helion.pl
KSIĘGARNIA INTERNETOWA · WYDAWNICTWO

IDŹ DO
PRZYKŁADOWY ROZDZIAŁ
SPIS TREŚCI

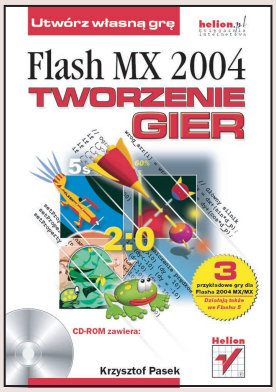
KATALOG KSIĄŻEK
KATALOG ONLINE
ZAMÓW DRUKOWANY KATALOG

TWÓJ KOSZYK
DODAJ DO KOSZYKA

CENNIK I INFORMACJE
ZAMÓW INFORMACJE O NOWOŚCIACH
ZAMÓW CENNIK

CZYTELNIA
FRAGMENTY KSIĄŻEK ONLINE

Flash MX 2004. Tworzenie gier



Autor: Krzysztof Pasek
ISBN: 83-7361-451-6
Format: B5, stron: 124

Wszyscy wiedzą, że Flash jest doskonałym narzędziem do tworzenia serwisów WWW i prezentacji multimedialnych. Jednak na tym jego możliwości się nie kończą. Chcesz się przekonać, co jeszcze można zrobić za pomocą Flasha? Połącz pracę z rozrywką i stwórz własną grę! Opracuj scenariusz, zaprojektuj plansze oraz obiekty i napisz kod. Nie musisz wcale opanowywać skomplikowanych środowisk programistycznych i rozbudowanych aplikacji graficznych. Jedyne, czego potrzebujesz, to Flash MX 2004 i pomysł. Wkrótce przekonasz się, że stworzenie gry we Flashu leży w zasięgu Twoich możliwości.

W książce „Flash MX 2004. Tworzenie gier" znajdziesz informacje niezbędne do stworzenia prostej gry. Nauczysz się korzystać z narzędzi rysunkowych i animacyjnych, poznasz język ActionScript, dowiesz się, jak przypisywać dźwięki do klatek i zdarzeń, a także wraz z autorem stworzysz kilka gier. Przeczytasz o:

- Narzędziach z panelu Tools
- Klatkach, warstwach, scenach i symbolach
- Animacji i akcjach
- Tworzeniu przycisków
- Obsłudze myszy i klawiatury
- Tworzeniu kodu w języku ActionScript
- Korzystaniu z dźwięku
- Głównych elementach każdej gry komputerowej

Wydawnictwo Helion
ul. Chopina 6
44-100 Gliwice
tel. (32)230-98-63
e-mail: helion@helion.pl

Spis treści

Rozdział 1. Rysowanie 7
1.1. Grafika we Flashu 7
Panel Tools 7
Kilka uwag 9
Gradienty 10
1.2. Operowanie scenami, warstwami i ujęciami 11
Ujęcia 11
Warstwy 13
Tworzenie maski 14
Sceny 15
1.3. Symbole 15
Tworzenie symboli 15
Edycja symbolu 16
Symbol i jego klony 18
Obiekty typu Graphic 19
Grupy 19

Rozdział 2. Animacja 21
2.1. Tweening 21
2.2. Kontrolowanie filmu za pomocą akcji 23
Okno Actions 23
Akcje sterujące filmem 24
Przykład użycia akcji 24

Rozdział 3. Odbieranie danych od użytkownika 29
3.1. Przyciski 29
Listwa czasowa przycisku 29
Zdarzenia 30
Przykład użycia przycisków 31
3.2. Pola tekstowe 35
3.3. Mysz i klawiatura 37
Właściwości _xmouse i _ymouse 37
Obiekt Mouse 37
Obiekt Key 38

Rozdział 4. Movie Clip ... 39
4.1. Ścieżki dostępu ... 39
Identyfikator ... 39
Ścieżki dostępu ... 40
4.2. Właściwości ... 42
Dostępne właściwości ... 42
Odwołania do właściwości ... 43
4.3. Zdarzenia ... 44
4.4. Operowanie obiektami w ActionScript ... 45
Przykład użycia właściwości ... 45
Dynamiczne kopiowanie obiektów ... 47

Rozdział 5. Budowa kodu ActionScript ... 49
5.1. Zmienne i podstawowe operatory ... 49
Zmienne ... 49
Nazwy zmiennych ... 50
Typy danych ... 50
Operator „=” ... 51
Operatory arytmetyczne ... 51
5.2. Instrukcja warunkowa ... 52
Przykład użycia akcji if ... 55
5.3. Tablice ... 58
5.4. Pętle ... 59
Pętla for ... 59
Pętla while ... 60
Pętla do while ... 60
5.5. Funkcje ... 60

Rozdział 6. Dźwięki ... 63
6.1. Przypisanie dźwięku do ujęcia ... 63
6.2. Obiekt Sound ... 64

Rozdział 7. Samolot ... 67
7.1. Założenia gry ... 67
7.2. Ogólne ustawienia sceny ... 68
7.3. Stworzenie tras ... 69
7.4. Samolot ... 71
7.5. Zatrzymanie gry ... 72
7.6. Ruch ... 72
7.7. Udoskonalenia ... 78

Rozdział 8. Skaczące żaby ... 79
8.1. Założenia ... 79
8.2. Ogólne ustawienia sceny ... 80
8.3. Tworzenie planszy ... 80

8.4. Punktacja ... 81
8.5. Postacie ... 83
8.6. Ruch ... 85
8.7. Interakcje żab ... 88
8.8. Efekt cząsteczkowy — krew ... 90

Rozdział 9. Bitwa w kosmosie ... 93
9.1. Założenia ... 93
9.2. Ogólne ustawienia sceny ... 94
9.3. Tworzenie menu ... 95
9.4. Gra ... 97
9.5. Obiekty ... 102
9.6. Amunicja ... 110

Skorowidz ... 119

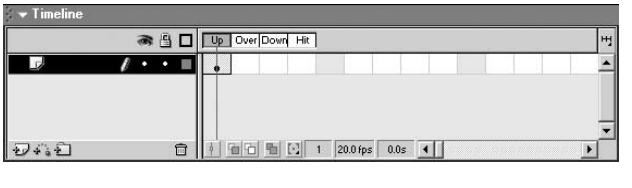
Rozdział 3.
Odbieranie danych od użytkownika

3.1. Przyciski

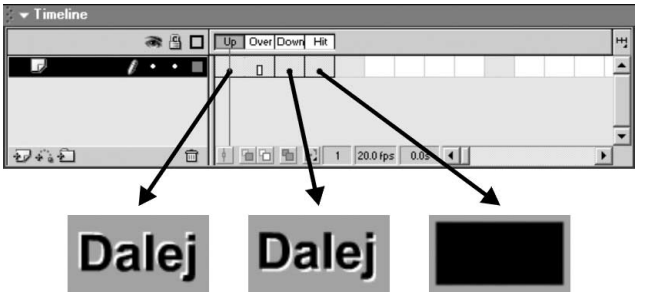
Obiekty typu *Button* wyraźnie różnią się od obiektów typu *Graphic*. Najważniejsze jest to, że każdy przycisk automatycznie reaguje na mysz, może posiadać identyfikator (specjalne „imię", do którego możemy odwoływać się w *ActionScript*) i przypisane akcje. Jeśli chcesz przypisać akcje do obiektu, zrób to tak samo, jak w przypadku ujęcia. Kiedy zaznaczony jest przycisk, tytuł okna *Actions* zmienia się na *Actions - Button*.

Listwa czasowa przycisku

Automatyczne reakcje przycisków na mysz polegają na przechodzeniu do jednego z czterech pierwszych ujęć obiektu. Ujęcia te są podpisane *Up*, *Over*, *Down* i *Hit* (rysunek 3.1). Do ujęcia *Up* przycisk przenosi się, kiedy kursor znajduje się poza obszarem przycisku. Ujęcie *Over* jest wyświetlane, kiedy kursor znajdzie się w obrębie przycisku, a ujęcie *Down*, kiedy na obszarze przycisku zostanie naciśnięty lewy przycisk myszy. Obszar przycisku nie jest określony na podstawie aktualnego ujęcia na jego listwie, zawsze zależy od zawartości ujęcia *Hit*. Możesz umieszczać w nim dowolne wypełnienia, obiekty i tekst. Nawet grubsze linie umożliwią kliknięcie. Pamiętaj jednak, że w większości przypadków użycie tekstu nie daje dobrych rezultatów, ponieważ użytkownik jest zmuszony wycelować kursorem w linię litery. Tekst lepiej zastąpić prostokątem (rysunek 3.2).



Rysunek 3.1.
Listwa czasowa przycisku



Rysunek 3.2.
Obszar aktywny przycisku w ujęciu Hit

Aby zróżnicować zawartość poszczególnych ujęć wewnątrz przycisku, musisz je najpierw przekształcić w ujęcia kluczowe. Pod tym względem listwa czasowa przycisku nie różni się niczym od listwy czasowej sceny ani listw innych obiektów, takich jak symbole graficzne (*Graphic*) lub filmowe (*Movie Clip*).

Zdarzenia

Inaczej niż było to w przypadku ujęć, do przycisków nie można bezpośrednio przypisać akcji. Instrukcje przypisane do ujęcia są wykonywane, kiedy film do niego przejdzie. W przypadku obiektu nie ma takiej jasności. Trzeba dokładnie określić, kiedy akcje mają być wykonane. Aby związać kod ze zdarzeniem dotyczącym myszy, trzeba użyć konstrukcji `on`:

```
on(zdarzenie_myszy) {
  // Lista
  // instrukcji
}
```

Dla przycisków dostępne są zdarzenia związane z myszą. Oczywiście, zamiast `zdarzenie_myszy` trzeba podstawić nazwę konkretnego zdarzenia:

- `press` oznacza sytuacje, kiedy klawisz myszy jest naciskany na obszarze przycisku,
- `release` określa puszczenie klawisza myszy w obrębie przycisku,
- `releaseOutside` dotyczy naciśnięcia klawisza myszy na przycisku i puszczenie go poza przyciskiem,
- `rollOut` określa przesunięcie kursora poza obszar przycisku,
- `rollOver` oznacza przesunięcie kursora na przycisk,
- `dragOut` dotyczy przesunięcia kursora poza obszar przycisku z naciśniętym klawiszem myszy,
- `dragOver` występuje po zdarzeniu `dragOut`, kiedy kursor jest przemieszczany z powrotem na przycisk,

- `keyPress ("klawisz")` jest wywoływane, jeśli został naciśnięty klawisz „`klawisz`”. Najłatwiej ustawić klawisz w trybie *Normal Mode* panelu *Actions*. Przejdź do okienka tekstowego przy zaznaczonej opcji *Key Press* i wciśnij wybrany klawisz.

Do jednego bloku `on` można przypisać kilka zdarzeń, oddzielając je przecinkami. To samo zdarzenie może wywołać wiele różnych bloków kodu.

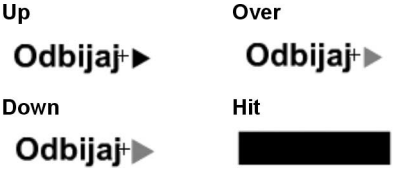
```
on (release, keyPress "<Space>") {
    fscommand("fullscreen", "true");
    play();
}
```

W trybie *Normal Mode* instrukcja `on` jest dodawana automatycznie, gdy tylko przypiszemy do przycisku akcję. Flash stosuje wtedy domyślnie zdarzenie `on(release)`. Możemy oczywiście w każdej chwili zmienić je na inne. Wystarczy zaznaczyć linię `on(release)`, by w polu parametrów akcji pojawiły się wszystkie opisane wyżej opcje.

Przykład użycia przycisków

Do animacji z piłką dodajmy sterowanie przyciskami. Film będzie składał się z trzech części — odbijania piłki, upuszczenia jej i podniesienia (stąd można wrócić do odbijania). W prosty sposób utworzymy płynne przejście do upuszczenia piłki z dowolnego momentu odbicia.

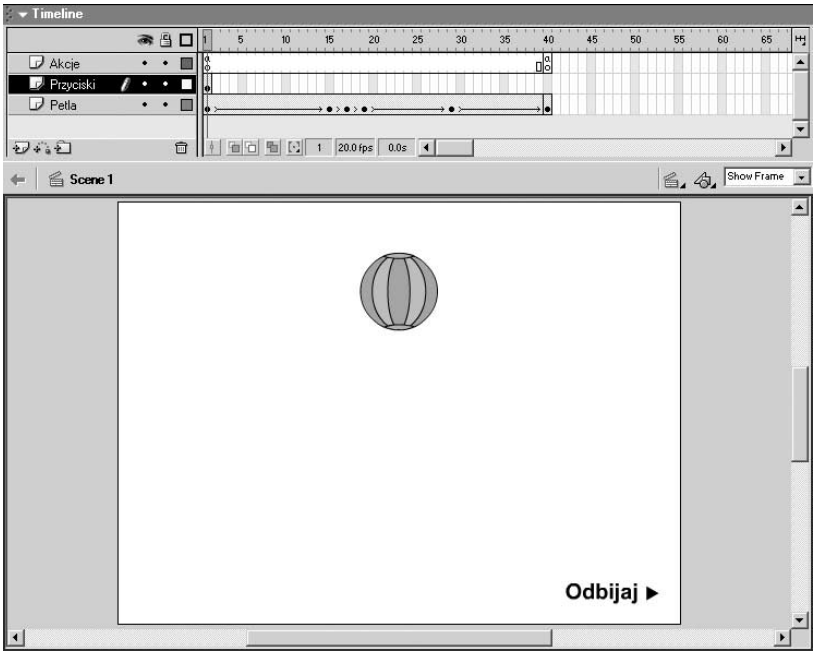
1. Otwórz poprzedni przykład z animacją piłki.
2. Warstwę, na której znajduje się animowana piłka, nazwij *Petla*.
3. Usuń z pierwszego ujęcia akcję `play()`.
4. Utwórz warstwę *Akcje*. Do pierwszego jej ujęcia przypisz akcję `stop()`. Utwórz ujęcie kluczowe w czterdziestej klatce.
5. Przenieś akcję z 40. ujęcia warstwy *Petla* do 40. ujęcia na warstwie *Akcje*. W oknie *Actions* możesz używać zwykłej metody wycinania i wklejania tekstu (*Ctrl*+*X*, *Ctrl*+*V*).
6. Dodaj warstwę *Przyciski*. Utwórz przycisk (obiekt typu *Button*) o nazwie „Odbijaj” (rysunek 3.3). Umieść w nim napis „Odbijaj”, utworzony narzędziem *Text*. O polach tekstowych opowiemy więcej za chwilę; na razie upewnij się tylko, że utworzony napis jest polem tekstowym typu *Static*. Spróbuj zróżnicować wygląd ujęć *Up*, *Over* i *Down* (nie zapomnij przekształcić ich najpierw w ujęcia kluczowe), a w ujęciu *Hit* umieść prostokąt nieco większy od napisu.



Rysunek 3.3. *Przykładowe ujęcia przycisku „Odbijaj”*

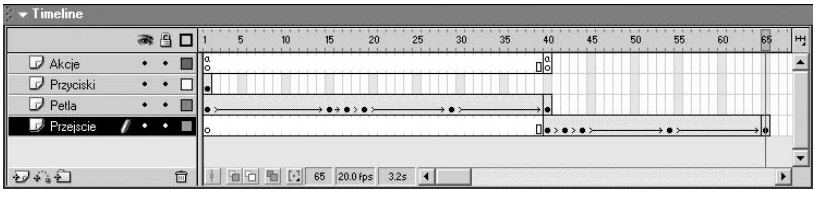
7. Gdy zakończysz edycję wewnątrz obiektu, umieść jego klon w pierwszym ujęciu warstwy *Przyciski* (rysunek 3.4). Zaznacz go i otwórz okno *Actions*. Dodaj do przycisku akcję `play`. W oknie *Actions* pojawi się skrypt:

```
on (release) {
  play();
}
```



Rysunek 3.4. *Klon przycisku umieszczony w pierwszym ujęciu kluczowym warstwy Przyciski*

8. Utwórz warstwę *Przejscie*. Na warstwie *Petla* zaznacz ujęcia od 15. do 40. i wybierz z menu kontekstowego polecenie *Copy Frames*. Zaznacz czterdzieste ujęcie na warstwie *Przejscie* i użyj polecenia *Paste Frames* aby wkleić ujęcia (rysunek 3.5).



Rysunek 3.5. *Warstwa Przejscie*

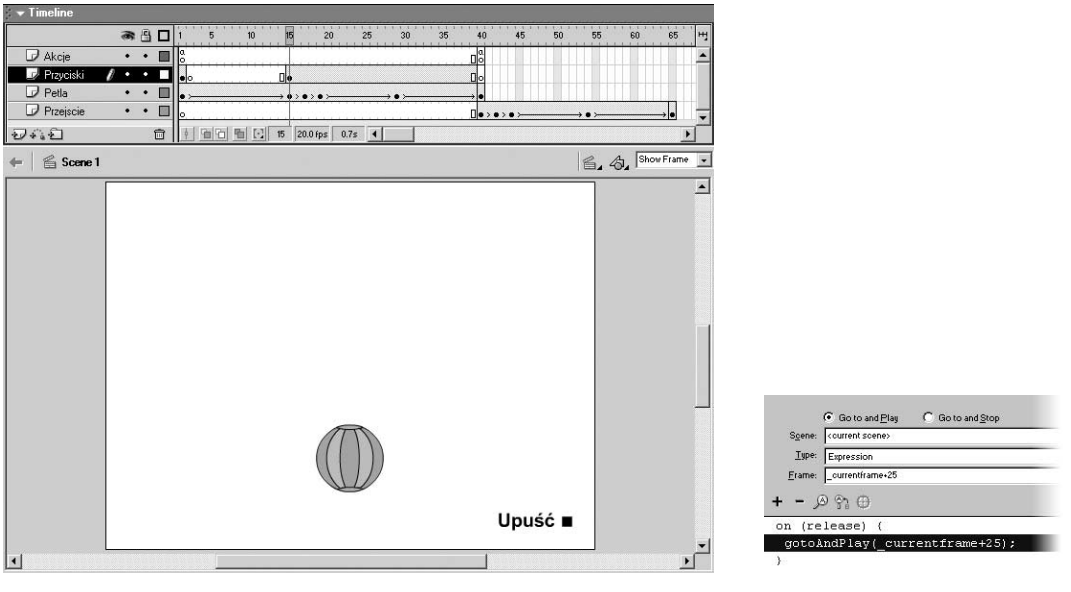
Można zaznaczyć wiele ujęć naraz przeciągając myszą wzdłuż listwy czasowej, lecz łatwo wtedy niechcący przenieść ujęcia kluczowe w inne miejsce. W związku z tym lepiej użyć innej metody: zaznacz pierwsze z szeregu ujęć (tutaj piętnaste), przytrzymaj *Shift* i kliknij ostatnie (tu: czterdzieste).

Kopię ujęć z pętli stworzyliśmy, aby zapewnić płynne przejście do następnej części animacji. Można zrobić to lepiej, ale postaramy się wykorzystać już poznane metody.

9. Na warstwie *Przyciski* dodaj puste ujęcia kluczowe w drugiej, piętnastej i czterdziestej klatce.

10. Utwórz podobny do poprzedniego przycisk „Upusc" (rysunek 3.6). Jego klon wstaw do piętnastego ujęcia warstwy *Przyciski*. Przypisz do niego poniższe akcje.

```
on (release) {
  gotoAndPlay(_currentframe + 25);
}
```

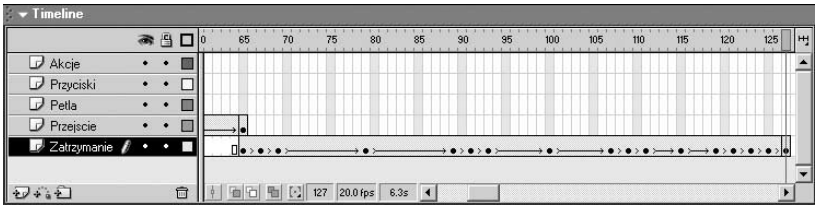


Rysunek 3.6. *Przycisk „Upusc" oraz parametry przypisanej mu akcji goto (Flash MX)*

Nie mówiliśmy jeszcze o właściwościach, ale `_currentframe` oznacza tu aktualne ujęcie.

Aby wpisać wyrażenie `_currentframe + 25` w polu *Frame* parametrów akcji `goto` wybierz z listy *Type* pozycję *Expression*.

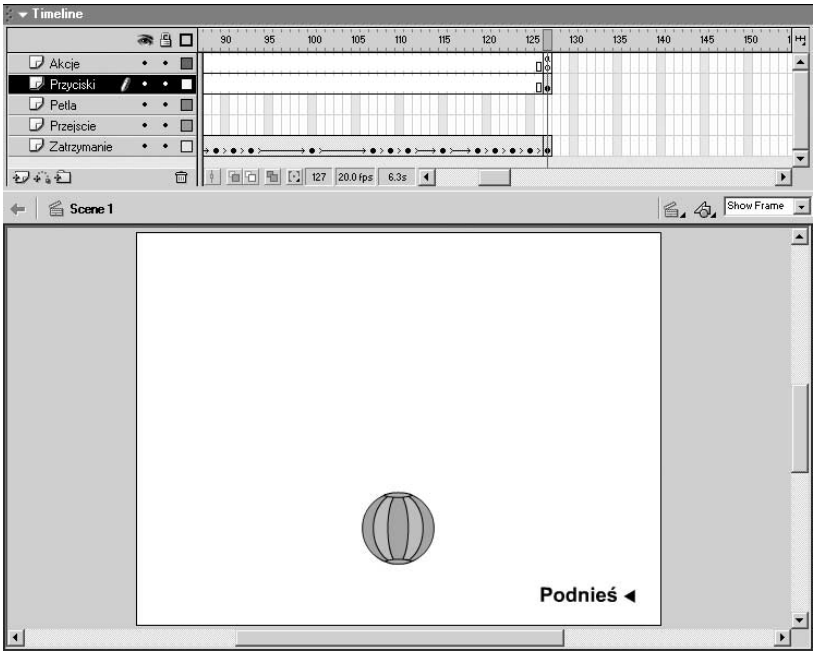
11. Utwórz warstwę *Zatrzymanie*. Skopiuj ujęcie z klatki sześćdziesiątej piątej warstwy *Przejście* do tej samej klatki na warstwie *Zatrzymanie*. Dodaj po nim animację kilku słabnących odbić i zatrzymania piłki (rysunek 3.7).



Rysunek 3.7. *Animacja na warstwie Zatrzymanie*

12. W ostatnim ujęciu animacji piłka powinna znaleźć się na takiej wysokości jak w ujęciu piętnastym.

13. W klatce, w której znajduje się ostatnie ujęcie animacji z warstwy *Zatrzymanie*, dodaj puste ujęcia kluczowe na warstwach *Akcje* i *Przyciski*. W ujęciu na warstwie *Akcje* dodaj akcję `stop()` a w ujęciu warstwy *Przyciski* umieść klon kolejnego przycisku: „Podnies" (rysunek 3.8).



Rysunek 3.8. *Przycisk „Podnies"*

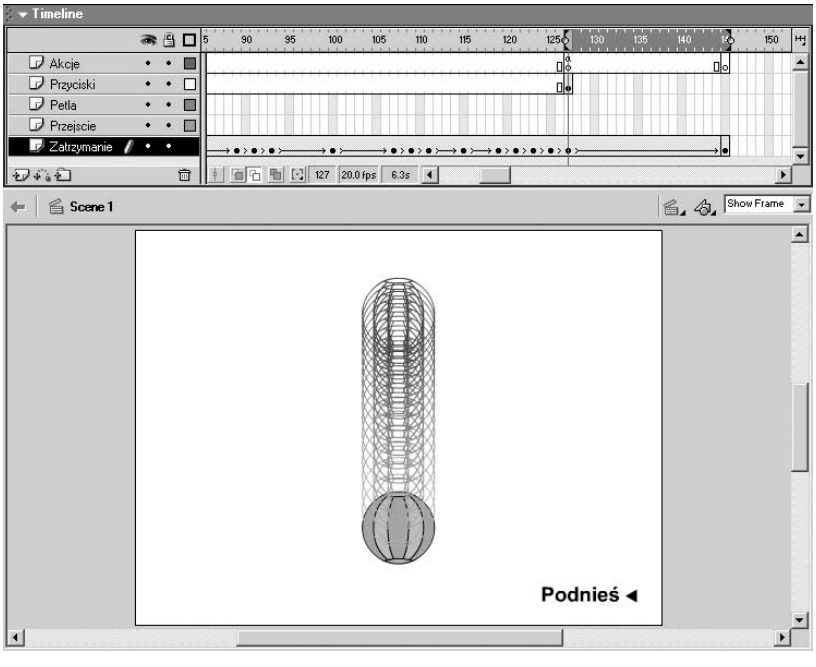
14. Do przycisku dodaj akcję:

```
on (release) {
  play();
}
```

15. Ostatniemu ujęciu na warstwie *Zatrzymanie* przypisz tweening *Motion* z parametrem *Ease* ustawionym na 50. Kilkanaście klatek za tym ujęciem dodaj ujęcie z podniesioną piłką (jak w pierwszej klatce filmu).

16. Na warstwie *Akcje* utwórz puste ujęcie kluczowe w tej klatce, w której znajduje się ostatnie ujęcie na warstwie *Zatrzymanie* (rysunek 3.9). Do nowego ujęcia dodaj akcję:

```
gotoAndStop(1);
```

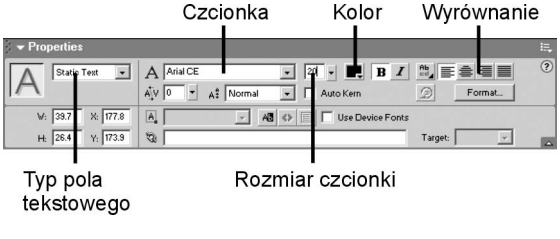


Rysunek 3.9. *Podnoszenie obiektu „Pilka”*

17. Przetestuj film (*Ctrl+Enter*). Przebieg animacji jest teraz sterowany za pomocą przycisków.

3.2. Pola tekstowe

Do tworzenia pól tekstowych służy narzędzie *Text*. Parametry narzędzia i zaznaczonych pól tekstowych pojawiają się w panelu *Properties* (rysunek 3.10).

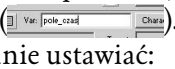


Rysunek 3.10. *Parametry statycznego pola tekstowego*

Istnieją trzy typy pól tekstowych:

- Pole typu *Static Text* po prostu wyświetla napis. Nie można wpływać na nie przez *ActionScript*.
- Pole *Dynamic Text* służy do wyświetlania tekstu, który można zmienić w czasie działania programu. Może posiadać przypisany identyfikator (pole *Instance Name*) i zmienną (pole *Var*).
- Pole typu *Input Text* pozwala pobrać tekst, który użytkownik wpisał do okna. Pole tego typu także może mieć identyfikator i skojarzoną zmienną.

Tekst początkowo przypisany do pola możesz edytować z aktywnym narzędziem *Text* (z aktywnym zwykłym kursorem wystarczy dwukrotne kliknięcie pola). Zaznaczone pole ma wtedy białe tło i zamienia się w okno tekstowe. Jego rozmiary można zmieniać, pociągając za narożny uchwyt.

Edycja tekstu w *ActionScript* jest możliwa poprzez identyfikator (właściwość *text*) lub zmienną. Zmienne są tworzone automatycznie i reprezentują tekst zawarty w polu. Nazwę zmiennej należy wpisać w polu *Var* (

). Taką zmienną traktuje się jak każdą inną — tekst można swobodnie ustawiać:

```
// w polu tekstowym wyświetli się zadany komunikat
pole_czas = "zostalo 25 sekund";

// tutaj w polu „tytul" zostanie podana nazwa obiektu
// który wywołał tę akcję
tytul = _name;
```

i odczytywać:

```
// użytkownik podaje dowolnie wybrany numer klatki filmu
// w polu „numer_ujęcia" i przenosi się tam klikając przycisk
on (release) {
   gotoAndStop(parseInt(numer_ujecia));
}

// w polu „output" pojawi się trzeci, czwarty i piąty znak
// pobrany z pola „input" wypełnionego przez użytkownika
output = input.substring(2, 5);
```

Zmienne przypisane do pól tekstowych nie są kasowane, kiedy film np. przechodzi do ujęcia w którym nie ma tego pola. Gdy pole znów się pojawi, będzie nadal zawierać tekst przypisany do zmiennej. Jeśli wiele pól tekstowych ma w polu *Var* podaną tę samą nazwę zmiennej, są one związane z jedną zmienną. W takim wypadku wszystkie pola będą zawierać ten sam tekst.

3.3. Mysz i klawiatura

Przyciski i pola tekstowe pozwalają nam prosto i skutecznie sterować filmem Flasha za pomocą myszy i klawiatury. Podobne możliwości otwiera przed nami *ActionScript*. Jednak aby zagłębić się w skryptowy język Flasha, potrzebujemy kilku nowych terminów.

Musimy przede wszystkim wyjaśnić znaczenie słowa „obiekt". Do tej pory posługiwaliśmy się tylko obiektami graficznymi — takimi, które można zobaczyć na ekranie, np. przyciskami i polami tekstowymi. W *ActionScript* możemy operować niektórymi z tych obiektów. Jednak mamy do dyspozycji jeszcze inne, wirtualne obiekty, np. obiekt daty (*Date*), który przechowuje dane o dacie i godzinie (możemy też pobrać za jego pomocą aktualny czas z komputera), oraz obiekt dźwięku (*Sound*). W tej chwili będzie nas interesować tylko obiekt myszy (*Mouse*) oraz klawisza (*Key*). Możliwe jest również tworzenie własnych obiektów.

Zarówno obiekty graficzne jak i wirtualne mają swoje właściwości (*properties*) i charakterystyczne metody działania (*methods*). Przykładowo, obiekt *Movie Clip* ma właściwość `_rotation`, dzięki której można określić kąt jego obrotu na scenie.

Właściwości _xmouse i _ymouse

`_xmouse` i `_ymouse` są właściwościami dostępnymi dla symboli i sceny (która także jest swego rodzaju obiektem), zawierającymi współrzędne kursora. Należy pamiętać, że właściwości te pobrane z obiektu reprezentują położenie myszy względem jego środka.

Jeśli chciałbyś stworzyć obiekt zastępujący zwykły kursor w twoim filmie, mógłbyś przypisać mu następujące akcje:

```
onClipEvent (mouseMove) {
   _x = _root._xmouse;
   _y = _root._ymouse;
}
```

`mouseMove`, `mouseDown` i `mouseUp` to związane z myszą zdarzenia obiektu typu *Movie Clip*.

Obiekt Mouse

Poprzez obiekt `Mouse` mamy dostęp do metod `show()` — pokaż kursor i `hide()` — ukryj kursor.

```
Mouse.hide();
```

Metody `addListener` i `removeListener` pozwalają dodawać do myszy zdarzenia i usuwać je. Można napisać na przykład:

```
mylistener = new Object;
mylistener.onMouseUp = function() {
  trace("Kliknieto prawy klawisz myszy");
}
Mouse.addListener(mylistener);
```

Można używać też takich przypisań do symboli graficznych:

```
playbutton.onMouseDown = function() {
  play();
}
```

Obiekt Key

Najczęściej używanymi metodami i właściwościami tego obiektu są:

- stałe reprezentujące kody klawiszy (na liście akcji *Objects/Movie/Key/Constants* lub *Built-in Classes/Movie/Key/Constants*), np. `Key.SPACE` czy `Key.RIGHT`,
- `isDown` — zwraca wartość typu *Boolean*, informując nas w ten sposób, czy naciśnięty jest klawisz o podanym kodzie. Przykładowo:

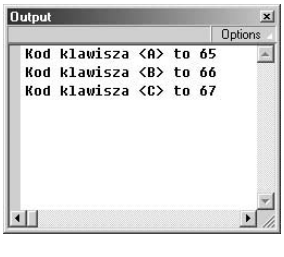
  ```
  if (Key.isDown(Key.SPACE)) { ... },
  ```

- `getCode` — zwraca kod ostatnio naciśniętego klawisza.

Do najczęściej używanych klawiszy można odwołać się przez obiekt *Key*. Kody pozostałych klawiszy znajdziesz w pomocy Flasha (*Using Flash/Keyboard Keys and Key Code Values lub* lub *ActionScript Reference Guide/Keyboard Keys and Key Code Values*). Jednak wygodniej jest utworzyć i zapisać w jakimś łatwo dostępnym miejscu film pomocniczy. Umieszczenie w nim obiektu typu *Movie Clip* z przypisanymi akcjami podobnymi do poniższych:

```
onClipEvent (keyDown) {
  trace("Kod klawisza <"+String.fromCharCode(Key.getCode())+"> to
  ↳"+Key.getCode());
}
```

sprawi, że po wybraniu polecenia *Test Movie* kody naciskanych klawiszy będą pojawiać się w oknie *Output* (rysunek 3.11).



Rysunek 3.11. *Działanie akcji trace — komunikaty podane jako parametr pojawiają się w oknie Output*