Crazy-Cuts: From Theory to App

Yotam Elor, Doron Shaked, and Alfred M. Bruckstein

This column is a place for those bits of contagious mathematics that travel from person to person in the community, because they are so elegant, surprising, or appealing that one has an urge to pass them on.

Contributions are most welcome.

➢ Please send all submissions to the Mathematical Entertainments Editor, **Ravi Vakil,** Stanford University, Department of Mathematics, Bldg, 380, Stanford, CA 94305-2125, USA
e-mail: vakil@math.stanford.edu

Crazy-Cut puzzles are fascinating and often quite challenging: Given a planar shape (as for example depicted in Figure 1a), find a cutting curve that divides the shape into two parts, identical up to Euclidean transformations (rotations and translations). The solution, as seen in Figure 1b, is not trivial to find. Several further Crazy-Cut dissection puzzles are shown in Figures 2 and 12. Apparently, such puzzles were invented by Henry Dudeney (see Eriksson [1]) about one hundred years ago, but it was Martin Gardner who popularized them in the 1970s in his *Scientific American* column [2] and in his books [3]. Gardner also posed the challenge of finding a formal algorithm for solving Crazy-Cut puzzles. In this article, we describe some recent work on algorithms for solving Crazy-Cut challenges that led to the design of a puzzle game for smart-phones and pad computers.

Eriksson was the first to propose an algorithm for solving polygonal Crazy-Cut challenges [1]. In Eriksson's algorithm, two points on the perimeter of the shape are selected, and, starting from these points, two congruent paths are constructed, one of them (the "master") leads by following the boundary of the shape, whereas the second path (the "slave") is led by the congruence constraint and is allowed to cross the shape. If the two starting points happen to be in "correspondence" when there is a solution to the challenge, the "slave" path will split the shape into two identical pieces; see an example in Figure 3. Eriksson proved that by checking all possible pairs of points, one will find the correct cut if such a cut exists or will determine that the shape cannot be cut into two identical parts. Following Eriksson's work, Rote, with co-workers, improved and further elaborated upon Eriksson's algorithm [4, 5]. El-Khechen *et al.* then proposed an algorithm for a variant of the problem in which the two pieces are required to be mirror congruent, that is, identical up to reflection, rotation, and translation [6].

A previous work by the second and third authors of this article reconsidered Crazy-Cut puzzles from a different point of view. Quoting [7]: "We first analyze the inverse problem of assembling a planar shape from two identical shapes that have partially matching boundaries. This problem may be regarded as solving a simple jigsaw puzzle of two pieces (with no drawings on them)." Then the self-docking analysis readily provides a Crazy-Cut algorithm, and more importantly for us here, the insights also provide the mathematical basis required to design Crazy-Cut riddles systematically.

In this work, we improve on our previous analysis, thereby enabling a simpler Crazy-Cut algorithm. Furthermore, based on the improved analysis, a formal method to design Crazy-Cut riddles is proposed.

Self-Docking and the Grammar of Crazy-Cuts

In this section we review the analysis of [7] and improve upon it. A simple planar shape (with no holes) may be represented

 © 2012 Springer Science+Business Media, LLC
DOI 10.1007/s00283-012-9281-4

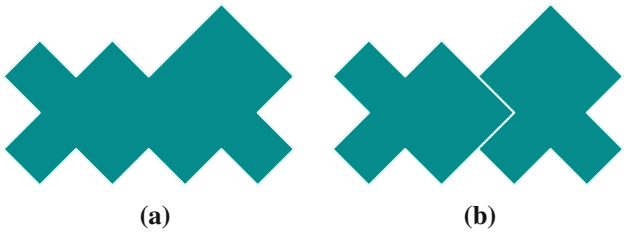


Figure 1. (a) The puzzle, (b) the "crazy cut" into two identical parts.

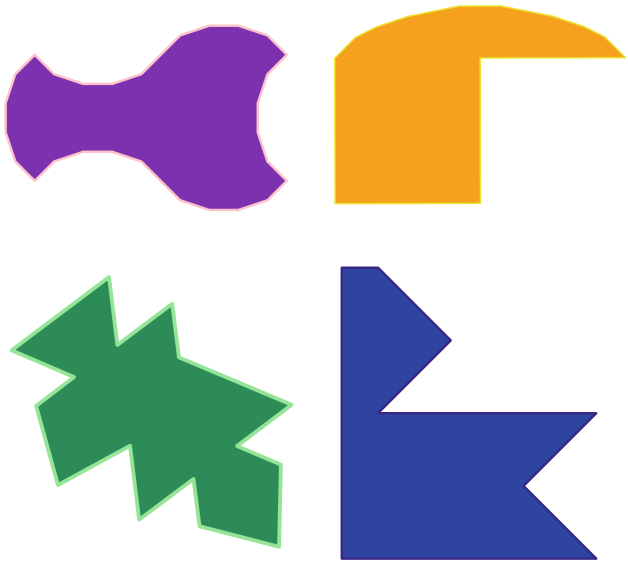


Figure 2. Some further Crazy-Cut challenges. The solutions can be found in Figure 13.

by the closed planar curve of its boundary. The curve is described by a function $k(s)$, with the value of $k(s)$ being the curvature of the boundary at point s for any $s \in [0, L]$, L being

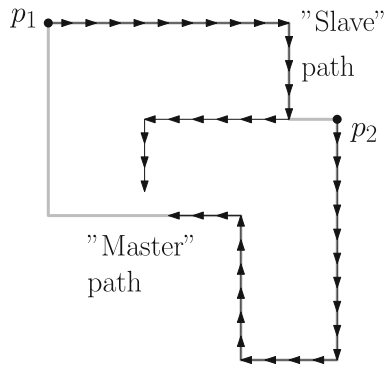


Figure 3. Illustration of Eriksson's algorithm.

the total length of the boundary, and $s = 0$ being an arbitrarily selected starting point. An example of a smooth boundary description can be found in Figure 4a. If the boundary is nonsmooth, we can define $k(s)$ as having δ-function components describing sharp angles at breakpoints; see Figure 4b. It is important to note that such boundary descriptions are Euclidean-invariant, that is, invariant under translations and rotations in the plane. Since the task is finding two pieces that are identical up to Euclidean transformations, using a Euclidean-invariant shape descriptor is both natural and necessary.

Given two shapes S_I and S_{II}, which dock to each other, we may now ask what characterizes the matching portions of their boundaries. If $k^I(s)$ and $k^{II}(s)$ describe the boundaries of the two shapes in a clockwise traversal from arbitrary initial conditions, and the portion between s_A^I and s_B^I on the boundary of S_I matches the portion between s_A^{II} to s_B^{II} on the boundary of S_{II}, we shall have (see Figure 5) that:

$$k^I(s_A^I + s) = -k^{II}(s_B^{II} - s) \qquad s \in [0, s_B^I - s_A^I]$$

Clearly along the common boundary portions of the shapes S_I and S_{II} we have the same traversal rate (arc-length traversal is unit-speed clockwise travel along the boundary!)

AUTHORS

YOTAM ELOR received his B.Sc. in Electrical Engineering and his B.A. in Physics in 2007 from the Technion–Israel Institute of Technology. He is working toward a Ph.D. in the Department of Computer Science, Technion, under the supervision of Professor Alfred M. Bruckstein. He is mainly interested in distributed (swarm or multi-agent) robotics. When he is not studying ants, Yotam can be found at the pool or hanging out with his wife-to-be.

Department of Computer Science
Technion–Israel Institute of Technology
Technion, Haifa 32000
Israel
e-mail: yotame@cs.technion.ac.il

DORON SHAKED received his B.Sc. from the Ben Gurion University in Beer Sheva, Israel, in 1988. He received his M.Sc. and D.Sc. degrees from the Technion–Israel Institute of Technology in 1991 and 1995, respectively. Since then he has been with Hewlett-Packard Laboratories Israel in Haifa, where as a principal researcher he has led multitalented research teams focusing in the areas of printing automation, technologies, image enhancement, and data mining.

HP-labs
Haifa, Israel 32000
e-mail: doron.shaked@hp.com

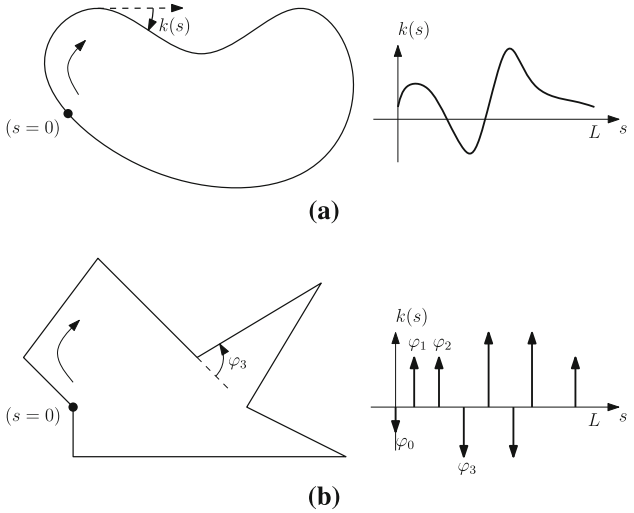


Figure 4. Euclidean-invariant boundary signatures for shape description: (a) Smooth case, and (b) polygonal case.

and the velocity vectors at each point are in opposite directions. Let J_I be the portion of S_I, which is docked to S_{II}, and let J_{II} be the portion of S_{II}, which is docked to S_I. J_I and J_{II} can be described by

$$J_I : k^I(s_A^I + s) = -k^{II}(s_B^{II} - s) \qquad s \in [0, s_B^I - s_A^I]$$
$$J_{II} : k^{II}(s_A^{II} + s) = -k^I(s_B^I - s) \qquad s \in [0, s_B^{II} - s_A^{II}],$$

that is, the matching portions, have the same length and are (up/down) and (left/right) mirror reflections of each other; see Figure 5. For any two curves A and B, we will use the notation $A = \bar{B}$ to imply that A is an (up/down) and (left/right) mirror reflection of B; for example, we have $J_I = \overline{J_{II}}$.

Up to this point, the discussion was for two arbitrary shapes S_I and S_{II}. However, we are interested in matching identical shapes, that is, $S_I \equiv S_{II} \equiv S$. Let J and $\bar{J}$ be the $k(s)$

ALFRED M. BRUCKSTEIN received his B.Sc. and M.Sc. degrees at the Technion, Haifa, in 1976 and 1980, respectively. He then earned a Ph.D. degree in Electrical Engineering from Stanford University, California. Since October 1984 he has been with the Technion, where he holds the Ollendorff Chair in Science. His present research interests are in swarm/ant robotics, image and signal processing, analysis and synthesis, pattern recognition, and various aspects of applied geometry. In his free time, and during boring meetings or lectures, he enjoys drawing and designing logos.

Department of Computer Science
Technion–Israel Institute of Technology
Technion, Haifa 32000, Israel
e-mail: freddy@cs.technion.ac.il

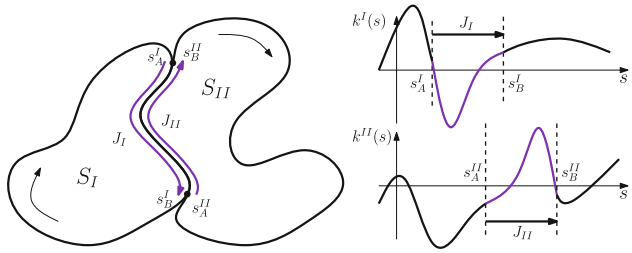
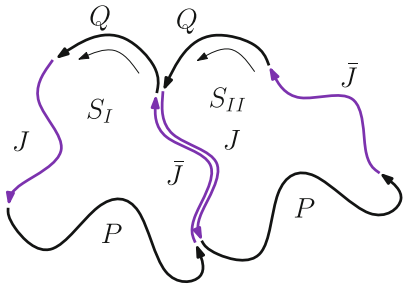


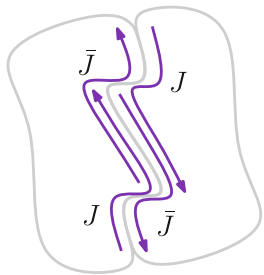
Figure 5. Two shapes docking.

description of the boundary portions in the description of S over which the docking is done. An intriguing result of [7] is that the intervals cannot partially overlap, that is, J and $\bar{J}$ are either disjoint or fully overlap. If the intervals J and $\bar{J}$ are disjoint, it means that there are two distinct portions on the shape's boundary that can be matched, see Figure 6a. On the other hand, when the intervals fully overlap, the region has the property of self reversal, that is, $J = \bar{J}$; see Figure 6c. As an informal proof of the impossibility of partial overlap, consider the partially overlapping J and $\bar{J}$ in Figure 6b. By definition, J is the portion of the boundary over which the docking is done. J of Figure 6b is only a part of that portion, as can be seen in Figure 6c where J then extends to comprise the whole docking portion, and J and $\bar{J}$ fully overlap.

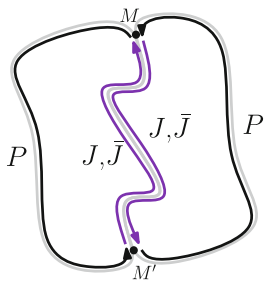
Lemma (Self-Docking Dichotomy) [7]. *A planar shape either "docks" to itself over totally disjoint matching portions of its boundary or over the exact same portion of its boundary, and it cannot possibly have a self-docking that matches over boundary portions that are only partially disjoint.*



(a) J and $\bar{J}$ are disjoint



(b) Impossible case: J and $\bar{J}$ partially overlap



(c) J and $\bar{J}$ fully overlap

Figure 6. Two identical shapes docked.

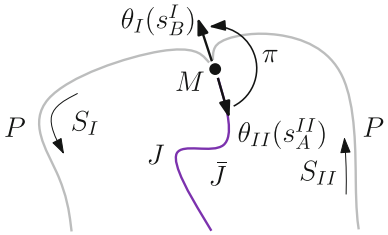


Figure 7. Point M of Figure 6c.

The consequences of these observations are far-reaching indeed: If the docking is over the same portion of the boundaries ($J = \bar{J}$), then the boundary of the concatenated shape will necessarily be the concatenation of two identical boundary curves (see Figure 6c), that is, the boundary of S can be described by PJ and the boundary of the concatenated shape by $P\pi P\pi$ where π is a segment of infinitesimal length over which a turn of 180° occurs.

To observe that the π turn is indeed required, fix any coordinate system and let $\theta_I(s)$ be the direction of the tangent to the boundary of S_I at point s. Similarly, let $\theta_{II}(s)$ be the direction of the tangent to the boundary of S_{II}. Recall that point s_B^J is the point connecting J to P on the boundary of S_I, and s_A^{II} is the point connecting P to $\bar{J}$ on the boundary of S_{II}. Both s_B^J and s_A^{II} correspond to point M of Figures 6c and 7. Because of the congruence relation between J and $\bar{J}$, we see that $\theta_I(s_B^J) = \theta_{II}(s_A^{II}) + 180°$. Hence, in order to connect P to P on the boundary of the concatenated shape, an "on-the-spot" turn of 180° is required. The relation $\theta_I(s_B^J) = \pi + \theta_{II}(s_A^{II})$ holds even when there are δ-functions at the endpoints of P or J, however this case is quite confusing; see Figure 8 and the explanations in the legend.

Note that in the $P\pi P\pi$ case, the cut curve is completely "out of sight," that is, hidden inside the composite shape, and in fact any symmetrical cut from the beginning to the end of P (M to M' in Figure 6c) will yield a possible solution. In case the self-docking is along disjoint portions of the boundary (see Figure 6a), the boundary of S can be described by $PJQJ$ and the boundary of the concatenated shape by $PJQ\pi QJP\pi$.

Hence if a shape can be represented as the docking of two identical jigsaw-puzzle pieces, its boundary must be of the form:

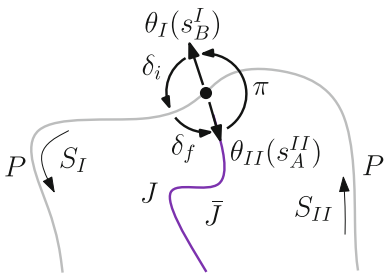


Figure 8. Example of a $P\pi P$ portion from the boundary of a $P\pi$ $P\pi$ composite shape. The curvature of P includes two delta functions: $\delta_i = 105°$ at the beginning of P and $\delta_f = 75°$ at its end. The curvature of the boundary of the composite shape at point M is given by $k(M) = \delta_f + 180° + \delta_i$. In the example, $k(M) = 360° = 0$ so the boundary is smooth at M.

$$\mathbb{S} = \begin{cases} P\pi P\pi & \text{if } S = PJ \text{ with } J = \bar{J} \\ P\bar{J}Q\pi QJP\pi & \text{if } S = P\bar{J}QJ \end{cases}$$

In our previous analysis [7] we allowed any connecting angles between $P\bar{J}Q$ and QJP (or P and P) as long as the resulting boundary is simple and closed. In this work the analysis was sharpened by noting that the connecting angles equal π.

Crazy-Cut Algorithm for Polygons

In this section, it is assumed that $k(s)$ is the boundary of a polygon with n vertices. The analysis described in the previous section yields an efficient algorithm for finding the crazy cut if such a cut exists, or determining that such a cut is not possible. In order to find a crazy cut or to determine impossibility, we have to determine whether the polygon's $k(s)$ has, from some starting point, the structure $P\pi P\pi$ or the structure $P\bar{J}Q\pi QJP\pi$. Testing for the structure $P\pi P\pi$ is the same as testing for a 180° rotational symmetry, which can be done easily. However, testing for the structure $P\bar{J}Q\pi QJP\pi$ is more challenging. The challenge arises from the possibility of several vertices being absent from the boundary descriptor ($k(s)$), that is, vertices that vanish because of the emergence of internal angles of 180°; see an example in Figure 9. The possibility of missing vertices makes the recognition of the composing polylines (P, Q, and J) more complex.

Consider any three polylines P, Q, and J such that $k(s) = P\bar{J}Q\pi QJP\pi$ is a boundary of a polygon. Denote the vertex connecting P to $\bar{J}$ by $v_{P\bar{J}}$, the vertex connecting J to P by v_{JP}, and the vertex connecting P to P by v_{PP}. The algorithm of [7] is based on the following observation: at least two of these three vertices must not vanish. Based on the observation, all portions of $k(s)$ of the form $P\pi P$ can be found by performing a threefold search:

1 . For every selection of two vertices, check whether the clockwise path connecting them is a valid $P\pi P$ segment. It is easy to see that all portions of $k(s)$ of the form $P\pi P$ where the vertices v_{JP} and $v_{P\bar{J}}$ did not vanish will be discovered in this search route. Note that the midpoint vertex, that is, the vertex connecting P to P, might be missing, for example, see the top right $P\pi P$ polyline in Figure 10.
2. For every selection of two vertices, let the clockwise path connecting them be the P polyline, and check whether the continuance of the boundary conforms to πP. Note that

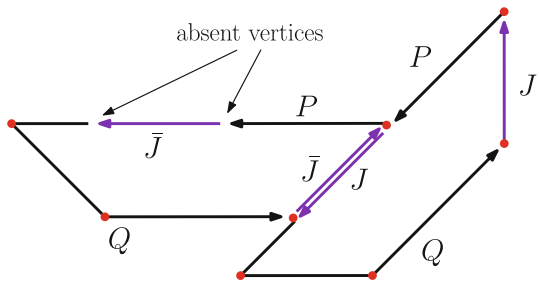


Figure 9. Example of absent vertices. The two vertices connecting P to J and J to Q "vanish," that is, they are missing from the description of the polygon's boundary ($k(s)$).

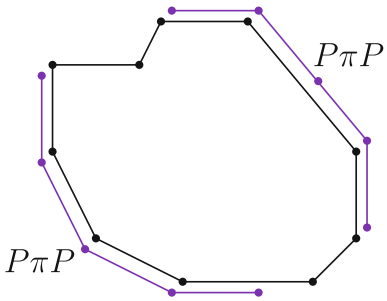


Figure 10. Example of the search for $P\pi P$ polylines. The top right $P\pi P$ polyline is found in the first search route; note the "missing vertex" in the center of the polyline. The bottom left $P\pi P$ polyline is found in the second search route; note the "missing vertex" at the end of the polyline.

the $P\pi P$ polyline does not have to end in a vertex, for example, see the bottom left $P\pi P$ polyline in Figure 10. This search pattern will discover all $P\pi P$ instances where the vertices v_{JP} and v_{PP} did not vanish.

3. For every selection of two vertices, let the counterclockwise path connecting them be the P polyline, and check whether the continuance of the boundary conforms to πP hence discovering all $P\pi P$ instances where the vertices v_{PP} and $v_{P\bar{J}}$ did not vanish.

Following the remainder of the algorithm of [7], for every candidate for the $P\pi P$ portion, we check whether the rest of the boundary conforms to $\bar{J}Q\pi QJ$: We start assembling the candidates for the J and $\bar{J}$ segments by following the boundary on both ends of the $P\pi P$ segment. We match the length of the first edge on both ends and make sure the next turn angles are negatives of each other. We continue until one of the conditions is broken. If the angle condition was met and one of the edge segments is shorter, we stop the $J\bar{J}$ search in the vertex of the short edge and in the middle of the longer edge. If the edge-length condition has been met, but not the turn-angle condition, we stop at both vertices. Notice that this stage cannot fail (we start with a successful vertex condition and may well stop on the first edge). The remaining boundary segment should now correspond to the $Q\pi Q$ segment. The latter is verified by cutting it in midpoint, and checking exactly as we did the $P\pi P$ segment (in stage 1 of the search). Having found three polylines P, Q, and J such that $k(s) = P\bar{J}Q\pi QJP\pi$, we have found the cutting curve!

Note that according to the algorithm of [7], after identifying appropriate P, Q, and J polylines one must verify that the resulting cutting curve does not intersect with the boundary. But if P, Q, and J are such that $k(s) = P\bar{J}Q\pi QJP\pi$, the curve cannot intersect the boundary. Thus the verification is unnecessary.

Recall that n is the number of vertices of the polygonal shape. To summarize the complexity of the algorithm: For every one of the three search routes, there are $O(n^2)$ candidates for a $P\pi P$ segment. For every $P\pi P$ candidate, the traversal of all the other conditions, including checking the $P\pi P$ segment, the J and $\bar{J}$ segments, and the $Q\pi Q$ segment amounts to an $O(n)$ processing time. Hence the total algorithm complexity is $O(n^3)$.

Design of Crazy-Cut Challenges

A popular talk about Crazy-Cuts given by Bruckstein in 2011, and the crowd's positive reaction to the riddles presented in the talk, brought about the idea of designing and programming a Crazy-Cut puzzle game for smartphones and pad computers. The analysis in the first section of this article readily provides the theoretical basis for generating Crazy-Cut challenges. We know that there are two types of Crazy-Cut shapes: $P\pi P\pi$ and $P\bar{J}Q\pi QJP\pi$. However, since $P\pi P\pi$ shapes have infinitely many solutions that are relatively easy to find, it was decided to use solely riddles of the form $P\bar{J}Q\pi QJP\pi$. Therefore, in order to construct a Crazy-Cut shape challenge, three paths P, Q, and J must be defined such that $P\bar{J}QJ$ and $P\bar{J}Q\pi QJP\pi$ are simple and closed. Clearly there are many methods for constructing a set of paths fulfilling these stated requirements. After several attempts, the following method was chosen. Initially, P, Q, and J are set to be straight line segments such that $P\bar{J}QJ$ is a quadrilateral. Then, P, Q, and J are altered while keeping their endpoints fixed. Note that special attention must be paid in order to avoid creating polyline intersections in $P\bar{J}QJ$ and $P\bar{J}Q\pi QJP\pi$. Finally, two copies of the shape are docked to each other to yield the riddle shape. An example of the process is presented in Figure 11.

Another game-play issue to consider is user interaction, in other words, how the user specifies his hypothesis of the location and structure of the cut. Given that we have touch-screen platforms, a straightforward solution is asking the user to draw the hypothesis with his finger. One should carefully determine how precise the hypothesis must be in order to be considered a valid solution. Finger-pointing is very inaccurate, hence in our experiments, when the required precision was too high, the game became frustrating, because the user knew the solution but could not draw it accurately enough. On the other hand, if the required precision was too low, very crude hand-drawn curves were considered valid even for highly complex cuts, thus making the game too easy. While testing iPhone-size devices it became clear that every level of sensitivity is either too low or too high. Therefore, it was decided to use a different mechanism.

Since only polygon riddles were designed, the cut-line hypothesis is limited to polylines. In the selected mechanism, for every riddle, a polyline hypothesis with the appropriate number of vertices is presented to the user. The user can drag and place the vertices with his finger. When the cut hypothesis is similar enough to the solution, the hypothesis is accepted, and the puzzle is considered solved.

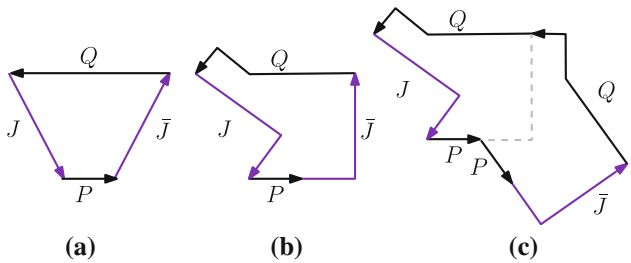


Figure 11. Constructing a Crazy-Cut riddle: (a) The initial quadrilateral, (b) after alteration, and (c) the docking.

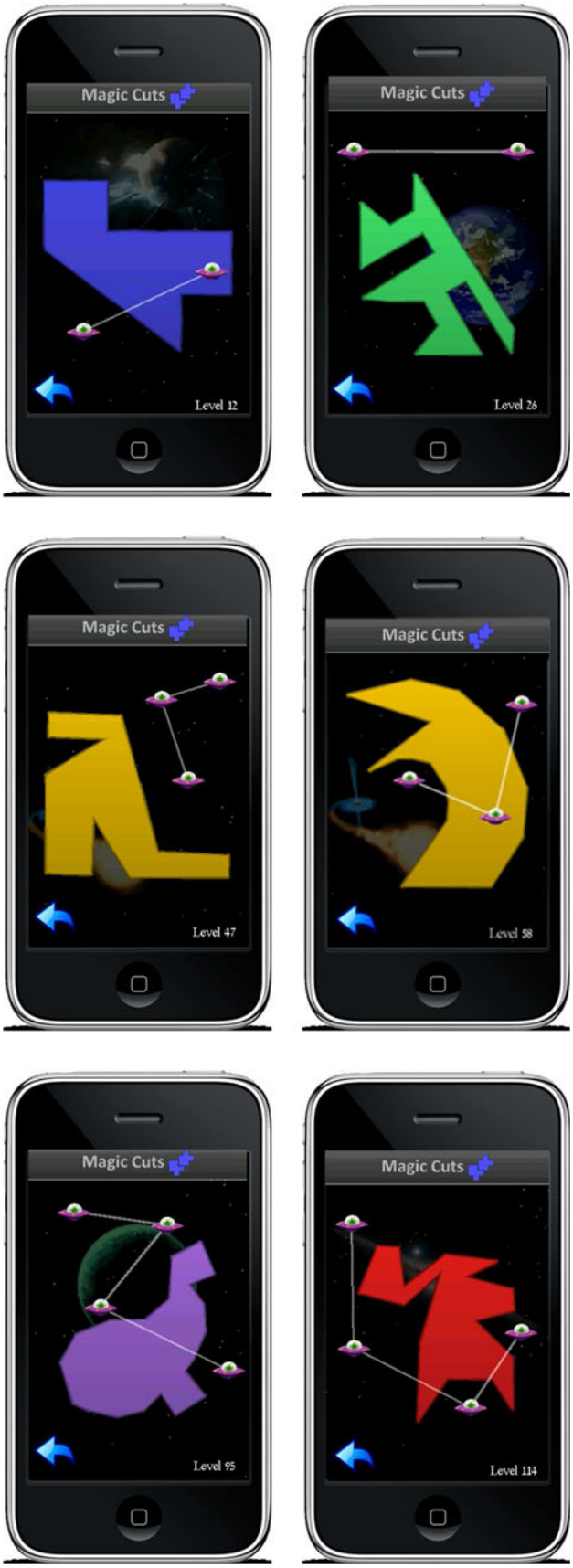


Figure 12. Some Crazy-Cut challenges taken from "Magic-Cuts".

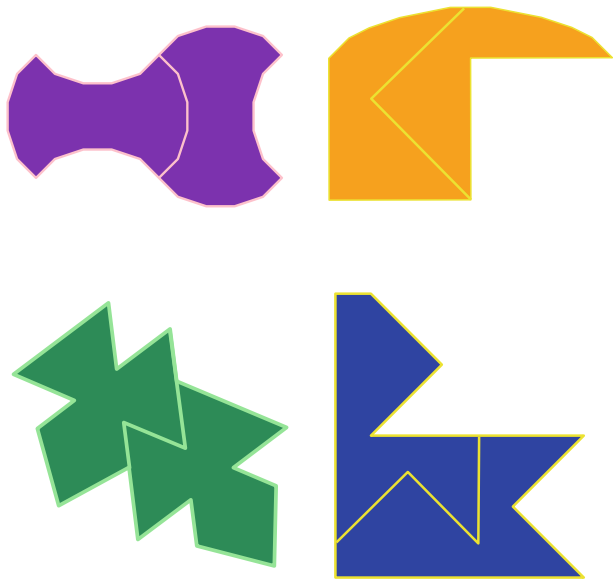


Figure 13. Solutions to the challenges presented in Figure 2.

A space theme was chosen for the "skin" of the game. The cut vertices are designated by tiny flying saucers, and the polyline itself by straight laser beams between the flying saucers. For more Crazy-Cut challenges from the game, see Figure 12. The game is freely available for Android and iOS platforms under the name "Magic-Cuts."

REFERENCES

[1] K. Eriksson. Splitting a polygon into two congruent pieces. *The American Mathematical Monthly*, 103(5):393–400, 1996.

[2] M. Gardner. *Mathematical Games, Scientific American*, volume 237, pages 132–137, 1977.

[3] M. Gardner. *My Best Mathematical and Logic Puzzles*. Math & Logic Puzzles. Dover, 1994.

[4] G. Rote. Some thoughts about decomposition of a polygon into two congruent pieces. unfinished draft, http://page.mi.fu-berlin.de/rote/Papers/pdf/Decomposition+of+a+polytope+into+two+congruent+pieces.pdf, 1997.

[5] T. Fevens, J. Iacono, D. El-Khechen, and G. Rote. Partitioning a polygon into two congruent pieces. In *Kyoto International Conference on Computational Geometry and Graph Theory, Kyoto CGGT2007, Kyoto, Japan*, 2007.

[6] T. Fevens, D. El-Khechen, and J. Iacono. Partitioning a polygon into two mirror congruent pieces. In *Proceedings of the 20th Canadian Conference on Computational Geometry, CCCG*, Montreal, Quebec, August 13–15 2008.

[7] A. Bruckstein and D. Shaked. Crazy cuts: Dissecting planar shapes into two identical parts. In Martin Ralph Hancock, Edwin and Malcolm Sabin, editors, *Mathematics of Surfaces XIII*, volume 5654 of *Lecture Notes in Computer Science*, pages 75–89. Springer: Berlin, Heidelberg, 2009.