WPROWADZENIE DO HTML5

Nauka HTML5 i JavaScriptu na przykładzie gier

JEANINE MEYER

Helion

Tytuł oryginału: The Essential Guide to HTML5: Using Games to learn HTML5 and JavaScript

Tłumaczenie: Marek Pętlicki

ISBN: 978-83-246-4107-9

Original edition copyright 2010 by Jeanine Meyer.
All rights reserved.

Polish edition copyright 2012 by HELION SA.
All rights reserved.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from the Publisher.

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiejkolwiek postaci jest zabronione. Wykonywanie kopii metodą kserograficzną, fotograficzną, a także kopiowanie książki na nośniku filmowym, magnetycznym lub innym powoduje naruszenie praw autorskich niniejszej publikacji.

Wszystkie znaki występujące w tekście są zastrzeżonymi znakami firmowymi bądź towarowymi ich właścicieli.

Wydawnictwo HELION dołożyło wszelkich starań, by zawarte w tej książce informacje były kompletne i rzetelne. Nie bierze jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Wydawnictwo HELION nie ponosi również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Wydawnictwo HELION
ul. Kościuszki 1c, 44-100 GLIWICE
tel. 32 231 22 19, 32 230 98 63
e-mail: helion@helion.pl
WWW: http://helion.pl (księgarnia internetowa, katalog książek)

Pliki z przykładami omawianymi w książce można znaleźć pod adresem:
ftp://ftp.helion.pl/przyklady/whtmjs.zip

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
http://helion.pl/user/opinie/whtmjs
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

• Kup książkę
• Poleć książkę
• Oceń książkę

• Księgarnia internetowa
• Lubię to! » Nasza społeczność

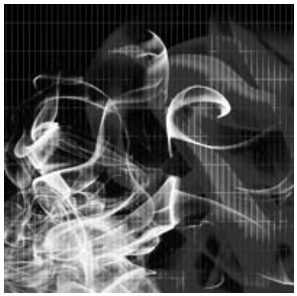
Spis treści

Kup książkę Poleć książkę

Kup książkę Poleć książkę

Kup książkę Poleć książkę

Kup książkę Poleć książkę



Rozdział 5

Pamięć

W tym rozdziale omówimy następujące zagadnienia:

- rysowanie wielokątów;
- wypisywanie tekstów na elemencie `canvas`;
- techniki programistyczne reprezentowania informacji;
- programowanie przerwy;
- obliczanie minionego czasu;
- metoda mieszania kart.

Wprowadzenie

Ten rozdział demonstruje dwie wersje gry karcianej znanej jako gra w pamięć (ang. *memory game*). Karty są układane na stole koszulką do góry, a gracz odkrywa dwie naraz (klikając je myszą), starając się dopasować pary. Program usuwa dopasowania ze stołu, ale jeśli karty nie są dopasowane, odwraca je znów koszulkami do góry. Gdy gracz znajdzie wszystkie dopasowania, gra prezentuje wynik w postaci czasu trwania gry.

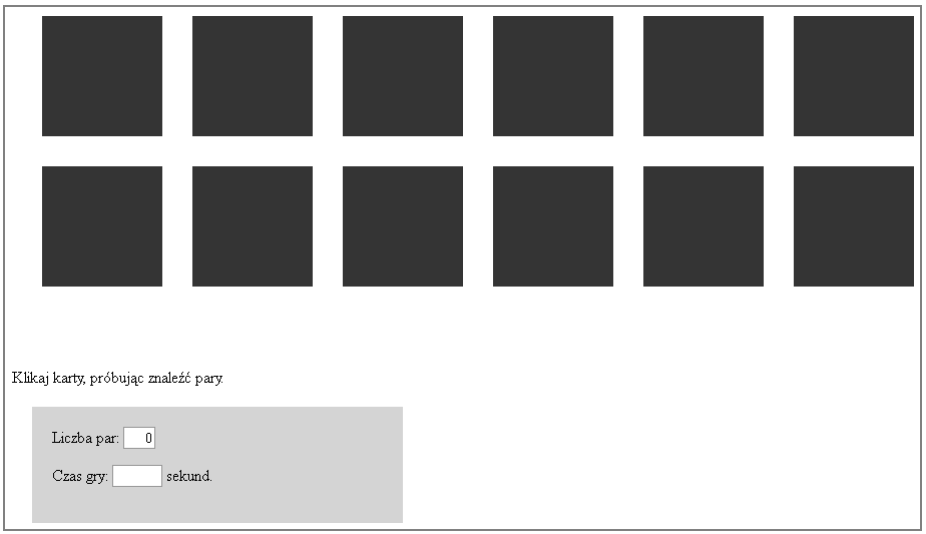
Pierwsza wersja gry będzie na kartach rysować wielokąty, druga wykorzystuje zdjęcia. Między tymi wersjami znajdziesz też więcej różnic, które mają na celu zobrazowanie funkcji HTML5, ale zachęcam do wyszukiwania nie tylko różnic, ale też jak największej liczby podobieństw tych aplikacji.

Rysunek 5.1 prezentuje początkowy ekran pierwszej wersji aplikacji. Gdy gracz ukończy grę, formularz prezentujący liczbę par wyświetli dodatkowo czas gry.

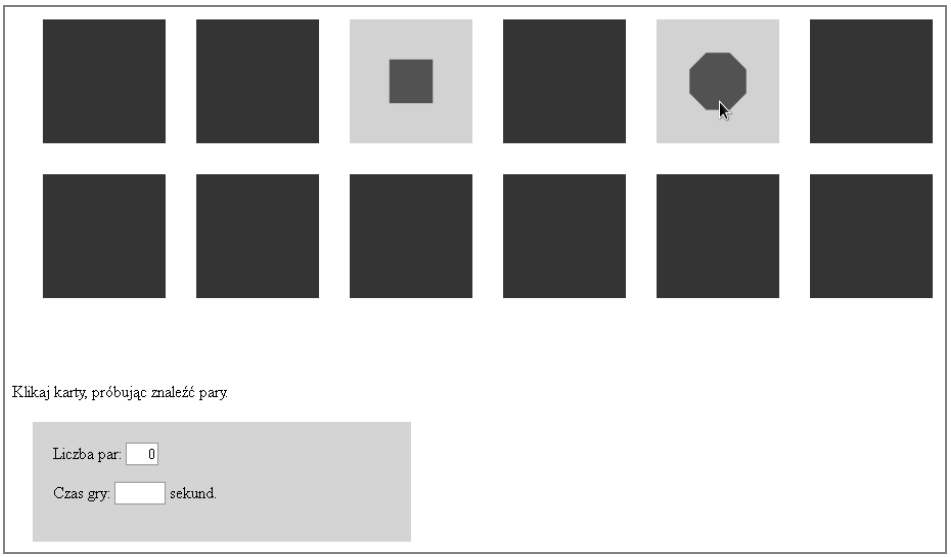
Rysunek 5.2 prezentuje efekt kliknięcia dwóch kart. Wielokąty na kartach nie pasują, więc po chwili przerwy program „odwraca" karty, wyświetlając ich koszulki.

Gdy dwie karty pasują do siebie, aplikacja usuwa je ze stołu i modyfikuje w formularzu liczbę dopasowań (rysunek 5.3).

Kup książkę Poleć książkę



Rysunek 5.1. Ekran początkowy gry „Pamięć”

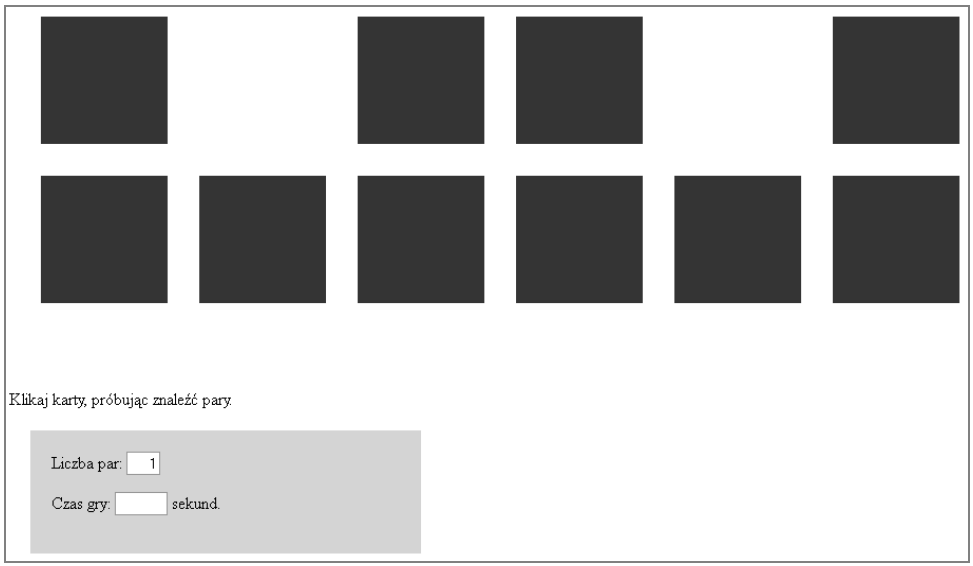


Rysunek 5.2. Dwie karty odkryte, nie ma pary

Rysunek 5.4 przedstawia wynik końcowy: w tym przypadku 6 dopasowań w czasie 28 sekund.

W drugiej wersji gry fronty kart zamiast wielokątów wykorzystują zdjęcia osób. Mimo że wiele gier tego typu dopasowania ma zdefiniowane jako identyczne obrazy, to w tym przypadku traktujemy tę sytuację jako „2 kier pasuje do 2 karo” (czerwone kolory) w zwykłej talii kart. Innymi słowy: dopasowanie jest zdefiniowane jako dwa różne zdjęcia tej samej osoby. Takie założenie gry wymaga zdefiniowania specjalnej struktury danych, która określi dopasowania. Ta wersja gry dodatkowo demonstruje technikę wypisywania tekstu na elemencie `canvas`, co przedstawia rysunek 5.5.

Kup książkę Poleć książkę



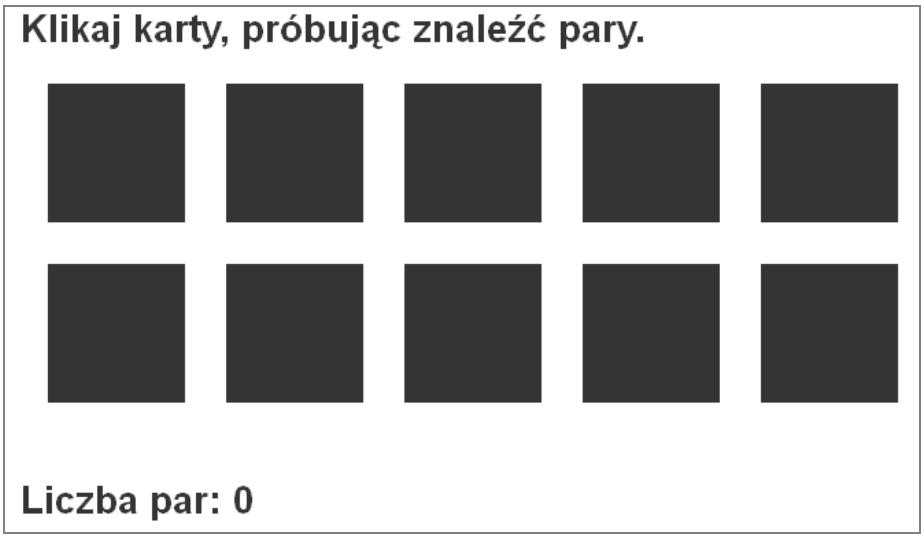
Rysunek 5.3. Aplikacja usunęła dwie dopasowane karty

Rysunek 5.4. Pierwsza wersja gry: widok ekranu po ukończeniu

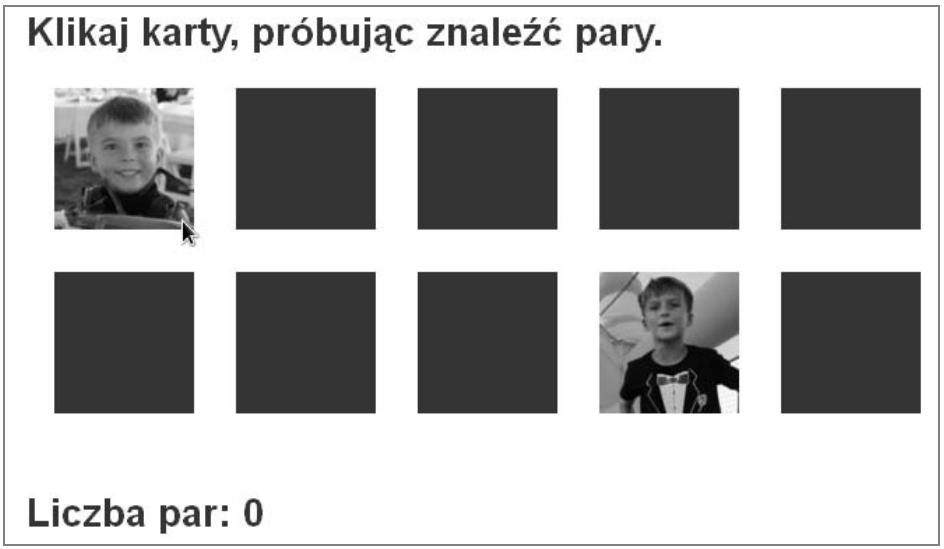
Przykład sytuacji po kliknięciu dwóch kart prezentuje rysunek 5.6.

Ponieważ odkryte karty zawierają zdjęcia dwóch różnych osób, po krótkiej przerwie, pozwalającej graczowi przyjrzeć się obrazom, gra odwraca karty i pozwala graczowi spróbować ponownie. Rysunek 5.7 przedstawia sytuację dopasowania: dwa różne zdjęcia tej samej osoby.

Kup książkę Poleć książkę



Rysunek 5.5. Druga wersja gry „Pamięć", ekran początkowy

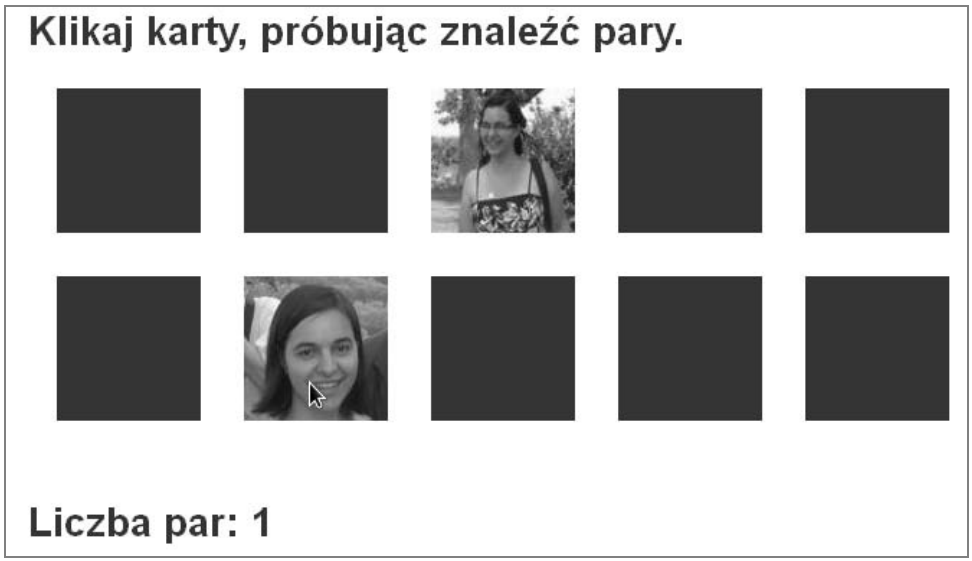


Rysunek 5.6. Dwie odkryte karty, brak dopasowania

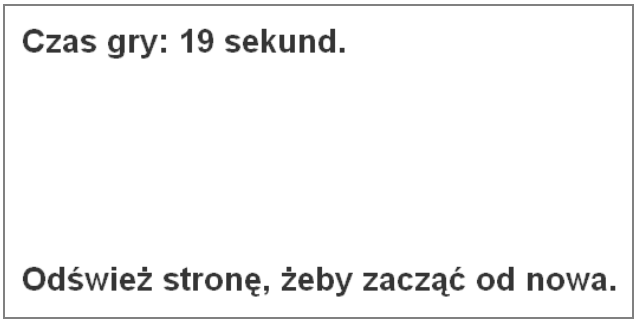
Dopasowane zdjęcia są usuwane ze stołu. Gdy zostaną usunięte wszystkie karty, na ekranie zostaje wyświetlony całkowity czas gry wraz z instrukcją, w jaki sposób rozpocząć grę od nowa, co przedstawia rysunek 5.8.

W tę grę można grać, wykorzystując zdjęcia dostępne wraz z kodem HTML na serwerze FTP. Dużo więcej frajdy sprawia jednak gra ze zdjęciami znanych sobie osób. Można zacząć od zestawu dwóch-trzech par zdjęć i stopniowo skompletować zdjęcia całej rodziny, klasy albo klubu. W pierwszej wersji gry zamiast wielokątów można zaimplementować własne, bardziej efektowne kreacje.

Kup książkę Poleć książkę



Rysunek 5.7. Udane dopasowanie: dwa różne zdjęcia tej samej osoby



Rysunek 5.8. Ostatni ekran gry (druga wersja). Wszystkie zdjęcia zostały dopasowane, więc na stole nie ma już ani jednej karty

Podstawowe wymagania

Komputerowa wersja gry w karty wymaga reprezentowania koszulek (tylnej strony) kart, które są identyczne, oraz ich frontów. W naszych przykładach w tej drugiej roli zastosujemy wielokąty i zdjęcia. Aplikacja musi umieć stwierdzić, czy odkryte karty pasują do siebie i gdzie na stole znajdują się poszczególne karty. Dodatkowo gracz potrzebuje informacji zwrotnych dotyczących tego, co się dzieje w grze. W rzeczywistej grze gracze odwracają po dwie karty i sprawdzają, czy znaleźli dopasowanie, co zajmuje chwilę. Jeśli dopasowania nie ma, karty są ponownie odwracane koszulką do góry.

Program komputerowy musi wyświetlać fronty wybranych kart, a po odkryciu drugiej karty musi na chwilę zatrzymać działanie, aby gracz zobaczył, co zostało odkryte. Ta pauza jest przykładem działania podejmowanego przez komputer w celu zasymulowania sytuacji świata rzeczywistego, czyli

Kup książkę Poleć książkę

w tym przypadku prawdziwej gry w karty. Aplikacja powinna również wyświetlić aktualną liczbę znalezionych par, a po zakończeniu gry ilość czasu, jaki upłynął od jej rozpoczęcia. Wersje aplikacji wykorzystujące wielokąty i zdjęcia różnią się nieco w zakresie sposobu prezentowania tych informacji graczowi.

Oto podsumowanie funkcji, jakie muszą realizować te dwie gry:

- rysowanie koszulek kart;
- losowanie kolejności kart na stole w celu uniknięcia powtarzania się tego samego układu;
- wykrywanie kliknięcia w kartę i rozróżnianie między pierwszym a drugim kliknięciem;
- po rozpoznaniu kliknięcia wyświetlenie frontu właściwej karty przez narysowanie wielokąta (wersja pierwsza) lub wyświetlenie zdjęcia (wersja druga);
- usunięcie dopasowanych par;
- podejmowanie odpowiednich reakcji nawet w przypadku, gdy gracz wykonuje pozornie bezsensowne działania, jak dwukrotne kliknięcie tej samej karty lub pustego miejsca na stole, na którym wcześniej leżała karta.

Funkcje HTML5, CSS i JavaScriptu

Omówmy funkcje HTML5 i JavaScriptu specyficzne dla gier implementowanych w tym rozdziale. Ponownie naszą implementację oprzemy na wiedzy zdobytej do tej pory, dotyczącej ogólnej struktury dokumentów HTML, umiejętności rysowania na elemencie `canvas` czworokątów, obrazów i ścieżek złożonych z odcinków, wykorzystania formularzy i tablic.

Nowościami w tym rozdziale będą: wykorzystanie zdarzenia zaplanowanego w czasie, użycie obiektu `Date` do obliczania czasu, jaki upłynął, wypisywanie tekstu na elemencie `canvas` oraz kilka użytecznych technik programowania, które przydadzą się też w innych aplikacjach.

Podobnie jak w poprzednich rozdziałach, ten podrozdział omawia funkcje HTML oraz techniki programowania w ujęciu ogólnym. Kod przykładowych aplikacji znajduje się w podrozdziale „Zbuduj własną aplikację". Jeśli chcesz, możesz przeskoczyć do tego podrozdziału i zobaczyć kod, ale sugeruję, żebyś wrócił w to miejsce, żeby zapoznać się z objaśnieniami technik użytych w kodzie.

Reprezentowanie kart

Każdy, kto trzymał w ręce prawdziwą kartę, wie jak ona wygląda. Z jednej strony znajduje się rysunek informujący o znaczeniu karty, a z drugiej wzór ozdobny (nazywany koszulką lub rewersem), który jest identyczny na wszystkich kartach z tej samej talii. Gdy rozłożymy karty na stole, wiemy doskonale, gdzie się znajdują, wiemy też, które karty są odwrócone frontem, a które koszulką do góry. Komputer musi te informacje zapisać w jakiejś strukturze (*zakodować*). Kodowanie informacji jest jednym z kluczowych elementów większości programów komputerowych, nie tylko gier.

W tym rozdziale (i w dalszej części książki) opiszę sposoby realizacji tego zadania. Należy pamiętać, że nie istnieje jedno, idealne podejście do tego problemu. Różne strategie budowania aplikacji jednak często bazują na podobnych założeniach.

Kup książkę Poleć książkę

Nasze podejście do zakodowania informacji o kartach opiera się na obiektach definiowanych przez programistę. Tworzenie takich obiektów w JavaScripcie wiąże się z pisaniem funkcji konstruktorów; w naszym przypadku będzie to funkcja `Card()`. Zaleta wykorzystania obiektów polega na tym, że w języku JavaScript możemy w prosty sposób wykorzystać notację z kropką, uzyskując dostęp do różnych elementów obiektu (informacji). W podobny sposób wykorzystywaliśmy obiekty w rozdziale 4., przy okazji armaty i procy.

Obiekt `Card` będzie wyposażony we właściwości przechowujące informacje o położeniu karty na stole (`sx` i `sy`) i jej wymiarach (`swidth` i `sheight`), funkcję rysującą koszulkę karty na elemencie `canvas` oraz informację o tym, w jaki sposób rysować front karty.

W przypadku pierwszej wersji gry, w której rysujemy wielokąty, informacja o froncie karty zawiera liczbę odcinków do narysowania (kod implementujący rysowanie takiego wielokąta omawiamy w dalszej części). W przypadku drugiej wersji gry, w której rysujemy zdjęcia, wartością będzie referencja `img` do obiektu `Image` definiującego obraz do wyświetlenia. Oprócz obiektu obrazu będziemy przechowywać liczbę, która powiąże w parę różne obrazy. Do rysowania obrazu z pliku wykorzystamy standardową metodę `drawImage()`.

Oczywiście, w komputerze karty nie istnieją jako fizyczne obiekty, z dwoma stronami. Aplikacja rysuje na elemencie `canvas` tę stronę karty, która powinna być widoczna. Do rysowania koszulki służy funkcja `flipback()`. Usunięcie karty polega na narysowaniu w miejscu karty prostokąta w kolorze tła.

Obie aplikacje wykorzystują funkcję pod nazwą `makedeck()`, która „przygotowuje talię", czyli generuje odpowiednie obiekty klasy `Card`. W przypadku pierwszej wersji w obiekcie `Card` zapisujemy liczbę wierzchołków wielokąta. Na tym etapie aplikacja jednak niczego nie rysuje. Wersja druga gry dodatkowo definiuje tablicę pod zmienną `pairs`, w której wymienione są nazwy plików fotografii. Wykorzystując tę tablicę, możesz podmienić zdjęcia fotografiami członków własnej rodziny lub przyjaciół.

> *Wskazówka: Jeśli chcesz wykorzystać moje przykładowe pliki, wystarczy, że pobierzesz z serwera FTP archiwa z przykładowymi kodami z tej książki. Jeśli zechcesz podmienić zdjęcia, musisz odpowiednio zmodyfikować kod aplikacji. W kodzie znajdują się informacje, w jaki sposób należy to zrobić.*

Funkcja `makedeck()` tworzy obiekty klasy `Image` i wykorzystuje tablicę `pairs` do ustawienia ich właściwości `src`. Gdy kod tworzy obiekty klasy `Card`, ustawia ich atrybuty `index`, które definiują pary, to znaczy pasujące fotografie mają taką samą wartość `index`. Tak samo jak w wersji wykorzystującej wielokąty, aplikacja na etapie inicjalizacji nie rysuje niczego na elemencie `canvas`. Po narysowaniu karty wyglądają identycznie, ale powiązane z nimi informacje są zróżnicowane. Po zainicjalizowaniu karty są ułożone zawsze w tej samej kolejności, mieszanie następuje później.

Informacje o pozycji (właściwości `sx` i `sy`) są interpretowane w inny sposób w każdej z wersji aplikacji. W przypadku wersji wykorzystującej zdjęcia informacja odnosi się do lewego górnego wierzchołka karty. W przypadku wersji wykorzystującej wielokąty te współrzędne identyfikują środek wielokąta. Znając współrzędne środka, możemy jednak wyliczyć współrzędne wierzchołka i odwrotnie.

Kup książkę Poleć książkę

Wykorzystanie obiektu Date do obliczana czasu

Musimy zmierzyć czas, jaki graczowi zajęło znalezienie wszystkich dopasowań. JavaScript udostępnia sposób mierzenia czasu. Kompletny kod znajdziesz w punkcie „Zbuduj własną aplikację", a w tym miejscu przedstawię jedynie fragmenty prezentujące sposób mierzenia czasu między dwoma zdarzeniami w kodzie JavaScript.

Wywołanie metody `Date()` generuje obiekt daty i czasu, np.:

```
starttime = new Date();
starttime = Number(starttime.getTime());
```

Powyższy fragment kodu zapisuje w zmiennej `starttime` liczbę milisekund (tysięcznych sekundy), jakie upłynęły od początku 1970 roku (w tym miejscu nie ma znaczenia, dlaczego czas jest mierzony od początku 1970 roku).

Gdy nasze gry „Pamięć" stwierdzą, że gracz znalazł wszystkie dopasowania, kreator `Date()` jest wywoływany ponownie:

```
var now = new Date();
var nt = Number(now.getTime());
var seconds = Math.floor(.5 + (nt - starttime) / 1000);
```

Oto analiza tego kodu:

1. Stworzenie nowego obiektu `Date` i przypisanie go zmiennej `now`.
2. Odczytanie czasu z obiektu `now` z użyciem metody `getTime()`. Wartość ta jest przekształcana na liczbę (funkcja `Number()`) i przypisywana zmiennej `nt`. To oznacza, że zmienna `nt` zawiera liczbę sekund, jakie upłynęły od początku 1970 roku do momentu wywołania metody `Date()`. Następnie program od tej wartości odejmuje zapisaną na początku gry zmienną `starttime`.
3. Dzielenie przez 1000 sprowadzi wartość do sekund.
4. Dodanie wartości `.5` spowoduje, że funkcja `Math.floor()` właściwie zaokrągli wartość do pełnych sekund.

Jeśli interesuje Cię większa dokładność pomiaru, możesz pominąć wyliczenie zaokrąglenia i dodawanie wartości `.5`.

Takiego kodu możesz użyć wtedy, gdy chcesz zmierzyć czas wykonania fragmentów programu.

Obsługa pauzy

Gdy gramy w tego typu grę, używając prawdziwych kart, nie odczekujemy świadomie przed odwróceniem kart na drugą stronę. Ale, jak wspominałam wcześniej, implementacja komputerowa musi odczekać chwilę, żeby dać graczowi czas na obejrzenie obydwu odkrytych kart. Jak pamiętasz z rozdziałów 3. i 4. (animacja odbijającej się piłki oraz strzelanie z armaty i procy), do definiowania opóźnienia wykonania kodu w równych odstępach czasu wykorzystywaliśmy funkcję `setInterval()`. Do odczekania ustalonego czasu przed kontynuowaniem działania gry możemy wykorzystać inną funkcję o zbliżonym działaniu: `setTimeout()`. Pełny kod wykorzystujący tę funkcję znajdziesz

Kup książkę Poleć książkę

w podrozdziale „Zbuduj własną aplikację”. Zobaczmy, w jaki sposób definiuje się tego typu zdarzenie czasowe i co się stanie, gdy czas oczekiwania upłynie.

Funkcja `setTimeout()` ustawia pojedyncze zdarzenie czasowe, które jest wywoływane z ustalonym opóźnieniem. Funkcja `choose()`, wywoływana po tym, gdy gracz kliknie element `canvas`, sprawdza wartość zmiennej `firstpick` w celu określenia, czy kliknięta została pierwsza, czy druga karta z rzędu. W każdym z tych przypadków program rysuje front karty w tym samym miejscu, w którym znajdował się rysunek koszulki karty. Jeśli kliknięcie było drugim z rzędu i karty pasują do siebie, kod ustawia wartość zmiennej `matched` na `true` (jeśli karty nie pasują, wartość ta ustawiana jest na `false`). Jeśli gra nie jest jeszcze skończona, wywoływany jest następujący kod:

```
setTimeout(flipback, 1000);
```

To spowoduje, że po upłynięciu jednej sekundy (1000 milisekund) zostanie wywołana funkcja `flipback()`. Funkcja `flipback()` wykorzystuje zmienną `matched`, która pozwala jej zdecydować, czy w miejscu odkrytych kart ma narysować ich koszulki (wartość `false`), czy usunąć karty ze stołu, rysując w ich miejscu kolor tła (wartość `true`).

Funkcji `setTimeout()` można użyć do definiowania indywidualnych zdarzeń czasowych. Wystarczy określić odstęp czasu oraz funkcję, która ma być wywołana z takim opóźnieniem. Pamiętaj jedynie, że jednostką czasu jest milisekunda.

Rysowanie tekstu

Standard HTML5 definiuje mechanizm rysowania tekstów na elemencie `canvas`. Dzięki temu mamy dostęp do bardziej dynamicznego, elastycznego sposobu wyświetlania tekstów niż dawniej. Łącząc tekst z prostokątami, liniami, łukami i obrazami, możemy uzyskać ciekawe efekty. W tym punkcie opiszę ogólne zasady posługiwania się tekstem w połączeniu z elementem `canvas`, przedstawię też prosty przykład do samodzielnego wypróbowania. Jeśli chcesz, możesz pominąć ten fragment i przejść do podrozdziału „Zbuduj własną aplikację”, w którym omawiam kod aplikacji prezentowanej na rysunkach od 5.5 do 5.8, czyli gry „Pamięć” z wykorzystaniem fotografii.

W celu umieszczenia tekstu na elemencie `canvas` napiszemy kod, który ustawia krój czcionki, a następnie wywołuje metodę `fillText()`, która rysuje tekst na elemencie `canvas` w ustalonym położeniu `x`, `y`. Poniższy przykład tworzy kilka napisów z wykorzystaniem różnych krojów czcionek (koniecznie przeczytaj uwagę w dalszej części tekstu).

```
<html>
  <head>
    <title>Czcionki</title>
    <script type="text/javascript">
      var ctx;
      function init(){
        ctx = document.getElementById('canvas').getContext('2d');
        ctx.font="15px Lucida Handwriting";
        ctx.fillText("this is Lucida Handwriting", 10, 20);
        ctx.font="italic 30px HarlemNights";
        ctx.fillText("italic HarlemNights",40,80);
        ctx.font="bold 40px HarlemNights"
        ctx.fillText("HarlemNights",100,200);
```


Kup książkę Poleć książkę

```
        ctx.font="30px Accent";
        ctx.fillText("Accent", 200,300);
      }
    </script>
  </head>
  <body onLoad="init();">
    <canvas id="canvas" width="900" height="400">
      Twoja przeglądarka nie obsługuje elementu canvas standardu HTML5.
    </canvas>
  </body>
</html>
```

Ten dokument uruchomiony w przeglądarce WWW daje efekt przedstawiony na rysunku 5.9.

Rysunek 5.9. Tekst z użyciem różnych czcionek wyświetlony na elemencie canvas z użyciem metod font() i fillText()

> *Ostrzeżenie: Pamiętaj, aby stosować czcionki dostępne w komputerach graczy. W rozdziale 10. dowiesz się, w jaki sposób wykorzystywać mechanizm standardu CSS pod nazwą* `font-family`*, który udostępnia systematyczny sposób definiowania preferowanych czcionek oraz czcionek zastępczych, jeśli podstawowa czcionka nie zostanie znaleziona.*

Należy pamiętać, że mimo tego, iż tekst na elemencie `canvas` wygląda zupełnie „normalnie", to należy go postrzegać tak jak druk na papierze albo obraz, a nie tekst, jaki znamy ze stron WWW, który można zaznaczać, lub pole tekstowe, którego wartość można modyfikować. Oznacza to, że w celu zmodyfikowania tekstu musimy napisać kod JavaScript, który wymaże starą zawartość. Wykorzystujemy do tego atrybut `fillStyle`, który ustawiamy na wartość zmiennej `tablecolor`, a następnie wywołujemy metodę `fillRect()` z odpowiednimi współrzędnymi.

Kup książkę Poleć książkę

Po narysowaniu ścieżki tekstu musimy wypełnić ją kolorem. Właściwość `fillStyle` musimy więc ustawić na inną wartość niż kolor tła (`tablecolor`). Możemy na przykład wykorzystać ten sam kolor, którego użyliśmy do namalowania koszulek kart. Krój czcionki ustawiamy w ramach inicjalizacji gry:

```
ctx.font = "bold 20pt sans-serif";
```

Użycie czcionki `sans-serif` ma sens, ponieważ krój ten należy do standardowych krojów internetowych, dostępnych na wszystkich komputerach.

Oto kod wypisujący liczbę dopasowań w danym momencie gry:

```
ctx.fillStyle = tablecolor;
ctx.fillRect(10, 340, 900, 100);
ctx.fillStyle=backcolor;
ctx.fillText("Liczba par: " + String(count), 10, 360);
```

Pierwsze dwie instrukcje wymazują aktualny wynik, a dwie kolejne wypisują aktualny. Wyrażenie `"Liczba par: " + String(count)` wymaga nieco więcej wyjaśnień. Jego funkcja jest następująca:

- Przekształca na łańcuch znaków wartość zmiennej `count`, która jest liczbą.
- Łączy ten łańcuch znaków z tekstem `"Liczba par: "`.

Łączenie łańcuchów znaków (konkatenacja) jest realizowane za pomocą operatora dodawania (plus), co demonstruje wieloznaczną interpretację tego operatora. Jeśli obydwa operandy są liczbami, znak plus powoduje dodawanie arytmetyczne. Jeśli operandy są łańcuchami znaków, operator plus powoduje połączenie tych łańcuchów w jeden (konkatenację). Sytuację, w której jeden operator ma wiele znaczeń, nazywa się **przeciążaniem operatorów**.

Co zrobi JavaScript, jeśli jeden z operatorów jest łańcuchem znaków, a drugi liczbą? Odpowiedź jest zależna od ich kolejności. W internecie można znaleźć mnóstwo przykładów, w których autorzy nie dokonali konwersji typów operandów, a konkatenacja znaków działa prawidłowo. Tajemnica leży we właściwej kolejności operacji.

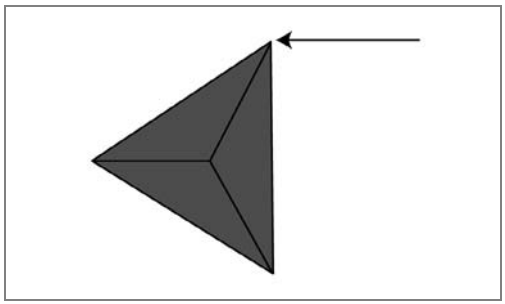
Ja jednak sugeruję, żeby unikać niespodzianek i zawsze stosować operandy właściwych typów. Jeśli nagle okaże się, że program dodaje dwie jedynki i w efekcie otrzymujemy 11, a następnie 111, zamiast 1, 2, 3, to znak, że łączymy znaki, zamiast dodawać liczby i że należy dokonać konwersji typów.

Rysowanie wielokątów

Rysowanie wielokątów to dobra ilustracja możliwości HTML5 w zakresie rysowania. Aby zrozumieć proces tworzenia kodu używanego do rysowania wielokątów, warto wyobrazić sobie taki wielokąt jako kształt przypominający koło rowerowe ze szprychami rozchodzącymi się z jego środka do każdego z wierzchołków. Szprycha nie pojawia się na rysunku, ale służy jako pomoc w zrozumieniu procesu rysowania wielokąta. Koncepcję tę objaśnia rysunek 5.10, na przykładzie trójkąta.

W celu zmierzenia kąta między szprychami dzielimy wartość `2 * Math.PI` (reprezentującą pełne koło) przez liczbę wierzchołków. Wartość kąta wykorzystamy do wywołania metody `moveTo()`, która przesunie pozycję ścieżki w odpowiednie miejsce.

Kup książkę Poleć książkę



Rysunek 5.10. Reprezentacja trójkąta jako kształtu geometrycznego ze szprychami rozchodzącymi się z jego środka do wierzchołków. Strzałka wskazuje punkt, od którego zaczynamy rysowanie

Program rysuje wielokąty jako wypełnione ścieżki rozpoczynające się od punktu wskazanego strzałką na rysunku 5.10, wyznaczonego jako połowa wartości kąta między szprychami. Aby dotrzeć do tego punktu, wywołujemy metodę `moveTo()` z wartością promienia radius oraz wykorzystujemy funkcje `Math.sin()` i `Math.cos()`. Następnie metodę `lineTo()` wywołujemy `n - 1` razy, postępując w kierunku zgodnym z ruchem wskazówek zegara. W przypadku trójkąta `n – 1` to dwa kolejne wierzchołki. W przypadku ośmiokąta będzie to siedem kolejnych wierzchołków. Po zakończeniu pętli wywołujemy metodę `fill()`, która spowoduje wyświetlenie wypełnionego wielokąta. Kompletny kod z komentarzami znajdziesz w podrozdziale „Zbuduj własną aplikację".

Uwaga: Rysowanie wielokątów zajmuje czas, ale w tej aplikacji to nie jest problem. Gdyby program wymagał rysowania dużej liczby skomplikowanych grafik, warto zastanowić się nad technikami przyspieszenia ich rysowania, jak na przykład użycie gotowych grafik. To rozwiązanie wymaga jednak pobierania dodatkowych plików, co z kolei również zajmuje czas. Należy przeprowadzić eksperymenty w celu stwierdzenia, które rozwiązanie jest najbardziej korzystne w danej sytuacji.

Mieszanie kart

Jak wspomniałam wcześniej, gra „Pamięć" wymaga tego, aby kolejność kart była różna przy każdej rozgrywce. Najlepszy sposób realizacji tego zadania wcale nie jest zagadnieniem trywialnym. W rozdziale 10. przy okazji gry w 21 znajdziesz odnośnik do artykułu omawiającego najefektywniejszy algorytm mieszania talii kart.

W przypadku gry „Pamięć" zaimplementujemy sposób, który wykorzystywałam, grając w tę grę jako dziecko. Razem z innymi graczami rozkładaliśmy karty, a następnie wybieraliśmy po dwie karty i zamienialiśmy je miejscami. Gdy uznaliśmy, że liczba zamian była wystarczająca, zaczynaliśmy właściwą rozgrywkę. W tym punkcie omówię sposób zaimplementowania tego pomysłu w naszej aplikacji. Funkcję `shuffle()` znajdziesz w podrozdziale „Zbuduj własną aplikację", w tabeli z pełnym kodem aplikacji.

W celu napisania kodu JavaScript niezbędnego do mieszania kart powinniśmy zdefiniować, co znaczy „wystarczająca liczba zamian". Możemy przyjąć, że taką liczbą jest trzykrotna liczba kart w talii

Kup książkę Poleć książkę

zapisanych w tablicy `deck`. Ale przecież nie mamy fizycznych kart, a jedynie interpretujące je informacje. W takim razie co będziemy zamieniać? Oczywiście, tylko informacje identyfikujące kartę. W przypadku pierwszej wersji gry, wykorzystującej wielokąty, będzie to liczba boków, czyli właściwość `info`. W przypadku wersji wykorzystującej zdjęcia będą to właściwości `info` i `img`.

Do losowania kart wykorzystamy wyrażenie `Math.floor(Math.random() * dl)`, gdzie `dl` określa liczbę kart w talii. W każdym przebiegu takie losowanie wykonamy dwukrotnie, po czym zamienimy miejscami wylosowane dwie karty. Losowanie może dwukrotnie wskazać tę samą kartę, co oznacza, że zostanie ona zastąpiona przez siebie samą, ale nie jest to powód do zmartwienia. Jeśli coś takiego się zdarzy, po prostu jeden przebieg nie przyniesie zmian kolejności kart w talii. Zaplanowaliśmy dużą liczbę takich zamian, więc jedna mniej przejdzie niezauważona.

Kolejnym wyzwaniem jest realizacja samej zamiany. Do tej operacji potrzebne nam będą pomocnicze zmienne, przechowujące tymczasowo nasze przenoszone wartości. W przypadku wersji wykorzystującej wielokąty wykorzystamy jedną zmienną, pod nazwą `holder`, a w przypadku wersji ze zdjęciami będą to dwie zmienne: `holderimg` i `holderinfo`.

Kliknięcia w karty

Następnym krokiem będzie wyjaśnienie ruchów graczy, a dokładniej obsługa kliknięć kart. Obsługę zdarzeń kliknięć myszą rozwiążemy w sposób podobny do tego, jaki znamy z rozdziału 4. Wykorzystamy metodę `addEventListener()`:

```
canvas1 = document.getElementById('canvas');
canvas1.addEventListener('click', choose, false);
```

Ten fragment kodu umieścimy w funkcji `init()`. Funkcja `choose()` musi zawierać kod wykrywający kliknięta kartę. Program wykorzysta współrzędne kliknięcia myszą na elemencie `canvas`. Technikę odczytu współrzędnych również znamy z rozdziału 4.

Niestety, różne przeglądarki obsługują zdarzenia myszy na różne sposoby. Omówiłam to pokrótce w rozdziale 4., ale przypomnę te informacje również w tym miejscu. Poniższy kod działa prawidłowo w przeglądarkach Chrome, Firefox i Safari:

```
if ( ev.layerX || ev.layerX==0) {
  mx= ev.layerX;
  my = ev.layerY;
} else if (ev.offsetX || ev.offsetX==0 ) {
  mx = ev.offsetX;
  my = ev.offsetY;
}
```

Ten kod działa prawidłowo, ponieważ w przypadku gdy przeglądarka nie obsługuje właściwości `ev.layerX`, jej wartość będzie interpretowana jako `false`. Jeśli istnieje, ale ma wartość `0`, również będzie interpretowana jako `false`, ale za to wyrażenie `ev.layerX==0` zwróci wartość `true`. Dzięki temu jeżeli program znajdzie użyteczną wartość `ev.layerX`, użyje jej. W przeciwnym razie kod sprawdzi istnienie właściwości `ev.offsetX`. Jeśli żadna z tych właściwości nie działa, zmienne `mx` i `my` nie zostaną przypisane.

Kup książkę Poleć książkę

Karty są prostokątami, więc łatwo będzie po prostu kolejno przejrzeć wszystkie karty i porównać ich współrzędne z pozycją myszy (`mx`, `my`) w momencie kliknięcia: wystarczy odczytać współrzędne lewego górnego wierzchołka oraz szerokość i wysokość karty. Warunek `if` skonstruujemy w następujący sposób:

```
if ((mx > card.sx) && (mx < card.sx + card.swidth) && (my > card.sy) && (my < card.sy +
card.sheight)) {
```

> *Uwaga: W następnym rozdziale zajmiemy się sposobami dynamicznego definiowania kodu HTML z poziomu JavaScriptu. Poznamy w nim sposoby definiowania zdarzeń dla poszczególnych elementów na ekranie zamiast całego elementu* `canvas`.

Wartość zmiennej `firstpick` ustawiamy na `true`, co sygnalizuje, że następne kliknięcie będzie pierwszym z dwóch w serii. Po kliknięciu pierwszej karty program zmieni tę wartość na `false`, a po kliknięciu drugiej ponownie na `true`. Tego typu zmienne, przełączane między dwoma stanami, często nazywa się **flagami** (ang. *flag*) lub **przełącznikami** (ang. *toggle*).

Zapobieganie oszustwom

Ten punkt omawia specyficzne zagadnienia dotyczące tej konkretnej gry, ale z tych przemyśleń wynika też ogólna nauka budowania interaktywnych aplikacji. Istnieją co najmniej dwa sposoby, w jakie gracz może próbować oszukać grę: klikając dwukrotnie na tej samej karcie lub klikając w miejscu, w którym znajdowała się wcześniej usunięta karta.

Przed pierwszym problemem zabezpieczymy się, umieszczając poniższy kod po warunku sprawdzającym, czy kliknięcie nastąpiło w obrębie karty:

```
if ((firstpick) || (i != firstcard)) break;
```

Ten wiersz kodu spowoduje wyjście z pętli `for`, jeśli indeks `i` jest prawidłowy, to znaczy jeśli kliknięta karta jest pierwszą w sekwencji lub drugą, ale inną od pierwszej.

Uniknięcie drugiego problemu (kliknięcia w miejscu, w którym już nie ma karty) wymaga więcej pracy. Gdy aplikacja usuwa kartę ze stołu, oprócz narysowania tła w miejscu karty może dodatkowo ustawiać specjalną wartość (np. `-1`) w jej właściwości `sx`. To będzie wirtualny znak, że karta jest usunięta. Operacja ta powinna być wykonana w funkcji `flipback()`. Funkcja `choose()` sprawdzi wartość właściwości `sx` (`sx >= 0`). Obydwa mechanizmy zapobiegania oszustwom można zaimplementować w następującej pętli `for`:

```
for (i = 0; i < deck.length; i++){
  var card = deck[i];
  if (card.sx >=0)
    if ((mx > card.sx) && (mx < card.sx + card.swidth) && (my > card.sy) && (my < card.sy +
    card.sheight)) {
      if ((firstpick)|| (i!=firstcard)) break;
    }
}
```


Kup książkę Poleć książkę

Mamy tu do czynienia z trzykrotnie zagnieżdżonymi instrukcjami warunkowymi. Druga z nich jest jedyną instrukcją pierwszej. Trzecia zawiera pojedyncze wyrażenie `break`, które powoduje wyjście z pętli. Z reguły zalecam stosowanie nawiasów klamrowych ({ i }) w instrukcjach warunkowych, ale powyższy przykład stanowi skróconą wersję, jedynie do celów demonstracyjnych.

Przejdźmy wreszcie do naszych dwóch wersji gry „Pamięć".

Zbuduj własną aplikację

Ten podrozdział zawiera kompletne kody dwóch wersji gry „Pamięć". Każda z nich wykorzystuje sporą liczbę funkcji, dlatego na początek zapoznaj się z ich zestawieniami w tabelach. Znajdziesz w nich nazwy funkcji oraz informacje, przez co są wywoływane i co same wywołują.
Tabela 5.1 zawiera zestawienie funkcji pierwszej wersji gry, wykorzystującej wielokąty. Niektóre z funkcji są wywoływane w ramach obsługi zdarzeń.

Tabela 5.1. Funkcje w wersji gry „Pamięć" wykorzystującej wielokąty

Funkcja	Wywoływana przez	Wywołuje
`init()`	obsługę zdarzenia `onLoad` elementu `body`	`makedeck()` `shuffle()`
`choose()`	obsługę zdarzenia myszy zdefiniowaną przez metodę `addEventListener()` w funkcji `init()`	`Polycard()` `drawpoly()` wywołaną jako metoda `draw()` wielokąta
`flipback()`	obsługę zdarzenia czasowego zdefiniowanego przez funkcję `setTimeout()` wywołaną w funkcji `choose()`	
`drawback()`	`makedeck()` i `flipback()` jako metodę `draw()` klasy `Polycard`	
`Polycard()`	`choose()`	
`shuffle()`	`init()`	
`makedeck()`	`init()`	
`Card()`	`makedeck()`	
`drawpoly()`	`choose()` jako metodę `draw()` klasy `Polygon`	

Tabela 5.2 zawiera kod pierwszej wersji gry z kompletnymi komentarzami. Analizując kod, zwróć uwagę na podobieństwa między tymi dwoma wersjami aplikacji, omówione wcześniej w tym rozdziale. Pamiętaj również, że to jest jeden z niezliczonej liczby sposobów zaprogramowania gry tego typu.

Tabela 5.2. Pełny kod pierwszej wersji gry „Pamięć" wykorzystującej wielokąty

`<html>`	Początek elementu `html`
`<head>`	Początek elementu `head`
`<title>Gra "Pamięć"</title>`	Kompletny element `title`

Kup książkę Poleć książkę

`<style>`	Początek elementu `style`
`form {`	Definicja stylu dla elementu `form`
`width:330px;`	Szerokość
`margin:20px;`	Margines zewnętrzny
`background-color:pink;`	Kolor
`Padding:20px;`	Wypełnienie wewnętrzne
`}`	Koniec definicji stylu
`input {`	Definicja stylu dla elementów pól formularza
`text-align:right;`	Wyrównanie do prawej, odpowiednie dla liczb
`}`	Koniec definicji stylu
`</style>`	Koniec elementu `style`
`<script type="text/javascript">`	Początek elementu `script`. Deklaracja `type` nie jest niezbędna, ale zalecana
`var ctx;`	Zmienna przechowująca graficzny kontekst elementu `canvas`
`var firstpick = true;`	Deklaracja i inicjalizacja zmiennej `firstpick`
`var firstcard;`	Deklaracja zmiennej przechowującej informację o pierwszej karcie
`var secondcard;`	Deklaracja zmiennej przechowującej informację o drugiej karcie
`var frontbgcolor = "rgb(251,215,73)";`	Kolor tła frontów kart
`var polycolor = "rgb(254,11,0)";`	Kolor wielokątów
`var backcolor = "rgb(128,0,128)";`	Kolor koszulek kart
`var tablecolor = "rgb(255,255,255)";`	Kolor stołu
`var cardrad = 30;`	Promień wielokątów
`var deck = [];`	Deklaracja talii, początkowo pusta tablica
`var firstsx = 30;`	Współrzędna `x` pierwszej karty
`var firstsy = 50;`	Współrzędna `y` pierwszej karty
`var margin = 30;`	Odstęp między kartami
`var cardwidth = 4 * cardrad;`	Szerokość karty ustalona na czterokrotność promienia wielokąta
`var cardheight = 4 * cardrad;`	Wysokość karty ustalona na czterokrotność promienia wielokąta
`var matched;`	Ta zmienna jest ustawiana w funkcji `choose()` i wykorzystywana przez funkcję `flipback()`
`var starttime;`	Ta zmienna jest ustawiana w funkcji `init()` i służy do obliczenia czasu gry
`function Card(sx, sy, swidth, sheight, info) {`	Początek funkcji kreatora obiektu klasy `Card`
`this.sx = sx;`	Współrzędna pozioma
`this.sy = sy;`	… współrzędna pionowa
`this.swidth = swidth;`	… szerokość
`this.sheight = sheight;`	…. wysokość

Kup książkę Poleć książkę

`this.info = info;`	… `info` (liczba boków wielokąta)
`this.draw = drawback;`	Metoda rysująca kartę
`}`	Koniec funkcji
`function makedeck() {`	Początek funkcji `makedeck()` inicjalizującej talię
`var i;`	Zmienna wykorzystywana w pętli `for`
`var acard;`	Zmienna przechowująca pierwszą kartę z pary
`var bcard;`	Zmienna przechowująca drugą kartę z pary
`var cx = firstsx;`	Zmienna przechowująca współrzędną `x`, inicjalizowana na współrzędną `x` pierwszej karty
`var cy = firstsy;`	Zmienna przechowująca współrzędną `y`, inicjalizowana na współrzędną `y` pierwszej karty
`for(i = 3; i < 9; i++) {`	Pętla rysująca wielokąty: od trójkątów po ośmiokąty
`acard = new Card(cx, cy, cardwidth, cardheight, i);`	Nowy obiekt karty
`deck.push(acard);`	Dodanie karty do talii
`bcard = new Card(cx, cy + cardheight + margin, cardwidth, cardheight, i);`	Druga karta z takimi samymi parametrami, ale wyświetlana poniżej pierwszej
`deck.push(bcard);`	Dodanie karty do talii
`cx = cx + cardwidth + margin;`	Odstęp między kartami z marginesem
`acard.draw();`	Rysowanie karty na elemencie `canvas`
`bcard.draw();`	Rysowanie karty na elemencie `canvas`
`}`	Koniec pętli `for`
`shuffle();`	Mieszanie kart
`}`	Koniec funkcji
`function shuffle() {`	Początek funkcji `shuffle()`
`var i;`	Zmienna przechowująca referencję do karty
`var k;`	Zmienna przechowująca referencję do karty
`var holder;`	Zmienna dodatkowa, niezbędna do dokonania zamiany
`var dl = deck.length`	Zmienna przechowująca liczbę kart w talii
`var nt;`	Indeks karty do zamiany
`for (nt = 0; nt < 3 * dl; nt++) {`	Pętla `for`
`i = Math.floor(Math.random() * dl);`	Pobranie losowej karty
`k = Math.floor(Math.random() * dl);`	Pobranie losowej karty
`holder = deck[i].info;`	Zapisanie informacji karty `i`
`deck[i].info = deck[k].info;`	Umieszczenie informacji karty `i` w miejscu karty `k`
`deck[k].info = holder;`	Umieszczenie poprzedniej informacji karty `k` w miejscu karty `i`
`}`	Koniec pętli `for`
`}`	Koniec funkcji

Kup książkę Poleć książkę

`function Polycard(sx, sy, rad, n) {`	Początek funkcji konstruktora klasy `Polycard`
`this.sx = sx;`	Ustawienie współrzędnej `x`
`this.sy = sy;`	… współrzędnej `y`
`this.rad = rad;`	… promienia wielokąta
`this.draw = drawpoly;`	… metody rysującej
`this.n = n;`	… liczby boków
`this.angle = (2 * Math.PI) / n`	Obliczenie i przypisanie kąta
`}`	Koniec funkcji
`function drawpoly() {`	Początek funkcji `drawpoly()`
`ctx.fillStyle = frontbgcolor;`	Wypełnienie kolorem frontu karty
`ctx.fillRect(this.sx - 2 * this.rad, this.sy - 2 * this.rad, 4 * this.rad, 4 * this.rad);`	Wierzchołek prostokąta znajduje się u góry i po lewej środka wielokąta
`ctx.beginPath();`	Początek ścieżki
`ctx.fillStyle = polycolor;`	Wypełnienie wielokąta
`var i;`	Zmienna używana do indeksowania
`var rad = this.rad;`	Promień
`ctx.moveTo(this.sx + rad * Math.cos(-.5 * this.angle), this.sy + rad * Math.sin(-.5 * this.angle));`	Przeniesienie rysowania do pierwszego wierzchołka
`for (i=1; i < this.n; i++) {`	Pętla `for` rysująca kolejne odcinki
`ctx.lineTo(this.sx + rad * Math.cos((i - .5) * this.angle), this.sy + rad * Math.sin((i - .5) * this.angle));`	Dodanie odcinka wielokąta do ścieżki
`}`	Koniec pętli `for`
`ctx.fill();`	Wypełnienie ścieżki
`}`	Koniec funkcji
`function drawback() {`	Początek funkcji
`ctx.fillStyle = backcolor;`	Ustawienie koloru koszulki karty
`ctx.fillRect(this.sx, this.sy, this.swidth, this.sheight);`	Rysowanie prostokąta
`}`	Koniec funkcji
`function choose(ev) {`	Początek funkcji `choose()` (obsługa kliknięcia karty)
`var mx;`	Zmienna przechowująca współrzędną `x` myszy
`var my;`	Zmienna przechowująca współrzędną `y` myszy
`var pick1;`	Zmienna przechowująca obiekt klasy `Polygon`
`var pick2;`	Zmienna przechowująca obiekt klasy `Polygon`
`if (ev.layerX || ev.layerX == 0) {`	Czy przeglądarka obsługuje właściwości `layerX` i `layerY`?

Kup książkę Poleć książkę

`mx= ev.layerX;`	Ustawienie `mx`
`my = ev.layerY;`	Ustawienie `my`
`}`	Koniec warunku `if true`
`else if (ev.offsetX || ev.offsetX == 0) {`	Czy przeglądarka obsługuje właściwości `offsetX` i `offsetY`?
`mx = ev.offsetX;`	Ustawienie `mx`
`my = ev.offsetY;`	Ustawienie `my`
`}`	Koniec klauzuli `else`
`var i;`	Zmienna wykorzystywana w pętli
`for (i = 0; i < deck.length; i++){`	Pętla przeglądająca wszystkie karty w talii
`var card = deck[i];`	Referencja do obiektu karty w celu uproszczenia kodu
`if (card.sx >= 0)`	Sprawdzenie, czy karta nie jest oznaczona jako usunięta
`if ((mx > card.sx) && (mx < card.sx + card.swidth) && (my > card.sy) && (my < card.sy + card.sheight)) {`	Sprawdzenie, czy kliknięcie nastąpiło w obrębie rysunku karty
`if ((firstpick) || (i != firstcard)) break;`	Jeśli tak, sprawdzamy, czy to nie jest drugie kliknięcie w tę samą kartę; w takim przypadku kończymy przetwarzanie pętli
`}`	Koniec warunku `if true`
`}`	Koniec pętli `for`
`if (i < deck.length) {`	Czy nastąpiło wcześniejsze wyjście z pętli `for`?
`if (firstpick) {`	Jeśli to była pierwsza kliknięta karta…
`firstcard = i;`	… ustawiamy na nią referencję `firstcard`
`firstpick = false;`	Ustawienie `firstpick` na wartość `false`
`pick1 = new Polycard(card.sx + cardwidth * .5,card.sy + cardheight * .5, cardrad, card.info);`	Utworzenie obiektu wielokąta
`pick1.draw();`	Narysowanie wielokąta
`}`	Koniec warunku
`else {`	W przeciwnym wypadku…
`secondcard = i;`	… ustawienie referencji `secondcard` na drugą kliknięta kartę
`pick2 = new Polycard(card.sx + cardwidth * .5, card.sy + cardheight * .5, cardrad, card.info);`	Utworzenie obiektu wielokąta
`pick2.draw();`	Narysowanie wielokąta
`if (deck[i].info == deck[firstcard].info) {`	Sprawdzenie pary
`matched = true;`	Ustawienie zmiennej `matched` na wartość `true`
`var nm = 1 + Number(document.f.count.value);`	Inkrementacja liczby par

Kup książkę Poleć książkę

`document.f.count.value = String(nm);`	Wyświetlenie nowego wyniku
`if (nm >= .5 * deck.length) {`	Sprawdzenie, czy gra jest już zakończona
`var now = new Date();`	Odczyt obiektu daty i czasu
`var nt = Number(now.getTime());`	Wydobycie czasu i przekształcenie na liczbę
`var seconds = Math.floor(.5 + (nt - starttime) / 1000);`	Obliczenie sekund
`document.f.elapsed.value = String(seconds);`	Wyświetlenie czasu
`}`	Koniec warunku (czy koniec gry)
`}`	Koniec warunku (czy jest dopasowanie)
`else {`	w przeciwnym razie…
`matched = false;`	Ustawienie zmiennej `matched` na wartość `false`
`}`	Koniec instrukcji `else`
`firstpick = true;`	Przywrócenie domyślnego stanu zmiennej `firstpick`
`setTimeout(flipback, 1000);`	Ustawienie przerwy
`}`	Koniec warunku (jeśli to nie jest pierwsza karta)
`}`	Koniec warunku (kliknięcie w kartę, pętla `for` przerwana)
`}`	Koniec funkcji
`function flipback() {`	Początek funkcji `flipback()` obsługującej przerwę
`if (!matched) {`	Jeśli nie było dopasowania…
`deck[firstcard].draw();`	… rysowanie koszulki karty
`deck[secondcard].draw();`	… rysowanie koszulki karty
`}`	Koniec warunku
`else {`	Jeśli było dopasowanie (usuwanie kart)
`ctx.fillStyle = tablecolor;`	Ustawienie koloru wypełnienia na kolor stołu
`ctx.fillRect(deck[secondcard].sx, deck[secondcard].sy, deck[secondcard].swidth, deck[secondcard].sheight);`	Rysowanie w miejscu karty
`ctx.fillRect(deck[firstcard].sx, deck[firstcard].sy, deck[firstcard].swidth, deck[firstcard].sheight);`	Rysowanie w miejscu karty
`deck[secondcard].sx = -1;`	Ustawienie wartości powodującej pominięcie karty
`deck[firstcard].sx = -1;`	Ustawienie wartości powodującej pominięcie karty
`}`	Koniec warunku (brak dopasowania)
`}`	Koniec funkcji
`function init(){`	Początek funkcji `init()`

Kup książkę Poleć książkę

`ctx = document.getElementById ('canvas').getContext('2d');`	Zmienna `ctx` realizująca rysowanie
`canvas1 = document.getElementById ('canvas');`	Zmienna `canvas1` do obsługi zdarzeń
`canvas1.addEventListener('click', choose, false);`	Ustawienie funkcji obsługi zdarzeń
`makedeck();`	Utworzenie talii
`document.f.count.value = "0";`	Inicjalizacja licznika par
`document.f.elapsed.value = "";`	Usunięcie starej wartości czasu
`starttime = new Date();`	Odczyt czasu rozpoczęcia gry
`starttime = Number(starttime. getTime());`	Ta sama zmienna jest użyta do zapisania właściwej wartości czasu w sekundach
`shuffle();`	Mieszanie kart
`}`	Koniec funkcji
`</script>`	Koniec elementu `script`
`</head>`	Koniec elementu `head`
`<body onLoad="init();">`	Element `body`, konfiguracja uruchomienia funkcji `init()`
`<canvas id="canvas" width="900" height="400">`	Początek elementu `canvas`
`Twoja przeglądarka nie obsługuje elementu canvas standardu HTML5.`	Komunikat dla użytkowników niekompatybilnych przeglądarek
`</canvas>`	Koniec elementu `canvas`
` `	Przejście do nowego wiersza
`Klikaj karty próbując znaleźć pary.`	Instrukcje gry
`<form name="f">`	Początek elementu formularza
`Liczba par: <input type="text" name="count" value="0" size="1"/>`	Etykieta i pole tekstowe używane do wyświetlania informacji
`<p>`	Nowy akapit
`Czas gry: <input type="text" name="elapsed" value=" " size="4"/> sekundy.`	Etykieta i pole tekstowe używane do wyświetlania informacji
`</form>`	Koniec elementu `form`
`</body>`	Koniec elementu `body`
`</html>`	Koniec elementu `html`

Niezależnie od podjętych przez Ciebie decyzji programistycznych jeszcze raz zachęcam do komentowania kodu (używając dwóch ukośników // na początku) i robienia odstępów (pustych wierszy) między niezależnymi fragmentami kodu. Oczywiście, nie ma konieczności komentowania każdego wiersza, ale rozsądna liczba komentarzy z pewnością pomoże w przyszłości, gdy będziesz musiał wrócić do tego kodu w celu wprowadzenia poprawek lub udoskonaleń.

Kup książkę Poleć książkę

Grę można zmodyfikować, zmieniając rozmiar i kolor czcionki formularza lub kolor tła pola formularza. Więcej pomysłów na zmodyfikowanie aplikacji znajdziesz w dalszej części rozdziału.

Druga wersja gry „Pamięć", wykorzystująca zdjęcia, posiada bardzo podobną strukturę do wersji wykorzystującej wielokąty. Wersja druga nie wymaga osobnej funkcji do rysowania frontu karty. Tabela 5.3 zawiera listę funkcji wykorzystywanych przez drugą wersję gry.

Tabela 5.3. Funkcje w drugiej wersji gry „Pamięć", wykorzystującej fotografie

Funkcja	Wywoływana przez	Wywołuje
`init()`	obsługę zdarzenia `onLoad` elementu `body`	`makedeck()` `shuffle()`
`choose()`	obsługę zdarzenia myszy zdefiniowaną przez metodę `addEventListener()` w funkcji `init()`	
`flipback()`	obsługę zdarzenia czasowego zdefiniowanego przez funkcję `setTimeout()` wywołaną w funkcji `choose()`	
`drawback()`	`makedeck()` i `flipback()` jako metodę `draw()` klasy `Polycard`	
`shuffle()`	`init()`	
`makedeck()`	`init()`	
`Card()`	`makedeck()`	

Kod drugiej wersji aplikacji jest bardzo podobny do pierwszej. Większość logiki pozostaje bez zmian. Ta wersja do wyświetlania komunikatów do użytkownika korzysta z elementu `canvas`, dlatego dokument HTML nie zawiera formularza. Kod aplikacji jest przedstawiony w tabeli 5.4, z komentarzami tylko w miejscach różnic. W komentarzu sygnalizuję miejsce, w którym można zmodyfikować nazwy plików fotografii, jeśli zechcesz użyć własnych. Zanim zagłębisz się w analizę kodu drugiej wersji gry, zastanów się chwilę, które jej fragmenty będą się różnić od pierwszej, a które będą takie same.

Tabela 5.4. Kompletny kod drugiej wersji gry „Pamięć", wykorzystującej fotografie

`<html>`	
`<head>`	
`<title>Pamięć</title>`	Kompletny element `title`
`<script type="text/javascript">`	
`var ctx;`	
`var firstpick = true;`	
`var firstcard = -1;`	
`var secondcard;`	
`var backcolor = "rgb(128,0,128)";`	
`var tablecolor = "rgb(255,255,255)";`	
`var deck = [];`	
`var firstsx = 30;`	

Kup książkę Poleć książkę

`var firstsy = 50;`	
`var margin = 30;`	
`var cardwidth = 100;`	Jeśli zdjęcia mają inne wymiary, w tym miejscu ustaw szerokość…
`var cardheight = 100;`	… a w tym wysokość
`var matched;`	
`var starttime;`	
`var count = 0;`	Licznik
`var pairs = [`	Tablica par zdjęć pięciu osób
`["allison1.jpg", "allison2.jpg"],`	W tej tablicy zdefiniuj nazwy plików ze zdjęciami
`[ "grant1.jpg", "grant2.jpg"],`	…
`["liam1.jpg", "liam2.jpg"],`	…
`["aviva1.jpg", "aviva2.jpg"],`	…
`["daniel1.jpg", "daniel2.jpg"]`	Możesz użyć dowolnej liczby par, ale ostatnia para nie może mieć przecinka po nawiasie kwadratowym
`]`	
`function Card(sx, sy, swidth, sheight, img, info) {`	
`this.sx = sx;`	
`this.sy = sy;`	
`this.swidth = swidth;`	
`this.sheight = sheight;`	
`this.info = info;`	Definicja par
`this.img = img;`	Referencja do obrazu
`this.draw = drawback;`	
`}`	
`function makedeck() {`	
`var i;`	
`var acard;`	
`var bcard;`	
`var pica;`	
`var picb;`	
`var cx = firstsx;`	
`var cy = firstsy;`	
`for(i = 0; i < pairs.length; i++) {`	
`pica = new Image();`	Utworzenie obiektu klasy `Image`
`pica.src = pairs[i][0];`	Ustawienie pierwszego pliku

Kup książkę Poleć książkę

`acard = new Card(cx, cy, cardwidth, cardheight, pica, i);`	Utworzenie karty (obiektu klasy `Card`)
`deck.push(acard);`	
`picb = new Image();`	Utworzenie obiektu obrazu (`Image`)
`picb.src = pairs[i][1];`	Ustawienie drugiego pliku
`bcard = new Card(cx, cy + cardheight + margin, cardwidth, cardheight, picb, i);`	Utworzenie karty (obiektu klasy `Card`)
`deck.push(bcard);`	
`cx = cx + cardwidth + margin;`	
`acard.draw();`	
`bcard.draw();`	
`}`	
`}`	
`function shuffle() {`	
`var i;`	
`var k;`	
`var holderinfo;`	Tymczasowa zmienna używana do zamiany
`var holderimg;`	Tymczasowa zmienna używana do zamiany
`var dl = deck.length`	
`var nt;`	
`for (nt = 0; nt < 3 * dl; nt++) {`	
`i = Math.floor(Math.random()*dl);`	
`k = Math.floor(Math.random()*dl);`	
`holderinfo = deck[i].info;`	Zapisanie wartości `info`
`holderimg = deck[i].img;`	Zapisanie wartości `img`
`deck[i].info = deck[k].info;`	Wpisanie wartości `info` z `k` do `i`
`deck[i].img = deck[k].img;`	Wpisanie wartości `img` z `k` do `i`
`deck[k].info = holderinfo;`	Wpisanie oryginalnej wartości `info`
`deck[k].img = holderimg;`	Wpisanie oryginalnej wartości `img`
`}`	
`}`	
`function drawback() {`	
`ctx.fillStyle = backcolor;`	
`ctx.fillRect(this.sx, this.sy, this.swidth, this.sheight);`	
`}`	
`function choose(ev) {`	

Kup książkę Poleć książkę

`var out;`	
`var mx;`	
`var my;`	
`var pick1;`	
`var pick2;`	
`if (ev.layerX || ev.layerX == 0) {`	To jest kod obsługujący różnice między przeglądarkami
`mx= ev.layerX;`	
`my = ev.layerY;`	
`} else if (ev.offsetX || ev.offsetX == 0) {`	
`mx = ev.offsetX;`	
`my = ev.offsetY;`	
`}`	
`var i;`	
`for (i = 0; i < deck.length; i++){`	
`var card = deck[i];`	
`if (card.sx >= 0)`	
`if ((mx > card.sx) && (mx < card.sx + card.swidth) && (my > card.sy) && (my < card.sy + card.sheight)) {`	
`if ((firstpick) || (i != firstcard)) {`	
`break;}`	
`}`	
`}`	
`if (i < deck.length) {`	
`if (firstpick) {`	
`firstcard = i;`	
`firstpick = false;`	
`ctx.drawImage(card.img, card.sx, card.sy, card.swidth, card.sheight);`	Rysowanie zdjęcia
`}`	
`else {`	
`secondcard = i;`	
`ctx.drawImage(card.img, card.sx, card.sy, card.swidth, card.sheight);`	Rysowanie zdjęcia
`if (card.info==deck[firstcard].info) {`	Czy jest dopasowanie?
`matched = true;`	
`count++;`	Zwiększenie licznika
`ctx.fillStyle = tablecolor;`	

Kup książkę Poleć książkę

`ctx.fillRect(10, 340, 900, 100);`	Wymazanie w miejscu tekstu
`ctx.fillStyle = backcolor;`	Ustawienie koloru tekstu
`ctx.fillText("Liczba par: " + String(count), 10, 360);`	Wypisanie licznika
`if (count >= .5 * deck.length) {`	
`var now = new Date();`	
`var nt = Number(now.getTime());`	
`var seconds = Math.floor(.5 + (nt - starttime) / 1000);`	
`ctx.fillStyle = tablecolor;`	
`ctx.fillRect(0, 0, 900, 400);`	Wymazanie całej zawartości `canvas`
`ctx.fillStyle = backcolor;`	Kolor wypełnienia
`out="Czas gry: " + String(seconds) + " sekund.";`	Przygotowanie tekstu
`ctx.fillText(out, 10, 100);`	Wypisanie tekstu
`ctx.fillText("Odśwież stronę.", 10, 300);`	Wypisanie tekstu
`}`	
`}`	
`else {`	
`matched = false;`	
`}`	
`firstpick = true;`	
`setTimeout(flipback, 1000);`	
`}`	
`}`	
`}`	
`function flipback() {`	
`var card;`	
`if (!matched) {`	
`deck[firstcard].draw();`	
`deck[secondcard].draw();`	
`}`	
`else {`	
`ctx.fillStyle = tablecolor;`	
`ctx.fillRect(deck[secondcard].sx,deck[secondcard].sy, deck[secondcard].swidth, deck[secondcard].sheight);`	
`ctx.fillRect(deck[firstcard].sx, deck[firstcard].sy, deck[firstcard].swidth, deck[firstcard].sheight);`	

Kup książkę Poleć książkę

`deck[secondcard].sx = -1;`	
`deck[firstcard].sx = -1;`	
`}`	
`}`	
`function init(){`	
`ctx = document.getElementById('canvas'). getContext('2d');`	
`canvas1 = document.getElementById('canvas');`	
`canvas1.addEventListener('click', choose, false);`	
`makedeck();`	
`shuffle();`	
`ctx.font="bold 20pt sans-serif";`	Ustawienie kroju czcionki
`ctx.fillText("Klikaj karty, próbując znaleźć pary.", 10, 20);`	Wypisanie instrukcji na elemencie `canvas`
`ctx.fillText("Liczba kart: 0",10,360);`	Wypisanie licznika
`starttime = new Date();`	
`starttime = Number(starttime.getTime());`	
`}`	
`</script>`	
`</head>`	
`<body onLoad="init();">`	
`<canvas id="canvas" width="900" height="400">`	
`Twoja przeglądarka nie obsługuje elementu canvas standardu HTML5.`	
`</canvas>`	
`</body>`	
`</html>`	

Obydwa przedstawione programy są kompletnymi grami, ale to nie znaczy, że nie da się ich udoskonalić. Na przykład gracz nie może przegrać. Warto zatem zastanowić się nad udostępnieniem takiej możliwości, na przykład przez ograniczenie liczby prób lub ustalenie limitu czasu.

Obie aplikacje uruchamiają zegar od razu po załadowaniu. Niektóre gry czekają z rozpoczęciem odmierzania czasu do wykonania pierwszej czynności przez gracza. Jeśli chcesz zaimplementować tego typu podejście, zdefiniuj zmienną ustawioną pierwotnie na wartość `false`. Funkcja `choose()` po prostu sprawdzi jej wartość, a jeśli to `false`, ustawi ją na `true` oraz ustawi wartość zmiennej `starttime` na bieżący czas.

To jest gra dla jednego gracza. Można wymyślić sposób przystosowania jej dla dwóch graczy. Należałoby założyć, że osoby grają na zmianę, ale program śledzi ich wyniki w celu porównania.

Gracze lubią gry pozwalające na dobranie poziomu trudności. W tym celu można dostosować liczbę kart, skrócić czas wyświetlania odkrytych kart lub zastosować inne techniki.

Kup książkę Poleć książkę

Aplikacje możesz przystosować do własnych potrzeb, wykorzystując prywatne zdjęcia. Oczywiście, można użyć zdjęć przyjaciół i rodziny, ale można też stworzyć grę z elementami edukacyjnymi, wykorzystując logiczne pary ilustracji, jak symbole nut i ich nazwy, nazwy państw i ich stolic, mapy państw i ich nazwy itp. Można też zmienić liczbę par. Kod w swojej logice wykorzystuje długości tablic, nie ma więc potrzeby zmieniania wartości różnych zmiennych w kodzie w celu zmiany liczby kart w talii. Być może warto zmodyfikować szerokość i wysokość kart, aby zmieściły się na ekranie.

Inną możliwością jest użycie standardowej talii 52 kart (54 z jokerami). Przykład użycia prawdziwych kart znajdziesz w rozdziale 10., przy okazji gry w 21. W przypadku każdej gry w dopasowania warto zadbać o to, aby gracz miał jasną informację, czym dokładnie jest dopasowanie.

Testowanie aplikacji i wrzucenie jej na serwer

Gdy programiści testują swoje programy, z reguły robią rzeczy rozsądne. Użytkownicy, gracze i klienci mają jednak skłonność do robienia rzeczy nielogicznych. Dlatego zawsze warto poprosić osobę postronną o przetestowanie aplikacji. Poproś przyjaciół, żeby przetestowali Twoją grę. Zawsze warto zadbać o testerów, którzy nie brali udziału w pisaniu aplikacji.

Dokument HTML w wersji wykorzystującej wielokąty zawiera kompletną grę, ponieważ wszelkie grafiki są rysowane bezpośrednio w kodzie. Wersja gry wykorzystująca fotografie wymaga przesłania na serwer również wszystkich plików. Grę można urozmaicić przez użycie obrazów z zewnętrznych stron WWW (spoza Twojego serwera WWW). W takim przypadku w tablicy `pairs` należy podać pełne adresy URL obrazów.

Podsumowanie

W tym rozdziale nauczyłeś się implementować dwie wersje znanej gry „Pamięć" (znanej również jako „Koncentracja"), wykorzystując do tego celu funkcje standardów HTML5 i JavaScriptu, między innymi:

- funkcje i obiekty definiowane przez programistę;
- rysowanie wielokątów na elemencie `canvas` z użyciem metod `moveTo()` i `lineTo()` w połączeniu z funkcjami trygonometrycznymi zdefiniowanymi w module `Math`;
- wykorzystanie formularza do wyświetlania komunikatów dla użytkownika;
- rysowanie tekstu na elemencie `canvas` z użyciem wybranego kroju czcionki;
- rysowanie obrazów na elemencie `canvas`;
- wykorzystanie funkcji `setTimeout()` do wstrzymania działania programu;
- wykorzystanie obiektów `Date` do odmierzenia upływu czasu.

Przy okazji implementacji znanej gry miałeś okazję poznać sposoby reprezentowania informacji. Następny rozdział na chwilę odejdzie od elementu `canvas`. Poznamy sposoby dynamicznego generowania i pozycjonowania elementów HTML. Nauczymy się też wykorzystywać element `video`, wprowadzony w HTML5.

Kup książkę Poleć książkę

Skorowidz

Kup książkę Poleć książkę

F

G

H

I

J

K

Kup książkę Poleć książkę

L, Ł

M

N

O

Kup książkę Poleć książkę

Kup książkę Poleć książkę

W

Z

Ź

Kup książkę Poleć książkę

PROGRAM PARTNERSKI

GRUPY WYDAWNICZEJ HELION

1. ZAREJESTRUJ SIĘ
2. PREZENTUJ KSIĄŻKI
3. ZBIERAJ PROWIZJĘ

Zmień swoją stronę WWW
w działający bankomat!

Dowiedz się więcej i dołącz już dzisiaj!
http://program-partnerski.helion.pl

GRUPA WYDAWNICZA
Helion SA

Wprowadzenie do HTML5. Nauka HTML5 i JavaScriptu na przykładzie gier

Nauka jeszcze nigdy nie była tak wciągająca!

Standard HTML5 umożliwia tworzenie nie tylko dynamicznych, multimedialnych stron WWW, ale także zaawansowanych aplikacji internetowych i gier. Do niedawna, żeby to osiągnąć, konieczna była instalacja dodatkowych rozszerzeń. Teraz można to zrealizować, korzystając wyłącznie z HTML-a, JavaScriptu i kaskadowych arkuszy stylów CSS… Jeśli chcesz szybko i przyjemnie poznać podstawowe założenia HTML5 oraz natychmiast wykorzystać je w praktyce, sięgnij po tę książkę. W charakterze materiału do analizy użyto w niej popularnych gier, takich wisielec, kości czy blackjack.

Wprowadzenie do HTML5 pokaże Ci nowe, ekscytujące możliwości standardu HTML5 oraz sposoby integrowania HTML-a z JavaScriptem i CSS. Dzięki podanym przykładom zrozumiesz, jak te elementy wzajemnie się uzupełniają. Każdy rozdział jest poświęcony innej aplikacji i ma podobną strukturę. Na początku znajdują się wymagania, a następnie funkcje HTML5, CSS i JavaScriptu niezbędne do konkretnej implementacji. W dalszej części szczegółowo omawiana jest sama implementacja. Jeśli masz nieco większe doświadczenie w tworzeniu stron WWW, ta książka przybliży Ci sposoby wykorzystania JavaScriptu w połączeniu z dokumentami HTML oraz zapozna Cię z nowościami w HTML5, takimi jak obsługa zdarzeń, walidacja formularzy, wykorzystanie lokalnego magazynu danych czy elementy canvas, wideo i audio. Przykłady są wystarczająco proste, aby ich zrozumienie nie przysparzało problemów, ale na tyle inspirujące, by dać możliwość stworzenia ciekawych interaktywnych aplikacji WWW.

Połącz przyjemne z pożytecznym:

- Poznaj JavaScript, CSS i HTML5
- Naucz się rysować w przeglądarce internetowej
- Korzystaj ze zdarzeń
- Przechowuj dane w przeglądarce użytkownika
- Bądź na bieżąco z najnowszymi trendami

friendsof

Nr katalogowy: 8698

Księgarnia internetowa: http://helion.pl

Zamówienia telefoniczne: 0 801 339900, 0 601 339900

helion.pl księgarnia internetowa

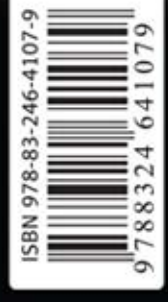
Sprawdź najnowsze promocje: http://helion.pl/promocje
Książki najchętniej czytane: http://helion.pl/bestsellery
Zamów informacje o nowościach: http://helion.pl/nowosci

Helion

Helion SA
ul. Kościuszki 1c, 44-100 Gliwice
tel.: 32 230 98 63
e-mail: helion@helion.pl
http://helion.pl

Cena: 49,00 zł

ISBN 978-83-246-4107-9



Informatyka w najlepszym wydaniu