Mike_Geig

Unity

PRZEWODNIK PROJEKTANTA GIER

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.

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.

Spis treści

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.

Wskazówka

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!

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

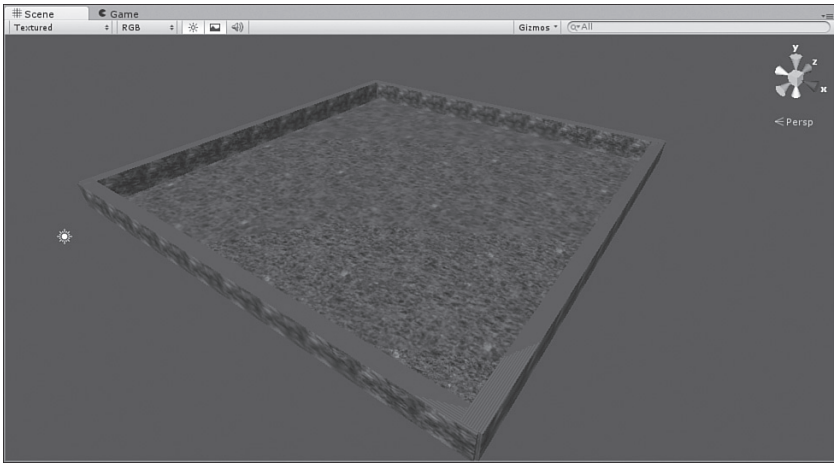
Wymagania dla tworzonej gry są proste. To nie będzie gra z rozbudowaną grafiką, natomiast zdecydowanie oparta na skryptach i interakcjach. Wymagania dla gry *Chaos Ball* przedstawiają się następująco.

- 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.

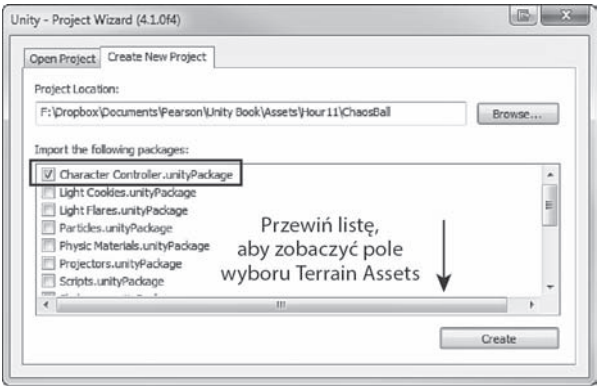


RYSUNEK 11.1. Arena dla tworzonej gry

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 Controler.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

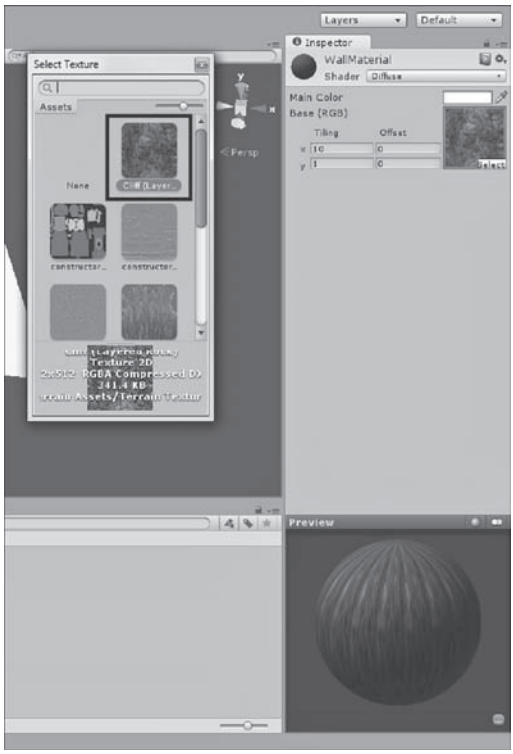
Konsolidacja obiektów

Być może zastanawiasz się, dlaczego utworzyłeś mur tylko dla jednej strony, choć 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 białych i kolorowych. W ten sposób powinieneś 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 teksturowaniem, 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 *Tilling* wzdłuż osi X ustaw wartość 10.



RYSUNEK 11.3. Dodanie tekstury do definiowanego materiału

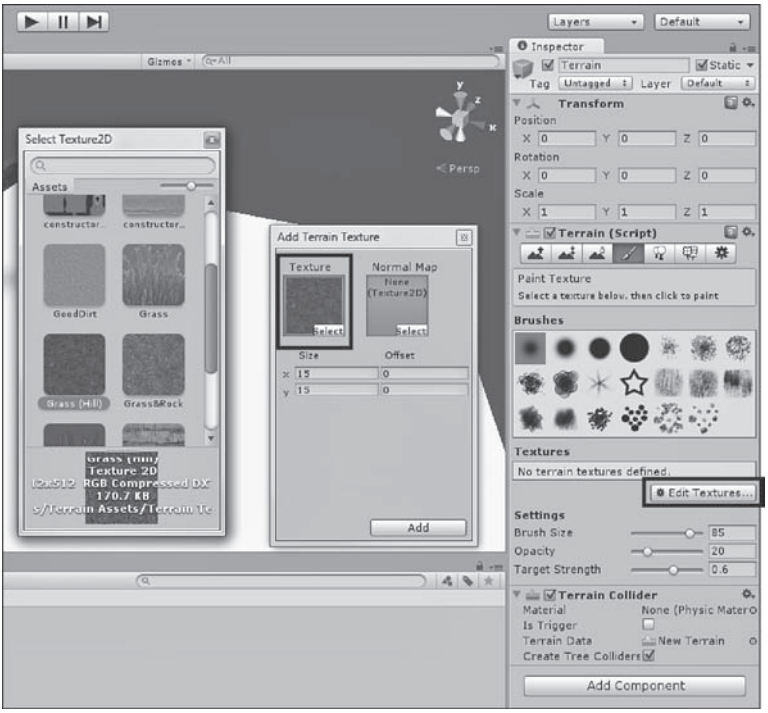
3. W panelu Inspector nałóż teksturę *Cliff (Layered Rock)* na materiał muru (patrz rysunek 11.3).
4. Kliknij i przeciągnij materiał na utworzony wcześniej obiekt muru.

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

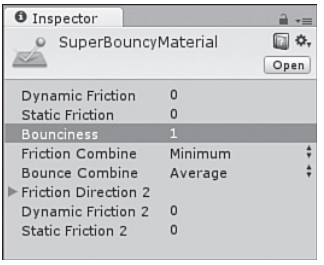
Materiał zapewniający rewelacyjne odbijanie się obiektu

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.

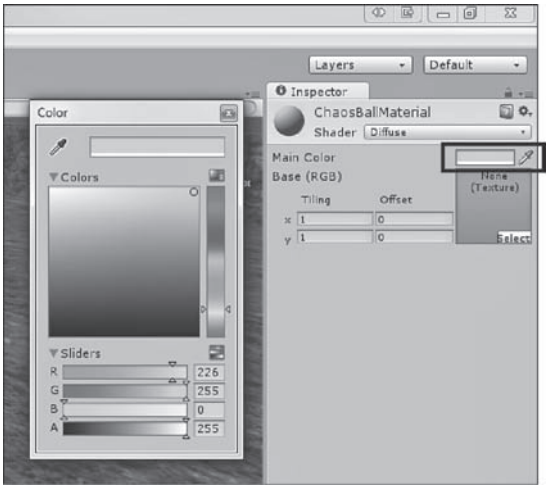
3. W panelu Hierarchy kliknij paletkę i przeciągnij na kontroler *First Person*. W ten sposób paletka zostanie zagnieżdżona w kontrolerze. Następnie zmień położenie paletki na (0, 0, 0.1) i zastosuj rotację (0, 0, 0). Paletka znajdzie się teraz w niewielkiej odległości przed kontrolerem.

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.

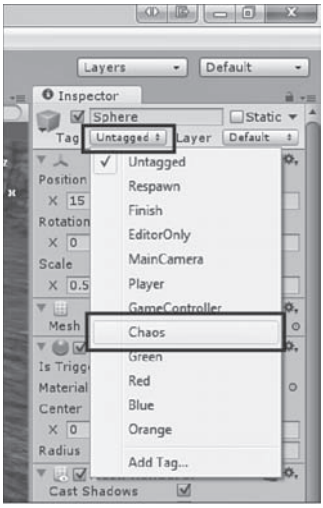
1. Dodaj kulę do sceny. Następnie zmień nazwę kuli na *Chaos*, umieść ją w położeniu (15, 2, 25) i przeskaluj (0.5, 0.5, 0.5).
2. Kliknij materiał *SuperBouncyMaterial* i przeciągnij go na kulę.
3. Utwórz nowy materiał (*niefizyczny*) dla kuli i nadaj mu nazwę *ChaosBallMaterial*. W sekcji koloru materiału 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 we właściwości *Constraints* zamroź położenie Y — nie chcesz, aby kula mogła poruszać się w górę oraz w dół.

5. 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.
6. Zaznacz kulę, a następnie w panelu Inspector zmień jej tag na *Chaos* (patrz rysunek 11.7).



RYSUNEK 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 zwrotną jest liczba z zakresu tworzonego przez dwie liczby będące parametrami.

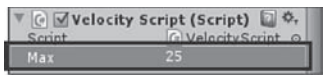
Uruchom scenę i zobacz, jak kula porusza się po arenie. Na tym etapie kula *chaos ball* jest już gotowa. W panelu Hierarchy powiel tę kulę czterokrotnie. Rozrzuć nowo utworzone kule po arenie (upewnij się, że zmieniają jedynie położenie w zakresie osi X i Z), a ponadto dla każdej z nich zastosuj inną 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 odszukaj 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*.



RYSUNEK 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

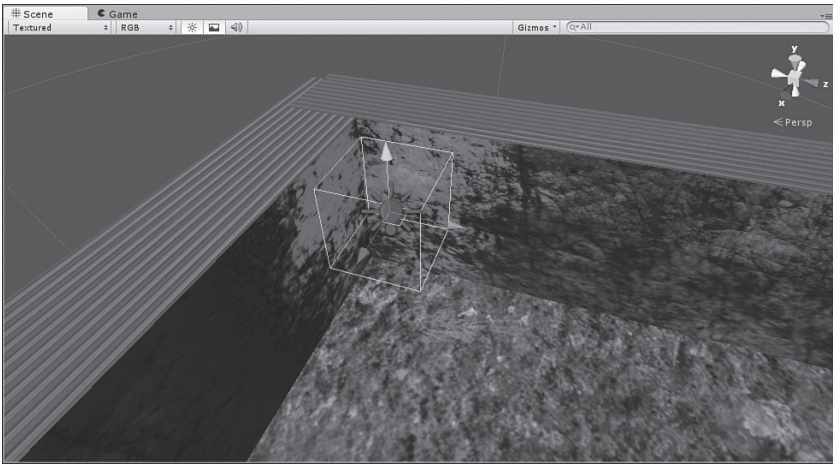
Po przygotowaniu wszystkich niezbędnych elementów możemy przystąpić do gamifikacji. Tym razem zamienimy je w dostarczającą rozrywki grę. W tym celu konieczne jest utworzenie czterech narożników, skryptów oraz kontrolera gry. Dopiero wtedy będziemy mieli gotową grę.

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.

Uwaga

Zmienna prywatna

Jak zapewne zauważyłeś, skrypt *GoalScript* zawiera zmienną prywatną o nazwie `solved` i metodę publiczną `IsSolved()`. Wartością zwrotną 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.

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 *GameControler*.
2. Utwórz skrypt o nazwie *GameControlScript* i umieść w nim kod przedstawiony w listingu 11.3. Gotowy skrypt dołącz do kontrolera gry.

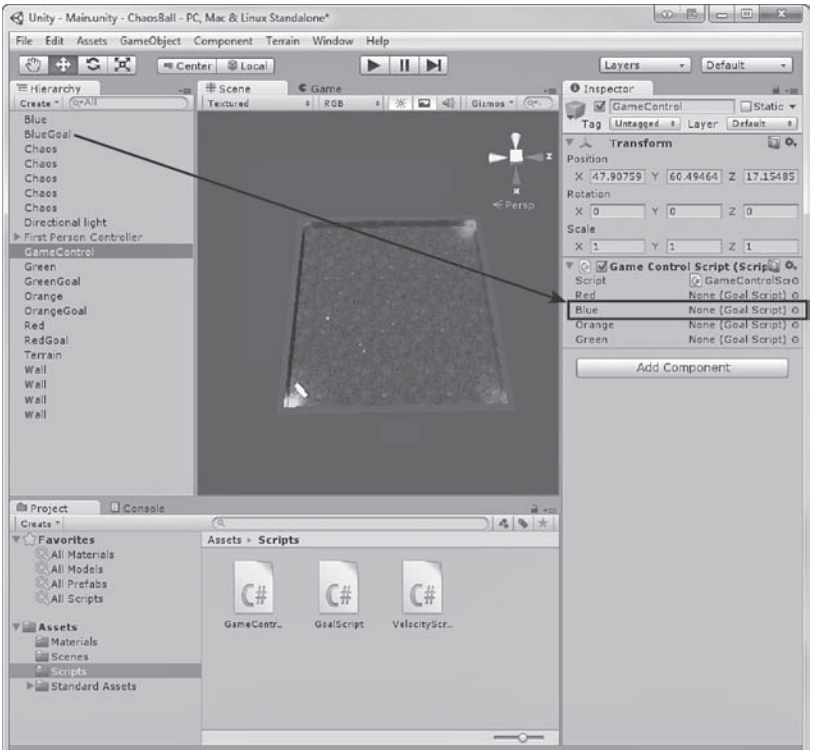
3. Po zaznaczeniu kontrolera gry kliknij i przeciągnij poszczególne narożniki (cele) na odpowiadające im właściwości w komponencie *Game Control Script* (patrz rysunek 11.10).

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 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 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 „przejść" 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

B

C

D

E

F

G

H

I

J

K

L

Ł

M

N

O

P

R

S

Ś

T

U

W

Z

Ź

Notatki

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 trafiłeś 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!

Helion

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

ISBN 978-83-283-0017-0

9 788328 300170

Informatyka w najlepszym wydaniu

cena: 69,00 zł