DOI: 10.2478/bhee-2018-0006

Review scientific paper/Pregledni naučni rad

IMPLEMENTATION OF GEOMETRIC TRANSFORMATIONS ON THE WEB USING CSS3 STANDARD

IMPLEMENTACIJA GEOMETRIJSKIH TRANSFORMACIJA NA WEBU PRIMJENOM CSS3 STANDARDA

Emir Skejić[1], Dženita Vejsilović[2]

Abstract: HTML (abbrev. of HyperText Markup Language) and CSS (abbrev. of Cascading Style Sheet) are the two key technologies to build Web pages. HTML provides the structure of the page, while CSS is responsible for its visual and sound layout. CSS3 is the latest standard for CSS, which is fully backward compatible with older versions of CSS. Most of the new features provided by CSS3 standard are implemented in modern Web browsers. This paper presents the implementation of geometric transformations on the Web by using CSS3 standard. The purpose of the developed HTML page was to give an overview of the capabilities that the CSS3 standard provides for handling geometric content on the Web. It has been analytically demonstrated that CSS3 as a descriptive language allows adding a variety of geometric transformations to websites, ensuring that the web site is visually attractive, well-designed and easy to navigate.

Keywords: web design, HTML, CSS3, geometric transformations

Sažetak: HTML (skr. od HyperText Markup Language) i CSS (skr. od Cascading Style Sheet) su dvije ključne tehnologije za izgradnju web stranica. HTML obezbjeđuje strukturu stranice, dok je CSS zadužen za njezin vizualni i zvučni izgled. CSS3 je najnoviji standard za CSS, koji je potpuno unatrag kompatibilan sa starijim verzijama CSS-a. Većina novih svojstava koje donosi CSS3 standard implementirana je u modernim web preglednicima. U ovom je radu predstavljena implementacija geometrijskih transformacija na webu primjenom CSS3 standarda. Svrha kreirane HTML stranice bila je dati pregled sposobnosti koje CSS3 standard pruža pri rukovanju geometrijskim sadržajem na webu. Analitički je demonstrirano da CSS3 kao deskriptivni jezik omogućava dodavanje raznih geometrijskih transformacija na web stranice, čime se osigurava da web-sajt bude vizualno atraktivan, dobro osmišljen i jednostavan za navigaciju.

Ključne riječi: web-dizajn, HTML, CSS3, geometrijske transformacije

INTRODUCTION

In this time and age, when websites play a very important role in marketing and online sales, but also in all other areas of life (education, news, entertainment, etc.), it is important to have a site that will attract and retain the end user's attention. For the appearance of the sites which can be seen on the Web today, we can give a credit to CSS, English abbreviation of words Cascading Style Sheets. The CSS standard introduced revolution to the web design world. The latest version of this standard, CSS3, was recommended by W3C (from the World Wide Web Consortium) in June 1999 and is an upgrade of previous versions. CSS3 is divided into modules, and each module has new, expanded features in relation to the features defined in the previous versions [1-3].

[1] Faculty of Electrical Engineering, University of Tuzla, BiH
[2] H&H Inc. Development and Research, Tuzla, BiH
emir.skejic@untz.ba, dzenita.vejsilovic@gmail.com
Paper submitted: September 2017 Paper accepted: October 2018

Some of the advantages of using CSS in web design are:

- control of the layout/environment of multiple pages using only one CSS document,
- greater precision in control of layout,
- different layout for different types of media (screen, printing, etc.), and
- many advanced and sophisticated techniques for shaping the appearance of the elements.

When websites already reached an admirable level of stylization by enhancing fonts and styles of different elements, there was a need to make the content on the page dynamic; for the elements to move and rotate; all for the purpose of making a better visual impression. In order for this to be achieved, CSS transformations and animations are used. CSS describes how HTML elements will be displayed on the screen, paper, or some other media.

Since the creation of HTML, there has never been a need for it to contain tags for the website's style. However, when tags such as <font> began to be

used, and when in HTML 3.2. colour attributes were added, complications in web development have occurred. The development of large websites, where each page needs to contain fonts and colour information, has become a long and expensive process. To solve this problem, W3C creates CSS, which removes page style formatting out of the HTML document structure [4].

Style declarations are stored in an external .css file. With this approach, the appearance of the entire site can be changed by changing only one file. When a web browser reads a CSS document, it formats the HTML code according to declarations read from the CSS file.

CSS has been composed of a set of rules written within selector and declaration block [5]. The declaration block contains property (attribute) and value. The text below illustrates an example of a declaration block.

```
h1 {
      color: blue;
      font-size: 12px;
}
```

- *The selector* is an identifier (HTML tag) that determines the HTML element to which the style is intended to be applied.
- *Value* describes how the web browser will present the element to the end user. Each attribute has a set of valid values, defined by formal grammar, as well as semantic meaning.

Setting the CSS attributes to specific values is the fundamental function of the CSS language. There are more than 100 different attributes and almost an infinite number of different values. However, all attribute-value pairs are not allowed. Each attribute has a defined set of allowed values. When a value, that is not valid for a given property, is used - the declaration is considered invalid and the browser will completely ignore it.

Today, geometric transformations on web pages can be effectively rendered using a CSS3 descriptive web language which can be used within a separate file or can be implemented within HTML. In this paper, it is analytically presented that applying CSS3 standard enables adding a variety of geometric transformations and visual effects into web pages. Following the introduction, the first chapter describes 2D geometric transformations. The second chapter introduces 3D geometric transformations, with an emphasis given to the detailed description of properties that play a significant role in representing objects in 3D space. The ability to incorporate geometric transformations into a website using HTML, CSS and JavaScript is presented in the third chapter of the paper. At the end of the work, a conclusion and an attachment with the code for the created web site are provided.

1. 2D TRANSFORMATIONS

1.1. Coordinate system

Each document displayed within a browser is in fact a coordinate system whose centre (0.0) is located in the upper left corner of the viewport. The value of x coordinate rises to the right, while the y value of the coordinate grows downward. In the case of 3D transformations, z-coordinate represents the distance of the display window from the viewer. When it has a positive value, the object is closer to the observer, and when the value of the coordinate decreases, the observer moves away from the object.

When the transform property is applied to the object, a local coordinate system is created on the object, with a centre in the object centre (50% width and 50% of the height of the object), as shown in Figure 1.

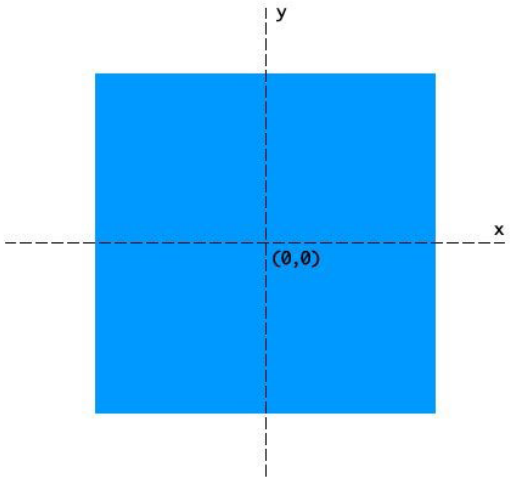

Figure 1: Local coordinate system

The centre of the coordinate system can be changed using the transform-origin property. For example, if transform-origin: 50px 70px; is indicated, the centre of the local coordinate system is 50 pixels away from the top left-hand corner of the x-axis and 70 pixels per y-axis (Figure 2). Transformation of any point within an object is done in relation to the local coordinate system.

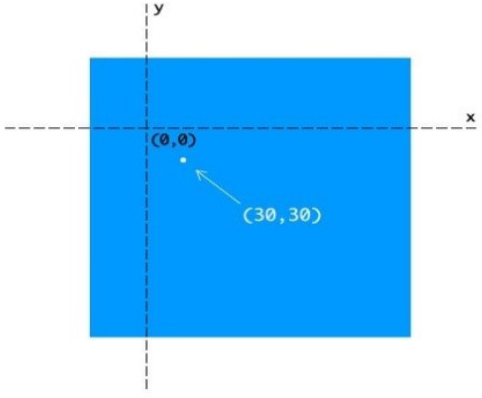

Figure 2: Change the centre of the local coordinate system of the element

If you want the maximum compatibility on different browsers, you must not forget to use the specified prefixes before each *transform* property. Currently, declarations in the text below provide support for Internet Explorer 9 ("-ms"), Chrome and Safari ("-webkit"), Firefox ("-MOSS"), Opera ("-o") and browsers that do not require a prefix, as Internet Explorer 10:

```
-ms-transform: translateX(400px);
-webkit-transform: translateX(400px);
-moz-transform: translateX(400px);
-o-transform: translateX(400px);
transform: translateX(400px);
```

1.2. 2D translation

The translate() method moves an element from its current position by x and/or y axis for a specified number of centimetres (cm), inches (inch), em, %, or pixels (px).

The following versions of this method are used:
translateX(n) - to translate an element only by *x*-axis;
translateY(n) - to translate elements only by *y*-axis;
translate(m,n) - to translate elements simultaneously by *x* and *y* axis.

In the following example, the transformation *translate(m,n)* moves the element for 50px right (by *x*-axis) and 100px down (by *y*-axis):

```
div {
  -ms-transform: translate(50px,100px); /* IE9 */
  -webkit-transform: translate(50px,100px); /* Safari */
  transform: translate(50px,100px);
}
```

To make easier to understand how this transformation actually works, the text below lists the mathematical execution of the translation matrix.

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & dx \\ 0 & 1 & dy \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (1)$$

If the transformation is:
transform:translate(50px,100px);
this means that the transformation matrix is:

$$\begin{bmatrix} 1 & 0 & 50 \\ 0 & 1 & 100 \\ 0 & 0 & 1 \end{bmatrix} \quad (2)$$

Let the object to be transcribed has a width of 200px and a height of 120px. In this case, the edge points in relation to the local coordinate system of the object have the following coordinates: *A*(-100,-60), *B*(100,-60), *C*(-100,60) and *D*(100,60).

The transformation is applied to each edge point:

$$A' = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 50 \\ 0 & 1 & 100 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} -100 \\ -60 \\ 1 \end{bmatrix} = \begin{bmatrix} -50 \\ 40 \\ 1 \end{bmatrix} \quad (3)$$

$$B' = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 50 \\ 0 & 1 & 100 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 100 \\ -60 \\ 1 \end{bmatrix} = \begin{bmatrix} 150 \\ 40 \\ 1 \end{bmatrix} \quad (4)$$

$$C' = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 50 \\ 0 & 1 & 100 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} -100 \\ 60 \\ 1 \end{bmatrix} = \begin{bmatrix} -50 \\ 160 \\ 1 \end{bmatrix} \quad (5)$$

$$D' = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 50 \\ 0 & 1 & 100 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 100 \\ 60 \\ 1 \end{bmatrix} = \begin{bmatrix} 150 \\ 160 \\ 1 \end{bmatrix} \quad (6)$$

The final result is a translated rectangular, as shown in Figure 3.

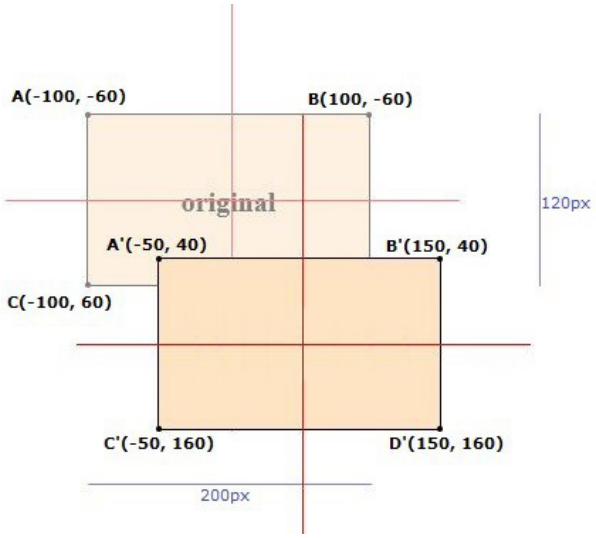

Figure 3: 2D translation

1.3. 2D rotation

The rotate() method rotates an item clockwise (positive direction) or in the opposite direction of the clockwise direction (negative direction) for the indicated number of degrees.

The following example demonstrates the rotation of the element in a positive direction for an angle of 45°.

```
div{
  -ms-transform:rotate(45deg);/* IE 9 */
  -webkit-transform:rotate(45deg);/*Safari*/
  transform:rotate(45deg);
}
```

If the rotation of the element is desired in the negative direction, the negative value of the angle of rotation will be used.

The procedure for executing the final coordinates is similar to the translation, except that in this case the rotation matrix is used in calculation, and the equation would look like this:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (7)$$

Figure 4 shows the rectangle rotated by 45°.

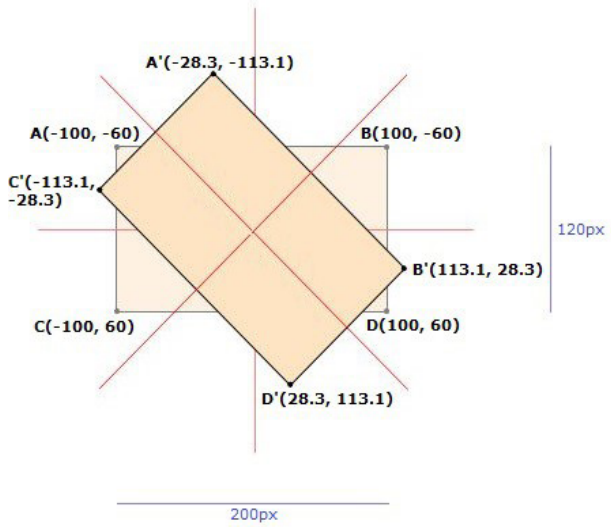

Figure 4: 2D translation

1.4. 2D scaling

The scale() method increases and/or decreases the size of an element relative to the forwarded parameters for width and height.

The following versions of this method are used:
scaleX(n) - to scale the element width (by *x*-axis);
scaleY(n) - to scale the height of the elements (by *y*-axis);
scale(m,n) - for scaling and the width and height of the elements (simultaneously by *x* and *y* axis).

In the following example, the element increases its width 2 times in relation to the original width, and the height 3 times in relation to the original height:

```
div{
   -ms-transform:scale(1.5, 1.5);/* IE 9 */
   -webkit-transform:scale(1.5, 1.5);/* Safari */
   transform:scale(1.5, 1.5);
}
```

In analogy to the above, the calculation matrix used for calculation is:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (8)$$

Figure 5 shows an example of scaled rectangle.

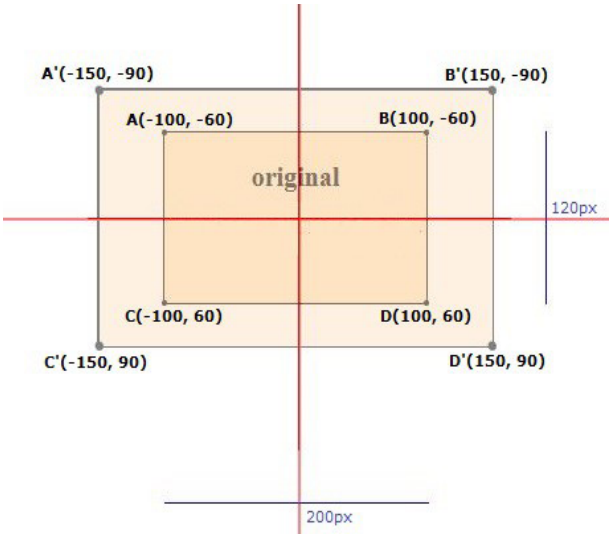

Figure 5: 2D translation

1.5. 2D matrix() method

The matrix() method combines all 2D transformations into one transformation.

The matrix() method receives 6 parameters that allow the rotation, scaling, translation, and skewing to be performed over the element:

```
div{
-ms-transform:matrix(0.9, -0.05, -0.375, 1.375, 220, 20);/* IE 9 */
-webkit-transform:matrix(0.9, -0.05, -0.375, 1.375, 220, 20);/* Safari */
transform:matrix(0.9, -0.05, -0.375, 1.375, 220, 20);
}
```

When the browser executes the transformation transform: *matrix(a, b, c, d, e, f)*; it multiplies each point of the element with a matrix:

$$\begin{bmatrix} a & c & e \\ b & d & f \\ 0 & 0 & 1 \end{bmatrix} \quad (9)$$

which is for a given example:

$$\begin{bmatrix} 0.9 & -0.375 & 220 \\ -0.05 & 1.375 & 20 \\ 0 & 0 & 1 \end{bmatrix} \quad (10)$$

If now this transformation is applied to the points of the considered rectangle, the effect will be as in Figure 6.

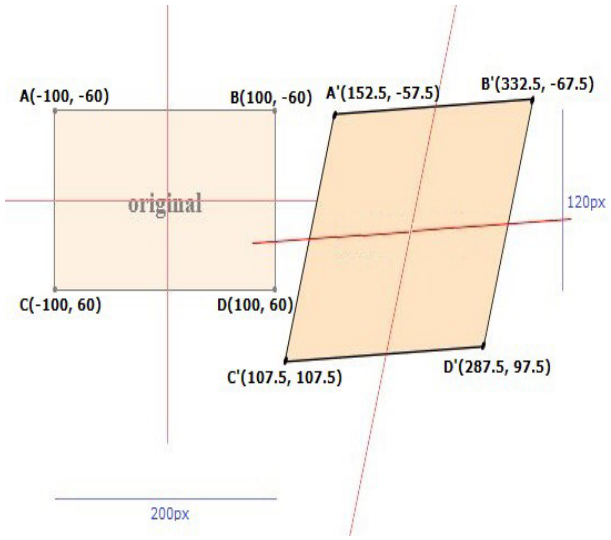

Figure 6: The usage of the *matrix()* method

This method is good from the performance point of view, because matrix multiplication is performed only once for applying more different transformations to an element. However, the basic disadvantage is that the user must, in order to determine the desired parameters, multiply the transformation matrices in the desired order itself.

1.6. Multiple transformations

Each time transformation is applied to an element, its co-ordinate system is also transformed. Therefore, the order of transformations is very important.

For example, let say it is given:
transform:rotate(45deg)translate(200px);
transform:translate(200px) rotate(45deg);

Using the above operations, two completely different transformed elements are obtained, as shown in Figure 7:

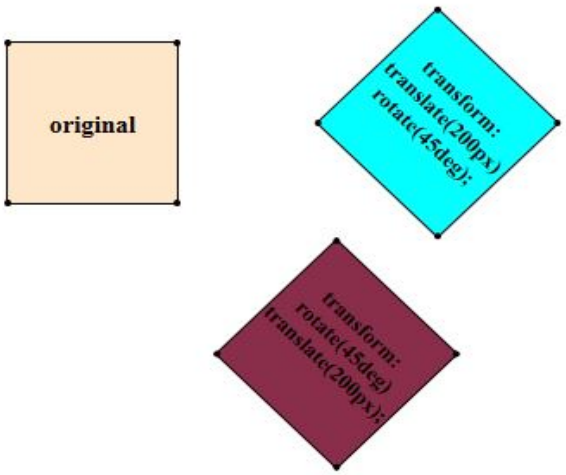

Figure 7: Multiple transformations

2. 3D TRANSFORMATIONS

Basic transformations in 3D are essentially an extension of the 2D transformations concept. However, there are some differences. Therefore, this section will provide detailed description of the properties that are of great importance when it comes to representing objects in 3D space [6].

2.1. Perspective

In order to give depth to the space, the *perspective()* property must be introduced and used. The perspective is actually the distance from the point z=0 to the observer. It can be defined in 2 ways:

transform: perspective(600px) or

perspective: 600px;

In the first instance, the perspective is given only to the specified element, while by applying the second mode the perspective it is given to all elements-children of the mentioned selector. The second way is mostly used to achieve the 3D space experience. Elements, for which the value of the perspective parameter is greater than zero (z > 0), become larger, while the elements with z <0 become smaller, i.e. they are more distant from the viewer. Part of the 3D element that is behind the viewer will not be displayed, when the coordinate of the element is greater than the value of the perspective of the element.

2.2. Perspective origin property

This property defines the position of the observer in relation to the local coordinate system of the object. This point is by default positioned in the centre of the element, i.e. it is the point (0,0) of the local coordinate system of the element, or the *perspective-origin(50%, 50%)*.

perspective-origin-x() – the position of the observer in relation to the object on the *x*-axis;
perspective-origin-y() – the position of the observer in relation to the object on the *y*-axis.

2.3. Backface visibility property

The *backface-visibility()* property defines whether the item is completely visible or not, when it is in the frontal position (front) in relation to the observer. This feature is useful when we want to determine the visibility of element's back (reverse) after applying rotation.

In Figure 8, we can see the usage of visibility properties on the cube with differently coloured pages.

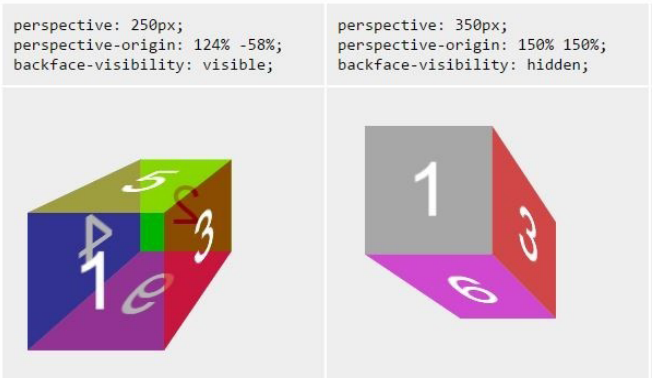

Figure 8: Application of different attribute values to a 3D object

Just as 2D transformations can be represented by matrices, so 3D transformations can be written in the corresponding matrix form. 3x3 linear transformation matrices in 3D space are represented by 4x4 matrices.

2.4. 3D translation

The *translate3d(x, y, z)* method translates elements along x, y, and z axes, for the values of the parameters specified in parentheses [7]. In addition to this method, individual methods for translating by particular axes can be used:

transform:translateX(dx);shortened from
transform:translate3d(dx,0,0);
transform:translateY(dy); shortened from
transform:translate3d(0,dy,0);
transform:translateZ(dz); shortened from
transform:translate3d(0,0,dz);

The translation matrix reads:

$$\begin{bmatrix} 1 & 0 & 0 & dx \\ 0 & 1 & 0 & dy \\ 0 & 0 & 1 & dz \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (11)$$

Figure 9 illustrates the result of the translation of the cube along all three axes using transform:translate3d(40px-,20px,50px);

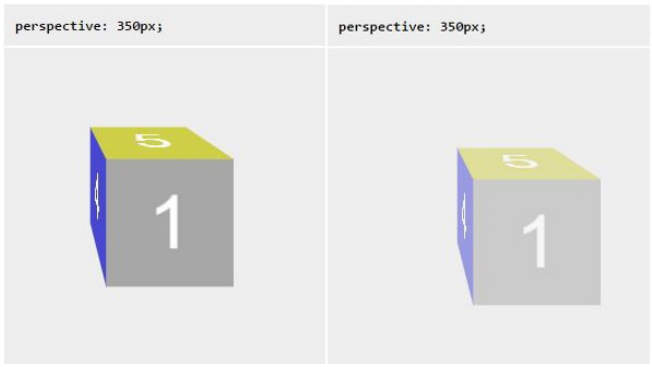

Figure 9: 3D translation

2.5. 3D scaling

The *scale3d (x, y, z)* method scales the element by x, y and z axis [8]. The input value serves as a multiplier of the existing value. This method modifies the size of the element. As the passed parameter is a vector, different dimensions can be scaled to different values. The transformation is characterized by a vector which tells how much the element is scaled and by which axis transformation is being performed.

In addition to this method, separate methods for scaling by individual axes can be used:

transform:scaleX(Sx);shortened from
transform:scale3d(Sx,1,1);
transform:scaleY(Sy); shortened from
transform:scale3d(1,Sy,1);
transform:scaleZ(Sz); shortened from
transform:scale3d(1,1,Sz);

The scaling matrix looks:

$$\begin{bmatrix} Sx & 0 & 0 & 0 \\ 0 & Sy & 0 & 0 \\ 0 & 0 & Sz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (12)$$

In Figure 10, the scaling result along the three axes is presented using method transform:scale3d(1.5,0.5,2);

Figure 10: 3D translation

2.6. 3D rotation

The *rotate3d(x, y, z, deg)* method defines a transformation that rotates an element around fixed axis without deforming it [9]. The rotation angle is defined by the input parameter, and if it is positive, the rotation is done in the clockwise direction, otherwise it is done in the opposite of clockwise direction.

The previous method allows the element's rotation around an arbitrary axis that does not have to pass through a coordinate centre. The rotation axis is determined by the normalized values of the first three parameters. In addition to this method, methods for rotating around particular coordinate axes can be used:

transform:rotateX(a); - shortened from
transform:rotate3d(1,0,0,a);
transform:rotateY(a);- shortened from
transform:rotate3d(0,1,0,a);
transform:rotateZ(a); - shortened from
transform:rotate3d(0,0,1,a);

The rotation matrices around *x*, *y*, and *z*-axis, respectively, read:

$$Rx = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha & 0 \\ 0 & -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (13)$$

$$Ry = \begin{bmatrix} \cos\alpha & 0 & -\sin\alpha & 0 \\ 0 & 1 & 0 & 0 \\ \sin\alpha & 0 & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (14)$$

$$Rz = \begin{bmatrix} \cos\alpha & \sin\alpha & 0 & 0 \\ -\sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (15)$$

If a user wants rotation around all three axes within one property, he must create an appropriate matrix for this purpose. Method *rotate3d (x, y, z, α)* presents the composition of the rotation matrix around the *x* axis, then *y* axis, and then around *z*-axis. When these matrices are correspondingly multiplied, it is obtained:

$$\begin{bmatrix} 1-2(y^2+z^2)Sq & 2(xySq-zSc) & 2(xzSq+ySc) & 0 \\ 2(xySq+zSc) & 1-2(x^2+z^2)Sq & 2(yzSq-xSc) & 0 \\ 2(xzSq-ySc) & 2(yzSq+xSc) & 1-2(x^2+y^2)Sq & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (16)$$

where:

$$Sc = \sin\left(\frac{\alpha}{2}\right)\cdot\cos\left(\frac{\alpha}{2}\right) \quad \text{and} \quad Sq = \sin^2\left(\frac{\alpha}{2}\right) \quad (17)$$

In Figure 11, a rotation result is presented using property transform:rotate3d(2,0.5,1,45deg);

Figure 11: 3D rotation

2.7. Method matrix3D()

The transformation resulting from the linear transformations composition can be realized by a matrix equal to the product of the matrix of the observed transformations in the composition. Matrix multiplication, and therefore the composition of linear transformations, is associative, but not commutative. So, for example, instead of writing:

transform: rotate3d(1, 0, 1, 45deg) translate3d(24px, 25px, 100px);

these two transformations can be written as one using the matrix3d() property:

transform:matrix3d(0.85, 0.49, 0.15, 0, -0.49, 0.70, 0.49, 0, 0.15, -0.49, 0.85, 0, 22.63, -20.32, 101-37, 1);

3. EXAMPLE

A static HTML page was created in order to demonstrate practical usage of the previously described CSS3 transformations. The following technologies have been used to build this site:

- **HTML** – HyperText Markup Language is a presentation language for creating web pages used to create the elements, the basic structure and page layout;
- **CSS** – Cascade Style Sheet, a style sheet language that allows you to give the proper look to the elements and perform the necessary transformations [7];
- **JavaScript** – a programming language that serves to manipulate the elements on the client side, after the web page is loaded. In this paper, it has been used to change displayed CSS selectors and apply different properties over them.

At the beginning, a cube is displayed, with the backface visibility feature turned on, which can be seen in Figure 12.

Figure 12: Elements arranged in the shape of a cube, backface visibility set to value 'visible'

The cube is presented as a combination of rotated and translated squares. Also, one small cube can be seen inside the main cube. Both cubes have a certain value of the *backface-visibility* attribute in order to better illustrate content that their pages are hiding. By clicking the Change shape button, the transformation of the elements gives the shape of the ring, as shown in Figure 13.

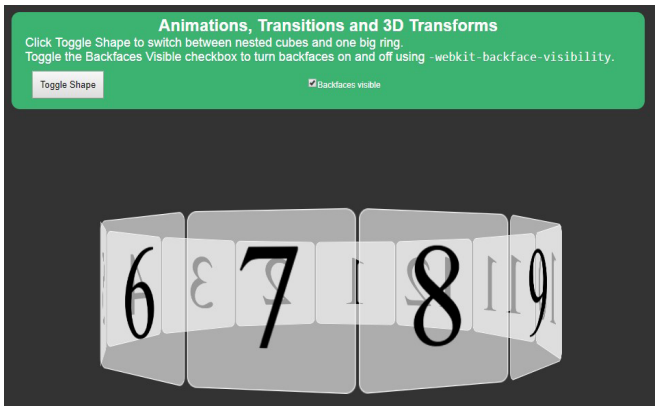

Figure 13: Elements arranged in the form of a ring, background visibility applied

In addition to the option of changing the shape of the main element (cube/ring), it is also possible to turn off the transparency of the element, i.e. that only elements that at a given moment face the user can be seen on the screen [10], [11]. This effect is achieved if the *backface-visibility* is turned off, and the ring then gets the appearance in Figure 14.

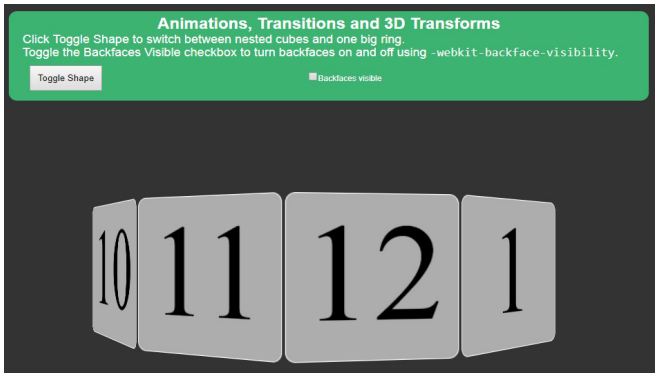

Figure 14: Elements arranged in the form of a ring, background visibility turned off

To create such an application, it was first necessary to create the appropriate HTML structure:

```
<div class="controls">
<h1>Animations, Transitions and 3D Transforms</h1>
<p>Click Toggle Shape to switch between nested cubes
and one big ring.</p>
<p>Toggle the Backfaces Visible checkbox to turn back-
faces on and off using <code>-webkit-backface-visibili-
ty</code>.</p>
<div><button onclick="toggleShape()">Toggle
Shape</button>
</div>
<div><input type="checkbox" id="backfaces" on-
click="toggleBackfaces()" checked>
<label for="backfaces">Backfaces visible</label></
div>
</div>
<div id="container">
<div id="stage">
<div id="shape" class="cube backfaces">
<div class="plane one">1</div>
<div class="plane two">2</div>
<div class="plane three">3</div>
<div class="plane four">4</div>
<div class="plane five">5</div>
<div class="plane six">6</div>
<div class="plane seven">7</div>
<div class="plane eight">8</div>
<div class="plane nine">9</div>
<div class="plane ten">10</div>
<div class="planeeleven">11</div>
<div class="planetwelve">12</div>
</div>
</div>
</div>
```

After the necessary elements were created, it was needed to put them in the right position and add them an animation which allows constant rotation, therefore CSS language was used. The code for this application segment is listed in the Appendix.

Finally, any changes that are made on client site (each click on a particular button/option) are defined using the Java Script programming language:

```
<script type="text/javascript" charset="utf-8">
function toggleShape()
{
    var shape = $('#shape);
shape.toggleClass('ring');
shape.toggleClass('cube');
}

function toggleBackfaces()
{
    var shape = $('#shape);
shape.toggleClass('backfaces');
}
</script>
```

4. CONCLUSION

Within this paper, the CSS language concepts and its role in the development of websites were analysed. A detailed description of the way CSS transformations work is given. An example of the created web page presents some of the basic advantages of using CSS3 standards, such as simplicity, platform independence, easy way of change, rapid development, etc. The most important feature of CSS3 standard is the ability to improve the appearance of the website through the simple integration of 2D and 3D images, animations, and videos into web content.

The disadvantage of using CSS transformations on the web is that CSS transformations are still in the experimental phase, and therefore require a usage of vendor prefixes in order to enable proper work on different browsers. Consequently, if you want to have cross-browser applications, it is necessary to constantly check the vendor prefixes and test the application on different browsers.

Today there are many incentives to add vendor prefixes. There are tools that allow developers to "paste" their own CSS code and get the code with the necessary vendor prefixes, which can then be added to a specific file. Also, there are many free scripts or plug-ins in the form of JavaScript files, which after being included in the code, after page loading, create new CSS files with specified vendor prefixes from the existing ones. However, this is not an ideal solution because it slows down the page loading process, and the problem remains with the inline and embedded CSS code (code written within the style attributes and style tags in HTML files), because that code won't be changed after the script's execution, since it is not explicitly written in a separate CSS file.

Based on all of the above, it can be concluded that the advantages of CSS3 standard prevail over its shortcomings. In this paper, it has been analytically demonstrated that CSS3 as a descriptive language allows the addition of various geometric transformations to web pages. Also, it is given an overview of the features this standard provides when handling geometric content. The designer is able to create simple, rich media content pages that are quickly loaded and have an impressive look. So if a web developer wants to make the website look more attractive, CSS3 transformations are the right choice. The next step in analytical scientific research is the comparative study of CSS3 features for presenting more complex geometric shapes and improving presentation in different web browsers.

ANNEX

```
#container {
    -webkit-perspective: 800;
    -webkit-perspective-origin: 50% 225px;
}

#stage {
    -webkit-transition: -webkit-transform 2s;
    -webkit-transform-style: preserve-3d;
}

#shape {
    position: relative;
    top: 120px;
    height: 200px;
    width: 200px;
    -webkit-transform-style: preserve-3d;
-webkit-animation: spin 8s infinite linear;
}

.plane {
    position: absolute;
    height: 200px;
    width: 200px;
    border: 1px solid white;
    -webkit-border-radius: 12px;
    text-align: center;
    font-size: 124pt;
    color: black;
    background-color: rgba(195,144,148,0.5);
    -webkit-transition: -webkit-transform 2s, opacity 2s;
    -webkit-backface-visibility: hidden;
}

#shape.backfaces .plane {
    -webkit-backface-visibility: visible;
}

#shape  {
    -webkit-animation: spin 8s infinite linear;
}

#shape.ring{
 -webkit-animation: spin 15s infinite linear;
}

@-webkit-keyframes spin {
    from { -webkit-transform: rotateY(0); }
    to   { -webkit-transform: rotateY(-360deg); }
 }

 /* ---------- cube ---------- */

.cube> .one {
      -webkit-transform: scale3d(1.2, 1.2, 1.2)   rotateX(-
90deg) translateZ(100px);
}
 .cube> .two {
       -webkit-transform: scale3d(1.2, 1.2, 1.2) transla-
teZ(100px);
}
 .cube> .three {
         -webkit-transform: scale3d(1.2, 1.2, 1.2) rotateY(-
90deg) translateZ(100px);
}
 .cube> .four {
       -webkit-transform: scale3d(1.2, 1.2, 1.2) rotateY(-
180deg) translateZ(100px);
}
.cube> .five {
         -webkit-transform: scale3d(1.2, 1.2, 1.2) rotat-
eY(-90deg) translateZ(100px);
}
```

```
.cube> .six {
        -webkit-transform: scale3d(1.2, 1.2, 1.2) rotateX(-90deg) translateZ(100px) rotate(180deg);
}
 .cube> .seven {
      -webkit-transform: scale3d(0.8, 0.8, 0.8) rotateX(-90deg) translateZ(100px) rotate(180deg);
 }
 .cube> .eight {
       -webkit-transform: scale3d(0.8, 0.8, 0.8) translateZ(100px);
 }
 .cube> .nine {
      -webkit-transform: scale3d(0.8, 0.8, 0.8) rotateY(-90deg) translateZ(100px);
 }
 .cube> .ten {
      -webkit-transform: scale3d(0.8, 0.8, 0.8) rotateY(-180deg) translateZ(100px);
 }
 .cube>.elven {
        -webkit-transform: scale3d(0.8, 0.8, 0.8) rotateY(-90deg) translateZ(100px);
 }
 .cube> .twelve {
        -webkit-transform: scale3d(0.8, 0.8, 0.8) rotateX(-90deg) translateZ(100px);
 }

/* ---------- ring ------------- */
 .ring> .one {
     -webkit-transform: translateZ(380px);
 }
 .ring> .two {
     -webkit-transform: rotateY(30deg) translateZ(380px);
 }
 .ring> .three {
     -webkit-transform: rotateY(60deg) translateZ(380px);
 }
  .ring> .four {
     -webkit-transform: rotateY(90deg) translateZ(380px);
 }
 .ring> .five {
    -webkit-transform: rotateY(120deg) translateZ(380px);
}
 .ring> .six {
    -webkit-transform: rotateY(150deg) translateZ(380px);
}
 .ring> .seven {
    -webkit-transform: rotateY(180deg) translateZ(380px);
}
.ring>.eight {
    -webkit-transform: rotateY(210deg) translateZ(380px);
}
.ring> .nine {
   -webkit-transform: rotateY(-120deg) translateZ(380px);
}
.ring> .ten {
    -webkit-transform: rotateY(-90deg) translateZ(380px);
}
.ring> .eleven {
    -webkit-transform: rotateY(300deg) translateZ(380px);
}
.ring> .twelve {
    -webkit-transform: rotateY(330deg) translateZ(380px);
}
```

REFERENCES

[1] J. Duckett, HTML and CSS: Design and Build Websites (1st Edition),John Wiley and Sons, 2011

[2] HTML/CSS Tutorial http://www.w3schools.com/ (accessed 03/02/2016)

[3] V. Simovic, M. Varga, R. Svetlacic, Animation of Objects on the Website by Application of CSS3 Language,International Journal of Computer and Information Engineering, Vol:11, No:12, 2017

[4] CSS Basics http://www.cssbasics.com/ (accessed 13/04/2016)

[5] Creating 3D Worlds with HTML and CSS http://keithclark.co.uk/articles/creating-3d-worlds-with-html-and-css/ (accessed 19/05/2016)

[6] P. Yadav, P. N. Barwal, Designing Responsive Websites Using HTML and CSS, International Journal of Scientific&Technology ResearchVolume 3, Issue 11, November 2014

[7] CSS Transforms https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Transforms (accessed 07/06/2016)

[8] The Art of Web http://www.the-art-of-web.com/css/3d-transforms/ (accessed 21/10/2016)

[9] CSS3 Transforms http://www.w3.org/TR/css3-transforms/ (accessed 05/07/2017)

[10] CSS3 Transitions and Transforms From Scratch https://webdesign.tutsplus.com/(accessed 12/09/2017)

[11] CSS3 Tutorial http://www.w3big.com/css3/(accessed 11/01/2018)

BIOGRAPHY

Emir Skejić was born on 16 January 1977 in Doboj. He received his doctorate in technical science from the field of electrical engineering on October 26, 2007 at the Faculty of Electrical Engineering, University of Tuzla. Since 2001, he has been employed at the Faculty of Electrical Engineering at the University of Tuzla, where he is currently the associate professor in the field of Computer and Information Science.

Dženita Vejsilović was born on June 27, 1992 in Tuzla. After completing the Common Grammar School in Tuzla, she attended the Faculty of Electrical Engineering at the University of Tuzla, and obtained her Bachelor in Electrical Engineering on March 7, 2016. Since 1 June 2015 she has been working as a web developer at H&H Inc. Development and Research Company in Tuzla. She is also studying for a master's degree at the Faculty of Electrical Engineering, University of Tuzla, in the field of Computer and Information Science.