PUZZLE PIECE TOPOLOGY:
DETECTING ARRANGEMENTS IN SMART OBJECTS INTERFACES

Lori L. Scarlatos
Brooklyn College, CIS
2900 Bedford Ave., Brooklyn, NY 11210
lori@sci.brooklyn.cuny.edu

ABSTRACT

Smart object interfaces enable a computer to respond to a group of users' manipulations of a physical environment. This unobtrusive interface is especially well suited for providing guidance as students attempt to solve mathematical and scientific puzzles. This paper introduces a formalism for describing arrangements of smart objects on a 2D surface, and suggests a strategy for efficiently representing such arrangements in a computer application. It then shows how these techniques are implemented in a Tangram with a smart objects interface, which provides multimedia feedback as children play with the puzzle pieces.

Keywords: tangible user interface, spatial relationships, computer vision, multimodal interface, graphical interaction, multimedia, tracking, educational application, Tangram

INTRODUCTION

Graphical user interfaces rely on metaphors to achieve a familiarity that allows people to learn to use them with minimal effort. Yet nothing is as easy as manipulating things in the real world. Consider, for example, a group of children attempting to solve a puzzle on a computer. One child – who is particularly adept at translating real-world actions into cursor manipulations – operates the mouse while the other children passively look on. Now compare this to a situation where a group of children is working with a 3D puzzle. They're all on the floor; everyone is participating and actively learning.

With smart object interfaces, a computer responds to one's very natural manipulations of objects in the physical realm. This allows people to focus on the task at hand without having to worry about how to give instructions to a machine. Yet building a smart objects interface is non-trivial. Some of the problems are:

- tracking multiple objects simultaneously in a noisy (real world) environment,
- representing and detecting states that the computer should respond to,
- providing feedback in a timely manner, and
- developing a solution general enough to apply to a large class of possible situations.

This paper focuses on solutions to the last three problems[1]. It also describes how these techniques are being used in a prototype smart objects interface for a mathematical puzzle at the Goudreau Museum of Mathematics in Art and Science. This work is unique in that it:

- Extends the standard set of 2D topological spatial relationships to express all possible arrangements of two solid 2D objects relative to one another. This gives us a formalism for describing states or configurations of the interface that we wish to respond to.
- Describes an abstract data representation that incorporates these topological relations, and shows how this may be quickly updated and used to detect full or partial solutions and respond appropriately. Such rapid response is essential in a smart objects interface.
- Implements these techniques in a puzzle designed to make math and science more accessible to middle school age children. Multimedia feedback to the children's actions provides gentle, unobtrusive guidance, and increases chances that children will successfully complete and learn from the puzzle.

Although the focus of this paper is on mathematical puzzles, the techniques described in this paper may

[1] Solutions to the first problem are described in a technical report by the author [Scarl98a].

be applied to any other smart objects interface that needs to detect 2D arrangements of objects. Some examples include system simulations, planning, and design applications.

MATHEMATICAL PUZZLES

For many students, math and science are "hard" subjects that must be avoided at all cost. Yet all children – even infants – are intrigued by physical puzzles. Babies will spend hours putting things in containers and taking them out again; older children will see how high they can pile things before they fall down. For many scientists and mathematicians, our job is to solve puzzles. By showing children the physical puzzles behind math and science, we can gain their interest and help them to understand and appreciate more abstract concepts.

Bernard Goudreau, an engineer and mathematics teacher, recognized this possibility and rose to the challenge. He built twenty-two mathematical puzzles and activities, and installed them in his Museum of Mathematics in Art and Science which he founded in 1980. Located in New Hyde Park, NY, this unique learning and resource center reaches more than 15,000 people annually through workshops, programs, special events and exhibitions. A large number of the visitors enjoy working with the puzzles most of all. Yet the help of a skilled instructor is often needed to clarify puzzle objectives, remind players of the rules, provide helpful hints when the players are "stuck", encourage players when they're on the right track, and explain the underlying mathematical concepts. With only one instructor available for each group of up to 35 visitors, students don't always get the help they need, and so they give up in frustration. The smart objects project was initiated to overcome this problem. Puzzles equipped with smart object interfaces can passively "observe" students as they play, offering help, hints, reminders and explanations only when they are needed.

Other researchers, mostly at the MIT Media Labs, have done some very innovative work on using physical objects to interface with computers. The Physics and Media group is working on self-sensing everyday objects [Verpl96a] and using the user's body as an input device [Smith98a]. The Epistemology and Learning group has developed programmable play objects [Resni96a] that allow children to build their own robots. Most directly related to the smart objects is the work being done by the Tangible Media group [Ishii97a]. Underkoffler's illuminating light interface [Under98a] inspired the approach I took for detecting the puzzle piece locations [Scarl98a]. Yet none of these efforts has produced the general methodology for tracking and detecting multiple objects that is needed for smart objects interfaces at the Goudreau Museum.

SPATIAL RELATIONS AMONG PUZZLE PIECES

If we want a computer to react appropriately to arrangements of puzzle pieces – such as the puzzle solution – we must be able to represent those arrangements internally. It is important that each arrangement have a single representation that is invariant to affine transformations[2]. Likewise, an internal representation should not be applicable to more than one physical arrangement of the pieces.

In describing arrangements of two-dimensional puzzle pieces, I assume the following for the purposes of this paper.

1. A puzzle consists of a partially ordered set of puzzle pieces. Identical puzzle pieces are interchangeable, and may therefore have identical labels. We will need to respond to at least one solution (or arrangement or state) to the puzzle. We will also need to be able to recognize partial solutions.

2. Each puzzle piece is a solid polygonal area that may be defined by its boundary. This boundary may be described in terms of one or more simple polygons. A simple polygon is described by a cyclic sequence of non-intersecting straight line segments (edges) connecting a set of vertices (endpoints). Without loss of generality, we can require that these endpoints be listed in a particular order (e.g. clockwise)[3].

3. In describing arrangements, we are primarily interested in how two puzzle pieces meet. This means that we need to indicate which edges/endpoints are touching. Disjoint arrangements are not considered. Most of the other standard topological relations are of little interest here. Because the pieces are solid, one piece cannot contain another, nor can their boundaries intersect. Pieces can overlap, but this only happens when one piece is stacked on

[2] Translation, scaling, and rotation in particular.

[3] More generally, we can insist that for any edge e_i defined by vertices v_i and v_{i+1} the solid area should be to the right of the vector from v_i to v_{i+1}. This allows us to have holes in the puzzle pieces.

top of another, which is not allowed in two-dimensional puzzles.

4. For symmetrical pieces, such as those shown in Fig. 1, pieces may fit together several different ways. Therefore multiple edges on symmetrical pieces may have the same label, so that different arrangements do not need to be defined for different rotations of the piece.

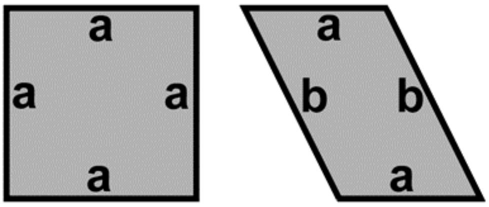

Figure 1. Symmetrical puzzle pieces may have duplicate edge labels.

5. When aligning puzzle pieces with respect to one another, we only need to be concerned with matching up endpoints. If both endpoints of a smaller edge lie on a larger edge, we assume that it does not matter if the piece is off-center[4].

6. The puzzle pieces' relationship to an external coordinate system is irrelevant. Therefore all spatial relations must be expressed relative to the pieces themselves.

Past Work

Researchers working on artificial intelligence (AI) and geographic information systems (GIS) have produced a large body of literature on spatial relationships. In the area of GIS, the focus is on either topological or direction spatial relations [Papad97a]. Although the 2D topological relations are well defined [Egenh89a] and are useful for describing relations among spatial entities in GIS, they include relations that cannot occur among solid objects (e.g. intersects) and do not represent *how* two entities meet. Direction spatial relations describe how entities relate within an external frame of reference (e.g. north or south-east) and are therefore not transformation invariant.

AI researchers typically extend the ideas behind spatial relationships for GIS to support reasoning about those relations [Elger96a], handle boundary uncertainties [Cleme97a] and express changes over time [Gagne96a]. Although these are worthy ideas, they still do not address the problem of how objects meet.

Spatial Relations of Physical Objects

Suppose we have a puzzle made up of an ordered set of n puzzle pieces: $P = [P_1, P_2, \ldots, P_n]$. Now consider two of those puzzle pieces, P_1 and P_2. Let $E_1 = [e1_1, e1_2, \ldots, e1_m]$ be the edges bounding P_1. Each edge $e1_i$ is delimited by vertices $v1_i$ and $v1_j$ from the set $V_1 = [v1_1, v1_2, \ldots, v1_m]$ where $j = (i \bmod m) + 1$. Furthermore, let us assume that a clockwise ordering is imposed on the vertices in V_1. Let P_2 be defined similarly. Then we can define the following relationships between P_1 and P_2.

Definition 1: Two puzzle pieces meet if at least one edge $e1_i$ in E_1 touches at least one edge $e2_j$ in E_2.

Edge $e1_i$ meets edge $e2_j$ if and only if
- one of the endpoints of $e1_i$ ($v1_i$ or $v1_{i+1}$) touches edge $e2_j$, or
- one of the endpoints of $e2j$ ($v2j$ or $v2j_{+1}$) touches edge $e1_i$.

We can show this by considering two possible cases.

Case 1: The edges lie on different lines in space. Then, by definition, these edges will intersect only if the point where these infinite lines intersect lies on both edges. If this intersection point is not any of the edges' endpoints, then the boundaries intersect; they do not meet, and so they are not considered here.

Case 2: The edges lie on the same line in space. If the edges meet at one point, then it must be at the endpoints. If the edges meet at more than one point, then consider the set of points that are touching. Because the edges are finite in length, this set of points must also be finite, with a beginning and an end. And because all of the points lie on a straight line, the bounds of this point set must coincide with an endpoint of one of the edges.

Table 1 illustrates all of the ways that two edges may meet. The **Representation** column illustrates all possible relationships between two edges. Columns labeled $\mathbf{a_1}$, $\mathbf{a_2}$, $\mathbf{b_1}$, $\mathbf{b_2}$ indicate whether or not that particular endpoint touches the other edge. Note that some of the cases are equivalent, depending on which edge one is looking at. Therefore we may impose the following rule: if the only point on an edge touching another edge is an endpoint, then that point must be either a) the first endpoint on the lower numbered puzzle piece, or b) the second endpoint on the higher numbered puzzle

[4] If the pieces need to be more precisely aligned, artificial endpoints may be added to the boundary description.

piece. This ensures a unique description of adjacencies. Given this rule, the **OK** column in Table 1 indicates which relationships satisfy this condition. This set of 12 possible relationships is complete, in that for any two edges that meet, there is a unique relationship that represents it.

OK	a_1	a_2	b_1	b_2	Representation
	0	0	0	0	disjoint
X	0	0	0	1	
	0	0	1	0	
X	0	0	1	1	
	0	1	0	0	
X	0	1	0	1	
	0	1	1	0	
X	0	1	1	1	
X	1	0	0	0	
X	1	0	0	1	
X	1	0	1	0	
X	1	0	1	1	
X	1	1	0	0	
X	1	1	0	1	
X	1	1	1	0	
X	1	1	1	1	

Table 1. For any pair of polygons that meet at their edges [a_1, a_2] and [b_1, b_2], how these edges meet may be expressed in terms of whether their endpoints touch the other edge.

Because we are dealing with the real world here, it is also important to consider some degree of fuzziness in these relations. Therefore I use a broad boundary definition for the vertices [Cleme97a]. This is easily implemented by considering a distance ε when examining the proximity of points to edges.

Definition 2: Assuming that puzzle piece P_1 precedes P_2 in the ordering, edge $e1_i$ touches edge $e2_j$ if either a) vertex $v1_i$ is within distance ε of edge $e2_j$, or b) vertex $v2_{j+1}$ is within distance ε of edge $e1_i$.

ARRANGEMENTS OF PHYSICAL OBJECTS

Given the spatial relations defined above, we may represent a puzzle arrangement as a graph with one node per puzzle piece. For each pair of puzzle pieces that meet, an arc connects the nodes. This arc is labeled with the edges that meet and a code (i.e. a decimal number representing the binary values in columns $\mathbf{a_1}$, $\mathbf{a_2}$, $\mathbf{b_1}$, $\mathbf{b_2}$) indicating how those edges meet. Fig. 2 shows a sample puzzle arrangement and its graph. We can now use this representation to define a solution.

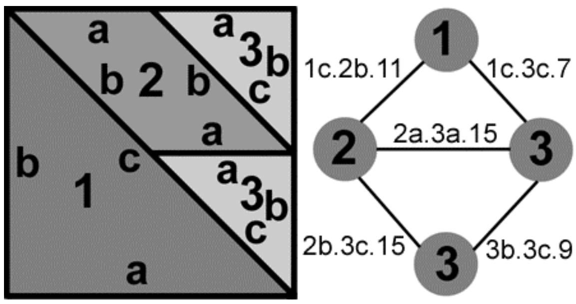

Figure 2. A puzzle solution (a) may be represented by a graph with labeled arcs (b).

Definition 3: A solution is a graph that represents all of the meeting edges in the desired puzzle arrangement. A partial solution is a graph that can be made into a solution by simply adding arcs.

It can be shown that the resulting graph is always unique in that a) an arrangement of puzzle pieces has only one graph representation, and b) a graph represents only one possible arrangement of the puzzle pieces.

Representing the Graph

Each arc in the graph may be uniquely represented by a substring of the form:

$P_i.e_i.P_j.e_j.code$

where P_i and P_j are puzzle pieces (i <= j), e_i is the edge on P_i that touches P_j (e_i <= e_j if i = j), e_j is the edge on P_j that touches P_i, and *code* is a code from Table 1 indicating how the edges meet.

After they are generated, these substrings may be sorted in alphabetical order and concatenated together, separated by a ':' character. For example, the following string represents the graph in Fig. 2:

1.c.2.b.11:1.c.3.c.7:2.a.3.a.15: 2.b.3.c.15:3.b.3.c.9

Because the individual substrings are unique, and because an alphabetical sorting is unique, the resulting string is a unique representation of the graph, and therefore of the arrangement itself.

Manipulating the Graph

For a graph with n arcs, it takes O(n lg n) steps to generate the substrings, sort them, and generate a full string representation. If puzzle pieces are moved one at a time, it takes O(lg n) time to add an arc to the graph (and a substring to the sorted list) and O(lg n) time to delete an arc.

The current puzzle arrangement is a solution if the string representation of the graph is identical to the string representation of the solution. This may be checked in O(n) time. The current puzzle arrangement is a partial solution if every string representing an arc in the graph is a substring of the solution string. If examining the sorted list of solution substrings, this comparison takes O(n lg n) time.

THE TANGRAM

To test these ideas out, I have chosen to implement a smart objects interface for an old Chinese puzzle known as the Tangram. Five triangles, one square, and one parallelogram, precisely cut from a large square, make up the pieces of this puzzle as shown in Fig. 3. Although one may choose to reconstruct literally hundreds of different shapes with the Tangram pieces, the first (and most important) challenge is to reconstruct the square from the pieces. In solving this initial problem, one may discover underlying geometry principles.

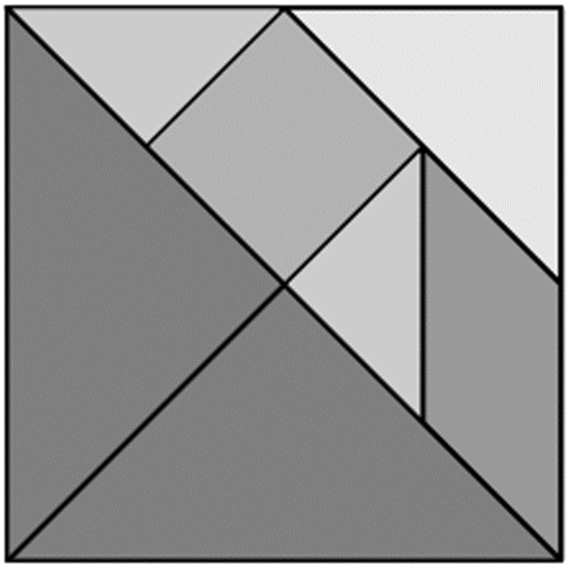

Figure 3. Tangram puzzle

Detecting Puzzle Pieces

For reasons described in [Scarl98a], I have chosen to use computer vision techniques to track the smart objects. Like Underkoffler [Under98a], I use a color QuickCam with an adjacent light source to capture images of the current puzzle arrangement on a table. A tinted reflective surface on the puzzle pieces helps to identify the individual pieces, and makes it easy to threshold out everything else in the scene.

The tracking module uses standard segmentation and feature extraction techniques to find individual puzzle pieces in the image. By comparing images over time, it is able to focus on those areas where the puzzle arrangement has changed. The tracking module returns information about the position and orientation of puzzle pieces. Missing or partially obscured pieces are also noted.

Determining Appropriate Responses

Given the output of the tracking module, the program then updates the internal representation of the puzzle arrangement and "decides" what to do. Following are some of the cases that arise, and the responses that they elicit.

- The current arrangement is the solution. Respond with a congratulatory message, and offer to explain what the students should have learned about geometry by playing with the Tangram.

- The current arrangement is a partial solution. Offer encouragement.

- Pieces are being arranged incorrectly, or there is a long pause in the action. Offer to give the players a hint.

- A piece is either missing or partially obscured, and has been for some time[5]. Warn the students that this is not allowed, and offer to review the rules.

Providing User Feedback

Multimedia feedback provides the unobtrusive guidance that this application calls for. We developed the graphical user interface – and many of the system responses to user actions – using Macromedia Director. In general, text and/or audio feedback is used to comment on the students' progress and offer help. Although one of the students must click a button to actually get that help, this is the extent of their direct interaction with the computer.

For example, when students first approach the puzzle they are greeted by an opening screen that offers to explain the objective and the rules of the puzzle. If a student clicks the button, an animation synchronized with text and a voice-over provides that explanation. This explanation then remains available to the students throughout the game. As another example, the interface offers to give students a hint when they appear to be "stuck" or are not making progress toward the solution. Again, students must click a button to actually see the hint.

CONCLUSIONS

In this paper I have introduced an extended set of topological spatial relations for smart objects in a 2D space, and described a strategy for using these relations to depict arrangements of a physical puzzle. This work is unique in that it expresses information about *how* two solid objects meet. I have also shown how to represent an arrangement internally (in the computer), and discussed how these techniques were incorporated in a smart objects interface for a real puzzle: the Tangram.

Future work will include developing an interface that will allow others to build smart object interfaces for other 2D puzzles, using topological relations to describe puzzle solutions. I also plan to extend these ideas to the third dimension.

[5] Pieces may disappear or cover one another briefly when in transition. Therefore the program will only issue a warning if a sufficient period of time passes and the piece does not reappear.

ACKNOWLEDGEMENTS

I first wish to thank Bill Wilhelms and Tony Scarlatos for initially suggesting the problem of smart objects, and for introducing me to the Goudreau Museum. I thank Beth Deaner, the director of the Goudreau Museum, and her staff for being so helpful and supportive throughout this project. I am grateful to Brian Smith, John Underkoffler, Brygg Ullmer, Joshua Smith, and Richard Borovoy at the MIT Media Labs for showing me their own innovative work in user interface design, and brainstorming on possible interfaces for the Goudreau Museum puzzles. I also had a terrific team of students who helped to implement the Tangram puzzle, supported by a CREW grant: Shalva Landy, Yuliya Dushkina and Natalia Grygorsky. Finally, I wish to thank Sheila Castenada, for awarding the CREW grant that supports these students, and the NSF for supporting this work through a POWRE research planning grant.

REFERENCES

[Cleme97a] Clementini, E, Felice, PD: Approximate Topological Relations, *International Journal of Approximate Reasoning*, vol. 16, pp. 173-204, 1997.

[Egenh89a] Egenhofer, M: *Spatial Query Languages*, Ph.D. dissertation, 1989.

[Elger96a] El-Geresy, BA, Abdelmoty, AI: Order in Space: A General Formalism for Spatial Reasoning, *Proceedings of 8th IEEE International Conference on Tools with Artificial Intelligence*, pp. 183-191, 1996.

[Gagne96a] Gagne, D, Trudel, A: A Topological Transition Based Logic for the Qualitative Motion of Objects, *Proceedings of 3rd International Workshop on Temporal Representation and Reasoning*, pp. 176-181, 1996.

[Ishii97a] Ishii, H, Ullmer, B: Tangible Bits: Towards Seamless Interfaces Between People, Bits and Atoms, *Proceedings of CHI '97*, pp. 234-241, 1997.

[Papad97a] Papadia, D, Theodoridis, Y: Spatial Relations, Minimum Bounding Rectangles, and Spatial Data Structures, *Int. J. Geographical Information Science*, vol. 11, no. 2, pp. 111-138, 1997.

[Resni96a] Resnick, M, Martin, F, Sargent, R, Silverman, B: Programmable Bricks: Toys to Think With, *IBM Systems Journal*, vol. 3, nos. 3&4, pp. 443-452, 1996.

[Scarl98a] Scarlatos, LL: Tracking Puzzle Pieces for a Smart Objects Interface, Brooklyn College Computer Science Technical Report 1-98, 1998.

[Smith98a] Smith, J, White, T, Dodge, C, Allport, D, Paradiso, J, Gershenfeld, N: Electric Field Sensing for Graphical Interfaces, to appear in *IEEE Computer Graphics and Applications*, May 1998.

[Under98a] Underkoffler, J, Ishii, H: Illuminating Light: An Optical Design Tool with a Luminous-Tangible Interface, *Proceedings of CHI '98*, pp. 542-549, 1998.

[Verpl96a] Verplaetse, C: Intertial Proprioceptive Ddevices: Self Motion-Sensing Toys and Tools, *IBM Systems Journal*, vol. 35, nos. 3&4, pp. 639-651, 1996.