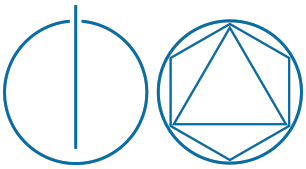
FAKULTÄT FÜR INFORMATIK

TECHNISCHE UNIVERSITÄT MÜNCHEN

Interdisciplinary Project in Mathematics

Random Generation of Tangrams

Wiebke Köpp



FAKULTÄT FÜR INFORMATIK

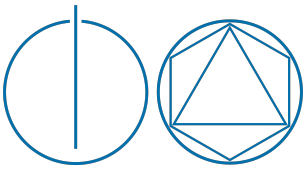
TECHNISCHE UNIVERSITÄT MÜNCHEN

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

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.

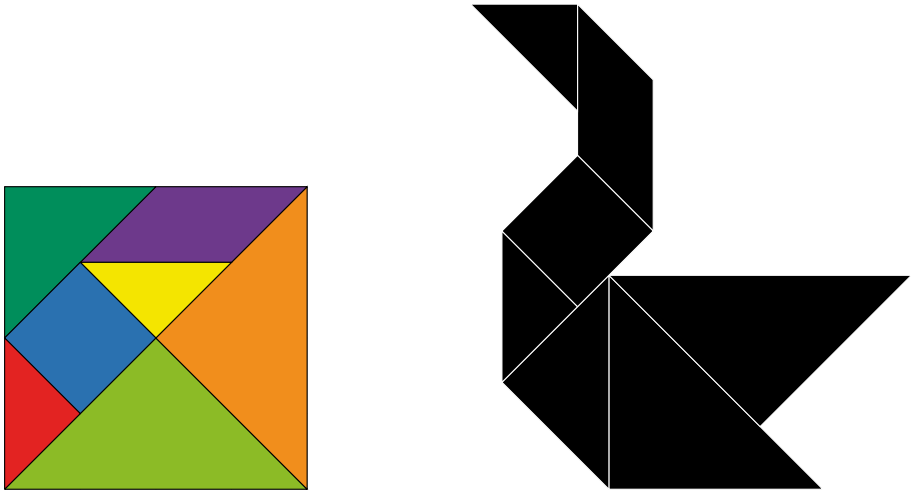


Figure 1.1: Dissection of a square into the 7 tans and an example for a given shape

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 [19] 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 [12], 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 [20] 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 [17], a connectionist approach to solving tangrams has been proposed. Kovalsky and [13] 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 [10] or Soduko [9], 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.

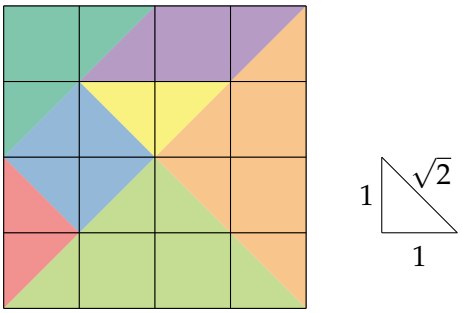


Figure 2.1: Dimensions of the tans

$\sqrt{2}$ 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 [18].

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).

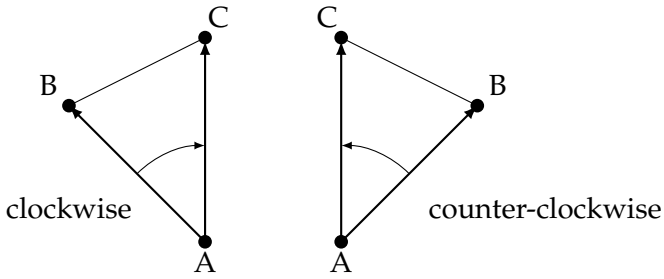


Figure 2.2: Relative orientation of three points or turning direction of consecutive line segments

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 [8]. 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 3.3.

2.4 Line Segments

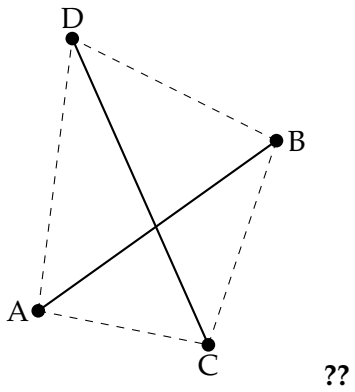


Figure 2.3: Intersecting line segments

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

This chapter deals with the algorithms specifically designed for the tangram generator. Before going into detail about these algorithms the overall structure of the application and its interface will be presented.

Once the user enters the site, the application starts to first pre-compute some values that are repeatedly used throughout the entire program, like direction vectors for tangrams, and then randomly generate tangrams using an algorithm described in section 3.1. As this takes some time depending on the device and the number of tangrams generated, a progress bar indicates the current state of the generation progress. At all time some information about how to interact with the application is displayed below the main interface. In the very beginning this also includes some information about tangrams in general. After the generation progress has finished the tangrams are sorted according to an interestingness measure and the 6 top ranked tangrams are displayed for the user to choose from. An option for generating new tangrams is also provided. Some candidates for interestingness measures are presented in section 3.2. If the user clicks on one of the displayed shapes, a bigger version of the tangram becomes visible and he or she can attempt to solve the puzzle by translating, rotating and flipping the puzzle pieces until they cover the whole given shape. At this point it is also possible to simply display the solution or receive a hint showing the position of one of the tans. In order to collect some statistics about which tangrams are preferred and how they are solved, some data is send to a database and can then be used to derive new interesting measures. Each time the user chooses one of 6 tangrams, his choice along with 6 tangrams. Statistics sent after solving a puzzle include the tangram itself as well as the time needed to solve the puzzle, the number of hints used and the number of actions performed to reach a solution.

The final user interface is shown in Figures 3.1 and 3.2.

3.1 Generation Process

Two approaches to randomly generate tangrams have been pursued. Both approaches have in common that they start out with generating an order for how the seven tans are placed and then position the first tangram. Subsequently new tans are placed by randomly choosing a vertex of one the already placed tans and choosing a vertex of the

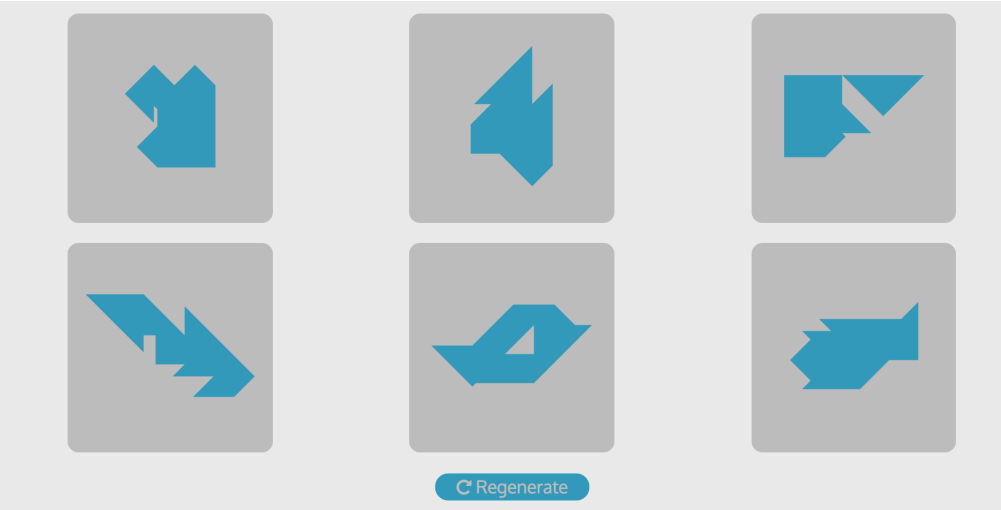


Figure 3.1: Interface showing 6 tangrams to the user to choose from

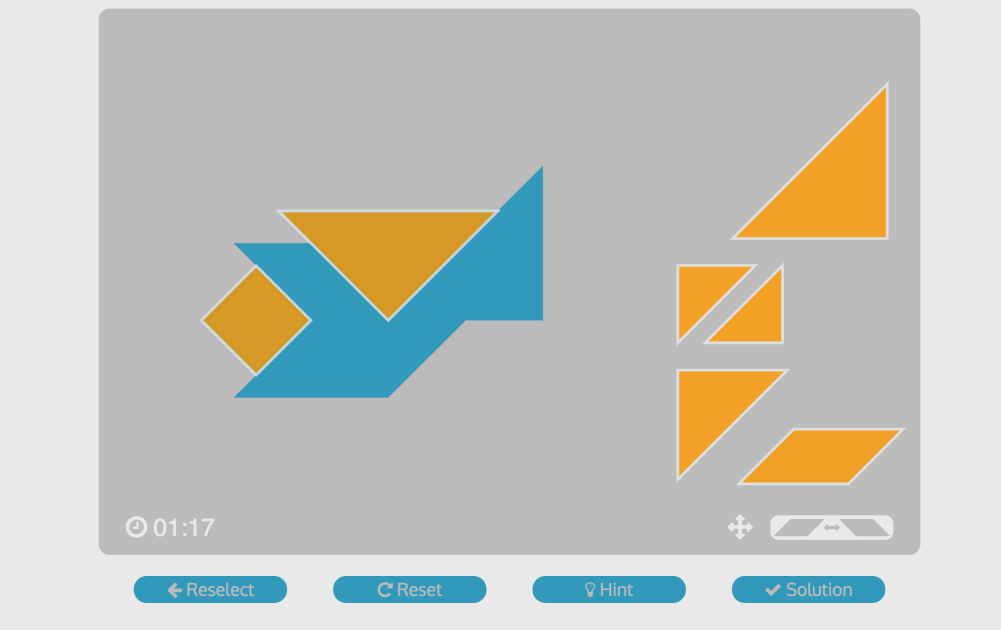


Figure 3.2: Interface allowing to solve a tangram

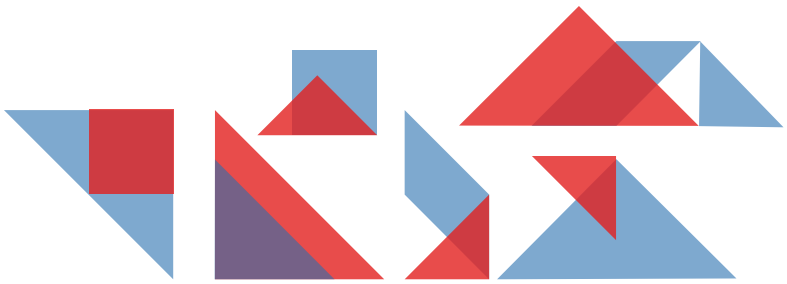


Figure 3.3: Invalid placements of the red tans when the respective blue one has already been present

new tan. The new tan and the already places pieces are then connected at those two points if the placement of the new tan does not violate the constraint that pieces cannot overlap. A collection of invalid placements is shown in Figure 3.3. If the placement fails, a new attempt of placing the tan is made. This process is continued until all 7 have been placed.

Determining if a placement is valid is done in three steps. The first step uses the point-in-polygon algorithm mentioned before. All points of the newly placed tan as well as between one and four points inside the tan are tested for containment in other tans. The points inside of a tan are needed to correctly detect cases where one puzzle piece is entirely contained in another one. The direction vectors to these points are are pre-calculated and include vectors to the center of each tan and depending on the size of a tan more vectors that are derived from partitioning tans in some way. Without modification this would introduce thirds and halfs into coordinates, which is why the direction vectors of tangrams have been scaled by 6 compared to the dimensions presented in chapter 2. This still does not correctly handle cases where a large tan is placed on top of a small tan such that it lies completely inside the newly placed tan. Thus the same test is conducted the other way around. The second steps uses line segment intersection to determine of any of the segments of already placed tans intersect with the segments of a new tan. Lastly, the bounding box of the tans including the newly placed tan is computed. The placement is then rejected if the the horizontal or vertical range is larger then a certain threshold. This step has been added in an attempt to avoid sequences of tans that are only attached at one point and favour tangrams that are somewhat compact.

As a first naive approach an orientation for each tan is sampled at very beginning. The attachment point at the already placed tans is chosen first. Then all points of the new tan are considered as possible attachment points in random order. If none of the points can be used, a new attachment point of the placed tans is samples.

Unfortunately, the fixed orientations together with the range threshold imposes a very strong restriction on the generation process which can lead to configurations where some tans are still missing, but cannot be placed anymore. Furthermore, this generation process leads to mostly loosely connected tangrams. Therefore, the second approach chooses orientations dynamically. Additionally, this approach steers the computation towards tangrams where tans have many edges in common.

The second approach starts out by sampling an orientation for the first tan and then places it. The process then continues to find two attachment points as before, however here, the orientation is sampled based on a probability distribution that favours orientations where the edges of the new tan align with already places pieces. This probability distribution is computed by checking if any of the segments meeting in the attachment point align for any of the orientations and apply a larger weight to such orientations. If the tan cannot be placed with the sampled orientation, a new orientation is sampled from the already computed probability distribution, where the probability of the just attempted orientation is set to zero. If none of the orientation lead to a valid placement, a new attachment point is chosen.

3.2 Interestingness Measures

The following list shows all properties that are computed after a tangram is generated sorted into different categories. While some of these are appropriate to be used directly as interestingness measures, others might be more suitable for filtering generated tangrams in respect to a certain properties. Altogether the properties have been chosen in an attempt to depict concepts like difficulty, visual aesthetics and correspondence to real world objects.

Properties of the outline: total number of vertices in the outline, number of vertices in the outline excluding holes, perimeter and number of hanging pieces
The first three properties are presumedly related to the compactness and therefore difficulty of tangrams. The number of hanging pieces is defined as the number of points where the outline touches itself and might capture the correspondence to real-world objects for a low non-zero result.

Properties of holes: number of holes, number of vertices involved in holes, total hole area and type of holes
These properties are potentially more useful for filtering on the supposition that tangrams with smaller holes that for example do not touch the outer outline are interesting. This is the reason for including the type of holes as a property. Holes can either touch the outer outline or not. For multiple holes, a mixed case can

also occur.

Properties of edges longest edge, shortest edge, number of matches edges
Shapes that correspond to real world objects often have small spikes which should be captured by the computation of the shortest edge in the outline.

Properties of points range in x and y, convex hull percentage, number of matched points
The first three properties again deal with the level compactness. The convex hull percentage is measures by first calculating the convex hull of the outline of a shape and then determining the percentage of area covered by the original shape. Thus, shapes with a convex hull percentage close to one are almost convex. Those often do not reveal much information about a tangram pattern and are therefore assumed to be difficult to solve. The number of matched points refers to the number of pairs of points that lie at the same position. This measure is highest when Ranking according to this number should result in somewhat star-like shapes.

Symmetry: A shape with symmetric features is assumed to be visually pleasing. Here, axis symmetry is considered.

3.3 Gameplay

The central algorithm needed during gameplay as well as for displaying the tangrams is the computation of the outline of a collection of non-overlapping tans. Drawing tangrams as individual tans could lead to the user being able to infer some of the structure due to the existence of lines between tans, which cannot be removed by setting borders without giving away some unwanted hints elsewhere. Additionally, computing the outline of the tans placed by the user and checking it against the outline of a given shape provides a neat possibility to detect all correct solutions to a tangram instead of just the generated one. In order to efficiently check the equality of the given outline containing only coordinates with integer numbers against the inexact coordinates of puzzle pieces, a snapping mechanism has been introduced.

The first step of computing the outline is calculating a set of line segment candidates that could be involved in the outline. These are the line segments of each individual tan, however split up into multiple segments where points from other tans touch a segment. Line segments that occur twice in this computation lie somewhere inside the tangram and therefore can be removed completely. The remaining segments are then traversed in a somewhat similar way as point in the convex hull computation. The computation starts with the point with the lowest x, and lowest y coordinate if there are multiple

points with the same x-value and initializes a last segment as a horizontal segment with the starting point as one endpoint. Then the segment in that point with the largest angle to the last segment is taken as a new last segment. This process is continued until the starting point is reached again and all points of the shape are contained by its outline. The second condition is needed to ensure that the process does not terminate to early in cases where the starting vertex should be revisited multiple times as the outline touches itself in that point. If the area of the computed outline is larger than the sum of areas of all puzzle pieces, the outline of the tangram contains holes that have yet to be calculated. However, holes are traversed in a very similar manner.

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 [16].

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) [14], 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 [7] 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 [11], 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 [15], which as a document-oriented database, functions very well with JSON objects.

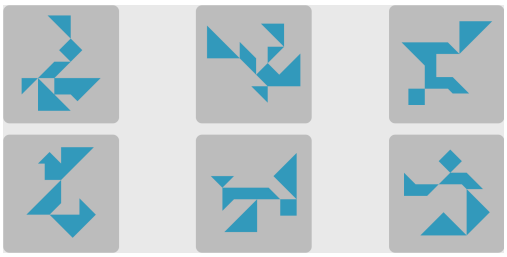
5 Results

5.1 Generation Process

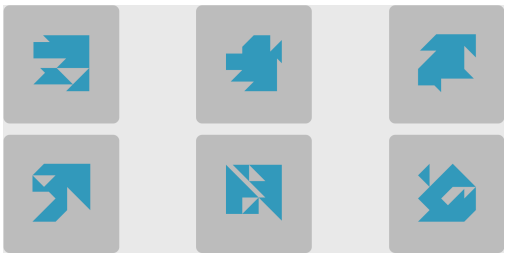
Section 3.1 presented two algorithms used to randomly generate tangrams. Additionally, there exist two parameters that can be adjusted to influence the generation process. The first parameter is a threshold setting the largest possible difference in x- and y-coordinates between two points. This parameter is used in both approaches. The second parameter applies only to the approach that steers towards a larger number of edge-matchings. It determines how high the probability for choosing an orientation with matched edges is compared to any other orientation. Figure 5.1 shows exemplementry results of generating six tangrams for both approaches and different parameter settings. Tangrams generated by the naive approach with a range of 50 are often only connected at points and contain next to no matched edges. When reducing the threshold to 30, more matched edges exist, but the resulting shapes are also somewhat similar and square-like. However, generating only these 6 tangrams took more then 70 attempts. The edge-matching approach obviously contains more matched edges. When adjusting the probability parameter, one can see a definite effect on the result when choosing a lower value, however increasing it past the standard value of 50 has no visually obvious implementations. The effect will possibly become completely obsolete once a larger number of tangrams is generated.

5.2 Interestingness Measures

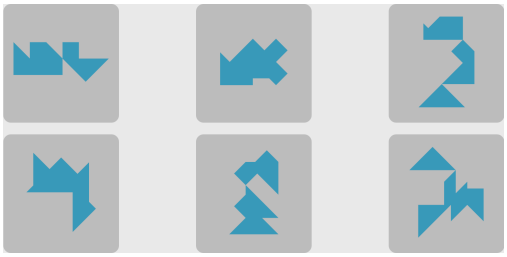
The properties presented in section 3.2. Figure 5.2 shows a few example for such measures. As expected, ranking according to both a high convex hull percentage and a low number of outline vertices leads to quite compact tangrams which are possibly harder to solve than tangrams where the tan pattern is more evident. Combining these two measures, leads to very different results depending on whether all outline points or only the outer outline points are considered.



(a) Naive approach

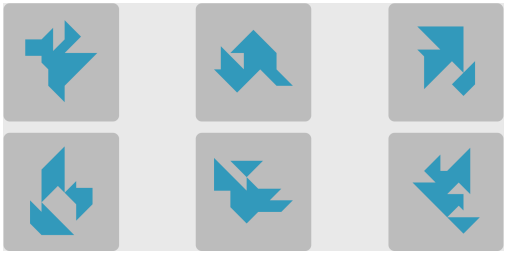


(b) Naive approach with lower range threshold

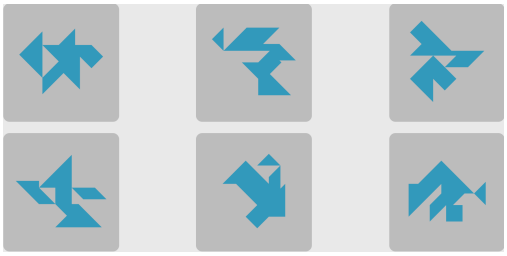


(c) Edge-matching approach

(d) Edge-matching approach with lower range threshold

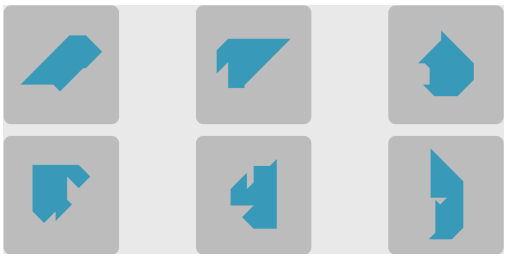


(e) Edge-matching approach with higher edge-matching weight

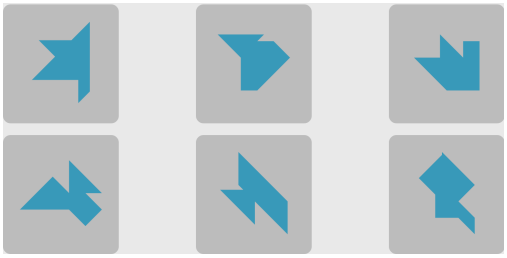


(f) Edge-matching approach with lower edge-matching weight

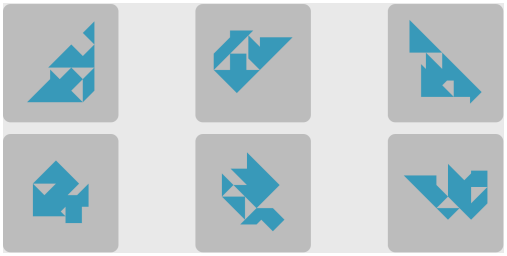
Figure 5.1: Results of generating six tangrams with different algorithms and different parameter settings



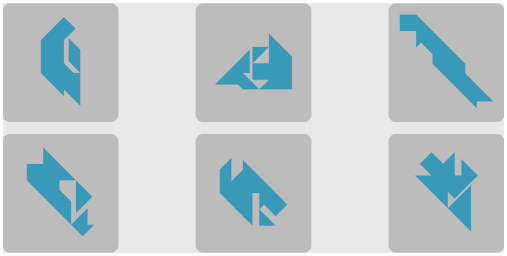
(a) Ranking based on a high convex hull percentage



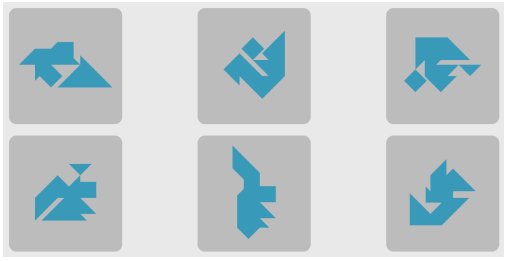
(b) Ranking based on a low number of outline vertices



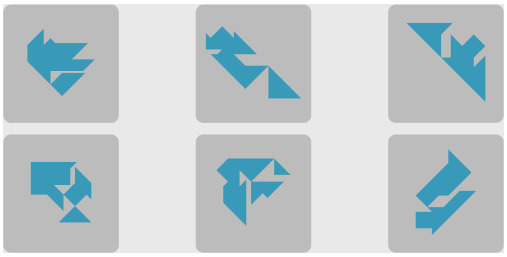
(c) Ranking based on a high convex hull percentage and a low number of outline vertices



(d) Ranking based on a high convex hull percentage and a low number of outer outline vertices



(e) Ranking based on a high number of matched vertices



(f) Ranking based on a short shortest edge

Figure 5.2: Results of generating 10.000 tangrams and ranking them according to different interestingness measures

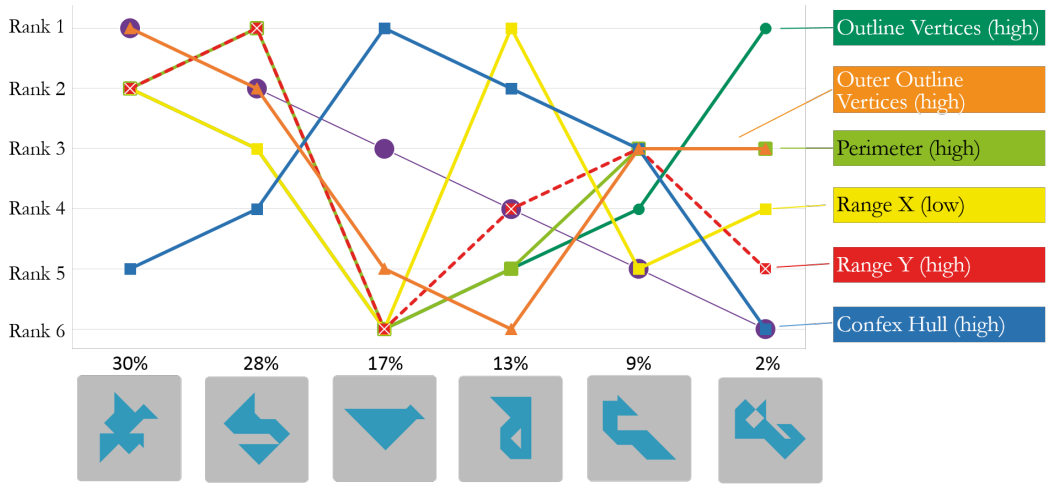


Figure 5.3

5.2.1 User testing

In addition to the interface described in the begin of chapter 3 an evaluation interface has been created in order to conduct a small user study

Out of 1000 tangrams with properties. compact holes just one hanging piece small edges First six were the same for all users

6 Conclusion and Future Work

In conclusion, this report showed that a tangram generator can be implemented as a browser application with solely client-side computations that is capable of generating a large number of tangrams within a reasonable time frame even on mobile devices. A first user study was conducted and showed that there exists at least a bit of consensus about which shapes are interesting. The game associated with the generator has received largely positive feedback. In just 2 days more than 150 tangrams have been solved by more than 40 users.

Potential enhancements for the generator include the improvement of interestingness measures and some additions to the game interface. Specifically a thorough examination of a larger scale evaluation should lead to promising combinations of existing properties. One possible measure for difficulty would be the number of solutions of a tangram or even just the number of possible locations for the large triangle as the placement of those restricts the possibilities for the other pieces most. Another feasible improvement which aims to enhance the user experience of the generator, would be to allow the user more control over the generation process by providing checkmarks or sliders for some selected properties.

Currently, the game interface reacts to only one finger touches on mobile devices. A gesture using two fingers for rotating elements has become more common. By making use of HTML5 Local Storage, a game could be left and then continued later. Similarly, a possibility to share interesting tangrams with other users could introduce elements of a more competitive gameplay. Furthermore, different rotation angles and additional attachment points could be investigated.

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] 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).

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

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

[10] 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.

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

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

[13] 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]`.

[14] C. McCormack, J. Watt, D. Schepers, A. Grasso, P. Dengler, J. Ferraiolo, E. Dahlström, D. Jackson, J. Fujisawa, and C. Lilley, eds. *Scalable Vector Graph-*

ics (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).

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

[16] 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).

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

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

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

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