FACTA UNIVERSITATIS (NIŠ)
Ser. Math. Inform. Vol. 34, No 1 (2019), 137–147
https://doi.org/10.22190/FUMI1901137M

COMPUTING TRIANGULATIONS OF THE CONVEX POLYGON IN PHP/MYSQL ENVIRONMENT

Sead H. Mašović, Muzafer H. Saračević, Predrag S. Stanimirović and Predrag V. Krtolica

Abstract. In this paper we implement the Block method for convex polygon triangulation in the web environment (PHP/MySQL). Our main aim is to show the advantages of the usage of web technologies in performing complex algorithm from computer graphics. The basic assumption is that once obtained, the results can be stored in a database and used for other calculations. Databases are convenient and structured methods of sharing and retrieving data. We have performed a comparative analysis of the developed program with respect to two criteria: CPU time in generating triangulation and CPU time in reading results from the database.
Keywords: Computer graphics, Polygon triangulation, Block method, PHP/MySQL.

1. Introduction and preliminaries

Polygon triangulation is an important problem applicable in computer graphics. Restricted to the convex case, the decomposition of a polygon is done into triangles by a maximal set of non-intersecting diagonals.

Let P_n denote a polygon with n vertices. The total number T_n of n-gon triangulations is

$$T_n = C_{n-2} = \frac{1}{n-1}\binom{2n-4}{n-2} = \frac{(2n-4)!}{(n-1)!(n-2)!}, \quad n \geqslant 3. \tag{1.1}$$

Here, C_n represents the nth Catalan number (see e.g. [9]).

The set of all triangulations of the convex polygon P_n is denoted by $\mathcal{T}_n$. Diagonal connecting vertices i and j are denoted by $\delta_{i,j}$. An outer face edge can be considered as a diagonal, while nonadjacent vertices are connected by an *internal diagonal*.

Received June 10, 2018; accepted January 29, 2019
2010 *Mathematics Subject Classification.* Primary 32B25; Secondary 68N15, 68P15

Many authors deal with the problem of how to generate the triangulation of a convex polygon based on some criterion. In this paper we implement the Block method for convex polygon triangulation [6] in the web environment using PHP/MySQL technologies.

The combination of PHP and MySQL is the most convenient approach to dynamic, database-driven web design application. Due to its open source roots, it is free to implement and is therefore an extremely popular option for web development.

PHP is extremely powerful and exceptionally fast it can run on even the most basic hardware, and it hardly puts a dent in the system resources. The main characteristics of PHP are described in [2].

According to the TIOBE Programming Community index[1], the PHP programming language is one of the top 10 most popular programming languages. Eighty percent of the top 10 million websites use PHP in one way or the other, including Facebook and Wikipedia.

PHP, as a scripting language, is popular among web developers because of its ability to interact with database systems.

MySQL is probably the most popular database management system for web servers.

MySQL is a fast and powerful, yet easy-to-use, database system that offers just about anything a website would need in order to find and serve up data to browsers.

The combination of PHP and MySQL can be used to build simple or complex and high traffic websites (see for e.g. [1, 7]). Similarly, the authors [4] used PHP/MySQL environment for computing the weighted Moore-Penrose inverse employing the partitioning method, as well as for storing the generated results.

This paper is organized as follows. In the Section 2 we present the main parts of the Block method for convex polygon triangulation. In Section 3 we describe the implementation of the algorithm in the PHP/MySQL environment. Section 4 includes a comparative analysis of the obtained numerical results.

2. Block method for convex polygon triangulation

Here we restate the Block method for convex polygon triangulation [6] which is the subject of our implementation.

The general strategy of the method is to decompose the problem into smaller dependant subproblems. Each subproblem is solved only once and used many times avoiding unnecessary repetitions of calculation.

The method is based on the usage of the previously generated triangulations for polygon with a smaller number of vertices. More precisely, the algorithm generates the set $\mathcal{T}_n$ using all the previously generated triangulations $\mathcal{T}_b$, where $b < n$. The set $\mathcal{T}_b$ is used as many times as necessary as a block, i.e. it is repeated several times in $\mathcal{T}_n$.

[1] https://www.tiobe.com/tiobe-index/

The formal statement of the subject method is given by the following equation

$$T_n = 2T_{n-1} + \text{rest}(R_n). \tag{2.1}$$

The general idea of the Block method uses T_{n-1} to generate T_n, which is illustrated in Figure 2.1, where one case of the transformation process from a P_5 triangulation into two corresponding P_6 triangulations is presented. In part (a) we see that the diagonals $\delta_{2,4}$ and $\delta_{2,5}$ make all vertices closed except the vertices 1, 2, 5, and 6 which form a quadrilateral. The parts (b) and (c) show two ways to triangulate a quadrilateral, which gives two P_6 triangulations having a P_5 triangulation as a starting block.

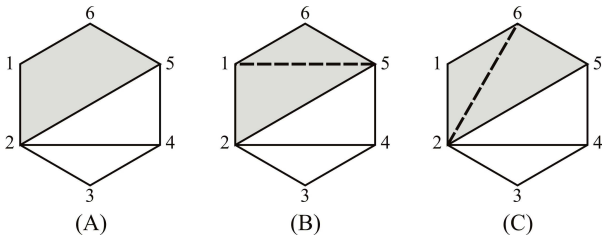

FIG. 2.1: Transformation from a P_5 into the corresponding P_6 triangulations. $P_5 = \{(2,4),(2,5)\} \rightarrow P_6 = \{(2,4),(2,5),(1,5)\&(2,4),(2,5),(2,6)\}$

Starting from the assumption that triangulation has at least two ears and that, in the worst case, one ear can be a vertex n, then we always have at least one ear among the rest of the vertices.

For the correctness of the algorithm in the procedure used for finding and eliminating closed vertices, the authors introduced a list of ordered pairs of the form

$$L = \{(1,1),(2,2),\ldots,(n,n)\}. \tag{2.2}$$

After the elimination of $n-l$ pairs the list L becomes

$$L = \{(s,i_s),\ s = 1,\ldots,l\},\ 4 \leq l \leq n,\ i_l = n. \tag{2.3}$$

The values $i_s,\ s = 1,\ldots,l$ are the vertex marks, while the values $1,\ldots,l$ represent the relative vertex positions in the list L.

Here we restate two additional algorithms 2.1 - Pair elimination & 2.2 - Form a quadrilateral, which are part of the Block Method Algorithm 2.3.

Algorithm 2.1 Pair elimination

Require: List L of the form (2.3) and vertices i_p and i_q, where $d(p,q)=2$.

1: Remove from the list L the pair placed between the pairs (p,i_p) and (q,i_q) in a circular manner.
2: Decrease by one the first pair members in the pairs following the eliminated one.

Algorithm 2.2 Form a quadrilateral

Require: List L of the form (2.2), integer n and array of $n-4$ diagonals (i.e a row in the table for $\mathcal{T}_n$).

1: Find a diagonal δ_{i_p,i_q} where $d(p,q)=2$ in the list L.
2: Call **Algorithm 2.1** for the parameters i_p and i_q.
3: Repeat Steps 1–2 $n-4$ times.

The main algorithm for the Block method is presented below.

Algorithm 2.3 Algorithm for the Block method

Require: An integer n and $\mathcal{T}_b$ with $row_b = C_{n-3}$ rows and $col_b = n-4$ columns

1: Create an empty table for $\mathcal{T}_n$ with $row_n = C_{n-2}$ rows and $col_n = n-3$ columns.
2: Fill the table for $\mathcal{T}_n$ by the triangulations from $\mathcal{T}_b$ duplicating each row from $\mathcal{T}_b$.
3: Fill the rest of the entered blocks (the last column in the first $2row_b$ rows) in the following way.
`for` $(i=1; i<=2row_b; i+=2)$
{
Make a list L of the form (2.2).
Call **Algorithm 2.2** with row i from the table for $\mathcal{T}_n$ as a parameter.
From the remaining four vertices in the list L make a diagonal δ_{i_1,i_3} and place it in the last column of the row i and diagonal δ_{i_2,i_4} and place it in the last column of the row $i+1$.
}
4: Fill the rest of the table for $\mathcal{T}_n$ containing $T_n - 2T_b$ rows.

4.1 Filling the first $n-4$ columns in the last $row_n - 2row_b$ rows.
$i = 2 * row_b + 1$;
Make the list L of the form (2.2).
Eliminate the vertices adjacent to n calling Algorithm 2.1 for the parameters 1 and $n-1$.
Fill the current table row i by diagonals $\delta_{2,n}, \delta_{3,n}, \ldots, \delta_{n-2,n}$.
The first $n-4$ columns in the rest $row_n - 2row_b - 1$ rows should be filled with the diagonals with the last vertex n, while the first vertices are combinations of the $(n-4)$th class in the set $\{2,3,\ldots,n-2\}$. The number of these combinations is $\binom{n-3}{n-4} = n-3$.

4.2 Filling the last column in the last $(row_n - 2row_b)$ rows.
`for` $(i = 2row_b + 2; i <= row_n; i++)$
{
Make the list L of the form (2.2).

Call **Algorithm 2.2** with the row i from the table for $\mathcal{T}_n$ as a parameter. From the remaining four vertices in the list L make a diagonal δ_{i_1,i_3} and place it in the last column of the row i.
}

3. PHP/MySQL implementation of Block method

The most used architecture for development of web applications is three-tier architecture (Figure 3.1). Three-tier web architecture is a unique system for developing web database applications which work around the three-tier model comprising the database tier at the bottom, the application tier in the middle and the client tier on top.

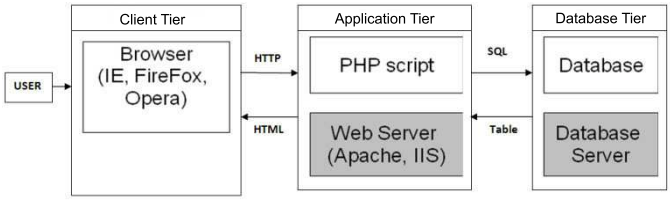

FIG. 3.1: Three-tier Web Architecture

The web interface of our application is given in Figure 3.2

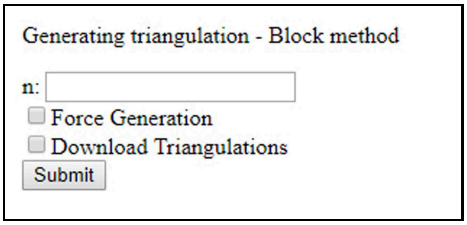

FIG. 3.2: Web interface of the application

According to the three-tier architecture, our application is organized as follows:

- On the *client tier* we have the web interface;
- Algorithm for the Block method is performed on *application tier*;
- Generated triangulations are stored on *database tier*;

In what follows, we presents a detailed view of the application scenario:

First, we have to enter the value n for which convex polygon we want to calculate triangulation.

Preconditions: $n \geq 4$

Second, when we press the submit button, Application search in database:

Case 1: Force Generation = Not marked

Have we already calculated triangulations of n in the database;

- If we have, the application displays the results of T_n in the browser;
- If we have not, the application checks if we have the results of T_{n-1} in the database:
  * If we have, then call Algorithm 2.3
  * If we do not, the preconditions of Algorithm 2.3 are not fulfilled;

Case 2: Force Generation = Marked

Have we already calculated triangulations of $n-1$ in the database;

- If we have, then call Algorithm 2.3
- If we have not, the preconditions of Algorithm 2.3 are not fulfilled;

Third, the output results can be downloaded in a `CSV` format if we mark "Download Triangulation".

Example 3.1. Let us illustrate how the application works on generating hexagon triangulations using the already known pentagon triangulations.

First, $n = 6$;
Preconditions fulfilled: $6 \geq 4$;

Second, the submit button is pressed

Case 1: Force Generation = Not marked

- The application checks if we have the results of T_5 in the database:
  * If we have, then call Algorithm 2.3

→ Generating triangulations and displaying results in browsers (Figure 3.3)

Done: 14 triangulations were found for P_{6} in 0.345 sec.
T_1= 3-6; 3-5; 1-3
T_2= 1-3; 1-5; 3-5
T_3= 4-6; 1-3; 1-4
T_4= 1-4; 1-3; 1-5
T_5= 2-5; 2-4; 2-6
T_6= 1-5; 2-4; 2-5
T_7= 4-6; 1-4; 2-4
T_8= 1-5; 1-4; 2-4
T_9= 2-5; 3-5; 2-6
T_10= 2-5; 1-5; 3-5
T_11= 3-6; 4-6; 1-3
T_12= 2-4; 2-6; 4-6
T_13= 3-6; 3-5; 2-6
T_14= 2-6; 3-6; 4-6

FIG. 3.3: Generating results for T_6

4. Comparative analysis and experimental results

The main idea of our implementation is to provide an appropriate client-server web application, in the free open source PHP/MySQL development environment, utilizing the minimum of resources: an internet browser and an operating system.

For a comparative analysis in presenting the advantages of web technologies, we implement an additional algorithm from the field of computer graphics (Orbiting Triangle method [8]).

Both algorithms are based on the usage of the previously generated triangulations for a polygon with a smaller number of vertices.

The execution times with respect to two criteria are presented in Table 4.1. The table column "Speedup" shows the quotient of the values contained in the previous two columns.

The testing is performed on the following configuration*: *CPU - Inter(R) Core(TM) i5-4210U CPU @ 1.70GHz 2.40GHz, RAM memory 8GB, Graphics card: NVIDIA GeForce 820M.*

Table 4.1: The execution times of computing triangulations (in seconds)

n	Number of triangulations	BM in generating	BM in reading from DB	Speedup	OTM in generating	OTM in reading from DB	Speedup
5	5	0.256	0.003	**85.33**	0.067	0.001	**67.00**
6	14	0.345	0.003	**115.00**	0.088	0.001	**88.00**
7	42	0.391	0.003	**130.33**	0.123	0.001	**123.00**
8	132	0.457	0.004	**114.25**	0.185	0.002	**92.50**
9	429	0.756	0.008	**94.50**	0.927	0.002	**463.50**
10	1,430	1.606	0.019	**84.53**	1.524	0.003	**508.00**
11	4,862	3.915	0.063	**62.14**	3.182	0.008	**397.75**
12	16,796	26.657	0.461	**57.82**	10.081	0.024	**420.04**
13	58,786	185.566	2.482	**74.76**	29.713	0.075	**396.17**
14	208,012	883.726	6.802	**129.92**	121.749	0.248	**490.92**
15	742,900	4,498.768	25.697	**175.07**	536.326	0.975	**550.08**

5. Conclusion

We implemented the Block method for convex polygon triangulation in the web environment using the open source software (PHP/MySQL). With this implementation we presented the advantages of web technologies in preforming a complex algorithm from computer graphics. The research also contributes to the manner in which an MySQL database is used for storing the obtained results and utilizing them for another calculation. As presented in the comparative analysis section, we can conclude that the advantages of using a database in performing complex algorithms are justified. This way of implementation provides a good basis for further application of the web technology in computing other algorithms.

A Important source code of the implementation

Source code for creating a MySQL database:

```
CREATE DATABASE IF NOT EXISTS triangulation;';
        if ($conn->query($sql)) {
            $conn->select_db('triangulation');
        } else {
            die('Could not create database: ' . $conn->error . '<br/>');
        }
```

The source code for database connection:

```
// Connection
$conn = new mysqli('localhost', 'root', '');
if ($conn->connect_errno) {
    die('Could not connect: (' . $conn->connect_errno . ')
        ' . $conn->connect_error . '<br/>');
    }
```

In our implementation, we use only one table for storing generated triangulations.

```
CREATE TABLE IF NOT EXISTS Triangulation
        (
            n int,
            T int,
            i int,
            j int,
            INDEX Triangulation_n_idx (n),
            INDEX Triangulation_T_idx (T),
            INDEX Triangulation_i_idx (i),
            INDEX Triangulation_j_idx (j)
        );
```

The source code of the implementation of Algorithm 2.3 - step 3:

```
        // Step 3
        // diagonal \delta_{i_1 ,i_3}
        $sql .= '
        INSERT INTO Triangulation
        SELECT DISTINCT a.n,
            a.T,
            1 AS i,
            ' . ($n-1) . ' AS j
        FROM Triangulation a
        WHERE a.n=' . $n . '
            AND a.T%2=0;
        ';
        // diagonal \delta_{i_2 ,i_4}
        $sql .= '
        INSERT INTO Triangulation
        SELECT ' . $n . ' AS n,
            a.T,
```

```
            a.v AS i,
            ' . $n . ' AS j
        FROM
            (SELECT a.T,
                a.j AS v
            FROM Triangulation a
            WHERE a.n=' . $n . '
                AND a.T%2=1
                AND a.i=1
            UNION
            SELECT DISTINCT a.T,
                2 AS v
            FROM Triangulation a
            WHERE a.n=' . $n . '
                AND a.T%2=1) a
            INNER JOIN
            (SELECT a.T,
                a.i AS v
            FROM Triangulation a
            WHERE a.n=' . $n . '
                AND a.T%2=1
                AND a.j=' . ($n-1) . '
            UNION
            SELECT DISTINCT a.T,
                ' . ($n-2) . ' AS v
            FROM Triangulation a
            WHERE a.n=' . $n . '
                AND a.T%2=1) b
            ON a.T=b.T
                AND a.v=b.v;
            ';
```

The source code of the implementation of Algorithm 2.3 - step 4:

```
// Step 4
for ($k = 1; $k <= $n-4; $k++) {
    $sql .= '
    INSERT INTO Triangulation
    SELECT ' . $n . ' AS n,
        (CASE a.T
            WHEN @curTn_1 THEN @curTn
            ELSE @curTn := @curTn + SIGN(@curTn_1 := a.T) * SIGN(@curK := 1)
                            * SIGN(@lastV := ' . ($n-2) . ')
        END) + ' . $N . ' AS T,
        (CASE
            WHEN a.i=' . $n . ' THEN @lastV
            ELSE a.i
        END) AS i,
        SIGN(
            CASE WHEN a.j=' . ($n-1) . ' AND @curK = ' . ($k+1) . '
                THEN @lastV:= a.i
                ELSE 1
            END) *
            (CASE
                WHEN a.j=' . ($n-1) . ' AND @curK <= ' . $k . '
                THEN ' . $n . ' * SIGN(@curK := @curK+1)
```

```
            ELSE a.j
        END) AS j
FROM
(SELECT a.T,
    a.i,
    a.j
FROM Triangulation a
WHERE a.n=' . ($n-1) . '
    AND a.T IN
(SELECT a.T
FROM Triangulation a
WHERE a.n=' . ($n-1) . '
    AND a.j=' . ($n-1) . '
GROUP BY a.T
HAVING count(a.i)>=' . $k . ')
UNION SELECT a.T,
    ' . $n . ' AS i,
    ' . $n . ' AS j
FROM Triangulation a
WHERE a.n=' . ($n-1) . '
    AND a.j=' . ($n-1) . '
GROUP BY a.T
HAVING count(a.i)>=' . $k . ' ) a ,
(SELECT @curTn := 0, @curTn_1 := 0, @curK := 0, @lastV := ' . ($n-2) . ') b
ORDER BY a.T,
    a.i,
    a.j;
';
if ($result = $conn->query('
    SELECT a.T
    FROM Triangulation a
    WHERE a.n=' . ($n-1) . '
        AND a.j=' . ($n-1) . '
    GROUP BY a.T
    HAVING count(a.i)>=' . $k . ';
    '
    )) {
    $N += $result->num_rows;
    $result->close();
    }
    else {
        die('Could not access Triangulation table: ' . $conn->error . '<br/>');
    }
}
```

REFERENCES

1. D. Lane, H. Williams: *Web Database Applications with PHP and MySQL, 2nd Edition*, O'Reilly Media, 2009.
2. M. Rahman: *PHP 7 Data Structures and Algorithms: Implement Linked Lists, Stack, and Queues Using PHP*, Packt Publishing, 2017.

3. M. Saračević, P. Stanimirović, S. Mašović, E. Biševac: *Implementation of the convex polygon triangulation algorithm*, Facta Universitatis Math. Inform. **27** (2012), pp. 213–228.
4. M. Tasić, P. Stanimirović, S. Pepić: *Computation of generalized inverses using Php/MySql environment*, Int. J. Comput. Math. **88** (2011), pp. 2429–2446.
5. P. Krtolica, P.Stanimirović, M. Tasić, S. Pepić: *Triangulation of Convex Polygon with Storage Support*, Facta Universitatis, **29:2** (2014), pp. 189–208.
6. P. Stanimirović, P. Krtolica, M. Saračević, S. Mašović: *Block Method for Convex Polygon Triangulation*, Rom. J. Inf. Sci. Tech. **15:4** (2012), pp. 344–354.
7. R. Nixon: *Learning PHP, MySQL & JavaScript, 5th Edition*, O'Reilly Media, 2018.
8. S. Mašović, I. Elshaarawz, P. Stanimirović, P. Krtolica: *Orbiting triangle method for convex polygon triangulation*, Applicable Analysis and Discrete Mathematics, **12** (2018), pp. 439–454.
9. T. Koshy: *Catalan Numbers with Applications*, Oxford University Press, New York, 2009.

Sead H. Mašović
University of Niš
Faculty of Science and Mathematics
Department of Computer Science
18000 Niš, Serbia
sead.masovic@pmf.edu.rs

Muzafer H. Saračević
University of Novi Pazar
Department of Computer Science
36300 Novi Pazar, Serbia
muzafers@uninp.edu.rs

Predrag S. Stanimirović
University of Niš
Faculty of Science and Mathematics
Department of Computer Science
18000 Niš, Serbia
pecko@pmf.ni.ac.rs

Predrag V. Krtolica
University of Niš
Faculty of Science and Mathematics
Department of Computer Science
18000 Niš, Serbia
krca@pmf.ni.ac.rs