

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/228877279>

A Novel Algorithm to Verify the Solution of Geometric Puzzle Games

Article · January 2009

DOI: 10.1109/SBGAMES.2009.10

CITATIONS

0

READS

140

8 authors, including:



Rafael Alves

Federal University of Rio de Janeiro

1 PUBLICATION 0 CITATIONS

[SEE PROFILE](#)



Esteban Clua

Universidade Federal Fluminense

217 PUBLICATIONS 734 CITATIONS

[SEE PROFILE](#)



Erick Passos

Instituto Federal de Educação, Ciência e Tecnologia do Piauí (IFPI)

23 PUBLICATIONS 150 CITATIONS

[SEE PROFILE](#)

Some of the authors of this publication are also working on these related projects:



Jecripe [View project](#)



IMUFF - Version Control for Images [View project](#)

A Novel Algorithm to Verify the Solution of Geometric Puzzle Games

Manoel Siqueira Júnior Rafael Alves Esteban Clua Erick Passos
 Clayton da Silva Anselmo Montenegro Júlio Cesar Oliveira*

UFF, Media Lab, Brazil

*UFRJ Co., Brazil

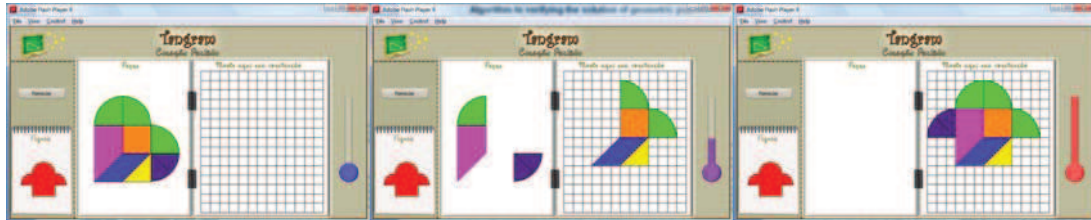


Figure 1: Example of a Tangram game, showing the algorithm verification of the player's progress with a thermometer.

Abstract

In this paper, we present a novel algorithm to solve the problem of correctly verifying the solution for geometric puzzles. When compared with others, this approach covers a satisfactory amount of cases. The method comprises the use of sixteen possible relations between polygon edges, which are classified to eliminate those that are not necessarily part of the final figure. This method provides for a precise verification of an arranged set of polygons that must form the same image as the desired solution, without the need of extra meta-data. Only the vertexes themselves (also the edge concavity and center position, when circumference arcs are present) are used the algorithm.

Keywords: geometric puzzles, polygon contour, Tangram

Authors' contact:

manoeljr88@gmail.com
 rafamachadoalves@yahoo.com.br
 {creis,anselmo,epassos,esteban}@ic.uff.br
 juliooceano@gmail.com

1. Introduction

Many games are related to puzzles and geometric piece arrangements. For this kind of games, it is necessary that algorithms validate the correctness of the arrangement and also give a feedback to the player, telling how close he is from the correct solution. Figure 1 shows the player's progress in a geometric jigsaw puzzle by the algorithm proposed in this work. All geometric jigsaw puzzles presented in this paper are based on the games proposed in Kaleff et al. [2002].

The algorithm that will be discussed in this paper is an alternative approach to the problem of identification if a determinate geometric jigsaw puzzle assembly represents the intended figure. This work is an extension of the algorithm presented by Scarlatos [1999], which is not suitable for jigsaw puzzle that have more than one possibility for a correct assembly.

This work is also an extension of a preliminary work made by Siqueira et al. [2008].

The proposed algorithm use a different approach than that presented by Scarlatos [1999], in order to verify different arrangements of parts that lead to the same figure, considering just the outline formed by the pieces together and not how they are arranged within the contour. The presented approach also analyses that the final assembly has no holes.

When compared with pixel by pixel approaches, the proposed algorithm has the advantage of being easier to get the partial solution and to solve the problem with parts that have any rotations.

It is important to note that the pieces of the geometric jigsaw puzzle and any assembly composed with them can be represented by a polygon, as well as the solution figure (which indicates the solution to be achieved in the game). For this work, the possible outline of the polygons can be defined by edges composed of circular arcs and/or straight segments. If a puzzle is composed of figures that are not polygons, a bounding polygon will be required for algorithm usage.

The remainder of this paper is organized as follows: section 2 presents some works related to the design and implementation of methods for the verification of relations between edges on polygonal figures. Section 3 carries out an analysis of the problems to be handled, while section 4 explains the proposed algorithm. Nevertheless, section 5 talks about some special cases where the algorithm does not correctly detect the contour of the assembly made by the player. Finally, the conclusion and future work associated with the proposal are presented in section 6.

2. Related Work

In this section, we attempt to compare the proposed technique with other implementations of mathematical jigsaw puzzle, specifically in relation to recognition of the solution, and works about generic methods for the

recognition of figures through the classification of relations between edges.

There are few games of geometric jigsaw puzzle that make the recognition of solution, in other words the games indicate to the player if he succeeded or not in completing the assembly. Most implementations found, for example, shown by Jacob [2002], Martins [2002] and Lankin [2001], only allow a passive visual comparison by the player.

The Tangram (geometric jigsaw puzzle) presented by Ztor [2005] makes the recognition of a solution, but this approach does not allow assembly with rotating alternatives. Moreover, the previews work does not make the partial recognition of a solution, an important feature for showing the player how far or close he is from the final solution. The solution presented in this paper allows both arbitrary rotation and the estimation of partial solution, besides allowing the use of pieces with edges composed of circumference arcs.

In Flashkit [1999], it is possible to find a great repository of source code, tutorials and other materials for developing games in Flash, including many examples of Tangrams, as those available in Lankin [2001], Jacob [2002], Martins [2002], Kunovi [2001] and Texas [2000]. However, none of these examples presents algorithms for verification of solution, serving only as a reference for implementation of the basic mechanics.

In Scarlatos [1999], it is shown a method to recognize a figure formed by several polygons juxtaposed through exact relations formed by their edges. This solution, however, supports only edges composed of straight segments and only recognize individually each possible solution. A figure that can be formed by more than one combination of parts may have multiple representations, making it impossible to use for puzzles that may have thousands of possible solutions even for simple instances. Although not satisfactory, the work referenced served as the basis for the method developed.

A situation in which different arrangements of pieces form the same figure is illustrated in Figure 2. In this example, the square figure formed by the union of polygons 2 and 3 may be attached to the polygon 1 in different rotations. While the method referenced by Scarlatos [1999] generates two different representations, the mechanism that is proposed in this paper recognizes both figures as equals, and therefore is more appropriate to the problem of verification of solutions of geometric jigsaw puzzle.

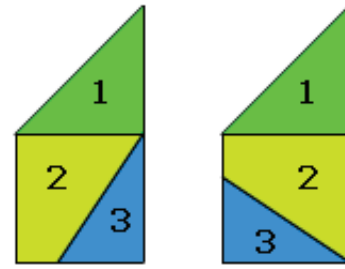


Figure 2: Two different arrangements that represent the same figure.

The authors did not found any other works that, at least using polygons with edges composed of straight segments, provide a general method for the recognition of solution of geometric jigsaw puzzle from the comparison with the figure solution.

3. Problem Considerations

For the arrangement of the pieces during the game to be verified, relations between the edges of different polygons, proposed in the work of Scarlatos (1999), are considered to assist in identifying the contours of the current figure mounted by the player. Besides the basic relations between edges, adjustments are made so that the final result of this assembly is adequately compared with the figure solution.

Both the representation of the solution and the assembly made by the player are composed of circular lists of edges, which describe appropriately the final design of each. It is important to note that the polygons formed during the game may contain holes and it does not interfere in the analysis of the solution proposed by the player.

For a correct usage of the algorithm it is necessary that:

- There is no overlap between the pieces of the geometric jigsaw puzzle, since the elimination of edges can not happen in some cases it was necessary
- Each of the pieces and the solution figure has to be designed either in a clockwise or anti-clockwise direction; in other words, all must be designed in the same direction at the definition of its vertices constituents. Thus, comparison of the assembly by the player with the correct solution is made with the same sense of drawing
- It must be kept in memory the coordinates of the vertices that form the polygons, and if the representation of edges is composed of circumference arcs, it is also necessary the center of the circumference arc and its concavity (concave or convex). Such as it is illustrated on Figure 3, the vertices A, B, C and D; the center and the concavity of circumference arc BC and DA should be stored in memory

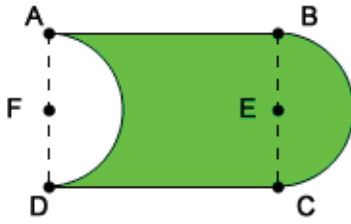


Figure 3: Example of a polygon with edges composed of straight segments and circumference arcs.

- The solution polygon can not have holes, since it is not taken into consideration by the algorithm the position in the figure that the hole is located. The presence of holes will only be checked by the player itself. A solution polygon with a hole is shown in Figure 4

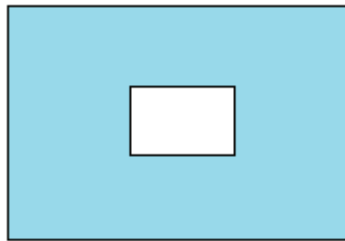


Figure 4: The solution can not contain holes, such as it is represented by the white rectangle.

- The game must be composed only of closed polygons without holes, and formed only by borders consisting of straight segments and circumference arcs, not supporting other types of curves

Despite these restrictions, infinity of forms of pieces is permitted so that the proposed algorithm properly analyzes the solution set by the player during the assembly of geometric jigsaw puzzle.

4. Heuristic for Verification

This heuristic for the verification of the solution of a geometric jigsaw puzzle uses the relations described in Scarlatos [1999]. Thus, it is added the verification of edges composed of circumference arcs, besides interpreting relations separately, removing the edges that are not part of the contour of the result obtained by the relations, so that only the edges of the contour remain. The edges of the contours are then rearranged so that the figure is represented in a unique way.

4.1 Relations

Only the relations between edges of different polygons will be considered. What characterizes each relation between a pair of edges is the verification of the vertices of each edge, checking if it belongs or not to the other analyzed edge.

Initially, it is supposed that only edges composed of straight segments are analyzed and that one of the edges is formed by the points a_1 and a_2 and the other by b_1 and b_2 . For each of the four corners, if it belongs to the other edge, it will be assigned value 1, otherwise, value 0. Concatenated values of a_1 , a_2 , b_1 and b_2 , in that order, has the representation of the relationship between two edges expressed in a binary system, converted into a decimal system.

Table 1 (attached at end of paper) shows the possible relations between edges, where the polygons were drawn in a clockwise direction. The arrows of one vertex to another indicate the direction of the edge they represent. The relations are presented in decimal form and are located just below the figure to which they belong.

To deal with edges composed of arcs, in addition to the vertices, it is also necessary the center of the circumference and its concavity (concave or convex) information. Therefore, different data structures are generated to be associated with two types of edges (straight segments and circumference arcs). Thus, relations are established for identifying the type of edge that each one has.

There are three groups of relations: those that split an edge, those that are null and those that totally or partially eliminate the edges. When it includes edges which are circumference arcs in the verification of relations, the interpretation obtained can be more comprehensive. Therefore, to simplify the algorithm, the relations are considered void, in cases where there are not explained in subsections 4.1.1, 4.1.2 and 4.1.3, without even checking the number of the result.

It is important to note that a relation may belong to different groups, since as it comes from the comparison between two edges. This can lead to an individually and differently treatment. For instance, the relation that has the number 12 in Table 1 has one of the edges eliminated completely and the other splitted.

In all cases presented in subsections 4.1.1, 4.1.2 and 4.1.3, the pair of edges can only be analyzed as follows:

- Both edges are straight segments
- Both edges are circumference arcs with equals radius and global center coordinate but with different concavities

Table 2 (attached at end of article) shows the possible relations between edges represented by circumference arcs in the described case immediately above, in which the polygons were drawn clockwise. The arrows from one vertex to another indicate the direction of the edge they represent and the point "c" indicates the center of both edges. The relations are presented in decimal form and are located just below

the figure to which they relate. As can be noted, not all relations are possible.

4.1.1 Split Relation

This type of relations occurs either when only one vertex is tangential to the edge examined or when two vertices that belong to one edge are contained in another edge but not touching in the extreme points of this. These cases are illustrated in Table 1: the relation 1 for the first case and the relation 12 for the second case. Table 2 illustrates the second case using the relation 12.

Based on the points a_1 , a_2 , b_1 and b_2 , below a description of the relations in that group can be:

- Relation 1 ($a_1 = 0$; $a_2 = 0$; $b_1 = 0$; $b_2 = 1$): edge formed by the points a_1 and a_2 is subdivided to point b_2
- Relation 2 ($a_1 = 0$; $a_2 = 0$; $b_1 = 1$; $b_2 = 0$): edge formed by the points a_1 and a_2 is subdivided to point b_1
- Relation 3 ($a_1 = 0$; $a_2 = 0$; $b_1 = 1$; $b_2 = 1$): subdivides the edge formed by the points a_1 and a_2 in two: one whose extreme points are a_1 and b_2 and the other, b_1 and a_2 , in this order, since that way the direction of edges is preserved
- Relation 4 ($a_1 = 0$; $a_2 = 1$; $b_1 = 0$; $b_2 = 0$): edge formed by the points b_1 e b_2 is subdivided in point a_2
- Relation 8 ($a_1 = 1$; $a_2 = 0$; $b_1 = 0$; $b_2 = 0$): formed by the points b_1 e b_2 is subdivided in point a_1
- Relation 12 ($a_1 = 1$; $a_2 = 1$; $b_1 = 0$; $b_2 = 0$): subdivides the edge formed by the points b_1 and b_2 in two: one whose extreme points are b_1 e a_2 and the other, a_1 e b_2 , in this order, because that way the direction of edges is preserved

4.1.2 Null Relation

This type of relation occurs in the verified edge's vertex either when the vertex do not touch the other, or the intersection between these two edges result only in a vertex in common. Both cases are illustrated in Table 1. For example, the relation 0 for the first case and the relation 6 for the second case. Table 2 uses the relation 0 to illustrate the first case.

Similarly to the previous subsection, below there is a description of the relations contained in that group:

- Relation 0 ($a_1 = 0$; $a_2 = 0$; $b_1 = 0$; $b_2 = 0$): as they are disjoint edges, there is nothing to do
- Relation 5 ($a_1 = 0$; $a_2 = 1$; $b_1 = 0$; $b_2 = 1$): as only the points a_2 and b_2 touch each other, there is no need to change the two edges, so there is nothing to do

- Relation 6 ($a_1 = 0$; $a_2 = 1$; $b_1 = 1$; $b_2 = 0$): as only the points a_2 and b_1 touch each other, no need to change the two edges, so there is nothing to do
- Relation 9 ($a_1 = 1$; $a_2 = 0$; $b_1 = 0$; $b_2 = 1$): as only the points a_1 and b_2 touch each other, no need to change the two edges, so there is nothing to do
- Relation 10 ($a_1 = 1$; $a_2 = 0$; $b_1 = 1$; $b_2 = 0$): as only the points a_1 and b_1 touch each other, no need to change the two edges, so there is nothing to do
- Relation 15 ($a_1 = 1$; $a_2 = 1$; $b_1 = 1$; $b_2 = 1$): If the vertex a_1 has the same global coordinate as b_1 , a_2 has the same global coordinate as b_2 and both edges are circumference arcs with same radius and global coordinate center (Figure 5), there is nothing to do

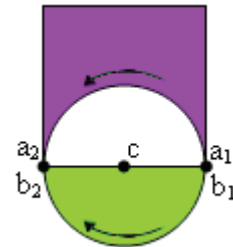


Figure 5: Case where relation 15 is null

4.1.3 Elimination relation

This type of relation occurs when the two vertexes of one edge touches the examined edge or when one of the vertexes of each edge touches another, but these points do not have the same global coordinate. Both cases are also illustrated in Table 1 and Table 2. Relation 3 and 5 are examples for the first and second case, respectively.

Similarly to the previous two subsections, following there is description of the relations contained in that group:

- Relation 3 ($a_1 = 0$; $a_2 = 0$; $b_1 = 1$; $b_2 = 1$): eliminates the edge b_1b_2
- Relation 5 ($a_1 = 0$; $a_2 = 1$; $b_1 = 0$; $b_2 = 1$): if a_2 and b_2 have different coordinates, it is eliminated the parts that touch the two edges connected
- Relation 7 ($a_1 = 0$; $a_2 = 1$; $b_1 = 1$; $b_2 = 1$): it is eliminated the edge b_1b_2 and the part of the edge a_1a_2 that has intersection with the edge b_1b_2
- Relation 10 ($a_1 = 1$; $a_2 = 0$; $b_1 = 1$; $b_2 = 0$): if a_1 e b_1 have different coordinate, it is eliminated the parts that touch the two connected edges
- Relation 11 ($a_1 = 1$; $a_2 = 0$; $b_1 = 1$; $b_2 = 1$): it is eliminated the edge b_1b_2 and the part of the

- edge a_1a_2 that has intersection with the edge b_1b_2 ;
- Relation 12 ($a_1 = 1$; $a_2 = 1$; $b_1 = 0$; $b_2 = 0$): it is eliminated the edge a_1a_2
- Relation 13 ($a_1 = 1$; $a_2 = 1$; $b_1 = 0$; $b_2 = 1$): it is eliminated the edge a_1a_2 and the part of the edge b_1b_2 that has intersection with the edge a_1a_2 ;
- Relation 14 ($a_1 = 1$; $a_2 = 1$; $b_1 = 1$; $b_2 = 0$): it is eliminated the edge a_1a_2 and the part of the edge b_1b_2 that has intersection with the edge a_1a_2 ;
- Relation 15 ($a_1 = 1$; $a_2 = 1$; $b_1 = 1$; $b_2 = 1$): it is eliminated the two edges if they are straight segments. If both edges are circumference arcs with same radius and global coordinate center, but have different concavities they will only be partial or total removed if one of the following conditions are met:
 - The vertex a_1 has the same global coordinate b_2 and b_1 has the same global coordinate a_2 (Table 2, relation 15). In this case, there is a complete elimination of the edges;
 - The vertex a_1 has the same global coordinate b_1 and b_2 has the same global coordinate a_2 (Figure 6). In this case, there is an elimination of the edges parts which is between the vertexes b_2 and a_2 ;

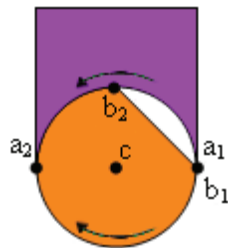


Figure 6: Relation 15 with elimination of the edges which are among the vertexes b_2 and a_2 .

- The vertex a_1 has not the same global coordinate b_1 and b_2 has the same global coordinate a_2 (Figure 7). In this case, there is elimination of the edges parts which is between the vertexes b_1 and a_1 ;

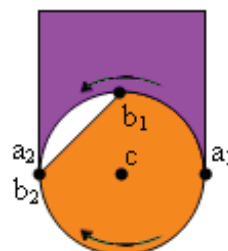


Figure 7: Relation 15 with elimination of the edges which are among the vertexes b_1 and a_1 .

- All vertexes have different global coordinates (Figure 8). In this case, there is elimination of the edges parts which are between the vertexes b_1 and a_1 and between b_2 and a_2 .

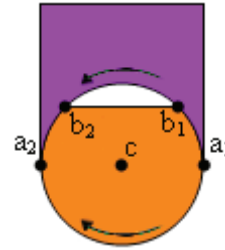


Figure 8: Relation 15 with elimination of the edges which are among the vertexes b_1 and a_1 and between b_2 and a_2 .

4.2 Adjusting Relation Outcomes

Firstly, it's important to know that there is a data structure that represents the list of final figures and each figure is represented by one edge circular list. The final figures are the contours obtained with the assembly of player pieces.

After the relations were applied, contour edges are found, but it's needed to adjust them. Thus one or more final figures are identified. This adjusting is done selecting an edge to be added in initial position of the circular list that represents any of the final figures. After, the next edge to be added in this data structure is selected. The chosen edge is that among those with a global coordinate of initial vertex equal to the end vertex of the last edge, added to the circular list and has the smallest angle with the last edge inserted in the circular list. This is done until a closed polygon is found. During this phase are also unified adjacent straight segments that form a 180° angle between them and adjacent circumference arcs having the same global coordinate center. These operations are repeated until all contours are identified.

With this method, the contours are defined and it is possible to have representations of holes. For this, it is necessary that filled polygons have a direction (clockwise, for example) and holes have the opposite (in the case of Figure 9, counterclockwise).

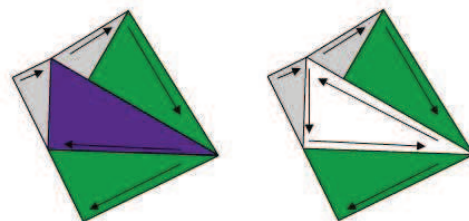


Figure 9: Differences between a filled polygon and a polygon with hole.

To identify the direction of a contour, it is simply necessary to be used the formula of area found in Bourke [1998], which returns positive if the figure is

drawn counterclockwise, and negative if the figure is drawn in another direction.

To verify if a filled polygon represents the solution, there is a circular list in which each element stored is represented by one of the following data structures: one containing the distance between the edge vertexes and the angle between this edge and the next on the list, while the other structure contains this information and the radius and the concavity of the circumference arc. These structures must follow the same order of the circular edge list that represents the solution submitted by the player. The same kind of list is made for solution figure of the game.

Once these lists are ready with the previous structures, the game solution is compared with the player solution. If any filled polygon forms the solution figure, the next step consists on verifying the existence of a hole in this polygon. One strategy suggested for this is presented in solution 2 described by Bourke [1987], which shows how to verify if a point is in a polygon. To identify if a hole belongs to a filled polygon, it is necessary only to apply this solution in all vertexes of the candidate hole.

If the filled polygon is equal to the solution figure and it doesn't have holes, it is possible to conclude that the player set the correct solution.

4.3 Completeness Level

For the calculation of the completeness level, it obtained the minimum between the completeness of the polygon contour assembled by the player that has most proximity to the polygon solution and the filled area of this contour. This measurement of progress is more a motivation of usability than of accuracy, since it allows a progress perception of the player.

5. Special Cases

There is infinity of geometric forms that the verifying heuristic presented in subsection 4 treats correctly. Although there are special cases that the contour isn't identified properly, depending on the manner that the pieces are designed, many of these cases are solved.

A possible special case would be a geometric jigsaw puzzle that has a solution figure illustrated in Figure 10. This solution is drawn in clockwise and composed of edges AB, BC, CE, EB, BF and FA, being AB and BF convexes and centered in point G; BC and EB are convexes and centered in point D.

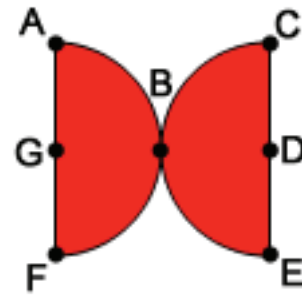


Figure 10: Solution figure of a special case.

It is supposed the available pieces for the solution assembly are two pieces shown in Figure 11. These two pieces were drawn clockwise. One of them is composed of edges AB, BA, being AB convex and centered in point C. The other is composed of edges DF, FD, being FD convex and centered in point E.

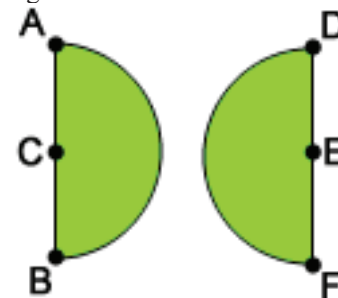


Figure 11: Available pieces for the solution figure assembly.

When the pieces of Figure 11 form the solution figure shown in Figure 10, the edges AB and FD won't be subdivided because the resulting relation between them will be null. Consequently, the contour of the solution figure won't be found.

Meanwhile, this problem can be solved, creating with a differently way the game pieces. Figure 12 shows an alternative of creation pieces that allows the correct assembly of the solution figure. The pieces shown in Figure 12 were drawn clockwise. One of them is composed of edges AB, BC and CA, being AB and BC convexes and centered in point D. The other is composed of edges EG, GH and HE, being GH and HE convexes and centered in point F.

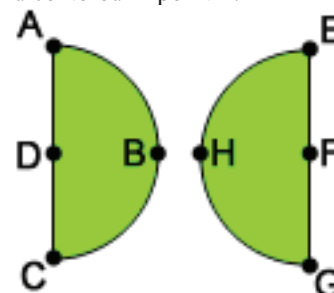


Figure 12: Alternative set of available pieces to assembly the solution figure.

When the pieces of Figure 12 intersect the vertexes B and H, forming the solution figure shown in Figure 10, the figure solution contour will be identified

because the responsible edges for the problem of the null relation of Figure 11 will already be subdivided. In other words, AC was subdivided in AB and BC and GE was subdivided in GH and HE.

Another special case example, similar to previous one, is a geometric puzzle that has the solution figure illustrated in Figure 13. This solution is drawn clockwise and composed of edges AB, BC, CD, DE, EG, GD, DH and HA, being DE and GH convexes and centered in point F.

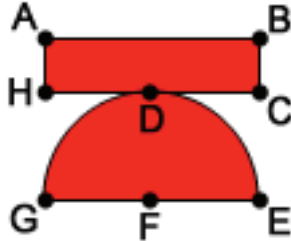


Figure 13: Solution figure of a special case.

Figure 14 shows a possible assembly for the pieces of this geometric puzzle. The pieces shown in this figure were drawn in clockwise. One of them is composed of edges AB, BC, CD and DA. The other is composed of edges EF and FE, being EF convex and centered in point G.

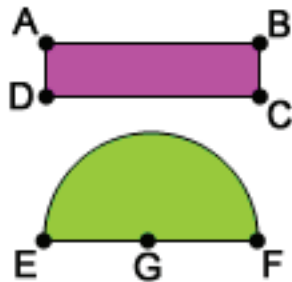


Figure 14: Available pieces for the assembly of the solution figure.

In this situation, the difference with the previous special case is the pair of edges that can originate the contour identification error. This pair is composed of a straight segment and a circumference arc.

Similar to the previous special case, there is a way of creating available pieces on game that is illustrated in Figure 15. Thus the purposed algorithm detects the solution figure contour. The pieces shown in Figure 15 were drawn clockwise. One of them is composed of edges AB, BC, CD, DE, EF and FA. The other is composed of edges GH, HI and IG, being GH and HI convexes and centered in point J.

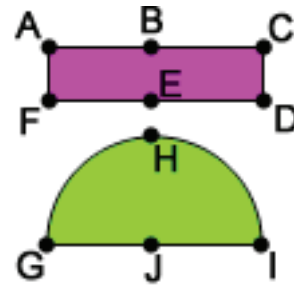


Figure 15: Alternative set of available pieces for the solution figure assembly.

6. Conclusion and Future Works

In this article, it was presented a new approach to the relations between the edges of adjacent polygons. Thus there are other ways of successfully verifying the solution of geometric puzzles other than the ones already established.

This approach comes along to mend faults of other approaches that do not consider only the contour of the solution assembled by the player, but all the arrangement of the pieces. Also the algorithm presented considers edges represented by circumference arcs.

In future, many adaptations must be made in this algorithm, such as:

- Adapting pieces and solutions with edges represented by different kind of curves
- Adding the calculation of intersection between edges to apply division relations of edges in special cases and thus giving more precision to this algorithm
- Including pieces and solutions with holes
- Adapting this algorithm for similar problem in three dimensions

Acknowledgements

This project is founded by the Brazilian ministry of Education and Science and Technology. Special thanks for professor Alexandre Machado, who helped in the English review, and Ana Kaleff, who presented geometric puzzles to us.

References

- LANKIN, A., 2001. *Tangram Implementation*. Available from: http://www.flashkit.com/movies/Games/Full_Game_Source/Tangram-Andrew_L-5966/index.php [Accessed 03 August 2008].
- JACOB, E., 2002. *Tangram Implementation*. Available from: http://www.flashkit.com/movies/Games/Full_Game_Source/Tangram-Eduardo_-8137/index.php [Accessed 03 August 2008].

- FLASHKIT, 1999. Section: *Movies*; Subsection: *Games*. Available from: <http://www.flashkit.com> [Accessed 03 August 2008].
- MARTINS, I. F., 2002. *Tangram Implementation*. Available from: http://www.flashkit.com/movies/Games/Tangram-Ilclio_-6471/index.php [Accessed 03 August 2008].
- KALEFF, A. M.; REI, D.M.; GARCIA S. S. *Quebra-cabeças geométricos e formas planas*. 3 ed. Niteroi: EdUFF, 2002.
- DE KUNOVI, N., 2001. *Tangram Implementation*. Available from: http://www.flashkit.com/movies/Games/Full_Game_Source/Tangram-Nicola_d-3822/index.php [Accessed 03 August 2008].
- JASP, T., 2000. *Tangram Implementation*. Available from: http://www.flashkit.com/movies/Games/Full_Game_Source/Tangram-Texas_Ja-2061/index.php [Accessed 03 August 2008].
- ZTOR, 2005. *Tangram Implementation*. Available from: <http://www.ztor.com/index.php4?ln=&g=game&d=tang> [Accessed 03 August 2008].
- SCARLATOS, L. L., 1999. *Puzzle piece topology: detecting arrangements in smart objects interfaces*.
- BOURKE, P., 1987. *Determining if a point lies on the interior of a polygon*. Available from: <http://local.wasp.uwa.edu.au/~pbourke/geometry/insidepoly/> [Accessed 21 June 2008].
- BOURKE, P., *Determining whether or not a polygon (2D) has its vertices ordered clockwise or counterclockwise*, 1998. Available from: <http://local.wasp.uwa.edu.au/~pbourke/geometry/clockwise/> [Accessed 21 June 2008].
- SIQUEIRA, M. M. J.; MACHADO, R. A.; SANTOS, W. DOS; CARVALHO, C.; SILVA, C.; MONTENEGRO, A.; PASSOS, E.; CLUA, E. *Algoritmo para Verificação da Solução de Quebra-cabeças Geométricos*. In: SBGames, 2008, Belo Horizonte.

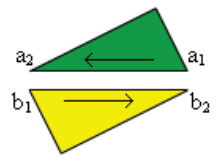
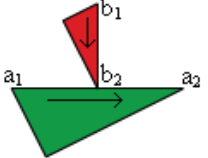
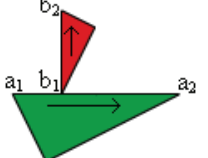
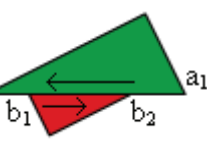
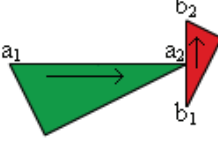
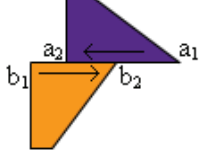
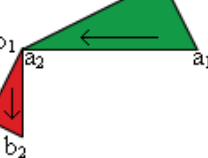
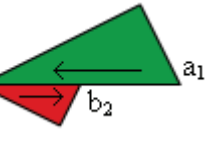
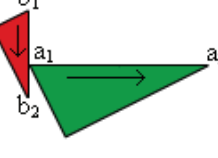
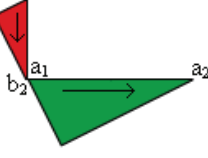
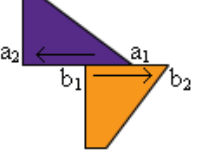
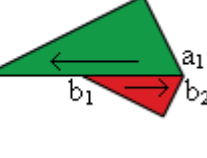
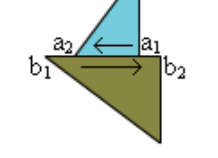
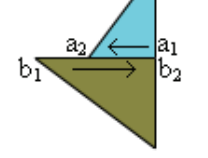
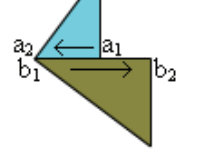
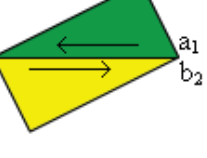
Representation of relations considering only straight segments			
			
			
			
			

Table 1: Possible representations of straight segment relations, adapted of Scarlatos [1999, p. 4].

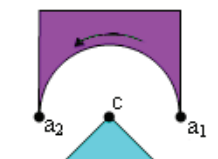
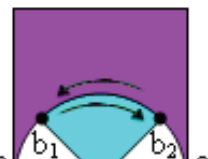
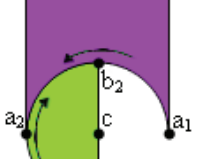
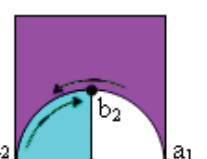
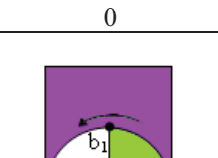
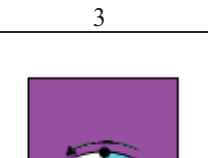
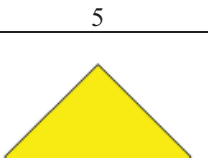
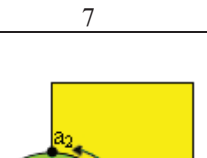
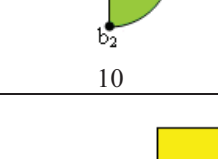
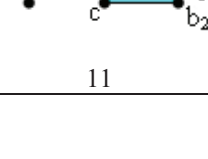
Representation of possible relations considering only circumference arcs			
			
			
			

Table 2: Possible representations of circumference arc relations.