Ćwiczenie dodatkowe 3
Działania matematyczne we Flashu

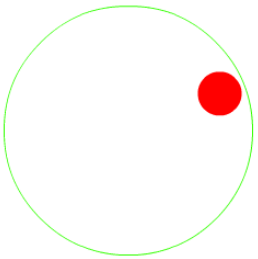
ActionScript pozwala na stosowanie wszelkich działań matematycznych. Do bardziej skomplikowanych operacji wymagany jest import klasy **Math**. Na przykład do wygenerowania liczby losowej możemy użyć funkcji **Math.random()**, która nie ma parametrów:

```
var randomNum:Number = Math.random();
```

W klasie **Math** znajduje się szereg metod, które pozwalają na użycie wielu definicji matematycznych lub wykonanie skomplikowanych operacji. Zawiera np. typowe stałe matematyczne, takie jak **Math.PI** (w przybliżeniu 3,14159265...), czyli stały stosunek obwodu koła do jego średnicy. Zawiera również metody trygonometryczne, takie jak **Math.sin()**, **Math.cos()** i **Math.tan()**, oraz szereg innych. Wykorzystując właściwości i metod klasy **Math** wraz z powoływaniem obiektów należących do klasy **Graphics** (rysowanie kół i prostokątów) uzyskamy bardziej złożone formy.

W wielu metodach klasy **Math** kąty należy podawać w radianach, a nie w stopniach. Przeliczanie tych jednostek należy do typowych zastosowań klasy **Math**:

```
var degrees = 121;
var radians = degrees * Math.PI / 180;
trace(radians) // 2.111848394913139
```



Przykład 1 – Tworzenie i animacja grafiki

Otwórz nowy plik i nie zmieniaj jego właściwości.
Otwórz panel **Operacje** i umieść w nim kod:

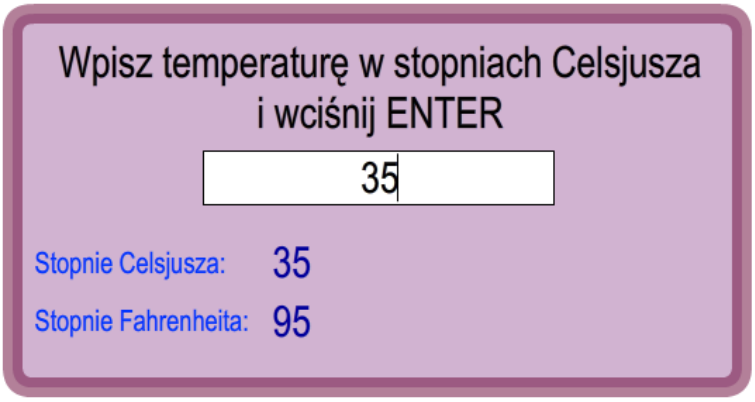
ś czasu | Informacje wyjściowe | Błędy kompilatora | Operacje - Klatka

```
1  stage.frameRate = 31;
2  var currentDegrees:Number = 0;
3  var radius:Number = 90;
4  var satelliteRadius:Number = 20;
5  var satelliteColor:uint = 0xFF0000;
6  var container:Sprite = new Sprite();
7
8  container.x = stage.stageWidth / 2;
9  container.y = stage.stageHeight / 2;
10 addChild(container);
11 var satellite:Shape = new Shape();
12 container.addChild(satellite);
13 addEventListener(Event.ENTER_FRAME, doEveryFrame);
14 function doEveryFrame(event:Event):void
15 {
16 currentDegrees += 4;
17 var radians:Number = getRadians(currentDegrees);
18 var posX:Number = Math.sin(radians) * radius;
19 var posY:Number = Math.cos(radians) * radius;
20 satellite.graphics.clear();
21 satellite.graphics.beginFill(satelliteColor);
22 satellite.graphics.drawCircle(posX, posY, satelliteRadius);
23 }
24 function getRadians(degrees:Number):Number
25 {
26 return degrees * Math.PI / 180;
27 }
```

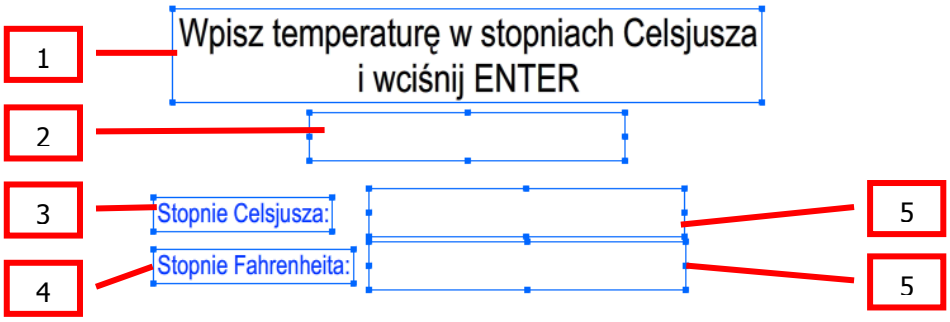
Można sprawdzić, co osiągnęliśmy.

Modyfikacja wartości zmiennych **currentDegrees** (w funkcji **doEveryFrame**) oraz **radius** i **satelliteRadius** na początku kodu umożliwia modyfikację efektów. Spróbujmy na przykład zmniejszyć promień ruchu obiektu (**radius**) i/lub zwiększyć wartość zmiennej **satelliteRadius**. Zmieńmy też kolor satelity (**satelliteColor**). Jest to jedynie przykład zastosowania rysunkowego interfejsu API do uzyskania efektów wizualnych, które są zaskakująco wyrafinowane, jeśli wziąć pod uwagę, jak prosto się je tworzy.

Przykład 2 – Proste obliczenia na danych z pól tekstowych

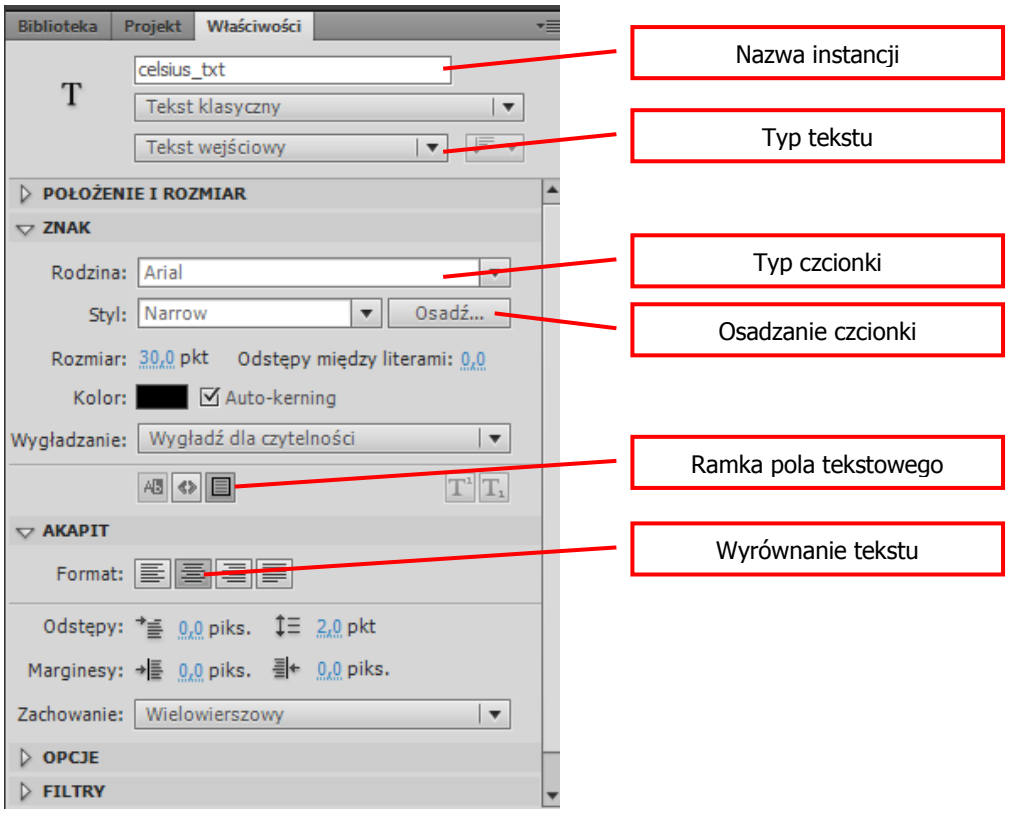


Utwórz nowy plik i nie zmieniaj jego właściwości.
Na stole montażowym umieść 6 pól tekstowych (na jednej warstwie) i ustaw ich właściwości podane w tabeli:



Numer pola	Typ tekstu	Nazwa instancji	Tekst do wpisania
Pole tekstowe 1	Tekst statyczny	*brak*	Wpisz temperaturę w stopniach Celsjusza wciśnij ENTER
Pole tekstowe 2	**Tekst wejściowy**	**celsius_txt**	-
Pole tekstowe 3	Tekst statyczny	*brak*	-
Pole tekstowe 4	Tekst statyczny	*brak*	-
Pole tekstowe 5	**Tekst dynamiczny**	**celsius2_txt**	Stopnie Celsjusza
Pole tekstowe 6	**Tekst dynamiczny**	**fahrenheit_txt**	Stopnie Fahrenheita

Tekst w polu 1 i 2 powinien być wyśrodkowany, a tekst w polu 3-6 wyrównany do lewej strony. Tylko pole tekstowe nr 2 posiada ramkę. Dobierz wielkość i kolor tekstu według własnego uznania. Wszystkie właściwości ustawisz w panelu **Właściwości tekstu**:



Tekst dynamiczny i wprowadzany wymaga osadzenia czcionki – wybierz dowolne z pól tekstowych i wciśnij przycisk **Osadź** w panelu **Właściwości**. Można tu skorzystać z opcji osadź CYFRY.

Aby uzyskać ciekawe tło na osobnej warstwie (pod warstwą z polami tekstowymi) wstaw prostokąt z zaokrąglonymi narożnikami (kąt zaokrąglenia około 15).

Jeśli wszystko jest gotowe to otwieramy panel **Operacji** i wpisujemy następujący kod:

Oś czasu | Informacje wyjściowe | Błędy kompilatora | Operacje - Klatka

```
1  celsius_txt.addEventListener(KeyboardEvent.KEY_DOWN, doConversion);
2
3  function doConversion(myevent:KeyboardEvent):void{
4      if (myevent.keyCode==Keyboard.ENTER){
5          var celsius:Number;
6          var fahrenheit:Number;
7
8          celsius = Number(celsius_txt.text);
9          fahrenheit = (9/5)*celsius+32;
10
11         fahrenheit_txt.text = String(fahrenheit);
12         celsius2_txt.text = String(celsius);
13         celsius_txt.text = "";
14         }
15 }
```

Działanie powstałego formularza jest następujące: użytkownik wpisuje w aktywne pole wartość temperatury w stopniach Celsjusza. Aby uzyskać wartość tej temperatury w stopniach Fahrenheita wystarczy wcisnąć klawisz ENTER. Formularz jest za każdym razem gotowy do kolejnego użycia.

Zwróć uwagę na operatory matematyczne użyte w kodzie: (/) – operator dzielenia, (*) – operator mnożenia i (+) operator dodawania. Jak łatwo się domyśleć (-) będzie operatorem odejmowania.

Niestety wartości z pól tekstowych są przechwytywane jako ciąg znaków, a nie wartość liczbowa. Konieczne jest więc dokonanie ich konwersji do postaci zrozumiałej dla działań matematycznych. Konwersję tę wykonaliśmy za pomocą funkcji konwersji **Number(**„str"**)**.

Przykład 3 – Mały kalkulator

Korzystając z poprzedniego ćwiczenia spróbuj sam zaprojektować formularz i kod ActionScript służący do sumowania dwóch liczb. Tym razem wynik pojawi się po naciśnięciu przycisku.

Dla „niecierpliwych" lub „nie gotowych" opis wykonania tego ćwiczenia poniżej:

W nowym pliku wstawiamy 3 pola tekstowe – dwa typu **tekst wprowadzany** i jedno typu **tekst dynamiczny**. Pamiętaj o osadzeniu czcionki.
Polom tym nadajemy nazwy instancji np. odpowiednio „*l1_txt*", „*l2_txt*" i „*wynik_txt*".
Projektujemy przycisk (możemy skorzystać z przycisku dostępnego we **Wspólnej bibliotece „Przyciski"** , do której mamy dostęp przez menu **Okno**). Przyciskowi umieszczonemu na stole montażowym nadajemy nazwę instancji „*oblicz_btn*".
Na koniec w panelu **Operacje** wstawiamy kod:

Oś czasu | Informacje wyjściowe | Błędy kompilatora | Operacje - Klatka

```
1  import flash.events.MouseEvent;
2
3  function obliczSuma(event:MouseEvent):void{
4      var suma:Number;
5      suma=Number(l1_txt.text)+Number(l2_txt.text);
6      wynik_txt.text=String(suma);
7  }
8
9  oblicz_btn.addEventListener(MouseEvent.CLICK, obliczSuma);
```

Uwaga. Jeśli w formularzu chcesz działać na liczbach niecałkowitych jako separatora użyj kropki.

Przykład 4 – Duży kalkulator

Ten kalkulator potrafi dodawać kolejne liczby, odejmować, dzielić i mnożyć. Niestety, do jego wykonania w podstawowej wersji i udoskonalania jego funkcjonalności potrzebny jest duży nakład pracy.

Zacznij od zaprojektowania grafiki. Jak widać potrzebne będą przyciski, ramka jako tło do pola tekstowego służącego do wyświetlania liczb oraz prostokąt wyznaczający pole kalkulatora.

Projektowanie przycisków i grafiki nie powinno Wam już zapewne sprawić trudności.

Pole tekstowe stanowi pole dynamiczne (nie zapomnij osadzić czcionki). Potrzebne będą nazwy instancji przycisków i pola tekstowego:
Pole tekstowe: „*wynik_txt*”
Przyciski 1-9: odpowiednio „*but1*”, „*but2*”.....
Przycisk 0: „*but10*”
Przycisk „,” (przecinek): „*but11*”
Przycisk „+” (dodawanie): „*plus_btn*”
Przycisk „-” (odejmowanie): „*minus_btn*”
Przycisk „:” (dzielenie): „*dziel_btn*”
Przycisk „x” (mnożenie): „*razy_btn*”
Przycisk „=”: „*wynik_btn*”
Przucisk „CE”: „*kasuj_btn*”

A teraz kod. Jest on dość skomplikowany i długi. Zaczynamy od deklaracji zmiennych i dodania obiektu nasłuchującego do przycisków:

mpilatora | Operacje - Klatka

Wycinki kodu

```
1  import flash.events.MouseEvent;
2
3  var holdNumber1:Boolean = false;
4  var holdNumber2:Boolean = true;
5  var Number1:Number = 0;
6  var Number2:Number = 0;
7  var wynik:Number = 0;
8  var wynik2:Number = 0;
9  var number:Number = 0;
10 var typeOperator:Number = 0;
11
12 but1.addEventListener(MouseEvent.CLICK, getNumber);
13 but2.addEventListener(MouseEvent.CLICK, getNumber);
14 but3.addEventListener(MouseEvent.CLICK, getNumber);
15 but4.addEventListener(MouseEvent.CLICK, getNumber);
16 but5.addEventListener(MouseEvent.CLICK, getNumber);
17 but6.addEventListener(MouseEvent.CLICK, getNumber);
18 but7.addEventListener(MouseEvent.CLICK, getNumber);
19 but8.addEventListener(MouseEvent.CLICK, getNumber);
20 but9.addEventListener(MouseEvent.CLICK, getNumber);
21 but10.addEventListener(MouseEvent.CLICK, getNumber);
22 but11.addEventListener(MouseEvent.CLICK, getNumber);
23
24 plus_btn.addEventListener(MouseEvent.CLICK, getNumber);
25 minus_btn.addEventListener(MouseEvent.CLICK, getNumber);
26 razy_btn.addEventListener(MouseEvent.CLICK, getNumber);
27 dziel_btn.addEventListener(MouseEvent.CLICK, getNumber);
28 wynik_btn.addEventListener(MouseEvent.CLICK, getNumber);
29 kasuj_btn.addEventListener(MouseEvent.CLICK, kasujWynik);
```

Teraz funkcja **getNumber**:

```
30
31 function getNumber(event:MouseEvent):void{
32     if (event.target == but1){
33         wynik_txt.appendText("1");
34     }
35     if (event.target == but2){
36         wynik_txt.appendText("2");
37     }
38     if (event.target == but3){
39         wynik_txt.appendText("3");
40     }
41     if (event.target == but4){
42         wynik_txt.appendText("4");
```

```
43      }
44      if (event.target == but5){
45          wynik_txt.appendText("5");
46      }
47      if (event.target == but6){
48          wynik_txt.appendText("6");
49      }
50      if (event.target == but7){
51          wynik_txt.appendText("7");
52      }
53      if (event.target == but8){
54          wynik_txt.appendText("8");
55      }
56      if (event.target == but9){
57          wynik_txt.appendText("9");
58      }
59      if (event.target == but10){
60          wynik_txt.appendText("0");
61      }
62      if (event.target == but11){
63          wynik_txt.appendText(".");
64      }
65      ////////////// zachowujemy liczby
66      if(holdNumber1==false){
67         Number1 = Number(wynik_txt.text);
68         }
69      if(holdNumber2==false){
70         Number2 = Number(wynik_txt.text);
71      }
72
73      if(event.target==plus_btn){
74          holdNumber1=true;
75          holdNumber2=false;
76          wynik_txt.text="";
77          typeOperator=1;
78      }
79      if(event.target==minus_btn){
80          holdNumber1=true;
81          holdNumber2=false;
82          wynik_txt.text="";
83          typeOperator=2;
84      }
85      if(event.target==razy_btn){
86          holdNumber1=true;
87          holdNumber2=false;
88          wynik_txt.text="";
89          typeOperator=3;
90      }
91      if(event.target==dziel_btn){
92          holdNumber1=true;
93          holdNumber2=false;
94          wynik_txt.text="";
95          typeOperator=4;
96      }
```

W tej samej funkcji piszemy dalej:

```
97      if(event.target==wynik_btn){
98          if (typeOperator==1){
99              if(wynik==0){
100                 wynik = Number1 + Number2;
101                 wynik_txt.text=String(wynik);
102                 Number1=0;
103                 Number2=0;
104                 number=Number(wynik_txt.text);}
105             else{
106                 wynik = number + Number2;
107                 wynik_txt.text=String(wynik);
108                 number=Number(wynik_txt.text);
109                 }
110         }
111         if (typeOperator==2){
112             if(wynik==0){
113                 wynik = Number1 - Number2;
114                 wynik_txt.text=String(wynik);
115                 Number1=0;
116                 Number2=0;
117                 number=Number(wynik_txt.text);
118                 }
119             else{
120                 wynik = wynik - Number1 - Number2;
121                 wynik_txt.text=String(wynik);
122                 Number1=0;
123                 Number2=0;
124                 number=Number(wynik_txt.text);
125                 }
126         }
127         if (typeOperator==3){
128             if(wynik==0){
129                 wynik2 = Number1 * Number2;
130                 wynik_txt.text=String(wynik2);
131                 holdNumber1 = false;
132                 number=Number(wynik_txt.text);
133                 }
134             else{
135                 wynik = number * Number2;
136                 wynik_txt.text=String(wynik);
137                 number=Number(wynik_txt.text);
138                 }
139         }
140         if (typeOperator==4){
141             if(wynik==0){
142                 wynik2 = Number1 / Number2;
143                 wynik_txt.text=String(wynik2);
144                 holdNumber1 = false;
145                 number=Number(wynik_txt.text);
146                 }
147             else{
148                 wynik = number / Number2;
149                 wynik_txt.text=String(wynik);
150                 holdNumber1 = false;
151                 number=Number(wynik_txt.text);
152                 }
153         }
154     }
155 }
```

Teraz napiszemy funkcję **kasujWynik**:

```
156
157 function kasujWynik(event:MouseEvent):void{
158     wynik_txt.text = "";
159     Number1 = 0;
160     Number2 = 0;
161     wynik = 0;
162     holdNumber1 = false;
163     holdNumber2 = true;
164 }
```

I to na razie koniec. Być może wystąpią jakieś nieoczekiwane przypadki matematyczne np. dzielenie przez O. Należałoby zoptymalizować kod pod tym kątem, jak również uczynić go bardziej „przyjaznym”. Może ktoś wymyśli inne, ciekawsze rozwiązanie☺?