x_{t2} - x_{t0}) \frac{(x_1 - x_0)(y - y_0) - (y_1 - y_0)(x - x_0)}{(x_1 - x_0)(y_2 - y_0) - (x_2 - x_0)(y_1 - y_0)} \\y_t &= y_{t0} + (y_{t1} - y_{t0}) \frac{(x_2 - x_0)(z - z_0) - (z_2 - z_0)(x - x_0)}{(x_2 - x_0)(z_1 - z_0) - (x_1 - x_0)(z_2 - z_0)}\end{aligned}$$

4.6 Interaction Sound

The Klump is manipulated via mouse interactions. If we look at the cursor movements and the way they are performed, we can see a similarity between the movements and "gestures" or "hand gestures". These interactions or gestures are linked to computer generated sound, to produce different sounds depending on the interactions.

To increase engagement when using the Klump, MIDI (Musical Instrument Digital Interface) support has been implemented to generate sounds. MIDI is a standard defined by the MMA (Midi Manufactures Association) that allows electronic musical instruments and equipment to communicate with one another. It defines a simple hardware interface and digital communications protocol based on messages. The MIDI specification includes a common language that provides information about events such as note on and off, pre-set changes, sustain pedal, pitch bend, and timing information. Microsoft operating systems support a standard method of accessing MIDI by using the Windows MCI (Media Control Interface) application programming interface. It is possible to access audio features independently of the PC sound hardware manufacturer.

The sounds are triggered when a user pulls a vertex. The vertices are divided into 20 groups depending on geometrical location, and each group is assigned a different instrument by being assigned to a specific MIDI channel.

When the mouse button is pressed over the Klump, a MIDI message is sent to the hardware and the MIDI synthesiser generates a single note, out of the 128 possible. Whilst the button is pressed, some of the characteristics of the note can be changed by moving the mouse. The note is defined by

- volume
- pitch – note's positive or negative shift in semitones from the initial pitch
- pan - position of the sound, left or right
- reverb - defines a lift where the sounds with reverb sound higher up in the speakers
- chorus - combination of delaying and de-tuning elements of the sound to create a richer, thicker texture

The last two are not programmable using the MCI on standard sound computer hardware, but it can usually be switched on and off in the computer's control panel. More sophisticated MIDI devices, like tone generators, may support full implementation of the protocol, and it could be possible to program these two features on a single note. The purpose of using all these characteristics is to achieve a "3D effect" with the sound linked to the interaction and movement.

Pan is linked to the pointer's horizontal position (left or right). When the vertical position of the cursor changes, the note is changed for one higher (up) or lower (down). The volume is related to the pointer's movements. If the pointer stops, the sound fades out to a minimum, and comes back up as soon as the movement starts again. This is a very basic implementation of "mouse momentum" where the mouse, or another input device, adopts dynamics properties. The note's pitch is linked to the horizontal position of the pointer, as shown in Figure 4.10. When the mouse is released, the note fades out.

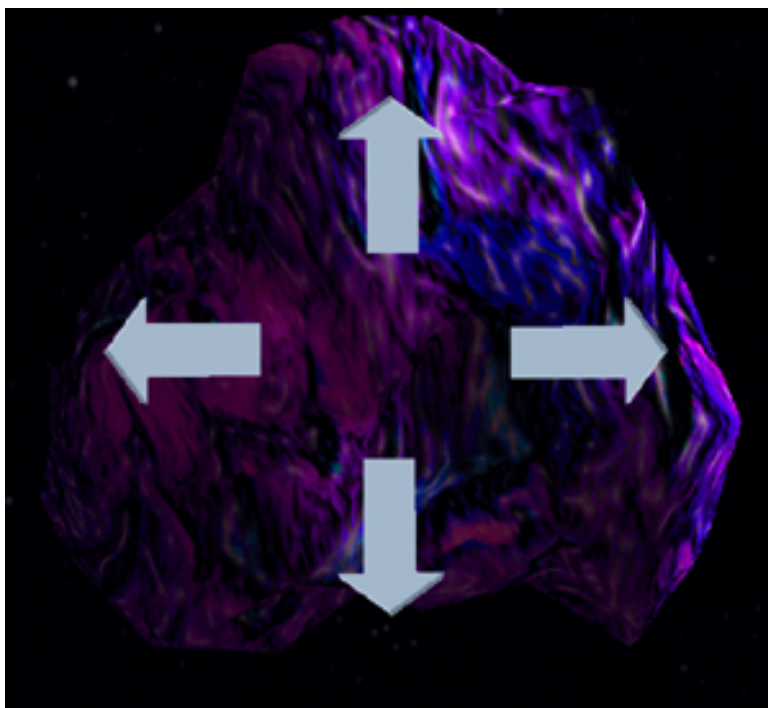


Figure 4.SEQARABIC10: MIDI effects directions

Apart from generating MIDI sounds, we are planning to investigate tools that will allow the users to produce the sounds themselves through, for example, sound selection or recording tools.

To this end, experiments have been carried out to implement the possibility of storing sound clips in DIVE which provides the functionality of capturing sound from an input source such as a microphone. These services have been used to capture the audio. Then MCI services for digital audio have been used to save the sound data as a WAVE format file on the hard disk.

4.7 Collaborative Device Issues

- Multiple Mice

The Klump application has been implemented as a stand-alone DIVE application. We have simulated two-mice-input SDG by running the Klump application on two separate networked computers with a mouse connected to each, as shown in Figure 4.11. Only one of the two running Klump processes actually draws the application window.

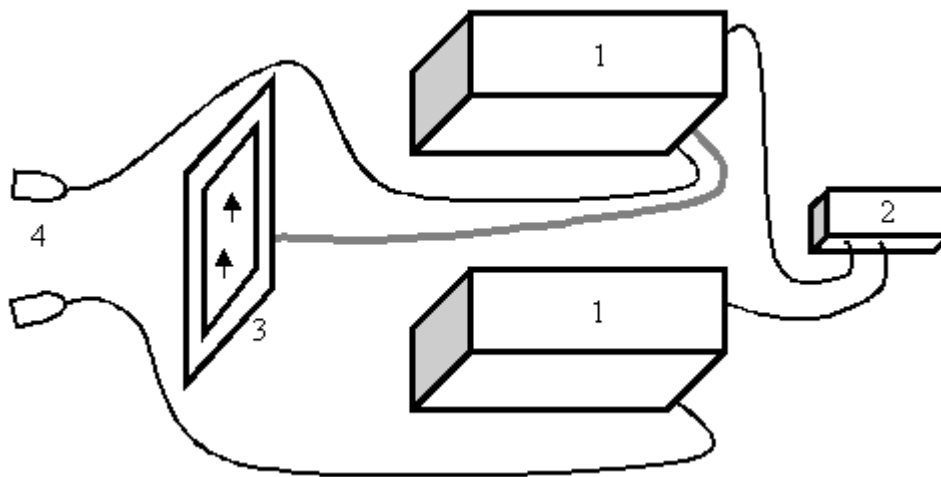


Figure 4.11: In the figure, the two computers are marked with '1'. The internet coupling hub is marked with '2.' The screen is marked with '3' and the two mice are marked with '4'. Power and keyboards are left out of the figure for simplicity.

Thus, only one computer is actually used to display the application window where we have replicated a cursor for the second computer's input within DIVE. The second mouse pointer for the process on the machine that is not visualising, is implemented as a virtual DIVE object, connected to the avatar. On start-up, the same DIVE process automatically finds the rendering process' avatar, which it changes to. This way, the two processes can share the same virtual interface. The object is modelled to look like a "normal" mouse cursor. Since the computers are connected directly to each other, we found it possible to decrease the network packet send buffer, yielding a decreased latency for the second mouse cursor. A new configuration for mouse interaction event handling was also added that also resulted in increased performance.

- Co-operation

To investigate how multiple input devices affect co-operation, we have implemented a few tools that enforce or encourage (depending on the tool) working together. For instance, when selecting textures, both users need to click on the icon at roughly the same time. It is also possible for one

user to shape the Klump while the other moves the texture. Since the Klump yields a natural focus, it generates a lot of discussion and negotiation. This will encourage collaboration when the possibility of combining textures is used. Then there will be a possibility for one user to change textures, and for two users to combine textures.

4.8 Kludding – A drawing tool as a form of midi ligature

Experiments with other tools for drawing in 3D with MIDI have also been made. An example of such a prototype was the “Kludding” tool (see Figure 4.12).

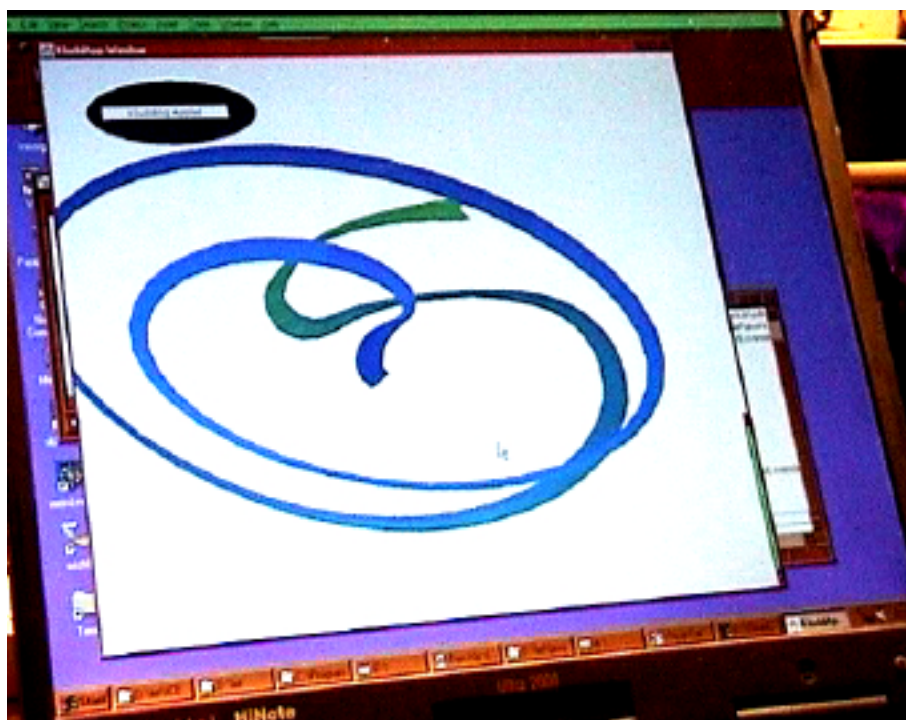


Figure 4.12: A typical view of the Kludding prototype drawing tool.

The kludding drawing tool is based on a physical model that includes a notion of inertia. The two parameters of this inertia are the mass of the drawing tool (e.g. the cursor) and the friction between the writing surface and the tool. In the prototype, both of these are adjustable by sliders on the side of the canvas. The user draws and interacts with the tool and its inertia. Via this interaction, the user interacts with a drawing and music creation process. As the user interacts with the cursor, curves and colors are laid on the canvas. The shape and color of the curves depend on the speed and direction of the tool. For example, fast movements do not make tight curves. The MIDI as well generates sound that also depends on the speed and direction of the movement.

The MIDI sounds generated by the tool movements are persistent. When drawn, they continue to sound. In this way the drawing on the canvas can be seen as a method of laying down MIDI sound ligature.

In addition to the drawing and interacting with the tool, the entire drawing canvas and MIDI sound can be scaled (up or down), copied, removed, and inserted. In this way the user can scale sounds

and drawing. Both the drawing and the sound scales. For an example of this scaling see Figure 4.13).

The Kludding application was made as a short demonstration in connecting a physical model to a tool manipulation feature. This led to some other MIDI extensions of the Klump application described above. For that purpose the Kludding application has been successful. Whether the kludding application will ever be developed further is not clear (kludding has yet to be tried by the children). Parts of it are very engaging. The tool purposely breaks a number of rules of “Direct Manipulation.” It was not intended to be a tool that was directly controlled, but instead a tool that you interact, and play, with.

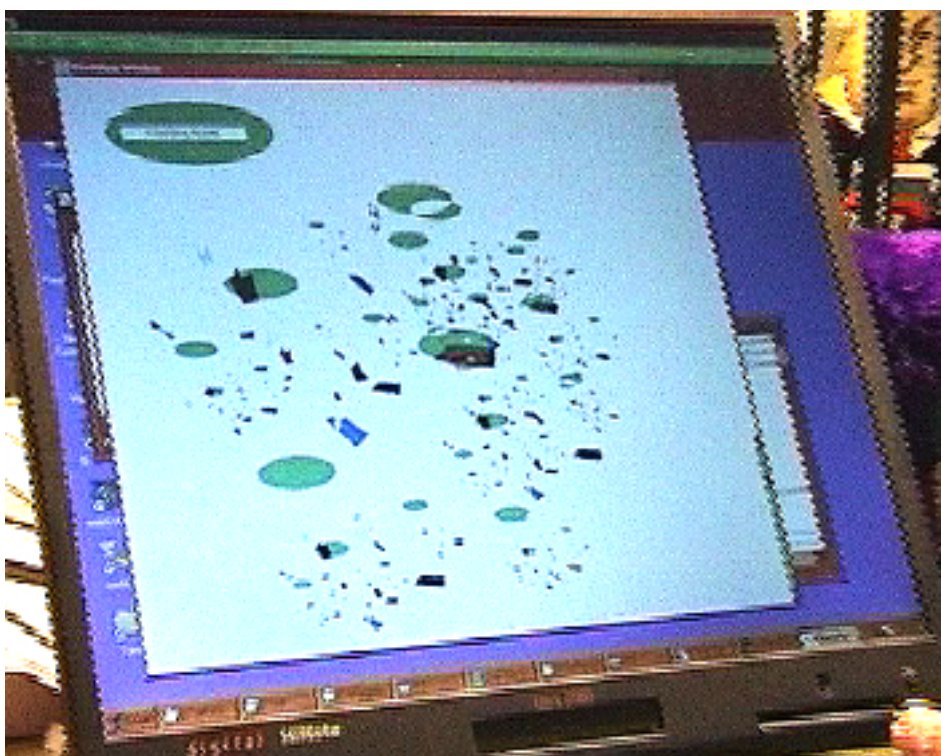


Figure 4.13: A view of the scaling feature of kludding ligatures.

4.9 Structure and Events

Going back to the examination of narrative, the *discourse* is the unfolding of the story – the way it is presented, how it is related to the 'audience.' This consists of two components, the *structure* and the *manifestation*.

To specify the discourse of the story, we have implemented a few tools to investigate ideas for working with the manifestation and the temporal structure of stories. The first tool is a theatre-like environment where it is possible to draw motion paths for characters on the floor. Also, the characters can be given internal behaviours, which tie back to *events*. Other ideas for this tool include incorporating a timeline (that could be replaced by a tangible clock at a later stage) to visualise and manipulate several scenes within the story.

The second tool, the theatre wheel, provides a greater structure to the theatre tool. This allows you to create several “scenes” that will constitute your story. The third tool, the story sphere, has the same purpose as the theatre wheel, but in a slightly different manner. It allows you to save your “scenes” as patches on a spherical ‘quilt’, which you can easily browse. To date, the only discourse tool presented to the children for further exploration is the StorySphere. The theatre thus has been seen as investigative experiment.

4.9.1 Theatre

The theatre tool is shown in Figure 4.14. The exterior of the theatre features a curtain that can be opened and closed interactively. There are two character objects on the stage, represented by red cubes. The character objects can be almost any DIVE object, including objects modelled by the user (possibly with the Klump application). There can be any number of characters on the stage.

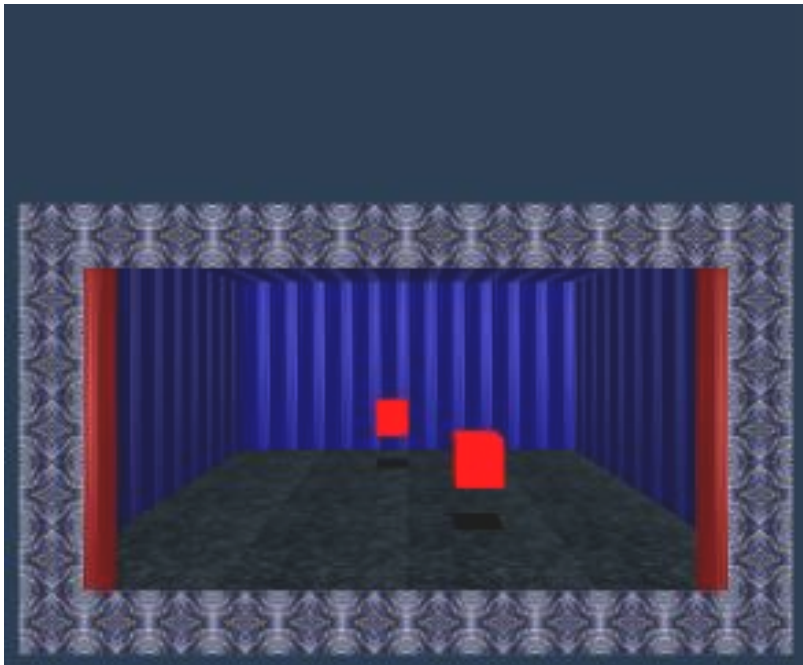


Figure 4.14: The Theater

The user can highlight objects by selecting them with the mouse. By clicking on the floor, the movement path of highlighted objects can be defined. When the user clicks on an object with a movement path, the object moves along the path until it reaches the end point. Objects that are at the end point of their movement paths can be reset to their respective start points.

4.9.2 Character Behaviours

We have explored some ideas for implementing character behaviours within the theatre tool. In the resulting implementation, each character can move across the theatre floor as described above but in addition, a lower level of behaviour for each path segment was added. The sub-behaviours are implemented in a stand-alone threaded subroutine package and can be applied to any character object. Thus, sub-behaviours can be created in almost any way desired, ranging from simple

procedural techniques such as

$$character\ height\ over\ floor = |amplitude * \sin(phase)|$$

to more complicated techniques such as live motion capture from user gestures.

We have implemented two different ways of letting the user assign sub-movements to characters. In the first approach, illustrated in Figure 4.15, every sub-behaviour is assigned a unique colour. The motion path segments display the colour of their assigned sub-behaviour. To choose a sub-behaviour, the user can click on control boxes at the path segment vertices. The path segment steps through all behaviour colours in turn. To aid the selection, a small icon displaying the currently selected behaviour appears when the user selects a control box.

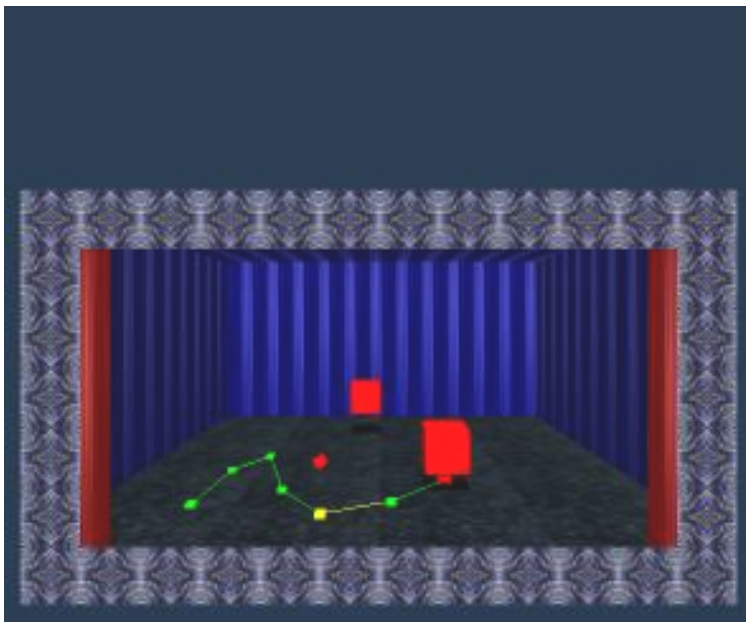


Figure 4.15: Assigning Submovements

The second approach to behaviour selection, shown in Figure 4.16, is a tool-and-icon-based solution. The boot tool is used to draw footsteps on the theatre floor and the tool represented by the theatre masks is used to assign sub-behaviours to path segments. The hourglass tool is used to assign pause times for movement path vertices and the clock tool replays the scene.

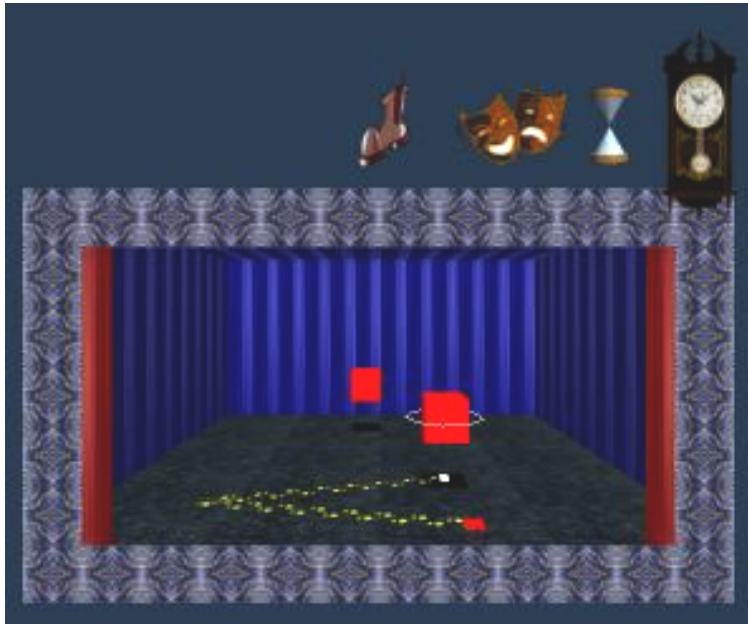


Figure 4.16: Behaviour selection

4.9.3 Theatre Wheel

To allow a higher level of temporal structure of the narrative, we allow the storytelling world to contain several theatres, each representing a different scene in the story. This is illustrated in Figure 4.17.

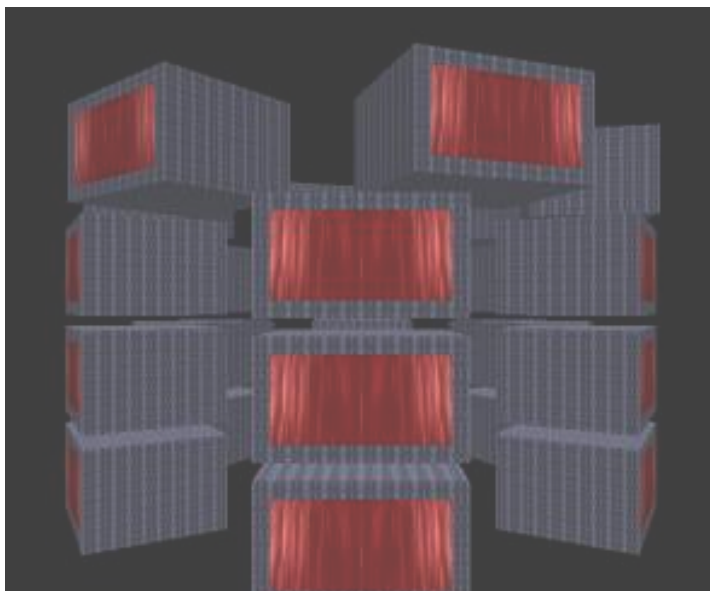


Figure 4.17: The Theatre Wheel

The current layout has four “wheels” with six theatres in each wheel. The wheels can be turned to let different theatres face the users, similar to the way the wheels are turned in a combination lock. The user can zoom to any one of the four forward-facing theatres, resulting in a close-up view (Figure 4.18). The idea is that the story progresses vertically, from top to bottom (or from bottom to

top). By turning the wheels, the user can “unlock” the story of his or her choice or even combine story segments by different authors.

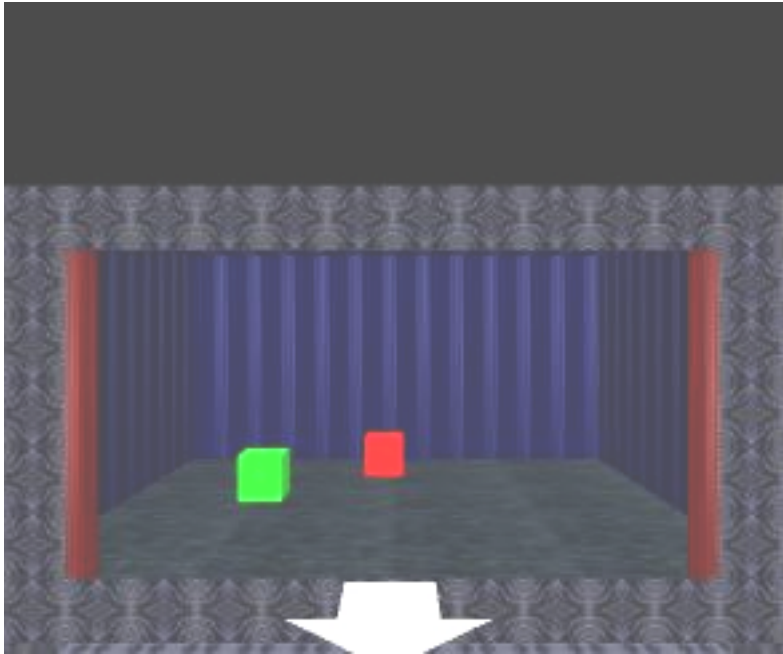


Figure 4.18: Close-up view of one Theatre on the Wheel

This “theatre wheel” metaphor for story structure specification can be likened to the classic wooden toy blocks with different animal body parts. In such a toy, the different body parts (e.g. a giraffe’s head, an elephant’s midriff, a tiger’s foot) appear on the faces of a cube and are twisted and combined to form a completely new animal that is the combination of the different animal body parts.

4.9.4 StorySphere

The story sphere is based on the idea of a “story quilt”, drawing from traditional American and African-American cultures. Further, it is based on the idea of juxtaposed images forming a greater story. Story quilts are in a similar artistic storytelling genre as triptychs and renaissance multi-panelled paintings (especially those depicting biblical scenes). The story sphere adapts this notion of a panelled quilt to a virtual 3D object. It then becomes a shared 3D storytelling device that you can manipulate, a container for events, characters, settings, etc.

The story sphere is a way of organising scenes into a greater story. It is based on the idea of a traditional “story quilt”, a set of panels each telling a small part of a story and forming a greater narrative. The story sphere is a 3D object with panels covering its face. Each “panel” is a small scene containing story elements (see Figure 4.19.) The sphere can be manipulated to display different panels, which can then be zoomed in on and worked on or presented individually. A story structure can be laid out onto the sphere and then later be used to play back collaborative story constructions.

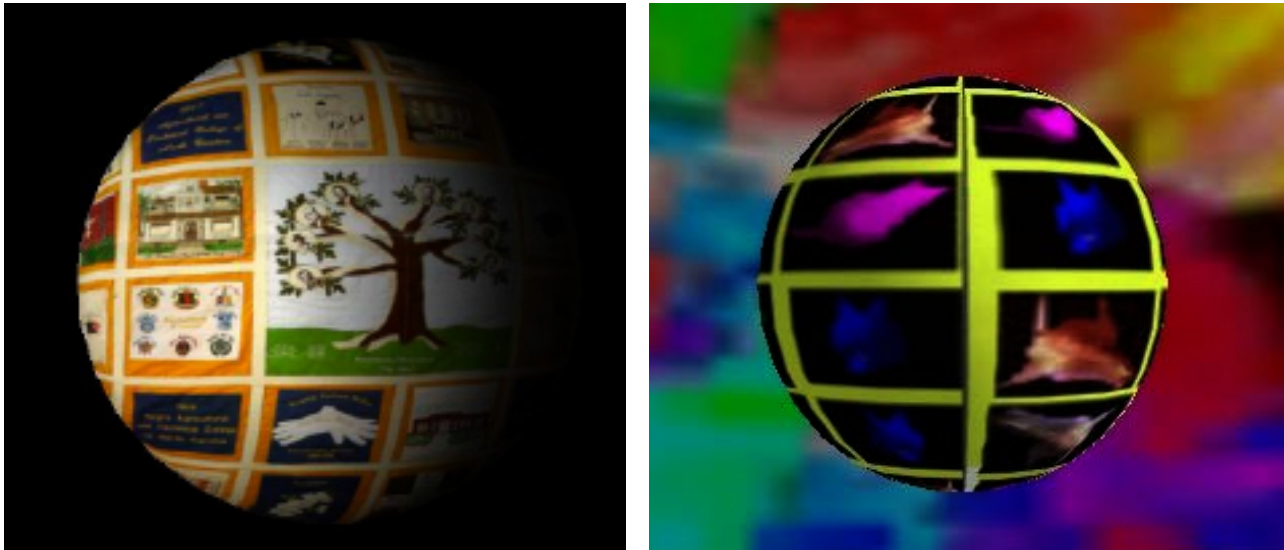


Figure 4.19: The Story Sphere (on the left: conceptual image; on the right: image of working prototype)

Primarily, the Story Sphere is a place to hold scenes and provide a mechanism for story structure. A structure for the story (time ordering, narration, etc.) can be stored in the cells of the sphere.. Using this structure, stories that are built can be played back for the audience. So in addition to being a method to store the story elements, the story sphere is a way to organise the greater structure of the scenes that are created.

The sphere can be rotated, spun, enlarged, augmented, etc. It can become a (nearly) infinite container for story scene elements. Together, the storytellers can rotate and select a scene to be worked on and then bring this scene to the foreground (by zooming into the sphere), work on that scene, then zoom out, rotate to another place and work on that. Also, an ordering of scenes can be specified by some mechanism so that a specific version of that story could be played back for a particular audience.

In addition, the story sphere provides a placeholder for storytellers to incorporate the stories of previous storytellers. The story sphere, upon start up, can already be loaded with scenes, containing settings, characters, events, from previous story creating sessions.

The story sphere also points toward the tangible interface of a ball that is rotated in real space. This ball could be the same object that is used for manipulating the Klump object for modelling. Thus, the same interface could be designed to support the capture of Klump modelling gestures and for manipulating the story sphere and selecting scenes to be worked on.

Above in Figure 4.19 is a picture of a prototype of the story sphere. The image of a story quilt has been mapped onto a sphere to offer an example of how such a story sphere might look. Each panel, of course, can contain a 3D scene of its own. Zooming out from the sphere may collapse these views of 3D scenes into simpler 3D or 2D depictions.

4.10 Participatory Design and Iteration

The Klump was partly developed using Cooperative Inquiry (see Work Package 2 deliverable). Since the Klump was at a much earlier stage of development than KidPad, there has been less data feedback. The following tables illustrate some of the feedback from children and researchers.

Children's Klump Design Suggestions

UK School Session-- 1999 January

//	Different input devices besides mouse (light pen)
/	have frames for Klump
/	ability to change shape of Klump
/	ability to change look inside of Klump
/	ability to draw with Klump shape
6	Total Design Suggestions
4	(Storytelling)

Researchers' Klump Design Suggestions

Swedish School Sessions-- 1999 January, May

improve second mouse performance
ability to record/playback sound from microphone
improve midi sound
story structure element needed
control of texture change
capability to draw straight lines
elements of surprise
more easily understood tool representations
make Klump start in full screen mode
easier way to start application in Windows
collaborative modelling functionality
different tools to pull with different widths
ability to paint/draw on Klump
Gestural interface for assigning

collaborative selection functions
other shapes than a sphere to start with

To show how we responded to some of the children's suggestion, here are a few iteration descriptions:

- **Ability to change shape of Klump:** In the first “pilot” test with the Klump in the schools, pulling at the Klump did not make permanent changes to its geometry. This was naturally what we knew we wanted to do, but it is interesting that the children remarked on this on the first meeting. This was implemented before we went into the schools again.
- **Control of texture change:** The textures changed automatically at random times the first time we brought the Klump to the Schools. However, some of the children thought that they could control it. Two children actually believed that if they clicked on the Klump at the same time, then the textures would change, and they worked quite hard at proving their theory. This led us to implement the collaborative texture change objects, which the children really seemed to like and it generated a lot of collaboration and discussion between the children. However, this was an example of enforced collaboration, there was no way for one child alone to change the texture. This will be changed to support one child changing the texture, plus an “special” interesting effect if there is co-operation (encouraged collaboration).
- **Customised Sounds:** The capability to record/playback sound from microphone: This was a suggestion from several of the researchers, since the children were making different noises to try to sound like the different characters they were making. Also, the child users tell a lot of their stories as they are modelling. Therefore, we have implemented this and will put it into future versions of the Klump.
- **Navigation:** Children have asked for methods to view the other side of the Klump. We have responded by implementing a simple navigation tools that allows the user to rotate the Klump and view it from different viewing angles.

4.11 Conclusion

This chapter has presented the Klump Storytelling application for the shared desktop. We have presented a means of creating the existents and structure of stories. Stories are created by working with the Klump object, and greater story structure is stored by working with devices such as the story wheel and story sphere. We have also shown the technical means of creating the shared Klump objects, suggested why we feel this object is evocative and demonstrated some mechanisms that encourage and enforce collaboration.

4.12 References

- [Benford 93] Benford, S. & Fahlén, L. E. 1993. "A spatial model of interaction in large virtual environments". In Proceedings of the third annual European Conference on CSCW (ECSCW'93), Milano, Italy.
- [Benford95a] Benford, S. D., Bowers, J. M., Fahlén, L. E., Greenhalgh, C. M. and Snowdon, D. N., User Embodiment in Collaborative Virtual Environments, Proc. 1995 ACM Conference on Human Factors in Computing Systems (CHI'95), May 7-11, Denver, Colorado, USA, ACM Press.
- [Benford95b] Benford, S., Bowers, J., Fahlén, L. E., Greenhalgh, C., Mariani, J., & Rodden, T. 1995. "Networked virtual reality and cooperative work". In Presence.
- [Fahlén94] Fahlén, L.E., Ståhl, O., Distributed Virtual Realities as Vehicles for Collaboration, Proc. Imagina'94, Monte Carlo, February 1994.
- [Greenhalgh95] Greenhalgh, C. M., and Benford, S. D., MASSIVE: A Virtual Reality System for Tele-conferencing, ACM Transactions on Computer Human Interfaces (TOCHI), 2 (3), pp. 239-261, ACM 14. Kay, A. 1996. "Revealing the Elephant: The Use and Misuse of Computers in Education". Educom Review. July/August, Vol. 31, No. 4, p. 22-28.
- [Hagsand93] Carlsson, Christer and Hagsand, Olof, DIVE-- A Platform for Multi-User Virtual Environments, Computers and Graphics, 1993, vol. 17(6).
- [Hansson97] Hansson, P., Wallberg, A., Simsarian, K.: "Techniques for "natural" interaction in multi-user CAVE-like environments", Poster/Short paper ECSCW '97, 1997.
- [Wallberg98] A. Wallberg, P. Hansson, B. Nord, J. Söderberg, L. E. Fahlén: "The Mimoid and Blob" Projects Presentation/Poster at ACM MultiMedia '98, Bristol UK

5 Designing Storytelling Technologies to Encourage Collaboration Between Young Children⁶

5.1 INTRODUCTION

Collaboration is an important skill for young children to learn. Educational research has found that working in pairs or small groups can have beneficial effects on learning and development, particularly in early years and primary education [Rogoff90, Topping92, Wood96]. Technology offers an opportunity to support and facilitate collaborative learning in many respects [Barfurth95, O'Malley92]. The computer can provide a common frame of reference and can be used to support the development of ideas between children. However, neither learning nor collaboration will occur simply because two children share the same computer [O'Malley92]. Numerous factors must be addressed, not least of which is the learner-machine interface. Today's technology is designed to support either one individual at one computer, or one individual collaborating with another individual at a different computer. However, much if not most, classroom computer use involves pairs or small groups sharing the same computer, especially in primary or elementary schools. What we have come to call *shoulder-to-shoulder collaboration*, as distinct from distributed collaboration, is not well supported with today's interfaces.

In this paper, we explore the design of storytelling technologies to help develop collaboration skills in children aged 5-7 years. This is a particularly interesting group to work with because previous research has shown significant changes in the ability to collaborate effectively within this age range [Wood95]. Young children find it difficult to collaborate effectively. Informal observation of behavior in our project has found that the youngest children (aged 4 and 5) have the most difficulty in working collaboratively and cannot work effectively at all in groups greater than 2.

We introduce an approach to the design of shared interfaces that involves subtly *encouraging* children to explore the possibilities of collaborating, without forcing them to do so. The aim is to provide opportunities for children to discover the positive benefits of working together, for example by being able to create new graphics and effects for their stories.

Encouraging collaboration is more proactive than only *enabling* collaboration. Something new is gained by choosing to work together, although the children may work independently if they wish. On the other hand, it is not as rigid as *enforcing* collaboration, for example by demanding that two children have to synchronize their actions in order to succeed, an approach that has been tried before with some positive gains in terms of individual development [Light97]. The approach of encouraging collaboration is intended to combine the educational goal of learning collaboration skills with our design philosophy of giving children control as much as possible. We also suspect that long-term educational gains might be made when children discover collaboration for themselves.

⁶ This chapter is based on a paper submitted to the ACM CHI 2000 conference

From an HCI point of view, the terms encouraging, enabling and enforcing collaboration can be related to previous approaches to the design of shared interfaces. Early approaches such as “What You See is What I See” (WYSIWIS) enforced strict synchronization of different users’ views onto a shared workspace [Stefik87]. Subsequent approaches such as relaxed-WYSIWIS [Stefik97], coupled with techniques for promoting multi-user awareness [Gutwin98] and concurrency control mechanisms for interleaving users’ actions [Greenberg94] have focussed on enabling the possibility of collaboration while retaining a high degree of individual autonomy. The approach of encouraging collaboration lies somewhere between these two and so offers a new variant on approaches to designing shared interfaces.

The research described here has been carried out within the KidStory project, a collaboration between researchers, classroom teachers, and children (5-7 years old) from England, Sweden, and the United States. The goal of the project is to develop collaborative storytelling technologies for young children. The KidStory technologies are based on the approach of Single Display Groupware (SDG), where several children interact with a single display using multiple input devices, for example, two independent mice [Buxton86, Bier91, Inkpen97, Stewart98, Stewart99]. In its first phase, KidStory has worked with two pre-existing technologies, a shared drawing tool called KidPad [Druin97] and a shared 3D environment called the Klump (an application of the DIVE collaborative virtual environment system [Fahlén93]), both initially with one mouse and later with multiple mice. KidStory has used the methods of cooperative inquiry [Druin99], to involve children as technology design partners in an intergenerational and interdisciplinary design team. To accomplish this, a year-long series of technology design sessions were conducted in two schools in England and Sweden involving more than 100 children.

The following section describes the initial KidStory technologies. We then introduce the approach of designing interfaces to encourage collaboration and describe its use in the redesign of KidPad and the Klump.

5.2 THE INITIAL VERSIONS OF KIDPAD and THE KLUMP

We have been working with two collaborative storytelling technologies, KidPad and the Klump. Both enable two or more children to create and tell stories together, but differ in style, KidPad being derived from drawing and the Klump from sculpting or modeling. In the following we describe them as they were at the start of this research, before being extended to encourage collaboration.

5.2.1 KidPad

KidPad is a shared 2D drawing tool that incorporates a zooming interface. Children can bring their stories to life by zooming between drawing elements (see Figure 5.1). Zooming and spatial structure lie at the heart of KidPad, since they enable children to add narrative structure to their stories by dynamically moving between different parts of a drawing. The creation of a story in KidPad, which involves creating links and zooming between picture/scenes or zooming deeper into the scene, is intended to allow the development of non-linear, complex structured stories. These story representations might make salient the links between scenes and the overall structure of the story. We anticipate that the focus of the children’s attention on these features of the story structure

will provide new opportunities for learning, in a different and complementary way to the creation of a story using more traditional drawing or word-processing packages.

The KidPad interface is designed around a series of graphical “local tools“ that children pick up and apply using a mouse [Bederson96]. The tools are:

Crayons – different coloured crayons can be used to create drawing elements.

Arrow –a selection tool that can pick up and move objects.

Eraser – can be used to delete drawing elements.

Magic wand – can be used to create zooms between different drawing elements. The child selects the drawing element to be the start of the zoom followed by the destination element and sees an arrow linking the two.

Hand – can be used to activate zooms when the story is being told. Selecting the start point of the zoom initiates an animated zoom to the end point.

Turn alive – this tool animates a story element by causing its outline to ripple, making it appear to be alive.

Bulletin Board – this tool enables children to save stories to a bulletin board.



Toolbox – this special tool is used to organize the other tools, and can be opened or closed.

Figure 5.1: A sequence of views in KidPad as we zoom into a simple story (from left to right, and then top to bottom)

KidPad is a Single Display Groupware system, which means that it supports several mice plugged into a single computer. Two or more children can independently grab and use different tools at the same time using their own mice. Any free tool can be picked up and the children see each other's cursors. As a result, this initial version of KidPad could be said to *enable* collaboration – the children can choose to work together or individually. Figure 5.2 shows an example of the KidPad interface.

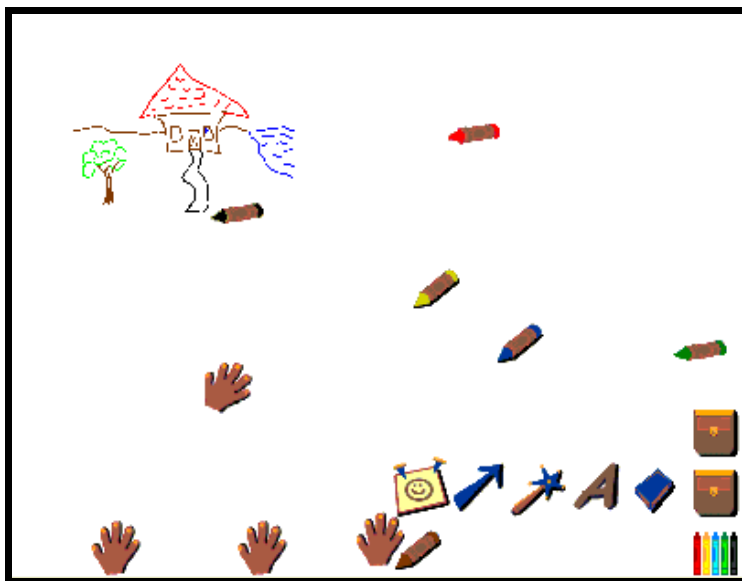


Figure 5.2: The initial version of KidPad showing all the toolboxes open at once with four simultaneous users.

KidPad is built on the Jazz¹ [Bederson99] and MID² open source Java toolkits. Jazz supports Zoomable User Interfaces by creating a hierarchical scenegraph for 2D graphics and MID supports multiple input devices for Java.

5.2.2 The Klump

In contrast to the drawing based approach of KidPad, our second storytelling tool, the Klump is based on a modeling approach. The Klump is a collaborative 3D storytelling tool based around an amorphous 3D object (in fact, a textured deformable 3D polygon mesh) that can be stretched, textured and coloured and that makes sounds as it changes and is manipulated. Figure 5.3 shows an image of the Klump after it has been stretched and textured.



Figure 5.3: the Klump, a deformable 3D modeling object

As with KidPad, two or more children can manipulate the Klump at the same time. The Klump is intended to be a more improvisational storytelling tool than a structured one. Our aim is for the Klump to provide a starting point for generating stories and characters in a way that a blank page sometimes may not. In other words, the real-time exploration of the properties of the Klump might lead to the creation of simple stories. We also intend that the flexible and amorphous nature of the Klump might inspire a wide range of different stories. Again, by supporting synchronous multi-user access and by displaying the children's cursors to one another, the Klump enables collaboration. The initial version of the Klump can be manipulated in the following ways:

Stretching – a point on the surface of the Klump can be grabbed using the mouse and can be pulled to deform its shape. There is an option to switch between pulling a single vertex and a group of vertices, thereby changing the kind of deformation that occurs. The single vertex option pulls out a thin volume of the Klump, whereas the group of vertices pulls out a thick volume. There is also a button to return the Klump back to its original spherical shape.

Texturing – a variety of pre-defined textures may be applied to the surface of the Klump by selecting buttons on the interface. These textures allow different facial expressions to be added to the front side of the Klump, giving it a sense of character, and enable its background colors to be changed.

Rotating – the texture on the surface of the Klump can be grasped and rotated around to a new position.

Finally, the Klump makes a variety of sounds to reflect these different manipulations.

5.3 INTERFACES TO ENCOURAGE COLLABORATION

The core technical innovation of this paper is the idea of designing interfaces to encourage or invite children to collaborate. This has been motivated by our experiences of using the initial versions of KidPad and the Klump in two schools, one in Sweden and one in England, during the 1998-1999 school year as part of a program of activities that included:

- **contextual inquiry** – sessions to observe how children work with existing storytelling technologies (e.g., crayons and paper) and how they collaborate.
- **participatory design** – initial sessions to establish the children in the role of design partners and co-inventors of technology, followed by sessions with KidPad and The Klump aimed at eliciting specific design suggestions. These are reflected in the redesign of these technologies described later on.
- **evaluation of the technologies** – observations of how the children used the initial versions of KidPad and the Klump.

Over the course of the year, the combination of these activities has resulted in more than fifty sessions in schools involving more than one hundred five and seven year olds. At the peak of this activity, there were weekly participatory design and contextual inquiry sessions.

Children were observed with respect to collaborative behavior and their ability to use the technology to tell stories. Children and teachers were encouraged to provide feedback on these

technologies that would instigate changes in design. Although after a few months, small-group and whole-class collaborative storytelling activities were being performed using these technologies, it was evident that some children found collaborating difficult.

Interfaces that encourage collaboration were proposed as a way of addressing this problem. Such interfaces should provide opportunities for children to discover the positive benefits of working together. Ideally, this should be achieved in as subtle and natural a way as possible, avoiding forced solutions. As noted in the introduction, encouraging collaboration is more proactive than only *enabling* it as was the case with the initial versions of KidPad and the Klump described previously. On the other hand it is not as extreme as strictly requiring collaboration, for example, demanding that two children have to press a button together to achieve an action, the approach that we described as “enforcing collaboration”.

In its strictest interpretation, the approach of encouraging collaboration without enforcing it would require that a single child could achieve on their own any action that two children could achieve together, but that the two would do so in an easier, more efficient or more fun way. However, a more relaxed interpretation, is that a single child can carry out all of the major classes of action supported by the tool, but that by working together, two children can achieve subtle extensions to and variations on these actions. For example, a single child or two children working independently can create a functioning drawing in KidPad, but two children collaborating can create an enhanced one. This more relaxed approach is the one that we have adopted in revising KidPad and the Klump. However, before describing their redesign, we briefly digress to consider the more general relationship between the approach of encouraging collaboration and previous work on the design of shared interfaces in some more detail.

5.3.1 Relationship to previous work on shared interfaces

Up to now, we have introduced the idea of interfaces that encourage collaboration within the context of educational applications. We now consider its broader relationship to CSCW technologies, especially how it compares to other approaches to synchronizing shared interfaces

How to synchronize shared interfaces has been a major concern for CSCW research. This has predominantly focused on distributed groupware where multiple users share a common workspace, for example a shared document, 2-D sketch tool or 3-D virtual world, using separate displays connected over a computer network. In such cases, the problem of synchronization can be broadly broken down into two parts.

How to synchronize what different users see? One of the first approaches was WYSIWIS (What You See Is What I See) where different users at different displays were forced to see the same part of a virtual workspace [Stefik87]. Experience with WYSIWIS led to less strictly coupled approach called relaxed WYSIWIS where different user’s views could diverge [Stefik97]. Systems adopting this approach typically introduce additional functionality to support users in being aware of where others are looking and what they are doing. This may take the form of various awareness widgets, such as ‘radar views’ in 2D workspaces [Gutwin98] or visible user embodiments (‘avatars’) in 3D

systems [Fahlén93].

How to synchronize object manipulations? Many CSCW systems allow users to collaboratively manipulate objects, changing their state. Examples include jointly editing a shared document or grasping and moving objects in a virtual world. This raises the problem of how to prevent conflicting updates. The most common solution is some form of locking, including simple turn-taking protocols, optimistic locking, non-optimistic locking and serialization protocols that allow participants to interleave their actions at various granularities [Greenberg94]. Another option is social locking where given sufficient mutual awareness, user's may be able to negotiate mutual access with minimal system intervention.

We suggest that these various strategies can be located along a “collaboration continuum” according to the extent to which they constrain individual autonomy and demand collaboration or leave users free to act independently. One extreme of the continuum involves what we have called *enforcing collaboration*, where the users are locked in step with one another. WYSIWIS and strict turn-taking can be found here. So can the work of Light, Foot and Colbourn, who modified the input of a standard computer so that two students had to enter information at the same time to succeed [Light97]. A kind of dual key control was used. It was found that this enforcement of collaboration improved individual cognitive development. At the other extreme is what we have called *enabling collaboration*, where the users can act independently, are mutually aware and are free to coordinate their actions if they wish. Relaxed-WYSIWIS and social locking can be found here.

Our approach of *encouraging collaboration* lies somewhere between the two. It is not so strict as to require users to work together, but it provides some explicit motivation for them to do so in terms of added benefit. As noted earlier, encouraging collaboration can be interpreted in different ways. The case where a single user could achieve any action, but multiple users can achieve it in a way that is easier or more fun lies towards the enabling end of the continuum. The case where a single user can carry out each general class of action, but where multiple users can achieve enhanced actions lies towards the enforcing end.

It should be noted that a single CSCW system can use different approaches for different actions. For example, collaborative virtual environments often enable collaboration for viewpoint control (each user steers their own viewpoint, but is made aware of others' viewpoints through their embodiments), but enforce it for object manipulation (there is a turn-taking or coarse locking protocol regarding who can grab a virtual object).

This discussion raises the question of how the approach of encouraging collaboration might be applied in areas other than educational applications. One possible application area is in entertainment and games applications where participants might choose to collaborate, pooling abilities and resources to mutual benefit. Another more subtle approach might be in situations where participants can benefit by sharing costs. People increasingly have to pay for the use of network resources, for example in video and audio streaming. Users who agree to collaborate, for example to receive or manipulate the same information might be rewarded by sharing the costs between them.

5.4 REDESIGNING KIDPAD AND THE KLUMP TO ENCOURAGE COLLABORATION

We now describe how KidPad and the Klump were redesigned according to the lessons learned from the various schools sessions. Our overall strategy was to introduce design changes that satisfied two criteria:

- first they should encourage collaborative activity, reflecting the project's educational agenda and reacting to the observations noted previously.
- second, they should be based on the children's own design suggestions, emerging from the cooperative inquiry process.

Our general approach has been to use the more frequently occurring of the children's ideas as the basis for deciding on new functionality, but to realize this functionality through the approach of "encouraging collaboration".

5.4.1 Redesign of KidPad

The basic approach that we followed in redesigning KidPad to encourage collaboration was to support tool "mixing". By this, we mean that when two (or sometimes more) children each use mixable tools at about the same time and place, the tools give enhanced functionality.

As a concrete example of this approach, consider the operation of the crayons in KidPad. The initial version provided three colors. A frequent design suggestion from the children was to provide more colors. We immediately added three more crayons, but that wasn't enough. Our final solution is to enable children to collaborate and combine their crayons to produce new colors. If two children draw with two crayons close together, then the result is a filled area between the two crayons whose color is the mix of the two. In this case, the children are not prevented from drawing as individuals, but they can gain additional benefit (new colors and filled areas) by working together.

Applying our approach involves examining combinations of actions to look for interesting benefits and effects. We can consider all actions combined with themselves, for example, what happens when two selection tools are used together in KidPad? We can also consider how actions combine with other actions, for example, what might happen if one child rotates the Klump while another stretches it? In each case, we look for effects that are natural and useful rather than contrived.

As described above, crayons in KidPad now work this way by drawing a filled in area between the two crayons using a color that mixes the two crayon's colors. By introducing collaborative color mixing, we added 15 mixed colors with the six crayons, and filled areas while encouraging collaboration and without adding any new tools. (see Figure 5.4). Also, we added a special "duplicating" tool that makes copies of other tools so several children could use the same tool type simultaneously. Figure 5.4 shows the redesigned interface with two children using mixed crayons.

We built in mixing capability for multiple uses of all tools, except the magic wand and toolboxes. In every case, we tried to add a special behavior that acts as if it is a natural extension from the behavior with a single user. We felt this design ideal to be important in order to make it as easy as possible for children to anticipate what the mixed behavior might be. The mixing behavior we added is:

Crayons – As described above.

Arrow – Two or more children can squash and stretch selected drawing objects.

Eraser – One user can erase bits of a drawing object, but two children can erase an entire drawing object at once.

Hand – Two or more children can zoom in and out by moving their hands apart, or closer together, respectively.

Turn Alive – Two or more children can control the animation properties of a wiggling object by moving the turn alive tools closer together or further apart.



Figure 5.4: Redesigned KidPad interface with mixed crayons being used. Note that inactive tools are faded. There are three active crayons, and two are currently being used to create a “mixed” area.

5.4.2 Redesign of the Klump

In redesigning the Klump to encourage collaboration, we have focused on combining the actions of stretching and texturing with themselves.

Stretching – the initial version of the Klump enabled toggling between two modes of stretching, pulling out a single vertex and pulling out a group of vertices. The revised version enables a single child to pull out only a single vertex on their own. However, if two children synchronously pull out two vertices that are close together on the Klump’s surface, the result is to pull out a whole group of vertices. Thus, the added benefit of collaborating is to be able to make a different shaped deformation.

Texturing – our redesigned version of the Klump enables the children to apply a limited number of textures to its surface by pressing buttons. The textures represent happy and sad faces as well as background textures for the three primary colors. These may be applied independently so as to combine each of the two faces with the three background colors. However, by pressing some

buttons together, the children may arrive at new combined textures. Three new faces become possible: laughing (pressing happy and happy), a kind of surprised expression (pressing happy and sad) and crying (pressing sad and sad). In addition, the background colors can be selected together to make new combined colors (similar to combining the crayons in the revised KidPad). A single user can also select the combined textures by selecting one button and then another a short time after (while the first is seen to rotate), but it requires speed and skill.

We have also extended the sounds made by the Klump to provide feedback as to when collaborative effects are being triggered, for example, by saying “cool” and “yippee”.

Figure 5.5 shows the revised Klump interface. In the center we see the Klump, currently with its laughing face on a red background. To its left are the two buttons that are used to apply happy and sad face textures. To its right are the three buttons for applying the colors. Above the Klump are two buttons that toggle between using a mouse for stretching and using it for rotating. The red button at the bottom returns the Klump to its original shape.

Figure 5.6 shows the difference between single-user and collaborative stretching. On the left we see the results of a single user stretching the Klump, pulling out a single vertex. On the right we see a collaborative stretch that pulls out a group of vertices, making a larger deformation.

Figure 5.7 shows the different facial expressions that can be obtained using the two buttons at the left of the interface. Faces 1 (happy) and 2 (sad) are obtained by a single user pressing the button. Faces 3 (laughing), 4 (surprised) and 5 (crying) are obtained when two users select combinations of the buttons at once (happy and happy gives laughing, happy and sad gives surprised, sad and sad gives crying).

5.4.3 Initial reflections on the revised interfaces

Although no formal program of evaluation has yet been carried out, the revised versions of KidPad and the Klump have been tested with a few groups of children.

The revised version of KidPad was introduced to our school in Nottingham. Pairs of children were given the common goal of recreating a well-known nursery rhyme. The children appeared to collaborate effectively, working on separate parts of the story and then joining together to use the collaborative tools to color in their picture.

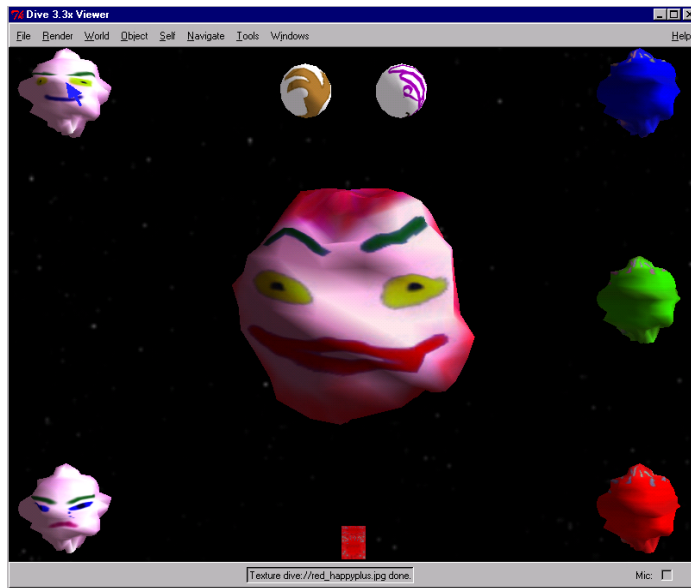


Figure 5.5: the revised Klump interface

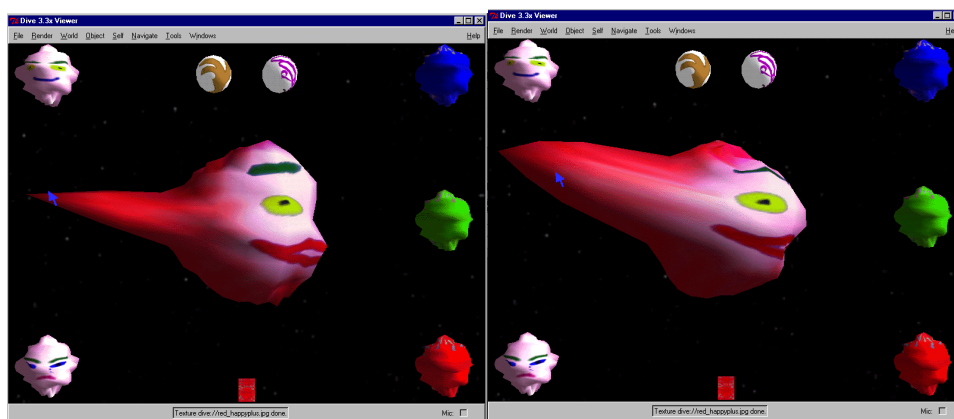


Figure 5.6: single user and collaborative stretching

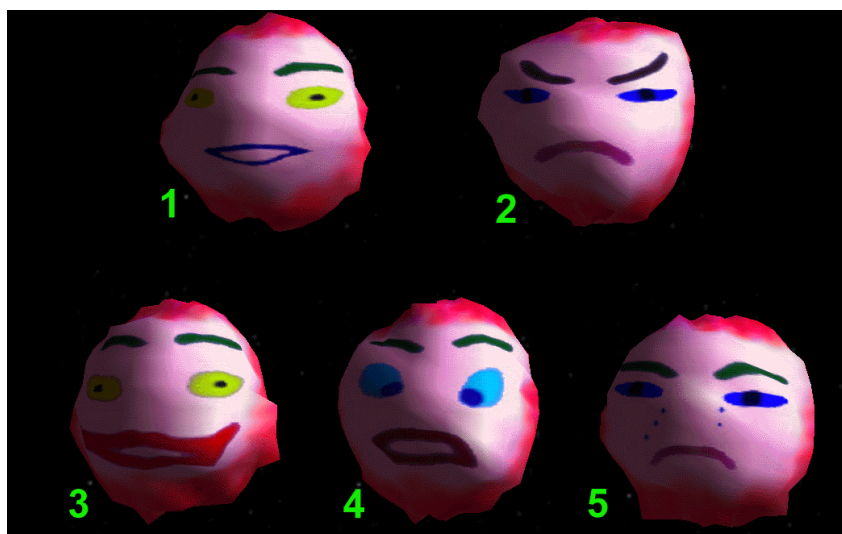


Figure 5.7 : facial expressions for the Klump

Two children from the UK tested the re-designed version of the Klump. While the children

explored features of the Klump, including the collaborative features, they did not show much interest in working together. This may in part, have been the result of them having no explicit ‘shared goal’. This session, however, did raise an issue that should be considered when developing tools to encourage children’s collaboration. When two young children carry out a collaborative action, the resulting effect has to be really obvious and noticeably different from the effect displayed when the children carry out the action independently.

The revised versions of both KidPad and the Klump were also informally tested with a small group of children that are design partners at the University of Maryland’s Human-Computer Interaction Lab. This formative evaluation showed that it took considerable experience with KidPad and the Klump for children to make use of the collaborative tools. For example, in a one-hour session where two boys (ages 10 and 8) used the Klump, it took almost 25 minutes for the children to discover the collaborative features. (These children on a previous occasion had used a less collaborative version of the Klump for a twenty minute session). They were then shown the collaborative features by an adult. In their comments afterwards said that they had enjoyed changing the faces and mixing colors.

Another formative study was carried out with six children (4 boys/2 girls; ages 7-10) using KidPad. For an hour and a half session, the three children who had previously worked with KidPad (a single-mouse version) showed strong differences in their use of collaborative tools, than the three other children who had never seen KidPad before. The children formed two teams, and each team worked on a computer with three mice. The children that already had used KidPad formed one group, and the children that hadn’t used KidPad formed another group. After introducing KidPad and the new collaborative tools to the group, the children freely explored the tools for 20 minutes. Then, the children were asked to create a story with at least three “scenes” to zoom to and from. The experienced children had little trouble creating a story. They collaborated throughout the process, making extensive use of the collaborative tools before starting the story, trying out the different possibilities. However, interestingly enough, they did not use the collaborative tool behaviors in the actual story creation.

The children that used KidPad for the first time had a harder time collaborating to create a story. They tended to experiment with the tools, including the collaborative tool behaviors. Most of what they did however was scribbling. This group found it hard to identify each other’s cursors and to negotiate collaboration.

These early observations suggest that young children are able to use some of the collaborative features of KidPad and the Klump and that they can enjoy doing so. On the other hand, the way these features work has to be made more obvious in some cases. Furthermore, discovering them in the first place is a problem and they had to be pointed out by an adult on several occasions. On reflection, we realize that our designs only showed the results of collaborating, but did not highlight in advance when the possibility existed. We have therefore begun to revise KidPad and the Klump to more explicitly show the potential to collaborate. An example of this can be seen in Figure 5.4 that is actually taken from the most recent version of KidPad. The two dots above the crayons are eyes that only appear when the crayons are close enough for the color mixing and filling to happen. We hope that steps such as these will help the children discover collaborative possibilities for

themselves.

5.5 SUMMARY AND FUTURE WORK

In summary, we have proposed a new approach to designing shared interfaces that is intended to support children in learning to collaborate. The approach, called encouraging collaboration, allows children to work as individuals, but gives added benefits if they choose to work together. We have demonstrated this approach applied to the design of two storytelling technologies within the more general framework of cooperative inquiry within UK and Swedish schools. We have compared our approach with other user interface mechanisms from CSCW.

Future work will involve further design changes to KidPad and the Klump to reflect our early experiences. We will then undertake a more rigorous programme of evaluation including the development of a more intricate coding system, focusing on verbal and non-verbal collaborative behaviors, tracked from video recordings of the children and computer tracking of the children's interactions.

5.6 REFERENCES

- [Barfurth95] Barfurth, M.A. (1995) Understanding the Collaborative learning process in a technology rich environment: The case of children's disagreements. *Proc CSCL 1995*
- [Bederson99] Bederson, B. B., & McAlister, B. (1999). Jazz: An Extensible 2D+Zooming Graphics Toolkit in Java., *University of Maryland Computer Science Tech Report #CS-TR-4015*.
- [Bederson96] Bederson, B. B., Hollan, J. D., Druin, A., Stewart, J., Rogers, D., & Proft, D. (1996). Local Tools: An Alternative to Tool Palettes. *UIST 96*, pp.169-170.
- [Bier91] Bier, E. A., & Freeman, S. (1991). MMM: A User Interface Architecture for Shared Editors on a Single Screen. *UIST 91*, pp. 79-86.
- [Light97] Light, P., Foot, T., and Colbourn, C., (1997) Collaborative interactions at the microcomputer keyboard. *Educational Psychology*, 7, 1, 13-21.
- [Buxton86] Buxton, W., & Myers, B. A. (1986). A Study in Two-Handed Input. *CHI 86*, pp. 321-326.
- [Druin99] Druin, A., (1999) Cooperative Inquiry: Developing New Technologies for Children with Children. *CHI'99*, pp. 223-230.
- [Druin97] Druin, A., Stewart, J., Proft, D., Bederson, B. B., & Hollan, J. D. (1997). KidPad: A Design Collaboration Between Children, Technologists, and Educators. *CHI 97*, pp. 463-470.
- [Fahlén93] Fahlén, L. E., Brown C. G., Stahl, O., Carlsson, C., (1993) A Space Based Model for User Interaction in Shared Synthetic Environments, *InterCHI'93*
- [Greenberg94] Greenberg, S. & Marwood, D. (1994) Real Time groupware as a Distributed System: Concurrency Control and its Effect on the Interface, *CSCW'94*
- [Gutwin98] Gutwin, C. & Greenberg, S., (1998) Design for individuals, Design for Groups:

Tradeoffs between Power and Workspace Awareness, *Proc, CSCW'98*, 207-217.

[Inkpen97] Inkpen, K., Booth, K. S., Klawe, M., & McGrenere, J. (1997). The Effect of Turn-Taking Protocols on Children's Learning in Mouse-Driven Collaborative Environments. *In Proc Graphics Interface (GI 97)* Canadian Information Processing Society, pp. 138-145.

[O'Malley92] O'Malley, C (1992) Designing Computer Systems to support peer learning, in *European Journal of Psychology of Education*, Vol. VII, No. 4, 339-352.

[Rogoff90] Rogoff, T. (1990) *Apprenticeship in Thinking: Cognitive development in social context*. Oxford University Press, Oxford.

[Stefik97] Stefik, M., Bobrow, D., Foster, G. Lanning, S., & Tatar, D., (1997) WYSIWIS Revised: Early experiences with multi-user interfaces, *ACM TOIS*, 5(2), 147-167.

[Stefik87] Stefik, M., Foster, G., Bobrow, D., Kahn, K., Lanning, S. & Suchman, L., (1987) Beyond the Chalkboard: Computer Support for Collaboration and problem Solving in Meeting, *CACM*, 30(1), 32-47.

[Stewart99] Stewart, J., Bederson, B. B., & Druin, A. (1999). Single Display Groupware: A Model for Co-Present Collaboration. *CHI 99*, pp. 286-293.

[Stewart98] Stewart, J., Raybourn, E., Bederson, B. B., & Druin, A. (1998). When Two Hands Are Better Than One: Enhancing Collaboration Using Single Display Groupware. *CHI'98 Extended Abstracts*, pp. 287-288.

[Topping92] Topping, K. (1992) Cooperative learning and peer tutoring: An overview. *The Psychologist*, 5(4), 151-157

[Wood96] Wood, D. & O'Malley, C., (1996) Collaborative learning between peers: An overview. *Educational Psychology in Practice*, 11(4), 4-9.

[Wood95] Wood, D., Wood, H., Ainsworth, S. & O'Malley, C., (1995) On becoming a tutor: Toward an ontogenetic model. *Cognition and Instruction*, 13(4), 565-581.

6 Conclusion and future directions

This deliverable D1.1 has presented the year 1 work in developing the shared storytelling platform. We have presented the two applications representing the two complementary directions we have explored in developing storytelling tools for children. In this first year, the primary achievement, and goal has been to develop storytelling tools that allow multiple children to sit side by side, and collaborate – a form of collaboration called “shoulder-to-shoulder” collaboration.

6.1 Summary of tool themes

The themes of exploration have been:

- **Local Tools** - an alternative user interface approach which replaces pull-down menus and tool palettes. Following the experience with KidPad, the Klump application has adopted the Local Tools approach over a menu and palette approach common in many GUIs. However these Local Tools are extended into the 3D environment of the Klump environment. In this way, the cursors are 3D objects as well as mechanisms for selecting the 3D cursors.
- **Zoomable User Interfaces** - the basic "canvas" that the stories are created on are *zoomable*. This means that children can create stories that can be zoomed into for more detail, and the zooming can in fact become a fundamental part of the story.
- **Single Display GroupWare** - support for multiple children simultaneously using a single computer, each with their own mouse
- **Storytelling authoring software** - the basic idea that children can learn communication skills by creating and telling stories.
- **Gestural modelling:** In interaction with the Klump, there is a strong notion of using “mouse gestures” to model both the shape of the Klump object as well as the colour of the textures, and the sound that is produced. What is meant by gestures is the movements of the mouse. In this way, users can pull out parts of the Klump surface as well as shape the sounds emitted from the Klump.
- **Subjective interaction:** Input device interaction among simultaneous users can differ. For example, while one user is moving or colouring the textured surface, another user can be modelling the Klump’s shape. Subjective interaction requires users being made aware of the mode of their input device when using it.
- **Multimodal output and interaction:** The Klump employs different modal outputs. These are visual, including shape and texture, as well as aural, including modulated midi sounds. It is believed that it is (at least partly) this coupling of modalities with the dynamic behaviours that make the Klump an engaging focus of attention.
- **Storytelling 3D shapes and structures:** The Klump provides mechanisms for shaping and

forming shapes that can be used for storytelling. The Klump's organic movements, abstract textures, and sound tend to create an environment where child users are given ideas for stories. Interaction with the Klump tends to be engaging for many users. Through this interaction, ideas for stories tend to spring out. We call these "blind" offers (they are blind in the sense that, as yet, the application is not aware of story context). In this way, the Klump provides a mechanism for facilitating an improvisational storytelling between the children working with it.

- **Children as design partners** - children will learn more, and the technology will be most appropriate if children are closely involved in the design of the technology.

6.2 Toward WP1.2

As we enter year two, we have begun to think and work on WP1.2 Shared Storytelling objects. In WP1.2 the project begins to explore methods to work with tangible interaction methods and alternatives to monitors on desktops for output. Along these lines, an analogue to the MID architecture presented in chapter 3, Multiple Output devices (MOD) will be explored. School sessions which involved the whole class suggested that wall-sized displays have greater sharing affordances than desktop monitors. This may be an obvious observation, but is leading us to explore other ways of sharing displays. An example of such a method would be a "campfire" display which would enable a group of children to sit around a display projected onto a central surface. Such a display, without any clear notion of "front and back" might have different interactional and participatory affordances. Notions of modelling inherent in the Klump application suggest that a squeezable physical interface might prove successful. Ideas for such an object centre around a beach ball sized sphere that has sensors for rotation and touch-pressure. This device would be big enough to be shared and provide for modelling objects such as the Klump.

7 Appendix A –Research projects and commercial products related to KidStory

7.1 Introduction

During year one this short review of projects and commercial products related to KidStory was collected. The main reason for doing this was to ensure that technology features present in other projects weren't duplicated in KidStory and to give the research team an idea of what other projects in the same area of research were doing. The project review consists of project proposal summaries, methodology summaries and a list of possible implications for KidStory. The commercial product review consists of product descriptions and a list of possible implications for KidStory.

7.2 Project Review

Below are some of the projects that have “children and narrative” in some form as a primary theme of the project. Where projects are involved with or funded by I3 this is noted.

7.3 Puppet – I3

Summary

The Puppet project tries to design a system that supports early learning by making use of knowledge about how children learn.

Methodology

In practice, the work involves creating a Virtual Puppet Theatre where children can construct, edit and run interactive plays. The project considers this to be a new form of computer-literacy. The project also aims to create possibilities for manipulating characters, roles and plots in ways not possible in physical plays. The first two years of the project focus on creating a single-user theatre while the third year adds multiple puppeteers. The work is inspired by ethnographic observations in the schools.

Implications for KidStory

This project also risks replacing a real-life activity with a virtual activity, although the project is too young to give details on the intended functionality of the virtual puppet theatre. A key difference between this work and the work in KidStory is in the methods for involving children. In Puppet, children are primarily seen as “informants.” In KidStory a strong attempt is to construct a relationship of “design partners.” This difference has been driven home elsewhere.

7.4 Today's Stories –I3

Summary

The main idea of this project is that children may learn from reflecting on their own actions and from other children's perspective on their actions. The goal is to devise a means of collecting “stories” (events, reflections, snapshots) from day-to-day life.

Methodology

The project is building a wearable device, KidsCam, that can be used to record short audiovisual sequences. The recorded sequences are then collected, annotated and presented in a multimedia environment. In the end, these multimedia documents constitute a diary of events that isn't tied to a particular child but presents the story from a fresh perspective.

Implications for KidStory

The idea of recording sound and video to make it possible to experience a story again at a later time (or share it with others) have been discussed at several occasions. This might prove a useful area for further research. Another aspect of this project is the use of everyday tangible interfaces. In KidStory we are trying to steer away from artefacts which are “computery” but yet retain the power that computer devices bring.

7.5 Pogo –I3

Summary

The idea of the Pogo project is to build a virtual world that supports storytelling. The world contains characters, props and tools. Different methods of "collecting" story elements from children will be used. The elements are assembled and re-told in the virtual world. At the core of the theoretical framework of the project lies observations of children's storytelling and discussions with the teachers.

Methodology

Specific details of the project are not yet available, but the project has discussed four different approaches for the virtual world:

- A playground without rules or role descriptions.
- A story puzzle. This appears to be similar to "Myst" or "Riven", computer games where the player discover different fragments of a story and have to figure out the "big picture".
- Role playing. Presumably, this means that the child observes a scenario in the virtual world and then tells a story about it.
- A story chain letter. Here, the idea seems to be to use the virtual world to create a story together.
- Also, popular characters from the virtual world will be brought into the real world through

tangibles.

Implications for KidStory

The Pogo world may turn out to be restricted - users cannot create new characters or new worlds. The work in Pogo wants to strongly integrate with existing in-school storytelling and drama practise. There are a number of good pedagogic researchers involved with school observations. The KidStory project should aim for "open" generic tools in the sense that the tools do not dictate what a user can do.

7.6 The Virtual Theatre Project

Summary

The goal of the Virtual Theatre project is to create a multimedia environment where the user can assume all of the roles associated with producing and performing plays and stories. The focus is on "improvisational" theatre. A user-provided script can be performed by the user together with computer-controlled actors that have human-like features such as the ability to improvise and respond to the actions of other actors on the set. The main focus is on the computer-controlled actors.

Methodology

This work builds strongly on work in autonomous agents. In addition to providing characters with "script"-like frameworks for behaviour, the project also builds on research in making virtual characters "lifelike." This project comes close to bringing along a lot of the problems with agent-oriented interfaces. This is the subtle difference between endowed and intrinsic behavior and where the creative (e.g. intelligent) inspiration and play come from – the characters or the tools. Children author "improvised" scenes by moving sliders of emotions and state (e.g. happy, shy, tired..).

Children have been involved in user testing, but whether the tool is driven by technology or through a needs process is unclear.

Implications for KidStory

One of the intended features of the Klump tool is the possibility to provide "set and setting." Thus the Klump tool might make offers that are subtle yet powerful enough to generate creative activity (e.g. producing hues of blue while making sounds of the sea). It is not desirable to use such a comprehensive. Many projects have followed this idea of the "virtual stage" to the point where the metaphor is becoming a bit tiring. The point of KidStory however is to move the stage of interaction out into the real world from the virtual world.

7.7 The NICE Project

Summary

NICE is an acronym for Narrative-based Immersive Constructionist/Collaborative Environments. The main point is to let the children build collaborative artefacts. It is hoped that this will engage the children in "authentic activity".

Methodology

In reality, the system revolves around a virtual island where children interact with different ecosystems, such as trees and flowers. When a seed has been planted, it needs supervision to grow (if it doesn't have enough water, you can bring out a rain cloud, for example). The events that take place on the island are recorded and presented in plain English (with illustrations) on the WWW. Several users can work simultaneously in the world. A CAVE (a cube with projection screens on three or more of the sides) is used for the visual presentation of the island. At the moment, the user interface consists of a 3D-mouse, but the intention is to try other ways of communication. Whether or not the children can create the existents and events of the story is unclear.

Implications for KidStory

When designing technology for children (and indeed for users of any age), it is important to ensure that the technology *add to* or *enhance* a real-life activity, rather than *replace* a real-life activity. The NICE project description implies that the NICE project does the opposite: it replaces the real-life activity of planting a seed and observing its growth with a virtual activity (i.e. in what way does planting a virtual seed enhance or add to the activity of planting and caring for a real physical plant?). Therefore, KidStory will most likely avoid activities similar to the ones being pursued in the NICE project.

7.8 The Playground Project –I3

Summary

The Playground project is building "Animated Playgrounds for Learning". One of the the main goals is to create a virtual playground that introduces (or teaches) formal thinking and computer programming by making use of metaphors that children can understand. The intended user interface, ToonTalk and OpenLogo, feature animated characters, speech, (virtual) personal assistants, gestures in Virtual Reality, and tactile interaction.

Methodology

Two implementations of the playground are being built, one based on an existing product called ToonTalk, and one based on OpenLogo, a system that will be built by the Playground project. The functionality of the systems will be evaluated by comparing them to one another.

The children have no direct influence over the design of the Playground systems. Instead, studies of child learning and social interactions guide the design decisions. The children's games, their game creation and the learning outcomes from the systems will be evaluated. The schools participating in

the project are situated in Sweden, the UK and Portugal.

Implications for KidStory

Any system containing animated characters will probably invite children to tell stories about it and ToonTalk offers a rich variety of storytelling opportunities. However, Playground is not explicitly concerned with storytelling. Instead, storytelling is regarded as one of the possible side-effects of using the system.

In a storytelling context, the Playground systems may be thought of as systems where you can "program" a story. That is, the story is defined and told by making use of a programming language. The KidStory tools are more direct: the user use gestures to indicate what he or she wants to do.

Another interesting assumption that Playground makes is that children can understand and learn how to use formal systems (that is, systems such as programming languages), and that a good way to teach them is through visual metaphors and through user participation. In a way, everyday languages such as English or Swedish are also formal systems, so perhaps the jump to other formal systems isn't so big.

7.9 Commercial products related to KidStory

Below is a collection of products that have "children and narrative" as a theme.

7.10 PuppetTime

Summary

PuppetTime is a new QuickTime media type that is used to control virtual puppets on a stage. There is a program called "PuppetTime Director" that can be used to create plays. This is done by placing events such as facial expressions, dialogue and movement onto a time line. The result can be saved either as a standard QuickTime movie or, if the end-user has a PuppetTime extension to QuickTime, as a PuppetTime stream.

Implications for KidStory

Using time lines for controlling the Klump on a virtual stage has been discussed at several occasions. However, the time-line concept may be difficult for young children to understand and use.

7.11 Zowie

Summary

The main idea of the Zowie toys is to offer entertainment and learning experiences by allowing the children to control the actions on the computer screen by manipulating a physical toy. The

computer keyboard is not used at all. Unfortunately, details on how the product works are not available, but presumably the physical toy is plugged into the serial port of the computer - it's not a mere keyboard overlay. There is a nice philosophy here of building computer environments to help augment the experience of using the tools by providing a backdrop narrative for play. Whether children can create and explore their own narratives is not yet clear.

Implications for KidStory

The Zowie products and the Tonka Workshop are very limited in their functionality. Once all features have been discovered the entertainment experience becomes repetitive. In other words, the tools are, as far as can be seen now, exploratory rather than creative. It is important that the KidStory tools are tools for collaborative creation, not merely exploration.

7.12 Orly's Draw-a-Story CD-ROM

Summary

Orly is a girl who lives on the island of Jamaica with her pet frog. Together they tell a set number of stories. At certain points in this story, there is a pause and a request that the child draws a part of the story (an existent). These created existents then appear in the rest of the story.

Implications for KidStory

As with many of Broderbund's titles, the finishing touches are done well. This CD-ROM goes beyond the typical "interactive book" in that it allows the user to create elements of the story. However the structure is fixed and there is no facility to combine and create story elements (e.g. new events, plots, etc). KidStory specifically aims to go beyond these experiences and empower children as authors. The toolbox for drawing is excellent however and a lesson on how to create evocative characters without spilling into the pitfalls of creating agents.

7.13 LEGO MindStorms

Summary

MindStorms is a new product line from LEGO. The core set is called Robotics. At the heart of Robotics is a LEGO piece called RCX that contains a computer that can be programmed by using a regular personal computer. Robotics also includes sensors and motors. Together they can be used to create a physical autonomous model that can move and sense its surroundings. It is also possible to program simple behaviours into the model. The RCX is programmed by connecting blocks (they look like jigsaw puzzle pieces) on the computer screen and then downloading the program using an IR device.

Implications for KidStory

The idea of bringing something out of the computer into the real world is attractive. Most computerized toys work the other way: a physical device is used to control something inside the computer. It would be very interesting to see an interface that does both

7.14 References

Lego, Inc.,

<http://www.legomindstorms.com/>

Myst, computer game, Cyan Software, Inc.. Distributed by Broderbund Software,

<http://www.broderbund.com/>

The NICE Project,

<http://www.ice.eecs.uic.edu/~nice/NICE/aboutnice.html>

The Playground Project,

<http://www.algonet.se/%7elmogren/PlaygroundHome.html>

The Pogo project,

No WWW page available, information taken from I3 workshop notes.

The Puppet Project,

<http://www.vision.auc.dk/LIA/puppet/>

PuppetTime Inc.,

<http://www.puppettime.com/>

Riven, computer game, Cyan Software, Inc.. Distributed by Broderbund Software,

<http://www.broderbund.com/>

Today's Stories,

<http://stories.starlab.org/>

The Virtual Theatre Project,

<http://ksl-web.stanford.edu/projects/cait/index.html>

Zowie Entertainment, Inc.,

<http://www.zowiepower.com/products/index.html>