Tangible Math

Lori L. Scarlatos

Department of Computer and Information Science, Brooklyn College of the City University of New York, 2900 Bedford Avenue, Brooklyn, NY 11210

Email: lscarlatos@yahoo.com

Educators recognize that group work and physical involvement with learning materials can greatly enhance the understanding and retention of difficult concepts. As a result, math manipulatives - such as pattern blocks and number lines - have increasingly been making their way into classrooms and children's museums. Yet without the constant guidance of a teacher, students can easily become distracted, confused, or frustrated.

This paper describes how math games with tangible user interfaces can address this need. Tangible user interfaces allow the students to work together in a physical environment, while the underlying computer system provides the guidance that a teacher would. This paper discusses pedagogical principles and an approach to designing and developing games that utilize tangible technologies. A library of functions, designed specifically for tracking visual tags in math games, is presented. The paper concludes with examples of math games that have been prototyped with this library.

Keywords: tangible user interface, physical computing, multi-modal input, collaborative learning, math manipulatives

INTRODUCTION

In an increasingly "flat earth" economy, math and science education is vital to a country's competitiveness in a global market. President Bush reiterated this national priority in his February 2006 State of the Union speech (Bush 2006). Yet it is not enough for students to learn math; they must also understand it.

Math games – using manipulatives, puzzles, and physical activities – provide numerous benefits in the learning environment. First, they provide concrete representations of abstract concepts, thereby helping more children to understand those concepts. Second, they provide and opportunity for children to explore and test their understanding of math concepts. Third, they enable groups of children to work together, talk about the problem at hand, and learn from one another. Fourth, they give active young children the opportunity to be physically engaged in their lessons, as opposed to sitting through a "boring" lecture or filling out workbooks. Finally, these physical activities give children a fun way to practice their math skills, which helps with retention and transfer as cognitive scientists have shown.

Because of all these benefits, schools and museums alike have embraced the use of physical learning activities to supplement learning. Teachers will often divide their classes into small groups and have them work on a particular manipulative. Other times, the teacher will use overhead displays to talk about the manipulatives while the children work at their desks with their own pieces. At the Goudreau Museum for Mathematics in Art and Science, children look forward to being able to play with the math puzzles after a more formal lesson. The trouble comes in trying to direct and help thirty-five children working on twenty-two puzzles all at once. Students often need an attentive teacher or guide, who will ask thought-provoking questions and make suggestions that will help

This paper, which primarily presents new results and unpublished work, contains references to some work that has been presented in conference and technical papers, as cited in the references.

them to discover the solution on their own. If the teacher is busy with one group, other groups that are "stuck" will quickly get bored and often find inappropriate activities to occupy themselves with. At the New York Hall of Science, student "explainers" are available to provide help as needed. This museum has discovered that having a knowledgeable "guide on the side" helps their visitors to get more out of the exhibits and activities. Yet in some settings, this is not always practical.

We address this issue by developing math games that employ tangible user interfaces. Tangible user interfaces are ideal because they allow students to explore, test, and practice mathematics both physically and collaboratively. By making the math manipulatives part of the user interface, the computer application is aware of what the children are doing, and can therefore provide appropriate feedback as needed. The software can also record everything the children do, so the teacher can refer back to it later.

Our contributions in this area are twofold. First, we have developed a pedagogical approach that emphasizes physical activity, collaboration, practice, and teacher customization. Second, we have created a library of functions that enable a developer to rapidly prototype a game with vision-based tracking in a tangible user interface. This library is available for free on our website. We have used both the approach and the library to develop a series of math games with tangible user interfaces.

This paper presents the background research motivating this work, the pedagogical ideals that guide this work, and the tools that we use to create educational applications. It also describes some of the math games that we developed, including one with a front-end tool that enables the teacher to customize an educational application to suit his or her curriculum, class, even individual students. Although the focus of this paper is on math games, the approach – and the tools – described here may be readily extended to other learning activities in the sciences, social sciences, and language arts.

RELATED WORK

The educational literature is riddled with examples of how collaborative and exploratory learning leads to greater understanding. This has led to calls for teachers to engage students in "active" learning. The National Council of Teachers of Mathematics (2000) states, "Students must learn mathematics with understanding [by] actively building new knowledge from experience ...". Holt (1983) argues that children, when encouraged to explore on their own, will come to understand mathematical principles that most educators would not dare to try to teach them. Gardner (1983) theorizes that different people can be "intelligent" in a variety of different ways, and that teachers must therefore address these multiple intelligences in order to teach effectively.

As a result, math teachers are being encouraged to involve students in collaborative activities, plan differentiated tasks, and use "workstations" to present the same material in different ways (Bransford et al, 2000). In response, teachers increasingly rely on math manipulatives – such as Cuissenaire Rods, Pattern Blocks, and Tangrams – to illustrate math concepts and provide students with a way of further exploring those concepts. Apparently children who normally have trouble with math find the concepts more understandable when they are represented by tangible objects.

Tangible User Interfaces

Tangible user interfaces (TUI) are computer-human interfaces in which the boundaries between data and representation are blurred (Ishii 1997). With this paradigm, data representations are tangible, or physical, entities in the environment. Manipulating these tangible entities results in corresponding manipulations of the data in the computer's memory. As a consequence, working with the data becomes far more intuitive (and potentially collaborative) than it is with a traditional graphical user interface using a mouse or keyboard. At a time when teachers are being encouraged to make learning experiences more hands-on, researchers are exploring ways of using TUI for educational purposes.

For example, several TUI projects are designed to develop children's language arts skills by encouraging them to tell stories (Zhou

et al, 2004 and Moher et al, 2005). In both of these cases, the TUI provides props for telling the stories as well as a means of recording the stories. In a variation on this theme, Stanton et al (2001) have had children help to design an interface that enables other children to tell their stories. The advantage of involving children in the design process is that the result is more likely to be usable by children than an application written by adults with little insight into how children think (Druin 1999).

Tangible user interfaces are also used to teach children about how systems work (McNerney 2004, Zuckerman 2005). In both of these cases, children physically connect computational components to build a system. Resulting connections among these components determines the behavior of the system; children are then encouraged to make changes to the system configuration to see the effects on the output. Horn and Jacob (2006) have taken this idea a step further, having developed a tangible programming language called Quetzal. Elements of the language are represented by physical components that look a lot like elements in a flow chart. Children connect these (in flow chart fashion) to "write" a simple program for an RCX brick.

A more recent trend is to look for frameworks for analyzing and creating applications with tangible user interfaces. Marshall et al (2003) define different ways of conceptualizing tangibles to support learning. Specifically, they look at how learners will use the tangibles ("ready-at-hand" versus "present-to-hand"), the learning activities they will engage in (expressive versus exploratory), and the model embodied by the tangibles (theoretical versus practical). Calvillo-Gomez et al (2003) and Ullmer et al (2005) both propose using "token+constraint" paradigms as a way of understanding all tangible user interfaces. In these paradigms, tokens are the physical representations of data and constraints define how those tokens can be manipulated. Fishkin (2004) uses metaphor (i.e. the way that the tangible represents abstract concepts) and embodiment (i.e. how closely tied the input and output representations are) in his taxonomy for comparing and contrasting work on tangible user interfaces.

Our project is unique in that it focuses on TUI in math and science educational software (Scarlatos 2002). Building on this initial work, we have developed our approach to a point where our principles and tools may be used by a much wider audience to create a variety of Tangible Math (and science) games.

PEDAGOGICAL APPROACH

Our approach to designing Tangible Math software is to extend tried-and-true math puzzles and activities – many recommended by the National Council of Teachers of Mathematics (2000) – using both TUI design concepts and gameplay design guidelines. Culminating from the experience of building and testing our own educational software with tangible user interfaces, we have developed a series of pedagogical guidelines that are described below. While each of these guidelines may appear to be distinct, all must be blended together to create a cohesive whole.

Physical Activity

Physical involvement in a learning activity – whether a child is moving objects or his/her own body – helps to make the activity more meaningful for the child. This is partly because the physical objects help children to see – and therefore better understand – abstract concepts in a new way. The other benefit of physical involvement is that the child must use more than his or her brain in the activity. It is difficult to doze off when one is jumping around on a number line.

The trick is discovering the best physical representation of the abstract concepts being taught, so that manipulation and computer-supported feedback are leveraged to enhance learning. Abstract concepts must be mapped to physical entities and environments in a way that helps the child to make the connection between the concrete and the abstract. For example, quantity may be translated from "area of a polygon" to "number of triangles that fit in this shape". One could argue that the kind of understanding that evolves from the physical notion of moving tiles around may be different from the mental image produced by looking at a polygon.

Collaborative Learning

Group work increases the learning potential of an activity by allowing children to scaffold off of each other's prior knowledge. This has been shown to be especially effective for girls (Inkpen et al 1995). Talking about the activity also helps children to develop meta-cognitive connections that are useful in transference of the knowledge.

Yet if one expects children to collaborate, one must provide a learning environment that gives all of the children equal access to the data and equal opportunities to manipulate that data. We, for example, have developed a TICLE (tangible interfaces for collaborative learning environments) table that we use in most of our tangible math games (Scarlatos 2002). This table (shown in figure 1) allows several children to work with a puzzle or math manipulative at the same time. A camera beneath the table (looking up) eliminates the problem of visual obscuration, while a touch screen gives all of the children equal access to the software's extra help.

Figure 1. Children using a TICLE table have equal access to the tangible data (puzzle pieces) and further information (from the touch screen).

Practice That's Fun

Practice helps to reinforce concepts that have been learned so that they are more deeply engrained in the learner's mind. This is the purpose of doing homework, though most children find typical homework assignments to be boring. Although new concepts are best taught by teachers, computer software is well suited to guiding students as they practice applying those concepts.

The best way to make practice fun is to make a game of it. Increasing difficulty, coupled with increasing rewards, helps to keep the activity interesting. In our observations, we have noticed again and again that when children feel that they have worked hard and then "won", they are eager to try the activity again, even if it is practicing math concepts. And how many children finish their homework and then think, "I won!"?

Immediate Feedback

One of the key advantages of having a computer "watch" what the children are doing is that the computer can provide immediate feedback. Our software typically provides three types of feedback.

First, when the children make progress toward a goal, encouragement is given: either verbal, visual, and/or a numeric score. When designing the form of the encouragement, an understanding of the system of rewards provided by commercial video games, as described by Oxland (2004), is essential. Our observations (Scarlatos et al 2002) have revealed that too much encouragement is seen as condescending, and can even become annoying.

Alternatively, if the rules of the game are not followed (e.g. the child does not use the correct pieces), the software must provide feedback that gets the child back on track. The need for this type of feedback is unique to TUI because, unlike a typical computer or video game, the data representations in a TUI are not under the control of the computer. We generally address this with a gentle reminder about what the goal is, and what constraints go along with that goal. Again, it is important not to be adamant about this; repeated reminders or corrections can become distracting and discouraging.

Finally, when the students are "stuck", the software must be able to give them hints. Students should have to ask for hints explicitly. Again, we have observed that most children will initially want to try to solve a problem without any hints. It is only when they have failed to make progress over a period of time (typically two to five minutes) that they will ask for help. The hints must be contextually relevant, i.e.

selected based on the state of the game or what the students have already done. At the same time, the hints must be designed to get children to think about the problem at hand; telling children to "put this there" doesn't help them to solve a similar problem later. This ability to transfer knowledge, and solve problems without the help of a computer later on, is an essential skill that all math games should strive to develop.

Information Gathering

One of the greatest advantages of using a computer for learning activities is that the computer can record what the student has done. The teacher may then review this information later, to determine where the student (or the class) is having trouble and deduce what they need to receive additional instruction on.

All of our tangible math games record student performance data for the teacher to look at later on. Storing this data in a flat file (with a time/date stamp) is simplest to implement and allows for anonymous activity. For more detailed analysis, it is preferable to store the information in a database. This allows the teacher to track a particular student's progress over time, look for classroom trends, or get different "views" of different student groups.

Teacher Customization

Despite a developer's best efforts, a computer game can never meet all of a teacher's needs for a particular subject and class combination. Even within a single class, a teacher may feel the need to provide slightly different learning activities for different groups of children.

For this reason, most of our tangible math applications rely on an input text file. This file defines a wide variety of parameters, from what the problem or puzzle is, to what tangible entities are being tracked, to what hints should be given in what circumstances. We then can (and have) develop(ed) a separate teacher's interface which is used to define these parameters.

TECHNICAL APPROACH

When developing math games, we take an iterative development approach using a multimedia authoring tool. For most of our applications, the tool we have chosen to use is Macromedia Director. In addition to being an excellent rapid prototyping tool, Director also provides a powerful programming language called Lingo. When Lingo does not satisfy all of a developer's needs, Director's capabilities are easily extended by integrating code libraries known as Xtras.

The remainder of this section describes our own Anna Xtra, which we use to track visual tags in a collaborative learning game. We created this Xtra because existing libraries did not adequately support the tracking of visual tags in games that use tangible user interfaces. The functions supported by this library, and their underlying algorithms, were first described by Mbogho and Scarlatos (2005). Here, we describe how Anna is used to build tangible math games, supplementing the descriptions with code samples. This Xtra is available as freeware on our web site: http://www.sci.brooklyn.cuny.edu/~lori/TICLE/.

Vision-Based Tagging

In our early work with tangible user interfaces, we experimented with computer vision using a third-party Xtra. Although we had good results from the Tangram application that we produced (Scarlatos 2002), calibration was difficult. This problem arose because the functions were looking for RGB colors with given tolerances or allowable variance. Although small changes in light are nearly imperceptible to human eyes, they create changes in the RGB values that make the color look very different to a computer. In fact, some colors that are close to one another in RGB space can look much more different to a human (e.g. a color with red added to it) than colors that are far apart (e.g. a color that is lightened by adding equal parts of red, green, and blue).

In response to this problem, we developed our own computer vision Xtra which we call Anna. Anna differs from earlier computer vision Xtras in that it looks for matching colors in two different color spaces: YIQ and HSV. YIQ corresponds to the color system used in video, and allows us to specify different tolerances in terms of luminance (Y) and chrominance (IQ).

Consider, for example, an environment where light conditions can change drastically, such as outdoors or near a window. When looking for a specific color, we would want the chrominance (IQ) to be a close match, but could allow much greater variance in the luminance (Y) values.

HSV (hue, saturation, value) corresponds more closely to human vision. In our experiments, we have discovered that looking for a match in HSV space produces the most reliable results when tracking visual tags. The reason for this is that, to make the tags stand out, we have chosen to use highly saturated colors. Using HSV we can narrowly restrict the range of hues (H) being sought, set a minimum saturation (S) threshold, and allow for greater variance in value (V or luminance). As a result, our tag tracking is much less sensitive to changes in light conditions in the environment.

An application using Anna is initialized by creating a tag reader object, which is used in all subsequent calls to Anna functions. Figure 2 provides the Lingo code for creating and destroying this object.[1]

```
-- Initialize Anna
global TRobj -- Anna object

on startMovie
  TRobj = new(xtra "Anna")
end startMovie

-- Clean up at the end
on stopMovie
  TRobj = 0
end stopMovie
```

Figure 2. Lingo code to start up Anna Xtra.

Tagging Strategies

As we continued to build math games using computer vision, we identified the following visual tagging strategies as effective for a number of 2D puzzles and manipulatives. Objects with these tags may be tracked on a TICLE table (with the tags on the underside). They may also be viewed by a camera from above (with the tags facing up) or from the side (with the pieces stacked vertically).

[1] In Lingo, "--" marks the beginning of a comment. The end of a line marks the end of a statement; statements that need more than one line use the continuation character "\".

Space-Filling Tags

Some math manipulatives and puzzle pieces are created from repetitions of a single elemental shape. For example, each Cuisenaire rod is a linear arrangement of N cubes, where N represents the length, area, and numeric value of the rod. Pentomino pieces are arrangements of five squares, just as Tetris pieces are arrangements of three squares. Pattern blocks are (mostly) shapes made up of one or more equilateral triangles: hexagon, trapezoid, rhombus and triangle. All of these manipulatives can be used to teach children about measuring area and creating shapes. Thus in games using these types of puzzle pieces, it is most important to track how the pieces fill the space.

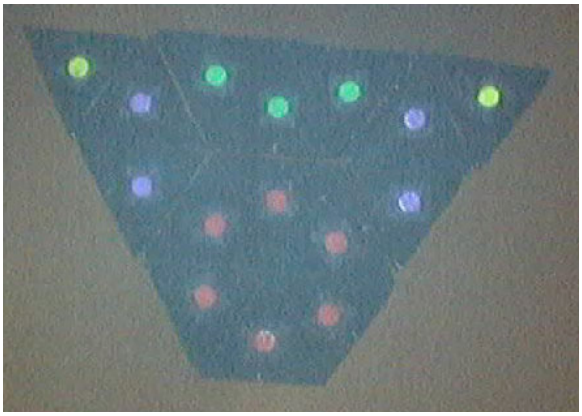

Figure 3. Space-filling tags mark the center of each elemental shape, using color to identify the larger piece using that shape.

Space-Filling Tags are created by placing a colored spot in the exact center of each elemental shape. For example, a hexagon would be tagged with six spots; each pentomino piece would have five spots. The colors of the spots identify the overall shape of the piece. The resulting pattern of spots seen by the camera, as shown in Figure 3, can then be used to orient and fill a triangular or square grid.

Figure 4 shows the Lingo code for interpreting these tags and using that information to fill in a grid space, which is represented in the computer's memory. This code assumes that functions (handlers) have been specified to check the size of the bounding rectangles (to make sure they are really tags) and

to calculate a grid index based on a tag's position in the image.

```
-- Given an array of colors (CList)
-- and a list of color labels (CLabel)
-- fill in a Grid with those labels

global TRobj -- Anna object

on fillGrid CList, CLabel, Grid

  -- initialize the Grid = all empty
  repeat with i = 1 to Grid.count
    Grid[i] = #empty
  end repeat

  -- Capture image of the game state
  getImage(TRobj)

  -- Loop through all of the tag colors
  repeat with i = 1 to CList.count

    -- find rectangles bounding
    -- all tags with this color
    cp = CList[i] -- color parameters
    bounds = hsv_getBoundingRecs(TRobj,\
          cp[#color], cp[#hueRange],\
          cp[#satLimit], cp[#valRange])

    -- get label to fill the grid with
    thisLabel = CLabel[i]

    -- loop through all rectangles, &
    -- label grid elements
    repeat with b in bounds

      -- ensure spot is the right size
      if sizeOK(b) then

        -- get the Grid index (gi)
        -- for this location
        gi = getIndex(b)

        -- label the grid element
        Grid[gi] = thisLabel
      end if
    end repeat
  end repeat

end fillGrid
```

Figure 4. Detecting and interpreting space-filling tags with Anna.

Orientation Tags

For some puzzle and game pieces, such as Tangrams and Scramble Squares, it is important to know not only what piece is visible and where, but also how it is turned (oriented) relative to the other pieces. This information can be used to determine a relative configuration or state of the puzzle that is independent of rotation and translation factors. The game can then use the state information to determine how close the players are to reaching a solution. The game can also use this information to select an appropriate hint when asked.

Orientation Tags are composed of three closely spaced collinear spots of color, similar to the strategy used by Underkoffler and Ishii (1998). One color (such as yellow) is reserved for marking the center of mass for the piece. The colors of the two adjacent spots uniquely identify the piece. So, using three colors (in addition to the color reserved for the center spot), one can uniquely identify nine pieces. The three spots form a vector, with its tail at the center, which indicates the orientation of the piece. Figure 5 shows a collection of Tangram pieces with orientation tags.

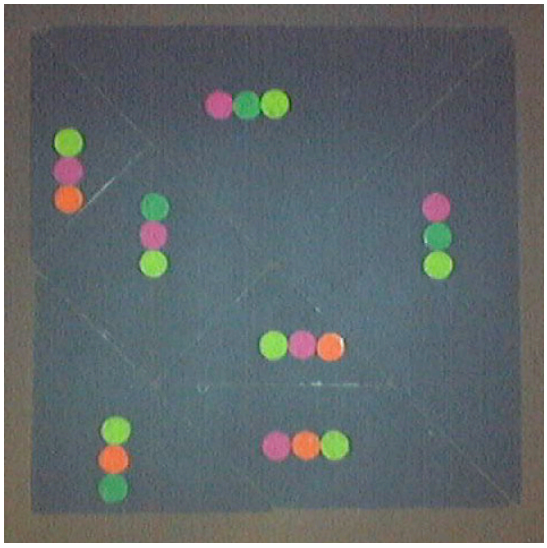

Figure 5. Orientation tags identify the puzzle pieces while also showing their location and orientation.

Figure 6 lists the Lingo code for finding a set of tags. Here, it is assumed that a Tag class (parent script) has been defined that includes properties representing an identifier (indicating the type of piece it is), an absolute location, and an orientation vector. Methods are defined to add spots to the tag pattern (addSpot2 and addSpot3) and then update the properties of the tag (update) based on these. If the spots in a tag object are not collinear, or the pattern of colors is not recognized in this game, then the update method returns a value indicating that the result

is not valid. The code in figure 6 also assumes that a function (getClosestTag) has been defined to determine which Tag is closest to a given bounding rectangle. Because this function uses partitioning techniques to increase efficiency, a value of VOID is returned if no tag center is within a reasonable distance of the given rectangle. The function inRange determines whether a tag is reasonably within the given distance of a given bounding rectangle.

Using the Tag data generated by this function, a game can then match these tags to known pieces of the puzzle, to determine where those pieces are. Related to an internal representation of the piece's geometry, this information can then be used to determine how two pieces are touching. This, in turn, can be used to determine the current state of the puzzle or game (Scarlatos 2002).

```
-- Given a list of colors (CList, where
-- CList[1] = the color of the center)
-- and a target distance between spots
-- (dist, measured center-to-center),
-- return a list of Tags, each one
-- having an identity, location
-- and orientation vector

global TRobj -- Anna object

on findTags CList, CLabel, dist

  -- initialize the Tag list
  Tag = []
  nTags = 0 -- number of tags found

  -- Capture image of the game state
  getImage(TRobj)

  -- Find all the center spots
  cp = CList[1] -- color parameters
  bounds = hsv_getBoundingRecs(TRobj,\
         cp[#color], cp[#hueRange],\
         cp[#satLimit], cp[#valRange])

  -- Create new tag for each spot found
  -- using bounds to indicate position
  repeat with b in bounds
    if sizeOK(b) then
      nTags = nTags + 1
      Tag[nTags] = new(script "Tag", b)
    end if
  end repeat

  -- Loop through remaining tag colors
  repeat with i = 2 to CList.count

    -- get color label for tag info
    thisLabel = CLabel[i]

    -- find rectangles bounding
    -- all spots of this color
    cp = CList[i] -- color parameters
    bounds = hsv_getBoundingRecs(TRobj,\
          cp[#color], cp[#hueRange],\
          cp[#satLimit], cp[#valRange])

    -- loop through all rectangles
    repeat with b in bounds

      -- ensure spot is the right size
      if sizeOK(b) then

        -- find the closest tag
        nearest = getClosestTag(b, Tag)

        -- if something was returned,
        -- see if this is part of a tag
        if (nearest <> VOID) then
          if inRange(b, nearest, dist) then
            nearest.addSpot2(thisLabel, b)
          else if inRange(b, nearest, \
                        dist*2) then
            nearest.addSpot3(thisLabel, b)
          end if
        end if
      end if -- sizeOK
    end repeat -- every rectangle
  end repeat  -- every color

  -- Update the tags, eliminating any
  -- that are invalid
  repeat with i = nTags down to 1

    -- update tag, returning status
    OKstatus = Tag[i].update()

    -- delete any tags that are not OK
    if NOT OKstatus then
      delete(Tag, i)
    end if
  end repeat

  -- Return resulting list of tags
  return Tag

end findTags
```

Figure 6. Detecting orientation tags with Anna.

Attribute Tags

Both mathematicians and scientists use classification and data analysis techniques to categorize things. Classification and data analysis are important components of the math and science standards (Van Der Wallen 2000), and are tested on state standardized tests. To gain experience with this activity, attribute materials are typically used. These may be structured sets – with exactly one piece for every

possible combination of values – or unstructured sets – made up of sample objects collected by the teacher or children. The attributes themselves may be either binary or multi-valued.

Binary attributes either have a given property, or they don't. For example, a rock may be either shiny or not shiny. Binary attributes are especially useful for a classification activity in the sciences known as dichotomous sorting. **Binary Attribute Tags** employ a solid-colored spot to represent the presence of an attribute, and a spot with a hole in it to represent the absence of that attribute. The color of the spot indicates which attribute it is. Figure 7 shows the undersides of an unstructured attribute set with five different attributes.

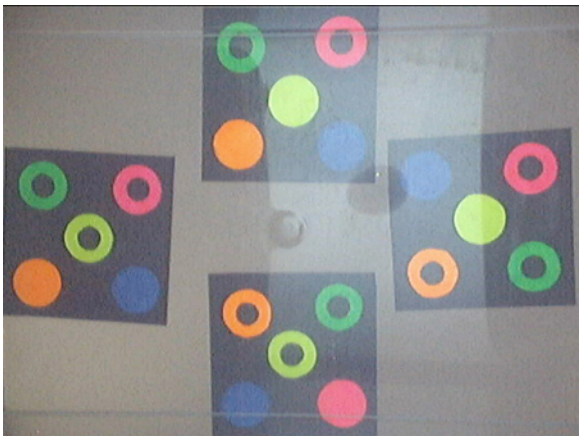

Figure 7. Binary attributes are either there (solid) or not there (hole).

Figure 8 shows Lingo code that determines, for a given collection of objects, what attributes those objects do and do not have. Although this code does not consider the locations of the attributes, the locations may be readily accessed. This could then be used to imply an ordering of a set or membership in a Venn diagram.

```
-- Given an array of colors (CList) and
-- % of size determining solidity (f)
-- return a list of attributes (Alist)
-- indicating how many positive (#yes)
-- and negative (#no) values are found

global TRobj -- Anna object

on getAttributes CList, factor

  -- initialize the Alist
  Alist = []

  -- Capture image of the game state
  -- & save a copy in cast member 1
  getImage(TRobj)
  showImage(TRobj, 1)

  -- Loop through all of the tag colors
  repeat with i = 1 to CList.count

    -- find rectangles bounding
    -- all tags with this color
    cp = CList[i] -- color parameters
    bounds = hsv_getBoundingRecs(TRobj,\
          cp[#color], cp[#hueRange],\
          cp[#satLimit], cp[#valRange])

    -- loop through all rectangles,
    -- counting attributes
    yesAtt = 0
    noAtt = 0
    repeat with b in bounds

      -- ensure spot is the right size
      if sizeOK(b) then

        -- count # of pixels with color
        count = hsv_countPixels(TRobj, 1,\
           cp[#color], b, cp[#hueRange],\
           cp[#satLimit], cp[#valRange])

        -- update count accordingly
        if (count >= (sizeOf(b) * f)) then
          yesAtt = yesAtt + 1
        else
          noAtt = noAtt + 1
        end if
      end if
    end repeat

    -- Add a record to the Alist
    aRecord = [#yes : yesAtt, #no :
noAtt]
    Alist[i] = aRecord
  end repeat

  -- Return the results
  return Alist

end getAttributes
```

Figure 8. Detecting and interpreting binary attribute tags with Anna.

Multi-valued attributes appear in activities that focus on sets, Venn diagrams, and partitionings. **Multi-Valued Attribute Tags** can be created from a bi-directional tri-color barcode (Mbogho and Scarlatos 2005). Two of the colors in the barcode represent binary values (0/1). Bars in these colors represent a 4-bit number (that is then repeated backwards) and mark the start (and end) of the code. By repeating the

code in this manner, the orientation of the barcode becomes unimportant. The third (background) color in the barcode represents an attribute category. Thus an image containing many different barcodes, representing several different attributes, can be interpreted without regard for location, orientation, or scale. Figure 9 shows a rock with three different attributes with values 0 (red), 3 (yellow), and 1 (blue).

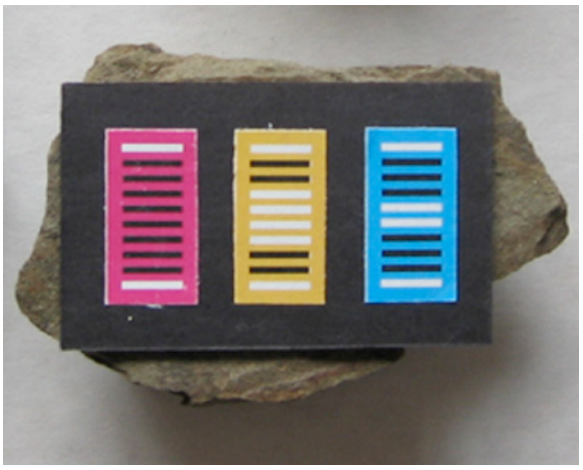

Figure 9. Bi-directional barcodes represent different attribute values, indicated by the background color.

Figure 10 shows the Lingo code for reading these barcodes. This function returns, for each attribute, a list of [value, position] pairs to indicate what attribute values are found where. This may then be used to check set orderings or location in a Venn diagram.

```
-- Given an array of attribute
-- names (Nlist) and colors (CList),
-- return a list of attributes (Alist)
-- that specifies the value & location
-- of each tag found for that attribute

global TRobj -- Anna object

on getBarCodes CList, NList

  -- initialize the Alist
  Alist = []

  -- Capture image of the game state
  getImage(TRobj)

  -- Loop through all of the tag colors
  repeat with i = 1 to CList.count

    -- get attribute name
    thisName = NList[i]

    -- find rectangles bounding
    -- all tags with this color
    cp = CList[i] -- color parameters
    bounds = hsv_getBoundingRecs(TRobj,\
          cp[#color], cp[#hueRange],\
          cp[#satLimit], cp[#valRange])

    -- loop through all rectangles,
    -- reading barcode values
    values = []
    repeat with b in bounds

      -- ensure barcode is the right size
      if sizeOK(b) then

        -- read the value of the barcode
        num = hsv_readTag(TRobj, \
           cp[#color], b, cp[#hueRange],\
           cp[#satLimit], cp[#valRange])

        -- add it to the list
        add (values, [num, centerOf(b)])
      end if
    end repeat

    -- Add a record to the Alist
    addProp(Alist, thisName, values)
  end repeat

  -- Return the results
  return Alist

end getBarCodes
```

Figure 10. Detecting and interpreting multi-valued attribute tags (bi-directional bar codes) with Anna.

GAMES

Nearly all of the games that we have developed use wooden pieces marked with visual tags. Early on, we discovered that children tend to be very rough with objects that they are handling, especially in museum settings. The wooden pieces can (and have) been stacked, knocked over, and dropped on the floor; the most that we have to do is occasionally touch up the tags.

Tangrams

The tangram is a classic puzzle used to teach geometry concepts, such as comparing area and creating shapes sets (Van de Walle 2000). Tangrams also help to develop problem-solving skills. Invented centuries ago in China, tangram puzzles have appeared in books, games, and state proficiency exams.

Figure 11. Tangram puzzle with tangible user interface.

We created software that uses orientation tags to track the tangram pieces on a TICLE table (figure 11). Children select a Tangram puzzle to solve, which determines the solution being sought and the set of hints that may appear, depending on the state of the puzzle when they ask for help. While the children are working on the puzzle, the software continuously polls the computer vision system to determine whether or not the children are making progress. Encouragement is offered periodically when they do make progress. Context-sensitive hints, as shown in figure 12, ask the children to solve sub-problems that are designed to help them "see" what they need to do next with the larger puzzle. The software records a history of moves made by the children, along with a history of the hints seen.

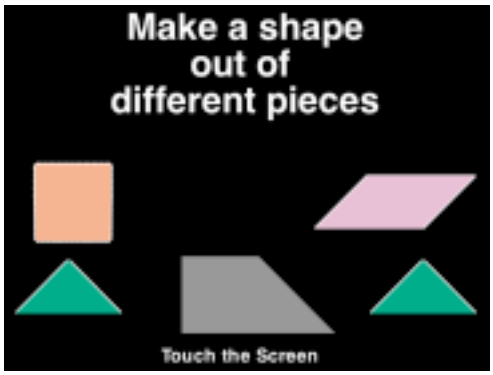

Figure 12. Hints are designed to help students to "see" how they might fill in a space in the puzzle.

Pattern Blocks

Patten blocks are simple polygonal shapes that include a hexagon, a quadrilateral (half of the hexagon), a rhombus (one third of the hexagon), and an equilateral triangle (one half of the rhombus). These blocks can be used in a wide variety of ways to teach several different concepts. For example, because the smaller blocks can be combined to create larger blocks, these can be used to teach fractions, multiplication, and division. Pattern blocks can also be used to teach tessellation concepts.

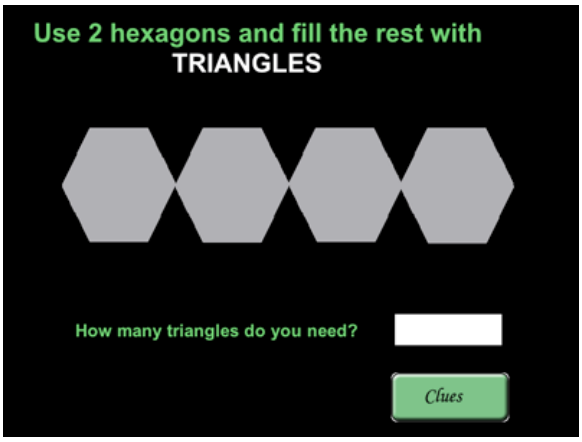

Figure 13. Problem posed by Pattern Block software.

We developed software that helps children to understand the concept of area by asking them to cover a pattern using specific pattern blocks. The software only looks at the space-filling tags on these pieces if the question was answered wrong; if the wrong pieces are being used, the software reminds the children how they can recognize the pieces they are looking for.

Cuisenaire Rods

Cuisenaire Rods have been used in classrooms for decades (Holt 1983) to teach a variety of number concepts. Among these are the difference between odd and even numbers; addition; multiplication as multiples of a given number; and fractions.

We have developed an application that focuses on the concept of odd versus even numbers. Children are given a sequence of increasingly difficult exercises that reinforce the concepts. Children may also choose to solve puzzles, in which they are asked to fill in a space

using a particular type of Cuisenaire rod. Figure 14 shows one of these puzzles. In all of these activities, the computer checks to ensure that the children are using the right type of rods (i.e. odd- or even-numbered length).

Figure 14. Puzzle using Cuisenaire rods reinforces lessons on odd versus even numbers.

Tower of Hanoi

The Tower of Hanoi is a classic puzzle known and loved by many. For younger children, it is a challenging puzzle that can be made successively harder by adding more and more disks to the puzzle. For older children, it is an ideal tool for demonstrating the concepts of recursion and recurrence functions.

Figure 15. Children playing with the Tower of Hanoi.

Our Tower of Hanoi software uses an overhead camera to track the movement of brightly-colored disks on a stand with three pegs (figure 15). In this application, the entire disk is a tag: if it disappears from view for an extended period of time, it is assumed that either the disk is not in play, or it is underneath a larger disk. Either condition constitutes a violation of the rules. While the students work with the tower, a diagram of Serpinski's triangle shows the sequence of steps as a path through a graph. In this diagram, the optimal solution to the problem is represented by the shortest path from one corner of the triangle to another. The diagram therefore helps children to "see" when they have made an extraneous move.

Attribute Sorting

Classification (the process of making decisions about how to categorize things) is an important skill that is employed both in math and in science. Children typically learn to classify objects by working with attribute sets (Van de Walle 2000).

We have developed a program that encourages children to analyze and categorize an unstructured attribute set assembled by the teacher. As shown in figure 16, the advantage of using real objects in a set is that children can make observations that would not be possible with virtual objects ... such as testing to see which rocks are harder than a nail. In addition to saving a list of hints seen and the state of the system at key stages, the software collects and stores the students' own reflections about what the objects on the table have in common.

Figure 16. Children testing the hardness of a rock for the attribute sorting program.

A separate teachers' program allows teachers to define the attribute sets they are using, and the hints that they wish to offer to the students. Hints can be anything that can be displayed in a web page, making this accessible to both

technical novices and technically savvy teachers. We have had teachers use this interface to define the attributes of rocks, leaves, flying insects, and buttons.

FUTURE TRENDS

This work is continuing in three ways: through development, testing, and extension.

Development includes the creation of additional software activities with tangible user interfaces. In the area of attribute sorting, we plan to create math games that use Venn Diagrams, "Which One Doesn't Belong", and ordered sets. We have also been working on an application that uses pattern blocks to learn about and create tessellations. In addition, we plan to develop teachers' interfaces for more of our games.

Testing involves bringing our software to the classroom and having teachers evaluate the practicality and benefits of using the games. Although we have been able to observe our software being used in museum settings, classroom use will allow us to learn more about the long-term benefits of using math games with tangible user interfaces.

Extension involves moving from two-dimensional puzzles to the third dimension. We are currently working with motion capture and other multimodal inputs to track student activity within a CAVE environment. This extension will encourage even more activity on the students' part, as well as increase the realm of possible learning activities that we can track.

CONCLUSION

We have presented both a pedagogical approach and software tools for developing educational software with tangible user interfaces. We have also described a series of software games that we developed using the presented approach and tools. Although the focus of our work has been on mathematics, the tagging strategies and pedagogy are clearly applicable to science, and probably other topics as well.

ACKNOWLEDGEMENTS

The author wishes to acknowledge the many talented students who have worked on this project. Audrey Mbogho wrote the Anna Xtra; Magdalena Jaworska, Shalva Landy, and Irene Lam worked on the games. Dave Ferguson, Eleanor Miele, Rosamond Welchman, Frederick Gardener, and Tony Scarlatos provided valuable insights and suggestions throughout this project. The participation and support of Beth Deaner, director of the Goudreau Museum of Mathematics in Art and Science, and Preeti Gupta and Frank Signorelli, from the New York Hall of Science, have been indispensible.

This work has been supported by the National Science Foundation under grant numbers 9984385 and 0203333.

REFERENCES

Bransford, J.D. et al, editors (2000) *How People Learn*, National Academy Press.

Bush, G.W. (2006) *State of the Union: American Competitiveness Initiative*. Summary available online at http://www.whitehouse.gov/news/releases/2006/01/20060131-5.html (February 2006).

Calvill-Gamez, E.H., Leland, N., Shaer, O. and Jacob, R.J.K. (2003) The TAC Paradigm: Unified Conceptual Framework to Represent Tangible User Interfaces, Proceedings of the Latin American conference on Human-computer interaction CLIHC '03, pp. 9-15.

Druin, A., Bederson, B., Boltman, A., Miura, A., Knotts-Callahan, D., Platt, M. (1999). Children As Our Technology Design Partners. In Druin, A. (ed.), *The Design of Children's Technology*, Morgan Kaufman Publishers, Inc., San Francisco, CA, pp. 52-72.

Fishkin, K.P. (2004) A taxonomy for and anlaysis of tangible interfaces, *Personal and Ubiquitous Computing*, 8(5), pp. 347-358 (Sept. 2004).

Gardner, H. (1983) *Frames of Mind: The Theory of Multiple Intelligences*, Basic Books.

Holt, J. (1983) *How Children Learn*, Da Capo Press.

Horn, M.S. and Jacob, R.J.K. (2006) Tangible programming in the classroom: a practical approach, *CHI '06 extended abstracts on Human factors in computing systems CHI '06*, pp. 869-874 (April 2006).

Inkpen, K., Booth, K. S., Klawe, M. & Upitis, R. (1995). Playing Together Beats Playing

Apart, Especially for Girls. *Proceedings of Computer Support for Collaborative Learning '95* (CSCL).

Ishii, H. and Ullmer, B. (1997) Tangible Bits: Towards Seamless Interfaces Between People, Bits and Atoms, Proceedings of the SIGCHI conference on Human factors in computing systems, Atlanta, GA, pp. 234-241.

Marshall, P., Price, S. and Rogers, Y. (2003) Conceptualising tangibles to support learning, *Proceeding of the 2003 conference on Interaction design and children*, Preston, England, pp. 101-109.

Mbogho, A. and Scarlatos, L. (2005). Towards Reliable Computer Vision-Based Tangible User Interfaces. *Proceedings of IASTED-HCI 2005 (Human Computer Interaction),* Phoenix, AZ, November 2005.

McNerney, T.S. (2004) From Turtles to Tangible Programming Bricks: Explorations in Physical Language Design, *Personal Ubiquitous Computing*, vol. 8, pp. 326-337.

Moher, T., Watson, B., Kim, J., Hindo, C., Gomez, L., Fransen, S. and McEneany, T. (2005) StoryGrid: a tangible interface for student expression, *CHI '05 Extended Abstracts on Human factors in computing systems*, Portland, OR, pp. 1669-1672 (April 2005).

National Council of Teachers of Mathematics (2000) *Principles and Standards for School Mathematics*, National Council of Teachers of Mathematics.

Oxland, K. (2004) *Gameplay and Design*, Addison Wesley.

Scarlatos, L.L. (2002). An Application of Tangible Interfaces in Collaborative Learning Environments, *SIGGRAPH 2002 Conference Abstracts and Applications*, 125-126.

Scarlatos, L.L., Landy, S.S., Breban, J., Horowitz, R. and Sandberg, C. (2002). *On the Effectiveness of Tangible Interfaces in Collaborative Learning Environments*, CUNY Graduate Center Technical Report TR-200204.

Stanton, D., Bayon, V., Neale, H., Ghali, A., Benford, S., Cobb, S., Ingram, R., O'Mally, C., Wilson, J. and Pridmore, T. (2001) Classroom collaboration in the design of tangible interfaces for storytelling, *Proceedings of the SIGCHI conference on Human factors in computing systems*, Seattle, Washington.

Underkoffler, J, Ishii, H. Illuminating Light: An Optical Design Tool with a Luminous-Tangible Interface, *Proceedings of CHI '98*, 542-549.

Van de Walle, J.A. (2001) *Elementary and Middle School Mathematics: Teaching Developmentally*, 4th Edition, Addison Wesley Longman.

Zhou, Z.Y., Cheok, A.D., Pan, J.H. and Li, Y. (2004) An interactive 3D exploration narrative interface for storytelling, *Proceeding of the 2004 conference on Interaction design and children: building a community*, Maryland, pp. 155-156 (June 2004).

Zuckerman, O., Arida, S. and Resnick, M. (2005) Extending tangible interfaces for education: digital montessori-inspired manipulatives, *Proceedings of the SIGCHI conference on Human factors in computing systems*, Portland, OR, pp. 859 - 868 (April 2005).