MITSUBISHI ELECTRIC RESEARCH LABORATORIES
http://www.merl.com

The Magix Series of Playful Learning Environments

Edith Ackermann, Carol Strohecker, Aseem Agarwala

TR97-24 December 1997

Abstract

In the Magix series of software learning environments, children can learn about topics in math and science through playful exploration. They experiment with patterns and dynamic systems through a constructive-dialogic style of interaction. In these colorful environments, children make and transform virtual objects and then study effects that emerge through their replication and variability. The learning environments are best situated on a portable computational device. Following a brief discussion of design rationale, this paper describes details of the environments through a series of appendices. Appendix 1 explains the constructive-dialogic style of interaction and its importance to Magix explorations. Appendix 2 describes attributes of the child-sized, portable platform that would be ideal for Magix environments. The bulk of our discussion is in Appendix 3, which is the functional description of one entry in the Magix series, PatternMagix. Appendix 4 explains recommended modifications to PatternMagix based on preliminary observations of its use by a few children. Appendix 5 outlines a second entry, AnimMagix. Appendix 6 describes initial component and object hierarchies for the implementations of the software prototypes.

MERL - A MITSUBISHI ELECTRIC RESEARCH LABORATORY

The *Magix* Series of Playful Learning Environments

Edith Ackermann
Carol Strohecker
Aseem Agarwala

TR97-24 December 1997

Abstract

In the *Magix* series of software learning environments, children can learn about topics in math and science through playful exploration. They experiment with patterns and dynamic systems through a constructive-dialogic style of interaction. In these colorful environments, children make and transform virtual objects and then study effects that emerge through their replication and variability. The learning environments are best situated on a portable computational device.

Following a brief discussion of design rationale, this paper describes details of the environments through a series of appendices. Appendix 1 explains the constructive-dialogic style of interaction and its importance to *Magix* explorations. Appendix 2 describes attributes of the child-sized, portable platform that would be ideal for *Magix* environments. The bulk of our discussion is in Appendix 3, which is the functional description of one entry in the *Magix* series, PatternMagix. Appendix 4 explains recommended modifications to PatternMagix based on preliminary observations of its use by a few children. Appendix 5 outlines a second entry, AnimMagix. Appendix 6 describes initial component and object hierarchies for the implementations of the software prototypes.

The *Magix* series of playful learning environments in which children experiment with dynamic systems by constructing and transforming virtual objects through dialogic interaction with a portable computational device

We present design principles and prototypical instantiations of a series of game-like learning environments that expand the existing repertoire of educational software. The environments support learning through playful exploration. Contrasting the instructional character of most software for learning, these environments foster a constructive-dialogic style of interaction.

Current educational software assumes that the system should guide the child: the software becomes an electronic teacher or tutor. Our approach emphasizes learning moreso than teaching. The model is one of experimentation, distributed control, and conversational exchange rather than sequential curriculum and unilateral control. The child shares control with the system, through dialog rather than conquest – a dialog that the child initiates.

Although the system does not impose a prescriptive sequence of activities or topics, it responds to the child's interventions with specific, consistent, context-sensitive functionality.

As today's children become adults in the 21st century, they will need increasingly to understand that the world operates not according to simple sets of rules or facts, but as a network of interdependent, multivariate systems. Both biological and physical phenomena can be understood in these terms: population growth, weather patterns, economic fluctuations, biological evolution, organizational behavior, traffic patterns, and countless others.

Today's computational tools enable better study and understanding of such complex phenomena, including the very phenomenon of human learning. Until now, psychological research has focused on limited-variable models of learning. Educational programs based on rigid instructional design have thrived as a result of this reductionist approach. We urgently need environments in which children can develop methods of inquiry and intuitions about multivariate, dynamic systems. Contemporary computational media are ideally suited for such pursuits.

Young children often have good intuitions about processes of balance, equilibrium, self-regulation and

other properties of dynamic systems. We develop a series of environments in which children can experiment with patterns and other emergent effects of such systems. These environments form an important basis for childrens' future study of math and science.

We start with two prototypical environments, PatternMagix and AnimMagix. In PatternMagix, children play in a world of colorful tiles and geometric operations, from which they forge mosaic-like patterns. In AnimMagix, children create whimsical creatures with anthropomorphic behaviors and launch them onto a field in which the creatures interact and affect one another.

The mode of work is as important as the topics in this development. Children construct objects, patterns, and system states and then negotiate changes in them. In this way they can see how changing one factor infuences the entire system.

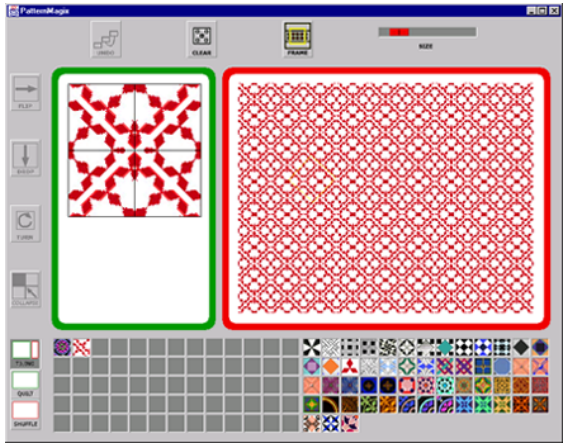
By constructing and transforming objects, children learn about the domains of social dynamics and geometric symmetry. Equally important, children

develop intuitions about the common underlying theme of dynamic systems. They also develop an ecological perspective on scientific practice, through sharing control and observing the balance of influences. The system does not impose a prescriptive sequence of activities or topics. Instead it allows the learner to initiate a dialog and responds by generating the unpredictable emerging effects and providing suggestions for further experimentation. Thus the constructive-dialogic nature of the environment offers potential for a unique contribution to existing educational software and edutainment.

We describe this style of interaction and its importance to *Magix* explorations in Appendix 1. Consistent with the emphasis on the child's high degrees of intellectual and emotional engagement, and the importance of these qualities for learning, we propose a platform that is both child-sized and portable. Attributes of this platform are detailed in Appendix 2. Appendix 3 is the functional description of one entry in the *Magix* series, PatternMagix. Appendix 4 explains recommended modifications to PatternMagix based on preliminary observations of its use by a few children. Appendix 5 outlines a second entry in the series, AnimMagix. Appendix 6 describes initial component and object hierarchies for implementations of the software prototypes.

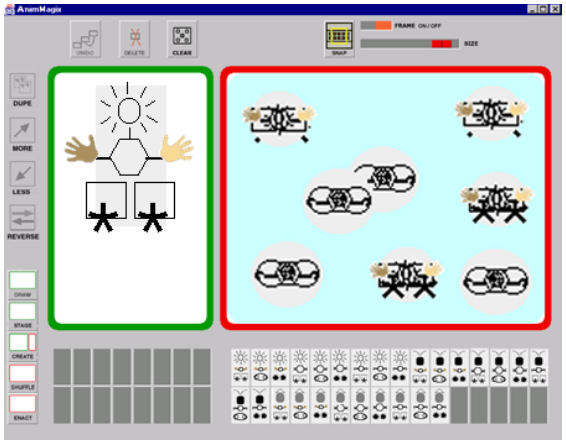
Appendix 1: The *Magix* Series

We are creating prototypes of software environments in which children can learn about math and science through playful exploration. In the *Magix* environments, youngsters experiment with patterns and dynamic systems through a constructive-dialogic style of interaction. The children make and transform virtual objects, and then study effects that emerge through their replication and variability. The learning environments are best situated on a portable computational device.



PatternMagix

Learning by doing; Learning by playing



AnimMagix

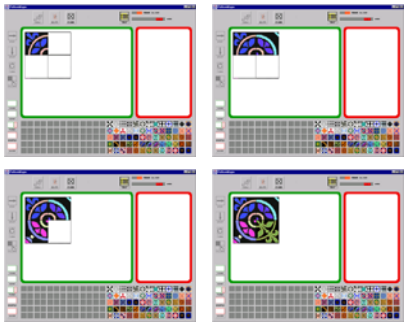
As today's children become adults in the 21st century, they will need increasingly to understand that the world operates as a network of interdependent, multivariate systems. Both biological and physical phenomena can be understood in these terms: for example, population growth, weather and traffic patterns, economic fluctuations, biological evolution, and organizational behavior. Young children have good intuitions about properties of dynamic systems, including properties such as balance, equilibrium, and self-regulation. In the *Magix* environments, children can develop these intuitions by experimenting with complex systems and contemplating their emergent effects. Thus the environments offer foundations for later study of math and science. Two of the environments, PatternMagix and AnimMagix, exemplify the approach. In PatternMagix, children play in a world of colorful tiles and geometric operations, from which they forge mosaic-like patterns. In AnimMagix, children create whimsical, anthropomorphic creatures and then launch them onto a field in which the creatures interact, affecting one another's behaviors. We may develop an EcoMagix for experimenting with environmental dynamics, and a MediMagix for exploring interplay of physiological systems.

PatternMagix Interactions

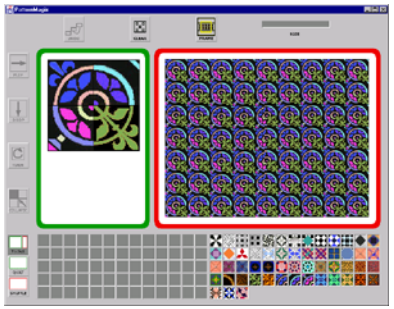
Interactions are modeled as a dialog between child and system.

As the child experiments by making tiles and spreading them into patterns, a kind of dance results. The Child Area (left) and System Area (right) expand and contract according to who is most active at a given moment. This alternation facilitates the *constructive-dialogic style of interaction.*

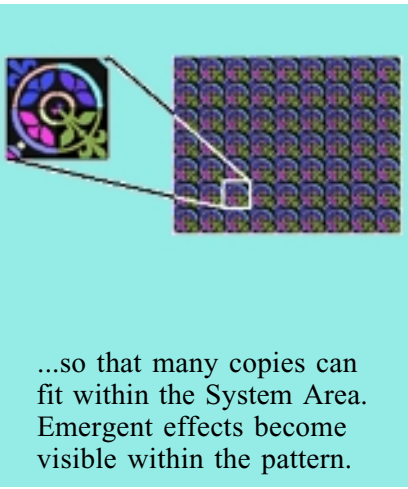
The child can use a few modes, functions, and geometric transformations to create colorful elements and intriguing patterns.



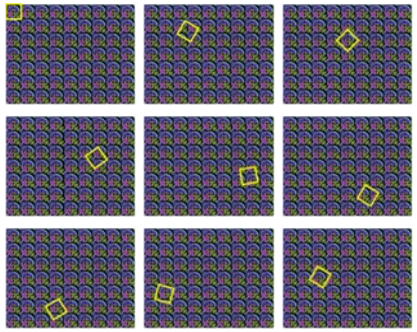
For example,
the child builds a tile...



...and the system takes its turn by adjusting the tile's size...



...so that many copies can fit within the System Area. Emergent effects become visible within the pattern.



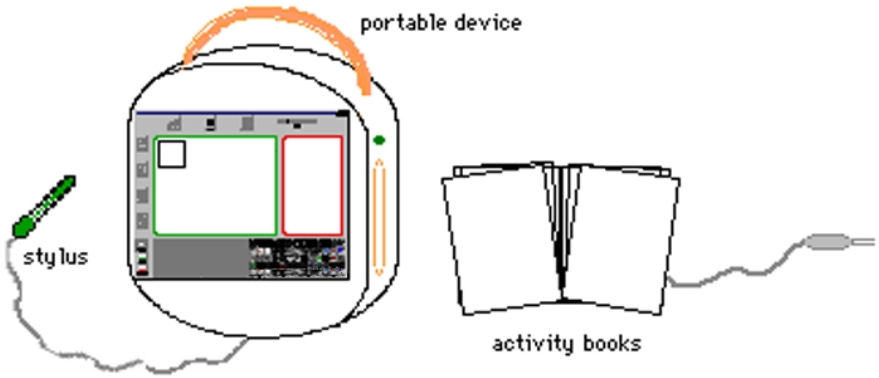
The system then "suggests" possibilities for new tiles. The child can manipulate the floating frame's position and select an area as the basis for a new tile.

Different modes enable varying degrees of control as the child "converses" with the system to create objects and patterns.

Appendix 2: A Child-sized, Portable Computational Device

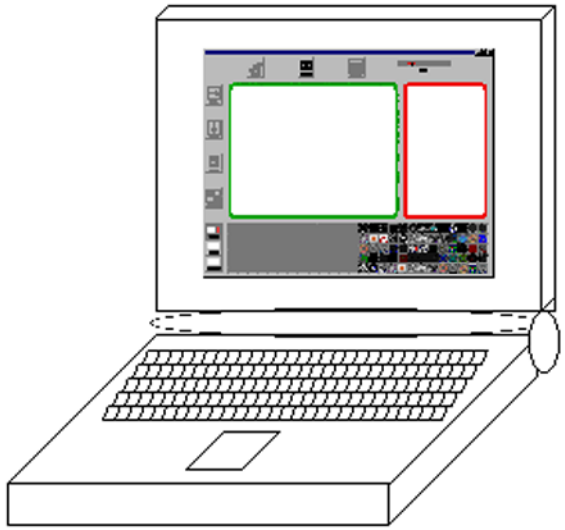
Children use workspaces differently from adults. They need to change position often – squirming, sitting, standing, and jumping complement and express children's thinking and it's important to support rather than inhibit them.

Children like to have their own objects to play with. These objects need to be lightweight, colorful, durable, and above all portable. With such a carefully designed platform for the Magix series, children can play quietly at home or carry it with them as they ride in a car-seat or show it to friends on an outdoor playground (etc.).



Thus, at best, the Magix series is supported by a specifically designed, self-contained device that the child can carry and which presents its functionality playfully and colorfully. This device has an attached stylus for inputs from the child and is augmented by complementary materials in other media (e.g., books, kaleidoscope, creature figures, etc.).

Alternatively, the series could be supported by a more traditional implementation on a standard laptop computer, making use of its associated pointing device (mouse, trackball, thumbpad, or etc.)

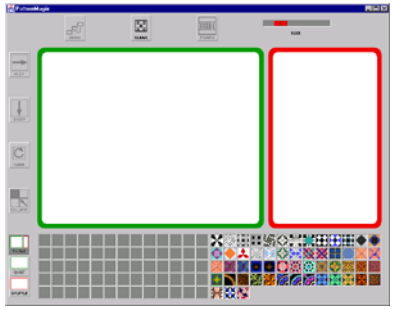


Appendix 3: *PatternMagix* Functional Description

The learning environment presents four distinct conceptual and physical areas.

Child's **Area** for creating playthings

Buttons for modes of work, general functions, and specific operations



System's **Area** for automatic transformations

Library with ready-made entries and empty slots for child's creations

General functions include
Undo, Delete, Clear, and Frame.

UNDO: The system returns to the previous state.

DELETE: An individual unit is removed.

CLEAR: Everything from the active area (child's or System's) is removed.

FRAME: A yellow box that delineates an area to be defined as a new tile.
ON / OFF: Toggles the Frame off and on.
SIZE: Adjusts the size of the Frame.

In the most recent design, there are five working modes:
Draw, Quilt, Tiling, Shuffle, and Kaleid.

DRAW: The child designs a tile by hand.

QUILT: The child arranges tiles into patterns by hand.

TILING: The child and system alternate as the child builds tiles and the system spreads them into patterns and suggests selections for new tiles.

SHUFFLE: The system introduces variations in a pattern by repeatedly applying operations of geometric symmetry.

KALEID: The system introduces variations in a pattern by applying kaleidoscopic transformations.

These modes range from maximal degree of control for the child to maximal degree of control for the system.

Specific operations include
Flip, Drop, Turn, and Collapse.

FLIP: Transforms a tile around the Y-axis.

DROP: Transforms a tile around the X-axis.

TURN: Rotates the tile in 90-degree increments.

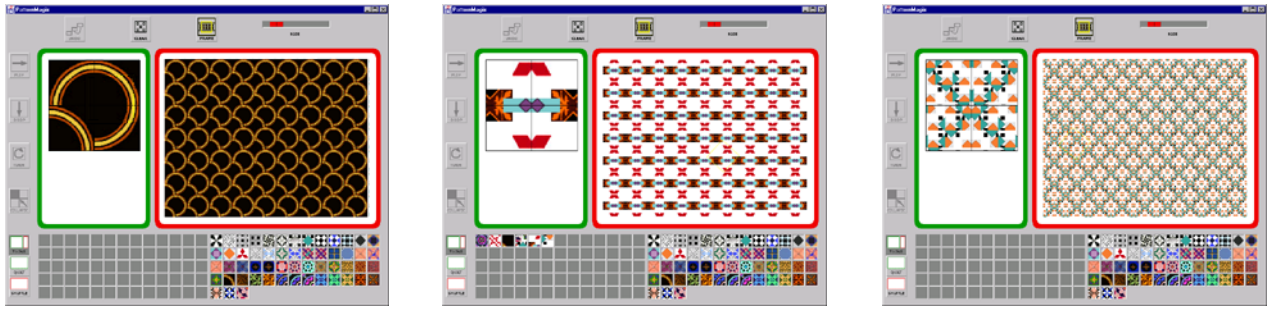
COLLAPSE: Tiles build as quadrants stacking left-to-right and top-to-bottom. Collapse shrinks them to a single quadrant, then defined as a new tile.

Interaction Scenarios

Interactions are modeled as a dialog between child and system

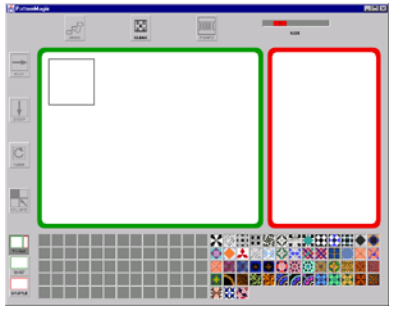
As the child changes modes or experiments by making tiles and spreading them into patterns, a kind. of dance results. The Child and System Areas expand and contract according to who is most active at a given moment.

Using the constructive-dialogic style of interaction in a few modes with basic functions and geometric transformations, the child can create colorful elements and intriguing patterns.

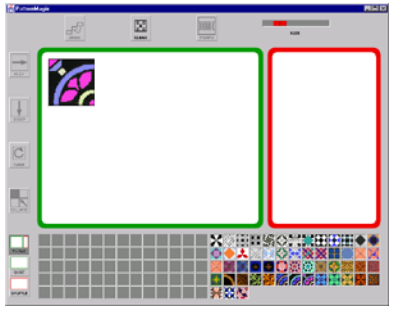


TILING Mode

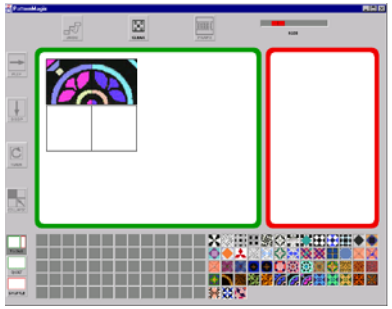
In Tiling mode, exchanges between the Child Area and the System Area occur most frequently, exemplifying the constructive-dialogic style of interaction.



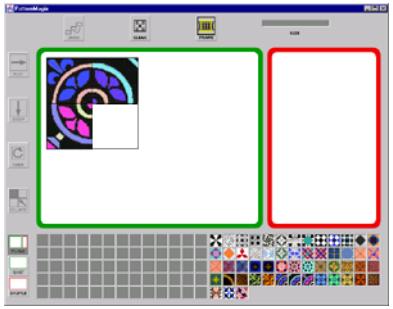
When PatternMagix opens, it greets the child with an invitation to construct a tile.



The child drags one of the ready-made tiles from the Library and uses it as a unit in a new construction.

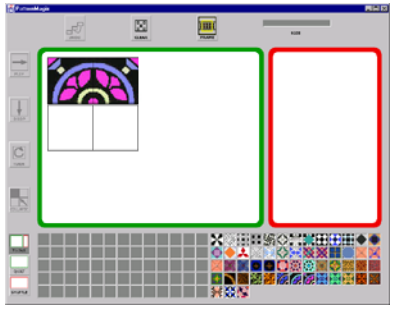


In Tiling mode, the child can drag other tiles from the Library into the working area....

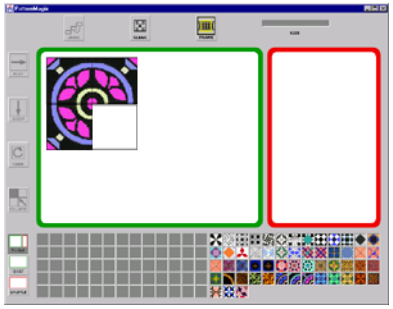


The system places each new tile within a grid structure. Additional tiles are placed left-to-right and top-to-bottom.

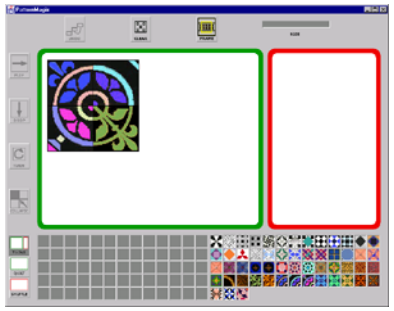
The child can continue the construction by changing to the Draw mode to create a tile from scratch (see below), or by staying in Tiling mode.



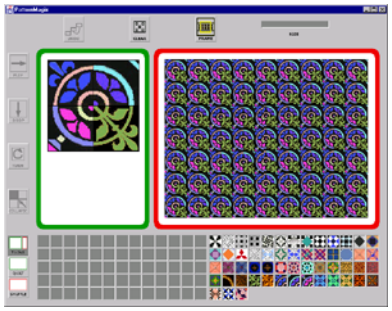
...or transform the same tile by applying operations of geometric symmetry.



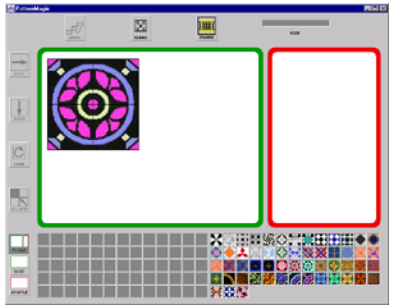
Flip reflects the tile around the y-axis. *Drop* reflects it around the x-axis. (A *Turn* is a 90-degree rotation to the right.)

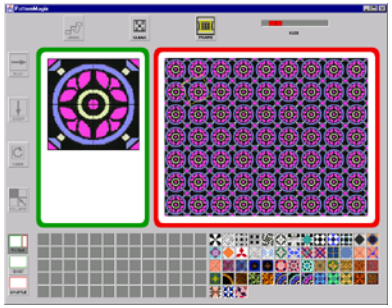


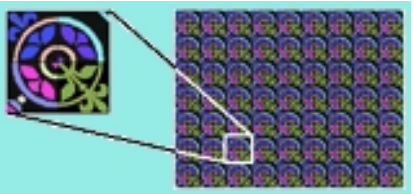
When the quadrant-grid is filled, the child can click on the System Area to continue the dialog.



The system interprets the contents of the grid as a new tile. It then replicates the tile and arranges it into a pattern.



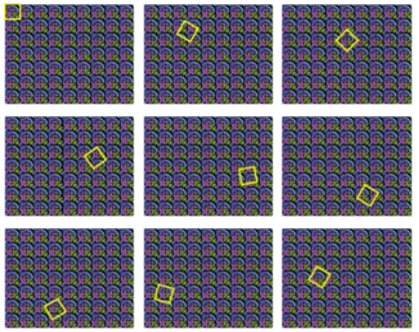




The system automatically adjusts the size of the new tile so that many copies can fit and emergent effects become visible within the pattern.

The system suggests possibilities for new tiles by displaying a highlighted Frame around the tile at the upper left of the pattern.

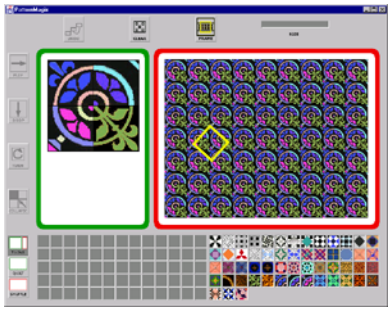
The Frame lingers momentarily and then begins to float randomly around the pattern. The Frame moves slowly and changes orientation as it goes.



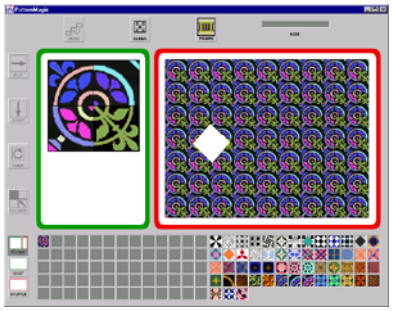
The child can stop the movement by clicking directly on the Frame, freezing its orientation. She can then reposition the Frame by hand.

The child can adjust the Size of the Frame or use the On / Off toggle to turn the Frame off, making it disappear.

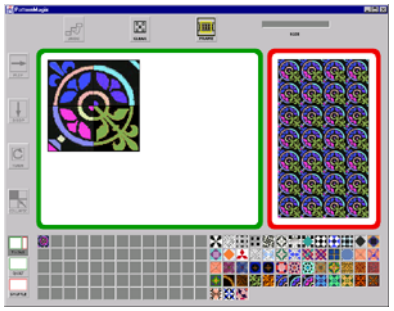
If the child turns the Frame back on, it reappears and resumes its free-floating movement.



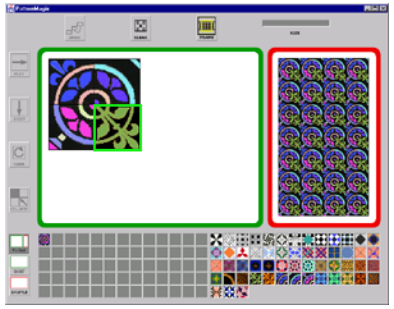
When the Frame satisfactorily delineates a new area of the pattern, the child can click on the Snap button.



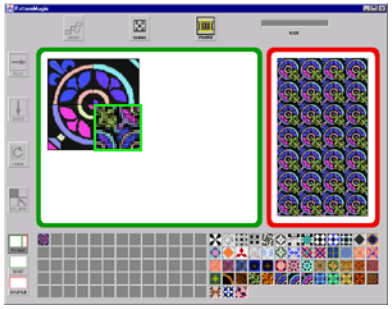
The system interprets a new tile by grabbing the framed part of the pattern, rotating it back to 0 degrees, and saving it to the child's section of the Library.



Now the child can click on the Child Area. The screen areas adjust size accordingly and the child can begin a new dialog.

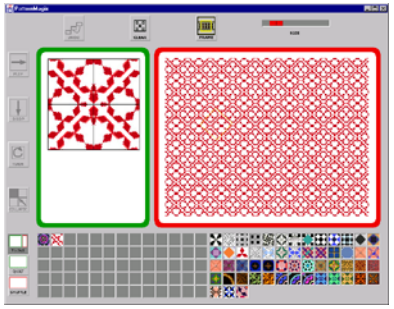


For example, the child can incorporate the Library's new element by clicking on one of the quadrants...



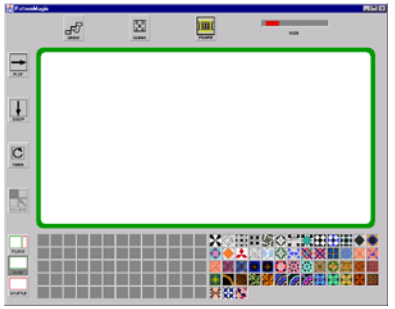
...and then clicking on the Library element. It appears in the highlighted quadrant.

The child can continue working with this new tile or its individual quadrants, or clear the area altogether.



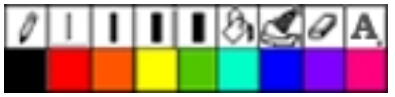
With this dialogic style of interaction and three simple geometric transformations, the child can create many intriguing patterns.

Manual Modes



These modes maximize the child's constructive capability. The Child Area expands to its maximum width, and the child's creations result solely from direct manipulation. Thus the "conversational" style is more monologic and the child has maximal control.

DRAW Mode



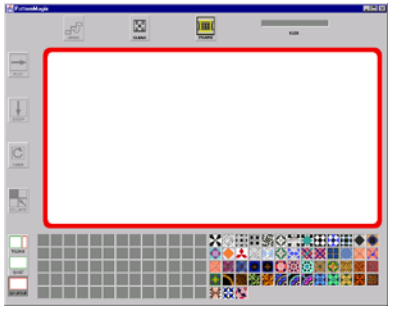
In Draw mode, the child uses a simple palette of tools to create freehand decorations for tiles. Newly made tiles are saved to the child's part of the Library.

QUILT Mode

In Quilt mode, tiles dragged from the Library become "patches" in a freeform "quilt". The child can use the Frame to bound new areas across patches. These unique selections are saved as new tiles in the child's part of the Library and are available for use in other modes.

Automatic Modes

These modes maximize the system's contribution. The System Area expands to its maximum width, and the system automatically generates variations of geometric patterns. The child relinquishes control temporarily but has an opportunity to contemplate the evolving transformations.

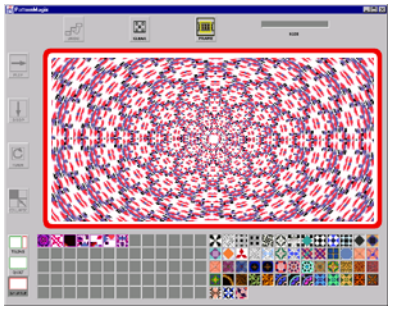


SHUFFLE Mode

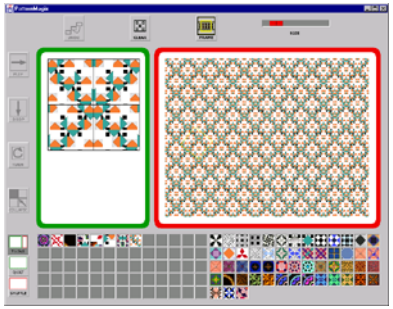
In Shuffle mode, the system cycles though a series of transformations, applied repeatedly to generate a dynamic pattern. Varying sequences of the simple operations of geometric symmetry – Flip, Drop, and Turn – create intriguing effects that inspire new creations when the child is in more constructive modes.

KALEID Mode

In Kaleid mode, still to be developed, potentials for greater complexity are served by variations of basic tile shapes and an expanded repertoire of geometric operations.

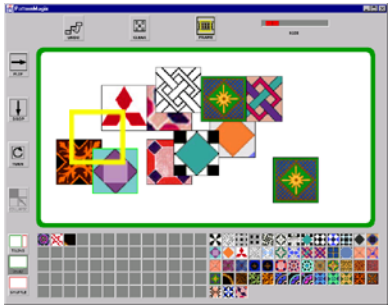


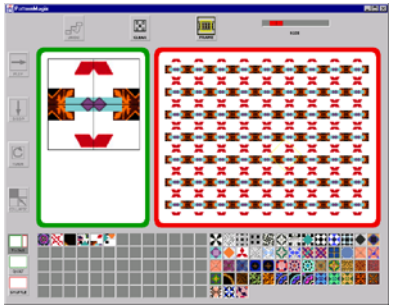
Moving from Mode to Mode



Each mode has its own way of generating new tiles. As the child explores transformations and resulting patterns, she can keep a trace of her work by framing a selection within a pattern. The selection is automatically saved in the child's part of the Library and is then available for use in other modes.

As the child becomes more and more familiar with PatternMagix, switching from mode to mode to create colorful tiles and patterns augments the constructive-dialogic style of interaction.





Appendix 4: Usage Observations and Modifications to *PatternMagix*

Observations of children's sessions with PatternMagix help us understand how individuals relate to and appropriate the tool. We can also learn something about the environment's "holding power," or ability to sustain children's interest over time. Here we describe lessons gleaned from brief sessions with two eleven-year-old children. Although these observations are extremely preliminary, the children have already given us new ways of thinking about the tool. In order to achieve a more complete understanding, we hope to work with many children over long periods of time. In this initial review, we infer some of the lessons from our own observations and others from statements by our young "consultants."

Vignettes

Different children work in different ways, of course. Susan and Brian tended to settle in a preferred mode, or home base, in which they worked for longer times, from which they departed occasionally, and to which they always returned. In one case this strategy had implications for a modification to the environment; in another case the existing design worked well.

Example: Susan's home base was the Quilt mode. She took pleasure in manually composing large and beautifully textured quilts. She spent long times filling the Quilt window and could certainly have made use of a bigger area. She wanted to be able to change the background from white to a color of her own choosing. She complained about the difficulty of adjusting and centering patches. Once she completed a quilt, Susan used the Frame to select a favorite area. She then wanted to use this particular selection as a "composite tile" to be spread into a pattern.

Example: Brian, in contrast, liked the Shuffle mode. He took pleasure in contemplating what the computer "does to a pattern." Building and spreading a tile in the Tiling mode seemed just a means toward "feeding" the Shuffle. He did this combined construction and mode-switching with great ease. He also wanted to print his Tiling pattern.

Depending on which mode and related working area the children preferred, specific goals emerged. According to their own particular interests, the children requested different modifications to the environment.

Example: In Quilt mode, some of Susan's "composite tiles" took the form of patches assembled around a central element, creating 3x3 composites that resembled "Tic-Tac-Toe" grids. This pattern of construction became a signature of her interactions. Several times she bounded such a 3x3 composite, scaling the Frame as necessary, with hopes of spreading "that very selection" into a pattern.

Example: Brian's contemplation of Shuffle patterns led him to make guesses of which geometric transformations the computer was using at a given time.

In designing PatternMagix, we worked to resolve the many conditions created by interworkings of the multiple Frame functions. In working with Susan and Brian we were somewhat surprised to see that those functions they invoked presented no problems in usage. These children did not yet employ the full range of possibilities in working with the Frame, but easily handled scaling and snapping.

Example: Susan immediately understood the floating Frame to be suggesting areas of a spread pattern that could be cut and re-used in a new design. She spontaneously referred to scissors and the idea of snipping a portion of a pattern. She followed the placement of the snapped portion as it entered the library to become available for subsequent use, and she took pleasure in observing the color inversion as the tile entered the library.

Brian switched easily between Tiling and Shuffle modes, and within the Tiling mode switched easily between the Child Area and the System Area. Susan was less interested in the back-and-forth between working areas. She did want the functionality that switching between Quilt and Tiling modes could give her, but worried about the inevitable scale change to her Quilt construction.

Example: Susan carefully assembled her 3x3 composites and wanted to see them spread into patterns. She understood that Framing and Snapping the construction was a way to carry it into Tiling mode, but she wanted to see a pattern at the same scale as her composite tile.

Lessons and Trade-offs

Because Susan focused mostly on Quilt mode and Brian focused on Shuffle, this discussion pertains mainly to those modes.

Depending on how a group of patches is spaced in quilt mode, it may not be easy to place a patch directly over the group's center. The implicit grid helps in manipulating and placing the tiles, but inhibits the degree of control that may be desired. Omitting the grid altogether would introduce new problems; however we could make the grid more resolute or introduce an on/off, "snap-to" function that would allow individuals to work with it in preferred ways. In any case, the grid unit measure should be a multiple of the base tile measure.

Likewise, the base tile measure should be a multiple of the window size for the working areas in every mode. This should mean that pattern-constructions can go all the way to the edge of a window.

The current tile size for Quilt mode is good. We should strive to keep these tiles as large as possible in order to maintain optimal pixel resolution. However, may want to introduce a scaling function so that children can shrink a composite unit in order to develop a pattern from it. This "zoom" function may have roles in other modes as well.

If we carry through with our own dialog metaphor and existing design decisions, we can answer Susan's desire to spread her 3x3 composites into patterns. One aspect of this capability is already resolved by the Framing and Snapping technique. However, the tile that then gets saved in the library is very small when it enters Tiling mode. This tile may be useful in a conventional quadrant-based construction, but children who want to see a pattern at maximum resolution may be frustrated by the scale change and loss of resolution. These children should have the sense that their work is being preserved, not disrupted.

We have already solved a similar issue by carrying a pattern from Tiling mode into Shuffle. This trace between modes supports our claim that "together with the dance between working areas, mode-switching enriches the conversation

between child and system." We could use the same method of leaving a trace between Quilt and Tiling modes. If Susan made her composite unit in Quilt mode and then switched to Tiling mode, her unit should be carried into the Child Area of Tiling mode. We may need to deal with fallout from this conditional substitution for the quadrant grid; however, the principle of carry-over is strongly justified. It provides a sense of resilience that can be reassuring in use of the tool.

Another aspect of this principle could address Susan's reluctance to move back and forth between the Child and System Areas in Tiling mode. Whenever a construction is completed in the Child Area, the System Area could blink or otherwise indicate its readiness to take its turn in the dialog.

There are also other moments when it would be helpful (and consistent with the dialogic principle) for the System could make its presence known. For example, as the sequence of geometric transformations plays out in Shuffle mode, the relevant operation's button could flash. We might also consider allowing the child to control ways in which the computer modifies a Shuffle pattern. Such a capability could give us insights into how to develop the Kaleid mode. However, adding this sort of control in Shuffle mode would raise inconsistencies in our formulations of manual vs. automatic modes and should be considered carefully. A reformulation that allows for both Child and System working areas in each mode might be added.

Recommendations

Here are a few recommendations for improving the PatternMagix. We may implement the short-term recommendations in the current prototype; the long-term recommendations would need to be considered in more depth if we worked on a later version.

Short-term recommendations

In Quilt mode, the implicit grid that guides placement of patches should be at a finer resolution and set at a multiple of the base tile measure. Likewise, the base tile measure should be a multiple of the window size for the working areas in every mode. This should mean that pattern-constructions can go all the way to the edge of a window.

We should carry over a composite unit made in Quilt mode into the Child Area of Tiling mode.

Whenever a construction is completed in the Child Area, the System Area should blink to indicate its readiness to take its turn in the dialog.

As a sequence of geometric transformations plays out in Shuffle mode, the relevant operation's button should flash.

Implement "dupe" (duplicate) as a control for an individual tile.

Implement “expand” as an inverse of the “collapse” function. This may call for a slider control rather than two sets of buttons.

Rename “Snap” to “Snip” and implement a print-pattern function called “Snap.”

Long-term considerations

Think through the use of the grid in Quilt mode and consider introducing an on/off, “snap-to” function that would allow individuals to work with the grid in preferred ways.

Consider allowing the child to control ways in which the computer modifies a Shuffle pattern. How would such a capability affect the Kaleid mode? How would it affect our formulations of manual vs. automatic modes? Should we develop a reformulation that allows for both Child and System working areas in each mode?

Add Kaleid and Draw modes, and the capability to change background colors in Quilt mode. However we should probably not allow the background color be carried over into the Child’s Area of Tiling mode, nor should we allow the child to change the background color in this area.

We may be overconstraining the order in which a quadrant construction occurs. The only requirement should be that a pattern can be spread under just two conditions: when there is a single tile in the upper left quadrant, or when all four quadrants are full. At the moment we determine these states by mandating the order of tile placement within the grid: upper left, then upper right, then lower left, the lower right. It would be better if the child could place tiles in any order, and we would have a conditional test for when there is a single tile in the upper left quadrant, or when all four quadrants are full. On the other hand, a nice quality emerges from the constraint: in thinking about what tile to choose next for a construction, its placement affects consideration of which geometric transformation/s to apply.

Implement image processing techniques to compensate for loss of resolution as tiles and patterns increase in size.

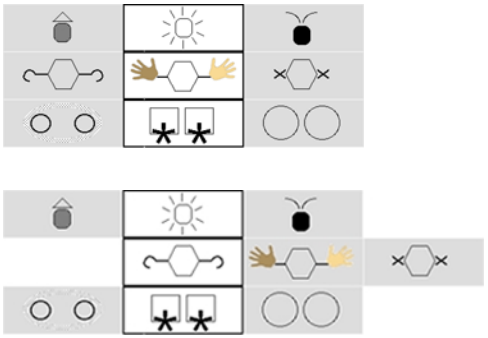
Enable a “zoom” function as a way of changing the view of a field. Note that distinctions between zoom, expand, and size should be preserved.

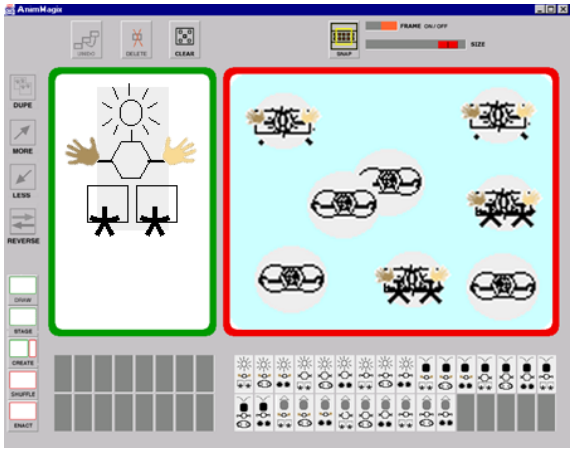
Appendix 5: *AnimMagix* Overview

Children create whimsical creatures with anthropomorphic behaviors and launch them onto a field in which the creatures interact and affect one anothers' behaviors. They become like a dancers on an ice rink. Their movements and intentions seem to change as they glide from one partner to the next. Children can explore these emerging effects by isolating and saving interacting creatures for further experimentation.

AnimMagix maintains the constructive-dialogic style of interaction as well as the general functions introduced in PatternMagix: Undo, Delete, Clear, and Snap. The Library includes ready-made creatures and empty slots for the child's creations. Working in five modes – Draw, Stage, Create, Shuffle, and Enact – the child can duplicate creatures and apply basic operations to inhibit, augment, or reverse behaviors.

In Create mode, the Child Area offers a tripartite niche for assembling creatures' heads, bellies, and bases. Each section can display several instances of the associated body part. The child clicks until the desired part is visible.





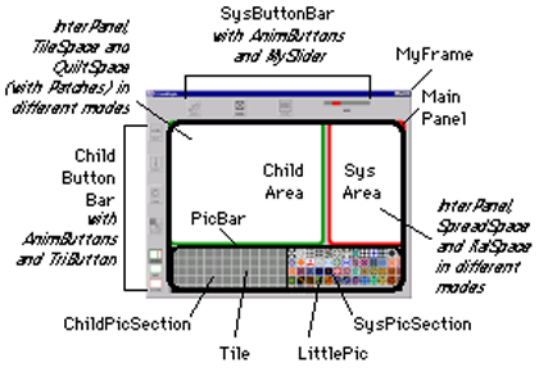
Each body part represents a behavioral component that combines others to compose an overall behavior for the creature. The System Area shows many creatures in bird's-eye view. As the creatures influence one another's behaviors, complex patterns emerge.

In Draw mode, the child adds colorful masks and costumes. Personalized creatures are saved to the child's part of the Library.
In Stage mode, creatures dragged from the Library become dancers who interact with each other as dyads, triads, and groups.
In Shuffle mode, the system cycles through a series of transformations to modify individuals' behaviors and thus the overall "dance".
In Enact mode, the system applies transformative operations to each component of an individual, again affecting the overall "dance".

The operations are basic transformations of cybernetic systems, which echo those found in living, learning organisms. Each operation can have an effect on a specific behavior: "More" augments, "Less" inhibits, and "Reverse" imposes an opposite state.

Appendix 6: *Magix* Component and Object Hierarchies

The *Magix* learning environments are implemented in Java and make use of elements in its API hierarchy. The interface builds as a set of nested objects and subclasses. Each element includes information about associated bitmaps, screen locations, etc. It also includes instructions for an action to be executed in the event of a mouse click. If the element is clicked but has no such instructions, it passes the event message up to the next element in the hierarchy.



Magix components inherit mainly from Java's *Panel* and *Canvas* classes. Panel subclasses include MainPanel, ChildButtonBar, SysButtonBar, ChildArea, SysArea, PicBar, and Tile. Canvas subclasses include LittlePic, TileSpace, QuiltSpace, SpreadSpace, KalSpace. Drop, Flip, Turn, highlight (invert), and graying out extend Java's ImageFilter class. Patches extend the Rectangle class.