A NEW USER INTERFACE FOR CLONING OBJECTS IN DRAWING SYSTEMS

LOUTFOUZ ZAMAN

A THESIS SUBMITTED TO THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
Masters of Science

GRADUATE PROGRAMME IN COMPUTER SCIENCE
YORK UNIVERSITY
TORONTO, ONTARIO

APRIL 2010
A NEW USER INTERFACE FOR CLONING OBJECTS IN DRAWING SYSTEMS

by Loutfouz Zaman

A thesis submitted to the Faculty of Graduate Studies of York University in partial fulfilment of the requirements for the degree of

MASTER OF SCIENCE
© 2010

Permission has been granted to: a) YORK UNIVERSITY LIBRARIES to lend or sell copies of this thesis in paper, microform or electronic formats, and b) LIBRARY AND ARCHIVES CANADA to reproduce, lend, distribute, or sell copies of this thesis anywhere in the world in microform, paper or electronic formats and to authorise or procure the reproduction, loan, distribution or sale of copies of this thesis anywhere in the world in microform, paper or electronic formats.

The author reserves other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without the author’s written permission.
A NEW USER INTERFACE FOR CLONING OBJECTS IN DRAWING SYSTEMS

by Loutfouz Zaman

By virtue of submitting this document electronically, the author certifies that this is a true electronic equivalent of the copy of the thesis approved by York University for the award of the degree. No alteration of the content has occurred and if there are any minor variations in formatting, they are as a result of the conversion to Adobe Acrobat format (or similar software application).

Examination Committee Members:
1. Scott MacKenzie (Chair)
2. Wolfgang Stuerzlinger (Supervisor)
3. Jian-Guo Wang (Outside Member)
4. Parke Godfrey (Dean’s Representative)
ABSTRACT

Cloning objects is a common operation in graphical user interfaces. One example is calendar systems, where users commonly create and modify recurring events, i.e. repeated clones of a single event. Inspired by the calendar paradigm, we introduce a new cloning technique for 2D drawing programs. This technique allows users to clone objects by first selecting them and then dragging them to create clones along the dragged path. Moreover, it allows editing the generated sequences of clones similar to the editing of calendar events. Novel approaches for the generation of clones of clones and interpolation are also presented.

We compared our new clone creation technique with generic duplication via copy-and-paste, smart duplication, and a dialog driven technique on a standard desktop system. The results show that the new cloning method is always superior to dialogs and to smart duplication for most conditions. We also compared our clone editing method against rectangular selection. The results show that our method is better in general. In situations where rectangle selection is effective, our method is still competitive. Participants preferred the new techniques overall, too.
# Table of Contents

## Contents

**CHAPTER 1 Introduction** .................................................................................................................. 1
  1.1 Contributions .......................................................................................................................... 3  
  1.2 Thesis Outline ....................................................................................................................... 3  

**CHAPTER 2 Motivation & Related Work** ...................................................................................... 5
  2.1 The Calendar Paradigm ......................................................................................................... 5 
  2.2 Motivation ............................................................................................................................ 8  
  2.3 Modern Applications of Cloning ......................................................................................... 9 
    2.3.1 Cloning in Commercial and Open-source Applications ................................................ 9 
    2.3.2 Cloning in Virtual Immersive Environments ............................................................. 12 
    2.3.3 Cloning and Spreadsheets ............................................................................................. 14 
  2.4 Selection and Alignment ....................................................................................................... 15 
  2.5 Context-aware Copy-and-paste ............................................................................................ 16 
  2.6 Filling and Creating Objects along the Path of Motion ...................................................... 16 
  2.7 Survey of Copying Interfaces .............................................................................................. 19 
    2.7.1 Classification of Copying Interfaces ............................................................................ 19 
    2.7.2 Discussion ..................................................................................................................... 25
CHAPTER 3 A New User Interface for Cloning Objects in Drawing Systems ........... 28

3.1 Clone Creation .............................................................................................................. 29

3.1.1 Freeform Direct Cloning ......................................................................................... 29

3.1.2 Cloning vs. Copying ................................................................................................. 30

3.1.3 Visualizing Clone Dependencies with Arrows ....................................................... 31

3.1.4 Linear Direct Cloning ............................................................................................... 32

3.1.5 Retraction .................................................................................................................. 33

3.1.6 Change of Spacing ................................................................................................. 35

3.1.7 Extending Sets of Clones and Clones of Clones ..................................................... 37

3.1.8 Clones of Clones ...................................................................................................... 37

3.1.9 Clones of Groups ..................................................................................................... 38

3.2 Clone Editing ............................................................................................................... 39

3.2.1 Alternation ............................................................................................................... 40

3.2.2 Interpolation ............................................................................................................ 41

3.2.3 Splitting Sets of Clones, Unlinking, and Deleting Clones ...................................... 42

CHAPTER 4 Implementation Details .................................................................................. 44

4.1 Skencil ......................................................................................................................... 44

4.1.1 Skencil-C .................................................................................................................. 45

4.2 Direct Cloning ............................................................................................................. 46

4.2.1 Dragging Operation and Outlines ........................................................................... 46

4.2.2 Freeform and Linear Direct Cloning ...................................................................... 48
| 4.2.3 | Clone Outlines .................................................................................. 49 |
| 4.2.4 | Varying Spacing Method .................................................................. 50 |
| 4.3  | Clone Editing .................................................................................... 53 |
| 4.3.1 | Interpolation ................................................................................... 56 |
| 4.3.2 | The Undo Mechanism ........................................................................ 57 |

CHAPTER 5 Preliminary User Study ................................................................ 60

| 5.1  | Evaluated Interfaces ........................................................................ 60 |
| 5.2  | Participants ...................................................................................... 62 |
| 5.3  | Apparatus .......................................................................................... 63 |
| 5.4  | Procedure and Experimental Design .................................................. 63 |
| 5.5  | Observations from Pilots of this Preliminary User Study ................... 69 |
| 5.6  | Statistical Models ............................................................................ 70 |
| 5.7  | Results ............................................................................................. 71 |
| 5.7.1 | Task 1 .................................................................................................. 73 |
| 5.7.2 | Task 2, 3, 4 ....................................................................................... 73 |
| 5.7.3 | Tasks 5 and 6 ................................................................................... 74 |
| 5.7.4 | Task 7 ................................................................................................ 74 |
| 5.7.5 | Task 8 ................................................................................................ 74 |
| 5.7.6 | Task 9 ................................................................................................ 74 |
| 5.7.7 | Task 10 ............................................................................................. 75 |
| 5.7.8 | Error Rates ....................................................................................... 75 |
LIST OF TABLES

Table 1 Survey of copying interfaces ................................................................. 22
Table 2 Description of tasks ............................................................................. 65
Table 3 Improving the surveyed systems......................................................... 102
LIST OF FIGURES

Figure 1-1. Editing recurring events in Google Calendar........................................ 1

Figure 2-1. Splitting a series of repeated events into two independent ones in Google
Calendar ........................................................................................................................ 7

Figure 2-2. Shifting clones using Inkscape’s Tile Cloning........................................ 11

Figure 2-3. Clone Options in 3D Studio Max............................................................. 12

Figure 2-4 A group of objects is created with a single mouse-drag operation in MIVE.. 18

Figure 2-5. Decreasing the number of table columns interactively in Microsoft Word.. 19

Figure 2-6. Classification of copying interfaces....................................................... 20

Figure 3-1. Freeform direct cloning......................................................................... 30

Figure 3-2. Linear direct cloning ............................................................................. 33

Figure 3-3. Retraction ............................................................................................. 34

Figure 3-4. Changing spacing by changing the number of clones......................... 36

Figure 3-5. Changing spacing by moving the last outline ...................................... 36

Figure 3-6. Extending a set of clones...................................................................... 37

Figure 3-7. Clones of clones .................................................................................. 38

Figure 3-8. Clones of groups .................................................................................. 39

Figure 3-9. Edit clones dialog box in Skencil-C...................................................... 39

Figure 3-10. Alternation......................................................................................... 40
Figure 3-11. Interpolation ................................................................. 41
Figure 3-12. Splitting a set of clones .................................................. 42
Figure 3-13. Unlinking a clone ........................................................... 43
Figure 3-14. Deleting a clone ............................................................. 43
Figure 4-1. Skencil 0.6.17 running on Mac OS X .................................. 45
Figure 4-2. Pseudo-code for determining the direction in the spacing algorithm ........ 51
Figure 4-3. Pseudo-code for determining the new spacing distance and re-filling the clone position stack with new locations ......................................................... 52
Figure 4-4. Pseudo-code for the Clone Editing Algorithm: EDITALL and EDITFROM .... 56
Figure 4-5. Pseudo-code for obtaining size and color intervals for interpolation ........ 57
Figure 4-6. Pseudo-code for adding the undo information .......................... 58
Figure 4-7. Pseudo-code for Cloning Undo ............................................ 59
Figure 5-1. Tiled cloning user interface in Skencil-C ................................ 62
Figure 5-2. Tasks ............................................................................. 64
Figure 5-3 Average task completion time. Error Bars: ± 1 Standard Error (SE) ....... 72
Figure 5-4 Average number of corrections. Error Bars: ± 1 SE ................... 78
Figure 5-5 Example of a ranking question from the questionnaire featuring a Likert scale ........................................................................................................... 79
Figure 5-6. Feedback charts. Error Bars: ± 1 SE .................................... 80
Figure 5-7. Creating a grid of clones in one operation ............................... 83
Figure 6-1. Tasks in user study 1, (a) task 1, (b) task 2 and 3 ........................ 87
Figure 6-2. Average task completion times for User Study 1. Error Bars: ± 1 SE........... 88
Figure 6-3. Main interaction between task and technique. Error bars: ± 1 SE............... 89
Figure 6-4. Mean number of undo’s. Error bars: ± 1 SE........................................ 90
Figure 6-5. Mean ranks for each technique in User study 1, higher is better. Error bars: ± 1 SE.................................................................................................................. 92
Figure 6-6. Master object designated by a dashed outline........................................... 93
Figure 6-7. Task in second study: (a) task 1, size 1, (b) task 2, size 1, (c) task 3 size 1, (d) task 4, size 1, (e) task 4 size 2, (f) task 5, size 1 ............................................................. 95
Figure 6-8. Average task completion times for User Study 2. Error Bars: ±1 SE........... 96
Figure 6-9. Interaction between task and technique in User Study 2 .............................. 97
Figure 6-10. Mean ranks for each technique in User study 2, higher is better.. Error bars: ± 1 SE........................................................................................................................................ 99
Chapter 1
Introduction

One of the key features of calendar applications is the creation and modification of repeated events. Repeated events are typically created via a dialog, where the user specifies the parameters of the repetition such as title, time, duration, repetition intervals, and the last repetition, if applicable. After such a series of repeated events is created, the user can modify the entire series or individual events within the series. For this, the user has to select an event from the series of repeated events and modify the attributes of the repetition via a dialog. After the modifications are complete, a pop-up will ask the user which of the events to change, see e.g. Figure 1-1, which shows the pop-up from Google Calendar (Google, 2009). The following options are commonly provided: change all events in the series, change only the current instance, or change all following events, i.e. all subsequent events.

![Edit Recurring Event](Image)

*Figure 1-1. Editing recurring events in Google Calendar*

Our fundamental idea is to apply this notion of editing sequences to the drawing domain. The difference between cloning and copying is that copying creates a set of independent objects while cloning retains information about the cloning structure, i.e.
tracks which object was copied from where. Repeated events in a calendar can be considered “live” copies, since changes to the attributes of one of the events are usually applied to other events. In the graphics domain, the equivalent is a set of shapes that are clones of each other, such as a row of rectangles. Considering their order of creation, e.g. from left to right, or from top to bottom, such clones are then similar to a series of events in an ordered sequence. The similarity is even more apparent, if the objects were cloned via copy & paste. Changes to the physical appearance, such as shape or color, are then similar to changing event attributes, such as time. Editing all following events can be thought of as modifying clones created after a given clone.

The general topic of selection and editing of objects on a screen has been well studied and understood. However, several modern and popular drawing systems include cloning techniques, and most, if not all of them, are dialog driven. This is an indication that cloning and clone editing are not that well understood, especially in desktop graphics environments, where dialogs have been almost universally replaced by other user interface techniques. This poses the question if there are good direct manipulation techniques for clone creation and editing. Also, it is unknown how effective such techniques are. Finally, we do not know how effective cloning can be, since no formal evaluation has been done in the past.

In this thesis, we present Skencil-C (Skencil, 2005a), a mouse-based system which implements a new interface to cloning objects in 2D graphics environment, based on the paradigm discussed above. We evaluated the new techniques for creating and
editing clones against existing techniques, such as tiled cloning, smart duplication and rectangular selection editing, in two user studies. To our knowledge, this is the first evaluation of cloning techniques in 2D drawing systems. The description of the new cloning interface (Chapter 3) along with the user studies (Chapter 6) also appear in GI 2010 proceedings (Zaman and Stuerzlinger, 2010).

1.1 Contributions

This thesis presents the following contributions.

- Several new techniques for creating and editing clones in 2D drawing programs inspired by the calendar paradigm.
- A technique for performing linear interpolation among cloned objects in 2D drawing programs.
- Evaluation of the new cloning techniques against existing techniques.
- A survey of existing applications that already support cloning or can potentially benefit from cloning.

1.2 Thesis Outline

This thesis begins with an overview of motivation and related work on cloning in Chapter 2 and describes the calendar paradigm which was used as the inspiration for the new cloning interface developed in this thesis. A classification of copying interfaces is presented along with a table of selected systems that support various forms of cloning.
Following the related work, Chapter 3 describes the new cloning interface followed by the details of implementation in Chapter 4.

Chapter 5 describes the preliminary user study which was designed to explore the performance and the potentials of the new cloning interface and all its features.

Chapter 6 describes the two user studies which explore the selected features of the new clone creation and editing methods.

Chapter 7 concludes the thesis with a summary of the results, an overview of topics for future work, and suggestions for improvements to the systems surveyed in Chapter 2.
Chapter 2
Motivation & Related Work

This chapter describes the Calendar Paradigm. It then presents motivation, related work and applications which use cloning interfaces. Finally, we present a classification schema for copying interfaces. We use this schema to survey selected copying and cloning interfaces in a tabular form at the end of the chapter.

2.1 The Calendar Paradigm

Computer-based calendar applications enable the user to create and modify repeated events. This is typically done via a dialog, where the user specifies the parameters of a repeated event such as title, time, duration, location, repetition intervals, and the last repetition if applicable. After a series of repeated events is created, the user can modify the entire series, or individual events within the series. The user typically selects an event from the existing series of repeated events, opens a dialog, and modifies the attributes of the event. Usually, after the modifications are applied, a dialog box pops up asking the user how to apply these changes. Windows Calendar (Microsoft, 2009f) provides two options: change all events in the series or change only the current instance. Google Calendar (Google, 2009) provides an additional option: change all following events, i.e. all subsequent events too, see Figure 1-1.

Most calendar applications pose constraints to what attributes are modifiable and how they can be modified. For example, Apple iCal (Apple, 2009) doesn’t allow
modifying past events relative to the current date and time. Windows Calendar allows changing the title of a repeated event only through changing the title of the first event in the series. In some applications like in Apple iCal, the time of an event occurrence can be modified by dragging the event to a new time slot. However, in most applications such a time change operation doesn’t provide an option to apply the new time change to all the events in the series. In Windows Calendar, changing time of an event unlinks the event from the series. Only when the first event of a series is revised are these individual time changes reintegrated back into the series. In Google Calendar a time change removes the event completely and irreversibly from the series, so that no further action can get the event back into the series, unless the user performs an undo on the change. Also, in Google Calendar it is possible to split a series of events irreversibly into two. Certain editing scenarios determine if the changes should propagate along the series, or if the series should be split into two independent series. For example, let’s consider a sequence of repeated events titled “an event” in Google Calendar occurring every day of the week starting on Monday, see Figure 2-1a. The user changes the title of the event on Wednesday to “NEW EVENT” and applies this change to all the following events, see Figure 2-1b. The user then shifts the repeated event on Monday back by thirty minutes and applies this change to all the events in the series. In this case, only the events on Monday and Tuesday are affected by the change, as the events beyond Tuesday are no longer part of the series! This demonstrates that the series has indeed been split into two, see Figure 2-1c.
In summary, the following operations are provided by the mentioned calendar applications: repeated entries can be created; modifications can be performed on the entire series of repeated entries, individual entries, and all events after (and including) a particular one (ordered by time of occurrence as in Google Calendar); some operations result in entries becoming unlinked from a series or a series is split; and there may be constraints on which attributes are modifiable. Generalizing from this notion of editing sequences of events in calendars suggests that this paradigm can be applied in other domains, particularly in the drawing domain.

Repeated events in a calendar can be thought of as “live” copies, since by changing the attributes of one of the repeated events the user can apply changes to the attributes of others. In the graphics domain, “live” copies can be considered as clones of shapes. Sets of clones in their order of creation (from left to right, from top to bottom,
etc) can be thought of as series of repeated events in ordered sequences. Changing the physical appearance (such as shape and color) corresponds to changing event attributes such as time and location. Editing all following events, as in the case of Google Calendar corresponds to modifying clones positioned after a given clone (depending on the order of creation, it can be either from left to right, or from top to bottom, etc).

2.2 Motivation

There are four basic principles of graphic design: contrast, repetition, alignment, and proximity (Mullet and Sano, 1995; Williams, 2007). Repetition means reuse of the same or similar elements throughout a design and ensures a sense of unity, consistency and cohesiveness. Alignment ensures that elements in a design are not placed randomly – every element is connected via invisible lines, even far apart elements. Proximity is about moving things closer or farther apart to achieve an organized look, and groups related items. Contrast emphasizes elements. The concept of rhythm is related to repetition and repeats similar elements with a variety of forms or spatial intervals (Meggs, 1992). 3D design also involves lots of repetition and alignment (Chen, Bowman et al., 2004; Chen and Bowman, 2006). The novel cloning tool presented here supports at least two of these principles (repetition and alignment) directly and makes operations related to them more efficient. This allows graphic designers to use cloning as a primary mechanism in their work.
2.3 Modern Applications of Cloning

Today, cloning features are available in popular 2D and 3D applications, including calendars, e.g. Google Calendar, 2D drawing systems, e.g. Inkscape (Inkscape, 2009), and 3D drawing systems, e.g. Autodesk 3D Studio Max (Autodesk, 2009). Cloning is also frequently used in spreadsheets, through formulas (Microsoft, 2009a; b; c).

2.3.1 Cloning in Commercial and Open-source Applications

Clones can be created in Inkscape using the Tiled Cloning interface. First the user selects an object or a whole group and then presses the appropriate shortcut key, which pops up a dialog, see Figure 2-2a. In this dialog the user can adjust a plethora of parameters for the cloning operation such as symmetry, shift, scale, rotation, blur & opacity, color, and trace. In the symmetry tab the user can arrange the clones, either into rows and columns as for tables, or by filling a specified rectangular region. One of seventeen symmetry transformations can be applied during tiled cloning. Some of these transformations include simple translation, certain fixed angle rotations, reflection, glide reflection, combination of reflection and glide reflection, and other various combinations of reflection and rotation. Parameters in other tabs allow the user to interpolate clone attributes by providing a percentage values for each row and column. For example, the shift for each object can be interpolated by adding or subtracting a percentage of the width or the height of the original object. Parameter randomization is also available. Interpolation can be alternated, meaning that every second tile clone is excluded, and it can be cumulated, meaning that the parameter values of all the preceding tiles are added.
to the next tile. Interpolation of the parameters other than shift works in the same manner, except that cumulating is not available for blur & opacity. When the user clicks “Create”, the document gets filled with cloned objects, see Figure 2-2b. Certain configurations can produce tiles which are not spread evenly. By pressing “Unclump”, the user can solve this problem. “Remove” allows the user to delete the clones and to re-adjust the parameters without re-opening the Tiled Cloning dialog. When the clones are created they get linked to the original object, which causes all changes made to the original object to be reflected in the clones. However, changes made to a clone don’t affect other clones or the original. Unlike in calendar applications, the order of clones is not considered in Inkscape. Another limitation is that there is no immediate way of knowing if a particular object is a clone of some other object and what its siblings are, if applicable. There is a non-trivial way to identify the master copy of a clone by pressing a shortcut or selecting the appropriate menu, however, nothing of this sort is provided with regard to siblings. The interface presented in this thesis attempts to overcome these limitations by providing interactive visualization of all clone-to-clone and clone-to-master relationships. This is discussed in detail in the next chapter. Nevertheless, Inkscape provides the most complete vector graphics cloning system known to us today. It provides a powerful interface that allows the user to create cloned objects with the various combinations of interpolated parameters.
Autodesk 3D Studio MAX also provides the ability to clone objects (Hurwicz, 2003). The user clones the selected object using Clone Options dialog. The dialog displays a set of radio buttons, where the user can select between a copy, an instance or a reference, see Figure 2-3. “Copy” creates an independent copy of an object. “Instance” creates a linked clone. Changes in the original are reflected in the instance, and vice versa. “Reference” creates a special clone which can have properties of its own and the user has the control over which properties are linked among clones and which are unique to the reference.

Figure 2-2. Shifting clones using Inkscape’s Tile Cloning
Text styles, which are employed in text editors such as Microsoft Word, can also be considered as a form of cloning – in the sense that the formatting is then cloned in all chosen places in the document. When the user selects the style, he/she expects that the style remains consistent with all the parts of the text or the inserted media which are associated with a particular style.

2.3.2 Cloning in Virtual Immersive Environments

Although creation and editing of objects appears to be a primitive task, scalability of these operations to more than one object becomes important when large projects are tackled.

Chen et al. studied interfaces for cloning objects in virtual immersive environments. In their work, they developed six cloning interfaces (PORT-C, HOMER-C, CbyE, SM, D-Slider, Virtual Keypad) and compared their usability in the domain-specific task of structural engineering in virtual environment. The new
techniques resulted in significant performance gains and a better workflow (Chen and Bowman, 2006).

Cloning tasks were identified to have two major components: selection and clone generation. The clone generation sub-task was analyzed and the design space of cloning was proposed. The design space attributes of cloning according to Chen et al. (Chen, Bowman et al., 2004) are as follows: number of clones, which can be either explicitly defined or randomized; spacing, defining how far two adjacent copies are from each other which can be either equal or unequal; position, where the newly generated object can be part of the original object (in-place) or not in-place, allowing the new object to be put in any other position; direction, defining the area in which the clones are laid out, which may follow the axes of a coordinate system, or may follow a certain shape (e.g., a function defined by several points selected by the user); shape, which can be the same as the original, or can vary either with a fixed scale or a randomized one like in MIVE (Smith, Salzman et al., 2001); and finally, visual attributes, such as color and texture which can change. Parts of this design space analysis are used in this thesis to classify and summarize selected copying interfaces into a survey, and it is presented in Section 2.7.

Besides discussing the design space of cloning, Chen et al. (Chen and Bowman, 2006) developed and compared six cloning interfaces for the domain-specific task of structural engineering in virtual environment. These new techniques resulted in significant performance gains and a better workflow. The techniques designed using
domain characteristics were generally superior to generic techniques that were modified to consider domain characteristics.

2.3.3 Cloning and Spreadsheets

Cloning is employed in spreadsheet applications through formulas. Fujima et al. (Fujima, Lunzer et al., 2004a; 2004b) generalized this notion to enable seamless connections between web applications. Input was clipped from source Web pages into cells on a spreadsheet. Then, connecting these cells through formulas enabled computations to happen. The results could then be transferred to target applications. Cloning of cells enabled the handling of multiple requests simultaneously. However, the user interface in all of the above-mentioned applications is indirect, mostly through dialogs.

The following is an example of how the interface works. The user interface for stock value lookups based on a stock identifier was used from one financial web site, while a currency exchange converter was used from another using a technique called “clipping”. The stock values were then linked as feeds to the currency exchange converter using a technique called “connecting” and, as the result, the user could see a stock’s value in their preferred currency. Motivated by the idea of subjunctive interfaces, a technique called “cloning” was presented, which allows copying of derived instances. As this retains the links and the mathematical relationships (like in a spreadsheet) the user can see and compare multiple stock values in different currencies.
2.4 Selection and Alignment

Single object selection in graphical user interfaces is a very well studied task. Selection of more than one object is related to the work presented here, as it is one alternative for manipulating a set of cloned objects. Multiple object selection is often accomplished via rectangle selection in mouse-based systems or lasso selection in pen-based systems. There are novel group selection techniques, based on the Gestalt principle, which scale to larger object groups and work even well in non-rectangular layouts, see e.g. (Dehmeshki and Stuerzlinger, 2008; 2009). Tapping is another alternative to the Lasso selection. Tapping selection times were shown to be different for different size and spacing of targets, whereas circling selection times differed significantly for different levels of target cohesiveness and shape complexity (Mizobuchi and Yasumura, 2004). The Handle Flags selection technique also displays potential selections that an ink stroke or group could belong to and allow the user to access the desired item (Grossman, Baudisch et al., 2009). However, most of this work targets pen-based systems and is not directly applicable to mouse-based systems. Moreover, we point out that selection alone is not sufficient for clone editing, as a selection operation does not yield information about the cloning structure, nor does it guarantee that the selection aligns with that structure.

Alignment is also related to our work. The alignment stick is a post-hoc alignment technique where objects are aligned by being pushed against the alignment “stick”, based on the ruler metaphor (Raisamo and Räihä, 1996). Snap-dragging (Bier and Stone, 1986) allows the user to draw new alignment objects. Thus, it becomes possible to align
precisely along curves, not just the coordinate axes. Constraint editors also support alignment and have been studied extensively. They are not widely used, because solving constraints is not always an easy task and, more importantly, users do not want to define and maintain constraints for simple drawing tasks (Masui, 2001). Many object drawing applications use snapping to grids for object alignment. HyperSnapping (Masui, 2001) allows interactive changing of the snapping grid while dragging objects.

2.5 **Context-aware Copy-and-paste**

The issue of context-awareness for copy-paste operations is another motivation for our work. Citrine (Stylos, Myers et al., 2004) is an example of a context-aware interface that extends the copy-and-paste paradigm by recognizing the type of data on copy and then performing different operations on paste. The idea of a clipboard extension via a malleable physical interface allows each clip to have its own dedicated key control (Block, Villar et al., 2008). CorelDraw includes a smart duplication technique, where the user can copy and paste an object repeatedly (Foster D. Coburn, 2006). The relative offset of the first copy is applied to all subsequent paste operations. This makes it fast to create a row of objects. This feature is also available in Adobe Illustrator CS4. Microsoft Word and PowerPoint 2007 include this as well, but the feature is curiously missing from the 2008 version.

2.6 **Filling and Creating Objects along the Path of Motion**

The Fill Interpreter (Neely, Booth et al., 1989) is an early system that unifies region
filling, brushing and compositing in a graphical programming language. Filling is a different approach to object replication, which does not preserve the identity of the original object(s).

Our new cloning technique enables the creation of clones from a selected object through a drag operation with a modifier key. Creation of (adjacent) copies of objects through dragging has previously been presented in the context of 3D virtual environments in the MIVE System (Smith, Salzman et al., 2001). Once an object is selected from the inventory, it can be instantiated in the scene. If the user clicks and drags on a surface, multiple adjacent copies are instantiated along the path of motion, see Figure 2-4. The created objects can all have the same appearance, or may differ from each other in a random order if multiple types for that object are assigned. Alignments happen automatically through the use of semantic constraints. One downside of this system is that the structure of the copy operation is not retained.
Microsoft Word (Microsoft, 2009d) allows creating tables by dragging the cursor on a special panel where the number of rows and columns can be changed interactively. If more rows or columns are required, the user drags the cursor away from the upper left corner. If the user drags the cursor in the opposite direction, i.e. towards the upper left corner, the number of rows and columns decreases. An example is illustrated in Figure 2-5 where the user decreases the number of columns from two to one by moving the
mouse cursor leftwards. One disadvantage of this interface is that rows and columns of table cells have to be generated using a special panel, rather than in the document. *Drag-and-release* duplication is a technique related to the one presented here. It can be found in applications such as Adobe Illustrator CS4. However, only a *single* duplicate can be created this way at a given time, creation of multiple duplicates still requires multiple actions.

![Figure 2-5. Decreasing the number of table columns interactively in Microsoft Word](image)

### 2.7 Survey of Copying Interfaces

Copying interfaces are present in many different classes of systems. Almost all systems that allow users to create and/or edit content or input data provide some sort of copying interface. The survey presented in Table 1 lists all major classes with selected systems representing each class. Replication features for each system are presented along with a description and characterization of each copying interface.

#### 2.7.1 Classification of Copying Interfaces

In this section, we present a classification schema and compare copying interfaces according to it. This was inspired by the design space for cloning proposed by Chen and Bowman (Chen and Bowman, 2006). The goals of this classification schema are as
follows:

1) to spot potential weaknesses and strengths of the existing copying interfaces;

2) to establish similarities of copying interfaces among different applications and classes;

3) to spot differences among the copying interfaces of different applications and classes, and to investigate how these differences can be used to come up with more effective solutions for a particular class of system by borrowing the copying concepts from other classes.

<table>
<thead>
<tr>
<th>Replication method</th>
<th>Attributes preserved</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dragging</td>
<td>All</td>
</tr>
<tr>
<td>Interaction through dialog</td>
<td>Some</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Number of replications</th>
<th>Editing methods and limitations</th>
</tr>
</thead>
<tbody>
<tr>
<td>One</td>
<td>Direct manipulation</td>
</tr>
<tr>
<td>Proportional to dragged distance</td>
<td>Dialog</td>
</tr>
<tr>
<td>Defined as created</td>
<td>Link to related copies</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Visual appearance</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Same as the original</td>
<td>Full support</td>
</tr>
<tr>
<td>Randomized</td>
<td>Limited support</td>
</tr>
</tbody>
</table>

| Attribute interpolation of copies      |                                  |

*Figure 2-6. Classification of copying interfaces*

We use the following seven dimensions to analyze and classify copying interfaces: *replication method, number of replications, visual appearance, attributes preserved, editing methods and limitations, link to related copies, attribute interpolation*
of copies. The first four are concerned with the creation of copies. The remaining three are concerned with their editing. The feature and the method to achieve it are listed in the first two columns. Typical methods include dragging or interaction through a dialog. Dragging may also require modifier key(s) pressing. Number of replications indicates the amount of copies the corresponding method can produce. Copies can be produced one at a time, their number can be proportional to the dragged distance, or it can be predefined. Visual appearance describes the deviation of visual attributes between the copy and the original. Appearance can be either same as the original, or it can be random as in the case of MIVE (Smith, Salzman et al., 2001), or it can vary due to interpolation like in Inkscape (Inkscape, 2009). Attributes preserved describes how many attributes of the original object are preserved in the copy(ies). All attributes are usually preserved, but sometimes certain attributes may not be preserved, like for example cell comments are not replicated using data filling in spreadsheets in Excel (Microsoft, 2009a).

The last three columns describe the respective copy editing mechanisms of the corresponding replication methods. Similarly to the replication method column, editing method column describes how copies can be edited once they are created. Typically, it is done either through direct manipulation or a dialog. Link to related copies indicates whether the system is aware that the copy has a master object and/or related to other copies. Typically, systems can either have support for editing multiple objects with one operation, or they can have limited support. If there is no link to related copies, it automatically implies that editing multiple copies simultaneously is not possible. Finally,
Attribute interpolation of copies describes whether the editing feature allows for the attributes of copies (e.g. color, shape, size, orientation, etc) to be interpolated. Although very promising, this feature appears rarely in today’s systems. It emphasizes the power of editing multiple copies simultaneously, and hence can be only available in systems that support multiple copy editing and links to related copies. Among today’s existing systems Inkscape (Inkscape, 2009) is arguably the most powerful system in this respect.

Table 1 Survey of copying interfaces

<table>
<thead>
<tr>
<th>Class/Application</th>
<th>Feature</th>
<th>Replication method</th>
<th>No. of replications</th>
<th>Visual appearance</th>
<th>Attributes preserved (during copying)</th>
<th>Editing method and limitations</th>
<th>Link to related copies¹</th>
<th>Attribute interpolation of copies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Diagram Editors</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Microsoft Word/Power Point 2007</td>
<td>Smart duplication</td>
<td>Ctrl-D</td>
<td>One</td>
<td>Same as the original</td>
<td>All</td>
<td>DM² or Dialog</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Microsoft Visio 2007 (Microsoft, 2009e)</td>
<td>Shape duplication</td>
<td>Dragging or dialog</td>
<td>One</td>
<td>Same as the original</td>
<td>All</td>
<td>DM or Dialog</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Spreadsheets</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Microsoft Excel 2007</td>
<td>Cell filling</td>
<td>Dragging</td>
<td>Defined as created</td>
<td>Data preserved</td>
<td>All except cell comments</td>
<td>DM or Dialog</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>C3W (Fujim a, Lunzer et al., 2004b)</td>
<td>Cloning Scenarios</td>
<td>Pop-up menu on a cell to be cloned</td>
<td>One</td>
<td>New instances of cells hold same values as the original</td>
<td>All</td>
<td>DM or Dialog</td>
<td>Yes³</td>
<td>No</td>
</tr>
</tbody>
</table>

¹ “No” implies editing multiples is not available

² “DM” stands for direct manipulation

³ Yes, but not in a typical sense: editing an entry doesn’t change the attributes of related entries, but the initial relations are preserved.
<table>
<thead>
<tr>
<th>Class/Application</th>
<th>Feature</th>
<th>Replication method</th>
<th>No. of replications</th>
<th>Visual appearance</th>
<th>Attributes preserved (during copying)</th>
<th>Editing method and limitations</th>
<th>Link to related copies</th>
<th>Attribute interpolation of copies</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Microsoft Word 2007</strong></td>
<td>Inserting a Table</td>
<td>Initial creation through panel dragging or dialog. Adding clones through right click dialog.</td>
<td>Explicitly defined through dialog or defined as created through the panel technique</td>
<td>N/A</td>
<td>N/A</td>
<td>DM or Dialog. Multiple cols, rows or cells can be edited or deleted</td>
<td>Editing a row or a column affects other rows or cols</td>
<td>No</td>
</tr>
<tr>
<td><strong>Microsoft Word 2007</strong></td>
<td>Word Art 2007</td>
<td>Dialog</td>
<td>Defined as typed</td>
<td>Randomized shape, color and texture can be changed</td>
<td>Color, texture and font preserved for each character</td>
<td>Dialog</td>
<td>Yes, editing characters keeps the same style</td>
<td>No</td>
</tr>
<tr>
<td><strong>Calendars</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mozilla Sunbird (Mozilla, 2010)</td>
<td>Creation of repeated events</td>
<td>Dialog</td>
<td>Explicitly defined</td>
<td>Same as the original</td>
<td>All</td>
<td>Dialog only for multiple entries. Dragging only for single entry. No following events editing option</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Google Calendar</td>
<td>Creation of repeated events</td>
<td>Dialog</td>
<td>Explicitly defined</td>
<td>Same as the original</td>
<td>All</td>
<td>Dialog. Following events editing option. Dragging and certain editing scenarios cause splits</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Microsoft Windows Calendar</td>
<td>Creation of repeated events</td>
<td>Dialog</td>
<td>Explicitly defined</td>
<td>Same as the original</td>
<td>All</td>
<td>Dialog*. No following events editing option</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

---

* Dragging removes occurrence from the series, but if the first occurrence is changed, dragged occurrences are put back into the series; time of event can be changed by changing the first event in the series
<table>
<thead>
<tr>
<th>Class/ Application</th>
<th>Feature</th>
<th>Replication method</th>
<th>No. of replications</th>
<th>Visual appearance</th>
<th>Attributes preserved (during copying)</th>
<th>Editing method and limitations</th>
<th>Link to related copies</th>
<th>Attribute interpolation of copies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apple iCal</td>
<td>Creation of repeated events</td>
<td>Dialog</td>
<td>Explicitly defined</td>
<td>Same as the original</td>
<td>All</td>
<td>Dragging or Dialog. Past events can't be modified</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td><strong>2D Graphics Editors</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Xfig</td>
<td>Copy</td>
<td>Dragging a corner of a shape</td>
<td>Explicitly defined</td>
<td>Same as the original</td>
<td>All</td>
<td>DM</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Skencil</td>
<td>Duplicatio n</td>
<td>Panel button or shortcut</td>
<td>One</td>
<td>Same as the original</td>
<td>All</td>
<td>DM or Dialog</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>InkScape</td>
<td>Simple cloning</td>
<td>Menu or shortcut</td>
<td>One</td>
<td>Same as the original</td>
<td>All</td>
<td>DM or Dialog</td>
<td>Yes, only if editing the original</td>
<td>No</td>
</tr>
<tr>
<td>InkScape</td>
<td>Tile cloning</td>
<td>Dialog</td>
<td>Explicitly defined</td>
<td>Varies if interpolation took place</td>
<td>All</td>
<td>DM or Dialog</td>
<td>Yes, only if editing the original</td>
<td>Yes</td>
</tr>
<tr>
<td>InkScape</td>
<td>Pattern along path</td>
<td>Dialog</td>
<td>Proportional to distance and size</td>
<td>Same as original</td>
<td>All</td>
<td>Dialog</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td><strong>Adobe Photoshop CS4 (Adobe, 2008)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Copying and moving a selection</td>
<td>Dragging a selection using the move tool while holding a modifier key (Alt)</td>
<td>One</td>
<td>Same as the original</td>
<td>All</td>
<td>DM or Dialog</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Texture tiling</td>
<td>Dialog</td>
<td>Proportional to size</td>
<td>Same as the original</td>
<td>All</td>
<td>Dialog</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Clone Stamp (replication of an image area with brush)</td>
<td>Dialog box</td>
<td>Explicitly defined</td>
<td>Same as the original</td>
<td>All</td>
<td>DM or Dialog</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td><strong>3D Graphics Editors</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AutoCad</td>
<td>Array (2D formation of copies in rectangular or circular order)</td>
<td>Dialog box</td>
<td>Explicitly defined</td>
<td>Same as the original</td>
<td>All</td>
<td>DM or Dialog</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Sesame (Oh, Stuerzlinger)</td>
<td>Copying 2D shapes</td>
<td>Dragging</td>
<td>One</td>
<td>Same as the original</td>
<td>All</td>
<td>DM or Dialog</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Class/Application</td>
<td>Feature</td>
<td>Replication method</td>
<td>No. of replications</td>
<td>Visual appearance</td>
<td>Attributes preserved (during copying)</td>
<td>Editing method and limitations</td>
<td>Link to related copies¹</td>
<td>Attribute interpolation of copies</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------</td>
<td>--------------------</td>
<td>---------------------</td>
<td>-------------------</td>
<td>---------------------------------------</td>
<td>--------------------------------</td>
<td>-----------------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>et al., 2006)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3D StudioMax</td>
<td>Cloning</td>
<td>Dialog</td>
<td>One</td>
<td>Same as the original</td>
<td>All</td>
<td>DM or Dialog</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>MIVE (Smith, Salzman et al., 2001)</td>
<td>Space filling</td>
<td>Dragging</td>
<td>Proportional to distance</td>
<td>Randomized</td>
<td>N/A</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>PORT-C, HOMER-C, CbyE, SM, D-Slider, Virtual Keypad (Chen and Bowman, 2006)</td>
<td>Cloning</td>
<td>DM: Joystick, selection box and state transition menus³</td>
<td>Explicitly defined</td>
<td>Same as the original</td>
<td>All</td>
<td>No. All editing is performed during creation</td>
<td>Ghost copies are displayed until apply button is pressed</td>
<td>No</td>
</tr>
</tbody>
</table>

2.7.2 Discussion

From the data in the table, many observations can be made. For diagram editors, besides standard copy-and-paste techniques, shape duplication can be achieved by direct manipulation. Microsoft PowerPoint can be classified both as a diagram editor and 2D graphics editor, and shares a lot of features with the vector-based 2D graphics editors. For all presented diagram and 2D graphics editors it is typical that all visual attributes are preserved while copying. Also, typically only one copy is created. However, there are systems with more powerful user interfaces that create explicitly defined numbers of

---

³ Wand control with four actions/buttons (PORT-C); copy-and-paste style replication with the help of snap-to-grid (HOMER-C); automatic calculation of the position of the next objects with only first object given as input (CbyE - Copy by example); user-defined box constrains the volume that the cloned objects occupy (SM – Space Metaphor); moving the tablet to toggle direction of cloning (D-Slider-Dynamic Slider Widget); numeric virtual keypad to enter the amount of copies (Virtual Keypad)
replications, or replicate proportional to the dragged distance. Systems that permit creating multiple simultaneous copies also support editing multiple copies and interpolation among them. Similar to 2D graphics systems, the user interfaces to all 3D graphics systems preserve the visual attributes and allow creation of either one copy at a time, an explicitly defined number of replications, or replicate proportional to dragged distance. In spreadsheets such as Microsoft Excel, new objects are filled with the copy of the object in the dragged cell. If the object is alphanumeric or symbolic, it is copied to the next cell as the user drags the cursor over it. If it’s a formula, then it is applied to the values in the next row, and the corresponding result is inserted into the cell. The number of cells created is proportional to the dragged distance as in MIVE (Smith, Salzman et al., 2001) and as in table creation by dragging on a panel in Microsoft Word (Microsoft, 2009d). Repeated event creation in calendar provides links to other related repeated events and allows editing multiple events similarly to the copying interface of Inkscape (Inkscape, 2009). However, calendar systems typically allow also editing the events following (and including) the selected event.

The cloning interface presented in this thesis was built by combining the most powerful techniques across the different classes presented in Table 1.

The clone creation part of the cloning interface borrows the ideas from the space filling technique of MIVE (Smith, Salzman et al., 2001), the table creation by dragging on a panel (Microsoft, 2009d), and the data filling in spreadsheets technique of Microsoft Excel (Microsoft, 2009a). Clones that share a common master copy are linked together

26
into a set as in Inkscape (Inkscape, 2009). The objects can be edited following selected objects from the set of related clones like in calendar systems where all following appointment events can be edited relative to the selected event. A more detailed look at the new cloning user interface is presented in Chapter 3. The implementation details are presented Chapter 4. In Chapter 7 we included another survey which provides suggestions on how the systems surveyed in the table above may benefit from the findings of our research.
Chapter 3
A New User Interface for Cloning Objects in Drawing Systems

The cloning interface implemented for this thesis allows the creation of multiple clones in drawing systems from a selected object. As these clones are arranged into sets, it is also possible to edit multiple clones with a variety of clone editing techniques. The clone creation technique was inspired by the space filling technique introduced in MIVE (Smith, Salzman et al., 2001) and the table creation by dragging on a panel technique of Microsoft Word (Microsoft, 2009d). The new copying interface emphasizes direct manipulation as opposed to the menu driven cloning interfaces used in other applications, such as Inkscape (Inkscape, 2009). Hence, it was named direct cloning. The idea of repeated event editing in calendar systems, such as Google Calendar (Google, 2009), was adapted to the drawing context and is called clone editing in the new cloning interface. The position of a clone relative to other clones in a drawing system is used as a metaphor for the time of event occurrence in calendar systems. Moreover, the clone editing interface goes beyond the boundaries of the calendar paradigm by providing editing techniques, which are specific to the graphics domain such as skip editing and interpolation. Direct cloning, along with the various clone-editing techniques, presented in this thesis offer a new interface for cloning in the drawing domain.
3.1 Clone Creation

In our system clones can be created either along a free form path or along a linear path. The corresponding user interface techniques are named freeform direct cloning and linear direct cloning. All operations discussed in the following are integrated with undo and other common user interface mechanisms, as applicable.

The idea of the clone creation technique presented here revolves around the same principle as in MIVE. However, the goal is different. Unlike in MIVE where objects are selected for placement from a menu, the direct cloning technique works on the currently selected object or group. Direct cloning provides more flexibility than the filling technique of MIVE, because it offers options for varying the spacing between the objects as well as changing the path of creation, which is discussed later in more detail. Also, unlike in the table creation interface of Microsoft Word, the user creates objects by moving the cursor directly in the document and not on a panel.

3.1.1 Freeform Direct Cloning

To create clones along a free path, a single object has to first be selected via normal selection techniques. Dragging the object while holding down a modifier key, in our system <WINDOWS>, then activates cloning. A static outline representing the proposed location of a clone appears each time the distance between the current and the previous outline (or the selected object) exceeds a threshold. We currently use 20% of the diameter of the bounding box of the manipulated object, see Figure 3-1a. The last small outline with the four sided arrow cursor moves with the mouse cursor. After the user releases
both the left mouse button and the <WINDOWS> key, the direct cloning action is complete and the proposed clones are replaced with clones of the selected object and are inserted in the document see Figure 3-1b. In effect, the selected object gets replicated a number of times proportional to the length of the cursor drag. We called this process freeform direct cloning. To aid the user, the current count of clones is displayed in the status bar area during dragging. The arrows that appear upon completion illustrate the order in which the clones were created. They point to the object that was created just before. These arrows are only shown when any object in a set of clones is selected to visualize the entire sequence.

![Figure 3-1. Freeform direct cloning](image)

**3.1.2 Cloning vs. Copying**

Cloning is different from copying in the sense that dependencies exist between the cloned objects. After the objects are created these dependencies assist the user with editing multiple clones at a time. The cloning interface presented here adheres to this principle. To achieve it, several techniques were introduced for editing multiple clones and a
technique to visualize the clone relationships and dependencies to the user. This is in contrast to most other cloning interfaces, where the clone dependencies are typically not visualized, consider e.g. calendars. In our system, the first clone is pointing to the original object, the second to the first, and so on. Direct cloning, clone editing, and the visualization of clone dependencies makes the new user interface a cloning interface rather than a copying one.

3.1.3 Visualizing Clone Dependencies with Arrows

The arrows that appear upon completion illustrate the order in which the clones were created. These arrows always point to the object that was created just before. These arrows are only shown when any object in a set of clones is selected to visualize the entire “chain”.

Once the clones are created, they are visualized with connecting arrows between each pair, see Figure 3-1b. The objects can be both clones, or one can be the original (the root object) and the other one can be a clone. An arrow points to the object, which was created before it. Thus, a direct cloning action results in a chain of cloned objects. Each of them points to the previous one in the direction opposite to the direction of creation. These arrows bind the clones logically in this order all the way to the root object (the original). The arrows are visible only when an object belonging to the set of clones is selected, see Figure 3-1b. If the user deselects an object from the set of clones by clicking on an empty area or any other, unrelated object, the arrows disappear. If the user selects an object, which belongs to some other set of clones, the arrows for the new set of clones
become visible, linking the objects for the newly selected set of clones. The arrows for the previously focused set of clones become hidden.

A design decision was to make the arrows represent a precedence relationship between the clones, rather than a parent-child relationship between the original and the clones. Doing otherwise would conflict with the calendar paradigm that is used as a reference for the implementation of the clone-editing interface. Repeated events in the calendar can be thought of as clones, since by changing the properties of one of the repeated events the user can apply changes to the properties of others. In Google Calendar, when the user changes the properties of a repeated event, a dialog box pops up asking the user how to apply these changes. Applying this paradigm to the graphics domain suggests that the position of the clone relative to other clones in the set of clones becomes a metaphor for the time of the occurrence of the event, and the arrows demonstrate this dependency. In this context, linking the clones to each other using arrows in the order of their creation is more suitable, rather than linking them all to the root into a parent-child relationship. Another argument against visualization of a parent-child representation is that having all arrows point to the same object leads to a lot of visual clutter and also doesn’t capture the sequence in which the clones were created.

3.1.4 Linear Direct Cloning

Because horizontal and vertical arrangements are very common, the system also supports creation of clones along a straight path in either dimension. This technique is called
linear direct cloning. The same idea is used in Xfig (Smith, 2009; Xfig, 2009) for constraint copying.

Holding the <ALT> key in addition to the <WINDOWS> key and dragging a selected object activates this mode. Then, the system assists the user by snapping a straight path to one of the four main directions. The direction is chosen based on the predominant direction of the moving mouse cursor. Figure 3-2 illustrates an example when the cursor is moved to the right, and the outlines and the resulting clones are aligned horizontally.

![Figure 3-2. Linear direct cloning](image)

3.1.5 Retraction

Retraction was inspired by and the table creation technique of Microsoft Word (Microsoft, 2009d). It can be activated when the system is in the linear direct cloning state and the mouse cursor is moved towards the original object, i.e. in the direction opposite to the initial movement direction. Then, clone outlines are deleted one by one until the selected object is reached. Figure 3-3 illustrates this.
The user changes the direction of cloning interactively by changing the direction of the mouse cursor. When the change occurs, clone outline retraction happens. Figure 3-3a illustrates the state before the clone outline retraction is activated. Figure 3-3b illustrates the state when the first clone outline is deleted as the user intersects it with the mouse cursor while moving the cursor in the direction opposite to the initial direction of cloning. If the user is moving the cursor beyond the selected object in the direction directly opposite to the initially chosen one, new outlines start to come forth on the opposite side of the original object, see Figure 3-3e.

Figure 3-3. Retraction

In the case of freeform direct cloning it does not make too much sense to enable clone outline retraction, since the path of creation is unpredictable (it can be a curve or a spiral which can change direction continuously). In this situation, controlling the number of clones while dragging may become tedious for a regular user, since the path of
creation needs to be memorized. On the other hand, if the path is a straight line (horizontal or vertical), it is more straightforward to keep track where to backtrack in order to delete the excessive clone outlines. This is why this feature is more appropriate in the case of direct cloning along a linear path.

### 3.1.6 Change of Spacing

Spacing can be changed in two ways: by changing the number of proposed clones or by shifting each outline by an arbitrary distance. Both methods can be used in any order and multiple times within a single direct cloning operation. The first method allows the user to distribute clones differently by changing the distances between them, while keeping the mouse cursor in the same position. This is accomplished by scrolling the mouse wheel, which increases or decreases the number of proposed clones incrementally. This spacing technique is illustrated using the example in Figure 3-4. Initially, two outlines are created, see Figure 3-4a. Scrolling the mouse wheel down decreases the number of proposed clones, see Figure 3-4b. Note that the cursor remains at the same position. After the mouse button is released, two clones are created, see Figure 3-4c.
In the second method, moving the mouse cursor changes the spacing between clones. In this case, the number of clones does not change, but the inter-object distances vary. This mode is activated by releasing the <ALT> key while in linear direct cloning. Then, moving the cursor towards or away from the selected object, changes the spacing between the proposed clones, see Figure 3-5a,b. The minimum permitted distance between the proposed clones is zero, i.e. objects that touch each other. Releasing the mouse button generates the clones, see Figure 3-5c.

Figure 3-5. Changing spacing by moving the last outline
3.1.7 Extending Sets of Clones and Clones of Clones

An existing set of clones can be extended by performing direct cloning on the last object in the set, see Figure 3-6. First, the last object in a sequence of clones is selected. Then, a direct cloning operation on this object will extend the set. However, new clone operations can be performed not only on the last object in a set, but on any other object as well, including the master object, i.e. the original, first object. This is referred to as creating clones of clones. Each object in a set of clones of clones can have infinitely many subsets of clones of its own.

![Figure 3-6. Extending a set of clones](image)

3.1.8 Clones of Clones

So far, only the creation of clones from the master object has been described. The direct cloning mechanism also allows for the creation of clones from a clone. To do so, the user first selects an intermediate clone from an existing set of clones and then performs direct cloning using this clone as the original for the direct cloning action. This creates clones of clones.

An example is illustrated in Figure 3-7. The middle object in a set of three clones is initially selected. Direct cloning is then performed on that clone, which results in a new
subset of clones, see Figure 3-7b. In another step and as shown in Figure 3-7d, direct cloning is then performed on the middle object in the new, extended group. This yields another subset of clones, see Figure 3-7e. Figure 3-7f will be discussed in Section 3.2.

![Figure 3.7. Clones of clones](image)

### 3.1.9 Clones of Groups

Direct cloning can be performed not only on individual objects, but on groups as well, as shown in Figure 3-8. First, a group of objects must be selected. Direct cloning will then generate a clone pointing to each object in the original group. An example is illustrated in Figure 3-8. First a set of clones is selected (Figure 3-8a). Figure 3-8b shows an intermediate step in direct cloning on this group. Releasing the button results in the group being cloned. Each object in the cloned group points to the corresponding object in the original group, see Figure 3-8c.
3.2 Clone Editing

In electronic calendars, the attributes of repeated events, such as time or location, can be changed for any event. The system then queries the user how to apply these changes via a pop-up dialog. Usually these changes can be applied to the selected event only, to all the events in the series, or to all the events following the chosen event. We implemented the same editing mechanism in our drawing system. When an attribute of a clone is changed, a dialog box pops up to ask how to apply this change, see Figure 3-9.

![Figure 3-9. Edit clones dialog box in Skencil-C](image)

In most 2D graphics software, objects have several attributes that can be modified. The most common ones are: position, orientation, size, color, and shape. Position is usually handled differently from the other attributes in that it can be modified only via direct manipulation. In our clone-editing interface, all the attributes except position can be edited, and the changes can be replicated across clones in a set. This
design decision was made to avoid user frustration through a pop-up every time a clone is moved. Editing works also for clones of clones and clones of groups. An example of editing clones of clones is illustrated in Figure 3-7ef. First, an object in a set of clones of clones is selected. Then, its shape is changed from a square into a circle. A pop-up box is displayed right after completion of the edit. Assuming the user clicks on “All following”, all objects created after or cloned from the selected object are changed, as illustrated in Figure 3-7f.

3.2.1 Alternation

Editing can be alternated by selecting “All following skipping one” in the pop-up box, see Figure 3-9. An example is shown in Figure 3-10. Assume we have a set of four clones, all, as shown in Figure 3-10a. The first object is selected and its color is changed to black. The user chooses “All following skipping one” in the pop-up box, and the changes are applied only to every second object. Alternation of editing works for clones of clones and clones of groups as well, like every other editing feature.

![Figure 3-10. Alternation](image)

40
3.2.2 Interpolation

Our clone-editing interface supports interpolation of attributes, currently only size and color are supported. This method interpolates one or more attributes between the master and the selected object in a linear manner. Figure 3-11 illustrates interpolation with both attributes. We start with a set of clones, see Figure 3-11a. Then the color and size of the last object in the set are changed, into black and a smaller size, respectively, see Figure 3-11b. Right clicking on the last object and selecting “Interpolate to master” then activates interpolation. Figure 3-11d shows the result, which assigns appropriately interpolated values to intermediate objects.

\[\text{Interpolate to Master}\]
\[\text{Unlink Remaining}\]
\[\text{Unlink Object}\]
3.2.3 Splitting Sets of Clones, Unlinking, and Deleting Clones

Clone relationships can be broken at arbitrary objects. This results in two separate sets of clones. An example with a set of four clones is shown in Figure 3-12. First, the user right clicks on the object that will be the “new” master and selects “Unlink remaining”, see Figure 3-12a. This splits the set into two independent clone sets. Figure 3-12b and c illustrate the resulting sets of clones.

![Diagram of splitting clones](image)

Figure 3-12. Splitting a set of clones

Clones can be unlinked from their sets. An example is shown in Figure 3-12. First, the object to be unlinked is selected and “Unlink Object” is selected in the right-click menu. The system will then pop-up a dialog asking how to apply this change, as in Figure 3-9. There are three choices. If the user chooses to changes all objects, then all clones will be unlinked. If “All following” is selected, then all the objects including the
selected one will be unlinked from the set of clones. Choosing “selected” will only unlink the selected clone, see Figure 3-12b.

Figure 3-13. Unlinking a clone

Clones can be deleted from their sets by selecting a clone and pressing the <DELETE> key. Figure 3-14 shows an example. When the <DELETE> key is pressed on a selected clone, a pop-up appears asking how to apply changes, similar to unlinking. It is then possible to delete all the objects, only the ones following the selected, or only the selected one as in Figure 3-14b.

Figure 3-14. Deleting a clone
Chapter 4
Implementation Details

In this chapter, we describe the implementation details of the new cloning interface. The novel direct cloning and clone editing techniques were implemented by extending Skencil (Skencil, 2005a).

4.1 Skencil

We chose Skencil as the implementation platform for the cloning interface introduced in this thesis. Skencil is a free vector graphics editor, released under the GNU LGPL license, see Figure 4-1. It allows users to create standard vector drawings such as illustrations, clip art, and various geometric patterns. Skencil is similar in functionality to graphics editors such as CorelDRAW, Adobe Illustrator, and Inkscape. We chose it because it offers great versatility in editing objects, unlike most other freely available vector graphics editors. Objects can be easily resized, rotated, scaled, morphed, and even colored by direct manipulation. The availability of a large set of editing operations was an important factor in the choice of the implementation platform. The larger the variety of independent editing operations that can be performed, the stronger the need is for a clone-editing interface. Also, since there are more operations to compare, it is easier to reveal which ones are more useful for clone editing. Skencil is implemented in Python, with some parts written in C for speed. Given that scripting languages are often easier to deal with, choosing Skencil over other open-source graphics editors such as Inkscape
would cut down on developing time.

Figure 4-1. Skencil 0.6.17 running on Mac OS X

4.1.1 Skencil-C

We called our system Skencil-C. The “C” stands for “cloning”. The chosen naming style follows the analogy introduced with PORT-C (Chen and Bowman, 2006). Version 0.6.17 of Skencil was extended to include clone creation via direct cloning and the clone editing, as described in Chapter 3. A clone can be created in Skencil-C from any shape or group of shapes, such as circles, rectangles, Bezier polygons, clipart, etc.
4.2 Direct Cloning

In Skencil-C, direct cloning is implemented as an extension to the move operation. The outlines of clones are displayed during the operation in every location where a clone is to be created. Thus, the outlines function as visual guides, helping the user to arrange the clones before they are created.

4.2.1 Dragging Operation and Outlines

This section explains in detail the role of the dragging operation and the outlines in direct cloning.

The ability to drag objects freely is a feature that is common to most, if not all, vector graphics editors and other popular 2D graphic applications such as Adobe Photoshop, GIMP, Microsoft PowerPoint, and Visio. From the software architecture point of view, object dragging is essentially a fundamental action in the user interface. Other examples of such actions are resizing, changing color, grouping, or shape editing. In essence, object dragging and other actions are atomic operations on the object in an open document, and hence can be compared to transactions in a database.

There are at least two methods for a dragging transaction that are commonly found in user interfaces. The first method is when the selected object(s) move(s) along with the cursor interactively. This is often referred to as dragging the object itself. In the second method only the selection outline is dragged while the original object remains stationary until the mouse button is released. Upon that release, the object is then moved to the new position. The dragged outline serves as a guide suggesting the location where
the object is to be placed once the mouse button is released, since the object itself is not moved until the dragging transaction is complete.

The following is an example of a dragging transaction scenario that utilizes the dragged outline method. The user selects an object with a mouse. The selected object is then highlighted with an outline to visually confirm to the user that the selection operation succeeded. If the user presses the left mouse button down on the selection and drags the cursor, the outline of the object (represented by a dashed bounding rectangle) is moved, while the object itself stays at the initial location until the left mouse button is released, see Figure 3-1a. When the button is released, the object is placed in the new location and the outline disappears.

Skencil uses the outline method for the dragging transaction. We consider direct cloning as a special case of the dragging transaction due to the similarities between the two. In both cases, the objects are displaced, and the outline(s) indicates the destination(s).

During the dragging transaction, the dragged object stays in its original location and, instead, the outline is moved along with the mouse cursor. The dragged object is displaced only when the mouse button is released. The presence of the original object is a requirement for the direct cloning transaction to start. By implementing direct cloning as a special case of the dragging transaction, the dragged object can be used as the original object of the direct cloning transaction, and the moving outline can be used to display the position(s) of the clone(s) during the cloning transaction. As the selection outline is
dragged, new outlines are added as appropriate to designate the location of new clones, see Figure 3-1b.

In Skencil-C, *direct cloning* was implemented as an extension of the MOUSEMOVE module. Direct cloning is activated when an object is dragged and either the `<ALT>` key (for linear direct cloning) or the `<WINDOWS>` key (for freeform direct cloning) is pressed. `<COMMAND>` key is used in the Mac OS X port instead of `<WINDOWS>` key.

The positions of the outlines are tracked in Skencil-C. They are pushed onto the stack when a new outline needs to appear (e.g. if the cursor has moved far enough away from the previous outline), or popped when the last outline needs to be removed. The stack stores the outlines before they are replaced by the clones, which are then inserted into the document when the direct cloning transaction is complete. Until then, this stack is sent to the draw module, which displays the dashed rectangle outlines for all the positions stored in the stack.

### 4.2.2 Freeform and Linear Direct Cloning

The method for freeform cloning is the simpler of the two. When the user starts dragging, the position of the outline of the original object is inserted as the first element in the stack of the clone positions. As the user drags the cursor further, the distance between the dragged selection outline and the position of the previous object in the stack of clone positions is compared against a threshold. This threshold is set to be 20% larger than the diameter of the bounding of box of the original object. This is what we refer to
as *preset spacing*. As soon as the distance exceeds the threshold, a new location is pushed onto the stack of the clone positions. As the selection is dragged and the locations are inserted into the stack, these locations are used to draw the dashed outline guides in the document. They serve as a visual aid indicating where the new clones will be inserted once the *direct cloning* transaction is complete when the user releases the mouse button.

The method for the linear *direct cloning* is an extension to the freeform method. The alignment can be either vertical or horizontal. If the user starts horizontal dragging first, then horizontal snapping is activated, otherwise vertical is activated.

### 4.2.3 Clone Outlines

The process of creating a stack of outline locations is the same in both freeform and linear direct cloning. However, the method for setting the locations of the clones varies. To achieve alignment, the outlines in the linear direct cloning mode have either their $x$ or $y$ coordinate fixed, depending on whether the clone outlines are aligned vertically or horizontally. Another difference is that the number of outlines starts to decrease in the aligned mode if the user starts to move the mouse in the opposite direction. In the aligned mode this is possible, because the distance between the original and the currently dragged object can be checked during the dragging transaction to determine how many clone outlines fit between the original object and the mouse position. If the stack holds more locations than appropriate, the extra outlines are popped. Hence, any excessive clone outlines are removed when the user starts to move in the opposite direction.
When the user releases the mouse button, the clone outlines are replaced with the clones of the original object. The undo information for this transaction is pushed onto the undo stack, so that the direct cloning transaction can be reversed if necessary. The undo mechanism is described in detail later.

4.2.4 Varying Spacing Method

During direct cloning it is possible to vary the space between clones using the two methods, see Section 3.1.6. The first method is changing the number of clone outlines that fit between the original object and the mouse pointer position. Increasing the number of clone outlines decreases the spacing between them, while decreasing the number of clones increases the spacing.

The algorithm for changing the number of clone outlines method is presented as follows. At first, the direction of clone creation has to be determined. There are only four of possibilities for the direction: up, down, left and right. The direction can be easily determined from the location of the first two clones (0 and 1) in the clone position stack, see Figure 4-2.
The next step is to determine the new spacing distance and to re-fill the clone position stack with the new locations. Depending on whether the direction is horizontal or vertical, either the width or the height of the outline is used in the calculation, see Figure 4-3a,b.

\[
d = \text{distance between the original and the mouse pointer} \\
w = \text{width of a clone outline} \\
h = \text{height of a clone outline} \\
n = \text{new number of clone outline (assigned by scrolling the mouse wheel)} \\
s = \text{new spacing between the clones} \\
\text{loc_original} = \text{location of the original}
\]
loc_original = clonePosStack[0]
delete clonePosStack //it becomes empty
clonenPosStack.append(loc_original) //set the location of the original as first element in the list
//location is horizontal
if direction == Left or direction == Right:
    s = (d - w*n)/(n+1)
    for i in range (1,n+1):
        prev_x = clonePosStack[i-1].x
        prev_y = clonePosStack[i-1].y
        if direction == Left:
            p = Point(prev_x+w+s,prev_y)
        else if direction == Right:
            p = Point(prev_x-w-s,prev_y)
        clonePosStack[i].append(p)
//location is vertical
else if direction == Up or direction == Down:
    s = (d - h*n)/(n+1)
    for i in range (1,n+1):
        prev_x = clonePosStack[i-1].x
        prev_y = clonePosStack[i-1].y
        if direction == Up:
            p = Point(prev_x,prev_y+h+s)
        else if direction == Down:
            p = Point(prev_x,prev_y-h-s)
        clonePosStack[i].append(p)

Figure 4-3. Pseudo-code for determining the new spacing distance and re-filling the clone position stack
with new locations
4.3 Clone Editing

In the implementation of Skencil-C, clones are represented in a tree, which is stored as a list of lists. The original object is located at the root of the tree. Editing all objects in the tree is then essentially the task of traversing the entire tree and changing each object. Editing all the objects following some object in the tree is a task of traversing the list where the object is located starting at that object until the end of the list and making the changes to all nodes encountered. The same applies, if the objects are arranged into a list, which is a branchless tree.

When a direct cloning transaction is completed, the clones are inserted into the document and a reference to each newly created clone is inserted into the clone tree list. This list maintains the references of the objects in the clone tree in the order of creation to preserve the dependencies between them. For example, the cloning structure shown in Figure 3-1c, can be stored internally as a list \([1, 2, 3]\), where the numbers represent the memory addresses of the clones. The clone tree list has a nested structure to accommodate the tree branches. The clone tree mentioned in Figure 3-1b has only a single leaf; therefore its representation doesn’t require a nested list. However, the example in Figure 3-7 requires a nested representation, which is \([1, 2, [4, [6, 7], 5], 3]\). Clone “4” was added later; therefore its memory address number is larger than the others’. The new object depends on clone “2”, which explains its position. Given this data structure, and assuming that clone “4” is edited, the changes are then applied to all the clones following, i.e. the clones “6”, ”7”, and “5” are affected.
as expected. Clone “6” is placed inside a bracket which indicates that it forms a branch. Therefore, \([6, 7]\) is a branch and a sub-tree belonging to the stem \([4, [6, 7], 5]\). Similarly, branch \([4, [6, 7], 5]\) belongs to the stem \([1, 2, [4, [6, 7], 5], 3]\).

The data stored in the clone tree list is also used to draw the arrows between the clones indicating the dependencies and the order of creation. Also, it is used for editing, since it reveals which clones must be modified if the user chooses to “apply changes to all following” or “apply changes to all” when the dialog box pops up. The methods for drawing the arrows and editing are the same and they are named as \texttt{EDIT\_ALL} and \texttt{EDIT\_FROM}. The code excerpts in python for these methods are provided in the pseudo-code below. They correspond to a heavily simplified form of the original code in Skencil-C written in python, see Figure 4-4a,b. A lot of details have been omitted regarding the syntax of the clone tree list structure, the drawing of arrows, the implementation of step editing, and the editing process itself, which involves duplication, replacement of the clones and updating the undo information.
//the following method edits all the clones belonging to a tree

def EditAll(clone):
    //first find the enclosing list of the clone
    enclosed_list = find_enclosed_list_of(clone)
    edit_all(enclosed_list, clone)

def edit_all(tree, clone):
    for item in tree:
        if item[0] is a list:
            edit_all(item, clone)
        else: //item is not a list
            edit(item)
Interpolation between the root and the selected object is implemented by replacing every intermediate object with a copy of the previous one and by adding the interval values to the parameters of intermediate objects. Initially, the differences between the height and the width of the root and the selected object are calculated and the interpolation
increments are obtained by dividing the differences by the number of the intermediate objects. The intervals of color interpolation are obtained by performing the same operations on RGB values, see Figure 4-5.

```plaintext
//obtaining size intervals
diff_x = sel_width - root_width
diff_y = sel_height - root_height
interval_w = diff_x / n
interval_h = diff_y / n

//obtaining colors intervals
diff_r = sel_col[0] - root_col[0]
diff_g = sel_col[1] - root_col[1]

interval_col_r = diff_r / n
interval_col_g = diff_g / n
interval_col_b = diff_b / n
```

Figure 4-5. Pseudo-code for obtaining size and color intervals for interpolation

### 4.3.2 The Undo Mechanism

Skencil allows the user to undo every operation that is performed on a document (Skencil, 2005b). To achieve this, the information on how to undo any operation is maintained. Each function that changes the state of the document returns the corresponding undo information. This information is represented by a tuple, which
consists of the method name and its arguments. When the method changes the state of the
document, a tuple containing the undo information is returned. Then, this information is
pushed onto the undo stack.

Figure 4-6 shows a hypothetical example of a method that attempts to change the
width of an object. When the method is called, the undo tuple is created. This undo tuple
consists of the method `SetWidth` and its argument, which holds the current value of the
width. After the tuple is created, the width is assigned to a new value and the undo tuple
is returned. The caller of the function pushes the undo tuple onto the document stack.
Now, if the user wants to undo the operation, the tuple is retrieved and assembled into a
function call, which sets the width to the old value. The tuple containing the new value is
returned and this is used to create the redo information.

```python
def SetWidth(self, width):
    undo = self.SetWidth, self.width
    self.width = width
    return undo

//Somewhere in the document class, where the SetWidth method is
called:
undo = self.SetWidth(width)
self.add_undo(undo)
```

**Figure 4-6.** Pseudo-code for adding the undo information

This undo information method was also employed to enable the user to undo
cloning operations in Skencil-C. Support for both deletion of the cloned object and
undoing the operations was added. This enables users to undo operations without restarting their work and also enables a better evaluation of the usability of the new interface, since undo and erase/delete operations can be useful and low-cost indicators of usability problems (Akers, Simpson et al., 2009).

The easiest way to implement undo is to keep a copy of the clone tree list after each operation. For this, a method was implemented that returns copies of clone trees, along with a method that sets the clone tree list and returns the undo information. The pseudo-code for a typical cloning undo appears in Figure 4-7.

```python
def SetCloneTree(self, clone_tree_list):
    clone_tree_list_copy = []
    self.copy_tree_list(clone_tree_list_copy, self.clone_tree_list)
    undo = self.SetCloneTree, [clone_tree_list_copy]
    self.clone_tree_list = clone_tree_list[0]
    return undo

//The caller:
//The clone tree list is first preserved
clone_tree_list_copy = []
self.copy_tree_list(clone_tree_list_copy, self.clone_tree_list)
//some functions are called which modify the copy of the tree list
//And the undo information is created and pushed onto the stack
self.add_undo(self.set_clone_tree([clone_tree_list_copy]))
```

*Figure 4-7. Pseudo-code for Cloning Undo*
Chapter 5
Preliminary User Study

This chapter presents the results of a preliminary user study which explored the usability of the new cloning interfaces. It starts with the description of the experimental design, followed by an analysis of task completion times. The chapter analyzes error rates and continues with an analysis of questionnaires and participant feedbacks. The chapter introduces the modification to the user interface based on the results and ends with a discussion and a conclusion.

5.1 Evaluated Interfaces

Our novel cloning interface was implemented by extending Skencil, an open source vector graphics editor written in Python. We named our system Skencil-C (Skencil, 2005a).

Freeform direct cloning was evaluated against duplication. Linear cloning and interpolation were evaluated against an existing tiled cloning interface as implemented in Inkscape. Tiled cloning allows creation of clones by placing them into “tiled” grid arrangements. Just like our cloning technique, tiled cloning allows interpolation of scale and color row-wise and column-wise. We re-implemented tiled cloning in Skencil-C to ensure that our evaluation compares the differences between the techniques and not between other attributes of the two systems. We implemented only those features of tiled cloning that are meaningful to be evaluated against our cloning technique.
Our user interface for tiled cloning is illustrated in Figure 5-1. The following functionality is supported: specifying the number of clones in each row and column of a grid, shift percentages for each row and column to change the spacing between objects, scale percentages for rows and columns in both X and Y, and color percentages for each color channel for each row and column. Three buttons are available: a “Reset” button to clear all fields, a “Remove” button for deleting all clones of the currently selected object, and a “Create” button to create clones from the selected object. Subsequent presses of “Create” on the same set of clones can be used for corrections. Every field can take negative values or be zero, except for the number of clones per row and column. Alternation check marks for each parameter in row and column were also added, but also not used during the experiment.
5.2 Participants

Twelve paid participants (8 males) were recruited. The participants were between 20 and 35 years of age with a median of 26 years. An average computer use of 42.7 hours per week was reported and 2.38 years of experience with 2D graphics editing software. Two of the participants reported left-handedness, but chose to perform the experiment with the right hand. None of the participants had previous experience with the system, or the cloning interfaces.
5.3 **Apparatus**

The user study was conducted using a high-end laptop with a 22” external display at 1680×1050 in full-screen mode and a USB wheel mouse. All events and timings were logged. No audio feedback was provided to the participants.

5.4 **Procedure and Experimental Design**

We used a repeated measures design. Ten different tasks and two task sizes (small and large) were used to compare linear direct cloning vs. tiled cloning and freeform direct cloning vs. duplication – a 10×2×2 design. Hence, participants performed each task four times, for two different sizes and with two different techniques. Based on a small pilot, this was the maximum number of repetitions we could ask participants to perform within the allocated hour. The measured dependent variables were task completion time and the number of corrections, such as undo, <DELETE> key, excessive “Create” button presses, and excessive editing operations.

There was a combination of cloning and hybrid tasks, which require both clones creation and editing, see Table 2. Tasks 1 to 7 required only cloning. Small creation tasks required creating three new clones, while larger tasks involved nine new clones. The small hybrid editing tasks required creating four new clones, while the larger one required nine. The tasks were grouped into six categories during the experiment and referred to as sessions. After each session participants took a small break and were asked to fill out a part of the questionnaire for the completed session. Figure 5-2 illustrates
smaller tasks as they appeared to participants. Task 5 & 6 are similar to 2 & 3, just with different spacing.

Figure 5-2. Tasks
### Table 2 Description of tasks

<table>
<thead>
<tr>
<th>Task</th>
<th>Cloning techniques compared</th>
<th>Size (no. of clones)</th>
<th>Default row spacing</th>
<th>Default column spacing</th>
<th>Clone arrangement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Direct/Duplication</td>
<td>4/10</td>
<td>N/A</td>
<td>N/A</td>
<td>Curve</td>
<td>Creation</td>
</tr>
<tr>
<td>2</td>
<td>Direct/Tiled</td>
<td>4/10</td>
<td></td>
<td>Yes</td>
<td>Row</td>
<td>Creation</td>
</tr>
<tr>
<td>3</td>
<td>Direct/Tiled</td>
<td>4/10</td>
<td>Yes</td>
<td></td>
<td>Column</td>
<td>Creation</td>
</tr>
<tr>
<td>4</td>
<td>Direct/Tiled</td>
<td>4/10</td>
<td>Yes</td>
<td>Yes</td>
<td>Grid</td>
<td>Creation</td>
</tr>
<tr>
<td>5</td>
<td>Direct/Tiled</td>
<td>4/10</td>
<td></td>
<td>No</td>
<td>Row</td>
<td>Creation</td>
</tr>
<tr>
<td>6</td>
<td>Direct/Tiled</td>
<td>4/10</td>
<td>No</td>
<td></td>
<td>Column</td>
<td>Creation</td>
</tr>
<tr>
<td>7</td>
<td>Direct/Tiled</td>
<td>4/10</td>
<td>No</td>
<td>No</td>
<td>Grid</td>
<td>Creation</td>
</tr>
<tr>
<td>8</td>
<td>Direct/Tiled</td>
<td>5/10</td>
<td></td>
<td>Yes</td>
<td>Row</td>
<td>Size interpolation</td>
</tr>
<tr>
<td>9</td>
<td>Direct/Tiled</td>
<td>5/10</td>
<td></td>
<td>Yes</td>
<td>Row</td>
<td>Color interpolation</td>
</tr>
<tr>
<td>10</td>
<td>Direct Cloning and Clone Editing vs. Tiled Cloning and Rectangular Selection Editing</td>
<td>5/10</td>
<td>Yes</td>
<td>Yes</td>
<td>Grid</td>
<td>Coloring</td>
</tr>
</tbody>
</table>

The sessions were counterbalanced using a 6×6 balanced Latin Square. Within each session the tasks, techniques and task sizes were also counterbalanced. If there was
only one task in the session, only the task size and technique were counterbalanced. At the start of each new session participants were given a practise session to get familiarized with the requirements of the task(s). Each trial began with a pop-up message informing participants which technique (direct cloning, tiled cloning, or duplication) was to be used to complete the task. Control measures were implemented to prevent participants from using the wrong technique by mistake. Participants were also required to match the displayed targets accurately, with differences up to two pixels being acceptable. Beyond that, the result had to be corrected.

The first session included only the a single task – creation of objects along a curve. Two techniques were compared: freeform direct cloning and duplication. The direct cloning condition required participants to drag the given object to the right along the specified path, see Task 1 in Figure 5-2. The lighter colored objects illustrated the target locations for the clones. In this task direct cloning was evaluated against duplication, as tiled cloning does not support free-form paths. In the duplication mode, the participants duplicated the object on the left side of the screen, via CTRL-D, and then dragged it to the first target location. Then, the participants had to select the newly duplicated object, duplicate and move it, and so on. No snapping onto the path was provided. The spacing of the clones during direct cloning was fixed and unchangeable. As a result, the participants had to follow the path accurately so that the clones would be placed precisely in the target slots for the direct cloning condition. Although this makes
the task somewhat artificial, we still decided to include it because we wanted to see how well this works relative to raw duplication.

The second session included tasks 2, 3 and 4. All these three tasks required fixed spacing, matching the default settings for both direct cloning as well as tiled cloning. So, the task did not require any adjustments for spacing for both techniques. In task 2, the participants were asked to create a row of clones, a column in task 3, and a grid in task 4, see Figure 5-2. The size was four objects along one dimension for small tasks and ten for large tasks. For the grid task in the direct cloning condition the participants were required to first create a row of clones, then to select that group, and then to create a column of clones from it.

The third session contained tasks 5–7. These tasks were the same as tasks 2–4 with the exception that the spacing was different, see Task 7 in Figure 5-2. Hence, in the tiled cloning condition, the participants had to iteratively guess the shift values. Spacing between targets differed for different size condition to avoid confounds due to memorization. However, the spacing values for direct cloning and tiled cloning were identical, since there is no possibility for interference between the two user interfaces.

The forth session consisted of a single size interpolation task, see Task 8 in Figure 5-2. For the direct cloning condition, the participants were asked to create a row of clones from a given object, to rescale the last object and perform interpolation to match the parameters of target slots. In the tiled cloning condition, the participants were asked to
guess the values for the scale factors in the tiled cloning menu. Similar to session 3, the same measures against memorization were used.

Session 5 included only task 9, a color interpolation task. The work area and reference image were displayed next to each other, see Task 9 in Figure 5-2. The layout used again the default spacing. Target colors were picked different for each of the two size conditions to avoid memorization effects. In the direct cloning condition, the participants needed to color the last object and then interpolate. In the tiled cloning condition, the participants needed to guess the increments of R, G and B values to complete the task. As an aid, the participants were told which value to change, but not by how much or the sign of the change.

The sixth session consisted of a coloring task. In this task participants had to both create a grid of clones with default spacing and then color the objects into the colors as in the reference image to the right, see Task 10 in Figure 5-2. In the direct cloning condition, participants were asked to create a grid just like in task 4. Then, they had to change the color of some objects and apply the changes to all following objects in order to obtain the design in the reference image. For the tiled cloning condition, participants had to create a grid of clones using the tiled cloning menu. Since tiled cloning doesn’t provide any means for editing the attributes of individual clones and spreading the changes to a subset of cloned objects, participants had to select individual objects using rectangular selection and re-color them. The participants were informed about the fastest
way to do this, which is to select the entire grid first and to color it, then to select various subsets of the grid and coloring the selections.

5.5 Observations from Pilots of this Preliminary User Study

Prior to this full-scale study, two pilot studies were conducted to identify problems with the interfaces and the initial experimental design. Four volunteers were recruited for the first pilot study and another five for the second from among the graduate students in Human-Computer Interaction and related fields. As a result, the interfaces and certain other conditions of the experiment were tweaked according to the participants’ feedbacks and performance. The full-scale study was compressed to include fewer tasks so that the study would fit in less than an hour. The pilot studies helped to reveal a number of problems with the initial implementation of the interface. Initially, the only available method to change the spacing between clones was by changing the number of clones. The second method which employs changing the gaps between the clones by shifting the last outline was not implemented prior to the pilot studies. The first pilot study quickly identified that changing the spacing using this method is too complex for an average user with no prior experience using this interface. The completion times for the tasks with non-default spacing were too long and participants were unable to place clones accurately on average. As a result, spacing of clone outlines using the shift method was implemented, which improved the results greatly.

A problem was fixed with activating direct cloning itself. Prior to the pilot studies, it was only possible to activate the state while the object was dragged and not
before. Nearly all the participants demonstrated struggling when trying to activate the state. One of the participants complained about this.

Another problem with direct cloning technique was the observed struggle with completing the cloning operation. Prior to the pilot studies, the user would have to terminate the cloning by releasing the mouse button first. If the `<ALT>` or the `<WINDOW>` key is released first, then the system would revert to the dragging mode deleting all the clone outlines. Most participants tended to release both the mouse button and the pressed key at the same time. Obviously, the two cannot be released simultaneously, since every event has a unique timestamp. Therefore, there are equal chances that the user would release the key first or the mouse button first. This problem was eliminated by disabling reversion to the moving state when direct cloning was activated once during a transaction.

5.6 Statistical Models

Analysis of variance (ANOVA) is a collection of statistical models, and their associated procedures, in which the observed variance is partitioned into components due to different explanatory variables. In all the analyses that appear in this thesis we used a repeated measures ANOVA model with significance level of 0.05 ($\alpha=0.05$), i.e. a chance of 1 in 20. This is the norm for the field of Human-Computer Interaction research. The result of an ANOVA uses the $F$ statistic, named after Sir Ronald A. Fisher, and the corresponding probability value $p$. If a test of significance gives a $p$ value lower than the $\alpha$-level, the null hypothesis is rejected. Such results are informally referred to as
“statistically significant”. The Tukey–Kramer method is a single-step post-hoc multiple comparison procedure and statistical test generally used in conjunction with an ANOVA to find which means are significantly different from one another. It compares all possible pairs of means and identifies where the difference between two means is greater than the standard error would be expected to allow. In our analyses we used the Tukey-Kramer post-hoc test to reveal groupings, whenever the ANOVA results were significant.

5.7 Results

No ordering effects were observed, as session order had no significant main effect, $F_{5,6} = 2.72, p > .05$. The interaction with session order was also not significant, neither for task, $F_{45,54} = 1.24, p > .05$, nor for technique, $F_{5,6} = 3.13, p > .05$, nor for task size, $F_{5,6} = 1.84, p > .05$.

The main effect of technique was significant, $F_{1,6} = 189.44, p < .0001$ in favour of direct cloning. Average task completion times with standard errors for each of ten tasks are presented Figure 5-3.
A Tukey-Kramer analysis was performed to reveal task groupings. There was a grouping of Tasks 2, 3, and 4 and tasks 5 and 6. The first group happened to include all the tasks in the second session. The second group partially included tasks in session 3. We report all results according to this grouping. All reports on the significance of size in this study imply tasks for the smaller size (1) took less time than for the larger size 2.
5.7.1 Task 1

There is a significant difference for technique, $F_{1,6} = 107.41, p < .0001$, with direct cloning being faster than duplication. Also, the interaction between technique and task size was significant, $F_{1,6} = 193.49, p < .0001$, with duplication taking significantly longer than direct cloning for larger tasks. This is not very surprising as duplication is much more repetitive than direct cloning along the path.

5.7.2 Task 2, 3, 4

A Tukey-Kramer analysis of the group consisting of tasks 2–4 puts tasks 2 and 3 into a single group. This means that there is no significant difference between row and column creation for cloning with fixed spacing. Further analysis of this subgroup (tasks 2 and 3) revealed a significant difference for technique, $F_{1,6} = 320.40, p < .0001$, with direct cloning being faster, and size, $F_{1,6} = 6.14, p < .05$.

Analysis of task 4 found no significant difference between the techniques, $F_{1,6} = 2.99, p > .05$, and no significant difference for size, $F_{1,6} = 4.28, p > .05$. Nevertheless, in both cases, the average completion times for direct cloning techniques were slightly slower. It can be hypothesized that the necessary intermediate selection in the direct cloning method for the creation of a grid produced enough overhead to slow it down. We saw this as motivation to redesigning the grid cloning technique, see Section 5.11.
5.7.3 Tasks 5 and 6
There is a significant difference between direct and tiled cloning for tasks 5 and 6, \( F_{1,6} = 91.27, p < .0001 \), with direct cloning being faster, and for size, \( F_{1,6} = 24.93, p < .005 \). There is no significant difference for task, \( F_{1,6} = 1.74, p > .05 \). This indicates that there is no fundamental difference between row and column creation.

5.7.4 Task 7
There is a significant difference for technique on task 7, \( F_{1,6} = 10.02, p < .05 \), with direct cloning being faster, and for size, \( F_{1,6} = 19.03, p < .005 \).

5.7.5 Task 8
There is a significant difference for technique on task 8, \( F_{1,6} = 66.45, p < .0005 \), with direct cloning being faster. Size was not significant, \( F_{1,6} = 5.10, p > .05 \).

5.7.6 Task 9
Analysis of task 9 reveals that direct cloning is significantly faster than tiled cloning overall, \( F_{1,6} = 108.10, p < .0001 \), with direct cloning being faster, and for size, \( F_{1,6} = 17.10, p < .01 \). The interaction between technique and task size was also significant, \( F_{1,6} = 11.45, p < .05 \), with tiled cloning taking much longer than direct cloning for larger tasks. These two observations may suggest that tiled cloning is not well suited for larger interpolation tasks.
5.7.7 Task 10

No significant difference was found for this task between direct cloning and tiled cloning, $F_{1,6} = .41$, ns. Although not significant, the average completion time for direct cloning was slightly faster for the larger, but not for the smaller tasks. Task size was significant, $F_{1,6} = 368.98, p < .0001$. Given that our analysis of task 4 revealed no significant difference for the creation of a grid, we excluded the time for the grid creation by post-processing the experimental logs. Performing an ANOVA on the filtered, editing-only completion times also revealed no significance $F_{1,6} = 3.22, p > .05$. The size was significant, $F_{1,6} = 305.57, p < .001$.

5.7.8 Error Rates

The use of undo and erase events was previously demonstrated to be a useful complement to user reported critical incidents for low cost usability evaluation of creation-oriented applications like SketchUp (Akers, Simpson et al., 2009). We logged the number of undo and delete operations in the use of both techniques and number of “Create” button presses for tiled cloning, since all the uses of this operation except the first one are not necessary and thus can be considered as another low-cost usability measure. We observed the interaction between the task size and the number of undo operations. Very high number of create button uses was logged during non-default spacing creation and interpolation as opposed to creation tasks with default spacing. This came to no surprise.

As a measure of error rates, the number of undo operations ($N_{\text{undos}}$) and delete operations ($N_{\text{delete}}$) were logged. Moreover, the number of extra “Create” button presses
(N_{ecbp}) for the tiled cloning condition was also logged, since all the uses of this operation except the first one are excessive. ECBP stands for Extra “Create” Button Presses. Similarly, the number of extra resize operations (N_{ero}) was logged for direct cloning in task 8.

Technique was significant for undos, $F_{1,6} = 14.14, p < .001$. Direct cloning had a higher undo rate than tiled cloning. The size of the task was also significant, $F_{1,6} = 9.00, p < .05$. Larger tasks had higher undo rates. Task itself was not significant, $F_{1,6} = .56, 	ext{ns}$. The difference for delete operations was insignificant and the operation was rarely used. Participants mostly tended to re-use the “Create” button to undo operations for tiled cloning rather than using undo or delete.

Larger size tasks had significantly higher ECBP rates, $F_{1,6} = 6.92, p < .05$. Task had significantly more ECBP’s, $F_{9,54} = 53.28, p < .0001$. A Tukey-Kramer test revealed that tasks in session 2 had significantly lower ECBP rates than tasks in session 3. This was to be expected, since tasks with varying spacing are harder to complete. Tasks, such as tasks 5–9, where parameters had to be guessed, have much higher ECBP rates. Tasks 5–7 involve the spacing of clones, whereas task 8 involved scale factors and task 9 involved color interpolation.

Task size was not significant for N_{ero} (Extra resize operations), $F_{1,6} = 1.60, p > .05$. This is fairly obvious because any resize operation is performed on the last object only, irrespectively of the task size.
To estimate which technique had more corrections overall, a new measure, called $C$, for corrections, was introduced and sums $N_{\text{undos}}$, $N_{\text{delete}}$, $N_{\text{ecbp}}$, and $N_{\text{ero}}$.

$$C = N_{\text{undos}} + N_{\text{delete}} + N_{\text{ecbp}} + N_{\text{ero}}$$

$C$ is visualized in Figure 5-4, and largely dominated by $N_{\text{ecbp}}$. Overall, for all tasks, there were more corrections for tiled cloning than for direct cloning (duplication in task 1), $F_{1,6} = 254.52, p < .0001$. The average correction rate for direct cloning technique was 0.25 and 1.5 for tiled cloning (duplication in task 1). A Tukey-Kramer analysis revealed two groups: a group with tasks 1–4 and 10 and a group with tasks 5–9. Tasks in the second group had higher correction rates. No significant difference between techniques or task sizes was found for the first group. Further analysis of the group with tasks 5 to 9 revealed a significant difference for technique, $F_{1,6} = 235.85, p < .0001$. For all tasks the correction rate in direct cloning is significantly lower than any correction rate in any tiled cloning task. A significant difference was found for task size, $F_{1,6} = 12.29, p < .05$. The interaction between technique and size was also significant, $F_{1,6} = 9.60, p < .05$. Direct cloning had fewer corrections overall than tiled cloning.
5.8 Questionnaire

After the end of each of the six sessions, participants were asked to fill out a survey about the session they just completed. Each question rated the two evaluated techniques with respect to the following factors: speed, accuracy, preference for smaller task, preference for larger task, and overall rating. A 7 point Likert scale was used, with the endpoints,
labelled -3 and 3, designated as “strongly prefer” for each technique, see Figure 5-5. After completion of all sessions, another questionnaire asked participants to rank the techniques in general across all tasks.

<table>
<thead>
<tr>
<th>Technique</th>
<th>Speed</th>
<th>Accuracy</th>
<th>Smaller task</th>
<th>Larger task</th>
<th>Overall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Condition</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Direct cloning</td>
<td>Strongly prefer</td>
<td>Prefer</td>
<td>Somewhat prefer</td>
<td>No preference</td>
<td>Somewhat prefer</td>
</tr>
<tr>
<td>Direct cloning</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Direct cloning</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Direct cloning</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Direct cloning</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

*Figure 5-5 Example of a ranking question from the questionnaire featuring a Likert scale*

The questionnaire data for speed shows preference for direct cloning across all sessions, average = -1.9, sd = 1.7.

For accuracy and for all tasks except the first one, participants again preferred direct cloning, average = -0.8, sd = 2.2. In the first session, there was a preference for duplication, average = 2.1, sd = 1.52, as it was easier to achieve high accuracy with this technique.

For smaller tasks, participants again preferred direct cloning clearly in sessions 1 to 5 and overall, average = -1.26, sd = 2.0. For task 1, participants preferred duplication, average = 0.67, sd = 1.8. For larger tasks, there was again a preference for direct cloning,
average = -0.9, sd = 2.2. The rating on task one was close to neutral, though, average = 0.25, sd = 1.8.

Finally, the overall preference for technique shows a clear preference for direct cloning in all instances, average = -1.2, sd = 1.97, except for task 1, where duplication was moderately preferred, average = 0.8, sd = 1.8.

Figure 5-6 illustrates charts of preference of techniques for each asked condition and session.

Figure 5-6. Feedback charts. Error Bars: ± 1 SE
5.9 Participant Feedback

In general there were positive comments from participants about the new direct cloning technique. One participant commented explicitly that tiled and duplication techniques were better for accuracy. Another participant felt that the snapping feature of linear direct cloning was helpful and stated that direct cloning was good for smaller tasks in general. This is also confirmed by the average rating for smaller tasks. Yet another participant stated that the intermediate selection operation in the linear direct cloning method for grid creation is a disadvantage for the technique, which is also visible in the results for task 4. This has been addressed, see Section 5.11. One participant did not like that there is a “phantom” outline associated with the cursor, which does not designate a cloned object. This is a valid concern and should be addressed in the future to reduce confusion.

5.10 Discussion

Overall, direct cloning is faster for creating and editing clones for almost all tasks. Linear direct cloning works very well for creation tasks, regardless if spacing is at the default setting or not. The exception is grid creation, where the additional step for selecting and grouping a row of clones slows down the performance of the technique. This is one of the limitations of the tool which is addressed in the next section. Even with that handicap, the participants made much fewer errors with direct cloning than tiled cloning.

While direct cloning was significantly faster than duplication for a free-form creation task, any task that requires accurate placements seems to be frustrating for users,
as revealed by the questionnaire data. As one of the future directions of this research, we plan to investigate the performance of direct cloning along predefined paths such as an arc, a diagonal line, or a free-form curve.

Some interesting behaviours were observed during the experiment. For all tasks involving variable spacing, participants were allowed to use any of the two spacing methods, moving the cursor to adjust spacing or changing the number of clones. The frequency of moving the cursor to change the spacing was higher overall. This is not that surprising, as it is simpler to get the number of clones right (which is visually easy to verify for reasonably small object numbers), compared to getting the spacing correct.

For interpolation tasks, direct cloning was significantly faster every time. But for these interpolation tasks, correction rates for tiled cloning skyrocketed, as this essentially involves guessing parameters. Moreover, and again with tiled cloning, participants were often confused about whether to increase a scaling factor or to decrease it, also due to the fact that amounts could be negative in certain fields. Understanding that issue requires technical knowledge that is not commonly found in users.

For the coloring task, there was no difference overall. Limiting the analysis to the editing portion of the task and larger task sizes, it could be shown that direct cloning was faster, but the difference was not significant. However, in hindsight, this task was quite complex and involved several sub-tasks, which makes it hard to draw concrete conclusions. Moreover, it can be argued that editing cloned structures is not necessarily faster if there are only one or two clones. This was the only task that involved editing
operations using clone structures and the calendar editing technique. The significance wasn’t achieved and one can argue that a new better controled study is required. The second user study described in Chapter 6 addresses this, as it reveals more about the effectiveness of the calendar inspired editing technique.

The data collected from the questionnaires with respect to significance reveals that participants preferred direct cloning technique overall. The exception was task 1, where duplication was preferred due to the higher accuracy that can be achieved.

### 5.11 Modified Grid Creation Method

Based on the results of the preliminary study, we implemented a new version of the grid creation method. This version avoids the intermediate step of having to select the newly created row of objects. Now, a single, diagonal mouse drag directly creates a grid of clones, see Figure 5-7.

![Figure 5-7. Creating a grid of clones in one operation](image)

83
5.12 Conclusion

We performed a preliminary user study to explore the performance of the direct cloning method. For this, we re-implemented the tiled cloning user interface from Inkscape, where the user can specify repetition parameters in a dialog.

The results of this study showed that direct cloning is practically almost always faster than tiled cloning, which is not very surprising as finding the numeric values for various input fields is less intuitive than direct manipulation. Also, tiled cloning needed more “undo” type of operations (including re-creates) on average. We also found no significant difference between row and column creation. However, grid creation was not well supported due to the necessity to first duplicate objects into a row (or column) and then to duplicate that into the grid. We addressed this by modifying the grid creation method, see above.

Additionally, if the spacing of objects did not match preset values, tiled cloning was significantly slower, and also had more “undo” operations. Direct cloning of objects along a free-form path is faster, but not always as accurate as direct placement. Unsurprisingly, all clone-based creation and editing operations on a set of cloned objects (such as creating and then coloring or interpolating) are faster in general compared to tiled cloning. Finally, participants had a marked preference for direct cloning in general.
Chapter 6
User Studies

We explored the performance of the direct cloning method in the preliminary user study described in Chapter 5 which gave us an insight on how to evaluate the interface. Based on the results from this study, we designed two user studies which explore the interface further. The goal of the first user study was to compare the modified grid creation method with *smart duplication* and tiled cloning. The goal of the second study was to explore the power of the clone editing interface in a more controlled environment than in task 10 of the preliminary user study.

6.1 Participants

Twelve paid participants (10 males) were recruited. The participants were between 19 and 30 years with a median of 24. An average computer use of 48 hours per week was reported and 4.45 years of experience with either diagram editors or presentation software as well as 3.58 years with drawing applications. Two of the participants reported left-handedness, but chose to perform the experiment with the right hand. None of the participants had previous experience with Skencil-C or cloning. These participants were used for both user studies described here.

6.2 Apparatus

The same apparatus was used as in the preliminary user study, see Section 5.3.
6.3 User Study 1

The goal of this user study was to compare *smart duplication* with the modified grid creation method and tiled cloning. Horizontal or vertical alignment was supported, when the `<ALT>` key is held. Participants were asked to use this feature during the study.

6.4 Procedure and Experimental Design

We used a $3 \times 2 \times 3$ repeated measures design. Three tasks and two task sizes (small and large) were used to compare linear direct cloning vs. tiled cloning vs. smart duplication. Each participant preformed only one repetition. The measured dependent variable was task completion time. The set of tasks focused only row and grid creation, because there was no evidence in the preliminary study that row and column creation would be different. Task 1 featured row creation while task 2 featured grid creation. The spacing of objects in both tasks matched the default preset for both direct and tiled cloning, whereas in smart duplication the first copy had to be positioned manually. Task 3 featured grid creation with non-preset spacing, i.e. the default value for object spacing was different than what was needed to perform the task. Task 3 was evaluated in a separate session because the preliminary study revealed that participants tended to get confused when faced with too many similar conditions. The two sessions were counter-balanced and the tasks, techniques and sizes were also counter-balanced within each session. Control measures were used to prevent the participants from using the wrong technique. We
collected all the error, undo and accuracy information like in we did in the preliminary study. However, we did not analyse it or report it.

At the end of the each session, participants were asked to rank each of the three techniques on a scale from 0 to 6. Moreover, participants were asked to provide an overall ranking for the techniques at the end of the experiment.

![Figure 6-1. Tasks in user study 1, (a) task 1, (b) task 2 and 3](image)

### 6.5 Results

Overall, and as computed by an ANOVA, the main effects of technique \( (F_{2,20} = 20.22, p < .0001) \), task \( (F_{2,20} = 39.12, p < .0001) \) and size \( (F_{1,10} = 38.13, p < .0001) \) were significant. The interaction between technique and task was also significant, \( F_{4,40} = 11.05, p < .0001 \), see Figure 6-3. No ordering effects were observed, as session order had no significant main effect, \( F_{1,10} = 3.06, p > .05 \). The interaction with session order was also not significant, neither for task, \( F_{2,20} = 3.31, p > .05 \), nor for technique, \( F_{2,20} = 0.81, ns \), nor for task size, \( F_{1,10} = 3.10, p > .05 \). A Tukey-Kramer analysis was performed to detect task groupings.
There was a grouping of tasks 1, 2 (fastest), and task 3 (slowest). The groupings were consistent with sessions. We report all further results according to this grouping.

![Figure 6-2. Average task completion times for User Study 1. Error Bars: ± 1 SE](image)

### 6.5.1 Tasks 1 and 2

There was a significant difference for technique, $F_{2,20} = 48.26, p < .0001$, task, $F_{1,10} = 12.43, p < .01$ and size, $F_{1,10} = 10.51, p < .01$. A Tukey-Kramer analysis revealed all three techniques to be different, with direct cloning being the fastest and smart duplication the slowest. The interaction between technique and size ($F_{2,20} = 4.77, p < .05$) was significant. Direct cloning for both size sets was faster than for both size sets of
smart duplication and faster than large tiled cloning tasks. The interaction between task and technique ($F_{2,20} = 16.14, p < .0001$) was also significant.

### 6.5.2 Task 3

There was a significant difference for technique, $F_{2,20} = 12.03, p < .0005$ and size, $F_{1,10} = 8.99, p < .05$. A Tukey-Kramer analysis revealed that direct cloning is faster than tiled cloning, but not smart duplication. Although direct cloning and smart duplication were not significantly different, direct cloning was still faster on average (17.44 s vs. 32.25 s). The interaction with size was not significant, $F_{2,20} = 0.40$, ns.

![Figure 6-3. Main interaction between task and technique. Error bars: ± 1 SE](image-url)
6.5.3 Error Rates

Similar to the preliminary study we logged the number of undo and delete operations in the use of all the cloning techniques. Technique was significant for undo’s, $F_{2.20} = 9.47, p < .005$. A Tukey-Kramer analysis revealed that smart duplication had significantly more undo’s (0.78 average) than tiled cloning (0.00) and direct cloning (0.01), see Figure 6-4. Task was not significant, $F_{2.20} = 0.70$, ns, and neither was size, $F_{1.10} = 0.20$, ns. None of the participants used the <DELETE> key, so no delete operations appeared in the logs.

![Figure 6-4. Mean number of undo’s. Error bars: ± 1 SE](image)
6.6 Discussion

Direct cloning outperformed smart duplication when the spacing preset matched the desired distance. For a non-matching preset direct cloning was not significantly faster than smart duplication, but still faster on average. We expect that this result may reach significance with more subjects. The study demonstrated also that the modified grid creation technique outperformed tiled cloning. Questionnaire analysis revealed that participants prefer direct cloning in both default, non-default spacing tasks as well as overall, see Figure 6-5.

We also compared some of the data from our preliminary with this study for creating a row of clones with non-preset spacing. This was to compensate for the fact that we did not include such a condition in this first study. That comparison indicates that smart duplication is somewhat faster on average than direct cloning for creating clones with non-preset spacing. However, as this is a cross-study comparison with non-matching sets of tasks we cannot say if this difference is statistically significant.
6.7 User Study 2

Clone editing is another important aspect of our cloning interface. In the preliminary study we saw evidence for it’s effectiveness, but encountered problems since we mixed creation with editing. This made that task too complex and compromised internal validity. Hence, we changed this in this second user study to get more precise information. All clones and their topologies were preloaded for each trial, so that participants didn’t have to create them. The tasks then required the user to (re-)color objects with at most three colors. This tested if the participants were capable of

Figure 6-5. Mean ranks for each technique in User study 1, higher is better. Error bars: ± 1 SE
understanding different arrangements of clones and how quickly they could edit them. Another aspect was to investigate how clone editing compares to editing with rectangular selection combined with the ability to add/subtract objects with the <SHIFT> key. A small pilot study revealed that some participants had trouble understanding the structure of non-grid cloning arrangements. We extended Skencil to designate the master object by a dashed outline to fix this issue, see Figure 6-6.

![Figure 6-6. Master object designated by a dashed outline](image)

### 6.8 Procedure and Experimental Design

We used a repeated measures design. Five tasks and two task sizes (small and large) were used to compare linear clone editing with rectangle selection editing in a 5x2x2 design. All 20 conditions were counter-balanced. The measured dependent variable was task completion time. The participants and apparatus from user study 1 were re-used for this study.

We used two different sizes. Size 1 tasks featured three clones counted from the master object or stem, and size 2 tasks seven clones. Tasks 1 and 2 required coloring a single branch, whereas tasks 3 to 5 required coloring all objects. The layouts used in the tasks covered two topologies: all clones directly connected to the master object, such as a star arrangement (Figure 6-7b), or a main “stem” with branches
to each node, such as a straight grid (Figure 6-7a,c), a “fishbone” grid (Figure 6-7d,e), and a star without center (Figure 6-7f). In each layout the target colors alternated between red and blue every two columns/spokes. For all layouts except the straight grid used in task 1, there were either thirteen “spokes” or thirteen “branches”. The selected objects or groups had to be colored to match the image on the left. In all tasks, the target images would appear on the left of the screen for reference, while the editable clone topologies would appear on the right.

In the rectangle selection condition, participants were asked not to use <SHIFT> in task 1, but were allowed to add/subtract objects with <SHIFT> in all other tasks, as necessary. For clone editing, participants were only allowed to select a single object at a time and to operate on it. The default visualization, which visualizes the topology only in the clone editing condition, served also as a reminder to participants to use the correct editing technique.

At the end of the experiment, participants were asked to rank selection and clone editing from 0 to 6 for rectangular and non-rectangular arrangements, as well as overall.
Figure 6-7. Task in second study: (a) task 1, size 1, (b) task 2, size 1, (c) task 3 size 1, (d) task 4, size 1, (e) task 4 size 2, (f) task 5, size 1
6.9 Results

The main effects of the editing technique, $F_{1,11} = 9.52, p < .05$, task $F_{4,44} = 59.72, p < .0001$ and size $F_{1,11} = 15.55, p < .005$ were significant. The interaction between technique and task was also significant, $F_{4,44} = 20.12, p < .0001$, see Figure 6-9. The interaction between technique and size was significant, $F_{1,11} = 30.64, p < .0005$, as between task and size, $F_{4,44} = 7.52, p < .0005$. A Tukey-Kramer analysis revealed three groupings of tasks: tasks 1–2 (fastest), task 3 (medium) and tasks 4-5 (slowest). We report all results according to this grouping.

*Figure 6-8. Average task completion times for User Study 2. Error Bars: ±1 SE*
6.9.1 Tasks 1 and 2

In this group, technique was not significant, $F_{1,11} = 3.57, p > .05$. But size, $F_{1,11} = 21.23, p < .001$, and task were, $F_{1,11} = 23.97, p < .0005$. The interaction between technique and size was significant, $F_{1,11} = 31.49, p < .0005$. Clone editing took about the same time for both sizes, but rectangle selection was slower for larger sizes. The
interaction between task and size was also significant, $F_{1,11} = 6.34, p < .05$. Task 2 was slower for larger sizes.

### 6.9.2 Task 3

Here, technique was significant, $F_{1,11} = 23.28, p < .001$. Rectangle selection was much faster (30.04 s vs. 50.98 s). Size was not significant, $F_{1,11} = 0.43, ns$, and there was no significant interaction.

### 6.9.3 Task 4 and 5

For this, technique was significant, $F_{1,11} = 23.36, p < .001$. Clone editing was the fastest (52.74 s vs. 82.78 s). Task was not significant, $F_{1,11} = 0.11, ns$. The interaction between technique and task was significant, $F_{1,11} = 10.42, p < .01$. The differences for task 4 were much larger than for task 5. Size was significant, $F_{1,11} = 12.99, p < .005$. Larger sizes took longer (52.97 s vs. 82.55 s). The interaction between technique and size was significant, $F_{1,11} = 32.57, p < .0005$, with no difference between small tasks with different techniques, but a big difference in larger sizes.

### 6.9.4 Questionnaires

The results from the questionnaires are shown in Figure 6-10. Overall, clone editing was preferred.
6.9.5 Error Rates

We logged the number of undo’s in the use of all editing techniques. For undo’s, neither task ($F_{4,44} = 1.01, p > .01$), technique ($F_{1,11} = 1.19, p > .01$), nor size ($F_{1,11} = 0.60, \text{ns}$) were significant.

6.10 Discussion

Unsurprisingly, both techniques performed equally well for the simple tasks 1 and 2, where only a single “branch” was to be colored – as long as the number of objects was small. However, with larger number of selected objects, rectangle selection became significantly slower. We attribute this to the larger distances that the cursor must cover.
with this technique and to arrangements (such as the branches of the star), where rectangle selection cannot be used effectively.

Tasks 3–5 involved coloring all objects. It was possible to complete this task a bit faster by first coloring all objects one color and then coloring only half the objects differently. About half the participants employed this strategy at least once, while the other half never did it. This seems to indicate the reliance on different strategies in human problem solving. Task 3 was ideally suited for rectangle selection, and it is no surprise that this technique was consistently faster here. However, tasks 4 and 5 also show that as soon as the layout does not match the kind of tasks that rectangle selection supports well, the technique quickly becomes non-competitive. Interestingly, the large “fishbone” grid pattern used in task 4 was particularly challenging for rectangle selection, even more than so than the star pattern in task 5.

In line with the timings, the analysis of the questionnaires showed that participants prefer rectangle selection for rectangular arrangements and clone editing for everything else.
Chapter 7

Conclusion

In this thesis we described our new interface for cloning objects in drawing systems and evaluated it experimentally. The process of refining the interface inspired us to conduct three major user studies and a number of small pilots. Each of these experiments was useful as they revealed the limitations of the interface and inspired redesign to accommodate users better. The experiments also revealed in which situations the new interface is more effective than the existing competing techniques such as smart duplication, tiled cloning and rectangular selection editing.

Chapter 5 described the preliminary user study which aimed to explore the potentials of the new interface. The results indicated that direct cloning is practically almost always faster than tiled cloning. Also, tiled cloning needed more “undo” type of operations (including re-creates) on average which was not surprising. It was revealed that the task which required using clone structures and the calendar inspired editing compromised the external validity and inspired a better controlled user study to reveal significance.

In Chapter 6 we presented the results of two user studies which investigated creation and editing of clones in mouse-based systems. Our new techniques were shown to be either faster or at least competitive for almost all situations.
In summary, we believe that clone editing is an ideal addition to drawing systems, as it works well overall and excels in situations where rectangle selection cannot be used effectively. Seen differently, our results underline the potential for interaction techniques that use the structure of a layout. This is also visible in the fact that rectangle selection performed only well for layouts which were strictly rectangular. Also, it is important to point out that users preferred direct cloning and clone editing overall and in the tasks the techniques were targeted at.

### 7.1 Improving the Surveyed Systems

We believe that the implications of the research presented in this thesis have the potential to improve the applications we surveyed in Chapter 2. Here we briefly review the systems surveyed earlier along with suggestions how these systems can be improved by the use of cloning, see Table 3.

<table>
<thead>
<tr>
<th>Class/Application</th>
<th>Feature</th>
<th>Replication method</th>
<th>No. of replications</th>
<th>Visual appearance</th>
<th>Attributes preserved (during copying)</th>
<th>Editing method and limitations</th>
<th>Link to related copies</th>
<th>Attribute interpolation of copies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Diagram Editors</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Microsoft Word/PowerPoint</td>
<td>Smart duplication</td>
<td>Ctrl-D</td>
<td>Proportional to distance</td>
<td>Same as the original</td>
<td>All</td>
<td>DM or Dialog. Editing a clone affects related clones specified by a pattern</td>
<td>Yes, w/ visual guides in focused editing mode and an option to unlink</td>
<td>Yes</td>
</tr>
</tbody>
</table>

*Table 3 Improving the surveyed systems*
<table>
<thead>
<tr>
<th>Class/ Application</th>
<th>Feature</th>
<th>Replication method</th>
<th>No. of replications</th>
<th>Visual appearance</th>
<th>Attributes preserved (during copying)</th>
<th>Editing method and limitations</th>
<th>Link to related copies</th>
<th>Attribute interpolation of copies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft Visio 2007 (Microsoft, 2009e)</td>
<td>Shape duplication</td>
<td>Dragging to create equally spaced clones w/ alignment and spacing</td>
<td>Proportional to distance</td>
<td>Same as the original</td>
<td>Some attributes such as title and links to other diagram objects don't need to be preserved</td>
<td>In the context of this application editing related copies is not desirable</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Spreadsheets and Web</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Microsoft Excel 2007</td>
<td>Cell filling</td>
<td>Dragging</td>
<td>Defined as created</td>
<td>Data preserved</td>
<td>All except cell comments</td>
<td>DM or Dialog</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>C3W(Fujima, Lunzer et al., 2004b)</td>
<td>Cloning Scenarios</td>
<td>Pop-up menu on a cell to be cloned or smart duplication</td>
<td>One</td>
<td>New instances of cells hold same values as the original</td>
<td>All</td>
<td>DM or Dialog</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Text Editors</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Microsoft Word 2007</td>
<td>Inserting a table</td>
<td>Dragging in on document instead of the panel</td>
<td>Explicitly defined through dialog or defined as created through the panel technique</td>
<td>N/A</td>
<td>N/A</td>
<td>DM or Dialog. Multiple cols, rows or cells can be edited or deleted</td>
<td>Editing a row or a column affects other rows or cols</td>
<td>No</td>
</tr>
<tr>
<td>Microsoft Word 2007</td>
<td>Word Art 2007</td>
<td>Typing the characters directly into the document and getting interactive feedback, instead of the existing dialog method where the</td>
<td>Defined as typed</td>
<td>Randomized shape, color and texture can be changed</td>
<td>Color, texture font preserved for each character</td>
<td>Dialog</td>
<td>Yes, editing characters keeps the same style</td>
<td>No</td>
</tr>
<tr>
<td>Class/Application</td>
<td>Feature</td>
<td>Replication method</td>
<td>No. of replications</td>
<td>Visual appearance</td>
<td>Attributes preserved (during copying)</td>
<td>Editing method and limitations</td>
<td>Link to related copies</td>
<td>Attribute interpolation of copies</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------</td>
<td>--------------------</td>
<td>--------------------</td>
<td>-------------------</td>
<td>---------------------------------------</td>
<td>-------------------------------</td>
<td>----------------------</td>
<td>-------------------------------</td>
</tr>
<tr>
<td>Calendars</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mozilla Sunbird</td>
<td>Creation of repeated events</td>
<td>Dialog</td>
<td>Explicitly defined</td>
<td>Same as the original</td>
<td>All</td>
<td>Dragging &amp; Dialog w/o limitations.</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Google Calendar</td>
<td>Creation of repeated events</td>
<td>Dialog</td>
<td>Explicitly defined</td>
<td>Same as the original</td>
<td>All</td>
<td>Dragging &amp; Dialog w/o split scenarios unless explicitly asked</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Microsoft Windows Calendar</td>
<td>Creation of repeated events</td>
<td>Dialog</td>
<td>Explicitly defined</td>
<td>Same as the original</td>
<td>All</td>
<td>Dragging &amp; Dialog w/o limitations for dragging and changing the title through the first event</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Apple iCal</td>
<td>Creation of repeated events</td>
<td>Dialog</td>
<td>Explicitly defined</td>
<td>Same as the original</td>
<td>All</td>
<td>Dragging &amp; Dialog, w/ all event modifiable (past and future)</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>2D Graphics Editors</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Xfig</td>
<td>Copy</td>
<td>Dragging a corner of a shape</td>
<td>Proportional to distance</td>
<td>Same as the original</td>
<td>All</td>
<td>DM</td>
<td>Yes, w/ visual guides in focused editing mode and the option to unlink</td>
<td>No</td>
</tr>
<tr>
<td>Class/Application</td>
<td>Feature</td>
<td>Replication method</td>
<td>No. of replications</td>
<td>Visual appearance</td>
<td>Attributes preserved (during copying)</td>
<td>Editing method and limitations</td>
<td>Link to related copies</td>
<td>Attribute interpolation of copies</td>
</tr>
<tr>
<td>--------------------</td>
<td>---------</td>
<td>-------------------</td>
<td>---------------------</td>
<td>------------------</td>
<td>----------------------------------------</td>
<td>--------------------------------</td>
<td>--------------------------</td>
<td>---------------------------------</td>
</tr>
<tr>
<td>Skencil-C</td>
<td>Direct Cloning and Editing</td>
<td>Dragging while holding a modifier key with the option of free and aligned cloning</td>
<td>Proportional to distance</td>
<td>Same as the original</td>
<td>All</td>
<td>DM or Dialog. Editing a clone affects related clones specified by a pattern</td>
<td>Yes, w/ visual guides in focused editing mode and the option to unlink</td>
<td>Yes, color, height and width</td>
</tr>
<tr>
<td>InkScape</td>
<td>Simple cloning to become direct cloning</td>
<td>Dragging while holding a modifier key with the option of free and aligned cloning</td>
<td>Proportional to distance</td>
<td>Same as the original</td>
<td>All</td>
<td>DM or Dialog. Editing a clone affects related clones specified by a pattern</td>
<td>Yes, if editing any clone not just the original w/ visual guides in focused editing mode and the option to unlink</td>
<td>Yes</td>
</tr>
<tr>
<td>InkScape</td>
<td>Tile cloning</td>
<td>Dialog</td>
<td>Explicitly defined</td>
<td>Same as original</td>
<td>All</td>
<td>DM or Dialog. Editing a clone affects related clones specified by a pattern</td>
<td>Yes, if editing any clone not just the original w/ visual guides in focused editing mode and the option to unlink</td>
<td>Yes</td>
</tr>
<tr>
<td>InkScape</td>
<td>Pattern along path</td>
<td>Dragging in addition to dialog</td>
<td>Proportional to distance and size</td>
<td>Same as original</td>
<td>All</td>
<td>DM or Dialog. Editing a clone affects related clones specified by a pattern</td>
<td>DM or Dialog. Editing a clone affects related clones specified by a pattern</td>
<td>Yes</td>
</tr>
<tr>
<td>Class/Application</td>
<td>Feature</td>
<td>Replication method</td>
<td>No. of replications</td>
<td>Visual appearance</td>
<td>Attributes preserved (during copying)</td>
<td>Editing method and limitations</td>
<td>Link to related copies</td>
<td>Attribute interpolation of copies</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------</td>
<td>-------------------</td>
<td>---------------------</td>
<td>------------------</td>
<td>---------------------------------------</td>
<td>--------------------------------</td>
<td>--------------------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>Adobe Photoshop CS4 (Adobe, 2008)</td>
<td>Copying and moving a selection</td>
<td>Dragging a selection using the move tool while holding a modifier key (Alt)</td>
<td>Proportional to distance if activated</td>
<td>Same as the original</td>
<td>All</td>
<td>DM or Dialog. Editing a clone affects related clones specified by a pattern</td>
<td>DM or Dialog. Editing a clone affects related clones specified by a pattern</td>
<td>Yes, size and shape of selection</td>
</tr>
<tr>
<td>Adobe Photoshop CS4</td>
<td>Texture tiling</td>
<td>Dialog</td>
<td>Proportional to size</td>
<td>Same as the original</td>
<td>All</td>
<td>Dialog</td>
<td>No</td>
<td>Yes, size color, alpha channel</td>
</tr>
<tr>
<td>Adobe Photoshop CS4</td>
<td>Clone Stamp (replication of an image area with brush)</td>
<td>DM</td>
<td>One</td>
<td>Depending on brush stroke</td>
<td>All</td>
<td>DM</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>3D Graphics Editors</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AutoCad</td>
<td>Array (2D formation of copies in rectangular or circular order)</td>
<td>Dragging for rectangular copying Drag-sweeping for circular copying.</td>
<td>Proportional to distance or angle</td>
<td>Same as the original</td>
<td>All</td>
<td>DM or Dialog. Editing a clone affects related clones specified by a pattern</td>
<td>Yes, w/ visual guides in focused editing mode and the option to unlink</td>
<td>Yes, size, distance, angle, radius</td>
</tr>
<tr>
<td>Sesame (Oh, Stuerzlinger et al., 2006)</td>
<td>Copying 2D shapes</td>
<td>Dragging</td>
<td>Proportional to distance</td>
<td>Same as the original</td>
<td>All</td>
<td>DM or Dialog. Editing a clone affects related clones specified by a pattern</td>
<td>Yes, w/ visual guides in focused editing mode and the option to unlink</td>
<td>Yes, size</td>
</tr>
<tr>
<td>Class/ Application</td>
<td>Feature</td>
<td>Replication method</td>
<td>No. of replications</td>
<td>Visual appearance</td>
<td>Attributes preserved (during copying)</td>
<td>Editing method and limitations</td>
<td>Link to related copies</td>
<td>Attribute interpolation of copies</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------</td>
<td>-------------------</td>
<td>---------------------</td>
<td>-------------------</td>
<td>--------------------------------------</td>
<td>-------------------------------</td>
<td>-------------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>3D StudioMax</td>
<td>Cloning</td>
<td>Dialog can be replaced with dragging where all clones are created as references by default. The type of clone can be changed to independent copy or instance during editing</td>
<td>Proportional to distance</td>
<td>Same as the original</td>
<td>All</td>
<td>DM or Dialog. Editing a clone affects related clones specified by a pattern, in addition to the existing copy/reference/instance interface</td>
<td>Yes, w/ visual guides in focused editing mode and the option to unlink</td>
<td>Yes, height, width, length, color, etc.</td>
</tr>
<tr>
<td>MIVE (Smith, Salzman et al., 2001)</td>
<td>Space filling</td>
<td>Dragging</td>
<td>Proportional to distance</td>
<td>Randomized</td>
<td>N/A</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>PORT-C, HOMER-C, CbyE, SM, D-Slider, Keypad (Chen and Bowman, 2006)</td>
<td>Cloning</td>
<td>DM: Joystick, selection box and state transition menus</td>
<td>Explicitly defined</td>
<td>Same as the original</td>
<td>All</td>
<td>If the cloned objects were linked after the apply button is pressed, then it would be possible to edit the clones after they were created.</td>
<td>Ghost copies are displayed until apply button is pressed</td>
<td>Yes</td>
</tr>
</tbody>
</table>

### 7.2 Limitations of the Presented Techniques

We want to emphasize at this point that our design targets quick and easy object cloning and editing for *common* tasks. In other words, we did not target a user interface that can achieve every possible replication or editing task. One option for that is to extend the dialog-based interface, and another to implement an “animation” interface that instantiates objects based on a start and end position, together with a interpolation method. However, these designs involve at least one more step than direct cloning, and hence are unlikely to
be faster. Moreover, if one targets non-linear interpolation, this necessitates a much more complex user interface, too. In our experience, this also requires users that understand animation concepts. An example of this kind of interpolation can be found in Adobe Flash CS4 (Girole, 2008; Haase, 2009).

Although our linear interpolation technique proved superior for the chosen tasks in the preliminary study, it is still inferior in general in its current form, since interpolation in tiled cloning is more versatile. For example, using tiled cloning, one can interpolate along both rows and columns in grids, which is not currently possible using our method.

Also, smart duplication allows creation of objects along an arbitrary path with high precision if snapping is used. However, our cloning technique currently supports precision cloning only along either a vertical path or a horizontal path.

### 7.3 Future Work

We presented a new direct cloning interface that is superior to tiled cloning. The method is superior to smart duplication for grid creations and competitive for row creations with non-preset spacing. The clone editing interface is effective in situations where traditional editing methods are problematic to use. Hence, we see our technique as an ideal complement to existing techniques for mouse-based drawing systems.

During the evaluation we disabled creation of overlapping clones, but we are planning to investigate this in the future. We plan to look at creation along other normalized paths such as arcs. We also plan to look at user interfaces that allow the user
to first create a path, and then to specify the replication of clones along that path, similar to facilities for laying out text along a path. However, any such user interface needs to support editing the path post-hoc, which makes this more difficult. We are also planning to investigate the efficiency of the presented pop-up editing interface. One of the ideas is to add another option to “edit all preceding” clones and investigate whether users would use it. A comparison with lasso selection in pen-based systems is also something we are considering. Last, but not least, we plan to investigate if some of our new techniques can be applied to calendar systems to make editing of repeated events simpler.
Bibliography


