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.

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.

Spis treści

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.

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 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

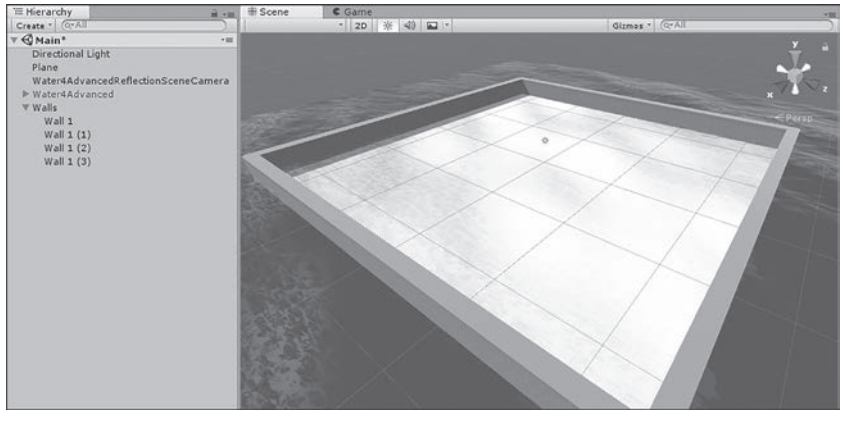
Wymagania dla tworzonej gry są proste. To nie będzie gra z rozbudowaną grafiką, natomiast zdecydowanie oparta na skryptach i interakcjach. Wymagania dla gry *Chaos Ball* przedstawiają się następująco.

- 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

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 zrobi to dopiero w następnej lekcji), teraz musisz zastosować przedstawione tutaj rozwiązanie.

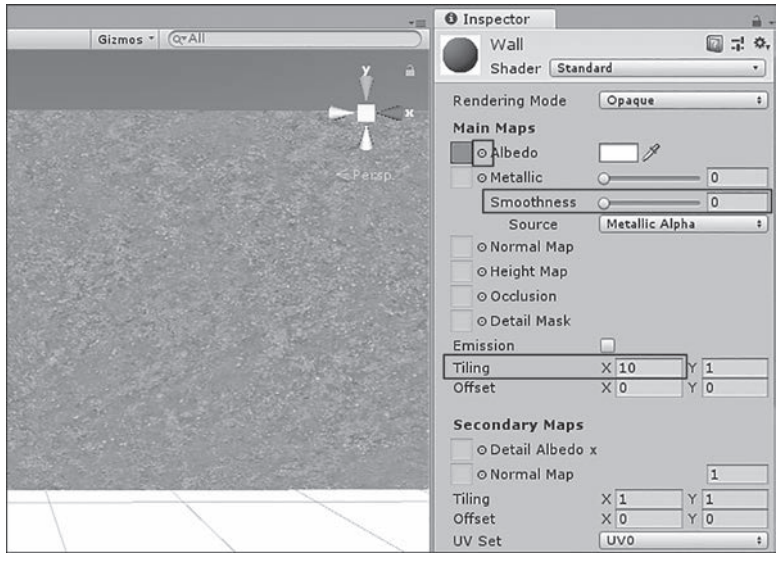
Teksturowanie

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 teksturowaniem, 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 *Tilling* 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 Asstes/Environment/Water/Water4Prefabs*. Przeciągnij na scenę zasób o nazwie *Water4Advanced*.
2. Umieść wodę centralnie w położeniu (0, 0.5, 0).

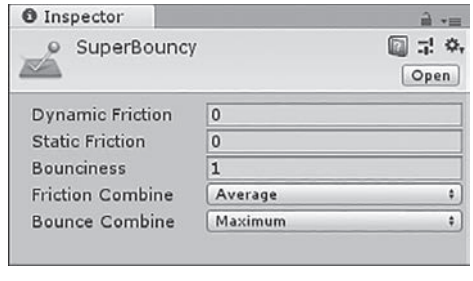


RYSUNEK 10.2. Dodanie materiału Wall

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.



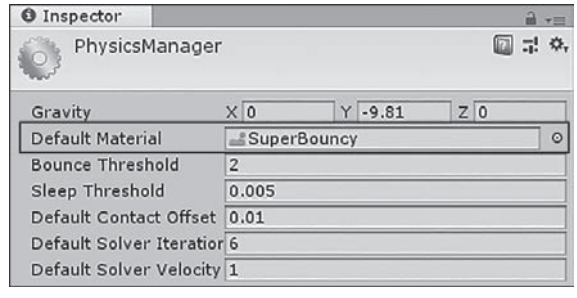
RYSUNEK 10.3. Ustawienia materiału SuperBouncy

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.

RYSUNEK 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.

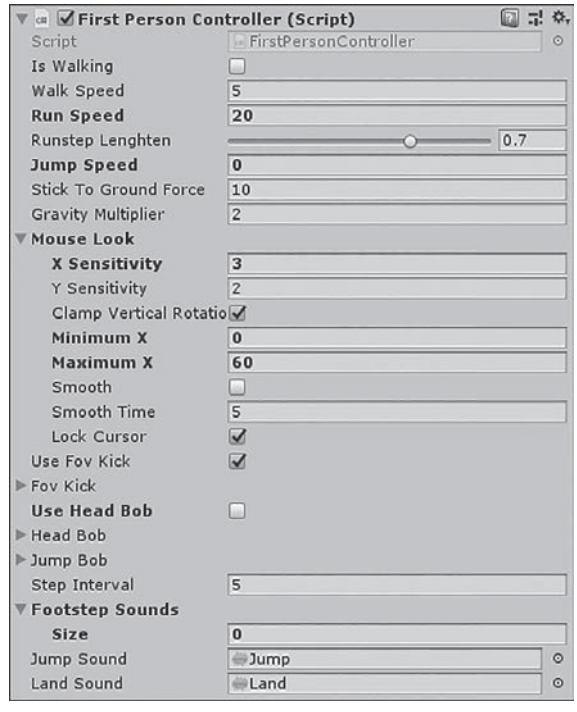
Ostatnim krokiem jest dostosowanie ustawień domyślnych *FPSController*, aby były bardziej odpowiednie dla tej gry. Zastosuj ustawienia pokazane na rysunku 10.5. Zmienione ustawienia różnią się od domyślnych tym, że zostały pogrubione.

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.

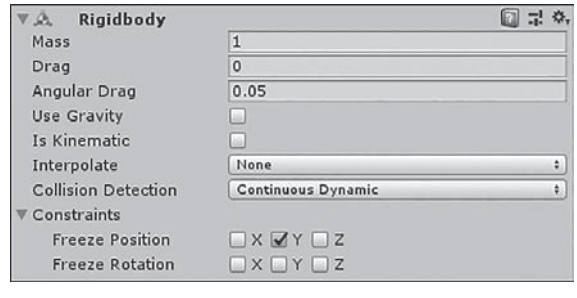
1. Dodaj kulę do sceny. Następnie zmień nazwę kuli na *Chaos*, umieść ją w położeniu (12, 2, 12) i przeskaluj (0.5, 0.5, 0.5).
2. Utwórz nowy materiał (*niefizyczny*) dla kuli i nadaj mu nazwę *ChaosBall*. We właściwości Albedo materiału wybierz kolor jasnożółty. Nowo utworzony materiał przeciągnij na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

RYSUNEK 10.5. Ustawienia FPSController

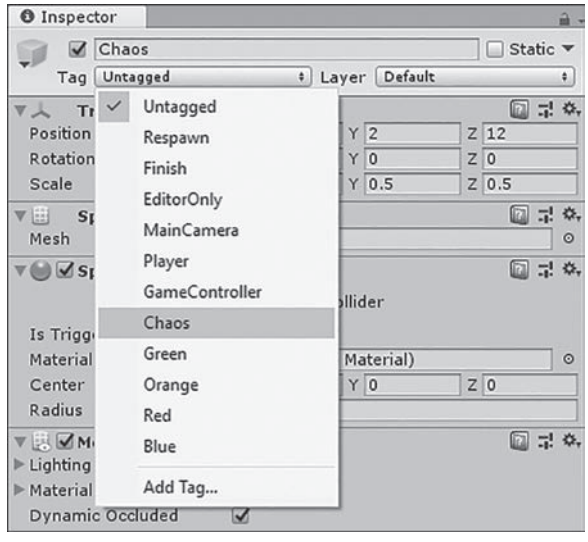


3. Do kuli dodaj komponent *Rigidbody*. Jak pokazałem na rysunku 10.6, usuń zaznaczenie przy *Use Gravity*. Z rozwijanego menu właściwości *Collision Detection* wybierz opcję *Continuous Dynamic*. Z kolei we właściwości *Constraints* zamroź położenie Y — nie chcesz, aby kula mogła poruszać się w górę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

RYSUNEK 10.6. Ustawienia komponentu Rigidbody kuli Chaos Ball



4. 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.
5. 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);
    }
}
```

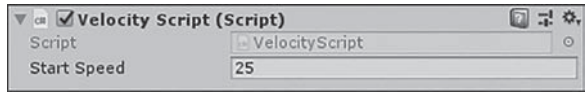
Uruchom scenę i zobacz, jak kula porusza się po arenie. Na tym etapie kula *chaos ball* jest już gotowa. W panelu Hierarchy powiel tę kulę czterokrotnie. Rozrzuć nowo utworzone kule po arenie (upewnij się, że zmieniają jedynie położenie w zakresie osi X i Z), a ponadto dla każdej z nich zastosuj inną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



RYSUNEK 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

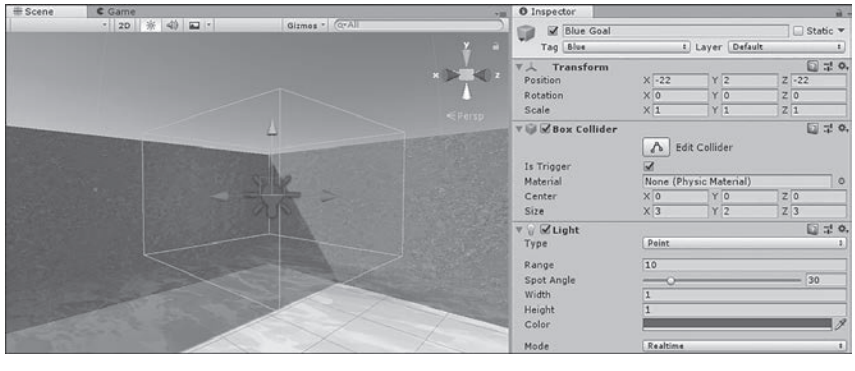
Po przygotowaniu wszystkich niezbędnych elementów możemy przystąpić do gamifikacji. Tym razem zamienimy je w dostarczającą rozrywki grę. W tym celu konieczne jest utworzenie czterech narożników, skryptów oraz kontrolera gry. Dopiero wtedy będziemy mieli gotową grę.

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**.



RYSUNEK 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

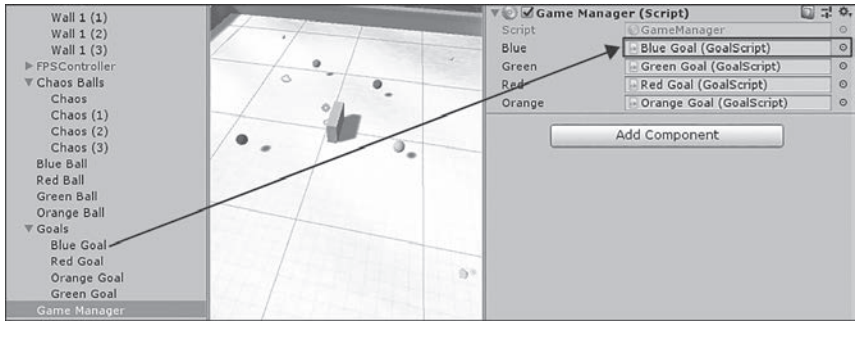
```
using UnityEngine;

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).



RYSUNEK 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 zagrania 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 „przejść" 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

B

C

D

E

F

G

H

I

K

Ł

M

N

O

P

R

S

Ś

T

U

W

Z

Ź

Notatki

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

ISBN 978-83-283-5786-0

9 788328 357860

INFORMATYKA W NAJLEPSZYM WYDANIU

Cena: 79,00 zł