

Mike_Geig

Unity

PRZEWODNIK PROJEKTANTA GIER



do własnej gry!

Helion

Tytuł oryginału: Sams Teach Yourself Unity® Game Development in 24 Hours

Tłumaczenie: Robert Górczyński

Projekt okładki: Studio Gravite / Olsztyn; Obarek, Pokoński, Pazdrijowski, Zaprucki
Materiały graficzne na okładce zostały wykorzystane za zgodą Shutterstock Images LLC.

ISBN: 978-83-283-0017-0

Authorized translation from the English language edition: SAMS TEACH YOURSELF UNITY GAME DEVELOPMENT IN 24 HOURS, ISBN 0672336960; by Mike Geig; published by Pearson Education, Inc, publishing as SAMS Publishing.
Copyright © 2014 by Pearson Education, Inc.

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 Pearson Education Inc.
Polish language edition published by HELION S.A. Copyright © 2015.

Unity is a trademark of Unity technologies.

Kinect is a trademark of Microsoft®.

PlayStation and PlayStation Move are trademarks of Sony®.

Wii is a trademark of Nintendo®.

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiegokolwiek 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.

Autor oraz Wydawnictwo HELION dołożyli wszelkich starań, by zawarte w tej książce informacje były kompletne i rzetelne. Nie biorą jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Autor oraz Wydawnictwo HELION nie ponoszą 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)

Drogi Czytelniku!

Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres

<http://helion.pl/user/opinie/unippg>

Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

- [Kup książkę](#)
- [Poleć książkę](#)
- [Oceń książkę](#)

- [Księgarnia internetowa](#)
- [Lubię to! » Nasza społeczność](#)

Spis treści

O autorze	19
Wprowadzenie	21
LEKCJA 1. Wprowadzenie do Unity	25
Instalacja Unity	26
Pobieranie i instalacja Unity	26
Poznajemy edytor Unity	29
Okno dialogowe Project	29
Interfejs Unity	31
Panel Project	32
Panel Hierarchy	34
Panel Inspector	35
Panel Scene	37
Panel Game	39
Wyróżnienie — pasek narzędziowy	41
Poruszanie się po panelu Scene w Unity	42
Narzędzie Hand	42
Tryb Flythrough	43
Podsumowanie	43
Pytania i odpowiedzi	45
Warsztaty	45
Quiz	45
Odpowiedzi	45
Ćwiczenie	46
LEKCJA 2. Obiekty gry	47
Wymiary i układy współrzędnych	48
Umieszczenie litery D w nazwie 3D	48
Użycie układów współrzędnych	48
Współrzędne świata kontra lokalne	50
Obiekty gry	50
Transformacje	51
Translacja	52
Rotacja	54
Skalowanie	55

Ryzyko związane z transformacjami	55
Transformacja a obiekty zagnieżdżone	57
Podsumowanie	57
Pytania i odpowiedzi	58
Warsztaty	58
Quiz	58
Odpowiedzi	58
Ćwiczenie	59
LEKCJA 3. Modele, materiały i tekstury	61
Podstawy modeli	62
Wbudowane obiekty 3D	63
Importowanie modeli	63
Modele i sklep Asset Store	65
Tekstury, shadery i materiały	66
Tekstury	67
Shadery	68
Materiały	68
Powracamy do shaderów	69
Podsumowanie	72
Pytania i odpowiedzi	72
Warsztaty	73
Quiz	73
Odpowiedzi	73
Ćwiczenie	73
LEKCJA 4. Teren	75
Generowanie terenu	76
Dodanie terenu do projektu	76
Rzeźbienie mapy wysokości	77
Narzędzia do rzeźbienia terenu oferowane przez Unity	80
Tekstury terenu	83
Importowanie zasobów terenu	83
Teksturowanie terenu	83
Podsumowanie	85
Pytania i odpowiedzi	86
Warsztaty	86
Quiz	87
Odpowiedzi	87
Ćwiczenie	87

LEKCJA 5.	Środowisko	89
	Generowanie drzew i trawy	90
	Malowanie drzewami	90
	Malowanie trawą	92
	Ustawienia terenu	94
	Efekty środowiska	95
	Symulacja nieba i horyzontu	96
	Mgła	98
	Efekt Lens Flare	99
	Woda	99
	Kontroler postaci	101
	Dodanie kontrolera postaci	101
	Naprawa świata	101
	Podsumowanie	103
	Pytania i odpowiedzi	103
	Warsztaty	103
	Quiz	103
	Odpowiedzi	104
	Ćwiczenie	104
LEKCJA 6.	Światła i kamery	105
	Światło	106
	Światło punktowe	106
	Światło kierunkowe	109
	Tworzenie światła z dowolnego obiektu	110
	Aureola	111
	Cookies	112
	Kamera	114
	Anatomia kamery	114
	Wiele kamer	116
	Podział ekranu oraz funkcja PiP	116
	Warstwy	117
	Praca z warstwami	118
	Użycie warstw	120
	Podsumowanie	121
	Pytania i odpowiedzi	123
	Warsztaty	123
	Quiz	123
	Odpowiedzi	123
	Ćwiczenie	123

LEKCJA 7.	Pierwsza gra — Amazing Racer	125
	Faza projektowania	126
	Koncepcja	126
	Reguły	126
	Wymagania	127
	Budowanie świata gry	128
	Rzeźbienie terenu	129
	Dodanie elementów do środowiska	129
	Kontroler postaci	130
	Gamifikacja	131
	Dodanie obiektów kontrolujących grę	131
	Dodanie skryptów	133
	Połączenie skryptów	134
	Testowanie gry	136
	Podsumowanie	138
	Pytania i odpowiedzi	138
	Warsztaty	138
	Quiz	139
	Odpowiedzi	139
	Ćwiczenie	139
 LEKCJA 8.	 Skrypty — część 1.	 141
	Skrypty	142
	Tworzenie skryptów	142
	Dołączanie skryptu	142
	Anatomia prostego skryptu	145
	Zmienne	147
	Tworzenie zmiennej	148
	Zasięg zmiennej	148
	Zmienne publiczne i prywatne	150
	Operatory	151
	Operatory arytmetyczne	151
	Operatory przypisania	152
	Operatory równości	152
	Operatory logiczne	153
	Konstrukcje warunkowe	153
	Konstrukcja if	154
	Konstrukcja if-else	155
	Konstrukcja if-else if	156

Iteracja	157
Pętla while	157
Pętla for	158
Podsumowanie	159
Pytania i odpowiedzi	159
Warsztaty	159
Quiz	160
Odpowiedzi	160
Ćwiczenie	160
LEKCJA 9. Skrypty — część 2.	161
Metody	162
Anatomia metody	162
Tworzenie metody	163
Dane wejściowe	167
Podstawy danych wejściowych	167
Skrypty przeznaczone do obsługi danych wejściowych	168
Dane wejściowe z określonych klawiszy	169
Dane wejściowe myszy	170
Uzyskanie dostępu do komponentów lokalnych	172
Uzyskanie dostępu do innych obiektów	172
Wyszukanie innych obiektów	173
Modyfikacja komponentów obiektu	175
Podsumowanie	176
Pytania i odpowiedzi	176
Warsztaty	177
Quiz	177
Odpowiedzi	177
Ćwiczenie	178
LEKCJA 10. Kolizje	179
Bryły sztywne	180
Kolizje	182
Komponent Collider	182
Materiały fizyczne	184
Wyzwalacze	185
Raycasting	187
Podsumowanie	189
Pytania i odpowiedzi	189

Warsztaty	190
Quiz	191
Odpowiedzi	191
Ćwiczenie	191
LEKCJA 11. Druga gra — Chaos Ball	193
Faza projektowania	194
Koncepcja	194
Reguły	194
Wymagania	194
Arena	195
Utworzenie areny	195
Teksturowanie	196
Materiał zapewniający rewelacyjne odbijanie się obiektu	197
Zakończenie prac nad areną	199
Elementy gry	199
Gracz	199
Kula chaos ball	200
Kolorowe kule	202
Obiekty kontrolne	203
Cele	203
Kontroler gry	205
Usprawnienie gry	207
Podsumowanie	208
Pytania i odpowiedzi	208
Warsztaty	208
Quiz	208
Odpowiedzi	209
Ćwiczenie	209
LEKCJA 12. Prefabrykaty	211
Podstawy prefabrykatów	212
Terminologia związana z prefabrykatami	212
Struktura prefabrykatu	213
Praca z prefabrykatami	214
Dodanie do sceny egzemplarza prefabrykatu	216
Dziedziczenie	217
Zerwanie połączenia z zasobem prefabrykatu	219
Tworzenie egzemplarza prefabrykatu w kodzie	220
Podsumowanie	220

Pytania i odpowiedzi	221
Warsztaty	221
Quiz	221
Odpowiedzi	221
Ćwiczenie	222
LEKCJA 13. Graficzny interfejs użytkownika	225
Podstawy GUI	226
Kontrolki GUI	227
Etykieta	228
Pole	229
Przycisk	229
Przycisk powtarzający	230
Pole wyboru	230
Pasek narzędziowy	231
Pole tekstowe	232
Obszar tekstu	233
Suwaki	233
Dostosowanie kontrolki do własnych potrzeb	234
Style GUI	234
Skórki GUI	235
Podsumowanie	239
Pytania i odpowiedzi	239
Warsztaty	240
Quiz	240
Odpowiedzi	240
Ćwiczenie	240
LEKCJA 14. Sterowanie postaciami	243
Kontroler postaci	244
Dodanie kontrolera postaci	244
Właściwości kontrolera postaci	245
Skrypty dla kontrolerów postaci	245
Praca z kontrolerem za pomocą skryptów	246
Zmienna typu CollisionFlags	249
Kolizje	250
Tworzenie kontrolera postaci	250
Konfiguracja podstawowa	251
Obsługa ruchu	252
Grawitacja	253

Skoki	253
Popychanie obiektów	254
Pełny kod skryptu	255
Podsumowanie	256
Pytania i odpowiedzi	256
Warsztaty	256
Quiz	256
Odpowiedzi	257
Ćwiczenie	257
LEKCJA 15. Trzecia gra — Captain Blaster	259
Faza projektowania	260
Koncepcja	260
Reguły	260
Wymagania	260
Świat	261
Kamera	261
Tło	261
Elementy gry	263
Gracz	263
Meteory	264
Pociski	265
Wyzwalacze	265
Obiekty kontrolne	266
Kontroler gry	266
Skrypt meteoru	267
Tworzenie meteorów	268
Skrypt wyzwalacza	269
Skrypt gracza	270
Skrypt pocisku	272
Usprawnienie gry	273
Podsumowanie	274
Pytania i odpowiedzi	274
Warsztaty	275
Quiz	275
Odpowiedzi	275
Ćwiczenie	276

LEKCJA 16. Systemy cząsteczek	277
Systemy cząsteczek	278
Cząsteczki	278
Systemy cząsteczek w Unity	278
Kontrolki systemu cząsteczek	278
Moduły systemu cząsteczek	280
Moduł domyślny	280
Moduł Emission	281
Moduł Shape	281
Moduł Velocity over Lifetime	283
Moduł Limit Velocity over Lifetime	283
Moduł Force over Lifetime	284
Moduł Color over Lifetime	284
Moduł Color by Speed	284
Moduł Size over Lifetime	284
Moduł Size by Speed	285
Moduł Rotation over Lifetime	285
Moduł Rotation by Speed	285
Moduł External Forces	285
Moduł Collision	285
Moduł Sub Emitter	288
Moduł Texture Sheet	288
Moduł Renderer	289
Edytor krzywych	290
Podsumowanie	290
Pytania i odpowiedzi	291
Warsztaty	291
Quiz	292
Odpowiedzi	292
Ćwiczenie	292
 LEKCJA 17. Animacje	 293
Podstawy animacji	294
Przygotowanie modelu	294
Animacja	295
Przygotowanie modelu do animacji	295
Model	297
Zasoby animacji	298

Stosowanie animacji	299
Dodawanie animacji	300
Właściwość Wrap Mode	301
Skrypty i animacje	301
Podsumowanie	303
Pytania i odpowiedzi	303
Warsztaty	304
Quiz	305
Odpowiedzi	305
Ćwiczenie	305
LEKCJA 18. Animator	307
Podstawy zasobu Animator	308
Rigging modeli	308
Czerwony rigging śmierci	309
Przygotowanie animacji	311
Utworzenie animatora	316
Panel Animator	317
Animacja Idle	317
Parametry	318
Stan i tzw. drzewo Blend Tree	319
Przejścia	320
Obsługa animacji za pomocą skryptów	322
Podsumowanie	322
Pytania i odpowiedzi	323
Warsztaty	323
Quiz	324
Odpowiedzi	324
Ćwiczenie	324
LEKCJA 19. Czwarta gra — Gauntlet Runner	325
Faza projektowania	326
Koncepcja	326
Reguły	326
Wymagania	326
Świat	327
Scena	327
Droga	327
Przewijanie drogi	328

Elementy gry	328
Dodatkowa energia	329
Przeszkody	330
Strefa wyzwalacza	330
Obiekt gracza	331
Obiekty kontrolne	335
Skrypt strefy wyzwalacza	335
Skrypt obiektu kontrolnego gry	335
Skrypt obiektu gracza	338
Skrypty obiektów dodatkowej energii i przeszkód	339
Skrypt SpawnScript	340
Zebranie wszystkiego w całość	341
Usprawnienie gry	343
Podsumowanie	343
Pytania i odpowiedzi	343
Warsztaty	343
Quiz	344
Odpowiedzi	344
Ćwiczenie	344
LEKCJA 20. Dźwięk	345
Podstawy dźwięku	346
Składniki dźwięku	346
Dźwięk 2D i 3D	347
Źródła dźwięku	347
Import klipów audio	348
Testowanie dźwięku w panelu Scene	350
Dźwięk 3D	350
Dźwięk 2D	351
Użycie dźwięku za pomocą skryptów	352
Rozpoczęcie i zatrzymanie odtwarzania dźwięku	353
Zmiana klipu audio	354
Podsumowanie	355
Pytania i odpowiedzi	355
Warsztaty	355
Quiz	355
Odpowiedzi	355
Ćwiczenie	356

LEKCJA 21. Programowanie na platformach mobilnych	359
Przygotowanie do programowania na platformach mobilnych	360
Konfiguracja środowiska	360
Aplikacja Unity Remote	362
Przyśpieszeniomierz	363
Programowanie dla przyśpieszeniomierza	364
Użycie przyśpieszeniomierza	365
Dane wejściowe pochodzące z ekranu dotykowego	366
Podsumowanie	366
Pytania i odpowiedzi	369
Warsztaty	369
Quiz	369
Odpowiedzi	369
Ćwiczenie	370
 LEKCJA 22. Kolejne wersje gier	 371
Amazing Racer	372
Poruszanie i rozglądanie się	372
Skoki	374
Chaos Ball	375
Captain Blaster	377
Gauntlet Runner	379
Podsumowanie	380
Pytania i odpowiedzi	381
Warsztaty	381
Quiz	381
Odpowiedzi	381
Ćwiczenie	382
 LEKCJA 23. Dopracowanie i wdrożenie	 383
Zarządzanie scenami	384
Ustalanie kolejności scen	384
Przełączanie scen	385
Zachowywanie danych i obiektów	387
Zachowywanie obiektów	387
Zachowywanie danych	387
Ustawienia Unity Player	390
Ustawienia niezależne od platformy	390
Ustawienia dla poszczególnych platform	390

Kompilacja gry	391
Okno Build Settings	392
Okno Game Settings	393
Podsumowanie	395
Pytania i odpowiedzi	395
Warsztaty	395
Quiz	395
Odpowiedzi	395
Ćwiczenie	396
LEKCJA 24. Zakończenie	397
Osiągnięcia	398
19 lekcji nauki	398
4 pełne gry	399
58 scen	400
Co dalej?	400
Buduj gry	401
Współpracuj z innymi	401
Pisz o tym	401
Dostępne zasoby	402
Podsumowanie	402
Pytania i odpowiedzi	402
Warsztaty	403
Quiz	403
Odpowiedzi	403
Ćwiczenie	403
Skorowidz	405

Lekcja 11

Druga gra — Chaos Ball

W czasie tej lekcji dowiesz się:

- ▶ jak zaprojektować grę *Chaos Ball*,
- ▶ jak utworzyć arenę w grze *Chaos Ball*,
- ▶ jak zbudować elementy gry *Chaos Ball*,
- ▶ jak przygotować obiekty kontrolne w grze *Chaos Ball*,
- ▶ jak jeszcze bardziej usprawnić grę *Chaos Ball*.

Nadeszła odpowiednia pora, aby po raz drugi wykorzystać zdobytą dotąd wiedzę do utworzenia gry. W tej lekcji opracujesz aplikację *Chaos Ball*, która będzie grą zręcznościową. Na początek zajmiesz się podstawowym zaprojektowaniem elementów gry. Następnie przejdziesz do zbudowania areny oraz obiektów gry. Każdy obiekt będzie unikalny i otrzyma specjalne właściwości obsługi kolizji. Kolejnym krokiem będzie dodanie interaktywności, aby gra zapewniała rozrywkę. Na końcu lekcji zajmiesz się przetestowaniem gry oraz wprowadzeniem w niej niezbędnych usprawnień.

Ukończony projekt

Aby ukończyć projekt gry, musisz wykonać kolejne kroki zaprezentowane w lekcji. Jeżeli napotkasz problemy, ukończoną wersję gry znajdziesz w materiałach przeznaczonych dla bieżącej lekcji. Przejrzyj te materiały, jeśli szukasz inspiracji!

Wskazówka
Wskazówka

Faza projektowania

Elementy fazy projektowania zostały omówione w lekcji 7., w której powstała pierwsza gra, zatytułowana *Amazing Racer*. Teraz od razu przystąpisz do ich zdefiniowania.

Koncepcja

Budowana w tej lekcji gra przypomina nieco gry zatytułowane *Pinball* i *Breakout*. Gracz będzie znajdował się na arenie. Każdy z czterech narożników areny jest w innym kolorze, a po arenie poruszają się cztery kule o kolorach odpowiadających zastosowanym w rogach. Na arenie, poza czterema kolorowymi kulami, znajduje się też kilka innych białych kul nazywanych *chaos ball*. Zadaniem białych kul jest tylko przeszkadzanie graczowi i sprawienie, aby gra stała się wymagająca. Białe kule są mniejsze od czterech kolorowych i poruszają się szybciej. Gracz dysponuje płaską paletką, którą będzie próbował umieścić kolorowe kule we właściwych narożnikach.

Reguły

Reguły określają sposób prowadzenia gry, a ponadto mają wpływ na pewne właściwości obiektów. Poniżej wymieniono reguły dla gry *Chaos Ball*.

- ▶ Gracz wygrywa, gdy wszystkie cztery kolorowe kule znajdą się w odpowiednich rogach. Nie ma warunku określającego przegraną.
- ▶ Uderzenie właściwego narożnika powoduje, że kula zostaje zatrzymana.
- ▶ Wszystkie obiekty w grze mają doskonałe właściwości odbijania się (przy zderzeniu nie tracą impetu).
- ▶ Żadna kula nie może opuścić areny.
- ▶ Szybkość kul zarówno kolorowych, jak i białych jest wybierana losowo.

Wymagania

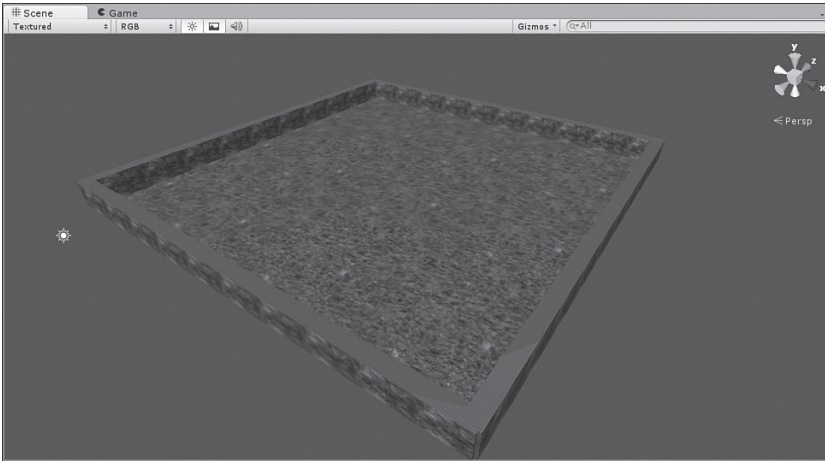
Wymagania dla tworzonej gry są proste. To nie będzie gra z rozbudowaną grafiką, natomiast zdecydowanie oparta na skryptach i interakcjach. Wymagania dla gry *Chaos Ball* przedstawiają się następująco.

- ▶ Fragment ogrodzonego murem terenu, który będzie służył w charakterze areny.
- ▶ Na teren i obiekty gry będą nałożone tekstury dostarczane standardowo z środowiskiem Unity.
- ▶ Kilka kul zarówno kolorowych, jak i białych. Wspomniane kule zostaną wygenerowane w Unity.

- ▶ Kontroler postaci, który znajdziesz w standardowych zasobach Unity.
- ▶ Kontroler gry, który utworzysz w Unity.
- ▶ Materiał fizyczny pozwalający na odbijanie się kul, który zostanie utworzony w Unity.
- ▶ Kolorowe oznaczenia narożników, które zostaną wygenerowane w Unity.
- ▶ Interaktywne skrypty, które przygotujesz w edytorze oprogramowania MonoDevelop.

Arena

Pierwszym krokiem jest utworzenie areny, na której będzie toczyła się akcja gry. W tym miejscu użyto pojęcia arena, aby wskazać, że teren jest całkiem mały i otoczony murem. Ani gracz, ani żadna z kul nie powinny opuszczać areny. Jak widać na rysunku 11.1, arena jest dość prosta.



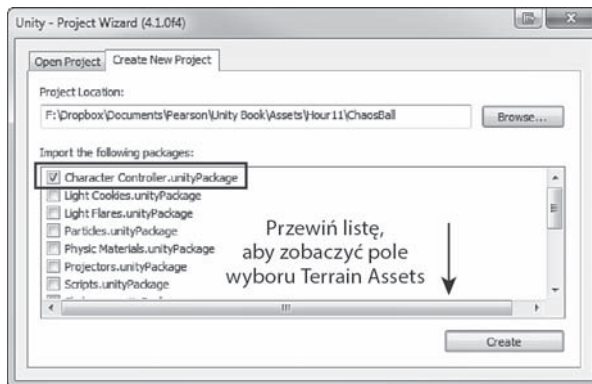
RYСУNEK 11.1.
Arena
dla tworzonej gry

Utworzenie areny

Jak wcześniej wspomniano, utworzenie areny będzie prostym procesem, co wynika z jej konstrukcji. Aby przygotować arenę, wykonaj wymienione poniżej kroki.

1. Utwórz nowy projekt i nadaj mu nazwę *ChaosBall*. Tym razem w oknie dialogowym *Create New Project* zaznacz pola wyboru *Character Controller.unityPackage* i *Terrain Assets.unityPackage* (patrz rysunek 11.2). Następnie do projektu dodaj teren.
2. Wymiary terenu to 50 na 50 (pamiętaj, aby użyć sekcji *Resolution* w ustawieniach terenu wyświetlanych w panelu *Inspector*). Do sceny dodaj światło kierunkowe i usuń obiekt *Main Camera*.

RYSUNEK 11.2.
Okno dialogowe
Create New Project



3. Do sceny dodaj sześcian. Umieść go w położeniu (0, 1.5, 2.5) i przeskaluj (1.5, 3, 51). Zwróć uwagę, jak zmodyfikowany sześcian stał się jedną ścianą muru otaczającego arenę. Nazwę sześcianu zmień na *Wall*.
4. W katalogu *Scenes* zapisz tę scenę pod nazwą *Main*.

Wskazówka

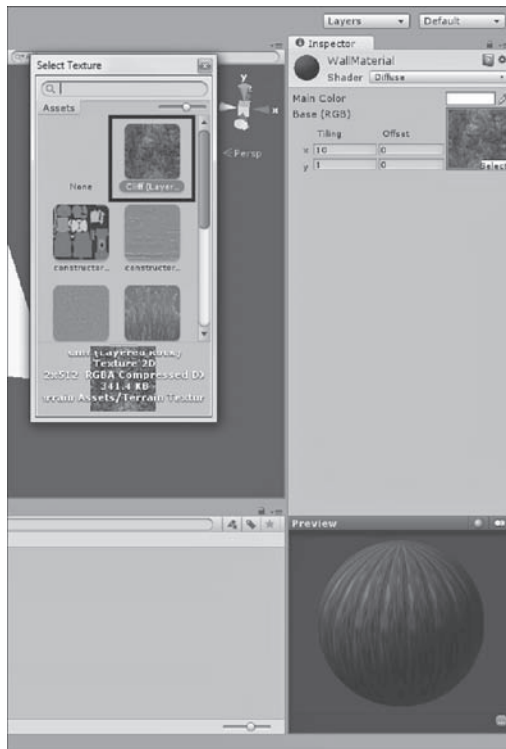
Konsolidacja obiektów

Być może zastanawiasz się, dlaczego utworzyłeś mur tylko dla jednej strony, choć powinienś dla wszystkich. Idea polega na tym, aby uniknąć zbędnego powieliania żmudnej pracy. Bardzo często zdarza się, że jeśli kilka wymaganych obiektów przedstawia się podobnie, wystarczy utworzyć jeden i kilkakrotnie go powielić. W omawianym przykładzie przygotujesz pojedynczą ścianę muru wraz z wymaganymi materiałami i właściwościami, a następnie po prostu trzykrotnie ją skopiujesz. Takie samo rozwiązanie zostanie użyte dla narożników, kul białych i kolorowych. W ten sposób powinienś się przekonać, jak odrobina planowania może pozwolić na zaoszczędzenie dużej ilości czasu.

Teksturowanie

Na obecnym etapie prac arena prezentuje się nędznie i nijako. Wszystko jest w kolorze białym, a otaczający ją mur składa się z tylko jednej ściany. Kolejnym krokiem jest więc dodanie pewnych tekstur i ożywienie areny. Potrzebujesz przede wszystkim tekstur dla dwóch obiektów: podłoża i muru. Podczas wykonywania tego kroku możesz śmiało poeksperymentować z teksturuowaniem, być może uzyskasz niezwykle interesujące efekty!

1. Za pomocą panelu Project w katalogu *Assets* utwórz nowy podkatalog o nazwie *Materials*. Do katalogu dodaj nowy materiał (prawym przyciskiem myszy kliknij katalog *Materials*, a następnie wybierz opcję *Create/Material*). Nowemu materiałowi nadaj nazwę *WallMaterial*.
2. Jak pokazano na rysunku 11.3, właściwości *Tiling* wzdłuż osi X ustaw wartość 10.



RYСУNEK 11.3.
Dodanie tekstury
do definiowanego
materiału

3. W panelu Inspector nałóż teksturę *Cliff (Layered Rock)* na materiał muru (patrz rysunek 11.3).
4. Kliknij i przeciągnij materiał na utworzony wcześniej obiekt muru.

Teraz przejdź do teksturowania podłoża. Ponieważ podłoże jest terenem, będzie teksturowane nieco inaczej, o czym powinienś już wiedzieć.

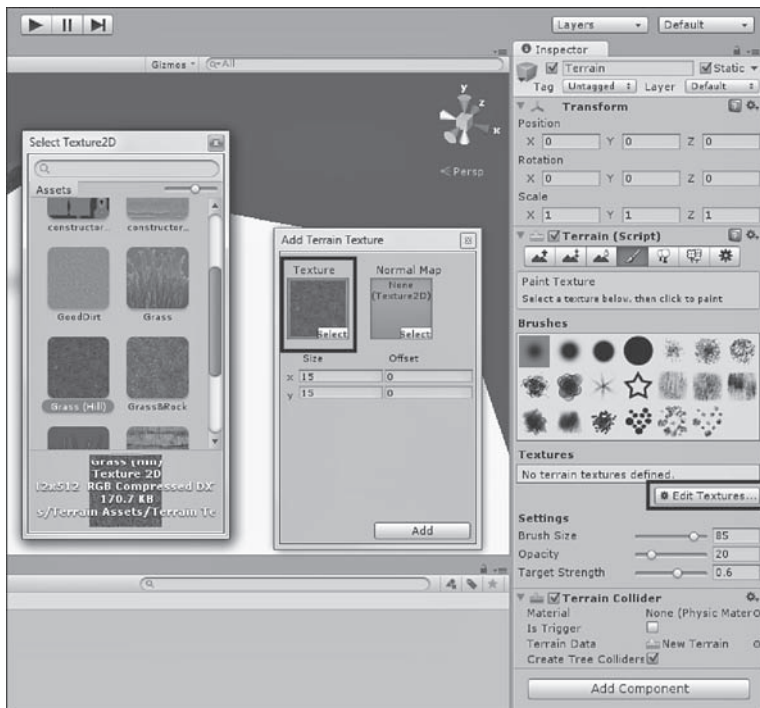
1. Po zaznaczeniu terenu w panelu Inspector wybierz narzędzie przeznaczone do teksturowania terenu (patrz rysunek 11.4).
2. Kliknij *Edit Textures/Add Texture*. W wyświetlonym oknie dialogowym *Add Terrain Texture* wybierz *Grass (Hill)* i kliknij przycisk *Add*.
3. Teren powinien otrzymać teksturę trawy.

Materiał zapewniający rewelacyjne odbijanie się obiektu

Dążymy do tego, aby obiekty odbijały się od ścian muru bez utraty pędu. Dlatego też konieczne jest przygotowanie materiału zapewniającego rewelacyjne odbijanie się obiektu. Jak pewnie sobie przypominasz, Unity oferuje pewien zestaw materiałów fizycznych. Jednak znajdujący się wśród nich materiał odbijania okazuje się niewystarczająco dobry do naszych potrzeb. Trzeba więc utworzyć nowy materiał, wykonując wymienione poniżej kroki.

RYSUNEK 11.4.

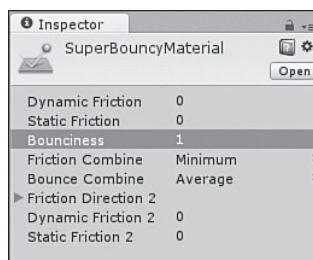
Dodanie tekstury
do terenu



1. Prawym przyciskiem myszy kliknij katalog *Materials* i wybierz opcję *Create/Physical Material*. Materiałowi nadaj nazwę *SuperBouncyMaterial*.
2. Ustaw dla materiału właściwości pokazane na rysunku 11.5. Ogólnie rzecz biorąc, konieczne jest zminimalizowanie wszystkich parametrów, które powodują redukcję energii.

RYSUNEK 11.5.

Ustawienia materiału
SuperBouncyMaterial



3. Kliknij nowy materiał, a następnie przeciągnij go na obiekt ściany muru. Materiał zostanie automatycznie zastosowany jako materiał fizyczny dla komponentu *Collider*. Powinieneś zobaczyć, że materiał jest wymieniony we właściwości *Material* komponentu *Box Collider*.

Zakończenie prac nad areną

Po przygotowaniu ściany i podłoża możesz przystąpić do zakończenia prac nad areną. Najtrudniejsze zadania zostały już wykonane i pozostało tylko powielenie ściany (kliknij ją prawym przyciskiem myszy w panelu Hierarchy, a następnie wybierz opcję *Duplicate*). Oto kroki do wykonania.

1. Jednokrotnie powiel ścianą i nową umieść w położeniu (50, 1.5, 25).
2. Ponownie powiel ścianę, umieść ją w położeniu (25, 1.5, 0) i zastosuj rotację (0, 90, 0).
3. Powiel ścianę utworzoną w poprzednim kroku (tę obróconą), a następnie umieść ją w położeniu (25, 1.5, 50).

Arena powinna teraz zawierać mur składający się z wszystkich czterech ścian i pozbawiony luk (patrz rysunek 11.1).

Elementy gry

W tej części lekcji utworzysz różne obiekty gry, które będą niezbędne do prowadzenia rozgrywki. Podobnie jak w przypadku muru otaczającego arenę, najłatwiej zbudować jeden obiekt, a następnie powielać go wedle potrzeb.

Gracz

Gracz w tej grze będzie zmodyfikowanym kontrolerem postaci *First Person*. Podczas tworzenia projektu należy zaznaczyć opcję importu pakietu kontrolera postaci. Kliknij kontroler postaci *First Person*, przeciągnij go na scenę, a następnie umieść w położeniu (46, 1, 4) i zastosuj rotację (0, 315, 0).

Przede wszystkim trzeba przesunąć kamerę w górę i odsunąć od kontrolera. W ten sposób gracz będzie miał lepsze pole widzenia podczas gry. Wykonaj zatem wymienione poniżej kroki.

1. W panelu Inspector rozwiń kontroler *First Person* (kliknij strzałkę wyświetlaną obok jego nazwy) i odszukaj *Main Camera*. Nie będziesz mieć wątpliwości, że znalazłeś właściwą, ponieważ jest oznaczona kolorem niebieskim.
2. Po zaznaczeniu kamery kontrolera umieść ją w położeniu (0, 5, -3.5) i zastosuj rotację (43, 0, 0). Kamera powinna być teraz z tyłu ponad kontrolerem i nieco obrócona w dół.

Kolejnym zadaniem jest dodanie paletki do sceny. Paletka to płaska powierzchnia używana przez gracza do odbijania kul. W celu dodania paletki wykonaj poniższe kroki.

1. Dodaj sześcian do sceny, zmień jego nazwę na *Bumper* i przeskaluj (3.5, 3, 1).
2. Kliknij utworzony wcześniej materiał fizyczny i przeciągnij na paletkę.

3. W panelu Hierarchy kliknij paletkę i przeciągnij na kontroler *First Person*. W ten sposób paletka zostanie zagnieżdżona w kontrolerze. Następnie zmień położenie paletki na (0, 0, 0.1) i zastosuj rotację (0, 0, 0). Paletka znajdzie się teraz w niewielkiej odległości przed kontrolerem.

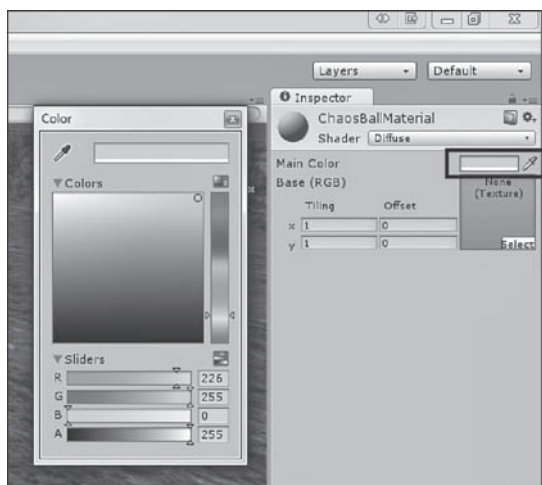
Ostatnim krokiem jest przyśpieszenie nieco gracza. Zaznacz kontroler *First Person*, a następnie w panelu Inspector rozwiń właściwość *Movement* komponentu *Character Motor (Script)*. Ustaw maksymalną szybkość do przodu na 11, natomiast maksymalną szybkość na boki na 10.

Kula chaos ball

Kule *chaos ball* to szybko i szaleńczo poruszające się kule, które mają utrudniać grę i przeszkadzać graczowi. Pod wieloma względami są podobne do kul kolorowych, a więc praca polega na nadaniu im uniwersalnych zasobów. W celu utworzenia pierwszej kuli typu *chaos ball* wykonaj opisane poniżej kroki.

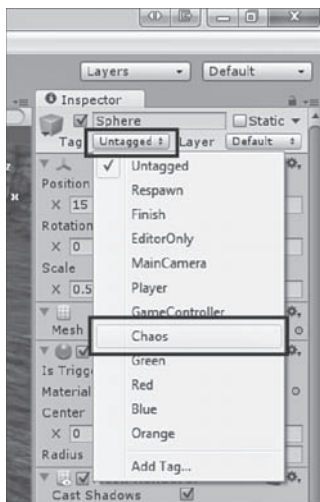
1. Dodaj kulę do sceny. Następnie zmień nazwę kuli na *Chaos*, umieść ją w położeniu (15, 2, 25) i przeskaluj (0.5, 0.5, 0.5).
2. Kliknij materiał *SuperBouncyMaterial* i przeciągnij go na kulę.
3. Utwórz nowy materiał (*niefizyczny*) dla kuli i nadaj mu nazwę *ChaosBallMaterial*. W sekcji koloru materiału wybierz kolor jasnożółty (patrz rysunek 11.6). Nowo utworzony materiał przeciągnij na kulę.

RYSUNEK 11.6.
Ustawienia
materiału
ChaosBallMaterial



4. Do kuli dodaj komponent *Rigidbody*. Właściwości *Angular Drag* ustaw wartość 0 i usuń zaznaczenie przy *Use Gravity*. Z rozwijanego menu właściwości *Collision Detection* wybierz opcję *Continuous*. Z kolei w właściwości *Constraints* zamroź położenie Y — nie chcesz, aby kula mogła poruszać się w górę oraz w dół.

- Przejdź do menedżera TagManager (wybierz opcję *Edit/Project Settings/Tags*), rozwiń sekcję *Tags*, klikając strzałkę znajdującą się obok nazwy sekcji, a następnie w pozycji *Element 0* dodaj tag *Chaos*. Przy okazji dodaj także tagi *Green*, *Orange*, *Red* i *Blue* — wykorzystasz je później w tym projekcie.
- Zaznacz kulę, a następnie w panelu Inspector zmień jej tag na *Chaos* (patrz rysunek 11.7).



RYСУNEK 11.7.
Wybór tagu
Chaos

W tym momencie kula jest ukończona, ale jeszcze nic się nie dzieje. Konieczne jest opracowanie skryptu pozwalającego na przesuwanie kuli po całej arenie. Utworzymy skrypt o nazwie *VelocityScript* i dołączymy go do kuli *chaos ball*. Pełny kod skryptu przedstawiono w listingu 11.1.

Listing 11.1. Skrypt *VelocityScript.cs*

```
using UnityEngine;
using System.Collections;

public class VelocityScript : MonoBehaviour {
    public float max = 50;

    // Metoda używana do inicjacji.
    void Start () {
        rigidbody.velocity = new Vector3(Random.Range(0, max), 0,
        ↳ Random.Range(0,
            max));
    }
    // Metoda wywoływana raz w trakcie każdej klatki.
    void Update () {
    }
}
```

Jak możesz zobaczyć w listingu, metoda `Random.Range()` jest użyta w celu nadania kuli prędkości początkowej dla osi X i Y, wygenerowanej losowo w przedziale od 0 do 50. Parametrami funkcji `Random.Range()` są dwie liczby, natomiast wartością zwrótną jest liczba z zakresu tworzonego przez dwie liczby będące parametrami.

Uruchom scenę i zobacz, jak kula porusza się po arenie. Na tym etapie kula *chaos ball* jest już gotowa. W panelu Hierarchy powiel tę kulę czterokrotnie. Rozrzuć nowo utworzone kule po arenie (upewnij się, że zmieniają jedynie położenie w zakresie osi X i Z), a ponadto dla każdej z nich zastosuj inną wartość rotacji wokół osi Y. Pamiętaj, że ruch wzdłuż osi Y jest niemożliwy i dlatego każda kula powinna mieć wartość 2 dla osi Y.

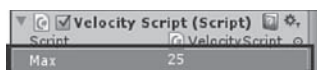
Kolorowe kule

Wprawdzie kula *chaos ball* jest żółta i to niewątpliwie kolor, ale określenie *kolorowe kule* dotyczy niezbędnych do wygrania czterech kul w różnych kolorach: czerwonym, pomarańczowym, niebieskim i zielonym. Podobnie jak w przypadku *chaos ball*, można przygotować tylko jedną kulę, a następnie powielić ją, tym samym ułatwiając sobie pracę.

Aby utworzyć pierwszą kulę, wykonaj wymienione poniżej kroki.

1. Dodaj kulę do sceny. Zmień jej nazwę na *Blue* i umieść ją w pobliżu środka areny. Upewnij się tylko, że wartość jej położenia dla osi Y wynosi 2.
2. Utwórz nowy materiał o nazwie *BlueMaterial* i ustaw mu kolor niebieski, dokładnie w ten sam sposób, jak to zrobiłeś dla kuli *chaos ball* (patrz rysunek 11.6). Następnie utwórz materiały *RedMaterial*, *GreenMaterial* oraz *OrangeMaterial* i ustaw im odpowiednie kolory (czerwony, zielony i pomarańczowy). Kliknij materiał *BlueMaterial* i przeciągnij na kulę.
3. Kliknij materiał *SuperBouncyMaterial* i przeciągnij go na kulę.
4. Do kuli dodaj komponent *Rigidbody*. Właściwości *Angular Drag* ustaw wartość 0 i usuń zaznaczenie przy *Use Gravity*. We właściwości *Constraints* zamroź położenie Y.
5. Podczas pracy nad kulą *chaos ball* utworzyłeś tag o nazwie *Blue*. Teraz zmień tag kuli na wspomniany *Blue*. Procedura zmiany jest taka sama jak w przypadku *chaos ball* (patrz rysunek 11.7).
6. Do kuli dołącz skrypt *VelocityScript*. W panelu Inspector odzyskaj komponent *Velocity Script (Script)* i zmień wartość jego właściwości *Max* na 25 (patrz rysunek 11.8). Ta zmiana spowoduje, że kolorowa kula będzie na początku poruszała się znacznie wolniej niż *chaos ball*.

RYСУNEK 11.8.
Zmiana
właściwości Max



Jeżeli teraz uruchomisz scenę, powinieneś zobaczyć niebieską kulę szybko poruszającą się po arenie. Można więc przystąpić do budowania trzech pozostałych kul, z których każda będzie kopią niebieskiej. W celu utworzenia pozostałych kul wykonaj wymienione poniżej kroki.

1. Powiel istniejącą niebieską kulą i nowym kulom nadaj nazwy odpowiadające ich kolorom, czyli *Red*, *Orange* i *Green*.
2. Nowym kulom nadaj tagi odpowiadające nazwie. Bardzo ważne jest, aby nazwa i tag były dokładnie takie same.
3. Przeciągnij odpowiednie materiały na nowe kule. Bardzo ważne jest, aby kolor kuli odpowiadał jej nazwie.
4. Nowe kule umieść w losowo wybranych położeniach, zmień dowolnie ich rotację, ale upewnij się, że wartość *Y* dla położenia wynosi 2.

Na tym etapie prac elementy gry są już gotowe. Kiedy uruchomisz scenę, zobaczysz wszystkie kule poruszające się po arenie.

Obiekty kontrolne

Po przygotowaniu wszystkich niezbędnych elementów możemy przystąpić do gamifikacji. Tym razem zamienimy je w dostarczającą rozrywki grę. W tym celu konieczne jest utworzenie czterech narożników, skryptów oraz kontrolera gry. Dopiero wtedy będziemy mieli gotową grę.

Cele

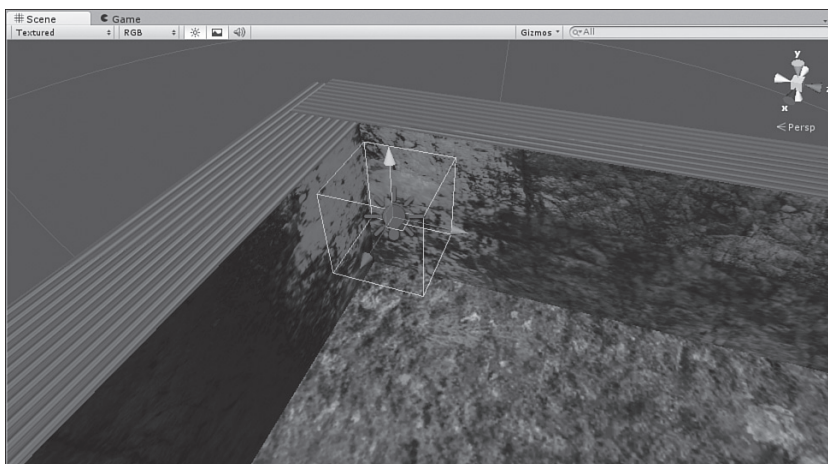
Każdy z czterech narożników jest w kolorze odpowiadającym jednej z kolorowych kul. Gdy kula znajdzie się w danym narożniku, wtedy gra sprawdzi wartość przypisanego jej tagu. Gdy wartości tagu i kolor narożnika są takie same, wtedy mamy dopasowanie. Po znalezieniu dopasowania dla kuli zostanie włączona właściwość *Kinematic* (jak pamiętasz, powoduje unieruchomienie obiektu), a cel zostanie uznany za osiągnięty. Podobnie jak w przypadku obiektów kul, także teraz można utworzyć jeden cel, a następnie powielić go wymaganą ilość razy.

Aby utworzyć pierwszy cel, wykonaj wymienione poniżej kroki.

1. Utwórz pusty obiekt gry (wybierz opcję *GameObject/Create Empty*), zmień mu nazwę na *BlueGoal*, przypisz tag o nazwie *Blue* i następnie umieść obiekt w położeniu (1.6, 2, 1.6).
2. Do obiektu dodaj komponent *Box Collider* i ustaw jego właściwość *Is Trigger*. Wielkość dodanego komponentu zmień na (1.5, 1.5, 1.5).
3. Dodaj światło do obiektu (wybierz opcję *Component/Rendering/Light*). Światło powinno być punktowe, w kolorze odpowiadającym oczekiwanemu przez dany narożnik kolorowi kuli (patrz rysunek 11.9). Intensywność światła ustaw na 3.

RYSUNEK 11.9.

Zdefiniowany
narożnik
w kolorze
niebieskim



Kolejnym krokiem jest utworzenie skryptu o nazwie *GoalScript* i dołączenie go do niebieskiego narożnika. Zawartość skryptu przedstawiono w listingu 11.2.

Listing 11.2. Skrypt GoalScript.cs

```
using UnityEngine;
using System.Collections;

public class GoalScript : MonoBehaviour {
    private bool solved = false;

    // Metoda używana do inicjacji.
    void Start () {
    }

    // Metoda wywoływana raz w trakcie każdej klatki.
    void Update () {
    }

    void OnTriggerEnter(Collider other)
    {
        if(other.tag == tag)
        {
            solved = true;
            other.rigidbody.isKinematic = true;
        }
    }

    public bool IsSolved()
    {
        return solved;
    }
}
```

Jak możesz zobaczyć w skrypcie, metoda `OnTriggerEnter()` sprawdza wartość tagu każdego obiektu zderzającego się z narożnikiem i porównuje z tagiem narożnika. W przypadku dopasowania tagów kula zostaje unieruchomiona, a cel oznaczony jako osiągnięty.

Zmienna prywatna

Jak zapewne zauważyłeś, skrypt *GoalScript* zawiera zmienną prywatną o nazwie `solved` i metodę publiczną `IsSolved()`. Wartością zwrótną metody jest zmienna. Być może zastanawiasz się, dlaczego wykonano dodatkową pracę, aby zmienna była publiczna. Ma to na celu uniemożliwienie wszelkim innym obiektom lub skryptom przypadkowego oznaczenia celu jako osiągniętego. Ponieważ żaden inny element nie może uzyskać dostępu do zmiennej `solved` poza samym obiektem celu, nie ma niebezpieczeństwa przypadkowej zmiany wartości wymienionej zmiennej. Z kolei metoda `IsSolved()` istnieje jedynie po to, aby poinformować obiekt kontrolny gry o osiągnięciu danego celu.

Uwaga
Uwaga

Po przygotowaniu skryptu i dołączeniu go do narożnika (celu) możesz przystąpić do powielania narożnika. Aby utworzyć pozostałe, wykonaj wymienione poniżej kroki.

1. Powiel narożnik *BlueGoal* i powstałym narożnikom nadaj nazwy odpowiadające kolorom, czyli *RedGoal*, *GreenGoal* i *OrangeGoal*.
2. Tag celu zmień na odpowiadający kolorowi.
3. Kolor światła narożnika zmień na odpowiadający celowi.
4. Umieść narożnik w odpowiednim położeniu. Kolory mogą być umieszczane w dowolnych narożnikach, jednak każdy kolor w oddzielnym. Trzy pozostałe położenia narożników to (1.6, 2, 48.4), (48.4, 2, 1.6) i (48.4, 2, 48.4).

Wszystkie narożniki powinny być już przygotowane i w pełni gotowe do działania.

Kontroler gry

Ostatni element niezbędny do zakończenia budowy gry to kontroler gry. Kontroler będzie odpowiedzialny za sprawdzanie wszystkich celów w każdej klatce i określanie, kiedy zostaną osiągnięte. Dla gry tworzonej w tej lekcji kontroler jest bardzo prosty. W celu jego utworzenia wykonaj wymienione poniżej kroki.

1. Do sceny dodaj pusty obiekt gry. Przenieś go w dowolne miejsce i zmień nazwę na *GameController*.
2. Utwórz skrypt o nazwie *GameControlScript* i umieść w nim kod przedstawiony w listingu 11.3. Gotowy skrypt dołącz do kontrolera gry.

3. Po zaznaczeniu kontrolera gry kliknij i przeciągnij poszczególne narożniki (cele) na odpowiadające im właściwości w komponencie *Game Control Script* (patrz rysunek 11.10).

Listing 11.3. Skrypt *GameControlScript*

```
using UnityEngine;
using System.Collections;

public class GameControlScript : MonoBehaviour {
    public GoalScript red;
    public GoalScript blue;
    public GoalScript orange;
    public GoalScript green;

    private bool isGameOver = false;

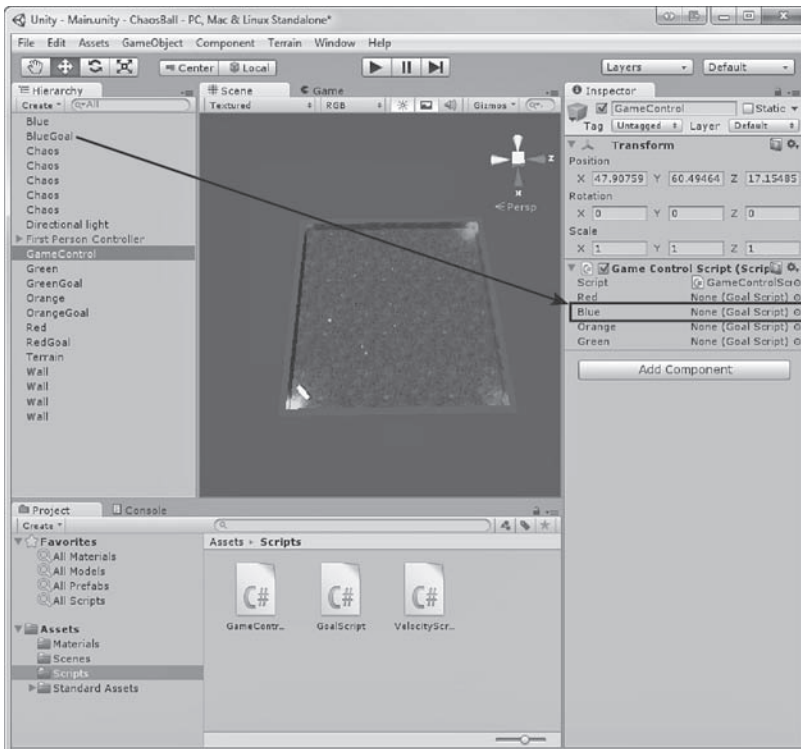
    // Metoda używana do inicjacji.
    void Start () {
    }

    // Metoda wywoływana raz w trakcie każdej klatki.
    void Update () {
        if(red.IsSolved() && blue.IsSolved() && orange.IsSolved()
           && green.IsSolved())
        {
            isGameOver = true;
        }
    }

    void OnGUI()
    {
        if(isGameOver)
        {
            GUI.Box(new Rect(Screen.width / 2 - 100,
                             Screen.height / 2 - 50, 200, 75), "Koniec gry");
            GUI.Label(new Rect(Screen.width / 2 - 30,
                               Screen.height / 2 - 25, 60, 50), "Dobra
                               ↪robota!");
        }
    }
}
```

Jak możesz zobaczyć w przedstawionym powyżej skrypcie, kontroler gry zawiera odniesienia do wszystkich czterech celów. W trakcie generowania każdej klatki kontroler sprawdza wszystkie cztery cele, aby ustalić, czy zostały osiągnięte. Jeżeli tak się stanie, zmiennej *isGameOver* przypisana zostaje wartość *true* i na ekranie wyświetlany jest komunikat kończący grę.

Gratulacje! W ten sposób zakończyłeś tworzenie gry *Chaos Ball*.



RYSUNEK 11.10.
Dodanie celów
do kontrolera gry

Usprawnienie gry

Wprawdzie ukończyłeś pracę nad grą *Chaos Ball*, ale niewątpliwie nie jest ona doskonała. Wiele pominiętych funkcji mogłoby znacznie poprawić grywalność. Wspomniane funkcje zostały pominięte, aby umożliwić Ci eksperymenty z grą i wprowadzanie w niej usprawnień. Można więc stwierdzić, że *Chaos Ball* to jedynie ukończony prototyp. Jest to działająca wersja gry, choć wymagająca wykończenia. Dlatego też zachęcam do ponownej lektury rozdziału i wyszukania aspektów gry, które można poprawić. Najlepiej postaw się w roli gracza i spróbuj odpowiedzieć na poniższe pytania.

- ▶ Czy gra jest zbyt łatwa, czy może zbyt trudna?
- ▶ Co może ułatwić lub utrudnić rozgrywkę?
- ▶ Co może spowodować, że gra zachwyci graczy?
- ▶ Które fragmenty gry są zabawne, a które nużące?

W przedstawionym na końcu rozdziału ćwiczeniu będziesz miał możliwość poprawiania gry i dodania do niej nowych funkcji. Jeżeli otrzymasz jakikolwiek błąd, oznacza to, że prawdopodobnie pominąłeś któryś z kroków. Upewnij się, że dokładnie sprawdziłeś wszystko, co powinno pomóc w rozwiązaniu ewentualnych błędów.

Podsumowanie

W tej lekcji opracowałeś grę zatytułowaną *Chaos Ball*. Na początek wykonałeś fazę projektowania. Określiłeś koncepcję, reguły i wymagania. Następnie przystąpiłeś do tworzenia areny. Przy tej okazji dowiedziałeś się, że można utworzyć jeden obiekt, a następnie wielokrotnie powielać go, w ten sposób oszczędzając czas. Później przeszedłeś do tworzenia gracza, kul *chaos ball* i kolorowych, (narożników) celów, a także kontrolera gry. Na końcu zyskałeś możliwość zagrania w grę oraz wyszukania aspektów wartych usprawnienia.

Pytania i odpowiedzi

Pytanie: Dlaczego do wykrywania kolizji kul wykorzystujemy wartość **Continuous** właściwości **Collision Detection**? Byłem przekonany, że jej użycie powoduje spadek wydajności gry.

Odpowiedź: Nieustanne wykrywanie zderzeń faktycznie może zmniejszyć wydajność gry. W omawianej grze jest jednak niezbędne. Ponieważ kula *chaos ball* jest mała i bardzo szybka, istnieje więc niebezpieczeństwo, że czasami mogłaby „przejsć” przez ściany.

Pytanie: Osiągnięcie celu jest określane na podstawie wartości tagu kuli. Czy ten sam cel można uzyskać na podstawie jedynie nazwy kuli?

Odpowiedź: Oczywiście! W omawianej grze tagi zostały użyte, by uprościć aplikację. Dzięki zastosowaniu tagów i edytorów skrypty mogły pozostać uogólnione. W ten sposób skrypt jest utworzony tylko raz, a wykorzystywany aż czterokrotnie.

Warsztaty

Poświęć nieco czasu i odpowiedz na zamieszczone poniżej pytania, a także upewnij się, że przyswoiłeś sobie materiał omówiony w tej lekcji.

Quiz

1. W grze został użyty dostępny w środowisku Unity materiał fizyczny zapewniający odbijanie się obiektu. Prawda czy fałsz?
2. Jak gracz może przegrać w utworzonej tutaj grze?
3. Które osie pozycji obiektów kul zostały zamrożone?
4. Osiągnięcie celu jest sprawdzane w metodzie `OnTriggerEnter()`. W wymienionej metodzie sprawdzamy, czy obiekt jest oczekiwaną kulą. Prawda czy fałsz?
5. Dlaczego w grze pominięto pewne podstawowe funkcje?

Odpowiedzi

1. Fałsz. Na potrzeby gry utworzyliśmy nowy materiał fizyczny zapewniający wręcz rewelacyjne odbijanie się obiektów.
2. To jest podchwytliwe pytanie. Gracz nie może przegrać.
3. To jest oś Y.
4. Prawda.
5. Aby dać czytelnikowi szansę na ich implementację.

Ćwiczenie

W procesie tworzenia gier najlepsze jest to, że można je kreować wedle własnego upodobania. Kierowanie się wskazówkami jest przydatne podczas nauki, ale w ten sposób nie osiągniesz satysfakcji, takiej jak z samodzielnego utworzenia własnej gry. W przedstawionym tutaj ćwiczeniu zyskujesz możliwość zmodyfikowania gry, aby stała się nieco bardziej unikalna. Dokładny sposób modyfikacji zależy tylko od Ciebie. Poniżej wymieniono jedynie kilka sugestii.

- ▶ Spróbuj dodać przycisk pozwalający graczowi na ponowne rozpoczęcie gry po jej zakończeniu. (Elementy graficznego interfejsu użytkownika nie zostały jeszcze omówione, ale ta funkcja istnieje w ostatnio utworzonej grze. Przekonaj się, czy potrafisz dodać wspomniany przycisk).
- ▶ Spróbuj dodać licznik, aby gracz wiedział, ile czasu potrzebował na zakończenie gry.
- ▶ Spróbuj dodać większe zróżnicowanie dla kuli *chaos ball*.
- ▶ Spróbuj dodać kolejny cel, który wymaga wszystkich kul *chaos ball*.
- ▶ Spróbuj zmienić wielkość lub kształt paletki gracza. Spróbuj utworzyć paletkę w wielu różnych kształtach.

Skorowidz

A

anatomia
 kamery, 114
 metody, 162
 skryptu, 145
animacja
 Idle, 311, 317
 Jump, 332
 modelu, 294, 302
 ruchu, 320
 Run, 333
 WalkForward, 313
 WalkForwardTurn, 314
animacje, 293, 295
 przygotowanie, 311
 przygotowanie modelu, 294, 295
 zasoby, 298
 zmiana, 304
animator, 307, 316
aplikacja Unity Remote, 362
aplikacje mobilne, 362
arena, 195, 199
aureola, 111

B

bezpieczeństwo danych, 389
billboardy, 92
blok
 lokalny, 150
 metody, 163
błąd, 111
błędy konsoli, 378
bryły sztywne, 180, 181
budowanie świata gry, 128

C

cookies, 112
cząsteczki, 278, 279
czcionki, 239

D

dane
 wejściowe, 167, 169
 wejściowe myszy, 170
 wejściowe z ekranu dotykowego, 366
 wyjściowe skryptu, 145
dodawanie
 animacji, 300
 celów, 207
 cookie, 112, 113
 efektu Lens Flare, 99
 egzemplarza prefabrykatu, 216
 elementów, 129
 karty, 33
 klipu, 333
 klipu animacji, 314
 kontrolera postaci, 101, 244
 mgły, 98
 Motion Field, 320
 obiektów kontrolujących, 131
 parametru Jumping, 334
 parametrów, 319
 płaszczyzny, 288
 reflektora, 109
 scen, 385
 skryptów, 133
 symulacji nieba, 97
 światła kierunkowego, 110
 światła punktowego, 108
 tagów, 174
 tekstury, 197, 198
 terenu, 76
 trawy, 93
 warstw, 121
 wody, 100
 żołnierza, 300
dodatkowa energia, 329
dokumentacja, 361
dołączanie skryptu, 142
dopasowywanie
 komponentów Collider, 183
 stylów, 235

- dopracowanie
 - gry, 383
 - szczegółów, 101
- dostęp do
 - komponentów lokalnych, 172
 - obiektów, 172
- dostępne zasoby, 402
- dostosowywanie
 - gier, 371
 - kontrolki, 234
- droga, 327
- drzewa, 90
- drzewo Blend Tree, 319
- dziedziczenie, 213, 217
- dźwięk, 345
 - 2D, 347, 351
 - 3D, 350
 - w panelu Scene, 350

E

- edytor, 29
- edytor krzywych, 290
- efekt
 - cząsteczek, 280
 - Lens Flare, 99, 100
- efekty środowiska, 95
- efektywność, 175
- egzemplarz, 212
- ekran dotykowy, 366
- elementy
 - gry, 199, 263, 328
 - zasobu modelu, 67
- etykieta, 228

F

- fabryka, 164
- faza projektowania, 126, 194, 260, 326
- formaty map wysokości, 80
- funkcja PiP, 116

G

- gamifikacja, 131
- generowanie
 - drzew, 90
 - terenu, 76
 - trawy, 90

- gra
 - Amazing Racer, 125, 372, 400
 - Captain Blaster, 259, 377, 400
 - Chaos Ball, 193, 375, 400
 - Gauntlet Runner, 325, 379, 400
- gracz, 199, 263, 270, 331
- graficzny interfejs użytkownika, 225
- grawitacja, 249, 253
- GUI, Graphical User Interface, 225

H

- horyzont, 96

I

- ignorowanie
 - kamer, 122
 - światła, 122
- ikony
 - pomocnicze sceny, 39, 41
 - rotacji, 55
 - translacji, 53
 - transformacji, 42
- implementacja
 - ruchu, 378
 - strzelania, 379
- importowanie
 - klipów audio, 348
 - modeli, 63
 - zasobów, 102
 - zasobów terenu, 83
- instalacja Unity, 26
- interfejs, 31
- iteracja, 157

J

- język C#, 149
- język skryptowy, 142

K

- kamera, 114, 261
- katalog
 - Assets, 33
 - Materials, 74
 - Scenes, 36
 - Scripts, 133
 - Water, 100

- klip audio, 354
- kod skryptu, 146
- kody klawiszy, 170
- kolejność scen, 384
- kolizje, 179, 182, 250
- kolorowe kule, 202
- komentarze, 147
- kompilacja gry, 391
- komponent
 - Animation, 300
 - Audio Listener, 102, 116, 347
 - Audio Source, 348
 - Capsule Collider, 264
 - Collider, 182–184, 247
 - Collider Quarrel, 244
 - kontrolera postaci, 245
 - Light, 106
 - Mouse Look, 376
 - Respawn Script, 135
 - Rigidbody, 180, 186
- komponenty lokalne, 172
- koncepcja, 126, 194, 260, 326
- konfiguracja
 - sceny, 316
 - środowiska, 360
 - urządzenia, 363
- konfigurowanie ruchu, 372
- konsola, 147
- konsolidacja obiektów, 196
- konstrukcja
 - if, 154
 - if-else, 155
 - if-else if, 156
- konstrukcje warunkowe, 153
- kontroler
 - First Person, 372
 - gry, 127, 205, 266
 - postaci, 101, 130, 243, 248–251
- kontrolka
 - etykiety, 228
 - paska narzędziowego, 231
 - pola, 229
 - pola tekstowego, 232
 - pola wyboru, 230
 - przycisku, 229
 - przycisku powtarzającego, 230
 - suwaka, 233
- kontrolki
 - efektów systemu cząsteczek, 279
 - GUI, 227

- narzędzia Hand, 43
- panelu Game, 42
- przyciągania, 44
- systemu cząsteczek, 278
- trybu Flythrough, 44
- kontrolowanie poślizgu, 248
- kula chaos ball, 200

L

- lista
 - parametrów, 163
 - wymagań, 127
- logowanie, 28
- lustrzane odbicie, 315

Ł

- łącza internetowe, 29
- łączenie
 - komponentów Collider, 183
 - obiektów, 135
 - stylów, 235

M

- malowanie
 - drzewami, 90
 - terenu teksturą, 85
 - trawą, 92
 - wzniesień, 81
- mapa wysokości, 77
- materiały, 66, 68
- materiały fizyczne, 184
- menedżer
 - Input Manager, 167
 - TagManager, 122
- menu
 - Aspect, 41
 - Effects, 39
 - GameObject, 51
 - Layer, 118
 - Layers, 42, 120
 - wyboru układu, 42
- meteory, 264
- metoda, 162
 - Find(), 174
 - FindWithTag(), 174
 - Move(), 248

metoda

- OnControllerColliderHit(), 250
- OnGUI(), 235, 368
- SimpleMove(), 248
- Start(), 148
- Translate(), 376
- Update(), 148, 370, 379

metody wbudowane, 148

mgła, 98

model, 62–66, 297

- Jack, 309
- Robot Kyle, 66
- żołnierza, 296

moduł

- Collision, 285
- Color by Speed, 284
- Color over Lifetime, 284
- domyślny, 280
- Emission, 281, 283
- External Forces, 285
- Force over Lifetime, 284
- Limit Velocity over Lifetime, 283
- Renderer, 289
- Rotation by Speed, 285
- Rotation over Lifetime, 285
- Shape, 281
- Size by Speed, 285
- Size over Lifetime, 284
- Sub Emitter, 288
- Texture Sheet, 288
- Velocity over Lifetime, 283

moduły systemu cząsteczek, 280

modyfikacja

- komponentów obiektu, 175
- zmiennej, 150

MonoDevelop, 144

N

nakładanie

- animacji Idle, 318
- mapy wysokości, 78
- materiałów, 71
- shaderów, 71
- tekstur, 71

narzędzia

- do rzeźbienia terenu, 80
- rotacji, 55
- transformacji, 41, 53, 54

narzędzie

- Hand, 42
- Place Trees, 90, 91
- Terrain Settings, 94

nasłuchiwanie dźwięku, 347

nazwa metody, 162

niebo, 96

O

obiekt

- Finish, 133
- GameControl, 135
- PlayerPrefs, 388, 389
- WaterHazardDetector, 133

obiekty

- 3D, 63
- gracza, 331
- gry, 47, 50
- kontrolne, 131, 203, 342
- potomne, 298
- wbudowane, 51
- zagnieżdżone, 57

obliczanie wysokości, 79

obsługa

- animacji, 322
- danych wejściowych, 168
- ruchu, 252

obszar tekstu, 233

odczyt

- danych wejściowych, 169
- naciśnięcia klawisza, 170
- ruchu myszą, 171

oddalanie, 44

odszukanie animacji w modelu, 312

odtwarzanie dźwięku, 353, 354

okno

- Add Tree, 91
- Build Settings, 385, 392
- Create New Project, 196
- Game Settings, 393
- Import Heightmap, 79
- konsoli, 147
- Project, 29

opcja

- Add Layer..., 119
- Add Tab, 32
- Add Tree, 91
- Console, 147
- Create Other, 51

- Light, 110
- Maximize on Play, 41
- Remove Component, 37
- Render Settings, 112
- Rendering/ Camera, 116
- Skybox, 97
- Spotlight, 109
- Store in Root, 297
- Tags, 119
- Terrain, 76
- opcje transformacji, 53
- operatory
 - arytmetyczne, 151
 - logiczne, 153, 154
 - przypisania, 152
 - równości, 152
- osiągnięcia, 398
- osie przyśpieszeniomierza, 363
- oś rotacji, 54
- oświetlenie sceny, 38

P

- pakiety, 31
- panel
 - Animator, 317, 318
 - Game, 39
 - Hierarchy, 34, 35
 - Inspector, 35, 83, 143
 - Project, 32, 33
 - Scene, 37, 42, 350
- parametry, 163
- parametry animatora, 318
- pasek narzędziowy, 41, 42, 231
- pędzel, 84
- pętla
 - for, 158
 - while, 157
- PiP, Picture in Picture, 119
- planowanie gry
 - koncepcja, 126, 194, 260, 326
 - reguły, 126, 194, 260, 326
 - wymagania, 127, 194, 260, 326
- platformy mobilne, 359, 360
- płynne przewijanie, 263
- pobieranie modeli, 65
- pociski, 265
- początek układu współrzędnych, 49
- podział ekranu, 116
- pole, 229
 - tekstowe, 232
 - wyboru, 230
- połączenie
 - skryptów, 134
 - skryptu gracza, 272
 - z zasobem prefabrykatu, 219
- popychanie obiektów, 254
- poślizg, 248
- prefabrykaty, 211
- proces uprawnień, 127
- program
 - 3D Studio Max, 295
 - Blender, 295
 - Maya, 295
 - MonoDevelop, 144
- programowanie
 - dla przyśpieszeniomierza, 364
 - na platformach mobilnych, 360
- projektowanie GUI, 226
- projekty, 31
- proporcje ekranu, 261
- przeciążenie warstwami, 120
- przejścia, 320
- przejście ze stanu Jump, 335
- przełączanie
 - platformy, 377
 - scen, 385
- przenoszenie zasobów, 33
- przeszkody, 330
- przewijanie drogi, 328
- przybliżanie, 44
- przycisk, 229
 - Add Component, 267
 - Dalej, 40
 - Pauza, 40
 - powtarzający, 230
 - Start, 40
 - Stats, 41
- przygotowanie
 - animacji, 311
 - modelu, 295
- przyśpieszeniomierz, 363
- punkt rozpoczęcia gry, 127
- punkty kluczowe, 290

R

- raycasting, 187
- reflektor, spotlight, 109, 114
- reguły, 126, 194, 260, 326

rigging, 295, 308, 309, 311
 rotacja, 54, 56
 rozbieżność osi, 365
 rozdzielczość, 77
 rozdzielczość sceny, 378
 rozglądanie się, 373
 rozpoczęcie gry, 127
 rozwijane menu
 Aspect, 41
 Layers, 42, 120
 wyboru układu, 42
 rozwinięcie, unwrap, 67
 ruch, 378
 ruch gracza, 372
 rzeźbienie
 mapy wysokości, 77
 terenu, 80–82, 129

S

scena, 36, 327
 SDK, Software Development Kit, 360
 sekcja
 deklaracji, 146
 klasy, 146
 Using, 146
 shader, 66, 68
 Bumped, 70
 Diffuse, 70
 Specular, 70
 siatka, mesh, 61
 silnik gier, 401
 skalowanie, 55, 56
 domyślne, 65
 siatki, 65
 sklep Asset Store, 65
 składnia, 149
 składniki dźwięku, 346
 skoki, 253, 374
 skórki GUI, 235, 238
 skrypt, 132, 141, 161
 AudioScript, 356
 GameControlScript, 206
 GoalScript.cs, 204
 gracza, 270
 LoadSceneTwo, 386
 meteoru, 267
 MobileInputScript, 372
 MouseInputScript, 375
 MouseLookScript, 376

obiektu
 dodatkowej energii, 339
 gracza, 338
 kontrolnego gry, 335
 przeszkód, 339
 PlayerScript, 378, 379
 pocisku, 272
 PrefabGenerator.cs, 222
 SaveData, 388
 SpawnScript, 340
 strefy wyzwalacza, 335
 VelocityScript.cs, 201
 wyzwalacza, 269
 skrypty
 dla kontrolerów postaci, 245, 255
 do obsługi animacji, 322
 do obsługi danych wejściowych, 168
 do obsługi dźwięku, 352
 słowo kluczowe this, 387
 spłaszczanie terenu, 81
 sterowanie postaciami, 243
 strefa wyzwalacza, 330
 struktura
 kodu, 149
 prefabrykatu, 213
 style, 238
 style GUI, 234
 suwaki, 233
 symulacja nieba, 96, 97
 system
 kamer, 118
 Shuriken, 281
 systemy
 animacji, 293
 cząsteczek, 277, 278
 edytor krzywych, 290
 moduły, 280
 operacyjne, 28
 szkielet, 294

Ś

ściana, 72
 śledzenie dotknięć, 367
 środowisko, 89
 światło, 92, 106
 kierunkowe, 109
 powierzchniowe, 110
 punktowe, 106

T

- tagi, 175
- tekstura, 66–68
- tekstura Robot_Color, 67
- teksturowanie, 196
- tekstury terenu, 83
- teren, 75
 - dodanie drzew, 90
 - dodanie tekstury, 198
 - dodanie trawy, 93
 - dodawanie do projektu, 76
 - malowanie teksturą, 85
 - narzędzia, 80
 - rzeźbienie mapy wysokości, 77
 - spłaszczanie, 81
 - tekstury, 83
 - ustawienia, 94, 95
- testowanie
 - dźwięku, 349, 351
 - gry, 136
 - konfiguracji urządzenia, 363
- tło, 261
- transformacja, 51–57, 173, 176
- translacja, 52, 56
- trawy, 90, 92
- trójkąty, 62
- tryb
 - dźwięku, 38
 - Flythrough, 43
 - generowania, 38
 - wyświetlania, 38
- tryby właściwości Wrap Mode, 302
- tworzenie
 - animatora, 316
 - areny, 195
 - drzew, 90
 - efektu PiP, 119
 - egzemplarza, 213
 - egzemplarza prefabrykatu, 220
 - gier, 401
 - GUI, 226, 227
 - jeziora, 100
 - kontrolera postaci, 250
 - materiałów, 74
 - meteorów, 268
 - metody, 163
 - obiektów gry, 52
 - płaskowyżów, 81
 - prefabrykatu, 215

- projektu, 30
- promieni, 190
- skryptów, 142
- stanu, 319
- systemu cząsteczek, 279
- systemu kamer, 118
- światła, 110
- tekstur terenu, 86
- wielu egzemplarzy prefabrykatu, 217
- własnego stylu, 237
- własnego terenu, 129
- zmiennej, 148
- typ wartości zwrotnej, 162
- typy zmiennych
 - bool, 149
 - char, 149
 - double, 149
 - float, 149
 - int, 149
 - string, 149

U

- uaktualnianie prefabrykatu, 218
- układ współrzędnych
 - 2D, 47, 49
 - 3D, 48, 49
 - lokalny, 50
 - początek, 49
 - składnia, 49
 - świata, 50
- Unity Free, 26
- Unity Pro, 26
- uruchamianie gry, 40
- urządzenia mobilne, 371
- usprawnianie gry, 207, 273, 343
- ustawienia
 - animacji, 298
 - animacji WalkForward, 313, 314
 - dla platform, 390
 - dźwięku 3D, 351
 - grawitacji, 249
 - komponentu Capsule Collider, 264
 - niezależne, 390
 - obiektów, 96
 - riggingu, 309, 331
 - rozdzielczości, 77
 - skrętu, 315
 - terenu, 94, 95
 - Unity Player, 390
 - wiatru, 96

usuwanie niewidocznych powierzchni, 28
użycie

- animacji, 299
- brył sztywnych, 181
- dźwięku, 352
- edytora krzywych, 291
- metod, 165
- obiektu PlayerPrefs, 388
- przyśpieszeniomierza, 365
- reflektora, 114
- warstw, 120
- wbudowanych metod, 148
- wielu kamer, 116
- właściwości Wrap Mode, 303

W

warstwa, 117, 120
warstwy gry, 38
wartości domyślne, 392
warunek, 127
wbudowane obiekty 3D, 63
wdrożenie, 383
wiatr, 96
widoczność warstw, 121
wielkość
 aureoli, 112
 terenu, 77
wierzchołek, 62
właściwości
 animacji
 Idle, 312
 Jump, 332
 Run, 333
 dźwięku 3D, 352
 kamery, 115, 251, 262
 komponentu
 Animation, 300
 Audio Source, 348
 Collider, 183
 Rigidbody, 180, 181
 kontrolera postaci, 245, 246
 materiałów, 74
 materiału fizycznego, 185
 mgły, 98
 modułu
 Collision, 286
 Color by Speed, 285
 domyślnego, 282
 Emission, 282

Limit Velocity over Lifetime, 283
Renderer, 289
Texture Sheet, 289
Velocity over Lifetime, 283
narzędzia Place Trees, 91
osi, 168
przejścia ze stanu Jump, 335
shaderów, 70, 71
sześcianu, 71
światła punktowego, 106, 107
tekstury cookie, 113
trybu World, 287
zmiennej Touch, 367
źródła dźwięku, 353
właściwość
 Apply Root Motion, 322
 Culling Mask, 122
 Wrap Mode, 301, 302
woda, 99, 100
współrzędne
 lokalne, 50, 51
 świata, 50, 51
wybór
 ikon pomocniczych, 38
 instalowanych komponentów, 27
 katalogu, 27
 licencji, 27
 platformy, 391
 tagu, 175, 201
 układu, 32
wydajność, 94
wygląd trawy, 94
wymagania, 127, 194, 260, 326
 dotyczące kolizji, 182
wymiar, dimension, 48, 58
wypalanie, baking, 108
wyszukiwanie obiektów, 173
wyświetlanie
 dokumentacji, 361
 okna Project, 30
wywołanie kolizji cząsteczek, 287
wywoływanie metod, 166
wyzwalacz, 185, 265, 269, 330

Z

zachowywanie
 danych, 387
 obiektów, 387, 388
zagnieżdżanie, 36

- zarządzanie
 - projektem, 34
 - sceną, 36, 384
- zasięg zmiennej, 148
- zasoby, 33, 102
 - zasoby animacji, 298
 - terenu, 90
- zasób Animator, 308
- zmiana
 - animacji, 304
 - klipu audio, 354
 - rozdzielczości sceny, 378
 - scen, 386
 - szybkości, 137
 - typu kolizji, 288
 - właściwości, 37
 - wyglądu drzew, 92

- zmienna, 147
 - CollisionFlags, 249
 - GUIStyle, 235
 - prywatna, 205
 - Touch, 367
 - Vector3, 252
- zmienne
 - kontrolera postaci, 247
 - prywatne, 150
 - publiczne, 150

Ż

- źródła dźwięku, 347

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**

Unity

to zaawansowane środowisko do tworzenia gier 3D, prezentacji i animacji zarówno na urządzenia mobilne, jak i stacjonarne. Wokół tego silnika skupiona jest ogromna społeczność, prezentująca produkcje o różnym stopniu zaawansowania. Jeżeli Twoim marzeniem jest stworzenie gry według własnego scenariusza, jeżeli chcesz sprawdzić się w roli projektanta gier wideo, to trafisz na doskonałą książkę, w całości poświęconą silnikowi Unity!

Ta książka pozwoli Ci poznać zintegrowane środowisko (IDE) oraz dostosować je do własnych potrzeb. Nauczysz się tworzyć obiekty gry i wykonywać na nich operacje, stosować shadery i tekstury oraz korzystać z narzędzi do tworzenia terenu i środowiska Twojej gry. Kolejne rozdziały to interesujące informacje na temat oświetlenia, kamer oraz efektów dźwiękowych. Na sam koniec dowiesz się, jak dopracować stworzoną grę, wdrożyć ją oraz zmodyfikować tak, żeby działała na urządzeniach mobilnych. Dzięki lekturze stworzysz gry: *Gauntlet Runner*, *Captain Blaster*, *Chaos Ball*, *Amazing Racer*. Przekonaj się, jakie możliwości kryje silnik Unity!

Dzięki tej książce:

- poznasz środowisko Unity
- zbudujesz środowisko gry
- wykorzystasz skrypty
- oświetlisz scenę i użyjesz kamer
- stworzysz swoją pierwszą grę

**Zrealizuj swoje marzenie
o stworzeniu własnej gry!**



26751

numer katalogowy

księgarnia internetowa



<http://helion.pl>

zamówienia telefoniczne



0 801 339900



0 601 339900

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

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

Informatyka w najlepszym wydaniu

sięgnij po **WIĘCEJ**



KOD KORZYŚCI

ISBN 978-83-283-0017-0



cena: 69,00 zł