Grafika 2D z użyciem HTML Canvas

February 26, 2019

1 Introduction

Celem jest użycie 2D Graphics API w celu tworzenia obrazów na stronach WWW.

Większość nowoczesnych przeglądarek obsługuje interfejs API grafiki 2D, który można wykorzystać do tworzenia obrazów na stronie internetowej. Interfejs API jest implementowany za pomocą JavaScript, języka programowania po stronie klienta.

2 Podstawowa składnia JavaScript

- zmienne nie są typowane;

- deklaracja zmiennych za pomocą słowa var;

```
var x, y;
var name = "David";
```

- jest możliwa deklaracja kilkukrotnia; taka deklaracja mówi, że zmienna istnieje.

- JavaScript pozwala również używać zmiennych bez ich deklarowania. Jednak nie jest to dobry pomysł.

- Możesz zapobiec użyciu niezadeklarowanych zmiennych, a także niektórych innych niebezpiecznych praktyk, umieszczając następujące oświadczenie na początku programu:

```
"use strict";
```

- Mimo że zmienne nie mają typów, oni mają wartość.

- Wartością może być:
  - liczba,
  - łańcuch znaków,
  - boolean,
  - obiekt,
  - funkcja.
- Zmienna, która nigdy nie miała przypisanej wartości, ma specjalną wartość **undefined**. (Fakt, że funkcja może być użyta jako wartość danych może być dla ciebie zaskoczeniem, więcej o tym później.)

- Możesz określić typ wartości za pomocą operatora **typeof**: Wyrażenie typeof x zwraca ciąg znaków, który mówi typ wartości x. Łańcuch może być
  - "number",
  - "string",
  - "boolean",
  - "object",
  - "function" lub
  - "undefined".

  Zauważ, że typeof zwraca "obiekt" dla obiektów dowolnego typu, w tym tablic. Ponadto, `typeof null` to "object".

- "string" jest uważany za pierwotny typ danych, a nie za typ obiektu.

- Nie ma ścisłego rozróżnienia między liczbami całkowitymi a liczbami rzeczywistymi.

- W rzeczywistości JavaScript wykonuje wiele niejawnych konwersji, których możesz nie oczekiwać. Na przykład, podczas porównywania liczby do łańcuchu znaków za pomocą operatora ==, JavaScript spróbuje przekonwertować ciąg na liczbę. Tak więc wartość `17 == "17"` jest prawdziwa.

- 166/5000 JavaScript ma operatorów `===` i `!==` podobnych do `==` i`!=` z tym, że nigdy nie wykonują konwersji typów na swoich operandach. Więc, na przykład `17 === "17"` jest fałszywe.

- JavaScript nie ma rzutowania typów, tak jak istnieje w Javie. Można jednak używać funkcji `Number(), String() i Boolean()` jako funkcji konwersji. Na przykład,

  ```
  x = liczba (y);
  ```

  spróbuje przekonwertować `y` na liczbę. Możesz to zastosować, na przykład, gdy `y` jest łańcuchem znaków. Jeśli konwersja nie powiedzie się, wartością `x` będzie `NaN`, specjalna wartość oznaczająca "Not a Number". Funkcja `Number` konwertuje pusty ciąg na zero.

- Funkcje matematyczne w JavaScript są definiowane w obiekcie `Math`, który jest podobny do klasy Math w Javie: `Math.sin(x), Math.cos(x), Math.abs(x), and Math.sqrt(x)`

- Struktury sterujące JavaScript są prawie takie same jak w Javie lub C, w tym `if, while, for, do..while, switch`.

- JavaScript ma instrukcję `try..catch` do obsługi wyjątków, która jest podobna do języka Java, ale ponieważ zmienne są nietypowane, istnieje tylko jeden blok `catch` i nie deklaruje on typu dla wyjątku. (Oznacza to, że mamy `"catch (e)"` zamiast "catch (Exception e)".) Błąd można wygenerować za pomocą instrukcji `throw`. Można podać dowolny typ wartości. Możesz na przykład wygenerować ciąg znaków reprezentujący komunikat o błędzie:

  ```
  throw "Przepraszamy, ta wartość jest nieokreślona";
  ```

- Funkcje w JavaScript są definiowane za pomocą słowa kluczowego `function`. Ponieważ zmienne są bez typów, nie deklaruje się typu zwracanego, a parametry nie mają zadeklarowanych typów. Oto typowa definicja funkcji:

  ```
  function square (x) {
  return x * x;
  }
  ```

- JavaScript nie wymaga, żeby liczba parametrów w wywołaniu funkcji była dopasowana do liczby parametrów w definicji funkcji. Jeśli w wywołaniu funkcji podano zbyt mało parametrów, to dodatkowe parametry w definicji funkcji przyjmują wartość `undefined`. Możesz to sprawdzić w funkcji, sprawdzając, czy typ parametru jest "undefined". Może to być dobry powód, aby to zrobić: Pozwala to mieć opcjonalne parametry. Na przykład rozważ

```
funkcja multiple (str, count) {
      if (typeof count == "undefined") {
          count = 2;
      }
      var i;
      var copies = "";
      for (i = 0; i <count; i ++)
        copies + = str;
      }
      return copies;
}
```

- Możesz także podać dodatkowe wartości w wywołaniu funkcji. Wszystkie podane wartości są zawsze dostępne w funkcji w specjalnej zmiennej o nazwie `arguments`, która jest w istocie tablicą. Na przykład umożliwia to zapisanie funkcji sumującej, która przyjmuje dowolną liczbę wartości wejściowych:

```
function sum () {
     var i;
     var total = 0;
     for (i = 0; i <arguments.length; i ++) {
         total + = arguments[i];
     }
     return total;
}
```

  Dzięki tej definicji możesz wywołać `sum(2,2)`, `sum(1,2,3,4,5)`, a nawet `sum()`. (Wartość ostatniego wywołania funkcji wynosi zero.)

- Funkcje w JavaScript to są "obiekty klasy". Oznacza to, że funkcje są traktowane jako zwykłe wartości danych i możesz robić z nimi takie rzeczy, które robisz z danymi: przypisuj je do zmiennych, przechowuj je w tablicach, przekazuj jako parametry do funkcji, zwracaj je z funkcji. W rzeczywistości bardzo często robimy wszystkie te rzeczy!

- Zdefiniowanie funkcji za pomocą definicji takiej jak w przykładach pokazanych powyżej, jest prawie jak przypisanie funkcji do zmiennej. Na przykład, biorąc pod uwagę powyższą definicję funkcji `sum`, możesz przypisać `sum` do zmiennej lub przekazać ją jako parametr. A jeśli wartość zmiennej jest funkcją, możesz użyć tej zmiennej tak, jak chcesz użyć nazwy funkcji, aby wywołać funkcję. To znaczy, jeśli

  ```
  var f = sum;
  ```

  wtedy możesz wywołać `f(1,2,3)`, a będzie to dokładnie jak `sum(1,2,3)`.

- istnieje sposób zapisywania wartości danych funkcji w miejscu, w którym jest to potrzebne, bez nadawania jej nazwy lub definiowania za pomocą standardowej definicji funkcji. Takie funkcje nazywa się **"funkcjami anonimowymi"**. Składnia wygląda jak definicja funkcji bez nazwy. Oto, na przykład, anonimowa funkcja jest tworzona i przekazywana jako pierwszy parametr do funkcji o nazwie `setTimeout`:

  ```
  setTimeout (function () {
       alert ("Time's Up!");
  }, 5000);
  ```

  Wykonanie tej samej czynności bez funkcji anonimowych wymagałoby zdefiniowania standardowej nazwanej funkcji, która będzie używana tylko raz:

  ```
  function alertFunc () {
       alert ("Time's Up!");
  }
  setTimeout (alertFunc, 5000);
  ```

- W języku C funkcje mogą być przypisane do zmiennych i przekazywane jako parametry do funkcji. Jednak w C nie ma anonimowych funkcji. Funkcje anonimowe są jedną z charakterystycznych cech JavaScriptu. Coś podobnego zostało dodane do Java 8 w postaci "obliczeń lambda".

- **Tablice** tworzy się tak:

  ```
  var tablica =[1, 2, 3];
  var tablicaPusta=[];
  ```

  lub

  ```
  var tablica=new Array(1,2,3);
  var tablicaPusta=new Array();
  ```

Tablice w JavaScripcie są dynamiczne, co oznacza, że nie trzeba deklarować ich wielkości. Tablica powiększa się automatycznie.

```
var tablica=[1,2,3];
tablica[4]=5;
```

jest równoważne

```
var tablica=[1,2,3,undefined,5];
```

- JavaScript ma **obiekty**, ale nie ma dokładnie klas, przynajmniej nie w tym sensie, w jakim działa Java lub C ++. Obiekt jest zasadniczo **zbiorem par klucz / wartość**, gdzie klucz jest nazwą, podobnie jak nazwa instancji lub nazwa metody w języku Java, która ma przypisaną wartość. Termin "zmienna instancji" zwykle nie jest używany w JavaScript; preferowanym terminem jest "property(własność)".

  Wartością właściwości obiektu może być zwykła wartość danych lub funkcja (która jest tylko innym typem wartości danych w kodzie JavaScript). Możliwe jest utworzenie obiektu jako listy par klucz / wartość, zawartych przez { i }. Na przykład,

```
var pt = {x: 17, y: 42};

var ajaxData = {
     url: "http://some.place.org/ajax.php",
     data: 42,
     onSuccess: function () {alert ("zadzia␣la␣lo!"); },
     onFailure: function (error) {alert ("Niestety,
                         to się nie udalo:" + error); }
};
```

- Chociaż JavaScript nie ma klas, ma **konstruktory**, które można wywoływać za pomocą operatora `new` w celu tworzenia obiektów. Na przykład,

```
var now = new Date();
```

  wywołuje konstruktora `Date()`, który jest standardową częścią JavaScript. Gdy zostanie wywołana bez parametrów, nowa funkcja `Date()` tworzy obiekt reprezentujący bieżącą datę i godzinę.

- Konstruktor jest napisany jak zwykła funkcja; zgodnie z konwencją nazwa funkcji konstruktora zaczyna się od dużej litery. W pewnym sensie konstruktor faktycznie definiuje klasę. `Date`, na przykład, jest często określana jako klasa. Możliwe jest napisanie funkcji, która może być użyta jako konstruktor. Na przykład zobaczmy, jak zdefiniować klasę do reprezentowania punktów w 2D:

```
function Point2D (x, y) {
    if (typeof x! = "undefined") {
        this.x = x;
    }
    else {
        this.x = 0;
    }
    if (typeof y! = "undefined") {
        this.y = y;
    }
    else {
        this.y = 0;
    }
    this.move = function (dx, dy) {
        this.x = this.x + dx;
        this.y = this.y + dy;
    }
}
```

Po wywołaniu z operatorem `new` ta funkcja tworzy obiekt, który ma właściwości `x, y, move`. Wartość `move` jest funkcją. Oto przykład:

```
var p1 = new Point2D (17,42);
var p2 = new Point2D ();
```

- Definicja `Point2D` używa specjalnej zmiennej `this`, która odnosi się do konstruowanego obiektu, dzięki czemu `this.x` i `this.y` odnoszą się do właściwości tego obiektu. W definicji `move` słowo `this` odnosi się do obiektu, który jest używany do wywoływania funkcji. Oznacza to, że podczas wykonywania instrukcji `pt.move (a, b)`, odnosi się to do `pt`. (Jest to nieco inne niż znaczenie tego w Javie. Używanie tego jest obowiązkowe w JavaScript; nigdy nie można skrócić "this.x" jako "x").

3 JavaScript na stronach WWW

Trzy sposoby użycia JavaScript na stronach WWW:

1. kod pomiędzy elementami <script>

```
<script>

    // ... JavaScript code goes here ...

</script>
```

lub

```
<script type="text/javascript">

    // ... JavaScript code goes here ...

</script>
```

w HTML5 wartość `"text/javascript"` jest domyślną.

2. przez plik

   ```
   <script src="filename.js"></script>
   ```

3. przez event handler

   ```
   <h1 onclick="doClick()">My Web Page</h1>
   ```

JavaScript dla stron internetowych ma kilka standardowych funkcji, które pozwalają na interakcję z użytkownikiem za pomocą okien dialogowych.

- Najprostszym z nich jest `alert(wiadomość)`, które wyświetla komunikat użytkownikowi w wyskakującym oknie dialogowym, z przyciskiem "OK", który użytkownik może kliknąć, aby zamknąć wiadomość.

- W `prompt(pytanie)` wyświetli się pytanie w oknie dialogowym wraz z polem wprowadzania, w którym użytkownik może wprowadzić odpowiedź. Funkcja `prompt` zwraca odpowiedź użytkownika jako jego wartość zwracaną. Ten rodzaj okna dialogowego zawiera przycisk "OK" i przycisk "Cancel". Jeśli użytkownik kliknie "Cancel", zwracana wartość z podpowiedzi jest zerowa. Jeśli użytkownik kliknie "OK", wartość zwracana jest zawartością pola wejściowego, którym może być pusty ciąg znaków.

- Funkcja `confirm(pytanie)` wyświetla pytanie w oknie dialogowym wraz z przyciskami "OK" i "Cancel". Wartość zwracana to `true` lub `false`, w zależności od tego, czy użytkownik kliknął "OK" czy "Cancel".

Przykład użycia okien dialogowych

```
alert("I will pick a number between 1 and 100.\n"
        + "Try to guess it!");

do {

    var number = Math.floor( 1 + 100*Math.random() );
    var guesses = 1;
```

```
    var guess = Number( prompt("What's your guess?") );
    while (guess != number ) {
        if ( isNaN(guess) || guess < 1 || guess > 100 ) {
            guess = Number( prompt("Please enter an integer\n" +
                                "in the range 1 to 100") );
        }
        else if (guess < number) {
            guess = Number( prompt("Too low.  Try again!") );
            guesses++;
        }
        else {
            guess = Number( prompt("Too high.  Try again!") );
            guesses++;
        }
    }
    alert("You got it in " + guesses + " guesses.");

} while ( confirm("Play again?") );
```

4 Interakcja ze stroną WWW

Po załadowaniu strony internetowej wszystko na stronie jest zakodowane w strukturze danych zdefiniowanej przez DOM, do której można uzyskać dostęp z JavaScriptu jako zbiór obiektów. Istnieje kilka sposobów na uzyskanie dostępu do tych obiektów, ale używamy tu tylko jeden: `document.getElementById`.

Każdy element na stronie internetowej może mieć atrybut `id`. Na przykład:

```
<img src="somepicture.jpg" id="pic">
```

lub

```
<h1 id="mainhead">My Page</h1>
```

Dowolny element jest reprezentowany przez obiekt **DOM**. Jeśli element ma identyfikator, można uzyskać odwołanie do odpowiedniego obiektu DOM, przekazując identyfikator do funkcji `document.getElementById`. Na przykład:

```
var image = document.getElementById ("pic");
var heading = document.getElementById ("mainhead");
```

Przykłady wykorzystania w JavaScript

```
heading.innerHTML = "Best Page Ever!";
image.src = "anotherpicture.jpg";
heading.color = "red";
heading.fontSize = "150%";
```

Użycie elementów input

```
<input type="text" id="textin">

<select id="sel">
   <option value="1">Option 1</option>
   <option value="2">Option 2</option>
   <option value="3">Option 3</option>
</select>

<input type="checkbox" id="cbox">
```

przez JavaScript

```
var textin = document.getElementById("textin");
var sel = document.getElmenntById("sel");
var checkbox = document.getElementById("cbox");
```

Właściwości `textin.value` i `sel.value` reprezentują bieżące wartości tych elementów.

Możesz ustawić kod JavaScript, który zostanie wywołany w odpowiedzi na **zdarzenie**. Typowym sposobem wykonania tej czynności jest dodanie modułu obsługi zdarzenia `onload` do znacznika `<body>`:

```
<body onload = "init ()">
```

Obsługa zdarzeń. Trzy sposoby:

```
checkbox.onchange = checkBoxChanged;

checkbox.onchange = function() { alert("Checkbox changed"); };

checkbox.addEventListener( "change", checkBoxChanged, false );
```

5 Kontekst 2D Graphics

Element strony WWW

```
<canvas width="800" height="600" id="theCanvas"></canvas>
```

Kontekst graficzny odgrywa taką samą rolę w interfejsie canvas API, że zmienna typu Graphics2D jest w Javie. Typowym punktem wejścia jest

```
canvas = document.getElementById("theCanvas");
graphics = canvas.getContext("2d");
```

Pierwszy wiersz otrzymuje referencję do elementu canvas na stronie internetowej, używając jego id. Drugi wiersz tworzy kontekst graficzny dla tego elementu canvas.

Przykład minimalnej strony WWW z użyciem canvasu

```
<!DOCTYPE html>
<html>
<head>
<title>Canvas Graphics</title>
<script>
    var canvas;    // DOM object corresponding to the canvas
    var graphics;  // 2D graphics context for drawing on the canvas

    function draw() {
           // draw on the canvas, using the graphics context
        graphics.fillText("Hello World", 10, 20);
    }

    function init() {
        canvas = document.getElementById("theCanvas");
        graphics = canvas.getContext("2d");
        draw();  // draw something on the canvas
    }
</script>
</head>
<body onload="init()">
    <canvas id="theCanvas" width="640" height="480"></canvas>
</body>
</html>
```

Funkcje do rysowania prostokątów oraz tekstu

```
graphics.fillRect(x,y,w,h)

graphics.strokeRect(x,y,w,h)

graphics.clearRect(x,y,w,h)

graphics.fillText(str,x,y)

graphics.strokeText(str,x,y)
```

Ścieżki (Path) mogą zawierać **linie, krzywe Beziera i łuki kołowe**. Oto najczęściej używane funkcje do pracy ze ścieżkami:

```
graphics.beginPath()

graphics.moveTo(x,y)

graphics.lineTo(x,y)

graphics.bezierCurveTo(cx1,cy1,c2x,cy2,x,y)
```

```
graphics.quadraticCurveTo(cx,cy,x,y)

graphics.arc(x,y,r,startAngle,endAngle)

graphics.closePath()
```

Polecenia `graphics.fill()` i `graphics.stroke()` służą do wypełniania i przesuwania bieżącej ścieżki.

Przykład

```
graphics.beginPath();          // start a new path
graphics.moveTo(100.5,200.5);  // starting point of the new path
graphics.lineTo(300.5,200.5);  // add a line to the point (300.5,200.5)
graphics.stroke();             // draw the line
```

Obrys i wypełnienie(Stroke and fill)

```
graphics.lineWidth = 2.5;  // Change the current width

graphics.lineCap = "round"; //for endpoints it can be set to "round", "square", or "butt"

graphics.lineJoin = "round"; //for joins, it can be "round", "bevel", or "miter"


// Ustalenie koloru

graphics.fillStyle = "rgb(200,200,255)"; // light blue

graphics.strokeStyle = "#0070A0"; // a darker, greenish blue

// Użycie gradientu

var lineargradient = graphics.createLinearGradient(420,420,550,200);
lineargradient.addColorStop(0,"red");
lineargradient.addColorStop(0.5,"yellow");
lineargradient.addColorStop(1,"green");
graphics.fillStyle = lineargradient;  // Use a gradient fill!


// ustalenie czcionki zgodnie z CSS; domyślna jest "10px sans-serif"
graphics.font = "2cm monospace";  // the size is in centimeters
graphics.font = "bold 18px sans-serif";
graphics.font = "italic 150% serif";   // size is 150% of the usual size
```

Przekształcenia Trzy przekształcenia geometryczne

```
graphics.scale(sx,sy)

graphics.rotate(angle)

graphics.translate(tx,ty)

graphics.transform(a,b,c,d,e,f) // apply the transformation x1 = a*x + c*y + e,
                                //and y1 = b*x + d*y + f.

graphics.setTransform(a,b,c,d,e,f) // discard the current transformation,
                   //and set the current transformation to be
                   // x1 = a*x + c*y + e, and y1 = b*x + d*y + f.
```

Nie ma transformacji ścinania, ale możesz zastosować ścinanie jako transformację ogólną. Na przykład dla poziomego ścinania o współczynniku ścinania 0,5, użyj

```
graphics.transform (1, 0, 0.5, 1, 0, 0)
```

Kontekst graficzny (w tym przekształcenia, kolory bieżące itp) może być zachowany do steku

```
graphics.save() // push a copy of the current state of the graphics context,
                //including the current transformation, onto the stack.
graphics.restore() // remove the top item from the stack,
                //containing a saved state of the graphics context,
                // and restore the graphics context to that state.
```

Literatura

Uwaga!

Teoretyczne podstawy przekształceń geometrycznych umieszczone w prezentacji wykładu https://drive.google.com/drive/folders/0B0-jI5UeRsjcVExBTExOcXNlcGc

Przewodnik JavaScript dla HTML5 https://pdf.helion.pl/htm5rg/htm5rg.pdf

W języku angielskim

- książka interakcyjna: HTML Canvas Graphics http://math.hws.edu/graphicsbook/c2/s6.html (rozdział 2.6)

- podstawy języka JavaScript http://math.hws.edu/graphicsbook/a1/s3.html

6 Zadania

1. Plik `Lab2Ex1.html` proponuje rozszeżenia do standardowych funkcji rysowania HTML Canvas.

```
graphics.strokeLine(x1,y1,x2,y2) - draw the line segment from (x1,y1) to (x2,y2).
graphics.strokeCircle(x,y,r) - stroke the circle with center (x,y) and radius r.
graphics.strokeOval(x,y,rx,ry) - stroke the oval with center (x,y),
                            horizontal radius rx, and vertical radius ry.
graphics.strokePoly(x1,y1,x2,y2,x3,y3,...) - stroke the polygon with vertices
                         (x1,y1), (x2,y2), (x3,y3), ...
graphics.fillCircle(x,y,r) - fill the circle with center (x,y) and radius
graphics.fillOval(x,y,rx,ry) - fill the oval with center (x,y),
                         horizontal radius rx, and vertical radius ry.
graphics.fillPoly(x1,y1,x2,y2,x3,y3,...) - fill the polygon with
                            vertices (x1,y1), (x2,y2), (x3,y3), ...
```

Narysować obraz zgodnie z wariantem zadania (patrz Fig. 1) (używając zarówno standardowe jak i niestandardowe funkcje rysowania).

2. W pliku `Lab2Ex2.html` program domyślnie rysuje szereg kwadratów. Stworzyć narzędzia pozwalające na wykonywanie czynności

   - "czyszczenie" canvasu - Clear button:

     ```
     <button id="clearButton">Clear</button>
     ```

     (Hint! Przy inicializacji musi być

     ```
     document.getElementById("clearButton").onclick = doClear;
     ```

)
   - dodanie jednego nowego koloru do elementu <select>. Implementować nowy kolor przez funkcję `doMouseMove`.
   - opracowanie nowego narzędzia - rysowania szeregu wielokątów (zgodnie z wariantem zadania). Opcja ma być dostępna przez nowy element `<select>`

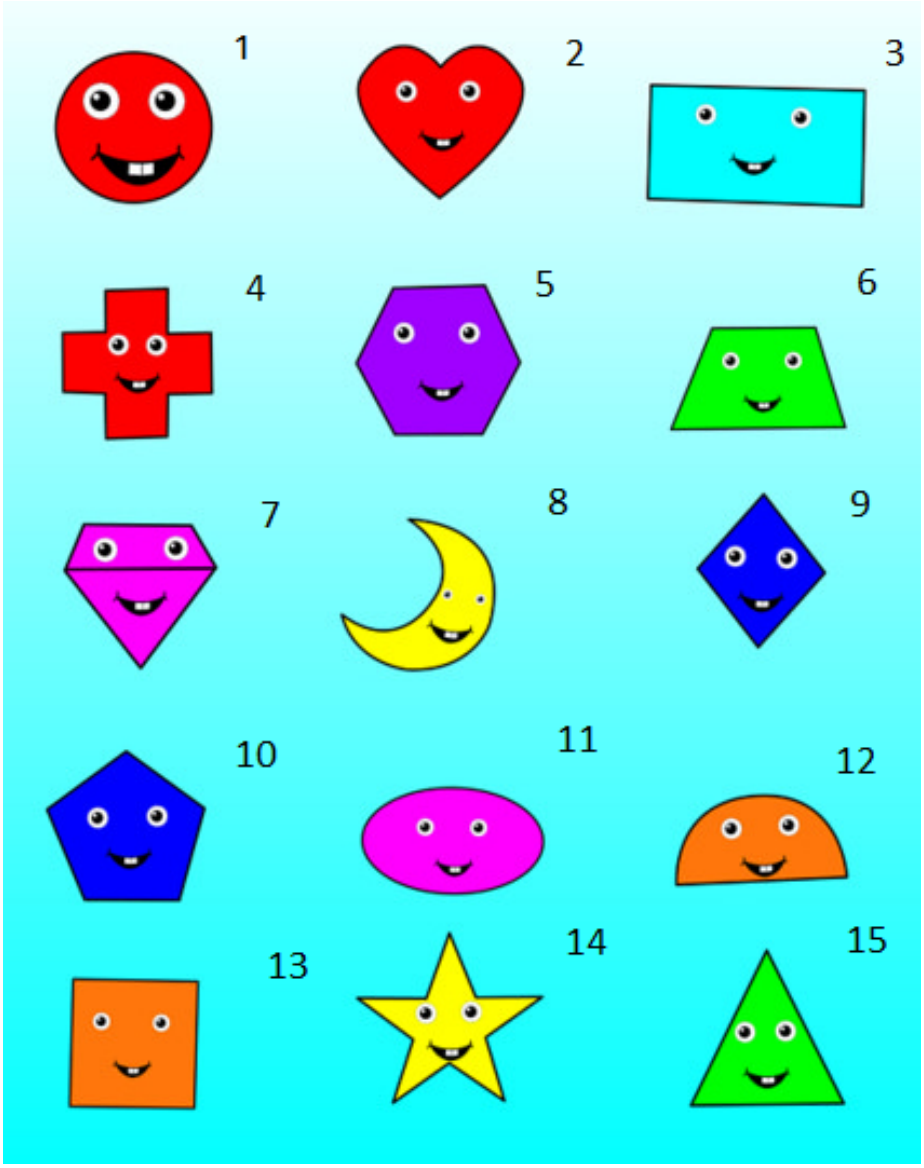

Figure 1: Warianty zadania