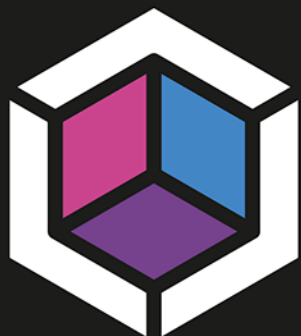
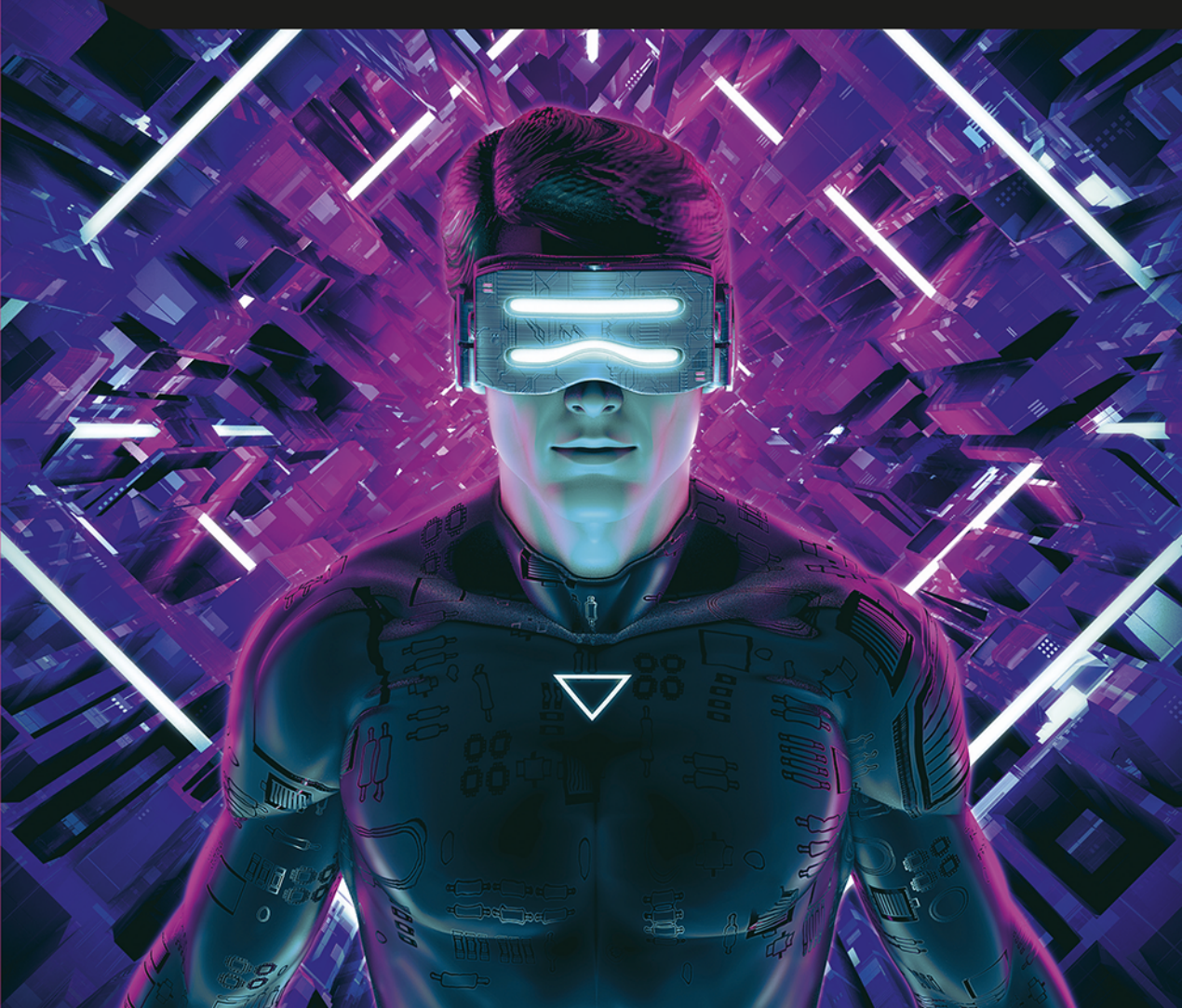


Mike Geig



unity

Przewodnik projektanta gier // **Wydanie III**



Tytuł oryginału: Sams Teach Yourself® Unity 2018 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: 978-83-283-5786-0

Authorized translation from the English language edition, entitled
UNITY 2018 GAME DEVELOPMENT IN 24 HOURS, SAMS TEACH YOURSELF, 3rd Edition
by GEIG, MIKE; published by Pearson Education, Inc, publishing as Sams Publishing.
Copyright © 2018 by Pearson Education

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 © 2020.

Microsoft® Windows®, and Microsoft Office® are registered trademarks
of the Microsoft Corporation in the U.S.A. and other countries. This book
is not sponsored or endorsed by or affiliated with the Microsoft Corporation.

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 Helion SA 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 Helion SA nie ponoszą również żadnej odpowiedzialności
za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Helion SA

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/unipp3>

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	28
Okno dialogowe Project	28
Interfejs Unity	30
Panel Project	32
Panel Hierarchy	34
Panel Inspector	35
Panel Scene	37
Panel Game	38
Wyróżnienie — pasek narzędziowy	40
Poruszanie się po panelu Scene w Unity	41
Narzędzie Hand	41
Tryb Flythrough	42
Podsumowanie	44
Pytania i odpowiedzi	44
Warsztaty	44
Quiz	45
Odpowiedzi	45
Ćwiczenie	45
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	51
Transformacje	51
Translacja	53
Rotacja	55
Skalowanie	56

Ryzyko związane z transformacjami	57
Położenie ikon pomocniczych transformacji	58
Transformacja a obiekty zagnieżdżone	59
Podsumowanie	60
Pytania i odpowiedzi	60
Warsztaty	61
Quiz	61
Odpowiedzi	61
Ćwiczenie	61
LEKCJA 3. Modele, materiały i tekstury	63
Podstawy modeli	64
Wbudowane obiekty 3D	65
Importowanie modeli	66
Modele i sklep Asset Store	67
Tekstury, shadery i materiały	69
Tekstury	69
Shadery	70
Materiały	71
Powracamy do shaderów	72
Podsumowanie	74
Pytania i odpowiedzi	75
Warsztaty	75
Quiz	75
Odpowiedzi	75
Ćwiczenie	76
LEKCJA 4. Teren i środowisko	79
Generowanie terenu	80
Dodanie terenu do projektu	80
Rzeźbienie mapy wysokości	82
Narzędzia do rzeźbienia terenu oferowane przez Unity	84
Tekstury terenu	87
Importowanie zasobów terenu	88
Teksturowanie terenu	89
Generowanie drzew i trawy	92
Malowanie drzewami	92
Malowanie trawą	94
Ustawienia terenu	97

Kontroler postaci	100
Dodanie kontrolera postaci	100
Podsumowanie	101
Pytania i odpowiedzi	101
Warsztaty	101
Quiz	102
Odpowiedzi	102
Ćwiczenie	102
LEKCJA 5. Światła i kamery	103
Światło	104
Wypalanie kontra czas rzeczywisty	104
Światło punktowe	105
Reflektor	107
Światło kierunkowe	108
Tworzenie światła z dowolnego obiektu	109
Aureola	110
Cookies	111
Kamera	113
Anatomia kamery	113
Wiele kamer	114
Podział ekranu oraz funkcja PiP	115
Warstwy	116
Praca z warstwami	118
Użycie warstw	119
Podsumowanie	123
Pytania i odpowiedzi	123
Warsztaty	123
Quiz	124
Odpowiedzi	124
Ćwiczenie	124
LEKCJA 6. Pierwsza gra — Amazing Racer	127
Faza projektowania	128
Koncepcja	128
Reguły	128
Wymagania	129
Budowanie świata gry	130
Rzeźbienie terenu	131
Dodanie elementów do środowiska	131

Mgła	132
Symulacja nieba i horyzontu	133
Kontroler postaci	134
Gamifikacja	135
Dodanie obiektów kontrolujących grę	136
Dodanie skryptów	138
Połączenie skryptów	139
Testowanie gry	141
Podsumowanie	142
Pytania i odpowiedzi	142
Warsztaty	143
Quiz	143
Odpowiedzi	143
Ćwiczenie	143
LEKCJA 7. Skrypty — część 1.	145
Skrypty	146
Tworzenie skryptów	146
Dołączanie skryptu	148
Anatomia prostego skryptu	149
Zmienne	151
Tworzenie zmiennej	152
Zasięg zmiennej	152
Zmienne publiczne i prywatne	154
Operatory	155
Operatory arytmetyczne	155
Operatory przypisania	156
Operatory równości	156
Operatory logiczne	157
Konstrukcje warunkowe	157
Konstrukcja if	158
Konstrukcja if-else	159
Konstrukcja if-else if	160
Iteracja	161
Pętla while	161
Pętla for	162
Podsumowanie	163
Pytania i odpowiedzi	163

Warsztaty	163
Quiz	163
Odpowiedzi	164
Ćwiczenie	164
LEKCJA 8. Skrypty — część 2.	165
Metody	166
Anatomia metody	166
Tworzenie metody	168
Użycie metod	170
Dane wejściowe	172
Podstawy danych wejściowych	172
Skrypty przeznaczone do obsługi danych wejściowych	174
Dane wejściowe z określonych klawiszy	174
Dane wejściowe myszy	176
Uzyskanie dostępu do komponentów lokalnych	177
Używanie metody GetComponent()	177
Uzyskanie dostępu do transformacji	178
Uzyskanie dostępu do innych obiektów	179
Wyszukanie innych obiektów	179
Modyfikacja komponentów obiektu	182
Podsumowanie	183
Pytania i odpowiedzi	183
Warsztaty	184
Quiz	184
Odpowiedzi	184
Ćwiczenie	184
LEKCJA 9. Kolizje	187
Bryły sztywne	188
Kolizje	190
Komponent Collider	190
Materiały fizyczne	192
Wyzwalacze	194
Raycasting	195
Podsumowanie	197
Pytania i odpowiedzi	198
Warsztaty	199
Quiz	199
Odpowiedzi	199
Ćwiczenie	199

LEKCJA 10. Druga gra — Chaos Ball	201
Faza projektowania	202
Koncepcja	202
Reguły	202
Wymagania	202
Arena	203
Utworzenie areny	203
Teksturowanie	204
Materiał zapewniający rewelacyjne odbijanie się obiektu	205
Zakończenie prac nad areną	206
Elementy gry	206
Gracz	207
Kula chaos ball	207
Kolorowe kule	210
Obiekty kontrolne	211
Cele	211
Kontroler gry	212
Usprawnienie gry	214
Podsumowanie	214
Pytania i odpowiedzi	215
Warsztaty	215
Quiz	215
Odpowiedzi	215
Ćwiczenie	215
 LEKCJA 11. Prefabrykaty	 217
Podstawy prefabrykatów	218
Terminologia związana z prefabrykatami	218
Struktura prefabrykatu	219
Praca z prefabrykatami	220
Dodanie do sceny egzemplarza prefabrykatu	223
Dziedziczenie	224
Tworzenie egzemplarza prefabrykatu w kodzie	227
Podsumowanie	227
Pytania i odpowiedzi	228
Warsztaty	228
Quiz	228
Odpowiedzi	228
Ćwiczenie	229

LEKCJA 12. Narzędzia do tworzenia gier 2D	231
Podstawy tworzenia gier 2D	232
Widok sceny dwuwymiarowej	233
Kamera ortogonalna	234
Dodawanie sprite'ów	235
Importowanie sprite'a	236
Typ sprite'a	236
Wielkość zaimportowanego sprite'a	238
Kolejność wyświetlania	239
Warstwa kolejności sortowania	239
Kolejność na warstwie	241
Fizyka 2D	242
Komponent RigidBody 2D	242
Komponent Collider w grze 2D	242
Podsumowanie	244
Pytania i odpowiedzi	244
Warsztaty	244
Quiz	244
Odpowiedzi	245
Ćwiczenie	245
 LEKCJA 13. Mapa kafelków 2D	 247
Podstawy mapy kafelków	248
Utworzenie mapy kafelków	248
Siatka	250
Paleta	250
Okno Tile Palette	251
Kafelki	252
Konfigurowanie sprite'ów	253
Tworzenie kafelków	253
Malowanie kafelkami	255
Dostosowanie palety do własnych potrzeb	258
Mapa kafelków i fizyka	259
Komponenty Collider i mapa kafelków	259
Używanie komponentu Composite Collider 2D	260
Podsumowanie	261
Pytania i odpowiedzi	261
Warsztaty	262
Quiz	262
Odpowiedzi	262
Ćwiczenie	262

LEKCJA 14.	Interfejs użytkownika	263
	Podstawy interfejsu użytkownika	264
	Płótno	264
	Komponent Rect Transform	265
	Punkt zaczepienia	266
	Dodatkowe komponenty płótna	270
	Elementy interfejsu użytkownika	270
	Obrazy	270
	Tekst	272
	Przyciski	273
	Tryby generowania płótna	277
	Tryb Screen Space — Overlay	278
	Tryb Screen Space — Camera	278
	Tryb World Space	279
	Podsumowanie	280
	Pytania i odpowiedzi	280
	Warsztaty	280
	Quiz	281
	Odpowiedzi	281
	Ćwiczenia	281
LEKCJA 15.	Trzecia gra — Captain Blaster	285
	Faza projektowania	286
	Koncepcja	286
	Reguły	286
	Wymagania	286
	Świat	286
	Kamera	287
	Tło	288
	Elementy gry	289
	Gracz	289
	Meteory	291
	Pociski	292
	Wyzwalacze	293
	Interfejs użytkownika	293
	Obiekty kontrolne	294
	Kontroler gry	295
	Skrypt meteoru	296
	Tworzenie meteorów	297
	Skrypt wyzwalacza	298

Skrypt ShipControl	299
Skrypt pocisku	301
Usprawnienie gry	302
Podsumowanie	303
Pytania i odpowiedzi	303
Warsztaty	303
Quiz	303
Odpowiedzi	304
Ćwiczenie	304
LEKCJA 16. Systemy cząsteczek	305
Systemy cząsteczek	306
Cząsteczki	306
Systemy cząsteczek w Unity	306
Kontrolki systemu cząsteczek	306
Moduły systemu cząsteczek	308
Moduł domyślny	308
Moduł Emission	311
Moduł Shape	312
Moduł Velocity over Lifetime	312
Moduł Limit Velocity over Lifetime	312
Moduł Inherit Velocity	312
Moduł Force over Lifetime	313
Moduł Color over Lifetime	313
Moduł Color by Speed	314
Moduł Size over Lifetime	314
Moduł Size by Speed	314
Moduł Rotation over Lifetime	315
Moduł Rotation by Speed	315
Moduł External Forces	315
Moduł Noise	315
Moduł Collision	315
Moduł Triggers	319
Moduł Sub Emitter	319
Moduł Texture Sheet	320
Moduł Lights	320
Moduł Trails	320
Moduł Custom Data	321
Moduł Renderer	321
Edytor krzywych	322
Podsumowanie	323

Pytania i odpowiedzi	324
Warsztaty	324
Quiz	324
Odpowiedzi	325
Ćwiczenie	325
LEKCJA 17. Animacje	327
Podstawy animacji	328
Przygotowanie modelu	328
Animacja	329
Rodzaje animacji	329
Animacja 2D	329
Utworzenie animacji	331
Narzędzia animacji	333
Okno Animation	333
Utworzenie nowej animacji	335
Tryb nagrywania	337
Edytor krzywych	339
Podsumowanie	340
Pytania i odpowiedzi	341
Warsztaty	341
Quiz	341
Odpowiedzi	341
Ćwiczenie	341
LEKCJA 18. Animator	343
Podstawy zasobu Animator	344
Rigging raz jeszcze	345
Import modelu	345
Konfigurowanie zasobu	346
Przygotowanie riggingu	347
Przygotowanie animacji	349
Utworzenie animatora	355
Panel Animator	357
Animacja Idle	357
Parametry	359
Stan i tzw. drzewo Blend Tree	360
Przejścia	362
Obsługa animacji za pomocą skryptów	363
Podsumowanie	363
Pytania i odpowiedzi	364

Warsztaty	365
Quiz	365
Odpowiedzi	365
Ćwiczenie	365
LEKCJA 19. Oś czasu	367
Podstawy osi czasu	368
Anatomia osi czasu	368
Utworzenie osi czasu	369
Praca z osią czasu	370
Okno Timeline	370
Ścieżki osi czasu	371
Klipy osi czasu	374
Nie tylko prosta kontrola	376
Łączenie klipów na ścieżce	376
Skrypty wykorzystujące oś czasu	378
Podsumowanie	379
Pytania i odpowiedzi	379
Warsztaty	380
Quiz	380
Odpowiedzi	380
Ćwiczenie	380
LEKCJA 20. Czwarta gra — Gauntlet Runner	381
Faza projektowania	382
Koncepcja	382
Reguły	382
Wymagania	382
Świat	383
Scena	383
Droga	384
Przewijanie drogi	385
Elementy gry	385
Dodatkowa energia	386
Przeszkody	387
Strefa wyzwalacza	388
Obiekt gracza	388
Obiekty kontrolne	390
Skrypt strefy wyzwalacza	390
Skrypt obiektu kontrolnego gry	391
Skrypt obiektu gracza	393

Skrypty obiektów dodatkowej energii i przeszkód	394
Skrypt SpawnScript	395
Zebranie wszystkiego w całość	396
Usprawnienie gry	397
Podsumowanie	398
Pytania i odpowiedzi	398
Warsztaty	398
Quiz	398
Odpowiedzi	398
Ćwiczenie	399
LEKCJA 21. Dźwięk	401
Podstawy dźwięku	402
Składniki dźwięku	402
Dźwięk 2D i 3D	403
Źródła dźwięku	403
Import klipów audio	405
Testowanie dźwięku w panelu Scene	406
Dźwięk 3D	407
Dźwięk 2D	408
Użycie dźwięku za pomocą skryptów	409
Rozpoczęcie i zatrzymanie odtwarzania dźwięku	409
Zmiana klipu audio	411
Mikser dźwięku	411
Utworzenie miksera dźwięku	411
Przekazywanie dźwięku do miksera	411
Podsumowanie	413
Pytania i odpowiedzi	413
Warsztaty	413
Quiz	414
Odpowiedzi	414
Ćwiczenie	414
LEKCJA 22. Programowanie na platformach mobilnych	417
Przygotowanie do programowania na platformach mobilnych	418
Konfiguracja środowiska	418
Aplikacja Unity Remote	419
Przyśpieszeniomierz	420
Programowanie dla przyśpieszeniomierza	422
Użycie przyśpieszeniomierza	423
Dane wejściowe pochodzące z ekranu dotykowego	424

Podsumowanie	426
Pytania i odpowiedzi	427
Warsztaty	427
Quiz	427
Odpowiedzi	427
Ćwiczenie	428
LEKCJA 23. Dopracowanie i wdrożenie	429
Zarządzanie scenami	430
Ustalanie kolejności scen	430
Przełączanie scen	432
Zachowywanie danych i obiektów	432
Zachowywanie obiektów	433
Zachowywanie danych	433
Ustawienia Unity Player	436
Ustawienia niezależne od platformy	437
Ustawienia dla poszczególnych platform	437
Kompilacja gry	438
Okno Build Settings	439
Okno Game Settings	440
Podsumowanie	442
Pytania i odpowiedzi	442
Warsztaty	442
Quiz	443
Odpowiedzi	443
Ćwiczenie	443
LEKCJA 24. Zakończenie	445
Osiągnięcia	446
19 lekcji	446
4 pełne gry	447
Ponad 50 scen	448
Co dalej?	448
Buduj gry	448
Współpracuj z innymi	449
Pisz o tym	449
Dostępne zasoby	449
Podsumowanie	450
Pytania i odpowiedzi	450

Warsztaty	450
Quiz	450
Odpowiedzi	450
Ćwiczenie	450
Skorowidz	453

Lekcja 10.

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 6., 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 żółtych 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 znika, a oświetlenie narożnika zostaje wyłączone.
- ▶ Wszystkie obiekty w grze mają doskonałe właściwości odbijania się (przy zderzeniu nie tracą impetu).
- ▶ Żadna kula (ani gracz) nie może opuścić areny.

Wymagania

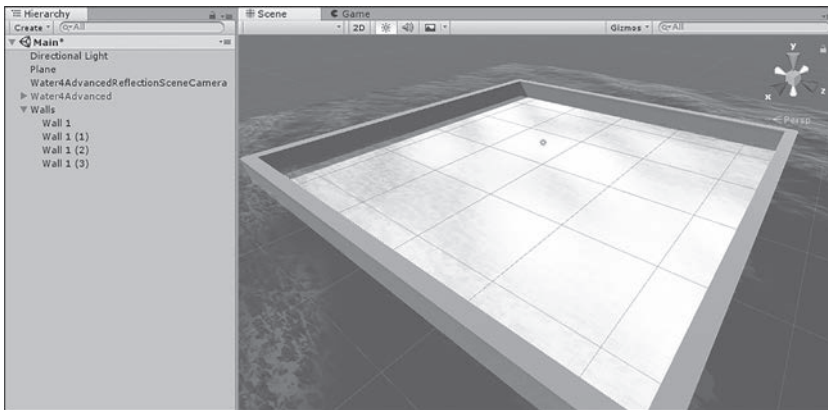
Wymagania dla tworzonej gry są proste. To nie będzie gra z rozbudowaną grafiką, natomiast zdecydowanie oparta na skryptach i interakcjach. Wymagania dla gry *Chaos Ball* przedstawiają się następująco.

- ▶ 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 żółtych *chaos ball*. 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 Visual Studio.

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 10.1, arena jest dość prosta.



RYSUNEK 10.1.
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*.
2. Wybierz opcję menu *Assets/Import Package*, a następnie zaznacz pakiety *Characters* i *Environment*.
3. Za pomocą opcji menu *GameObject/3D/Plane* dodaj płaszczyznę do sceny. Tę płaszczyznę umieść w położeniu (0, 0, 0) i przeskaluj (5, 1, 5).
4. Usuń obiekt *Main Camera*.
5. Do sceny dodaj sześcian. Umieść go w położeniu (-25, 1.5, 0) i przeskaluj (1.5, 3, 51). Zwróć uwagę, jak zmodyfikowany sześcian stał się jedną ścianą muru otaczającego arenę. Nazwę sześcianu zmień na *Wall 1*.
6. W katalogu *Scenes* zapisz tę scenę pod nazwą *Main*.

Wskazówka
Wskazówka**Konsolidacja obiektów**

Być może zastanawiasz się, dlaczego utworzyłeś mur tylko dla jednej strony, choć powinieneś dla wszystkich. Idea polega na tym, aby uniknąć zbędnego powielania 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 chaos ball i kolorowych. W ten sposób powinieneś się przekonać, jak odrobina planowania może pozwolić na zaoszczędzenie dużej ilości czasu. Warto w tym miejscu dodać, że cały proces może być jeszcze łatwiejszy dzięki użyciu prefabrykatów. Skoro ktoś (nie chcę używać nazwisk) nie przedstawił prefabrykatów (a robi to dopiero w następnej lekcji), teraz musisz zastosować przedstawione tutaj rozwiązanie.

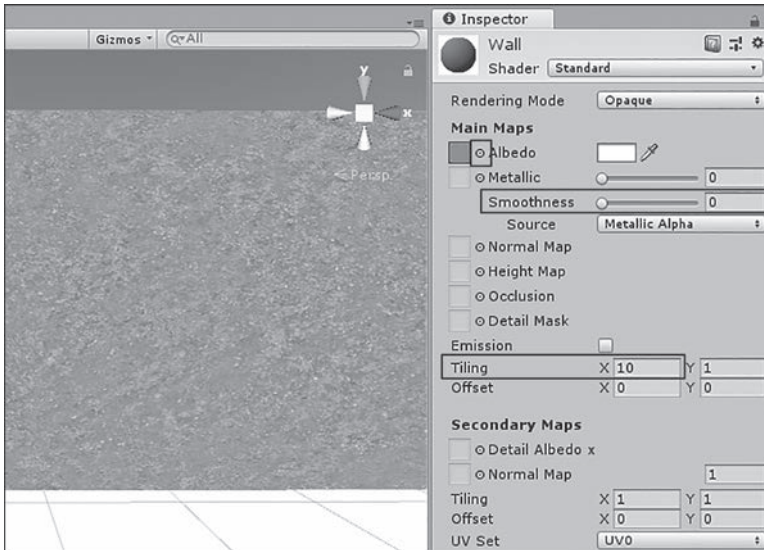
Teksturowanie

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! Pracę zacznij jednak od wykonania przedstawionych tutaj kroków.

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 *Wall*.
2. W panelu Inspector nałóż teksturę *Sand Albedo* na materiał muru. Możesz to zrobić przez przeciągnięcie materiału na właściwość Albedo lub kliknięcie ikony kółka obok słowa Albedo w panelu Inspector (patrz rysunek 10.2).
3. Suwak Smoothness przeciągnij do wartości 0.
4. Właściwości *Tiling* wzdłuż osi X ustaw wartość 10.
5. W panelu Scene kliknij i przeciągnij materiał na utworzony wcześniej obiekt muru.

Teraz przejdź do teksturowania podłoża. Unity dostarcza doskonałe shadery dla wody i dlatego też jeden z nich będzie tutaj wykorzystany.

1. W panelu Project przejdź do katalogu *Standard Assets/Environment/Water/Water4Prefabs*. Przeciągnij na scenę zasób o nazwie *Water4Advanced*.
2. Umieść wodę centralnie w położeniu (0, 0.5, 0).

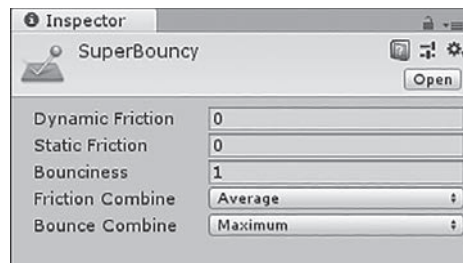


RYSUNEK 10.2.
Dodanie
materiału Wall

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.

1. Prawym przyciskiem myszy kliknij katalog *Materials* i wybierz opcję *Create/Physical Material*. Materiałowi nadaj nazwę *SuperBouncy*.
2. Ustaw dla materiału właściwości pokazane na rysunku 10.3. Ogólnie rzecz biorąc, konieczne jest zapewnienie 100% sprężystości, aby obiekt poruszał się z tą samą szybkością.



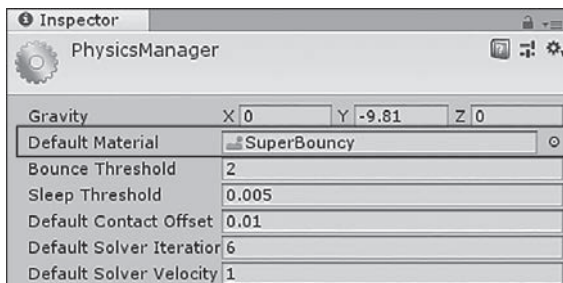
RYSUNEK 10.3.
Ustawienia
materiału
SuperBouncy

W tym momencie przygotowany materiał fizyczny można umieścić bezpośrednio na komponencie *Collider* ściany. Jednak problem polega na tym, że ten materiał musi zostać nałożony na wszystkie ściany, kule i obiekt przedstawiający gracza.

W zasadzie wszystko to, co w tej grze podlega kolizjom, będzie wymagało zdefiniowanego przed chwilą materiału *SuperBouncy*. Dlatego też możesz go nałożyć jako materiał domyślny dla wszystkich komponentów *Collider*. Do tego celu wykorzystaj menu *Physics Settings* i wykonaj wymienione tutaj kroki.

1. Wybierz opcję menu *Edit/Project Settings/Physics*. W panelu Inspector zostanie wyświetlone menu *Physics Manager*, tak jak pokazałem na rysunku 10.4.

RYСУNEK 10.4.
Menu *Physics Manager*



2. Materiał *SuperBouncy* wybierz w *Default Material*.

To menu jest również miejscem, w którym modyfikuje się podstawy fizyki obiektów, czyli kolizje, grawitację itd. W tym momencie powinieneś oprzeć się pokusie zmiany tych ustawień, aby nadać graczowi jakieś potężne możliwości, i pozostawić rzeczywistość taką, jaka jest.

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 (25, 1.5, 0).
2. Ponownie powiel ścianę, umieść ją w położeniu (0, 1.5, 25) i zastosuj rotację (0, 90, 0).
3. Powiel ścianę utworzoną w poprzednim kroku (tę obróconą), a następnie umieść ją w położeniu (0, 1.5, -25).
4. Utwórz pusty obiekt gry o nazwie *Walls*. Jego pozycję określ jako (0, 0, 0). Zgrupuj cztery utworzone ściany w tym nowym obiekcie.

Arena powinna teraz zawierać mur składający się z wszystkich czterech ścian i pozbawiony luk (patrz rysunek 10.1).

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. 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. Kontroler *FPSController* przeciągnij z katalogu *Assets/Characters/FirstPersonCharacter/Prefabs* na scenę.
2. Umieść kontroler w położeniu (0, 1, 0).
3. Rozwiń obiekt *FPSController* w panelu Hierarchy i odszukaj obiekt potomny *FirstPersonCharacter* (ma dołączoną kamerę).
4. Po zaznaczeniu obiektu *FirstPersonCharacter* umieść go 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. W panelu Hierarchy kliknij paletkę i przeciągnij na kontroler *First Person*. W ten sposób paletka zostanie zagnieżdżona w kontrolerze.
3. Następnie zmień położenie paletki na (0, 0, 0.1) i zastosuj rotację (0, 0, 0). Paletka znajdzie się teraz w niewielkiej odległości przed kontrolerem.
4. Zdefiniuj kolor paletki przez utworzenie nowego materiału (*nie fizycznego*) o nazwie *BumperColor*. Właściwości *Albedo* nowego materiału przypisz dowolną wartość, a następnie przeciągnij materiał na paletkę.

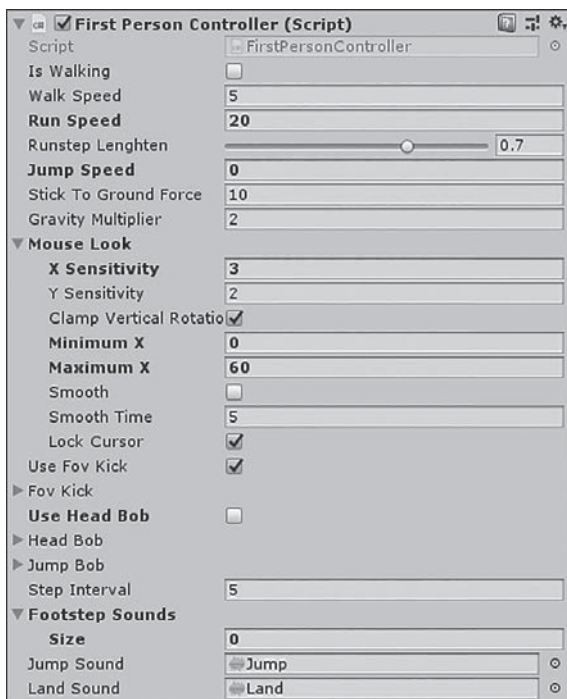
Ostatnim krokiem jest dostosowanie ustawień domyślnych *FPSController*, aby były bardziej odpowiednie dla tej gry. Zastosuj ustawienia pokazane na rysunku 10.5. Zmienione ustawienia różnią się od domyślnych tym, że zostały pogrubione.

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.

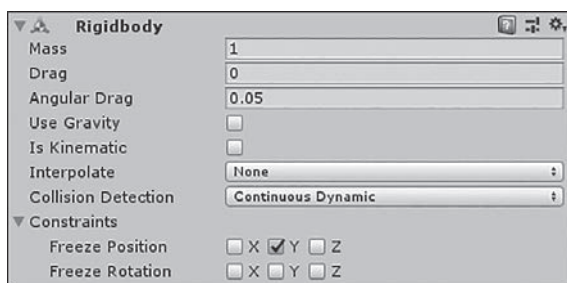
1. Dodaj kulę do sceny. Następnie zmień nazwę kuli na *Chaos*, umieść ją w położeniu (12, 2, 12) i przeskaluj (0.5, 0.5, 0.5).
2. Utwórz nowy materiał (*niefizyczny*) dla kuli i nadaj mu nazwę *ChaosBall*. We właściwości *Albedo* materiału wybierz kolor jasnożółty. Nowo utworzony materiał przeciągnij na kulę.

RYSUNEK 10.5.
Ustawienia
FPSController

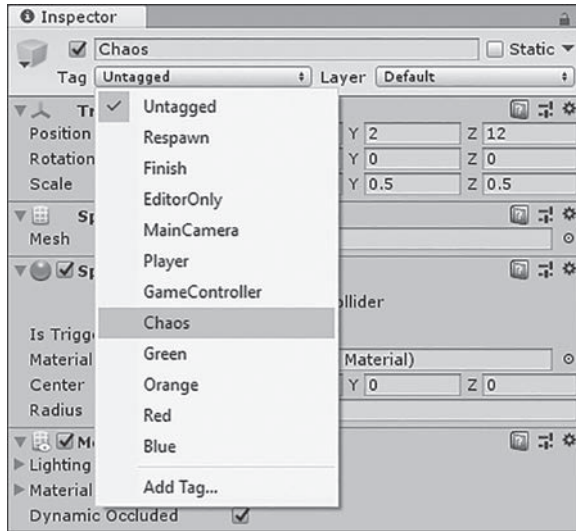


- Do kuli dodaj komponent *Rigidbody*. Jak pokazałem na rysunku 10.6, usuń zaznaczenie przy *Use Gravity*. Z rozwijanego menu właściwości *Collision Detection* wybierz opcję *Continuous Dynamic*. Z kolei we właściwości *Constraints* zamroź położenie Y — nie chcesz, aby kula mogła poruszać się w górę oraz w dół.

RYSUNEK 10.6.
Ustawienia
komponentu
Rigidbody kuli
Chaos Ball



- Przejdź do menedżera Tag Manager (wybierz opcję *Edit/Project Settings/Tags & Layers*), rozwiń sekcję *Tags*, klikając strzałkę znajdującą się obok nazwy sekcji, a następnie 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 10.7).



RYSUNEK 10.7.
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*. Skrypt przenieś do katalogu *Scripts*. Pełny kod skryptu przedstawiono w listingu 10.1.

Listing 10.1. Skrypt *VelocityScript.cs*

```
using UnityEngine;

public class VelocityScript : MonoBehaviour
{
    public float startSpeed = 50f;

    void Start () {
        Rigidbody rigidBody = GetComponent<Rigidbody> ();
        rigidBody.velocity = new Vector3 (startSpeed, 0, startSpeed);
    }
}
```

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. Rozrzuc 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. Na koniec utwórz pusty obiekt gry o nazwie *Chaos Balls*, umieść go w położeniu (0, 0, 0) i dodaj do niego kule, aby zachować porządek w panelu Hierarchy.

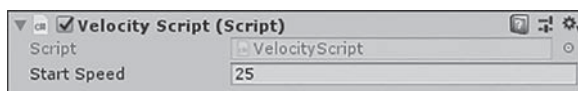
Kolorowe kule

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 Ball* 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 *BlueBall* i ustaw mu kolor niebieski, dokładnie w ten sam sposób, jak to zrobiłeś dla kuli *chaos ball*. Następnie utwórz materiały *RedBall*, *GreenBall* oraz *OrangeBall* i ustaw im odpowiednie kolory (czerwony, zielony i pomarańczowy). Kliknij materiał *BlueBall* i przeciągnij na kulę.
3. Do kuli dodaj komponent *Rigidbody*. Usuń zaznaczenie przy *Use Gravity*. Z rozwijanego menu właściwości *Collision Detection* wybierz opcję *Continuous Dynamic*. We właściwości *Constraints* zamroź położenie Y.
4. 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 10.7).
5. Do kuli dołącz skrypt *VelocityScript*. W panelu Inspector odszukaj komponent *Velocity Script (Script)* i zmień wartość jego właściwości *Start Speed* na 25 (patrz rysunek 10.8). Ta zmiana spowoduje, że kolorowa kula będzie na początku poruszała się znacznie wolniej niż *chaos ball*.

RYСУNEK 10.8.
Zmiana
właściwości
Start Speed



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 kulę i nowym kulom nadaj nazwy odpowiadające ich kolorom, czyli *Red Ball*, *Orange Ball* i *Green Ball*.
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

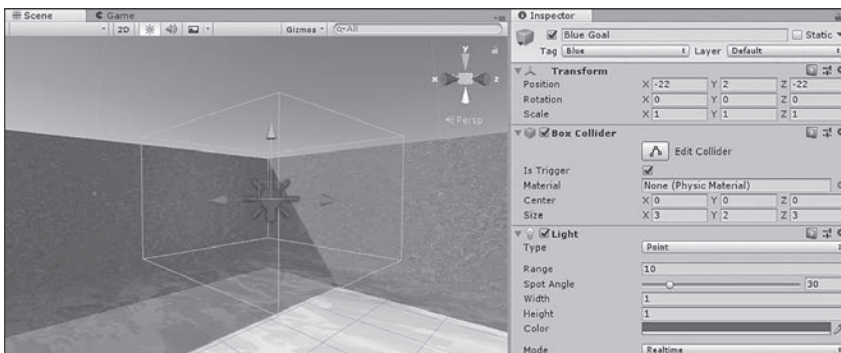
Po przygotowaniu wszystkich niezbędnych elementów możemy przystąpić do gamifikacji. Tym razem zamienimy je w dostarczającą rozrywki grę. W tym celu konieczne jest utworzenie czterech narożników, skryptów oraz kontrolera gry. Dopiero wtedy będziemy mieli gotową grę.

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 kula będzie zniszczona, 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 *Blue Goal*, przypisz tag o nazwie *Blue* i następnie umieść obiekt w położeniu (-22, 2, -22).
2. Do obiektu dodaj komponent *Box Collider* i ustaw jego właściwość *Is Trigger*. Wielkość dodanego komponentu zmień na (3, 2, 3).
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 10.9). Intensywność światła ustaw na 3, zaś właściwości *Indirect Multiplier* przypisz wartość 0.



RYСУNEK 10.9.
Zdefiniowany narożnik w kolorze niebieskim

Kolejnym krokiem jest utworzenie skryptu o nazwie *GoalScript* i dołączenie go do niebieskiego narożnika. Zawartość skryptu przedstawiono w listingu 10.2.

Listing 10.2. Skrypt GoalScript.cs

```
using UnityEngine;

public class GoalScript : MonoBehaviour
{
    public bool isSolved = false;

    void OnTriggerEnter (Collider collider)
    {
        GameObject collidedWith = collider.gameObject;
        if (collidedWith.tag == gameObject.tag)
        {
            isSolved = true;
            GetComponent<Light>().enabled = false;
            Destroy (collidedWith);
        }
    }
}
```

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 usunięta ze sceny, a cel oznaczony jako osiągnięty.

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

1. Powiel narożnik *Blue Goal* i powstałym narożnikom nadaj nazwy odpowiadające kolorom, czyli *Red Goal*, *Green Goal* i *Orange Goal*.
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 (22, 2, -22), (22, 2, 22) i (-22, 2, 22).
5. Umieść narożniki w nowym, pustym obiekcie gry o nazwie *Goals*.

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 *Game Manager*.

2. Utwórz skrypt o nazwie *GameManager* i umieść w nim kod przedstawiony w listingu 10.3. Gotowy skrypt dołącz do kontrolera gry.

Listing 10.3. Skrypt GameManagerScript

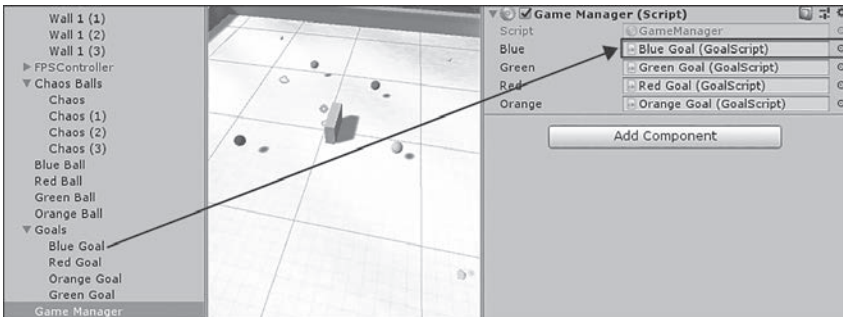
```
using UnityEngine;

public class GameManager : MonoBehaviour
{
    public GoalScript blue, green, red, orange;
    private bool isGameOver = true;

    void Update ()
    {
        // Jeżeli wszystkie cztery cele zostaną osiągnięte, następuje koniec gry.
        isGameOver = blue.isSolved && green.isSolved && red.isSolved &&
            ↪orange.isSolved;
    }

    void OnGUI()
    {
        if(isGameOver)
        {
            Rect rect = new Rect (Screen.width / 2 - 100, Screen.height /
                ↪2 - 50, 200, 75);
            GUI.Box (rect, "Koniec gry");
            Rect rect2 = new Rect (Screen.width / 2 - 30, Screen.height /
                ↪2 - 25, 60, 50);
            GUI.Label (rect2, "Dobra robota!");
        }
    }
}
```

1. Po zaznaczeniu kontrolera gry kliknij i przeciągnij poszczególne narożniki (cele) na odpowiadające im właściwości w komponencie *Game Manager (Script)* (patrz rysunek 10.10).



RYСУNEK 10.10.
Dodanie celów do kontrolera gry

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*.

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 poprawienia 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 zagrań w grę oraz wyszukania aspektów wartych usprawnienia.

Pytania i odpowiedzi

Pytanie: Dlaczego do wykrywania kolizji kul wykorzystujemy wartość **Continuous Dynamic** 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: Dlaczego utworzyłem tag **Chaos** i nigdy go później nie użyłem?

Odpowiedź: Ten tag przyda Ci się, gdy będziesz wprowadzać usprawnienia w grze zaproponowane w ćwiczeniu na końcu lekcji.

Warsztaty

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

Quiz

1. Jak gracz może przegrać w utworzonej tutaj grze?
2. Które osie pozycji obiektów kul zostały zamrożone?
3. Osiągnięcie celu jest sprawdzane w metodzie `OnTriggerEnter()`.
W wymienionej metodzie sprawdzamy, czy obiekt jest oczekiwaną kulą. Prawda czy fałsz?
4. Dlaczego w grze pominięto pewne podstawowe funkcje?

Odpowiedzi

1. To jest podchwytliwe pytanie. Gracz nie może przegrać.
2. Jest to oś Y.
3. Prawda.
4. 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 grze *Amazing Racer* utworzonej w lekcji 6. 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.
- ▶ Spróbuj przykryć wodę terenem, płaszczyzną lub innymi obiektami gry umieszczonymi wokół areny.

Skorowidz

A

Amazing Racer, 127, 447
 dodanie skryptów, 138
 faza projektowania, 128
 gamifikacja, 135
 horyzont, 133
 kontroler postaci, 134
 mgła, 132
 niebo, 133
 obiekty kontrolujące grę, 136
 połączenie skryptów, 139
 rzeźbienie terenu, 131
 świat gry, 130
 testowanie, 141
 wymagania, 129
animacja, 327
 Idle, 349, 351, 357
 WalkForwardStraight, 352
 WalkForwardTurn, 353
animacje
 2D, 329
 arkusza sprite'ów, 238
 czas, 336
 narzędzia, 333
 tworzenie, 331
 ustawienia, 350
animator, 343
 parametry, 359
 tworzenie, 355
aplikacja Unity Remote, 419
arena, 203
Asset Store, 67
Audio Listener, 115
aureola, 110

B

bezpieczeństwo danych, 436
billboardy, 95
blok metody, 167
błędy typu TerrainData..., 92
bryły sztywne, 188
budowanie świata gry, 130

C

Captain Blaster, 285, 448
 elementy gry, 289
 faza projektowania, 286
 obiekty kontrolne, 294
 świat, 286
 usprawnienie gry, 302
Chaos Ball, 201, 447
 arena, 203
 elementy gry, 206
 faza projektowania, 202
 gracz, 207
 kolorowe kule, 210
 kontroler gry, 212
 kula chaos ball, 207
 obiekty kontrolne, 211
 usprawnienie gry, 214
cookies, 111
cząsteczki, 306
 własne, 307

D

dane wejściowe, 172
 ekran dotykowy, 424
 klawiatura, 174
 mysza, 176
dodawanie
 elementów do środowiska, 131
 kontrolera postaci, 100
 sprite'ów, 235
dokładność kolizji, 260
dostęp
 do komponentów lokalnych, 177
 do obiektów, 179
 do transformacji, 178
dostosowanie palety, 258
droga, 384
drzewa, 92
drzewo Blend Tree, 360
dziedziczenie, 219, 224
dźwięk, 401
 2D, 403, 408
 3D, 403, 407

dźwięk

- Audio Listener, 402
- mikser, 411, 412
- odtwarzanie, 409
- priorytet, 405
- skrypty, 409
- testowanie, 406
- źródła, 403

E

edytor, 28

- krzywych, 322, 339

efekt cząsteczek, 308

egzemplarz, 218

elementy

- interfejsu użytkownika, 270
- okna Animation, 334

energia, 386

F

faza projektowania, 128, 202, 286, 382

Flare Layer, 115

formaty map wysokości, 84

funkcja PiP, 115

G

gamifikacja, 135

Gauntlet Runner, 381, 448

- elementy gry, 385

- faza projektowania, 382

- obiekty kontrolne, 390

- świat, 383

- usprawnienie gry, 397

generowanie

- drzew, 92

- płótna, 277

- terenu, 80

- trawy, 92

głębokość, 244

gra

- Amazing Racer, 127, 447

- Captain Blaster, 285, 448

- Chaos Ball, 201, 447

- Gauntlet Runner, 381, 448

gracz, 207, 289, 388

gry 2D, 231

H

horyzont, 133

I

IDE, Integrated Development Environment, 148

ikony pomocnicze, 40

- rotacji, 56

- sceny, 38

- transformacji, 58

import

- klipów audio, 405

- modelu, 66, 345

- sprite'a, 236

- zasobów terenu, 88

instalacja

- komponentów, 27

- Unity, 26

instrukcja warunkowa, 157

- if, 158

- if-else, 159

- if-else if, 160

interfejs

- Unity, 30

- użytkownika, UI, 263, 293

- elementy, 270

iteracja, 161

K

kafelki, 252, 255

- niestandardowe, 253

kamera, 113, 287

- ortogonalna, 234

- właściwości, 113

klipy

- audio, 405, 411

- osi czasu, 374

kody klawiszy, 175

kolejność

- na warstwie, 241

- wyświetlania, 239

kolizje, 187, 190

komentarze, 151

kompilacja, 430

- gry, 438

komponent

- Audio Listener, 115, 402
- Audio Source, 404
- Button, 274
- Collider, 190, 242, 259
- Collider 2D, 244
- Composite Collider 2D, 260
- Image, 271
- Playable Director, 370
- Rect Transform, 265
- Rigidbody, 188
- Rigidbody 2D, 242
- Text, 273

komponenty

- płótna, 270
- transformacji, 52, 54

konfigurowanie sprite'ów, 253

konsola, 151

konsolidacja obiektów, 204

kontroler

- gry, 129, 212, 295
- postaci, 100, 134
- postaci FPSController, 100

kontrolki

- konta, 41
- narzędzia Hand, 42
- panelu Game, 41
- przyciągania, 44
- systemu cząsteczek, 306
- trybu Flythrough, 43

krzywe, 322, 339

kula chaos ball, 207

Ł

łączenie

- klipów, 376
- skryptów, 139

M

malowanie

- drzewami, 92
- kafelkami, 255
- trawą, 94

mapa

- kafelków, 259
- kafelków 2D, 247
- wysokości, 82, 83

materiały, 71

- fizyczne, 192

Maximize on Play, 40

menu Layer, 119

meteory, 291, 297

metoda GetComponent(), 177

metody, 166

- blok, 167
- lista parametrów, 167
- nazwa, 166
- sygnatura, 167
- tworzenie, 168
- typ wartości zwrotnej, 167
- użycie, 170

mgła, 132

mikser dźwięku, 411, 412

model Ethan, 346

modele, 63

- importowanie, 66
- zasoby graficzne, 69

moduł

- Collision, 315
- Color by Speed, 314
- Color over Lifetime, 313
- Custom Data, 321
- Emission, 311
- External Forces, 315
- Force over Lifetime, 313
- Inherit Velocity, 312
- Lights, 320
- Limit Velocity over Lifetime, 312
- Noise, 315
- Renderer, 321
- Rotation by Speed, 315
- Rotation over Lifetime, 315
- Shape, 312
- Size by Speed, 314
- Size over Lifetime, 314
- Sub Emitter, 319
- Texture Sheet, 320
- Trails, 320
- Triggers, 319
- Velocity over Lifetime, 312

moduły systemu cząsteczek, 308

modyfikacja komponentów obiektu, 182

Mute audio, 40

N

- narzędzia
 - animacji, 333
 - do rzeźbienia terenu, 84
 - do tworzenia gier 2D, 231
 - transformacji, 41, 54
 - w oknie Tile Palette, 256
- narzędzie
 - Hand, 41
 - Paint Details, 95
 - Paint Trees, 93
 - Terrain Settings, 97
- nazwa
 - metody, 166
 - skryptu, 146
- niebo, 133

O

- obiekt
 - gry, 47, 51
 - Grid, 250
 - PlayerPrefs, 435
 - tekstowy, 272
- obiekty
 - kontrolne, 211, 294, 390
 - kontrolujące grę, 136
 - wbudowane 3D, 65
 - zagnieżdżone, 59
- obliczanie wysokości, 84
- obrazy, 270
- obsługa
 - animacji, 363
 - danych wejściowych, 174
- odtworzenie dźwięku, 409
- okno
 - Animation, 333
 - Audio Mixer, 412
 - Build Settings, 439
 - Edit Grass Texture, 96
 - Game Settings, 440
 - Project, 28
 - Tile Palette, 251, 256
 - Timeline, 370
- operatory
 - arytmetyczne, 155
 - logiczne, 157, 158
 - przypisania, 156
 - równości, 156

- oś czasu, 367
 - klipy, 374
 - okno Timeline, 370
 - skrypty, 378
 - ścieżki, 371
 - tryb podglądu, 371
 - tworzenie, 369
 - zablokowanie, 371
- oś rotacji, 55
- oświetlenie sceny, 38

P

- paleta, 250
- panel
 - Animator, 357
 - Game, 38
 - Hierarchy, 34
 - Inspector, 35, 36, 137
 - Project, 32
 - Scene, 37, 41
- pasek narzędziowy, 40
- pętla
 - for, 162
 - while, 161
- pierwsza gra, 127
- platformy mobilne, 417
- plótno, 264
 - tryby generowania, 277
- pobieranie Unity, 26
- pociski, 292, 301
- początek układu współrzędnych, 49
- podgląd siatki, 346
- podział ekranu, 115
- postać, 100, 134
- prefabrykaty, 217
 - struktura, 219
- priorytet dźwięku, 405
- programowanie
 - dla przyśpieszoniomierza, 422
 - konfiguracja środowiska, 418
 - platformy mobilne, 417
- przejścia, 362
- przenoszenie zasobów, 33
- przeszkody, 387
- przewijanie drogi, 385
- przycisk, 273
 - Dalej, 40
 - Pauza, 39
 - punktu zaczepienia, 269

- Start, 39
- Stats, 40
 - wyciszenia dźwięku, 406
- przyśpieszeniometer, 420
- punkt
 - rozpoczęcia gry, 129
 - zaczepienia, 266

R

- raycasting, 195
- reflektor, 107
- rigging, 345, 347
- rodzaje animacji, 329
- roślinność, 96
- rotacja, 55, 58
- rozdzielczość, 80
- rozpoczęcie gry, 129
- rozwijane menu
 - Aspect, 40
 - Layer, 119
 - Layers, 41
 - wyboru układu, 41
- rozwińnięcie, 70
- rzeźbienie terenu, 84, 131

S

- scena, 35, 383
 - dwuwymiarowa, 233
 - egzemplarz prefabrykatu, 223
 - niewidoczne elementy, 120
 - przełączanie, 432
 - ustalanie kolejności, 430
 - zachowywanie obiektów, 433
 - zarządzanie, 430
- sekcja Layer Collision Matrix, 122
- shadery, 70
 - wspólne właściwości, 73
- siatka, 64, 250
- skalowanie, 56, 58
 - siatki, 67
- sklep Asset Store, 67
- składnia, 153
- składniki dźwięku, 402
- skrypt, 138, 145, 165
 - ShipControl, 299
 - SpawnScript, 395
- skrypty
 - dołączanie, 148
 - meteoru, 296

- nazwy, 146
- obiektu
 - dodatkowej energii, 394
 - gracza, 393
 - kontrolnego gry, 391
 - przeszkody, 394
 - obsługa danych wejściowych, 174
 - pocisku, 301
 - sekcja klasy, 150
 - sekcja Using, 150
 - tworzenie, 146
 - wykorzystujące oś czasu, 378
 - wyzwalacza, 298, 390
- Slide, 359
- Slip, 359
- sortowanie, 239
- spłaszczanie terenu, 86
- sprite, 236
- strefa wyzwalacza, 388
- struktura
 - prefabrykatu, 219
 - metody, 167
- system 2D, 3D, 49
- systemy cząsteczek, 305
 - moduł
 - Collision, 315
 - Color by Speed, 314
 - Color over Lifetime, 313
 - Custom Data, 321
 - domyślny, 308
 - Emission, 311
 - External Forces, 315
 - Force over Lifetime, 313
 - Inherit Velocity, 312
 - Lights, 320
 - Limit Velocity over Lifetime, 312
 - Noise, 315
 - Renderer, 321
 - Rotation by Speed, 315
 - Rotation over Lifetime, 315
 - Shape, 312
 - Size by Speed, 314
 - Size over Lifetime, 314
 - Sub Emitter, 319
 - Texture Sheet, 320
 - Trails, 320
 - Triggers, 319
 - Velocity over Lifetime, 312

Ś

ścieżki osi czasu, 371
 światło, 104
 kierunkowe, 108
 punktowe, 105

T

tekst, 272
 tekstury, 69, 70, 204
 terenu, 87, 89
 teren, 79
 dodanie do projektu, 80
 generowanie, 80
 importowanie zasobów, 88
 narzędzia, 84
 spłaszczanie, 86
 tekstury, 69, 70, 204
 tworzenie tekstur, 91
 ustawienia, 97, 98
 wielkość, 81
 testowanie
 dźwięku, 406
 gry, 129, 141
 tło, 288
 transformacje, 51
 a obiekty zagnieżdżone, 59
 ikony pomocnicze, 58
 rotacja, 55, 58
 skalowanie, 56, 58
 translacja, 53, 58
 translacja, 53, 58
 trawa, 94, 96
 tryb
 dźwięku, 38
 Flythrough, 42
 nagrywania, 337
 Screen Space — Camera, 278
 Screen Space — Overlay, 278
 World Space, 279
 wyświetlania, 38
 tryby generowania płótna, 277
 tworzenie
 animacji, 331
 animatora, 355
 areny, 203
 egzemplarza, 219
 gier 2D, 231, 232
 kafelków, 253

mapy kafelków, 248
 meteorów, 297
 metody, 168
 miksera dźwięku, 411
 osi czasu, 369
 prefabrykatu, 221
 skryptów, 146
 zmiennej, 152
 typ sprite'a, 236
 typy zmiennych, 153

U

układ współrzędnych, 48, 50
 lokalny, 50
 świata, 50
 Unity Player
 ustawienia, 436
 uruchomienie, 40
 ustawienia
 generowania aureoli, 110
 obiektów drzewa, 99
 roślinności, 99
 rozdzielczości, 81
 terenu, 97, 98
 Unity Player, 436
 wiatru, 99
 użycie
 metod, 170
 przyśpieszeniomierza, 423
 warstw, 119

W

warstwy, 116, 118
 gry, 38
 kolejność, 241
 kolejność sortowania, 239
 warunek, 129
 wdrożenie, 429
 wiatr, 99
 widok 2D/3D, 38
 wielkość
 kamery ortogonalnej, 235
 sprite'a, 238
 terenu, 81
 właściwości
 animacji Idle, 351
 dźwięku 3D, 408
 kamery, 113

- komponentu
 - Audio Source, 404
 - Button, 274
 - Collider, 191
 - Image, 271
 - Rigidbody, 188
 - Text, 273
- materiałów fizycznych, 193
- mgły, 133
- modułu
 - domyślnego, 309
 - Limit Velocity over Lifetime, 313
 - Noise, 316
 - Renderer, 321
 - Texture Sheet, 320
- narzędzia Paint Trees, 93
- osi, 173
- shaderów, 73
- światła, 104
- światła punktowego, 105
- tekstury cookie, 111
- zmiennej typu Touch, 425
- właściwość
 - Culling Mask, 121
 - Normalized View Port Rect, 115
 - On Click (), 273
- włączenie/wyłączenie ikon transformacji, 41
- współrzędne świata, 50
- wstrzymanie, 40
- wybór ikon pomocniczych, 38
- wydajność, 96
- wyłączenie, 40
- wymiary, 48
- wysokość, 84
- wyszukanie obiektów, 179
- wyzwalacze, 194, 293

Z

- zachowywanie
 - danych, 433
 - obiektów, 433
- zagnieżdżanie, 35
- zarządzanie
 - projektem, 33
 - scenami, 430
 - sceną, 36
- zasięg zmiennej, 152
- zmienna typu Touch, 425
- zmiennie, 151
 - prywatne, 154
 - publiczne, 154
 - tworzenie, 152
 - typy, 153
 - zasięg, 152


Ź

- źródła dźwięku, 403

Notatki

PROGRAM PARTNERSKI

— GRUPY 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
Helion 

Za sukces gry komputerowej odpowiada nie tylko wciągająca fabuła. Niezwykle ważne są również strona wizualna, grywalność i poziom trudności — zbyt łatwa gra nie jest dla użytkownika wyzwaniem. Masz już projekt gry, która spełnia te warunki? Zatem potrzebujesz jeszcze tylko dobrego narzędzia do jej zbudowania. Takim narzędziem jest silnik Unity — zaawansowane środowisko do tworzenia gier 3D, prezentacji i animacji dla urządzeń stacjonarnych i mobilnych. Co istotne, wokół Unity jest skupiona ogromna społeczność pasjonatów, gotowa do niesienia pomocy początkującym.

Ta książka jest trzecim, uzupełnionym i zaktualizowanym wydaniem lubianego przewodnika, dzięki któremu błyskawicznie nauczysz się podstaw tworzenia gier w Unity. W ramach 24 lekcji, z których żadna nie powinna Ci zająć więcej niż godzinę, zapoznasz się z podstawami i zaawansowanymi technikami wykorzystywania silnika Unity. Znajdziesz tu przejrzyste instrukcje krok po kroku, wskazówki, praktyczne przykłady oraz ćwiczenia i quizy, dzięki którym utrwalisz nabytą wiedzę. Dowiesz się, jak korzystać z nowo udostępnionych w Unity narzędzi, między innymi do pisania 2D, jak używać maszyny stanów i czym jest nowa oś czasu. Bardzo ważnym elementem książki są lekcje dotyczące dopracowywania szczegółów oraz kompilacji całej gry.

W tej książce między innymi:

- **solidne wprowadzenie do silnika gier i edytora w Unity 2018**
- **tworzenie światów, obiektów i programowanie zachowania postaci**
- **implementacja intuicyjnych graficznych interfejsów użytkownika**
- **sterowanie postacią gracza i symulowanie zjawisk fizycznych w grze**
- **integracja dźwięku z akcją w grze**
- **wykorzystywanie przyśpieszeniomierzy i ekranów dotykowych w urządzeniach mobilnych**

Unity: jaką grę dziś zaprojektujesz?

Mike Geig — z zawodu nauczyciel, z pasji doświadczony projektant gier komputerowych i zapalony gracz. Prowadzi zajęcia w Stark State College oraz w Cleveland Institute of Art. Pracuje w Unity Technologies, gdzie tworzy materiały szkoleniowe. Jest uzdolnionym szkoleniowcem z zakresu projektowania gier, a także autorem kilku książek o tej tematyce.

Helion 



helion.pl



HELION SA
ul. Kościuszki 1c
44-100 Gliwice
tel.: 32 230 98 63
helion@helion.pl

Sprawdź nasze szkolenia!



AKADEMIA IT & BUSINESS

WWW.SZKOLENIA.HELION.PL

KOD KORZYŚCI
Sięgnij po więcej! ▶



ISBN 978-83-283-5786-0



9 788328 357860

INFORMATYKA W NAJLEPSZYM WYDANIU

Cena: 79,00 zł