Programowanie gier przy użyciu Unity i C#

Podręcznik
dla całkiem początkujących

—

Casey Hardman

Apress®

Apress®

Casey Hardman

Programowanie gier przy użyciu Unity i C#

Podręcznik dla całkiem początkujących

Przekład: Jakub Niedźwiedź

APN Promise, Warszawa 2020

Programowanie gier przy użyciu Unity i C#. Podręcznik dla całkiem początkujących

First published in English under the title
Game Programming with Unity and C# by Casey Hardman

APN PROMISE SA, ul. Domaniewska 44a, 02-672 Warszawa
tel. +48 22 35 51 600, fax +48 22 35 51 699
e-mail: wydawnictwo@promise.pl

ISBN: 978-83-7541-435-6 (druk), 978-83-7541-436-3 (ebook)

Przekład: Jakub Niedźwiedź
Redakcja: Marek Włodarz
Korekta: Ewa Swędrowska
Skład i łamanie: MAWart Marek Włodarz

Spis treści

O autorze

Casey Hardman jest hobbistycznym programistą gier, który czerpie inspiracje z zapewnianych przez gry możliwości imersji i interaktywności. Skupia się na pracy z silnikiem gier Unity. Od dziecka pielęgnował pasję do gier wideo. We wczesnej młodości te zainteresowania poprowadziły go do świata projektowania i programowania gier. Jest samoukiem, który zdobył doświadczenie w różnych osobistych projektach, niektórych niewielkich, a niektórych całkiem rozbudowanych. Jest stałym współpracownikiem wielu internetowych platform tworzenia gier i spędza zbyt dużo czasu przed komputerem.

O recenzencie technicznym

Robert Lair od ponad 25 lat tworzy zawodowo oprogramowanie i zajmował niemal każde stanowisko związane z różnymi elementami cyklu wytwarzania oprogramowania. Pełnił funkcję prezesa, dyrektora operacyjnego, wiceprezesa ds. rozwoju produktu, dyrektora technicznego, architekta oprogramowania, programisty i menedżera produktu. Uważa, że tworzenie dobrego oprogramowania jest sztuką i jest pasjonatem prawidłowo zaprojektowanego oprogramowania, wydajnych procesów wytwarzania oprogramowania, dokumentacji, ograniczania długu technicznego, organizacji i produktywności. Specjalizuje się w budowaniu gier Unity, a także aplikacji internetowych i mobilnych skupiających się na grywalizacji. Więcej na temat Roberta można dowiedzieć się na jego stronie internetowej www.robertlair.com.

Wstęp

Witamy na początku przygody z programowaniem gier w Unity. Ta książka ma na celu nauczyć, jak programować gry wideo od podstaw, zapewniając przy tym dużą ilość praktycznych doświadczeń. Nie skupia się na realizacji ambitnych projektów i nie zajmuje się wymyślną grafiką. Nauczymy się, jak programować i jak korzystać z silnika Unity. Dokładnie rozumiejąc te podstawowe zagadnienia, można będzie następnie poszerzać swoją wiedzę i tworzyć coraz bardziej skomplikowane i imponujące gry.

Całe oprogramowanie, z którego będziemy korzystać, jest wieloplatformowe. Ta książka będzie się głównie trzymać terminologii i przykładów związanych z systemem Windows, ale można korzystać też z innych typowych systemów operacyjnych, takich jak Mac lub Linux bez zbytnich problemów.

Jeśli chodzi o wymagania systemowe, to dowolny nowoczesny komputer zakupiony w ostatnich 6 latach nie powinien mieć trudności z uruchamianiem oprogramowania, z którego będziemy korzystać. Ponieważ nie będziemy zajmować się grafiką z najwyższej półki, ani przetwarzać długotrwałych algorytmów, tworzone projekty przykładowe powinny działać na większości komputerów. Jeśli ktoś ma obawy, to oficjalne i najbardziej aktualne wymagania systemowe dla Unity można znaleźć na stronie internetowej:

https://unity3d.com/unity/system-requirements

W rozdziałach 1–12 zaczniemy od wprowadzenia podstawowych pojęć samego silnika gier Unity i przygotujemy do działania wszystkie potrzebne narzędzia. Następnie przejdziemy do konkretnych szczegółów programowania i zaczniemy faktycznie sami pisać kod.

W dalszej części książki zajmiemy się po kolei poszczególnymi przykładami gier, tworząc grywalne projekty, które każdy może później samodzielnie rozbudowywać. Tutaj zdobędziemy najwięcej praktycznego doświadczenia. Będziemy implementować faktyczne mechanizmy gier, co jest kwintesencją programowania gier.

Projekt gry 1 „Tor przeszkód” (rozdziały 13–25) będzie widzianym z góry torem przeszkód, po którym gracz przemieszcza swoją postać przy pomocy klawiszy WASD lub klawiszy ze strzałkami, aby unikać zetknięcia się z niebezpieczeństwami w różnej

postaci: patrolującymi i wędrującymi wrogami, przelatującymi pociskami i pułapkami kolcowymi w podłodze. Będziemy ćwiczyć podstawowe ruchy i obroty, konfigurowanie poziomów, pracę z podstawowymi pojęciami Unity, takimi jak prefabrykaty i skrypty oraz przygotowywanie interfejsu użytkownika.

Projekty gry 2 „Tower defense" (rozdziały 26-32) będzie stanowić podstawę prostej gry typu „tower defense" (obrona wieżami), w której gracz umieszcza struktury obronne na polu gry. Wrogowie będą poruszać się z jednej strony pola gry na drugą, a elementy obronne gracza będą próbowały ich odeprzeć. Zbadamy podstawowe algorytmy wytyczania ścieżki (jak wrogowie poruszają się wokół przeszkód) i dalej rozwiniemy podstawowe pojęcia programowania.

Projekt gry 3 „Zabawa fizyką" (rozdziały 33-41) będzie trójwymiarowym placem zabaw z zastosowaniem mechanizmów fizyki, wsparciem dla kamery pierwszoosobowej i trzecioosobowej dla postaci gracza oraz bardziej złożonym poruszaniem się przy pomocy myszy, pozwalającym na skoki i wykorzystującym systemy grawitacji. Zbadamy możliwości fizyki w silniku Unity, począwszy od wykrywania obiektów przez rzutowanie promienia (raycast) do konfigurowania obiektów fizycznych typu Rigidbody.

Kod źródłowy

Towarzyszący książce kod źródłowy dostępny jest w serwisie GitHub pod adresem *https://github.com/Apress/game-programming-unity-csharp*. Znajdziemy tam kod źródłowy wszystkich listingów zawartych w książce.

Część I

Wprowadzenie

Rozdział 1

Instalacja i przygotowanie

Instalacja oprogramowania jest dość prosta – trzeba pobrać program instalacyjny, uruchomić go, wyrazić zgodę na warunki użytkowania, które się pojawią, wybrać miejsce instalacji programu na swoim komputerze i ewentualnie wybrać jakieś opcje dodatkowe. Łatwe, prawda? Nie będę wchodzić w dokładne szczegóły procesu instalacji. Po prostu pokażę, co trzeba zainstalować.

Instalowanie Unity

Unity często wydaje nowe wersje z nowymi funkcjami, poprawkami błędów i drobnymi ulepszeniami. Z tego powodu ostatnio wprowadzili centralne miejsce do instalowania różnych wersji o nazwie Unity Hub. Jest to lekka, niewielka aplikacja, która pozwala zainstalować faktyczny silnik Unity, włącznie ze starszymi jego wersjami. Pozwala też zarządzać wszystkimi wersjami silnika już zainstalowanymi na komputerze i przeglądać wszystkie swoje projekty Unity z jednego miejsca.

Czasami przydaje się zachowanie starszej wersji Unity nawet po zainstalowaniu najnowszej wersji. Możemy chcieć pracować nad starszym projektem na tej samej wersji, z którą zaczynaliśmy nad nim pracę, na wypadek gdyby jakieś nowe funkcje lub zmiany nie były kompatybilne z naszym starym projektem – rzeczy się zmieniają, a czasami nowe elementy psują wcześniejszą funkcjonalność. Czasami starsza funkcjonalność zostaje po prostu przepisana na nowo i nie jest zgodna z nowszym silnikiem. W tych przypadkach można zechcieć pozostać przy starej wersji do zakończenia danego projektu, aby uniknąć niepotrzebnego poświęcania czasu na zmienianie metody robienia czegoś w „nowy sposób".

Zainstalujemy więc najpierw Unity Hub, a następnie możemy instalować sam silnik Unity poprzez Hub. Aby pobrać aplikację Hub, należy skorzystać z następującego łącza

w swojej przeglądarce WWW: *https://unity3d.com/get-unity/download*. Na tej stronie należy kliknąć przycisk „Download Unity Hub" pokazany na rysunku 1–1.

Rysunek 1-1. *Przycisk Download Unity Hub*

Zacznie się pobieranie instalatora Unity Hub. Należy uruchomić instalator i stosować się do wyświetlanych poleceń.

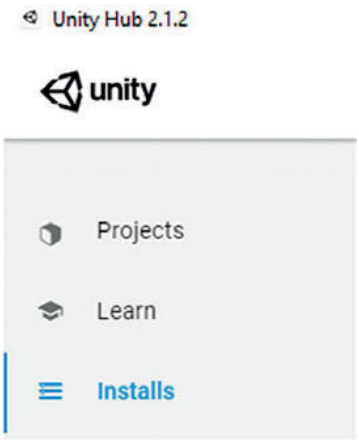
Po zainstalowaniu programu Unity Hub uruchamiamy go. W tym momencie Unity może poprosić o zaakceptowanie licencji, co może obejmować utworzenie konta użytkownika w Unity. Jest to niewielki, jednorazowy krok, a później będziemy zawsze automatycznie logowani na swoje konto. W razie konieczności założenia konta należy zapamiętać swoje hasło i nazwę użytkownika!

Aby zrozumieć, czym jest „licencja", trzeba wiedzieć, że kiedyś istniała darmowa wersja Unity i wersja Pro. Niektóre funkcje były niedostępne w darmowej wersji, a za wersję Pro trzeba było płacić, jeśli chciało się korzystać z tych funkcji – zwłaszcza dotyczyło to sposobów oświetlania sceny trójwymiarowej i bardziej wyrafinowanych efektów. Później po prostu niemal wszystkie funkcje zostały udostępnione w wersji darmowej, natomiast wersje płatne oferowały głównie takie rzeczy, jak wsparcie techniczne, dodatkowe zasoby i narzędzia do współpracy w grupie. Obecnie oferowane są trzy różne „licencje" na korzystanie z tego silnika: Personal, Plus i Pro.

O ile użytkownik lub firma, którą reprezentuje, osiąga przychody mniejsze niż 100000$ za ostatni rok, to może korzystać z licencji Personal na silnik Unity, która jest wolna od opłat. Jeśli ktoś zacznie osiągać korzyści majątkowe ze swoich gier, będzie w końcu musiał zaktualizować swoją licencję do Unity Plus (25$/miesiąc) lub Unity Pro (125$/miesiąc), aby nie naruszać warunków licencji Personal. Nie uprzedzajmy jednak faktów – na razie jesteśmy jedynie hobbistami.

Możemy też zostać poproszeni o wypełnienie krótkiej ankiety. Obejmuje ona kilka pytań dotyczących tego, jak zamierzamy korzystać z silnika, jakie są nasze obszary zainteresowań i inne podobne kwestie.

Po uzyskaniu licencji i założeniu konta, powinniśmy zobaczyć kartę „Installs" (instalacje) z lewej strony okna Unity Hub (zobacz rysunek 1–2).



Rysunek 1-2. *Lewy górny róg okna Unity Hub z wybraną opcją Installs*

Tutaj możemy zobaczyć wszystkie wersje silnika Unity, które zainstalowaliśmy na swoim komputerze. Możemy też instalować nowe wersje – choć zainstalowanie wielu wersji może szybko zapełnić miejsce na dysku twardym, aby więc tego uniknąć, możemy też odinstalować stare, nieużywane wersje. Po wybraniu karty „Installs", klikamy przycisk „Add" (dodaj) w prawym górnym rogu. Pojawi się okienko oferujące listę wersji, które możemy zainstalować. Najwyżej wymienione będzie najnowsze stabilne wydanie. Należy kliknąć to wydanie, aby je zaznaczyć, a następnie kliknąć Next (dalej) w prawym dolnym rogu.

Pojawi się lista do wyboru dodatkowych „modułów", które chcemy zainstalować z silnikiem Unity. Są to dodatkowe funkcje, które można dodać do instalacji i które zajmują dodatkowe miejsce. Można też zawsze dodać je później po normalnej instalacji, jeśli będą nam potrzebne. Najważniejsze są tu moduły wspierające budowanie (Build Support), które pozwalają nam zbudować projekt gry Unity dla różnych systemów operacyjnych, środowisk i sprzętów.

„Budowanie" projektu gry jest procesem zamieniającym projekt Unity (grywalny jedynie poprzez narzędzia silnika Unity) na faktyczną aplikację do uruchamiania gry w danym środowisku docelowym.

Głównymi platformami, dla których możemy budować projekty Unity, są

- PC ze wsparciem dla systemów Windows, Mac i Linux
- Android
- Apple iOS
- WebGL (możliwość uruchamiania gry w przeglądarce WWW)
- Xbox One
- PS4

Jak mówiłem, jeśli będziemy chcieli budować gry dla tych platform (nie będziemy się tym zajmowali w tej książce), zawsze możemy zainstalować odpowiednie moduły później poprzez Hub.

Możemy też wybrać instalację dokumentacji Unity lokalnie jako moduł. Dokumentacja może być niezwykle pomocna. Jest też dostępna online i samemu raczej korzystam z zasobów sieciowych, ale jeśli ktoś planuje korzystać z Unity bez dostępu do sieci, to może chcieć zainstalować dokumentację lokalnie, aby była dostępna przy braku połączenia internetowego.

Na razie możemy po prostu zaznaczyć opcję Build Support (obsługa budowania) dla wykorzystywanego systemu operacyjnego i ewentualnie dokumentację, a wyłączyć zaznaczenie wszystkich pozostałych opcji. Następnie klikamy przycisk Done (gotowe), aby rozpocząć instalację. Instalowana wersja pojawi się jako ramka w głównej części okna, a niewielki pasek nad nią będzie pokazywać postępy instalacji.

Po zakończeniu instalowania pasek ten zniknie i możemy teraz utworzyć projekt Unity w danej wersji silnika – za chwilę to zrobimy i uruchomimy silnik po raz pierwszy. Inną ciekawą funkcją aplikacji Unity Hub jest to, że będzie automatycznie uruchamiać odpowiednią wersję edytora Unity przy otwieraniu projektu (zakładając, że dana wersja jest nadal zainstalowana na komputerze).

Instalowanie edytora kodu

Kodu nie piszemy w tym samym rodzaju oprogramowania, które moglibyśmy wykorzystać do napisania książki albo życiorysu. Edytory kodu są edytorami tekstowymi, które są dostosowane do pisania kodu. Mają opcje wyróżniania specjalnych słów i symboli, wiedzą, jak odpowiednio formatować kod i często są wyposażone w wiele funkcji, które ułatwiają i przyspieszają pisanie kodu i pracę z kodem.

Edytorem wybranym na potrzeby tej książki jest Microsoft Visual Studio Code. Nie należy go mylić z oprogramowaniem Microsoft Visual Studio. Oba są podobnymi produktami (i podobnie nazwanymi) od tej samej firmy, z obu można korzystać za darmo i moglibyśmy korzystać z dowolnego z nich. Oba mogą edytować kod C# i integrują się z Unity.

Visual Studio Code jest lekkim narzędziem międzyplatformowym, które można w dużym stopniu rozszerzać. Ma minimalistyczny interfejs użytkownika i większość jego funkcji jest włączana poprzez instalowanie w samym programie rozszerzeń dodających dalsze funkcjonalności.

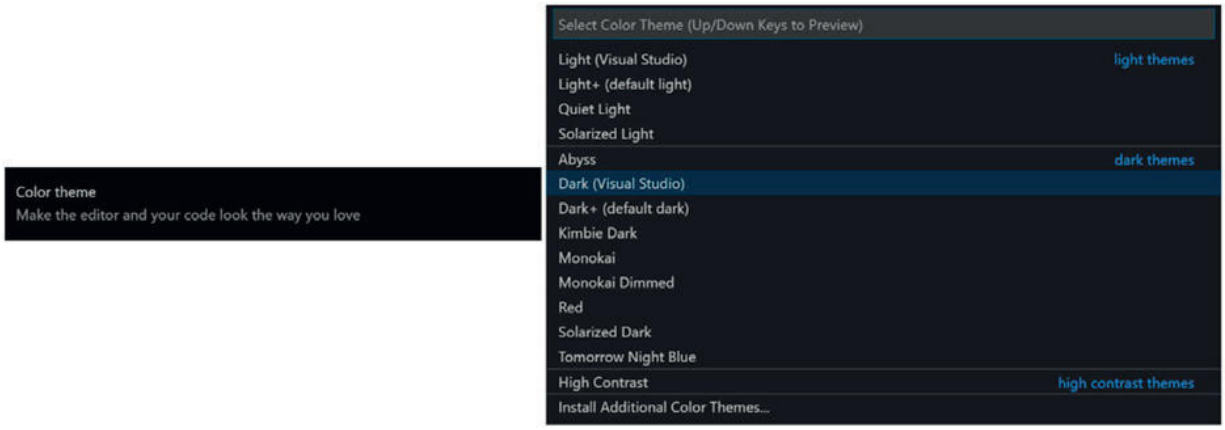
Visual Studio obsługuje systemy Windows i macOS, ale nie Linux. Jest wyposażony w więcej od razu wbudowanych funkcji. Jest bardzo potężnym narzędziem i ma mnóstwo zastosowań, w tym funkcje współpracy w zespołach i inne bardziej zaawansowane funkcje. Ogólnie jest „cięższym" produktem – wyposażonym w więcej funkcji, ale może zużywać więcej pamięci i działać nieco wolniej.

Wybieramy Code, ponieważ uważam, że bardziej nadaje się dla początkujących, gdyż nie przytłacza tak dużą ilością rozbudowanych funkcji, a większość zaawansowanych funkcji Visual Studio i tak nie będzie nam potrzebna.

Aby pobrać program Code, należy skorzystać z poniższego łącza w swojej ulubionej przeglądarce WWW: *https://code.visualstudio.com/download*.

Z tej strony można wybrać odpowiedni przycisk, aby pobrać oprogramowanie dla swojego systemu operacyjnego (Windows, Linux lub Mac). Instalator powinien zacząć się pobierać. Po zakończeniu pobierania należy go uruchomić i stosować się do wyświetlanych instrukcji.

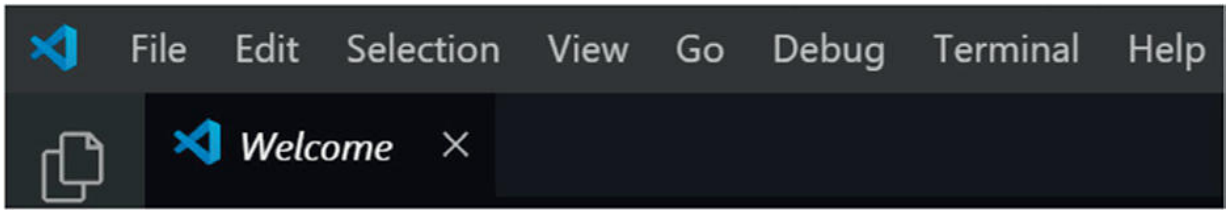
Po uruchomieniu Visual Studio Code po raz pierwszy zobaczymy stronę powitania służącą jako centrum z różnymi łączami i zasobami. Wielu programistów przywiązuje wagę do schematu kolorów wykorzystywanego przez edytor kodu. Ja najbardziej lubię ciemny schemat kolorów. Niektórzy wolą jasny – każdy może wybrać taki, jaki mu pasuje. Można to łatwo zmienić bezpośrednio ze strony powitalnej. Należy kliknąć przycisk „Color theme" (schemat kolorów), aby wyświetlić listę standardowych schematów kolorów, z której można wybrać dowolny schemat (zobacz rysunek 1–3).



Rysunek 1-3. *Przycisk Color theme na stronie powitania (z lewej strony) i okienko pojawiające się po jego kliknięciu (z prawej strony)*

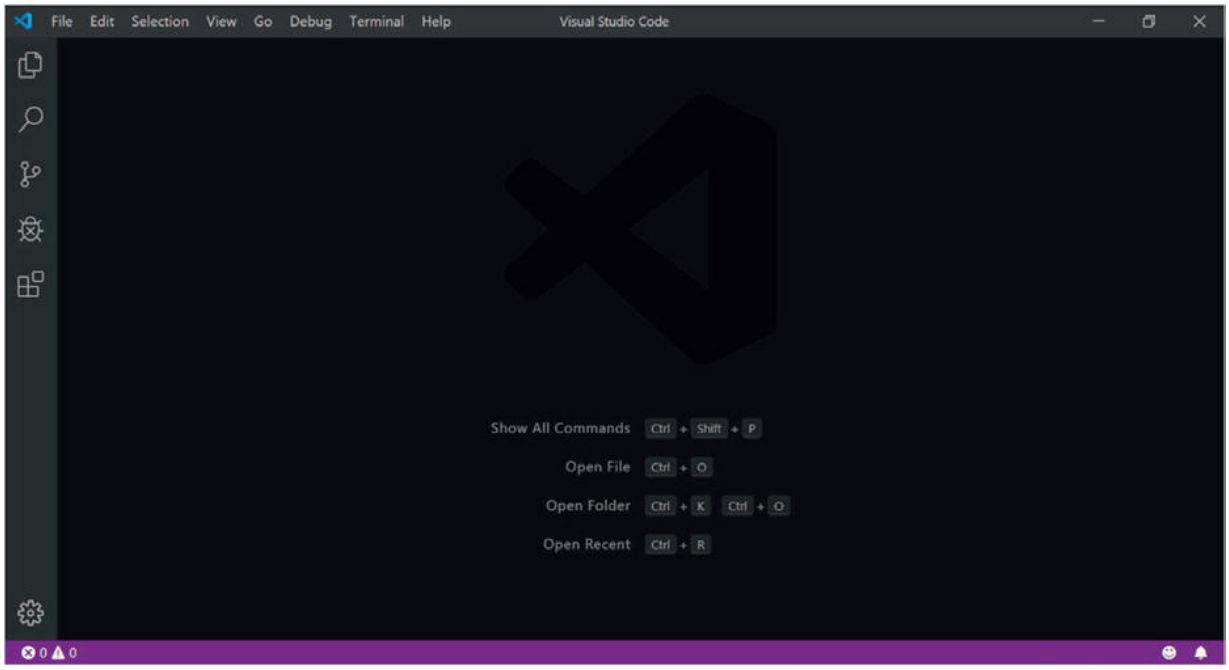
Po wybraniu żądanego schematu kolorów (można zawsze zmienić go później na inny) zamknijmy stronę powitania (Welcome). Dowolny plik lub strona otwarta w Code będą

miały swoją kartę u góry okna (zobacz rysunek 1–4), niezależnie od tego, czy będzie to plik z kodem do edycji, nad którym pracujemy, czy statyczna strona, taka jak strona powitania. Jeśli mamy otwartych wiele plików, możemy łatwo przechodzić do innego, klikając jego kartę. Na razie mamy otwartą tylko stronę powitania. Zamknijmy ją, aby uzyskać puste okno. Możemy to zrobić, klikając lewym przyciskiem myszy przycisk X na karcie lub korzystając ze skrótu klawiszowego Ctrl+W.



Rysunek 1-4. *Lewy górny róg okna Code, gdzie pojawiają się karty wszystkich stron i plików. Teraz mamy tu otwartą tylko kartę strony powitania*

Po zamknięciu strony niewiele się dzieje – mamy na środku okna dużą pustą przestrzeń, jak pokazano na rysunku 1–5.



Rysunek 1-5. *Okno programu Visual Studio Code bez otwartych żadnych plików*

Możemy teraz zainstalować tzw. „rozszerzenia” (extensions) programu Code, aby dołączyć dodatkową funkcjonalność do edytora. Rozszerzenia są zarządzane i instalowane poprzez przycisk z lewej strony okna. Pasek narzędzi z lewej strony okna może

przyjmować różne formy w zależności od tego, który z tych przycisków został naciśnięty ostatnio. Wskazując kursorem myszy poszczególne przyciski, można uzyskać informację, co one oznaczają. Klikamy przycisk Extensions (rozszerzenia) po znalezieniu go w ten sposób. Można też skorzystać ze skrótu klawiszowego Ctrl+Shift+X. Spowoduje to pojawienie się listy rozszerzeń z lewej strony ekranu. Ponowne naciśnięcie przycisku schowa listę rozszerzeń.

Wewnątrz listy rozszerzeń możemy wyszukiwać rozszerzenia, korzystając z okienka wyszukiwania u góry. Klikamy wewnątrz pola wyszukiwania i wpisujemy „C#". Powinniśmy zobaczyć na liście wyników element zatytułowany „C#" z opisem „C# for Visual Studio Code". Możemy też zauważyć, że wydawca danego rozszerzenia jest wymieniony pod tym opisem. W przypadku tego rozszerzenia wydawcą jest firma „Microsoft", która też odpowiada za program Visual Studio Code i sam język C#.

Klikamy to rozszerzenie, a pojawi się nowa karta ze szczegółami dotyczącymi danego rozszerzenia. Pod głównym opisem u góry strony zobaczymy przycisk do zainstalowania rozszerzenia. Klikamy ten przycisk, a rozpocznie się instalacja rozszerzenia. Jeśli pojawi się przy tym propozycja zainstalowania jakichś dodatkowych rozszerzeń, zezwólmy na ich zainstalowanie.

Następnie zainstalujemy rozszerzenie do debugowania w Unity. Pozwoli nam to na „dołączanie" naszego edytora kodu do Unity, abyśmy mogli wykorzystywać edytor kodu do ustawiania punktów przerwania w kodzie. Punkt przerwania jest miejscem w kodzie, które spowoduje zatrzymanie wykonywania programu, gdy będzie on uruchomiony w trybie debugowania. Po zatrzymaniu wykonywania kodu możemy przeglądać wszelkie informacje i dane w naszym programie, a następnie wznowić wykonywanie programu w dowolnym momencie. Jest to bardzo przydatna funkcja, z której później będziemy korzystać.

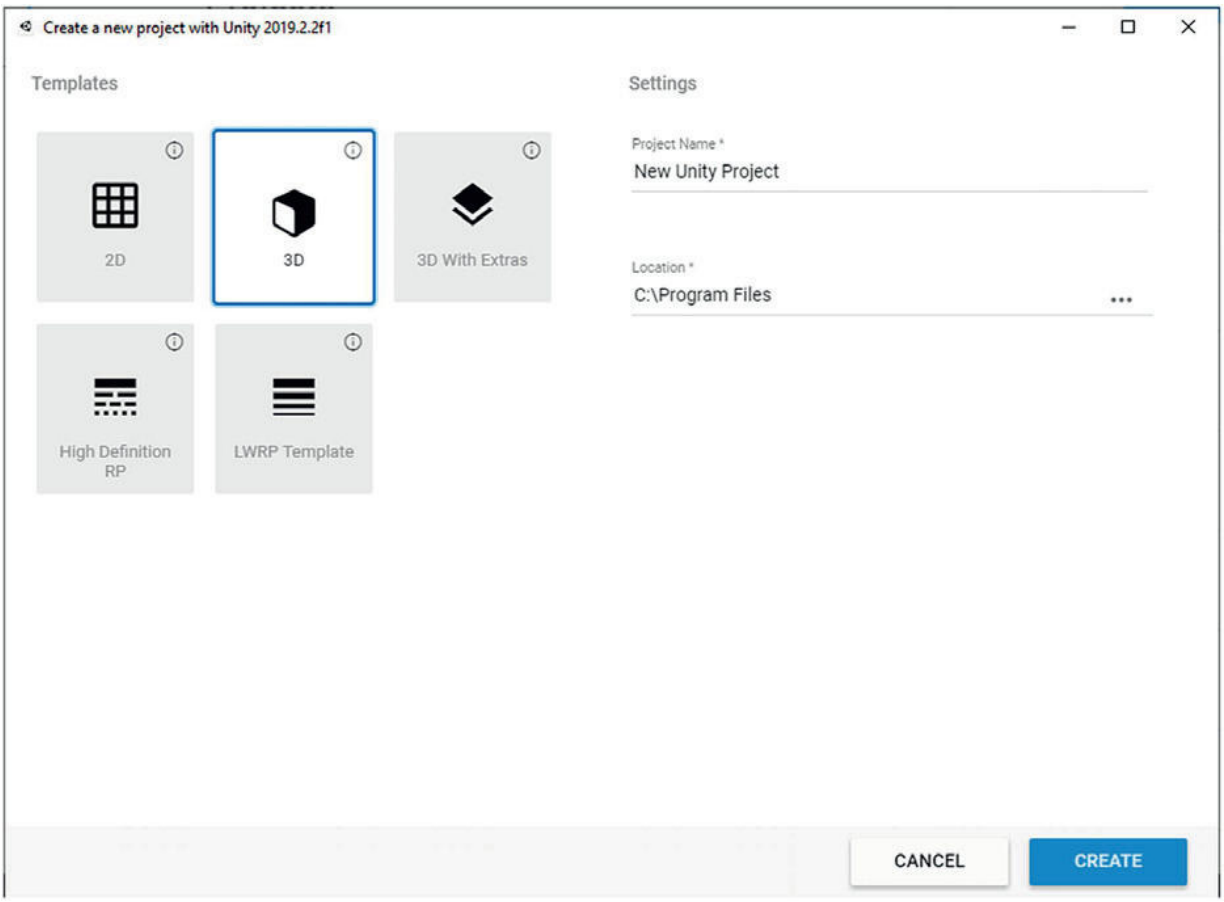
Dokładną nazwą rozszerzenia jest „Debugger for Unity" i jest ono wydane przez firmę Unity Technologies. Możemy je znaleźć w taki sam sposób – wpisując nazwę w pasku wyszukiwania u góry listy Extensions.

Po zainstalowaniu tych rozszerzeń możemy zamknąć oprogramowanie Visual Studio Code. Będziemy korzystać z niego więcej w części 2. Na razie skupimy się na samym edytorze Unity.

Tworzenie projektu

Teraz możemy skorzystać z Unity Hub do utworzenia swojego pierwszego projektu, abyśmy mieli pierwsze środowisko do nauki. W Unity Hub klikamy kartę Projects (projekty) z lewej strony, a następnie niebieski przycisk „New” (nowy) w prawym górnym rogu.

Pojawi się okno dialogowe (zobacz rysunek 1–6), pozwalające nam wybrać szablon, na którym chcemy oprzeć swój projekt.



Rysunek 1-6. *Okno dialogowe do utworzenia nowego projektu Unity*

Szablon jest po prostu punktem wyjścia dla naszego projektu. Najprostszymi szablonami są pierwsze dwa o nazwach „2D” i „3D”. Są to w zasadzie puste projekty przygotowane pod gry 2-wymiarowe i 3-wymiarowe.

Zaczniemy od pustego projektu trójwymiarowego, więc wybierzemy szablon 3D, klikając go. Po zaznaczeniu będzie miał on wokół siebie niebieską ramkę.

Z prawej strony opcji szablonu jest pole na nazwę projektu (możemy ją zmienić później) i katalog (ścieżkę do plików) na komputerze, w którym będą zapisywane pliki projektu. Nazwiemy nasz projekt „ExampleProject" (projekt przykładowy) – warto zwrócić uwagę na brak spacji pomiędzy dwoma słowami, ponieważ ścieżki do plików nie zawsze je lubią.

Katalog do zapisania projektu można zmienić na dowolny. Niezależnie od wybranej ścieżki do plików, wewnątrz tego folderu zostanie utworzony folder o nazwie takiej, jak nazwa projektu. Folder ten będzie „głównym katalogiem" naszego projektu i wszystkie pliki oraz zasoby naszego projektu będą przechowywane w tym folderze.

Po wybraniu żądanej ścieżki klikamy niebieski przycisk „Create" (utwórz) w prawym dolnym rogu i czekamy aż Unity utworzy podstawowe pliki projektu. Po zakończeniu pojawi się sam edytor Unity z otwartym naszym nowym projektem gotowym do edycji.

Podsumowanie

Poniżej znajduje się podsumowanie tego, czego dowiedzieliśmy się w tym rozdziale:

- Program Unity Hub będzie używany do pobierania nowych wersji edytora Unity, odinstalowywania starych wersji, których już nie potrzebujemy, tworzenia nowych projektów i otwierania istniejących projektów.
- Otwarcie projektu w Unity Hub uruchomi edytor Unity, w którym wykorzystujemy silnik Unity do opracowywania swojej gry.
- Projekt gry Unity jest przechowywany na komputerze wraz ze wszystkimi powiązanymi plikami (takimi jak grafiki i kod) w „katalogu głównym" o nazwie takiej samej, jak nazwa projektu.
- Nasz kod będzie pisany przy użyciu Visual Studio Code, edytora tekstu zaprojektowanego specjalnie do pisania kodu. Będzie nam on oferować przydatne funkcje, których nie mają zwykłe edytory tekstu, ułatwiając formatowanie i poruszanie się po kodzie.

Rozdział 2

Podstawy Unity

Gdy już skonfigurowaliśmy oprogramowanie Unity i przygotowaliśmy nowy projekt, nad którym możemy pracować, czas na zaznajomienie się z silnikiem. Z tym interfejsem użytkownika będziemy mieli całkiem sporo do czynienia podczas projektowania swoich gier, więc powinniśmy go wcześnie dobrze poznać.

Okna

Interfejs Unity jest podzielony na kilka okien, które służą różnym celom. Każde okno ma niewielką kartę w swoim lewym górnym rogu z nazwą danego okna, która określa, jakiego typu jest to okno.

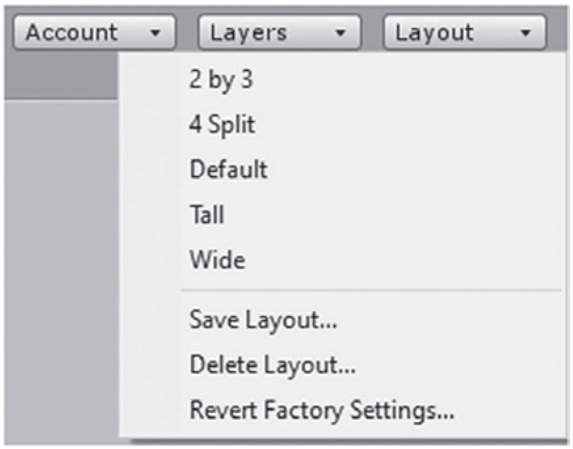
Jak w wielu dzisiejszych programach komputerowych, każde z tych okien jest odrębnym elementem programu, który można swobodnie przestawić, zmienić jego rozmiar lub całkiem usunąć z interfejsu. Istnieje też kilka innych typów okien, z których na razie nie korzystamy, więc nie widać ich na naszym ekranie – ale jeśli kiedyś będziemy chcieli z nich skorzystać, możemy je bez problemu dodać.

Możemy zauważyć, że jeśli klikniemy lewym przyciskiem myszy jeden z nagłówków okna, a następnie przytrzymamy wciśnięty przycisk myszy i przeciągniemy kursor myszy, okno może zostać przeniesione w inne miejsce ekranu całego programu. Możemy przy tym podzielić obszar jednego okna, aby zakryć jego część innym oknem. Jeśli chcemy coś zrobić z jakimś oknem, to najczęściej wystarczy po prostu zacząć je przeciągać myszą i zobaczyć, jak Unity na to zareaguje.

Można też łączyć okna w jednym obszarze, aby umieszczać ich karty obok siebie (zobacz rysunek 2–1). Ostatnio kliknięta karta przejdzie na pierwszy plan i wypełni cały obszar połączonego okna.

Rysunek 2-1. *Dwie karty okien połączone obok siebie w obszarze tego samego okna. Okno Project (projekt) jest aktywne i wypełnia całe okno, ale w tym samym obszarze można aktywować okno Console (konsola), klikając jego kartę*

Unity pozwala też na zapisanie wszystkich obecnych okien i ich rozmiarów oraz pozycji w postaci *układu*, któremu możemy przypisać nazwę. Robi się to przy pomocy niewielkiego przycisku rozwijanego „Layout" (układ) w prawym górnym rogu edytora Unity. Kliknięcie tego przycisku pokazuje wszystkie wbudowane układy, z których możemy wybierać, jak widać na rysunku 2–2.



Rysunek 2-2. *Przycisk Layout w prawym górnym rogu edytora Unity po kliknięciu go w celu pokazania dostępnych opcji*

Domyślnie ustawiony jest układ o nazwie „Default" (domyślny). Możemy spróbować kliknąć inny układ i zobaczyć, jak Unity automatycznie poprzestawia swoje okna. Możemy też wybrać opcję „Save Layout..." (zapisz układ) z listy rozwijanej, aby zapisać swój bieżący układ okien pod nową nazwą, żeby móc z niego korzystać później i ustawiać okna w żądany sposób. W ten sposób, jeśli kiedykolwiek stracimy przypadkowo jakieś okno lub niechcący przestawimy swoje ułożenie okien, możemy po prostu przywrócić wszystkie okna do żądanego stanu, ładując swój układ z tego rozwijanego menu.

Układy mogą być też bardzo przydatne przy zajmowaniu się różnymi aspektami programowania gier. Niektóre działania mogą być łatwiejsze przy ułożeniu okien w innych pozycjach. Możliwość zapisywania układów ułatwia nam przełączanie się pomiędzy różnymi działaniami – wystarczy tylko kilka kliknięć.

Układ domyślny zawiera wszystkie najważniejsze okna. Przyjrzyjmy się im wszystkim, aby dowiedzieć się, do czego służą.

Okno Project (projekt)

Okno Project pokazuje nam wszystkie nasze *aktywa*. Tym terminem w programowaniu gier określa się wszystkie elementy, które możemy wykorzystywać w swojej grze – grafiki, efekty dźwiękowe, muzykę, pliki z kodem, poziomy gry, itd.

Gdy przejdziemy do tworzenia aktywów, zobaczymy je tutaj w oknie Project. Okno to funkcjonuje podobnie do systemu plików w komputerze. Poszczególne aktywa można przechowywać w folderach zwanych też katalogami. Możemy mieć na przykład folder na wszystkie swoje efekty dźwiękowe, inny na wszystkie pliki z kodem (zwane skryptami), itd. Możemy je organizować w dowolny sposób w tym oknie i w tym oknie możemy je wyszukiwać, zaznaczać oraz wykorzystywać w silniku Unity.

W naszym projekcie zostało od razu skonfigurowanych tutaj kilka domyślnych folderów: folder Packages (pakiety), którym nie musimy się na razie zajmować oraz folder Assets (aktywa), który jest głównym folderem do przechowywania naszych aktywów. Można kliknąć strzałkę obok folderu, aby ukryć lub pokazać jego zawartość (jeśli ma coś w środku). Jeśli rozwiniemy folder Assets, zobaczymy, że ma już w sobie domyślnie folder Scenes (sceny) oraz element „SampleScene” (scena przykładowa).

Okno Scene (scena)

Okno Scene pozwala nam zobaczyć środowisko, w jakim toczy się nasza gra. To, co moglibyśmy nazywać „poziomem” w swojej grze, w Unity nazywane jest „sceną”. Sceny są zapisywane jako aktywa, więc zobaczymy je w swoim oknie Project (projekt), gdy je zapiszemy – ta scena jest zapisana jako element „SampleScene” (scena przykładowa) utworzony domyślnie w naszym nowym projekcie.

Każda scena zawiera w sobie swoją własną kolekcję obiektów. Okno sceny pozwala nam przeglądać scenę i nawigować po niej, obserwując świat gry jakby ze swobodnie unoszącej się kamery. Jest to nasz główny wgląd w środowisko gry.

Przykładowa scena, którą mamy teraz otwartą, właściwie nie ma nic w sobie – jedynie oświetlenie, które jest niewidocznym obiektem rzucającym światło na wszystkie elementy znajdujące się na scenie oraz kamerę, która jest obiektem, poprzez który gracz będzie oglądał scenę podczas gry.

Gdybyśmy mieli inne sceny (później dodamy kilka), to wszystkie przechowywalibyśmy w folderze Scenes (sceny), a podwójnie kliknięcie którejś z nich załadowałoby daną scenę, pozwalając nam ją oglądać i edytować w oknie Scene.

W oknie Scene poruszanie myszą przy wciśniętym prawym przycisku powoduje obracanie kamery, podobnie jak w przypadku rozglądania się w grze z perspektywą pierwszoosobową. Przy wciśniętym prawym przycisku myszy możemy korzystać z klawiszy WASD do przemieszczania kamery – podobnie jak w wielu grach. W, aby poruszyć się do przodu, S do tyłu, A w lewo i D w prawo. Możemy też użyć klawisza Q, aby przemieścić się bezpośrednio w dół i E, aby przemieścić się bezpośrednio w górę.

Okno Hierarchy (hierarchia)

Okno Hierarchy pozwala nam zobaczyć obiekty zawarte w bieżącej scenie. Jak wcześniej powiedziałem, scena jest właściwie kolekcją obiektów. Gdy przełączamy się z jednej sceny do innej, odkładamy na bok wszystkie obiekty z bieżącej sceny i pobieramy wszystkie obiekty z nowej sceny.

Obecnie w oknie Hierarchy widać te dwa obiekty z naszej sceny, o których wspominałem wcześniej: „Directional Light” (światło kierunkowe) i „Main Camera” (kamera główna). Domyślnie w scenie będziemy mieli od razu źródło światła i kamerę, więc zobaczymy je wymienione w oknie Hierarchy.

Są to obiekty typu *GameObject*. W uproszczeniu element GameObject jest jakimś obiektem obecnym w naszej scenie. Może to być rekwizyt, taki jak skrzynia, roślina lub drzewo. Może to być postać gracza lub wroga albo element, który gracz może zebrać z ziemi. Może to być niezwiązane z niczym źródło światła albo obiekt, który nic nie robi; jedynie istnieje, ale na scenie jest niewidoczny. W najprostszej formie może to być po prostu punkt w przestrzeni posiadający jakąś nazwę.

Okno Inspector (inspektor)

Okno Inspector jest bardzo ważnym oknem, z którego będziemy szeroko korzystać w naszych przygodach z Unity.

Jak właśnie wspominaliśmy, okno Hierarchy pokazuje listę wszystkich elementów GameObject na scenie. Możemy kliknąć jeden z tych elementów GameObject w oknie Hierarchy, aby go zaznaczyć. Zauważymy, że spowoduje to zmianę zawartości okna Inspector. Teraz pokazuje nam ono informacje na temat zaznaczonego elementu GameObject. U góry okna Inspector widać pole zawierające nazwę tego elementu

GameObject, które możemy kliknąć, aby wpisać nową nazwę, gdybyśmy chcieli ją zmienić. Istnieje też kilka przycisków rozwijanych, jeden o nazwie „Tag” (znacznik) i drugi o nazwie „Layer” (warstwa) – dowiemy się później, do czego one służą.

Jednak główną funkcjonalnością okna Inspector jest pokazanie nam wszystkich **składników** dołączonych do zaznaczonego elementu GameObject.

Składniki

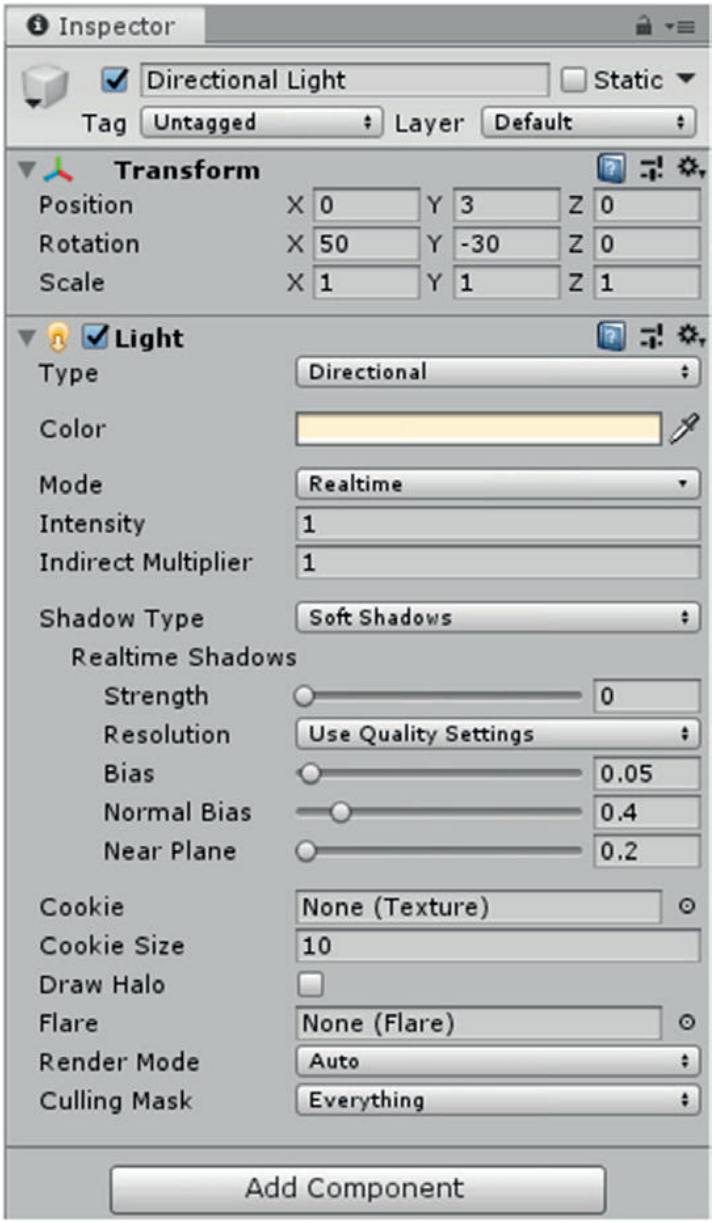
Składnik (*component*) to termin używany przez Unity na określenie jakiegoś elementu funkcjonalności gry, który jest dołączony do obiektu GameObject. Składnik nie może istnieć bez elementu GameObject, do którego jest dołączony.

Domyślnie w silniku Unity istnieje wiele różnych rodzajów składników, które służą różnym celom. Istnieje składnik Light (światło) do rzucania światła, czy to będzie źródło podobne do światła słonecznego obejmujące całą scenę, czy coś w rodzaju snopu światła latarki.

Przyjrzyjmy się składnikowi Light. Ponieważ okno Inspector pokazuje nam składniki przypisane do zaznaczonego elementu GameObject, spróbujmy kliknąć obiekt Directional Light (światło kierunkowe) w oknie Hierarchy, aby go zaznaczyć. Zawartość okna Inspector zostanie odpowiednio zaktualizowana i będzie wyglądać, jak na rysunku 2–3.

Pod zestawem informacji podstawowych u góry, obejmujących między innymi nazwę obiektu GameObject, zobaczymy nagłówki wszystkich składników dołączonych do danego elementu GameObject. Ten element GameObject ma dołączone dwa składniki: Transform (przekształcanie) i Light (światło). Możemy klikać nagłówki tych składników (zawierające nazwę składnika), aby je ukrywać („zwijać”) lub pokazywać je ponownie („rozwijać”) Pod nagłówkiem składnika wymienione są różne jego właściwości jako różnego rodzaju pola, które możemy zmieniać, aby wpływać na dany składnik – na przykład zmieniać intensywność światła, jego kolor, sposób rzucania cieni, itd. Nazwa pola jest po lewej stronie, a jego wartość po prawej stronie. Niektóre z tych wartości są liczbami, a niektóre są niewielkimi „suwakami”, które można przeciągać myszą – napotkamy tu wiele różnych rodzajów pól przeznaczonych do edycji różnego rodzaju wartości. Taka jest główna funkcjonalność okna Inspector: przeglądanie i edytowanie właściwości składników elementu GameObject.

Innym przykładem jest składnik Camera (kamera). Jest on bardzo istotny. To właśnie on „renderuje” (czyli „rysuje”) scenę na ekranie gracza, gdy gra on w grę. Jeśli zaznaczymy obiekt GameObject o nazwie Main Camera (kamera główna) w oknie Hierarchy, zobaczymy składnik Camera wymieniony w oknie Inspector.



Rysunek 2-3. *Okno Inspector z zaznaczonym domyślnym obiektem GameObject o nazwie Directional Light*

Podczas dalszej pracy nad grami w tej książce nabędziemy wiele doświadczenia w korzystaniu z wielu rodzajów składników. Unity ma też pomocną, oficjalną dokumentację, do której można łatwo uzyskać dostęp, klikając niewielką ikonę, która wygląda jak książka z symbolem ? na okładce, znajdującą się po prawej stronie nagłówka każdego składnika w oknie Inspector. Otworzy to dokumentację dla tego konkretnego typu składnika w domyślnej przeglądarce WWW.

Nasz kod będzie dołączany do elementów GameObject również w postaci składników. W tym przypadku składniki te nazywają się „skryptami”: to znaczy skrypt jest

składnikiem, który wykonuje nasz kod, gdy zostanie on dołączony do elementu GameObject. Gdy więc piszemy kod, sposobem dodania go do gry jest dołączenie go jako składnik typu skrypt do obiektu GameObject.

Oznacza to, że możemy ponownie wykorzystywać i mieszać ze sobą różne rodzaje funkcjonalności, jeśli odpowiednio będziemy pisać i definiować swoje skrypty.

Na przykład, nasz pierwszy projekt przykładowy jest grą, w której gracz musi unikać różnych przeszkód. Powiedzmy, że tworzymy tego rodzaju projekt i chcemy tworzyć różne rodzaje przeszkód, aby gra była ciekawsza. Chcemy mieć przeszkody, które wystrzeliwują kule ognia, obracające się ostrza i toczące się kolczaste kule przemieszczające się tam i z powrotem pomiędzy dwoma punktami.

Każdy z tych elementów funkcjonalności może być obsługiwany przez oddzielny składnik typu skrypt: Shooting (strzelanie), który okresowo wystrzeliwuje pociski przed danym elementem GameObject; Spinning (obracanie), który sprawia, że obiekt stale obraca się wokół swojej osi; Patrolling (patrolowanie), który sprawia, że obiekt przemieszcza się tam i z powrotem pomiędzy dwoma punktami. Następnie możemy mieć składnik o nazwie Hazard (niebezpieczeństwo), który będziemy dołączać do kul ognia, obracających się ostrzy lub wędrujących kolczastych kul, który będzie powodować śmierć gracza przy zetknięciu z danym obiektem.

Najlepsze w składnikach jest to, że możemy je mieszać ze sobą, aby tworzyć nowe typy przeszkód.

Ponieważ każdy oddzielny fragment funkcjonalności jest zawarty wewnątrz swojego własnego składnika, możemy sprawić, żeby dowolny obiekt strzelał pociskami, obracał się, patrolował lub zabijał gracza przy dotknięciu. Ponieważ nie ma ograniczenia liczby składników, które możemy dodać do pojedynczego elementu GameObject, możemy utworzyć element wystrzeliwujący ogniste pociski, który wiruje w kółko, dołączając składniki Shooting i Spinning do pojedynczego obiektu GameObject. Możemy do jego przeciwnej strony dołączyć ostrze, które będzie działać jak niebezpieczeństwo. Możemy utworzyć patrolującą, kolczastą kulę, która będzie wystrzeliwać ogniste pociski.

Na tym to polega. O ile każdy element funkcjonalności będzie zawarty w swoim własnym składniku typu skrypt, możemy po prostu dołączać dowolną kombinację składników typu skrypt do jednego obiektu GameObject, aby łączyć różne zakodowane przez nas rzeczy w jednym obiekcie.

Jest to jedna z głównych zalet systemu składników Unity. Zapewnia on swego rodzaju system klocków, w którym możemy dowolnie mieszać różne funkcje.

Dodawanie elementów GameObject

Zacznijmy zaznajamianie się z Unity, tworząc kilka elementów GameObject i manipulując nimi. Nie będziemy tworzyć modeli postaci, statków kosmicznych, dział, czy innych tego rodzaju obiektów wewnątrz silnika Unity. Unity nie jest pakietem do modelowania (tworzenia obiektów trójwymiarowych) ani edytorem obrazów (do tworzenia obiektów dwuwymiarowych). Jest to silnik gier; tworzymy modele i je animujemy w innym rodzaju oprogramowania, a następnie importujemy je do Unity, umieszczając je po prostu w folderze swojego projektu, a Unity pozwoli nam je następnie umieszczać w swoich scenach.

Ze względu na uczenie się funkcji silnika oraz zasad programowania, nie będziemy zajmować się wyrafinowanymi elementami graficznymi.

Jednakże Unity pozwala na tworzenie od ręki elementów GameObject dla podstawowych kształtów przy pomocy kilku prostych opcji.

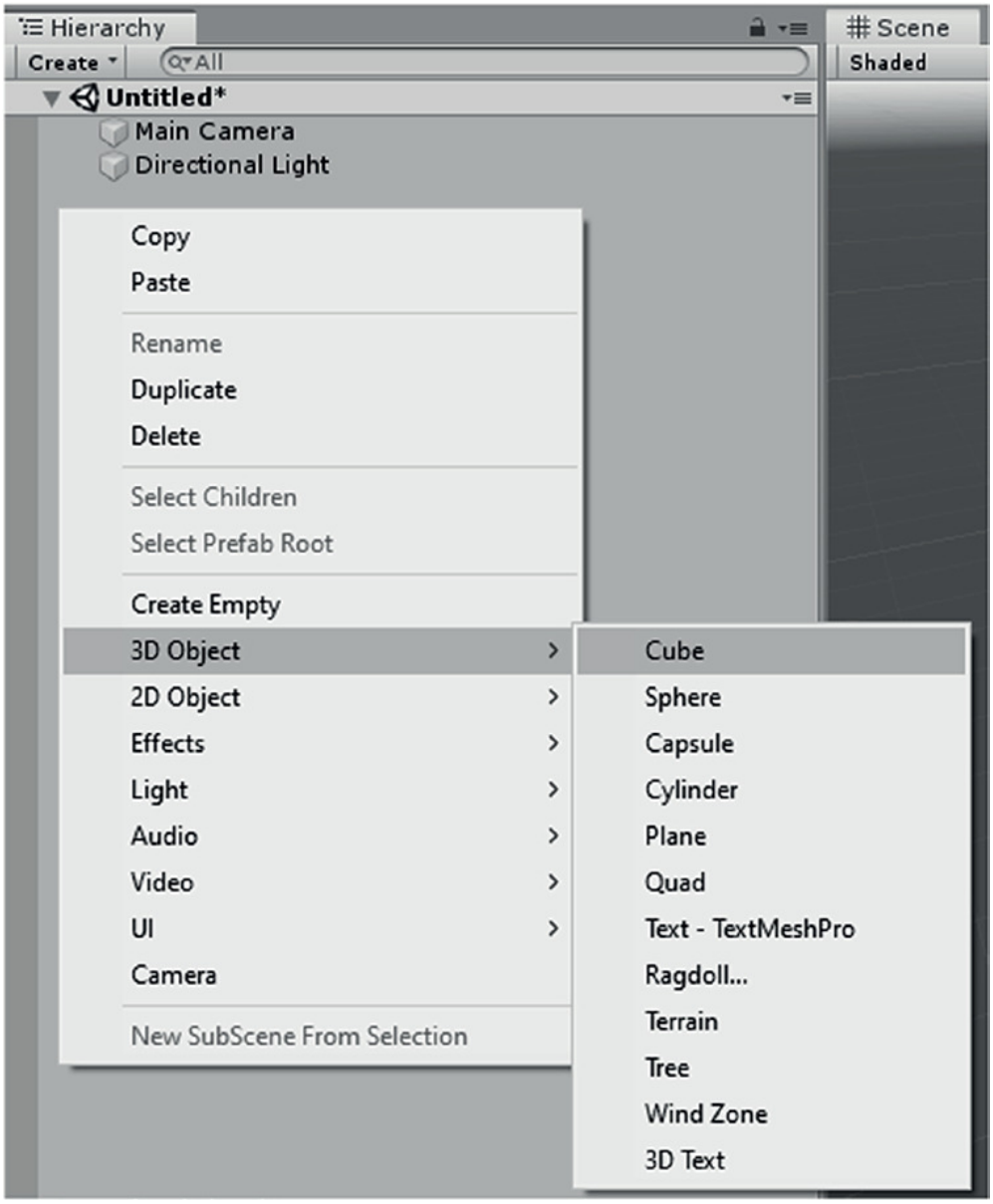
Tuż pod nazwą okna edytora Unity (paskiem tytułu) widać pasek menu z różnymi opcjami: File (plik), Edit (edycja), Assets (aktywa), GameObject, Component (składnik), Window (okno) i Help (pomoc). Można je klikać, aby rozwijać menu z dodatkowymi opcjami.

Menu GameObject może służyć do tworzenia prostych, często używanych elementów GameObject: kształtów, kamer, świateł, itd.

Korzystając z tego menu, możemy utworzyć sześcian poprzez wybranie opcji GameObject ➤ 3D Object (obiekt trójwymiarowy) ➤ Cube (sześcian). Alternatywnie możemy też kliknąć prawym przyciskiem myszy puste miejsce w oknie Hierarchy (poza nazwą jakiegoś istniejącego elementu GameObject) i z menu kontekstowego wybrać 3D Object ➤ Cube, jak pokazano na rysunku 2–4.

Po wykonaniu którejś z tych czynności, zauważymy, że obiekt Cube zostanie dodany do naszego okna Hierarchy i pojawi się w oknie Scene (scena), jeśli będzie „w zasięgu wzroku” naszej kamery.

Jeśli nie widać go w oknie Scene, to prawdopodobnie patrzymy na inny fragment sceny. Możemy łatwo umieścić dany obiekt GameObject w polu widzenia na aktualnej scenie, klikając ten obiekt w oknie Hierarchy, żeby go zaznaczyć, a następie umieszczając kursor myszy nad oknem Scene, żeby było ono aktywne i naciskając klawisz F. Jest to wygodny skrót, który pozwala skierować naszą uwagę na scenie na zaznaczone obiekty, aby były widoczne. Jeśli kiedykolwiek na scenie oddalimy się od swoich obiektów, to możemy w ten sposób łatwo do nich wrócić.



Rysunek 2-4. *Tworzenie sześcianu przez kliknięcie prawym przyciskiem myszy w oknie Hierarchy i wybór odpowiedniej opcji z menu kontekstowego*

Kontynuując nasze wcześniejsze rozważania dotyczące składników, sprawdźmy, jakie składniki są obecne we właśnie utworzonym sześcianie. Upewniając się, że sześcian jest zaznaczony, spójrzmy na okno Inspector.

Składnikiem umieszczonym na samej górze zawsze będzie Transform (przekształcanie). Każdy element GameObject ma składnik Transform. Składnik Transform określa położenie, rozmiar i obrót obiektu. Jest to istotna część elementu GameObject. Składniki można w kodzie usuwać z obiektów, ale nigdy nie można usunąć składnika Transform. Musimy usunąć cały obiekt GameObject, jeśli chcemy zniszczyć jego składnik

Transform. Każdy obiekt, który istnieje na naszej scenie, musi mieć jakieś położenie, prawda? Musi *gdzieś być*.

Poza składnikiem Transform, możemy zauważyć kilka innych składników. Mamy składnik Mesh Renderer (renderowanie siatki) i Mesh Filter (filtr siatki).

Słowo *mesh* (siatka) można traktować jako synonim określenia „model trójwymiarowy". Określenie *render* (renderowanie) jest bardziej wyrafinowanym słowem oznaczającym rysowanie lub wyświetlanie czegoś na ekranie. Mesh Renderer jest więc składnikiem, który pozwala na rysowanie modelu trójwymiarowego, choć samo rysowanie jest przeprowadzane przez składnik Camera.

Mesh Filter jest składnikiem, który zawiera siatkę obiektu trójwymiarowego, którą chcemy przekazać do składnika Mesh Renderer. Prawie zawsze, gdy mamy składnik Mesh Renderer, będziemy mieli też składnik Mesh Filter, ponieważ ten drugi informuje pierwszy, co ma zostać wyświetlone.

Obok nazwy składnika Mesh Renderer w oknie Inspector możemy zauważyć niewielkie pole wyboru. Tego rodzaju składniki mogą być włączane lub wyłączane poprzez kliknięcie tego pola wyboru. Jeśli ktoś nie wierzy, że składnik Renderer faktycznie rysuje sześcian na scenie, może spróbować kliknąć to pole wyboru, aby usunąć jego zaznaczenie. Zauważymy, że sześcian przestanie być wyświetlany na ekranie w oknie Scene. Po ponownym zaznaczeniu tego pola sześcian pojawi się z powrotem na scenie.

Podsumowanie

W tym rozdziale nauczyliśmy się następujących rzeczy:

- Edytor Unity składa się z różnych *okien*, które służą określonym celom. Dowolne okno można przestawić w inne miejsce lub zmienić jego rozmiar, klikając jego kartę i przeciągając myszą.
- *Aktywa* są plikami wykorzystywanymi w naszej grze, zawierającymi na przykład elementy graficzne, dźwięk lub kod. Można je przeglądać w *oknie Project* i często będziemy je dołączać do swojej gry po prostu przeciągając je z tego okna.
- *Scena* jest *rodzajem aktywów*, który przypomina środowisko gry, takie jak pojedynczy poziom. Możemy ładować je do przeglądania i edycji w *oknie Scene*.
- Element `GameObject` jest obiektem, który istnieje na scenie. Funkcjonalność tych obiektów jest określana przez *składniki*, które do nich dołączamy. Unity zapewnia wiele wbudowanych składników dla fundamentalnych spraw, takich jak wyświetlanie modelu trójwymiarowego, rzucanie światła, obsługę fizyki i kolizji, itd.

- Składniki dołączone do danego elementu GameObject można przeglądać w *oknie Inspector*. Tutaj możemy dostosowywać ich funkcjonalność, edytując pola z nimi związane – na przykład, jasność światła. Każdy składnik jest osobnym wystąpieniem ze swoimi własnymi wartościami, które można zmieniać, aby zapewniać inną funkcjonalność.
- Każdy element GameObject ma podstawowy składnik `Transform`, który określa *położenie*, *obrócenie* i *rozmiar* obiektu. Inne typy składników mogą być dodawane i usuwane na bieżąco poprzez kod, ale nie dotyczy to składnika Transform – może istnieć tylko jeden dla danego elementu GameObject i nie można go usunąć.

Rozdział 3

Manipulowanie sceną

Poznaliśmy podstawy najważniejszych okien w silniku Unity i wiemy, jak tworzyć proste obiekty i przeglądać ich składniki poprzez okno Inspector. Teraz zapoznamy się z przemieszczaniem, obracaniem i zmienianiem rozmiaru elementów GameObject na naszej scenie.

Narzędzia przekształcania

Ta część interfejsu edytora Unity, która znajduje się tuż pod paskiem tytułu i paskiem menu (z opcjami File, Edit, Assets, itd.), nazywa się *paskiem narzędzi*. Jest to pasek rozciągający się przez całą szerokość ekranu, który zawiera kilka przydatnych przycisków. Obejmuje to między innymi rozwijane menu Layout (układ), które poznaliśmy wcześniej i które znajduje się na prawym końcu paska narzędzi.

Teraz zapoznamy się z grupą przycisków po lewej stronie paska narzędzi, które pokazano na rysunku 3–1.

Rysunek 3-1. *Przyciski odpowiadające za narzędzia przekształcania. Obecnie zaznaczony jest drugi przycisk, co widać poprzez ciemniejsze tło niż w przypadku pozostałych przycisków*

Są to *narzędzia przekształcania*. Dowiedzieliśmy się już, że składnik Transform (przekształcanie) określa położenie, obrócenie i rozmiar obiektu, więc możemy się domyślić, że narzędzia przekształcania służą głównie do przemieszczania, obracania i zmiany rozmiaru elementów GameObject na scenie.

Dostępne jest sześć przycisków dla sześciu różnych rodzajów narzędzi. Z prawej strony może też pojawić się siódmy przycisk, który jest związany z niestandardowymi narzędziami edytora, którymi nie musimy się teraz zajmować. Jeśli go nie widać, nie trzeba się tym przejmować.

Każdy z tych przycisków można kliknąć, aby przełączyć się na inny rodzaj narzędzia. W danym momencie może być aktywne tylko jedno z tych narzędzi, a wszystkie one służą różnym celom.

Do przełączania się pomiędzy tymi narzędziami możemy też używać skrótów klawiszowych Q, W, E, R, T oraz Y, co jest często szybsze niż klikanie myszą.

Pierwszym narzędziem, aktywowanym skrótem klawiszowym **Q**, jest *Hand Tool* (narzędzie ręki), które pozwala na klikanie i przeciąganie myszą na scenie w celu zmiany widoku przedstawianego w oknie Scene (scena). Nie powoduje ono edytowania sceny. Jedynie pomaga w nawigowaniu po niej.

Pozostałe narzędzia pozwolą nam na edytowanie zaznaczonych obiektów GameObject. W oknie Scene zaznaczone narzędzie przekształcania udostępni niewielkie gadżety zwane „gizmo” na lub wokół zaznaczonych elementów GameObject. Gadżety te są prostymi narzędziami, które można klikać i przeciągać, aby wykorzystywać dane narzędzie przekształcania do interakcji z zaznaczonym elementem GameObject. Zauważymy, że po zaznaczeniu elementu GameObject i przełączaniu się pomiędzy tymi narzędziami, dany gadżet rysowany wokół obiektu zmienia się wraz ze zmianą wybranego narzędzia.

Klawisz **W** włącza narzędzie *Move Tool* (narzędzie przemieszczania). Gdy to narzędzie jest aktywne, wokół zaznaczonego elementu GameObject wyświetlany jest gadżet w postaci strzałek. Można przeciągać obiekt w określonych kierunkach, klikając te strzałki i przeciągając myszą wzdłuż nich. Można też przeciągać obiekt w płaszczyźnie wyznaczonej przez dwie strzałki na raz, klikając kwadratowy kształt pomiędzy strzałkami. Przytrzymanie wciśniętego klawisza Ctrl podczas przeciągania spowoduje przemieszczanie obiektu w przyrostach równych 1 jednostce odległości.

Klawisz **E** włącza *Rotate Tool* (narzędzie obracania). Wyświetla ono gadżety w postaci okręgów wokół zaznaczonego elementu GameObject. Klikanie i przeciąganie tych okręgów powoduje obracanie obiektu, a każdy okrąg odpowiada za obracanie go wokół innej osi. Można też kliknąć pomiędzy tymi okręgami, aby obracać obiekt wokół wielu osi na raz.

Klawisz **R** włącza *Scale Tool* (narzędzie skalowania). Wyświetla ono gadżety podobne do strzałek, ale zakończone grotami w kształcie sześcianów. Klikając i przeciągając te sześcienne groty można zmieniać szerokość (czerwony sześcian), długość (niebieski

sześcian) lub wysokość (zielony sześcian) obiektu. Kliknięcie i przeciągnięcie sześcianu znajdującego się w środku gadżetu powoduje skalowanie naraz wszystkich wymiarów obiektu – to znaczy jednoczesne zwiększanie lub zmniejszanie jego szerokości, długości i wysokości.

Klawisz **T** włącza *Rect Tool* (narzędzie prostokąta). Ma ono głównie zastosowanie w projektach dwuwymiarowych, ale ma też pewne zastosowania w trzech wymiarach. Gadżet w tym przypadku przedstawia prostokąt wokół zaznaczonego obiektu z kółeczkami w jego wierzchołkach. Można klikać i przeciągać boki lub wierzchołki tego prostokąta, aby rozciągać lub zwężać obiekt jako ten prostokąt, wpływając zarówno na jego położenie, jak i skalowanie. Może to być przydatne do zwiększania lub zmniejszania obiektu tylko z jednej strony.

W środku gadżetu również znajduje się kółko, które można klikać i przeciągać, aby zmieniać położenie obiektu w płaszczyźnie tego prostokąta. Można zauważyć, że narzędzie prostokąta działa jednocześnie na dwóch osiach (czyli na wyznaczonej przez nie płaszczyźnie). Próba obrócenia widoku sceny powoduje ustawienie tego prostokąta w innej płaszczyźnie, tak aby zawsze był zwrócony w kierunku naszej kamery, przez którą patrzymy na scenę.

Klawisz **Y** włącza narzędzie łączące narzędzia W, E i R, pokazując jednocześnie strzałki do przemieszczania obiektu, okręgi do jego obracania i sześcian na środku do jego skalowania.

Położenie i osie

Jak działa pozycjonowanie obiektu w przestrzeni trójwymiarowej? Może to wymagać nieco przyzwyczajenia, ale położenie w przestrzeni trójwymiarowej jest definiowane przez trzy wartości liczbowe określane jako współrzędne X, Y i Z:

- Współrzędna **X** określa położenie obiektu *na prawo i lewo*.
- Współrzędna **Y** określa położenie obiektu *w górę i w dół*.
- Współrzędna **Z** określa położenie obiektu *w przód i w tył*.

Te współrzędne są często zapisywane razem jako (X, Y, Z). Na przykład położenie (15, 20, 25) oznacza, że współrzędna X ma wartość 15, współrzędna Y ma wartość 20, a współrzędna Z ma wartość 25. Położenie (0, 0, 0) oznacza „środek świata”, czyli punkt środkowy naszej sceny. Jeśli dodamy 5 do współrzędnej X naszego położenia, to przemieścimy się o 5 jednostek w prawo. Jeśli odejmiemy 5 od współrzędnej X naszego położenia, to przemieścimy się o 5 jednostek w lewo. Podobnie działa to z wartościami Y i Z:

dodanie wartości przesuwa nas w jednym kierunku, a odjęcie w przeciwnym kierunku. Zwiększanie wartości współrzędnej Y przemieści nas w górę, a jej zmniejszanie przemieści nas w dół. Zwiększanie wartości współrzędnej Z przemieści nas do przodu, a jej zmniejszanie przemieści nas do tyłu.

Połączenie tych trzech wartości definiuje, gdzie w naszym świecie coś się znajduje. Wartość każdej z tych trzech wartości oznacza położenie obiektu względem którejś z osi wyznaczających przestrzeń trójwymiarową. Możemy usłyszeć o „osi X", „osi Y", „osiach X i Z", itp.

Skalowanie i obracanie działa w podobny sposób: również dokonuje się względem tych samych trzech osi, z których każda wyznacza inny kierunek w przestrzeni trójwymiarowej. Skalowanie względem osi **X** określa *szerokość* – rozciągnięcie obiektu w lewo i prawo. Skalowanie względem osi **Y** określa *wysokość* – rozciągnięcie obiektu w górę i w dół. Skalowanie względem osi **Z** określa *długość* – rozciągnięcie obiektu w przód i w tył.

Jestem pewien, że łatwiej teraz przyjdzie wyobrażenie sobie, że obracanie działa w bardzo podobny sposób. Obrócenie obiektu jest definiowane przez trzy wartości kątowe pomiędzy 0 i 360 stopni, które określają obrót obiektu wokół każdej z osi.

Można zauważyć, że narzędzia używane do przemieszczania, obracania i skalowania obiektów (odpowiednio W, E i R) używają kodu kolorystycznego. Oś X jest zawsze czerwona, oś Y jest zawsze zielona, a oś Z jest zawsze niebieska.

Jest to powszechnie stosowany kod. Jeśli ktoś będzie się zajmował modelowaniem trójwymiarowym, to zobaczy analogiczne rozwiązania – choć niektóre programy traktują oś Y jako kierunek w przód i w tył, a oś Z jako kierunek w górę i w dół (czyli inaczej niż w przypadku Unity).

Tworzenie podłoża

Wykorzystajmy to, czego nauczyliśmy się do tej pory, do utworzenia kilku sześcianów, rozmieszczenia ich na scenie i przeskalowania. Najpierw jednak utworzymy podłoże dla naszego przykładu.

Utwórzmy płaszczyznę, wykorzystując tę samą metodę, co wcześniej w przypadku tworzenia sześcianu: GameObject ➤ 3D Object (obiekt trójwymiarowy) ➤ Plane (płaszczyzna).

Płaszczyzna jest jak jeden bok sześcianu – jest cienką, płaską powierzchnią, która nie ma żadnej grubości. Płaszczyzny są jednostronne: to znaczy można je oglądać tylko z jednej strony, a nie widać ich od tyłu. Możemy spróbować ustawić swoją kamerę pod

płaszczyzną i popatrzyć w górę na nią od dołu. Nic nie będzie widać, tak jakby płaszczyzna w ogóle nie istniała. Będzie się ona jednak świetnie nadawać na podłoże naszej gry, gdyż nie spodziewamy się, że będziemy na nie patrzeć od dołu.

Ponieważ dokładnie wiemy, gdzie chcemy mieć podłoże, możemy ustawić jego położenie, korzystając z okna Inspector. Gdy zaznaczony będzie nowy obiekt Plane, przyjrzymy się jego składnikowi Transform (przekształcanie) w oknie Inspector.

Jak wspominaliśmy wcześniej, składnik Transform zawiera współrzędne Position (położenie – gdzie obiekt się znajduje), Rotation (obrócenie – jak obiekt jest obrócony) i Scale (skalowanie – jak jest duży bądź mały).

Pamiętajmy, że głównym zadaniem okna Inspector jest interakcja ze składnikami, a nie jedynie przeglądanie ich danych. Udostępnia nam więc ono wszystkie faktyczne wartości położenia, obrócenia i skalowania w składniku Transform. Możemy edytować wartości dla poszczególnych osi, po prostu klikając te pola i wpisując żądane liczby.

Jest to przydatny sposób konfigurowania obiektów, jeśli dokładnie wiemy, jak je chcemy przekształcić, ponieważ uzyskanie dokładnych wartości położenia i obrotu przy użyciu narzędzi przekształcania może być dość uciążliwe. Chcemy, aby nasza płaszczyzna przechodziła przez środek sceny, więc użyjemy okna Inspector, aby zmienić jej położenie na wartość 0 dla wszystkich trzech osi. Jeśli chodzi o obrót (pole Rotation), powinien być już ustawiony na (0, 0, 0), więc zostawiamy tak, jak jest.

Skalowanie i jednostki miary

Teraz przejdźmy do skalowania. Można by zapytać, jaka jest jednostka odległości w naszej przestrzeni. Co to faktycznie oznacza, gdy zmieniamy położenie elementu GameObject z 0 na 1? Jaka to właściwie odległość?

Może to być nieco zagmatwane dla niektórych. Ktoś mógłby oczekiwać jasnej odpowiedzi. Programiści Unity zdecydowali, co to oznacza, prawda? Czy to stopa, metr, czy może jard?

To jednak tak nie działa. Nie należy się jednak martwić; i tak jest to dość proste. Sami musimy zdecydować, czym dla nas będzie jednostka. Powiedzmy, że zdecydujemy, że 1 jednostka oznacza 1 metr. Niech tak będzie. O ile będziemy się do tego stosować we wszystkich pomiarach, to właśnie będzie oznaczać 1 jednostka. Możemy ustawiać wysokość osób na ok. 1.5 – 1.9 jednostek (w tekście będziemy stosować angielską konwencję zapisu ułamków dziesiętnych – z kropką, a nie przecinkiem – żeby nie wprowadzać niejasności przy komentowaniu kodu, gdzie wymagany jest taki zapis liczb). Jeśli chcielibyśmy, żeby coś miało długość 1 decymetra, ustawimy to na jedną dziesiątą

jednostki (0.1). Jeśli będziemy chcieli, żeby coś miało 30 cm, ustawimy jego wielkość na 0.3 jednostki.

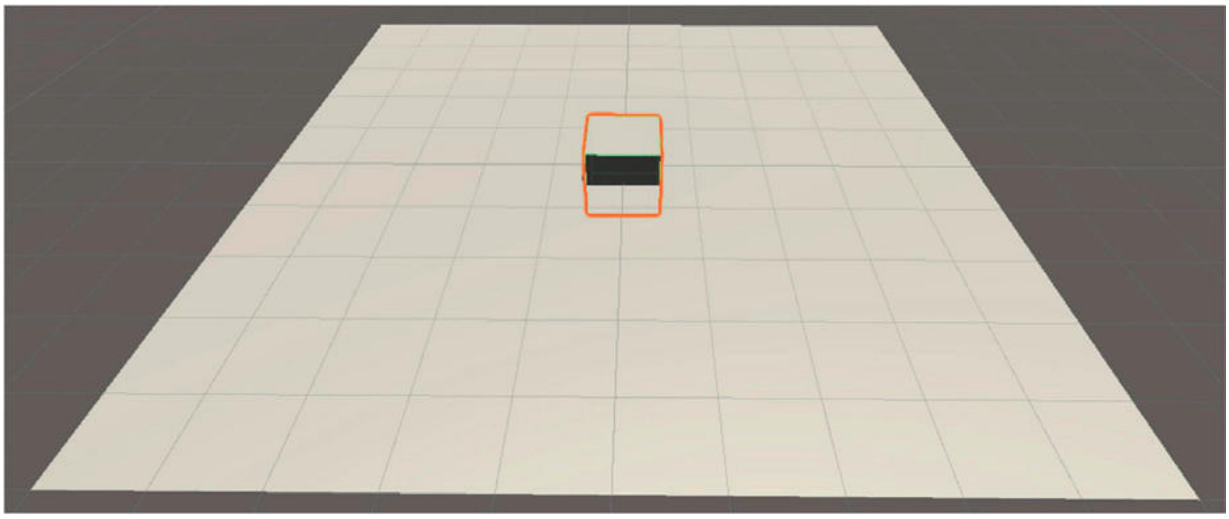
Jeśli chodzi o skalowanie w składniku Transform, to skala nie oznacza tego „ile coś ma jednostek długości, szerokości czy wysokości". Jest to w istocie *mnożnik*. Określa, jak ma zostać przemnożony rozmiar zdefiniowany w siatce obiektu (modelu trójwymiarowym).

Sama siatka ma zdefiniowany swój własny rozmiar, a następnie wartość skalowania (pola Scale) w składniku Transform po prostu przemnażają ten rozmiar.

Świetnie się to sprawdza w przypadku sześcianu. Siatka sześcianu definiuje sześcian o boku równym 1 jednostce (szerokości, wysokości i długości). Jeśli więc ustawimy jego skalę na 5 dla wszystkich osi, to będzie 5 razy większy na każdej osi, czyli każdy wymiar będzie wynosił 5 jednostek.

Z płaszczyzną sprawa jest nieco bardziej skomplikowana. Sama siatka definiuje płaską powierzchnię o szerokości i długości 10 jednostek (i całkiem cienką, czyli bez żadnej wysokości). Gdy więc mamy ustawione skalowanie (1, 1, 1) dla siatki płaszczyzny, to jest ona szeroka i długa na 10 jednostek.

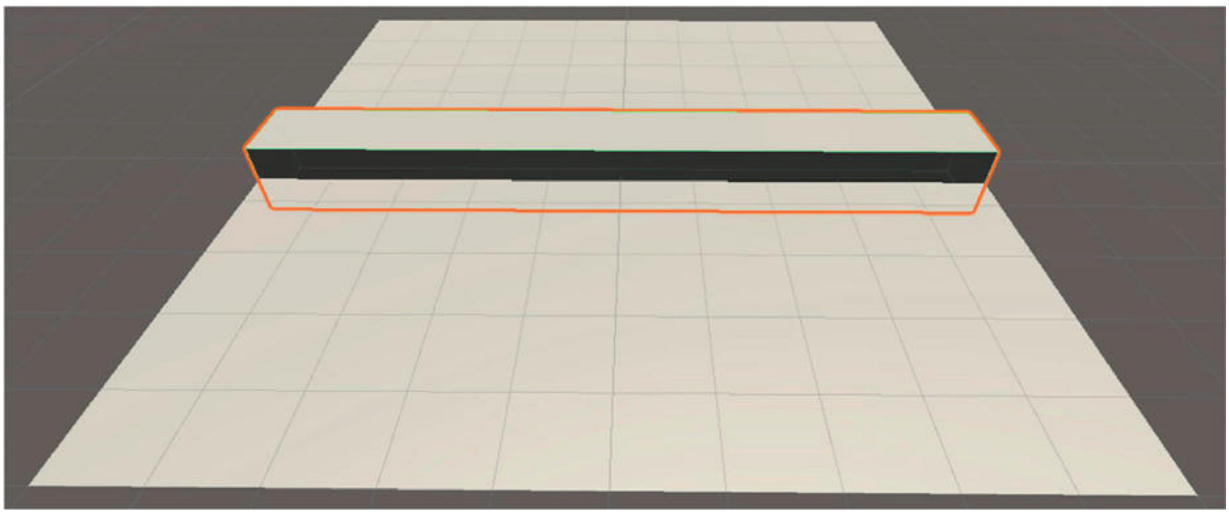
Możemy to zaobserwować, jeśli utworzymy domyślny obiekt Plane i Cube, pozostawimy dla każdego z nich domyślną wartość skalowania (1, 1, 1) i umieścimy je oba w tym samym miejscu. Można zobaczyć wtedy porównanie wielkości płaszczyzny i sześcianu, jak pokazano na rysunku 3–2.



Rysunek 3-2. *Obiekt Plane (płaszczyzna) i Cube (sześcian) ze skalowaniem ustawionym na (1, 1, 1) i umieszczone w dokładnie tym samym miejscu*

Choć skalowanie jest takie samo, to rozmiary obiektów są inne, ponieważ rozmiary ich faktycznych siatek nie są takie same. Szerokość i długość siatki sześcianu wynosi 1 jednostkę, natomiast szerokość i długość siatki płaszczyzny wynosi 10 jednostek. Ponieważ skalowanie określa jedynie mnożnik dla rozmiaru siatki, a nie określa faktycznego rozmiaru obiektu, to skala (1, 1, 1) nie wpływa w ogóle na rozmiar siatki. Jest to mnożenie przez 1, czyli wszystko zostaje bez zmian.

Gdybyśmy zmienili wartość skalowania dla sześcianu na (10, 1, 1), to będzie miał on teraz 10 jednostek szerokości, czyli tyle samo, co płaszczyzna, jak pokazano na rysunku 3–3.



Rysunek 3-3. *Płaszczyzna ze skalowaniem (1, 1, 1) w tym samym miejscu co sześcian ze skalowaniem (10, 1, 1). Szerokości sześcianu i płaszczyzny są takie same*

Podsumowując, warto zapamiętać: siatka ma swój własny rozmiar, a skalowanie jest po prostu mnożnikiem dla rozmiaru siatki. **Nie** jest to bezwzględna wartość określająca rozmiar obiektu.

Przejdźmy więc dalej. Chcemy mieć duże podłoże, żebyśmy nie musieli się później martwić jego powiększaniem, gdy będziemy chcieli umieścić na nim więcej elementów. Ustawmy skalę 10 na osiach X i Z – co w efekcie utworzy płaszczyznę o szerokości i długości 100 jednostek. Oczywiście najłatwiej po prostu wpisać wartości 10 w polach X i Z dla Scale (skalowanie) w oknie Inspector niż korzystać z narzędzia skalowania.

Podsumowanie

W tym rozdziale nauczyliśmy się następujących rzeczy:

- Jak manipulować położeniem, obróceniem i skalowaniem elementów GameObject przy pomocy narzędzi przekształcania (skróty klawiszowe W, E oraz R).
- Położenie jest reprezentowane przez wartości współrzędnych X, Y i Z. Dodanie wartości do współrzędnej przemieszcza obiekt w jednym kierunku, a odjęcie wartości od współrzędnej przemieszcza go w przeciwnym kierunku. X oznacza przemieszczenie w prawo (wartości dodatnie) i w lewo (wartości ujemne), Y oznacza przemieszczenie w górę (wartości dodatnie) i w dół (wartości ujemne), a Z oznacza przemieszczenie w przód (wartości dodatnie) i w tył (wartości ujemne). Łącząc wszystkie te trzy wartości, możemy zdefiniować punkt w przestrzeni trójwymiarowej.
- Skalowanie jest mnożnikiem dla rozmiaru faktycznego rozmiaru siatki renderowanego przez element GameObject. Nie oznacza to liczby jednostek szerokości, wysokości i długości elementu GameObject. Wartości skalowania X, Y i Z w składniku Transform oznaczają mnożniki dla odpowiednich rozmiarów renderowanej siatki.
- Pojedyncza jednostka w Unity nie odpowiada domyślnie jakiejś określonej liczbie centymetrów czy metrów. Musimy sami zdecydować, co jednostka będzie oznaczać dla nas i konsekwentnie trzymać się tej decyzji podczas ustawiania rozmiarów naszych obiektów i proporcji pomiędzy nimi.

Rozdział 4

Obiekty nadrzędne i podrzędne

Gdy już mamy przygotowane podłoże, zajmiemy się pewnymi, bardzo ważnymi pojęciami wykorzystywanymi przez silniki gier. Unity pozwala nam dołączać pojedyncze elementy GameObject do innych w systemie zależności, w którym kilka obiektów podrzędnych może być dołączonych do jednego obiektu nadrzędnego, dzięki czemu można je wszystkie przenosić, obracać i skalować razem z obiektem nadrzędnym. Powoduje to rozróżnienie pomiędzy dwoma sposobami patrzenia na położenie obiektu – jego *położeniem w świecie*, które oznacza, gdzie się on znajduje na scenie i jego *położeniem lokalnym*, które oznacza, gdzie się on znajduje w odniesieniu do swojego obiektu nadrzędnego. Daje nam to też możliwość takiego definiowania obiektów nadrzędnych i podrzędnych, w którym ustawiamy *punkt obrotu*, zmieniając punkt, wokół którego obiekty są obracane.

Podrzędne obiekty GameObject

Istnieje powód, dla którego okno Hierarchy nosi taką nazwę. Za chwilę wyjaśnimy ten powód.

Element GameObject może przechowywać „wewnątrz siebie” dowolną liczbę innych elementów GameObject. Jest to system zależności między obiektami: jeden element GameObject może być *obiektem nadrzędnym* dla wielu innych, a te przechowywane w nim elementy GameObject są nazywane jego *obiektami podrzędnymi*.

Technicznie Unity traktuje to jako składniki Transform dołączone do siebie, ponieważ w tym właśnie składniku widać znaczenie tego pojęcia. Dołączenie jednego