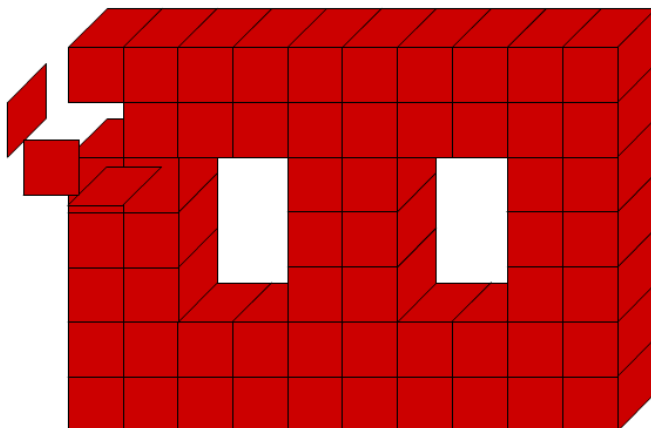


ZADANIE 1

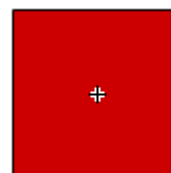
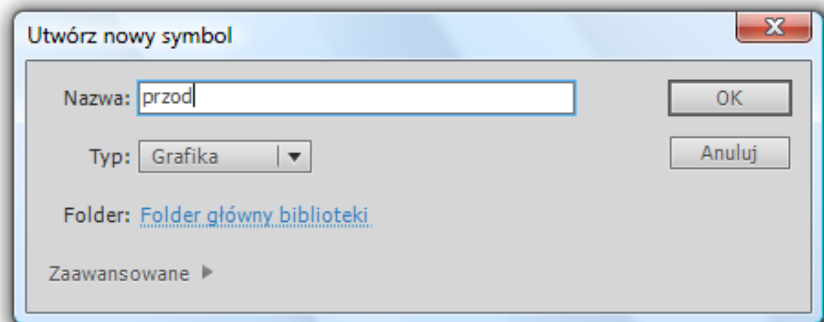
W zadaniu wykorzystane będą wszystkie typy symboli występujące we Flashu, zastosowane będzie podstawowa animacja ruchu i użyte zostaną bazowe komendy języka ActionScript 3.0 służące do sterowania listwą czasową animacji.

Animowana kostka



Elementy niezbędne do pracy:

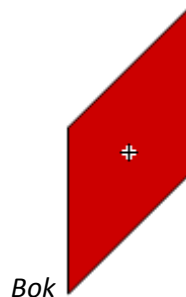
- **Trzy symbole graficzne** reprezentujące ściany kostki (nazwane odpowiednio „przod”, „bok” i „gora”). Tylko za pierwszym razem tworzymy nowy element od początku - używając polecenia Wstaw > Nowy symbol (pamiętajmy aby wybrać Typ: Grafika i nazwać symbol!). W środku rysujemy kwadrat np. o boku 50x50.



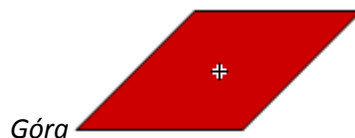
Kolejne element POWIELAMY w BIBLIOTECIE z istniejącego już symbolu. POWIELONE KOPIE nazywamy „przod” i „bok”. Elementy musimy edytować, aby dokonać zmian polegających na pochyleniu kwadratu w celu uzyskania kolejnych boków sześciianu:

Dla elementu „bok” stosujemy opcję pochylenia „w pionie” o kąt -45° (paleta narzędziowa **Przekształć**).

Dla elementu „gora” stosujemy opcję pochylenia „w poziomie” o kąt 45° (paleta narzędziowa **Przekształć**).

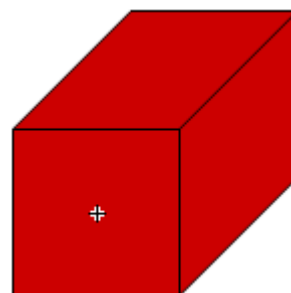


Bok

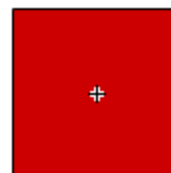


Góra

- **Jeden przycisk** o nazwie roboczej „kostka” (Wstaw > Nowy symbol). Wewnątrz tego symbolu zestawiamy kostkę pobierając przygotowane elementy z biblioteki. Zwróć uwagę na charakterystykę listwy czasowej przycisku (4 klatki reprezentujące stany przycisku). Kostkę zestawiamy w klatce 1, która nazywa się „W g...”. Jest to stan bazowy przycisku (gdy nie jest w użyciu).



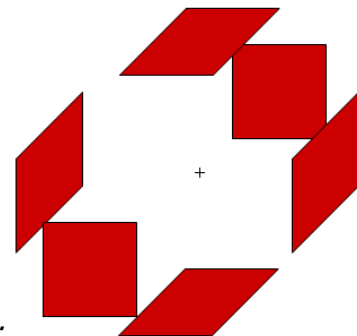
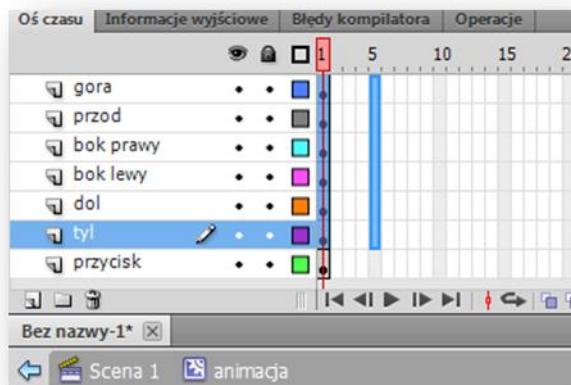
Kopiujemy zawartość tej klatki do klatek Over, Down i Hit - wystarczy w omawianych klatkach użyć klawisza F6. Jakkolwiek w klatce Hit należy usunąć górną i boczną ściankę kostki, a pozostawić tylko ściankę przednią.



- **Jeden klip o roboczej nazwie „animacja”** (Wstaw > Nowy symbol). Wewnątrz klipu wstawiamy 6 nowych warstw. Dolną warstwę nazywamy PRZYCISK, a pozostałe nazywamy tak, jak nazywa się odpowiednio każda ze ścianek kostki (także te niewidoczne!): TYŁ, DÓŁ, BOK LEWY, BOK PRAWY, GÓRA, PRZÓD. W klatce nr 1 warstwy PRZYCISK wstawiamy przycisk z kostką i nadajemy mu nazwę instancji „kostka_btn”. W kolejnych warstwach umieszczamy konsekwentnie element po elemencie wszystkie ścianki kostki (używamy gotowych ścianek z Biblioteki filmu!) dokładnie przykrywając zamieszczony w najniższej warstwie przycisk, tak jakbyśmy chcieli ponownie w tym samym miejscu odtworzyć kostkę.

Pozostałe czynności w klipie:

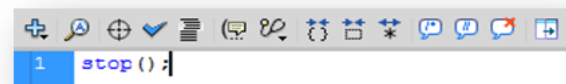
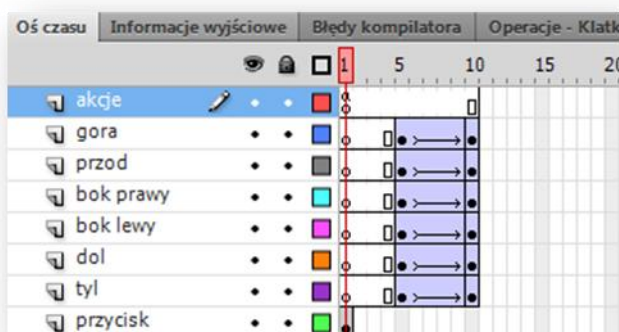
Jednym ruchem zaznaczamy wszystkie klatki ze ściankami i odsuwamy je do klatki nr 5:



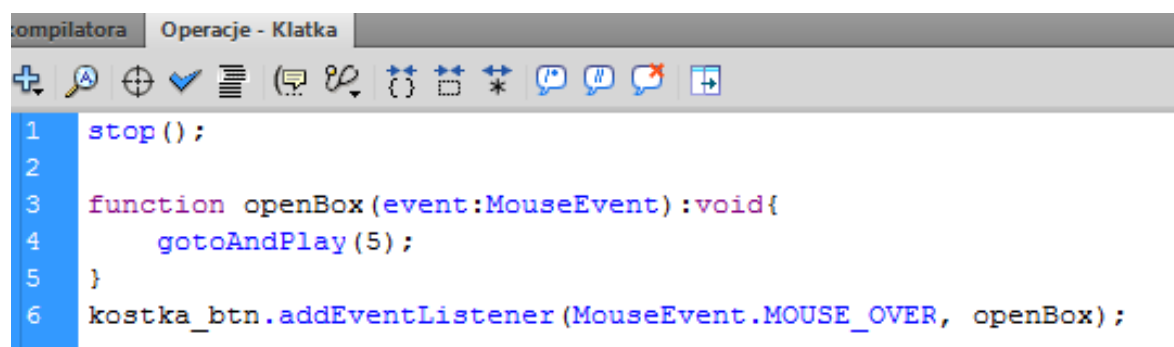
W klatce 10:

Tam tworzymy efekt otwieranego pudełka: na zaznaczonej klatce 5 (najlepiej wszystkich jednocześnie) otwieramy menu podręczne i wybieramy polecenie **Utwórz klasyczną animację**. Zaznaczamy teraz klatki nr 10 (też najlepiej wszystkie jednocześnie) i wstawiamy klatki kluczowe (F6). W klatce 10 każdy element kostki na scenie należy przesunąć tak, aby kostka się rozpadała.

Dodajemy jeszcze jedną warstwę i nazywamy ją „akcje”. Otwieramy panel **Operacje** i wstawiamy na początek akcję `stop()` ;



A poniżej akcji `stop()`; piszemy funkcję, która zapewni przejście do klatki 5 w bieżącym klipie.



Całość będzie wyglądała tak, że po najechaniu myszką na przednią ściankę kostki (przycisku) nastąpi rozjechanie się jej ścianek na boki, co zapewniła nam automatyczna animacja wykonana w klatkach 5-10

klipu. Ponieważ nie zastosowano akcji stop() na końcu animacji efekt pozostaje odwracalny – po zjechaniu z kostki ścianki powrócą na miejsce.

- Pozostało jedynie wstawić nasz klip z animowaną kostką do głównej sceny. Możemy utworzyć też efektowny murek z wielu takich klipów ustawionych obok siebie.

ZADANIE 2

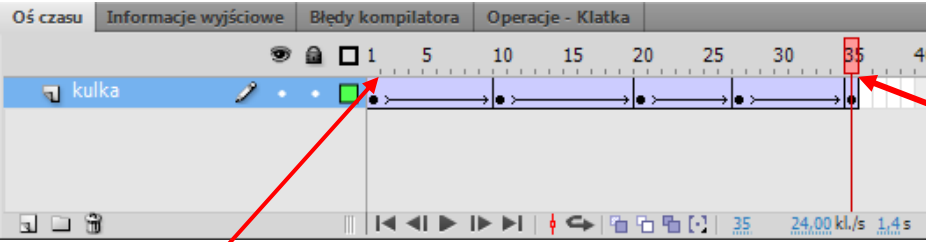
W zadaniu wykorzystane będą właściwości obiektów, metody i instrukcje warunkowe języka ActionScript 3.0, a także obsługa zdarzeń.

Animacja kulki

Utwórz animację polegającą na poruszaniu się obiektu – kulki - wokół sceny filmu. Wykonaj to dwoma wymienionymi niżej sposobami lub od razu przejdź do sposobu drugiego.

Sposób 1 – poruszanie obiektu techniką klatek kluczowych i klasyczną animacją ruchu

Przykładowa listwa czasowa wykonanego zadania:



Każda klatka z kropką to klatka kluczowa służąca do wprowadzania zmian obiektu (położenia, wyglądu itp.) a tym samym dokonania ustawienia parametrów na początku, w trakcie lub na końcu animacji

Pierwszej wstawionej klatce z zawartością (kulka) nadajemy **klasyczną animację ruchu** następnie wstawiamy na warstwie kolejne klatki kluczowe (F6) i w nich przestawiamy obiekt (lub zmieniamy jego wygląd)

Sposób 2 – sterowaniem ruchem obiektu poprzez język ActionScript 3.0.

Poruszany obiekt musi mieć nadaną nazwę instancji np. **ball_mc** (nazwy instancji nadajemy w oknie **Właściwości** obiektu).

Każdy obiekt posiada szereg właściwości, którymi można sterować z poziomu AS. Dla obiektów wyświetlanych podstawowymi właściwościami są położenie (właściwość **x** i właściwość **y**), wielkość obiektu (właściwość **width** i właściwość **height**), widoczność (**alpha**) itp. Przykładowy zapis odwołań do właściwości obiektu:

```
ball_mc.alpha = 0.5 //obiekt półprzezroczysty (zakres przezroczystości wynosi od 0 do 1)
```

```
ball_mc.width = 100 //szerokość obiektu wynosi 100
```

```
ball_mc.x ++ //obiekt będzie poruszał się ruchem jednostajnym w prawo.
```

Niezbędne do wykonania tego zadania będą instrukcje warunkowe języka ActionScript.

Korzystając z instrukcji warunkowej **if..else if**, można sprawdzać więcej niż jeden warunek. Jeśli nie użyjemy nawiasów przy większej niż jedna liniach instrukcji błąd ten może doprowadzić do nieoczekiwanego zachowania programu.

Instrukcja **switch** jest użyteczna, jeśli mamy kilka możliwych ścieżek wykonania, a wybór jednej z nich zależy od wartości **tego samego wyrażenia warunkowego**

```
if(warunek) {
    instrukcje;}
else if(warunek) {
    instrukcje;}
else{...}
```

```
switch(wyrażenie){
    case 0:
        instrukcje;
        break;
    case 1:
        instrukcje;
        break;
    default:
        instrukcje;
        break;
}
```

ActionScript pozwala pisać funkcje parametryczne, których parametrem może być określone zdarzenie (przykładowy nagłówek funkcji: `function ballMove(event:Event):void{...}`). Taką funkcję należy skojarzyć zarówno z danym typem zdarzenia jak i obiektem, będącym odbiornikiem zdarzenia. Za tę czynność odpowiada osobna linia instrukcji. Odbiornikiem zdarzenia może być zarówno każdy obiekt wyświetlany jak i kontener nadrzędny czyli scena. Przykładowo:

```
stage.addEventListener(Event.ENTER_FRAME,ballMove);
```

Zdarzenia dotyczą wielu aspektów związanych z czynnościami dokonywanymi przez użytkownika (np. wciskanie określonego klawisza na klawiaturze, ruszanie czy klikanie myszą) jak i operacje wynikające z dynamicznych akcji w obrębie naszego filmu, odbywanych się w sposób naturalny i cykliczny (zdarzenie `ENTER_FRAME` to nic innego jak przejście z klatki do klatki w trakcie odtwarzania filmu) lub okazjonalnie (np. zdarzenie `PROGRESS` lub `COMPLETE` notowane w trakcie lub tuż po załadowaniu zasobów oraz danych do aplikacji, np. pliku swf, pliku z obrazem – PNG, JPG – lub z dźwiękiem – mp3).

Funkcja, która obsługuje ruch kulki może zaczynać się następująco:

```
function ballMove(event:Event):void{
    switch (flaga){
        case 0:
            ball_mc.x -= speed;
            if(ball_mc.x<=0) {
                ball_mc.x = 0;
                flaga = 1;}
            break;
        case 1:
```

```
        ...;  
        break;  
        ...; }  
}
```

Powyżej funkcji można zadeklarować zmienne pozwalające ustawić prędkość poruszania się kulki oraz zmienną pomocniczą do badania stanu kulki:

```
var speed:int = 25;  
var flaga:uint = 0;
```

To wszystko w tym ćwiczeniu. W zadaniu 3 dodamy ruchomą przeszkodę, która spowoduje:

- a) zniknięcie kulki w momencie zetknięcia obu obiektów
- b) lub zmianę toru ruchu kulki (zadanie 4).

ZADANIE 3

W zadaniu wykorzystane będą metody `startDrag()` i `stopDrag()` – służące do obsługi „ręcznego” przeciągania obiektu po scenie - oraz metody `removeChild` (zdejmowanie obiektu z listy wyświetlania) i `hitTestObject` (wykrywanie kolizji).

Zniknięcie kulki po zetknięciu z przeszkodą

Na początek wstawiamy obiekt, który będzie pełnił rolę przeszkody dla toru ruchu kulki. Może to być pionowa ścianka. Obiekt konwertujemy do klipu filmowego i nadajemy mu nazwę instancji np. **scianaL_mc**.



Następnie piszemy dwie funkcje, które będą odpowiedzialne za obsługę przeciągania myszką obiektu po scenie:

```
function obstacleDrag(event:MouseEvent):void{  
    event.target.startDrag();  
}  
function obstacleStopDrag(event:MouseEvent):void{  
    event.target.stopDrag();  
}
```

i dodajemy elementy nasłuchujące konkretnych zdarzeń myszy, których odbiornikiem jest nasza przeszkoda:

```
scianaL_mc.addEventListener(MouseEvent.CLICK, obstacleDrag);  
scianaL_mc.addEventListener(MouseEvent.CLICK, obstacleStopDrag);
```

Jeśli chcemy aby kulka zniknęła po zderzeniu ze ścianką musimy wykorzystać metodę `removeChild`, która zdejmuje obiekt z listy wyświetlania.

```
removeChild(ball_mc);
```

Do wykrycia kolizji posługujemy się metodą `hitTestObject`. Konstrukcja może wyglądać następująco:

```
ball_mc.hitTestObject(scianaL_mc); //po wykryciu kolizji zwraca wartość true
```

Można też skonstruować funkcję i wykorzystać ją w odpowiednim miejscu:

```
function hitObject(object_,obstacle_):Boolean{
    return(object_.hitTestObject(obstacle_));
}

function ballRemove(event:Event):void{
    if(hitObject(ball_mc,scianaL_mc)==true){
        stage.removeEventListener(Event.ENTER_FRAME,ballMove);
        stage.removeEventListener(Event.ENTER_FRAME,ballRemove);
        removeChild(ball_mc);
    }
}
```

ZADANIE 4

Zmiana toru ruchu kulki po zetknięciu z przeszkodą – zadanie do samodzielnego wykonania

Wykonaj modyfikację kodu, tak aby po zetknięciu z ruchomą przeszkodą kulka zmieniła kierunek ruchu na przeciwny.

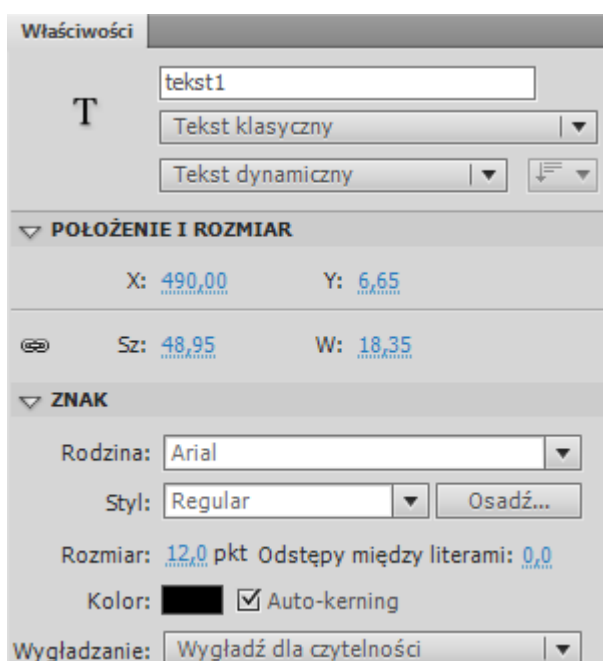
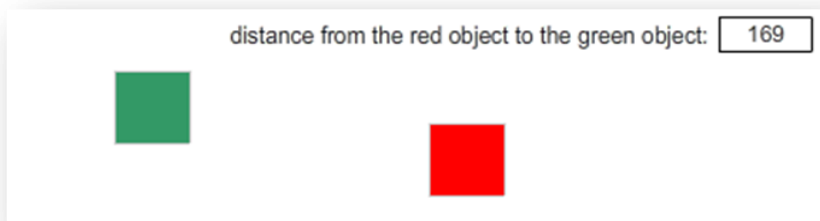
ZADANIE 5

W zadaniu wykorzystane będzie pole tekstu dynamicznego, ponownie opcje przenoszenia klipu, a także obiekt `Point` z metodą `distance` służącą do obliczania odległości między wskazanymi obiektami (punktami)

Odległość między dwoma ruchomymi obiektami

Elementy niezbędne do ćwiczenia:

- Dwa kwadraty (klipy) różnego koloru (np. zielonego i czerwonego). Kwadraty można rysować bezpośrednio na scenie głównej, a potem dopiero konwertować do klipu (skrót F8). Po konwersji klipom umieszczonym na scenie należy nadać tzw. **nazwę instancji** - odpowiednio „kw1” i „kw2”.
- Pole tekstu dynamicznego uzyskane z narzędzia Tekst z ustawionymi właściwościami:



Ponieważ tekst jest typu dynamicznego musimy osadzić czcionkę w projekcie (kliknij przycisk Osadź).

- Można też dodać tekst statyczny, który wyświetla stały komunikat w stylu: „distance from the red object to the green object:”.
- I najważniejsze – wpisz kod ActionScript w panelu **Operacje**:

```
import flash.geom.*;
import flash.events.MouseEvent;
import flash.text.TextField;

function drag(event:MouseEvent):void{
    event.target.startDrag();}

function dragStop(event:MouseEvent): void{
    event.target.stopDrag();}

function distanceCalculation(event:MouseEvent):void{
    var pt1:Point = new Point(kw1.x, kw1.y);
```

```
var pt2:Point = new Point(kw2.x, kw2.y);
var distance:Number = Point.distance(pt1, pt2);
tekst1.text = String(Math.floor(distance));
}
kw1.addEventListener(MouseEvent.CLICK, drag);
kw2.addEventListener(MouseEvent.CLICK, drag);
kw1.addEventListener(MouseEvent.CLICK, dragStop);
kw2.addEventListener(MouseEvent.CLICK, dragStop);
stage.addEventListener(MouseEvent.CLICK, distanceCalculation);
```

Dodatkowe objaśnienia kodu:

Obiekt **Point** reprezentuje miejsce w dwuwymiarowym układzie współrzędnych, gdzie **x** oznacza współrzędną poziomą, a **y** oznacza współrzędną pionową. Poniższy kod tworzy punkt we współrzędnych (0,0):

```
var myPoint:Point = new Point();
```

My utworzyliśmy dwa tego typu obiekty (punkty) o współrzędnych skopiowanych z naszych klipów. Było to działanie niezbędne do obliczenia odległości między tymi punktami. Interesujący nas wynik uzyskujemy stosując metodę **distance** obiektu **Point**, która wraca odległość między punktami **pt1** i **pt2**.

```
Point.distance(pt1:Point, pt2:Point):Number
```

ZADANIE 6

W zadaniu wykorzystane będzie rzutowanie współrzędnych lokalnych klipu na współrzędne globalne (metoda `localToGlobal`)

Odległość między ruchomym obiektem zagnieżdżonym na liście czasowej klipu a wybranym punktem sceny

Translacja przestrzeni współrzędnych

Jeśli dwa obiekty wyświetlane znajdują się w różnych kontenerach (innych obiektach wyświetlanych), mogą przez to znajdować się w różnych przestrzeniach współrzędnych. Metoda `localToGlobal()` klasy `DisplayObject` umożliwia translację współrzędnych do tej samej (globalnej) przestrzeni współrzędnych, którą jest przestrzeń współrzędnych obiektu `Stage`. Na przykład poniższy kod wyznacza odległość między punktami rejestracji dwóch obiektów wyświetlanych, `circle1` i `circle2`, które są zawarte w różnych obiektach wyświetlanych (kontenerach):

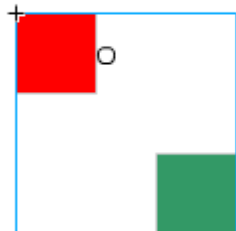
```
import flash.geom.*;
var pt1:Point = new Point(circle1.x, circle1.y);
pt1 = circle1.localToGlobal(pt1);
var pt2:Point = new Point(circle2.x, circle2.y);
pt2 = circle2.localToGlobal(pt2);
var distance:Number = Point.distance(pt1, pt2);
```

Podobnie, aby wyznaczyć odległość między punktem rejestracji obiektu wyświetlanego o nazwie `target` a konkretnym punktem na stole montażowym, można skorzystać z metody `localToGlobal()` klasy `DisplayObject`:

```
import flash.geom.*;
var stageCenter:Point = new Point();
stageCenter.x = this.stage.stageWidth / 2;
stageCenter.y = this.stage.stageHeight / 2;
var targetCenter:Point = new Point(target.x, target.y);
targetCenter = target.localToGlobal(targetCenter);
var distance:Number = Point.distance(stageCenter, targetCenter);
```

Utwórz klip zawierający dwa inne klipy. Dodaj dwa pola tekstu dynamicznego według schematu poniżej. Nadaj nazwy instancji wszystkim klipom i polom tekstowym.

PRZYKŁAD 2



distance from green object to the corner:
distance from red object to the corner:

Wykorzystaj poniższą metodę do obliczenia odległości kolorowych klipów od górnego lewego narożnika sceny.

```
var globalPoint:Point = myClip.localToGlobal(myPoint);
```