FAKULTÄT FÜR INFORMATIK

TECHNISCHE UNIVERSITÄT MÜNCHEN

Interdisciplinary Project in Mathematics

Random Generation of Tangrams

Wiebke Köpp

FAKULTÄT FÜR INFORMATIK

TECHNISCHE UNIVERSITÄT MÜNCHEN

Interdisciplinary Project in Mathematics

Random Generation of Tangrams

Zufällige Generierung von Tangrams

Author:	Wiebke Köpp
Supervisor:	Prof. Dr. Dr. Jürgen Richter-Gebert
Advisor:	Dipl.-Inf. Martin von Gagern
Submission Date:	April 10, 2015

I assure the single handed composition of this interdisciplinary project in mathematics is only supported by declared resources.

Munich, April 10, 2015 Wiebke Köpp

Abstract

Kurzfassung

Contents

1 Introduction

Tangram is an old Chinese dissection puzzle whose rules are easily understood, but which can also be quite challenging. Seven puzzle pieces, called tans, have to be placed within a given shape in a way such that the entire shape is covered. Additionally, none of the pieces are allowed to overlap and all seven tans are to be used. As shown below, the seven puzzle pieces are three- and four-sided convex geometrical shapes and can be derived from cutting a square in a specific way.

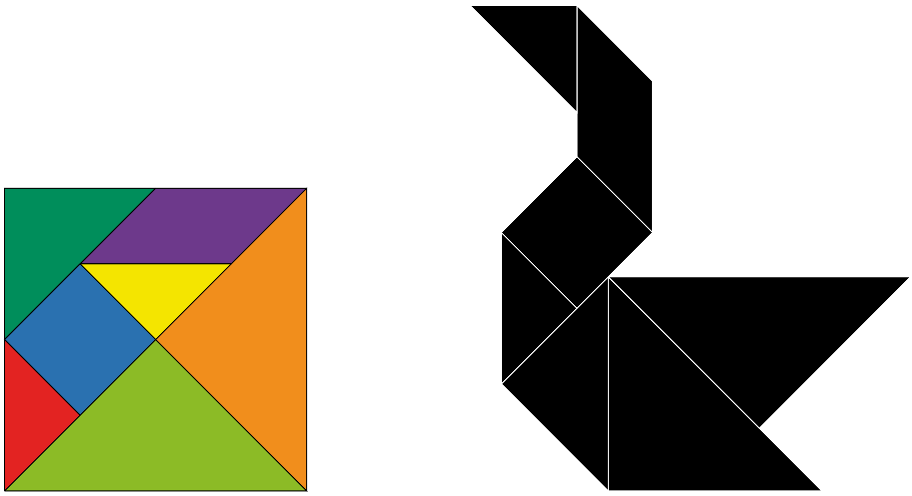


Figure 1.1: Dissection of a square into the 7 tans and an example for a given shape

Tangram is said be one of the most popular dissection puzzles all over the world and is even frequently used in education to teach children about mathematical concepts like symmetry, area, perimeter, shape similarity and even the pythagoras theorem.

The objective of this project is to create a tangram game that can be played in a browser. The user should first be presented with a number of tangrams to choose from and then be able to attempt solving the chosen tangram. While there exist many collections of tangrams published in books, adding them to database would require potentially time-consuming manual insertion. Therefore, a different approach will be applied here. Instead of displaying pre-defined shapes, the presented tangrams are generated randomly. The generated tangrams should not be entirely random however, but be interesting in some way. This is achieved by generating a large number of tangrams that are then ranked according to an interestingness measure. Additionally, the generation process is controlled in a way such that puzzles with immediately obvious solutions are avoided. Another requirement that has to be taken into account during generation is that the generated tangrams have to be connected, meaning that a newly randomly placed tan has to have at least one point in common with another already placed tan.

Related Work

An overview on the history of tangrams can be found in either [20] or [5]. Aside from more than 2,000 tangram examples, Slocum's book contains an extensive enumeration of tangram collections published all over the world. Elffers, on the other hand, additionally deals with some mathematical properties of different subclasses of tangrams. This includes grid tangrams [13], where the vertices of all tans can be placed in a coordinate system in way such that all coordinates are integers, and convex tangrams, of which only thirteen exist. The proof for the existence of only thirteen convex tangrams has first been published by Wang and Hsiung [21] in 1942.

Furthermore, there have been some attempts at computationally solving tangrams. Deutsch and Hayes [3] suggest a heuristic approach based on recursively splitting the tangram and treating the newly created parts in a similar manner as the original puzzle. In [18], a connectionist approach to solving tangrams has been proposed. Kovalsky and [14] apply their approach to edge-matching puzzles to tangrams, where edge-matching refers to the process of placing puzzle pieces with coloured edges in way such that the colours of adjacent edges match.

Possible candidates for interestingness measures for tangrams are the difficulty and visual aesthetics of a shape. For other well known puzzles such as Sokoban [11] or Soduko [10], various difficulty metrics have been described. However, these are often based on strategies applied during solving, which have not yet been researched as extensively for tangrams [1]. One example for an attempt to quantify the aesthetic value of polygons is Birkhoff's aesthetic measure. It can be calculated in terms of order and complexity of a shape, if these in turn can be quantified [6]. Nevertheless, many studies in this area also conclude that aesthetic preferences are usually biased due to a persons background [4]. Within tangram collections, shapes are often explicitly categorized according to their correspondence to real world objects like animals, people, numbers, letters or simple geometrical forms. Therefore, finding interesting tangrams also touches upon the subject of general purpose object recognition.

Outline

This report is organised as follows. Chapter 2 describes which mathematical concepts are applied in generating tangrams, measuring their interestingness and conducting computations on individual and groups of tans in a game setting. The following chapter first introduces the structure of the overall application and then describes the algorithms involved. Some implementation details are presented in chapter 4. This chapter specifically shows which features of the used programming language JavaScript are advantageous for the implementation of a random tangram generator in a browser. Chapter 5 shows the effect of different parameter settings during both generation and ranking of tangrams, as well as the results of a small user study. Finally, chapter 6 mentions potential future enhancements to different aspects of the application and concludes this report.

2 Background

One of the major prerequisites for conducting calculations on tangrams as well as individual puzzle pieces is a representation that supports the efficient computation of different properties of both individual and multiple pieces. The calculations involved in this project include, among others, the transformation of pieces, the detection of overlap between pieces and determining if a shape is completely covered by the seven tans. This chapter describes the required geometrical primitives and additionally shows how restricting the possible rotations of a tan to multiples of 45° leads to some simplifications.

2.1 Tangram and Tans

The puzzle pieces as well as tangram patters are polygons, which are usually represented by a sequence of points or line segments. As the puzzle pieces in tangram are always the same, this representation can however be simplified. When a tan is positioned on a two-dimensional plane it can already be fully described by its type, its position. There are five different tan types: the three-sided pieces: large, medium and small triangles, of which two large and two small triangles exist, and the four-sided pieces: square and parallelogram. In contrast to the other shapes, the parallelogram does not exhibit reflection symmetry in the same manner as the other pieces, but is rotationally symmetric. Therefore, it is the only piece that may have to be flipped in order to solve a tangram. The position of a tan can be defined by the position of just one vertex and the orientation of the tan, leading to a more lightweight representation of tans that can be easily updated in case the tan is transformed.

A tangram can then be simply described by its tans. The outline of a shape is useful for displaying tangrams as well as correctly detecting alternative solutions to an originally generated one. Additionally the outline plays an important role in the definition of interestingness measures. As the randomly generated tangrams are supposed to be connected, the outline of a tangram is a potentially self-touching polygon that also possibly contains holes.

All things considered, the required components for representing tans and tangrams are points defined by two coordinates in a two-dimensional plane and line segments and thus these will be studied more closely in the following.

2.2 Coordinates

Taking a look at the way the tan pieces are constructed from cutting a square, one can see that the irrational number $\sqrt{2} \approx 1.4142135623$ is essential for calculations surrounding tans. Figure 2.1 shows the dimensions of the tan pieces when the side length of the square is set to 4. With this setting, the hypotenuses of the two large triangles have length 4 and their legs have length $2\sqrt{2}$. The figure also shows that each tan is composed of a number of base triangles like the one displayed to the right of the square.

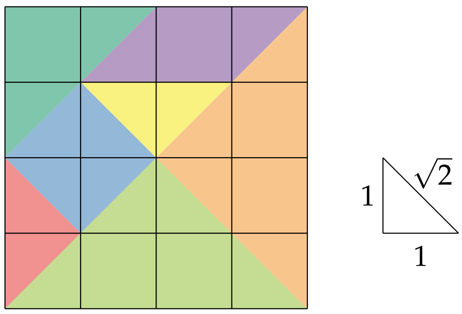


Figure 2.1: Dimensions of the tans

$\sqrt{2}$ will therefore occur in many coordinates of points and direction vectors and would implicate a heavy use of floating point arithmetic if coordinates would work with the number directly. Floating point arithmetic is computationally more expensive than integer arithmetic and requires special care in comparison operations due to rounding errors. A solution to this problem is basing all calculations on the commutative ring

$$\mathbb{Z}[\sqrt{2}] = \{a + b\sqrt{2} \mid a, b \in \mathbb{Z}\}$$

which essentially means that each number within a coordinate is represented by two integers. The fact that $\mathbb{Z}[\sqrt{2}]$ is closed under addition and multiplication. All other requirements for commutative rings can be shown in a similar way or be directly derived from knowledge about $\mathbb{Z}$.

$$\begin{aligned}
(a + b\sqrt{2}) + (c + d\sqrt{2}) &= (a + c) + (b + d)\sqrt{2} \\
(a + b\sqrt{2}) * (c + d\sqrt{2}) &= (ac + bd * \sqrt{2} * \sqrt{2}) + (ad\sqrt{2} + bc\sqrt{2}) \\
&= (ac + 2bd) + (ad + bc)\sqrt{2}
\end{aligned}$$

The usage of $\mathbb{Z}$ as the basis for coordinates is only possible due to the dimensions of the tans and the fact that tangrams have to be connected and none of the tans can overlap.

Additionally, rotations by 45° fit nicely into this scheme as $\sin(45°) = \cos(45°) = \frac{1}{2}\sqrt{2}$ and direction vectors within tans have only horizontal or vertical direction or take a form where both coordinates contain odd integers. This means that the factor of $\frac{1}{2}$ does not cause any problems.

2.3 Points

Points are represented by x- and y-coordinates that take the form of the coordinates described in the previous sections. At some points during the computation an extended representation is advantageous, particularly when points are being transformed, i.e. rotated or translated. Before a transformation, a point is transferred into projective plane, which is basically an euclidean plane embedded in the three dimensional space, here at $z = 1$. This, together with a projective definition of lines, leads to very interesting principles for intersecting lines and connecting points, but more importantly has a great impact on how the transformation of points can be calculated. The representation of a transformation within the usual euclidean space depends on the type of transformation. The description of a translation is different from the one of a rotation. Therefore, the composition of multiple transformations results in complicated expressions. This is not the case for the projective plane. Its properties lead to the uniform representation of transformations as matrices and thus simplifies the composition of arbitrary transformations to simple matrix multiplication. Table 2.1 shows how different transformations of a point $(x, y)^T$ in the euclidean and the projective plane can be expressed [19].

Transformation	**Euclidean Plane**	**Projective Plane**
Rotation	$\begin{pmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$	$\begin{pmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$
Translation	$\begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$

Table 2.1: Comparison of transformations in euclidean and projective plane

Aside from the transformation of an individual point and the obvious combination of points to form direction vectors, there exist some calculations to derive information about multiple points. The determinant of the 2×2 matrix containing two direction vectors as columns is one example. It helps to calculate the relative orientation between three points. Given three points A, B and C the determinant containing $(C - A)$ and $(C - B)$ determines in which order the points are positioned or whether C is on the right or left side of segment AB. A, B and C are collinear if the determinant is zero. For values smaller than zero the triangle formed by the three points has been defined in a counter-clock order, for values larger than zero the points are given in clockwise order (see Figure 2.2).

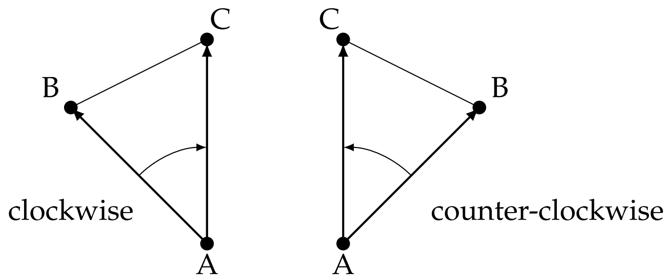


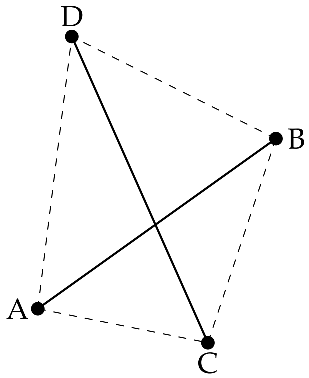
Figure 2.2: Relative orientation of three points or turning direction of consecutive line segments

The concept of relative orientation of three points is applied in multiple algorithms throughout this project. Two examples are a point-in-polygon algorithm that determines whether a given point is inside, outside or on the outline of a polygon and the computation of the convex hull of a shape or a set of points. One possibility to determine if a point lies inside a polygon is calculating the winding number which characterizes the number of times a polygon winds around a point [9]. The winding number is computed by casting a horizontal ray from the given point to right and then increasing and decreasing the winding number depending on whether a crossed line segment where the points are sorted according to their x-coordinate, points downwards or upwards.

The convex hull of a shape can be computed using a technique called Graham's scan [2, Chapter 33]. It relies on first sorting all points according to their angle in respect to the most lower left point and filtering out all points that have the same angle as a point further away form the most lower left point. Then, the sorted points are traversed and added or removed from the hull depending on their relative orientation to the two previously added points. This algorithm inspired the outline computation that will presented in Section 3.4.

2.4 Line Segments

[2, Chapter 33]



??

Figure 2.3: Intersecting line segments

Line segments are defined by their two endpoints. For some algorithms it makes sense to sort the points such that the first point of a segment is always the one with smaller x-coordinate or smaller y-coordinate if the x-coordinates are equal. The two questions that arise in the context of line segments are whether two line segments intersect or if a line segment contains a point. Both can be addressed by applying the concept of relative orientation of three points, introduced before. As seen in Figure **??**, two line segments intersect if the points of one line segment lie on opposite sides of the other line segment and vice versa. [2, Chapter 33]. When determining wether a point P lies on a line segments AB, a determinant of 0 for $(B - A)$ and $(P - A)$ shows that the three points are collinear. If this is the case, the the parameter t for which $P = A + t * (B - A)$ holds, can be calculated. If t is between 0 and 1, the point lies on the segment, as $A + t * (B - A)$ defines the line through A and B.

3 Design

3.1 Overall Application Structure

Commucate with database

3.2 Generation Process

First Naive Approach
As a first step towards the generation of interesting tangrams, the generation algorithms should favour configurations in which tans do not only have vertices, but also edges in common.

3.3 Interestingness Measures

convex hull

3.4 Gameplay

Outline computation Snapping

4 Implementation

The tangram generator and its associated interfaces for choosing one tangram out of a given number of presented ones and for solving a chosen tangram are implemented in JavaScript. JavaScript is a scripting language originally designed for adding interactivity to web pages by manipulating the structure and content of HTML-documents, but in recent years has also gained popularity in other domains like game development and server-side applications. Most modern browsers on both desktop and mobile devices include a JavaScript engine, which means that the user is not required to install additional frameworks for an application to execute properly. Other technologies for running client-side computations in a browser, like Java Applets, do not provide such widely spread support and have additionally experienced declining popularity due to security issues. In consequence, Javascript is well suited for an application targeted to support various input paradigms on different devices [17].

The Document Object Model (DOM) is an interface to HTML and XML documents. It allows accessing and changing the elements of a document and their properties as well as attaching event handlers to elements. Almost all changes in the interface of the tangram generator are realised with DOM manipulations. On startup, the web page contains structural elements for all parts of the interface that will be displayed during execution. This includes elements for each of the six tangrams, an area for playing the game and buttons for invoking processes not directly associated with a specific element. While some elements, like the buttons, are only hidden when first visiting the page, others have yet to be filled with content, like the elements displaying tangrams or the game play. When displayed, tangrams are drawn as Scalable Vector Graphics (SVG) [15], exploiting the fact that SVG is XML-based. The elementry of an SVG-element are therefore part of the DOM and can be treated like any other element. An alternative to using SVG as a drawing method is the HTML5 canvas element. Contrary to SVG, the canvas element is raster-based. After an element has been drawn it can not be updated in any way. Using SVG, moving a tan corresponds to updating the `x` and `y` attributes of the corresponding polygon. Achieving the same result with canvas on the other hand requires for the entire scene to be redrawn. Another advantage of using SVG as the graphics rendering methods, is the possibility of attaching event handlers to SVG elements. Event handlers are functions that are executed in case a certain event happens. Typical events in the scope of web pages are events involving user interaction through mouse, keyboard and touch or browser actions. The tangram generator almost solely makes use of mouse and touch events in order to make the interface interactive on both desktop and mobile devices. Event handlers for clicking and dragging are attached to an element once it is added to the DOM and the translation, rotation and flipping of tans.

JavaScript code is executed in a single thread and reacts asynchronously to events such as the ones described above. This implicates that heavier computations like the generation of a large number of tangrams block the simultaneous execution of any other code. Informing the user about the current state of the application during such computations is crucial to provide a satisfying user experience. Web workers [8] are a technology introduced to Javascript to allow the execution of scripts in an additional thread in the background. In contrast to the main execution thread, workers cannot directly access the DOM or use methods and properties associated with the current window. They can however communicate with the main thread in the form of messages that can be handled in the same way as any other event. Thus, Web workers enable showing the progress of the generation process without having to repeatedly interrupt it. The web worker handling the entire generation process is started immediately after the webpage with the tangram generator is entered. Each time a tangram has been generated, the worker sends a message to the main thread, which then updates the progress bar accordingly. After the desired number of tangrams has been generated, the web worker finishes by sending the generated tangrams to the main thread.

Which kind of messages can be exchanged between main thread and web workers is browser-dependent. Fortunately, all browsers are capable of sending `String` messages between threads. JavaScript Object Notation (JSON) is a key-value based data-interchange format with which an object can easily be transformed into a sendable String, so that objects like the generated tangrams can be exchanged as well. An example for a JSON representation of tan can be seen below.

```
{"tanType": 0,
 "anchor": {"x": {"coeffInt": 42, "coeffSqrt": -6},
            "y": {"coeffInt": 30, "coeffSqrt": -6}},
 "orientation": 3}
```

JSON objects are also used to send statistics about chosen tangrams and played games to a simple HTTP Server written in Node.js [12], a platform often referred to as server-side JavaScript. The server writes all JSON files it receives into a database. The database used here is MongoDB [16], which as a document-oriented database, functions very well with JSON objects.

5 Results

5.1 Generation Process

Comparison of the two algorithms Comparison of what happens with different settings for probability -> sufficiently large number of tangrams generated no visible difference on the first 6 tangrams

5.2 Interestingness Measures

Results of different measures

5.2.1 User testing

In addition to the interface described in section **??** evaluation interface Out of 1000 tangrams with properties. compact holes just one hanging piece small edges First six were the same for all users Overall .. took part.

6 Future Work and Conclusion

Interestingness: Finding the number of solutions. Even the number of possible locations for the large triangle could provide Including combinations Let the user decide on which properties he finds interesting potentially be providing sliders or checkmarks Apply techniques from information retrieval like Learning to rank algorithms and relevance feedback [7]

Game: Hint based on already places tans Save and Share tangrams Local storage to come back later More advanced touch interaction. Rotation by two fingers Rotations by multiples of 15 degrees

List of Figures

List of Tables

Bibliography

[1] B. Baran, B. Dogusoy, and K. Cagiltay. "How Do Adults Solve Digital Tangram Problems? Analyzing Cognitive Strategies Through Eye Tracking Approach." In: *Human-Computer Interaction. HCI Intelligent Multimodal Interaction Environments*. Ed. by J. A. Jacko. Vol. 4552. LNCS. Springer, 2007, pp. 555–563.

[2] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. 3rd ed. MIT Press, 2009.

[3] E. S. Deutsch and K. C. Hayes Jr. "A Heuristic Solution to the Tangram Puzzle." In: *Machine Intelligence* 7.Bernard Meltzer and Donald Michie (1972), pp. 205–240. Edinburgh University Press.

[4] R. S. Eberle. "The Role of Children's Mathematical Aesthetics: The Case of Tessellations." In: *The Journal of Mathematical Behavior* 35 (2014), pp. 129–143.

[5] J. Elffers. *Tangram. Das alte chinesische Formenspiel*. Du Mont, 1978.

[6] D. Filonik and D. Baur. "Measuring Aesthetics for Information Visualization." In: *Information Visualisation, 2009 13th International Conference*. IEEE. 2009, pp. 579–584.

[7] C. He, C. Wang, Y.-X. Zhong, and R.-F. Li. "A Survey on Learning to Rank." In: *2008 International Conference on Machine Learning and Cybernetics*. Vol. 3. IEEE. 2008, pp. 1734–1739.

[8] I. Hickson, ed. *Web Workers*. Candidate Recommendation. W3C. May 1, 2012. URL: `http://www.w3.org/TR/2012/CR-workers-20120501/` (visited on April/9/2015).

[9] K. Hormann and A. Agathos. "The Point in Polygon Problem for Arbitrary Polygons." In: *Computational Geometry* 20.3 (2001), pp. 131–144.

[10] M. Hunt, C. Pong, and G. Tucker. "Difficulty-Driven Sudoku Puzzle Generation." In: *UMAP Journal* (2007), pp. 343–362.

[11] P. Jarušek and R. Pelánek. "Difficulty Rating of Sokoban Puzzle." In: *STAIRS 2010: Proceedings of the Fifth Starting AI Researchers' Symposium*. IOS Press. 2010, p. 140.

[12] Joyent, Inc. *Node.js v0.12.2 Manual & Documentation*. 2014. URL: `https://nodejs.org/api/` (visited on April/9/2015).

[13] J. Köller. *Mathematische Basteleien - Tangram*. 1999. URL: `http://www.mathematische-basteleien.de/tangram.htm` (visited on 02/01/2015).

[14] S. Z. Kovalsky, D. Glasner, and R. Basri. "A Global Approach for Solving Edge-Matching Puzzles." In: (2014). arXiv preprint: `arXiv:1409.5957v1[cs.CV]`.

[15] C. McCormack, J. Watt, D. Schepers, A. Grasso, P. Dengler, J. Ferraiolo, E. Dahlström, D. Jackson, J. Fujisawa, and C. Lilley, eds. *Scalable Vector Graphics (SVG) 1.1 (Second Edition)*. Recommendation. W3C. Aug. 16, 2011. URL: `http://www.w3.org/TR/2011/REC-SVG11-20110816/` (visited on April/9/2015).

[16] MongoDB, Inc. *The MongoDB 2.4 Manual*. 2015. URL: `http://docs.mongodb.org/v2.4/` (visited on April/9/2015).

[17] Mozilla Developer Network and individual contributors. *JavaScript*. Mar. 24, 2015. URL: `https://developer.mozilla.org/en-US/docs/Web/JavaScript` (visited on April/9/2015).

[18] K. Oflazer. "Solving tangram puzzles: A connectionist approach." In: *International Journal of Intelligent Systems* 8.5 (1993), pp. 603–616.

[19] J. Richter-Gebert. *Perspectives on Projective Geometry: A Guided Tour Through Real and Complex Geometry*. 1st. Berlin; Heidelberg: Springer, 2011.

[20] J. Slocum. *The Tangram Book*. Sterling Publishing, 2003.

[21] F. T. Wang and C.-C. Hsiung. "A Theorem on the Tangram." In: *The American Mathematical Monthly* 49.9 (1942), pp. 596–599.