PRAKTYCZNE TWORZENIE GIER UNITY® i BLENDER™

Alan Thorn

Tytuł oryginału: Practical Game Development with Unity and Blender

Tłumaczenie: Zbigniew Waśko

ISBN: 978-83-283-0269-3

Wydawnictwo HELION
ul. Kościuszki 1c, 44-100 GLIWICE
tel. 32 231 22 19, 32 230 98 63
e-mail: *helion@helion.pl*
WWW: *http://helion.pl* (księgarnia internetowa, katalog książek)

Pliki z przykładami omawianymi w książce można znaleźć pod adresem:
ftp://ftp.helion.pl/przyklady/unible.zip

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
http://helion.pl/user/opinie/unible
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

Spis treści

ROZDZIAŁ 2

Od Blendera do Unity

Prostota jest szczytem wyrafinowania.

— Leonardo da Vinci

Po przeczytaniu tego rozdziału powinieneś:

- umieć przygotować Blendera do współpracy z Unity;
- rozumieć zagadnienia związane z przechodzeniem od Blendera do Unity;
- zdawać sobie sprawę z trudności w importowaniu i eksportowaniu siatek;
- umieć obchodzić się z ostrymi krawędziami i modyfikatorem *Edge Split* (rozcinanie krawędzi);
- umieć eksportować modele w formacie FBX.

Niemal w każdej grze wideo mamy do czynienia z wyświetlaniem modeli złożonych z wielokątów i dlatego zwanych siatkami. Ich geometryczna forma jest definiowana za pomocą wierzchołków, krawędzi i ścianek. W ten właśnie sposób przedstawiane są w zasadzie wszystkie przedmioty widoczne w grze — od elementów uzbrojenia po ludzi i wszelkie potwory. Artyści tworzący na potrzeby gier mają do dyspozycji wiele rozmaitych programów ułatwiających generowanie takich siatek, ale my będziemy używać wyłącznie Blendera. Jest to darmowy i działający na wielu platformach program do modelowania obiektów trójwymiarowych; można go pobrać ze strony: *www.blender.org/*.

Uwaga

Jeśli chciałbyś zobaczyć konkretne przykłady tego, co potrafi Blender, obejrzyj krótki film pt. *Sintel* zamieszczony na stronie: *www.youtube.com/watch?v=eRsGyueVLvQ*.

Nie będę opisywał Blendera od podstaw. Jest mnóstwo darmowych i komercyjnych poradników oraz kursów obsługi tego programu kierowanych również do początkującego użytkownika. Można tam nie tylko dowiedzieć się wszystkiego na temat interfejsu tego programu i stosowanych w nim skrótów klawiszowych, ale również poznać podstawy modelowania, teksturowania itp.

Wskazówka

> Jeśli jeszcze nigdy nie używałeś Blendera albo potrzebujesz odświeżyć swoją wiedzę o nim, zanim przystąpisz do dalszej lektury tego rozdziału, to zajrzyj do dodatku A. W części zatytułowanej „Książki" znajdziesz kilka pozycji, które z pewnością ułatwią Ci wstępne poznanie tego programu.

Głównym tematem tego rozdziału jest współpraca Blendera z Unity, ale będzie też okazja, żeby wspomnieć o modelowaniu i teksturowaniu. Jednak zgodnie z tym, co przed chwilą powiedziałem, skoncentruję się przede wszystkim na praktycznym aspekcie współdziałania tych dwóch programów. W szczególności pokażę, jak można skonfigurować interfejs Blendera, aby dało się w nim pracować szybciej i wydajniej. Następnie zobaczysz, na czym polega optymalny sposób eksportowania wymodelowanych siatek do programu Unity z zachowaniem odpowiedniego cieniowania, rozmiarów i struktury. Jeśli w tym momencie nie wiesz za bardzo, o czym mówię, nie przejmuj się. Wkrótce wszystko się wyjaśni.

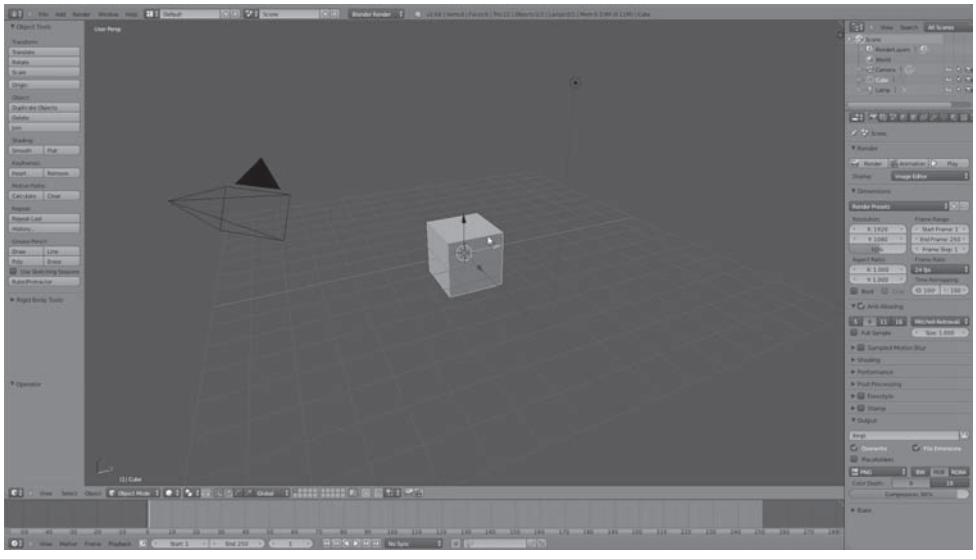
Konfigurowanie interfejsu Blendera

Gdy uruchomisz Blendera po raz pierwszy i nie zmienisz ustawień domyślnych, jego interfejs będzie wyglądał mniej więcej tak jak na rysunku 2.1. Mówię „mniej więcej", a nie „dokładnie", bo Blender jest dość często aktualizowany i jest niemal pewne, że wersja używana przez Ciebie będzie nowsza od mojej, a zatem możesz mieć przed sobą nieco inny widok.

Wielu ludziom odpowiadają te domyślne ustawienia i używają oni Blendera bez zmieniania czegokolwiek. Jednak ja się do nich nie zaliczam. Wprowadzam zmiany, i to dość znaczące. Opiszę je wszystkie wraz z uzasadnieniem, abyś mógł w pełni ocenić, czy moje podejście jest godne naśladowania, czy nie. Oto te modyfikacje.

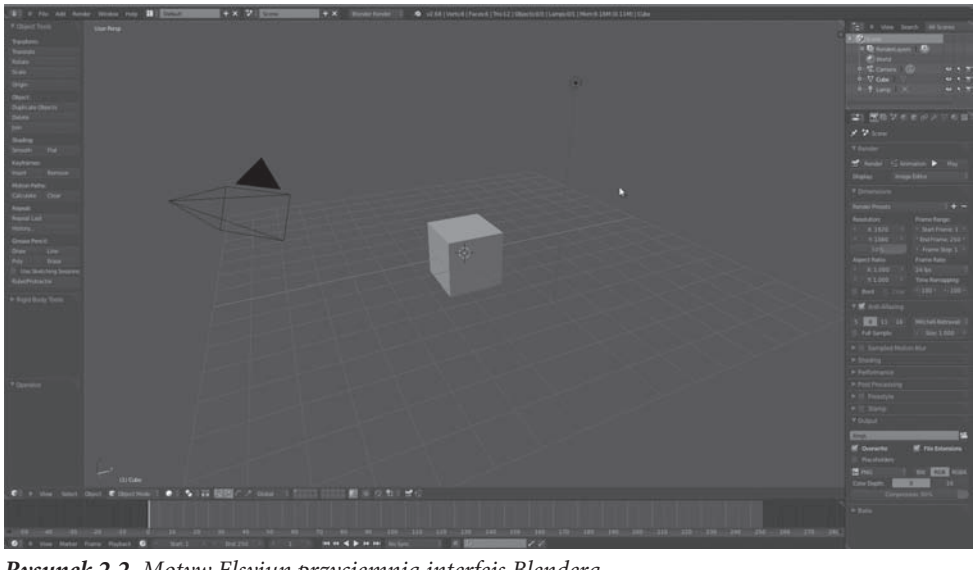
Ciemny motyw

Kolorystyka interfejsu Blendera jest zdefiniowana w postaci zestawów barw zwanych **motywami**. Motyw domyślny zawiera średnie odcienie szarości pokazane na rysunku 2.1. Stwierdziłem, że przy dłuższej pracy taki interfejs działa jednak rozpraszająco — oko ludzkie w naturalny sposób kieruje się w stronę miejsc jaśniejszych. A skoro naszą uwagę przyciąga interfejs, to trudniej się nam skupić na opracowywanym modelu. Aby temu zaradzić, przyciemniam interfejs, włączając motyw *Elsyiun* pokazany na rysunku 2.2.



Rysunek 2.1. *Blender w konfiguracji domyślnej (wersja 2.68a). Jeśli masz wersję późniejszą, jej interfejs może wyglądać nieco inaczej*

Źródło: Blender.



Rysunek 2.2. *Motyw Elsyiun przyciemnia interfejs Blendera*

Źródło: Blender.

Aby włączyć ten motyw kolorystyczny, wykonaj następujące czynności:

1. Wybierz *File/User Preferences* (plik/preferencje użytkownika) lub wciśnij klawisze *Ctrl+Alt+U*.
2. W oknie dialogowym *Blender User Preferences*, które się otworzy, kliknij zakładkę *Themes* (motywy).

3. Rozwiń listę *Presets* (ustawienia predefiniowane) i wybierz pozycję o nazwie *Elsyiun*, tak jak pokazano na rysunku 2.3. Kolory interfejsu natychmiast się zmienią.

Rysunek 2.3. *W oknie dialogowym Blender User Preferences wybierz motyw o nazwie Elsyiun*
Źródło: Blender.

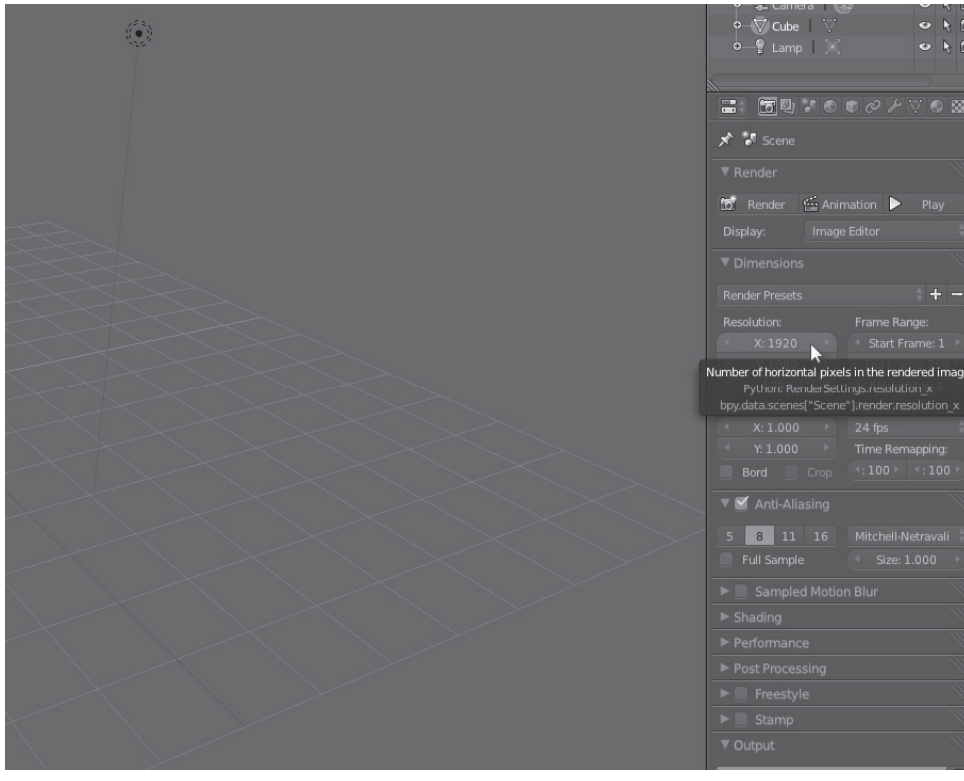
4. W celu zatwierdzenia wprowadzonej zmiany kliknij przycisk *Save User Settings* (zapisz ustawienia użytkownika) widoczny w dolnej części okna preferencji. Następnie zamknij to okno, aby wrócić do głównej przestrzeni roboczej.

Jeśli później uznasz, że jednak motyw domyślny był lepszy, możesz go przywrócić przez kliknięcie przycisku *Reset to Default Theme* (przywróć motyw domyślny) w oknie *Blender User Preferences*.

Etykietki bez pythonowych instrukcji

Blendera używają zarówno artyści tworzący modele, jak i programiści piszący skrypty rozwijające możliwości programu. Dlatego w ustawieniach domyślnych starano się uwzględnić potrzeby jednych i drugich. Jednak my, twórcy gier, będziemy działać głównie jako artyści, więc niepotrzebne nam będą elementy przydatne programistom.

Jednym z takich elementów są instrukcje w języku Python (język programowania dostępny w Blenderze jako język skryptowy) wyświetlane w etykietkach narzędziowych, które ukazują się, gdy przytrzymasz wskaźnik myszy na jakimś przycisku lub innym elemencie interfejsu (rysunek 2.4). Jako artyście informacje takie nie będą Ci potrzebne. Możesz je śmiało ukryć, aby dodatkowy tekst nie angażował bez potrzeby Twojej uwagi.



Rysunek 2.4. *Przy domyślnych ustawieniach Blendera w etykietkach narzędziowych wyświetlane są informacje zarówno dla artystów, jak i programistów*

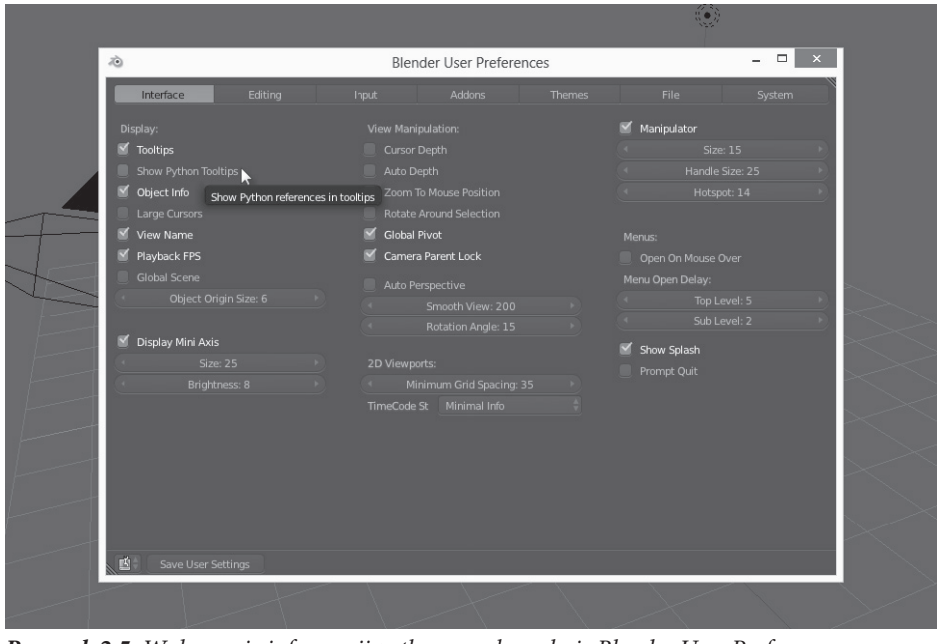
Źródło: Blender.

Aby wyłączyć wyświetlanie informacji pythonowych, wykonaj następujące czynności:

1. Otwórz okno dialogowe *Blender User Preferences* przez wybranie polecenia *File/User Preferences*.
2. Kliknij zakładkę *Interface* (interfejs).
3. Wyłącz opcję *Show Python Tooltips* (pokaż etykietki narzędziowe Pythona) — rysunek 2.5.
4. Kliknij przycisk *Save User Settings* (zapisz ustawienia użytkownika). (Jeśli tego nie zrobisz, dawne ustawienia powrócą po ponownym uruchomieniu programu).

Kontrolki z programu Maya

Dla mnie największym usprawnieniem w pracy z Blenderem i Unity jest możliwość wykonywania wielu czynności w Blenderze w sposób podobny jak w programie Maya. Przy ustawieniach domyślnych Blender oferuje własny zestaw skrótów klawiszowych i kontrolek do nawigowania w oknach widokowych. Niestety, wszystko odbywa się tu inaczej niż w programie Unity, który pod tym względem nie różni się od programu Maya.



Rysunek 2.5. *Wyłączanie informacji pythonowych w oknie Blender User Preferences*
Źródło: Blender.

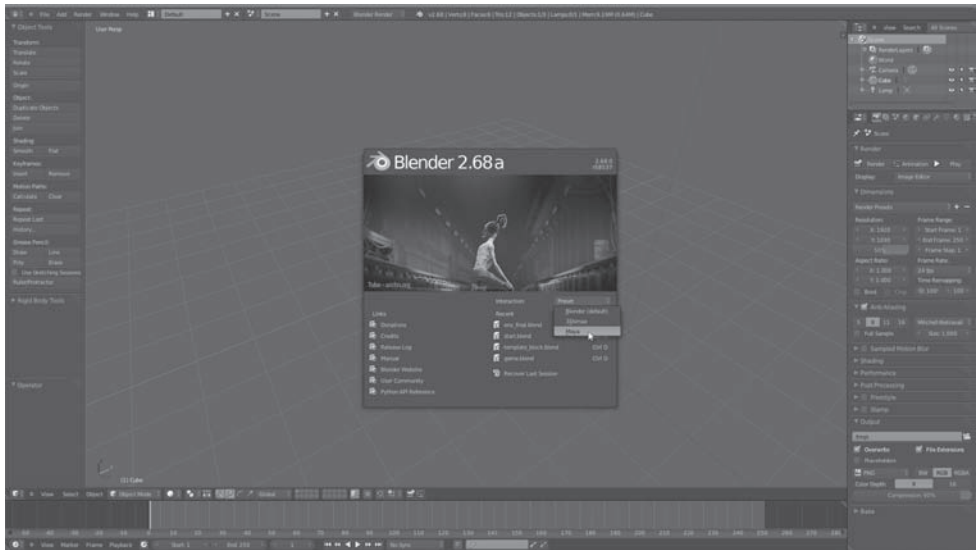
Ponieważ często będziemy musieli używać Blendera na przemian z Unity, byłoby dobrze, gdyby dało się uniknąć częstego przestawiania się z jednego systemu sterowania na inny. Z doświadczenia wiem, że bez ujednolicenia tych systemów łatwo o pomyłkę, a zastosowanie skrótu blenderowego w Unity (lub na odwrót) nie wróży nic dobrego. Rozwiązanie tego problemu tkwi w predefiniowanym ustawieniu o nazwie *Maya*, za pomocą którego można sprawić, że Blender będzie działał tak jak Unity. Po uaktywnieniu tego ustawienia otrzymujemy jeden spójny system sterowania obowiązujący w obu aplikacjach.

Uwaga

Ktoś może zapytać, dlaczego upodobniłem Blendera do Unity, a nie na odwrót. Powód jest wręcz banalny: po prostu Blender jest pod względem konfigurowalności bardziej elastyczny i nawet ma już wbudowane gotowe ustawienia, więc po co się męczyć, skoro można to zrobić szybko i skutecznie.

Zmianę systemu sterowania Blenderem można przeprowadzić na dwa sposoby. Pierwszy polega na wybraniu odpowiedniej opcji na ekranie powitalnym, a drugi wymaga posłużenia się oknem dialogowym *Blender User Preferences*. W pierwszym przypadku należy:

1. Uruchomić Blendera.
2. Na ekranie powitalnym rozwinąć listę *Interactions* (interakcje) i wybrać z niej opcję *Maya* — tak jak na rysunku 2.6.



Rysunek 2.6. *Wybieranie systemu sterowania Blenderem na ekranie powitalnym*
Źródło: Blender.

Wskazówka

> Ekran powitalny można wyświetlić również w trakcie pracy z Blenderem. W tym celu należy kliknąć ikonkę Blendera widoczną w nagłówku okna informacyjnego, który domyślnie znajduje się przy górnej krawędzi interfejsu (na rysunku 2.6 ikonka ta jest zaznaczona kółkiem).

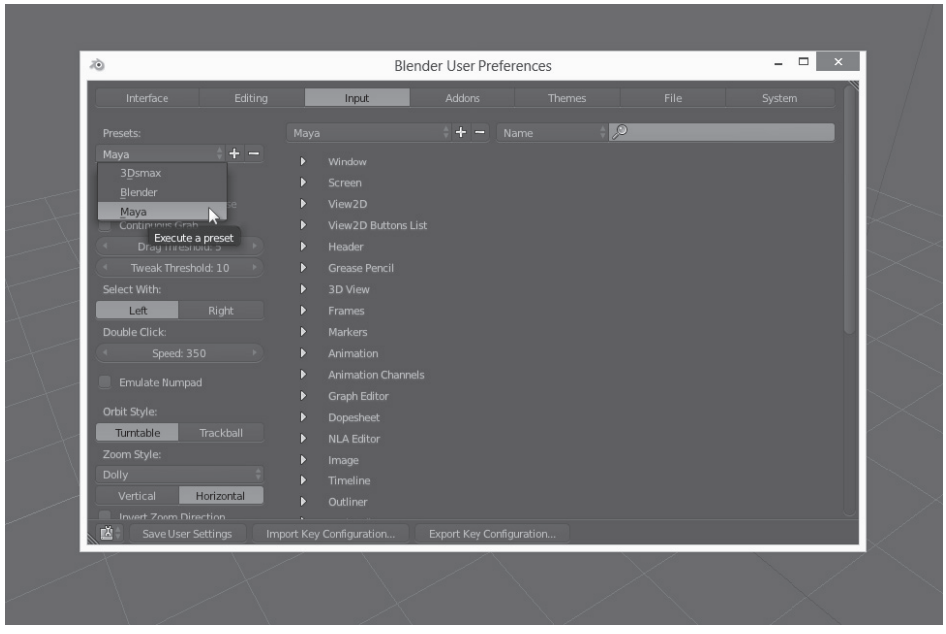
Drugi sposób wymaga następujących czynności:

1. Wybierz polecenie *File/User Preferences*.
2. W oknie dialogowym *Blender User Preferences*, które się otworzy, kliknij zakładkę *Input* (wejście).
3. Rozwiń listę *Presets* i wybierz pozycję o nazwie *Maya* (rysunek 2.7)[1].
4. W celu zatwierdzenia wprowadzonej zmiany kliknij przycisk *Save User Settings*.

„Błąd" zamykania bez zapisu

Zanim przejdziemy do bardziej zasadniczych spraw, chciałbym omówić kwestię pozornego błędu, który stał się już sławny w społeczności blenderowej. Otóż istnieje niepisane prawo, które mówi, że każda aplikacja użytkowa powinna reagować na polecenie zamknięcia odpowiednim komunikatem, jeśli użytkownik nie zapisał aktualnego

[1] Listy są dwie (na rysunku rozwinięta jest tylko jedna): jedna odnosi się do operacji wykonywanych za pomocą myszy, a druga do skrótów klawiszowych. Ekran powitalny ustawia obie listy w należyty sposób — *przyp. tłum.*



Rysunek 2.7. *Ustawianie systemu sterowania Blenderem w oknie dialogowym Blender User Preferences*

Źródło: Blender.

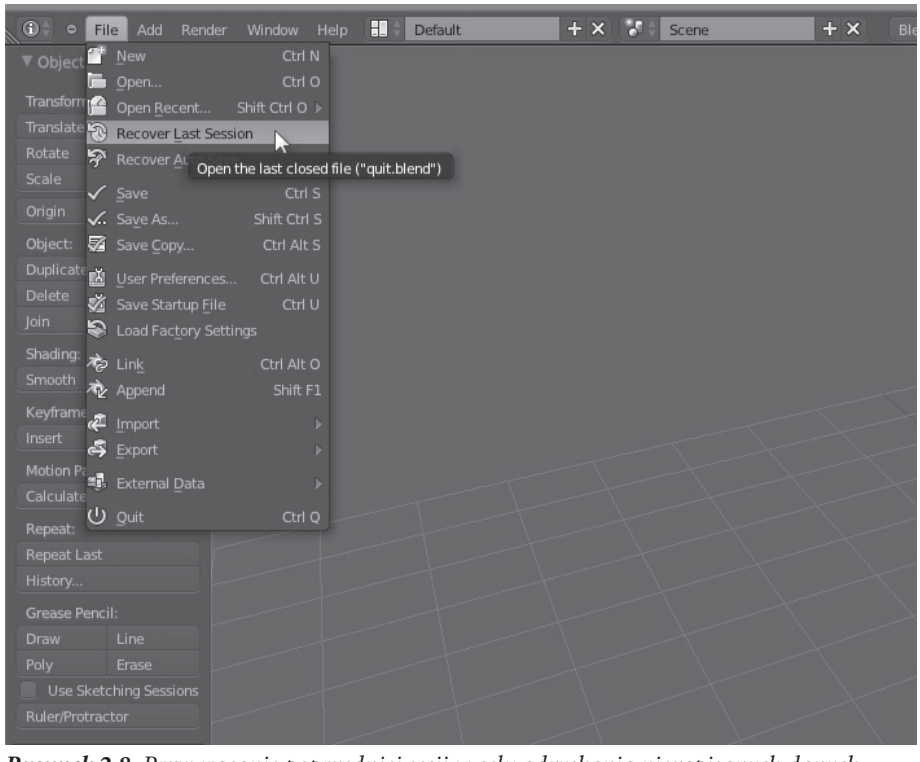
stanu swojej pracy. Niemal wszystkie programy, od pakietów Microsoft Office i LibreOffice aż po zaawansowane programy graficzne, takie jak Unity czy GIMP, w pełni respektują to prawo, a Blender nie! Gdy go zamykasz, po prostu znika z ekranu i robi tak niezależnie od tego, czy zapisałeś rezultat swojej pracy, czy nie. A gdy go uruchamiasz ponownie, jakby nigdy nic wyświetla nową, świeżą scenę i nadal nie daje żadnego znaku, że tamta niezapisana scena nadal istnieje i możesz do niej wrócić. Już niejeden użytkownik uznał, że wiele godzin pracy zostało bezpowrotnie stracone. W takich chwilach rzeczywiście można wpaść w furię, zwłaszcza jeśli sobie uświadomimy, że przecież można było wszystkiego uniknąć, gdyby tylko Blender „zachował się" w sposób konwencjonalny i wyświetlił stosowne ostrzeżenie.

Uwaga

Niektórzy próbowali uzasadniać „brak tej funkcji", twierdząc, że użytkownik Blendera powinien być profesjonalistą i dbać o rezultaty swojej pracy. Uważam jednak takie podejście za co najmniej niewłaściwe, ponieważ ignoruje ono fakt, że każdemu (również profesjonaliście) może się zdarzyć przypadkowe zamknięcie aplikacji bez zapisania opracowywanego pliku. Na szczęście problem jest tylko pozorny! Czytaj dalej…

Przekonanie, że Blender pozwala na utratę niezapisanych zmian, jest błędne. Jego twórcy przewidzieli takie sytuacje i zaimplementowali metodę odzyskiwania rezultatów poprzedniej sesji. Wystarczy po prostu wybrać polecenie *File/Recover Last Session*

(plik/odtwórz ostatnią sesję) — rysunek 2.8. Polecenie działa, ponieważ Blender tuż przed zamknięciem zapisuje bieżącą scenę w pliku *quit.blend* i wystarczy ten plik otworzyć. Wydaje się to całkiem proste (i tak jest — wszystko sprowadza się przecież tylko do kliknięcia odpowiedniego polecenia w menu), ale przeświadczenie o niedopracowaniu Blendera powoduje, że możliwość przywrócenia poprzedniej sesji często uchodzi uwadze użytkownika.



Rysunek 2.8. *Przywracanie poprzedniej sesji w celu odzyskania niezapisanych danych*
Źródło: Blender.

Być może to, co napisałem, nie do końca Cię przekonuje i nadal uważasz, że wzorem innych aplikacji Blender powinien wyświetlać ostrzeżenie o niezapisanych zmianach. Proponuję jednak odłożyć te dywagacje na bok. Nie piszę książki o tym, jakie funkcje dany program powinien zawierać, a jakich nie. Piszę o tym, jaki program jest i co można za jego pomocą zrobić. Sądzę, że tylko przy takim nastawieniu da się w pełni wykorzystać te możliwości programu, które w nim tkwią.

Przenoszenie modeli z Blendera do Unity

Jednym z zagadnień, które są najczęściej dyskutowane w rozmowach na temat toku pracy z użyciem Blendera i Unity, jest przenoszenie modeli wygenerowanych w Blenderze do silnika Unity. Popularną metodą jest zapisanie blenderowej sceny w pliku *.blend*

(natywny format Blendera) za pomocą polecenia *File/Save* (plik/zapisz), a następnie otwarcie tego pliku w Unity. Resztą zajmuje się już Unity — przynajmniej w teorii! Czasami — zależnie od okoliczności — otrzymasz dokładnie to, o co Ci chodziło, i nie musisz niczego poprawiać, ale najczęściej rezultaty będą trudne do zaakceptowania. Może się również zdarzyć, że ta prosta metoda w ogóle nie zadziała. Wyjaśnię teraz, dlaczego tak się dzieje, a potem na konkretnym przykładzie przedstawię właściwe rozwiązanie.

Uwaga

Na razie zajmujemy się eksportowaniem blenderowych modeli do Unity w sensie ogólnym. Pomijamy szczegóły dotyczące modeli animowanych, takich jak rigowane siatki postaci człekokształtnych lub innych stworów. Wrócimy do nich w dalszej części książki.

Pliki .blend

Blender domyślnie tworzy pliki w formacie *.blend*. Dlatego gdy wydasz mu polecenie *File/Save* (plik/zapisz), zapisze bieżącą scenę w pliku z rozszerzeniem *.blend*. Na szczęście Unity potrafi odczytywać takie pliki, więc możesz ich używać do przenoszenia danych z jednego programu do drugiego. Po prostu przeciągnij plik z Eksploratora lub Findera i upuść w obrębie interfejsu Unity.

Metoda wydaje się interesująca, ale jest kilka powodów, dla których zdecydowanie odradzam jej stosowanie.

- **Unity do zaimportowania pliku *.blend* potrzebuje Blendera.** Unity odczytuje pliki *.blend*, ale nie bezpośrednio. Tak naprawdę akceptuje tylko formaty: FBX, DAE, 3DS, DXF i OBJ. Gdy ma zaimportować dane z innego pliku, uruchamia w tle program macierzysty (Blendera dla pliku *.blend*) i za jego pomocą konwertuje ten plik na format FBX. A zatem każdy, kto chce importować pliki *.blend* do Unity, musi mieć zainstalowanego Blendera, nawet gdy nie ma zamiaru go używać. Unity potrzebuje go do przeprowadzenia konwersji. Wprawdzie Blender jest darmowy, ale już sama konieczność jego instalacji tylko po to, by otworzyć jakiś plik, może być irytująca — tym bardziej, że da się tego uniknąć (o tym za chwilę).
- **Pliki *.blend* ograniczają liczbę opcji eksportu i zwiększają ryzyko niekompatybilności.** Gdy Unity przeprowadza konwersję, nie masz na ten proces żadnego wpływu. Stosowane są wtedy ustawienia domyślne, które przecież nie zawsze są optymalne, i w rezultacie otrzymujesz niekoniecznie prawidłową siatkę. Oczywiście może się zdarzyć, że wszystko pójdzie dobrze, ale nie mając na nic wpływu, możesz liczyć tylko na szczęśliwy przypadek. Poza tym dochodzi jeszcze ryzyko wynikające z częstych modyfikacji samego Blendera. Jeśli do konwersji pliku zostanie użyta inna wersja niż ta, za pomocą której go wygenerowano, rezultat może być trudny do przewidzenia.

- **Pliki *.blend* są zbyt duże.** Nie jest to zarzut pod adresem tego formatu, lecz stwierdzenie, że przy ustawieniach domyślnych Unity z pomocą Blendera przeniesie do formatu FBX całą zawartość sceny, włącznie z obiektami, które nie powinny być importowane, np.: kamery, lampy, niewidoczne siatki, obiekty pozorne. W Blenderze są przydatne, ale ich import do Unity na ogół nie ma sensu.

Uwaga

Więcej informacji na temat formatów 3D i ich technicznych uwarunkowań znajdziesz w instrukcji obsługi programu Unity dostępnej pod adresem: *http://docs.unity3d.com/Documentation/Manual/3D-formats.html*.

Ćwiczenie: ręczny eksport do FBX

Po przeczytaniu powyższych uwag raczej nie będziesz przenosił modeli z Blendera do Unity za pomocą plików *.blend*. O ile na wczesnym etapie prac, podczas testowania siatek, możesz się jeszcze nimi posługiwać, o tyle do importowania gotowych zasobów powinieneś używać wyłącznie plików w formacie FBX — tym bardziej, jeśli będą to zasoby przekazywane innym członkom zespołu.

Jak już wspominałem, Unity posługuje się wewnętrznie formatem FBX nawet wtedy, gdy importuje pliki *.blend*. Ale jeśli sam wyeksportujesz z Blendera plik FBX, będziesz miał większą kontrolę nad jego zawartością niż podczas automatycznej konwersji pliku *.blend*. Ponadto uniezależnisz Unity od innych programów, bo z formatem FBX radzi sobie bez żadnej pomocy z zewnątrz.

W ćwiczeniu tym pokażę na przykładzie zwykłego stożka, jak takie eksportowanie należy przeprowadzać. Naturalnie Twoje modele będą bardziej rozbudowane i skomplikowane, ale wszystko, co tutaj powiem, ma zastosowanie do wszystkich modeli: prostych i złożonych. Więc zaczynajmy!

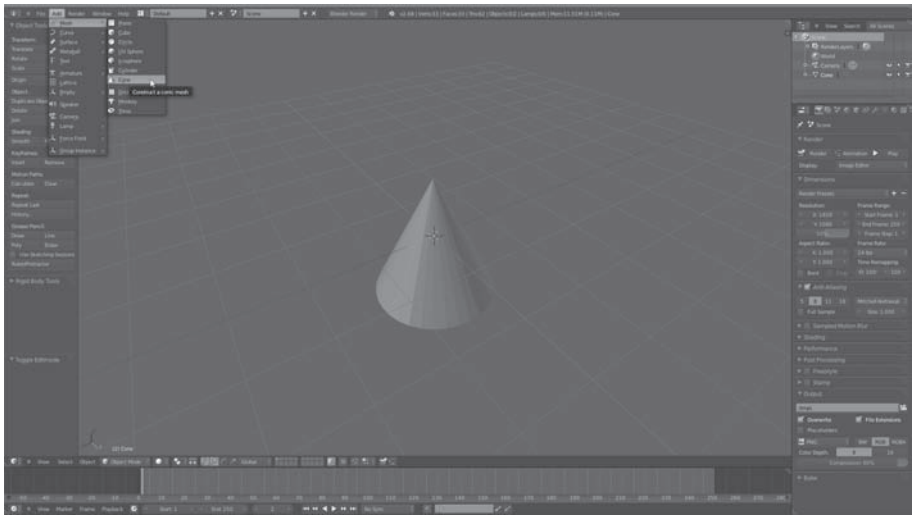
Tworzenie modeli

To ćwiczenie zaczniemy od zupełnie nowej sceny w Blenderze (zob. rysunek 2.1 lub 2.2). W codziennej pracy do eksportowania będziesz przystępował ze sceną zawierającą gotowe modele. Jednak tym razem usuwamy wszystko z wyjątkiem kamery. W pustej scenie umieść zwykły stożek — wybierz polecenie *Add/Mesh/Cone* (dodaj/siatkę/stożek). Będzie to model, który wyeksportujesz z przeznaczeniem umieszczenia go w Unity (rysunek 2.9).

Radzenie sobie z wadami cieni

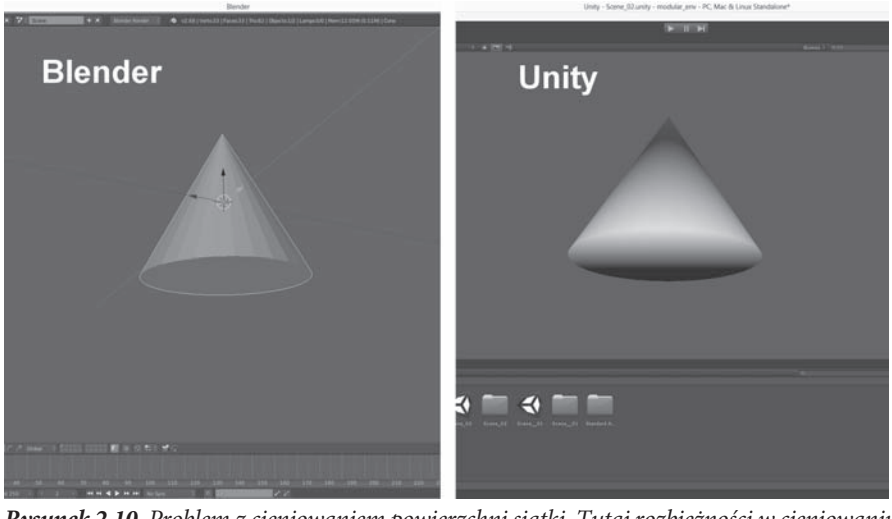
Uwaga

Ten punkt i następny zachowują ważność, pod warunkiem że używasz Blendera w wersji wcześniejszej niż 2.69. Właśnie w tym wydaniu aplikacji dodano funkcję dzielenia normalnych, która likwiduje omawiane tu usterki. Więcej informacji na ten temat znajdziesz pod adresem: *http://wiki.blender.org/index.php/Dev:Ref/Release_Notes/2.69*. Jeśli używasz wersji 2.69 lub nowszej, możesz od razu przejść do punktu zatytułowanego „Środek obiektu".



Rysunek 2.9. *W pustej scenie Blendera umieść stożek*
Źródło: Blender.

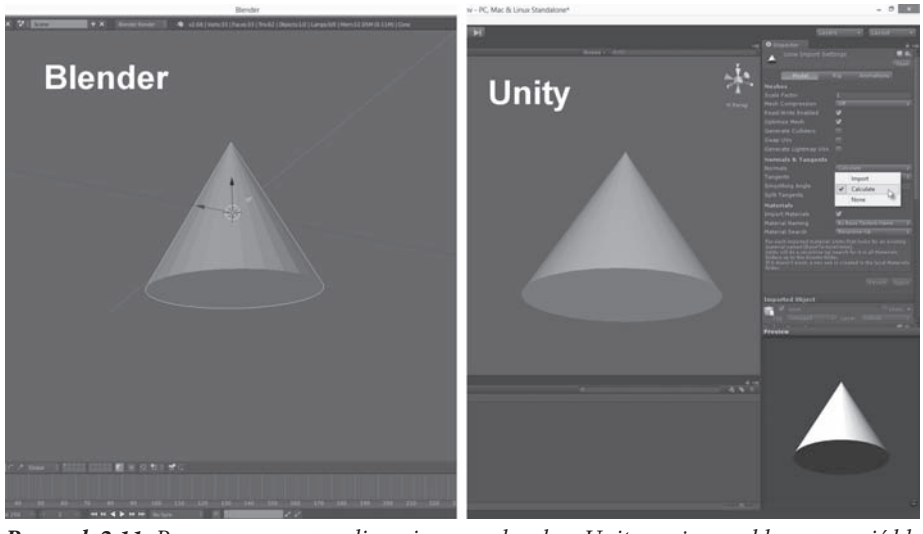
Idźmy dalej. Jeśli od razu przeniesiesz siatkę stożka z Blendera do Unity, czy to w pliku *.blend*, czy FBX (o którym więcej już za chwilę), najprawdopodobniej będziesz miał problem z wyrenderowaniem prawidłowych cieni na jej powierzchni. Zauważysz to, gdy tylko spojrzysz na siatkę w oknie widokowym Unity lub w polu podglądu na panelu *Inspector* (inspektor). Siatka zostanie wygładzona i będzie wyglądała dziwnie. Problem wynika z odmiennego podejścia obu programów do roli, jaką powinny odgrywać normalne. Rozbieżności w wyglądzie stożka wyświetlanego w Blenderze i w Unity są pokazane na rysunku 2.10.



Rysunek 2.10. *Problem z cieniowaniem powierzchni siatki. Tutaj rozbieżności w cieniowaniu między Blenderem i Unity najlepiej widać przy krawędzi łączącej powierzchnię boczną stożka z jego podstawą*
Źródło: Blender.

Uwaga

Dalszy ciąg tego punktu przedstawia sytuację hipotetyczną, więc nie musisz tego czytać. Chciałem Ci tylko pokazać, na jaki problem możesz natrafić, jeśli postąpisz zgodnie z podanym opisem. Innym efektem, jaki napotkasz, będą dużo mniejsze wymiary siatki w Unity aniżeli w Blenderze. Wynika to z domyślnie przyjmowanej przez Unity skali o wartości 0,01. Możesz się o tym przekonać, sprawdzając tę wartość w panelu *Inspector*, gdy siatka jest zaznaczona (rysunek 2.11). Aby to naprawić, po prostu zwiększ parametr *Scale Factor* (współczynnik skali) do poziomu 1. (Zagadnienie to będzie jeszcze omawiane).



Rysunek 2.11. *Przez ponowne przeliczenie normalnych w Unity można szybko naprawić błędy cieniowania*

Źródło: Blender.

W przypadku modeli tak prostych jak sześciany, stożki czy kule problem cieniowania można szybko rozwiązać w Unity. Trzeba tylko zaznaczyć właściwą siatkę w panelu *Project* (projekt), po czym w panelu *Inspector* wybrać z listy rozwijanej *Normals* (normalne) opcję *Calculate* (oblicz). (Nie zapomnij kliknąć potem przycisku *Apply*, czyli „zastosuj", aby zatwierdzić wprowadzone zmiany). Unity na podstawie kątów między ściankami ustali, które krawędzie mają być wygładzone, a które powinny pozostać ostre. W rezultacie otrzymasz stożek wyglądający niemal tak samo jak w Blenderze (zob. rysunek 2.11). Możesz zatem powiedzieć, że między aplikacjami występuje relacja podobieństwa 1:1.

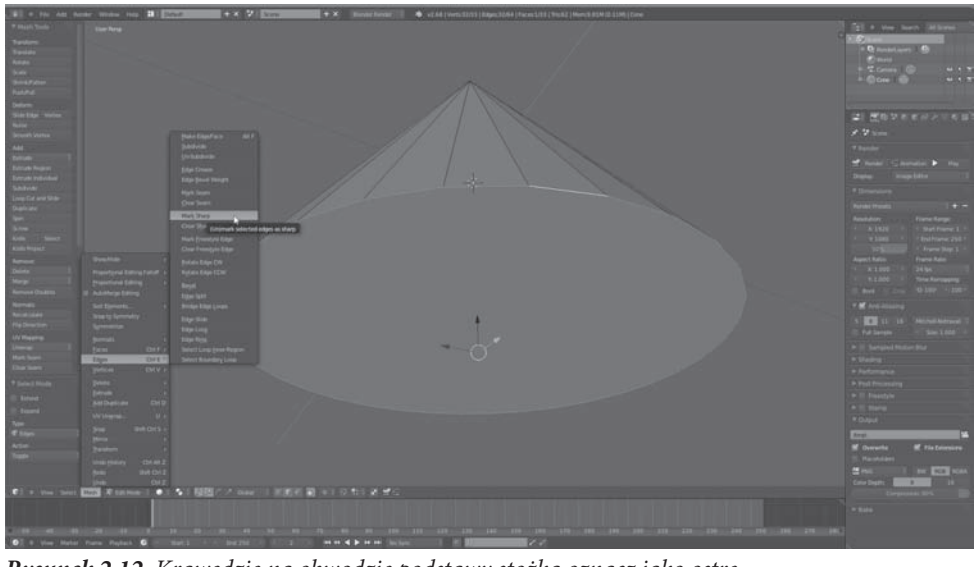
Powyższa metoda „naprawiania" błędów cieniowania ma jednak istotne ograniczenia. Jest to działanie automatyczne i nie masz żadnego wpływu na to, które krawędzie zostaną zacieniowane ostro, a które łagodnie. W przypadku stożka jest to wystarczające, ale przecież nie tylko takie siatki chciałbyś importować. Dlatego musimy wrócić do Blendera i tam odpowiednio przygotować model do eksportu. A zatem, jeśli wraz ze mną przenosiłeś stożek za pośrednictwem pliku *.blend*, usuń go teraz z Unity i ponownie otwórz Blendera.

Ostrość krawędzi i modyfikator Edge Split

W poprzednim punkcie pokazałem jeden sposób na rozwiązanie problemu złego cieniowania, a teraz zaprezentuję metodę pewniejszą i najczęściej stosowaną.

Jeśli chcesz, żeby po zaimportowaniu do Unity siatka wyglądała prawidłowo i nie ufasz zbytnio mechanizmom przeliczania normalnych w tej aplikacji, musisz jeszcze w Blenderze wskazać dokładnie, które krawędzie mają pozostać ostre. **Ostra krawędź** to takie miejsce na powierzchni siatki, gdzie spotykają się dwie ścianki, a ich cieniowania pozostają całkowicie odrębne — nie są w żaden sposób uśredniane. Zazwyczaj są to kanty takich brył jak sześcian czy graniastosłup, ale mogą to być także nagłe załamania ścian pod kątem 90°. Natomiast większość obiektów o kształtach organicznych, np. ludzka twarz, ma powierzchnie gładkie i nie ma tam miejsca na ostre krawędzie.

W przykładowym stożku powinieneś wskazać jako ostre krawędzie tworzące obwód jego podstawy, tam bowiem łączą się dwie zupełnie odrębne powierzchnie. Zaznacz więc cały obwód podstawy, po czym wybierz polecenie *Mesh/Edges/Mark Sharp* (siatka/krawędzie/oznacz jako ostre) — tak jak na rysunku 2.12.

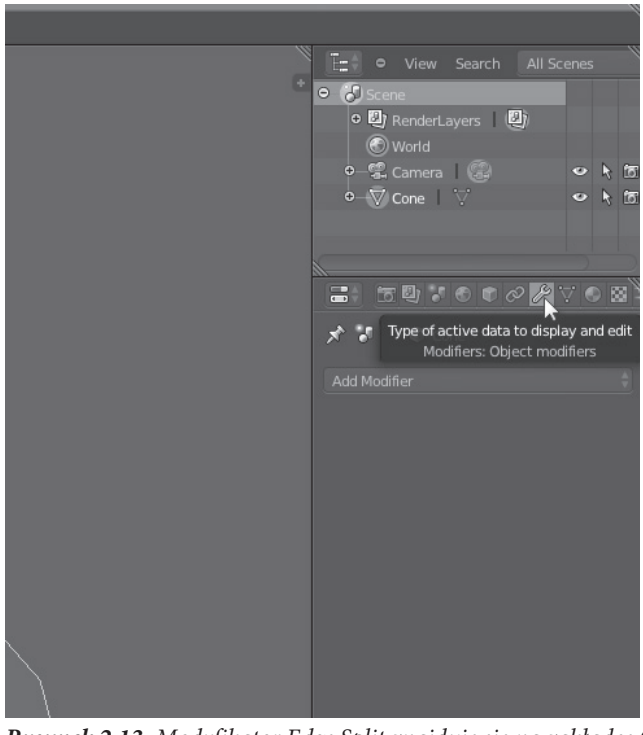


Rysunek 2.12. *Krawędzie na obwodzie podstawy stożka oznacz jako ostre*
Źródło: Blender.

Wskazanie krawędzi, które powinny być ostre, to dopiero pierwszy krok. Teraz trzeba jeszcze poddać siatkę działaniu modyfikatora o nazwie *Edge Split* (rozcinanie krawędzi), który porozcina siatkę wzdłuż ostrych krawędzi i w ten sposób wymusi na Unity odrębne cieniowanie rozłącznych powierzchni. Nie bój się, że przez taki zabieg zniszczysz siatkę, bo wszystko dzieje się w sposób odwracalny — modyfikator działa w Blenderze i zawsze możesz go wyłączyć, a wtedy siatka powróci do stanu pierwotnego (z nierozciętymi krawędziami).

Aby zastosować modyfikator *Edge Split*, wykonaj następujące czynności:

1. Zaznacz siatkę stożka.
2. Otwórz zakładkę *Object modifiers* (modyfikatory obiektu) panelu *Properties* (właściwości) — rysunek 2.13.

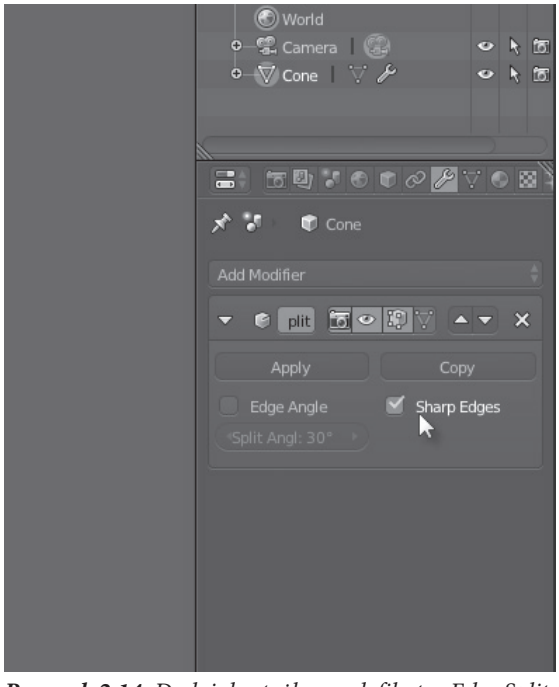


Rysunek 2.13. *Modyfikator Edge Split znajduje się na zakładce Object modifiers*
Źródło: Blender.

3. Na zakładce *Object modifiers* rozwiń listę *Add Modifier* (dodaj modyfikator) i wskaż opcję *Edge Split*. Modyfikator o takiej właśnie nazwie zostanie przypisany do obiektu stożka.
4. W właściwościach modyfikatora wyłącz opcję *Edge Angle* (kąt krawędzi), ale pozostaw włączoną funkcję *Sharp Edges* (ostre krawędzie) — tak jak na rysunku 2.14. W ten sposób nakażesz Blenderowi rozcięcie siatki tylko wzdłuż tych krawędzi, które oznaczyłeś jako ostre, a nie na podstawie kąta między przyległymi ściankami.

Ostrzeżenie

Nie klikaj przycisku *Apply* (zastosuj) w właściwościach modyfikatora (zob. rysunek 2.14), bo spowodujesz zatwierdzenie wprowadzonych zmian, a to będzie oznaczało, że staną się nieodwracalne. Po prostu zostaw modyfikator takim, jakim jest. Wrócisz do niego później, gdy będziesz eksportował stożek do pliku FBX.



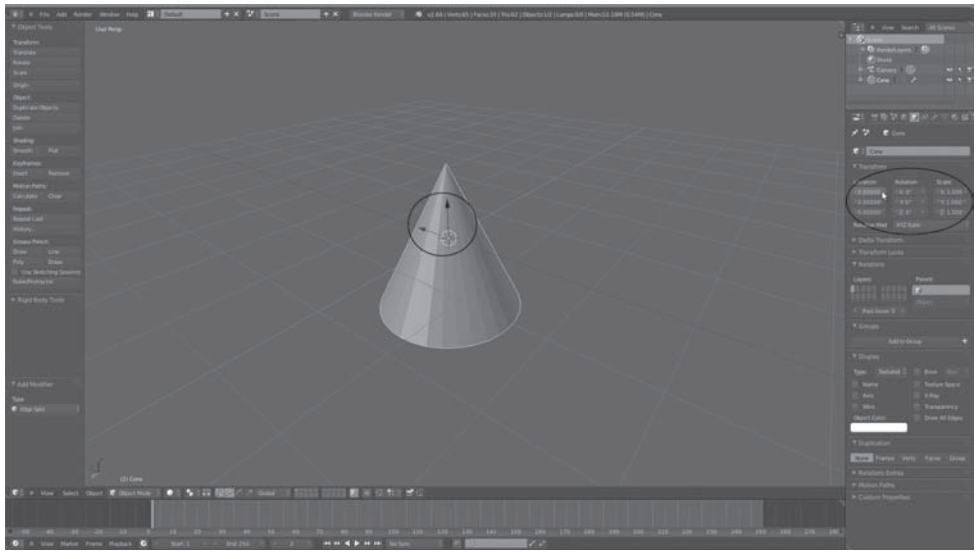
Rysunek 2.14. *Dodaj do stożka modyfikator Edge Split, aby wyostrzyć oznaczone krawędzie*
Źródło: Blender.

Uwaga

Zatwierdzenie (zastosowanie) modyfikatora przypisanego siatce wprowadza zmiany „nieodwracalne", albo inaczej „stałe", w sensie umownym. Należy to rozumieć jako niemożność odłączenia modyfikatora od obiektu i w ten sposób anulowania skutków jego działania. Jedynym sposobem na przywrócenie stanu pierwotnego jest ręczne zmodyfikowanie siatki — przez ponowne ustawienie wierzchołków, krawędzi i ścianek.

Środek obiektu

Zarówno w Unity, jak i w Blenderze każdy obiekt ma określony punkt, zwany w Unity środkiem transformacji (*pivot*), a w Blenderze po prostu środkiem (*origin*). (Ja będę używał nazwy blenderowej). Jest to punkt odniesienia dla geometrii siatki. Jeśli ją przesuniesz do nowego położenia o określonych współrzędnych scenicznych, to właśnie tam znajdzie się jej środek. Jeśli ją będziesz obracał, to właśnie ten punkt będzie środkiem obrotu. Krótko mówiąc: jest to początek lokalnego układu współrzędnych siatki. Gdy ją eksportujesz z Blendera, zapewne chciałbyś, aby położenie tego punktu było takie jak należy. Właśnie teraz pokażę Ci, jak to osiągnąć. Na rysunku 2.15 zaznaczyłem zarówno środek stożka, jak i miejsce na zakładce *Object* (obiekt), gdzie możesz odczytać jego położenie we współrzędnych globalnych. (Gizmo transformacji jest umieszczone dokładnie w środku siatki).



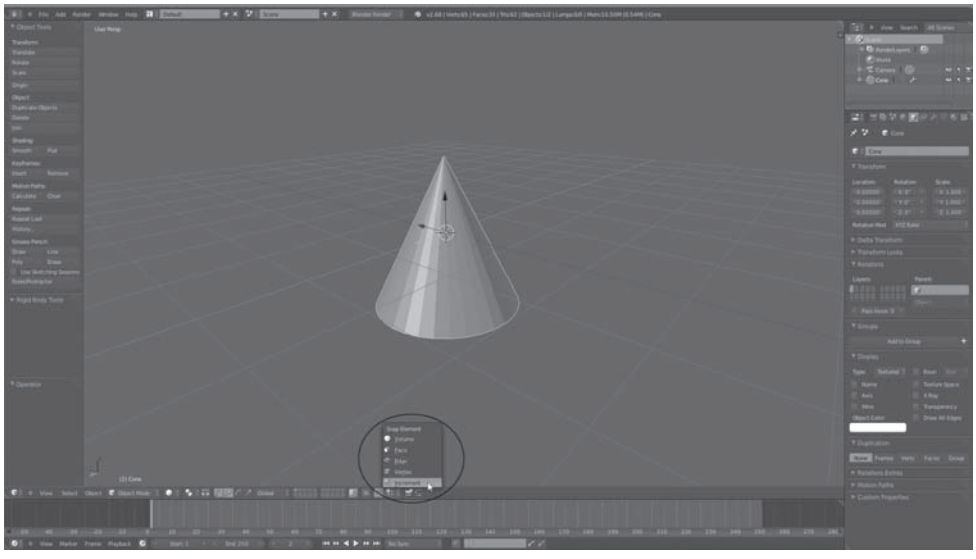
Rysunek 2.15. *Globalne (światowe, sceniczne) współrzędne położenia zaznaczonego obiektu można odczytać na zakładce Object panelu Properties. W oknie widokowym środek obiektu jest wskazywany przez gizmo transformacji*

Źródło: Blender.

Podczas modelowania w Blenderze często środek obiektu ląduje nie tam, gdzie byśmy chcieli. Czasami trzeba go umieścić w ściśle określonym położeniu względem siatki. Na przykład gdy obiektem jest postać człekokształtna, wskazane jest umieszczenie środka jej siatki na dolnej powierzchni stopy, żeby po zaimportowaniu do Unity właśnie tą częścią stykała się z podłożem.

Na rysunku 2.15 środek stożka domyślnie pokrywa się z jego środkiem masy. (Wskazuje na to położenie gizma transformacji). Widać też, że pokrywa się on również z początkiem globalnego układu współrzędnych. Załóżmy, że chcesz go umieścić na podstawie stożka, aby przy współrzędnych globalnych równych (0,0,0) bryła spoczywała na podłożu, a nie pod nim. Żeby to uzyskać, wykonaj następujące czynności:

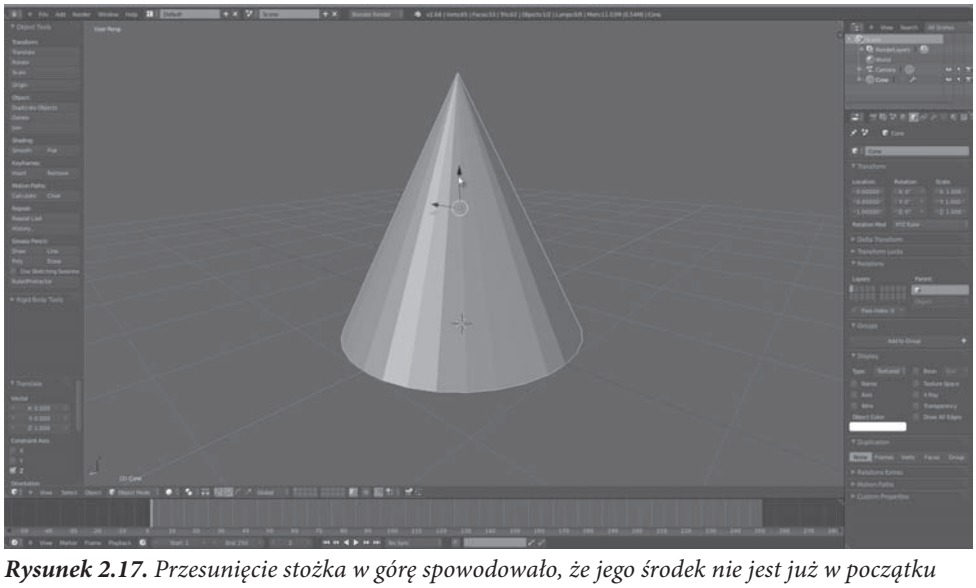
1. W oknie widokowym Blendera zaznacz stożek.
2. Włącz przyciąganie przyrostowe (*Incremental Snapping*) w sposób pokazany na rysunku 2.16. Jeśli teraz będziesz przesuwał stożek, będzie się on przemieszczał skokowo zgodnie z odległościami między liniami blenderowej siatki konstrukcyjnej, a nie płynnie jak przy przesuwaniu swobodnym.
3. Za pomocą gizma transformacji przesuń stożek w górę, aby jego podstawa znalazła się w siatce konstrukcyjnej. Zauważ, że w Blenderze oś skierowana do góry to oś *Z* i w trakcie przesuwania stożka w górę zmienia się wartość tej współrzędnej. Teraz położenie stożka powinno być określone wartościami (0,0,1). Jednak środek siatki przesunął się wraz z nią. Spójrz na rysunek 2.17. Stożek spoczywa na podłożu, tak



Rysunek 2.16. *Włącz przyciąganie przyrostowe, aby móc przesuwać obiekt skokowo zgodnie z siatką konstrukcyjną*

Źródło: Blender.

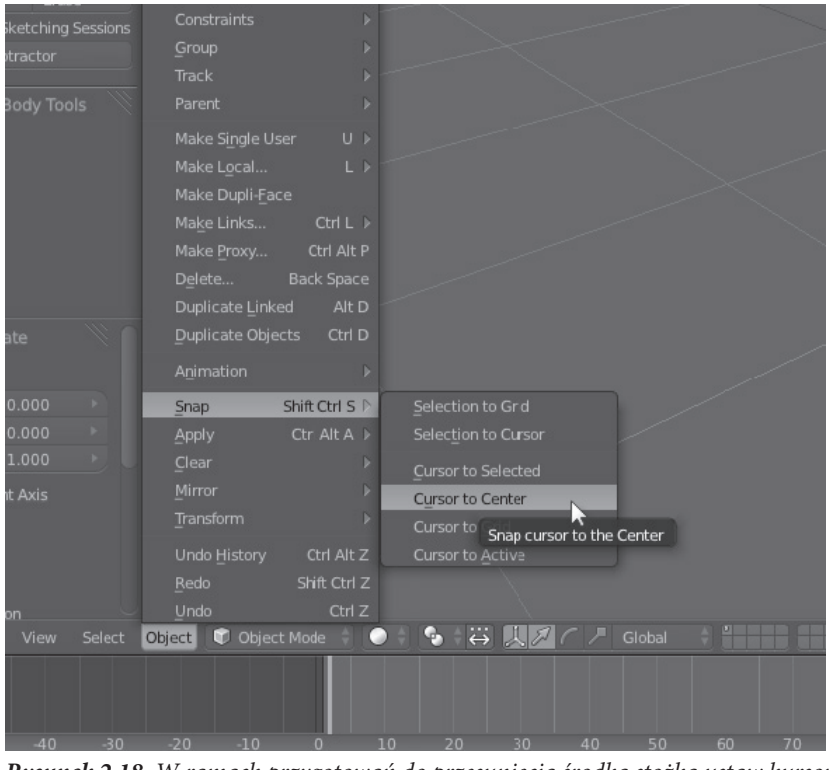
jak chcieliśmy, ale jego środek (wskazywany przez gizmo transformacji) nadal pokrywa się ze środkiem masy, a nie z podłożem. Musimy to tak skorygować, żeby bez przesuwania siatki jej środek znów znalazł się w punkcie o współrzędnych globalnych (0,0,0).



Rysunek 2.17. *Przesunięcie stożka w górę spowodowało, że jego środek nie jest już w początku globalnego układu współrzędnych*

Źródło: Blender.

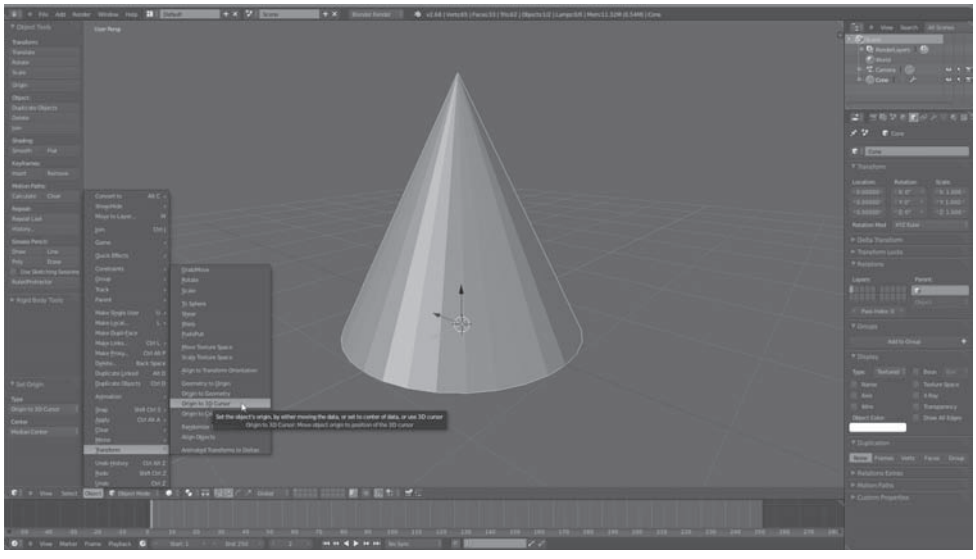
4. Aby z powrotem umieścić środek stożka w początku globalnego układu współrzędnych, użyj blenderowego kursora 3D (widoczny w oknie widokowym jako włosowaty krzyżyk celowniczy). Ustaw ten kursor w początku układu współrzędnych globalnych. W tym celu wybierz polecenie *Object/Snap/Cursor to Center* (obiekt/przyciągnij/kursor do środka) — tak jak na rysunku 2.18.



Rysunek 2.18. *W ramach przygotowań do przesunięcia środka stożka ustaw kursor 3D w początku globalnego układu współrzędnych*

Źródło: Blender.

5. Po ustawieniu kursora 3D w początku globalnego układu współrzędnych przyciągnij do niego środek stożka. W ten sposób początek lokalnego układu współrzędnych siatki pokryje się z początkiem układu globalnego. Aby to zrealizować, zaznacz siatkę stożka (jeśli jeszcze nie jest zaznaczona), a następnie wybierz *Object/Transform/Origin to 3D Cursor* (obiekt/przekształć/środek do kursora 3D). Gizmo transformacji przesunie się do środka globalnego układu współrzędnych, a to będzie oznaczało, że tam też znajduje się środek stożka. Co więcej, współrzędna *Z* stożka będzie teraz miała wartość 0, a nie, jak dotychczas, 1. Rezultat jest pokazany na rysunku 2.19.



Rysunek 2.19. *Umieść środek siatki w środku sceny, a proces importu w Unity będzie bardziej przewidywalny*

Źródło: Blender.

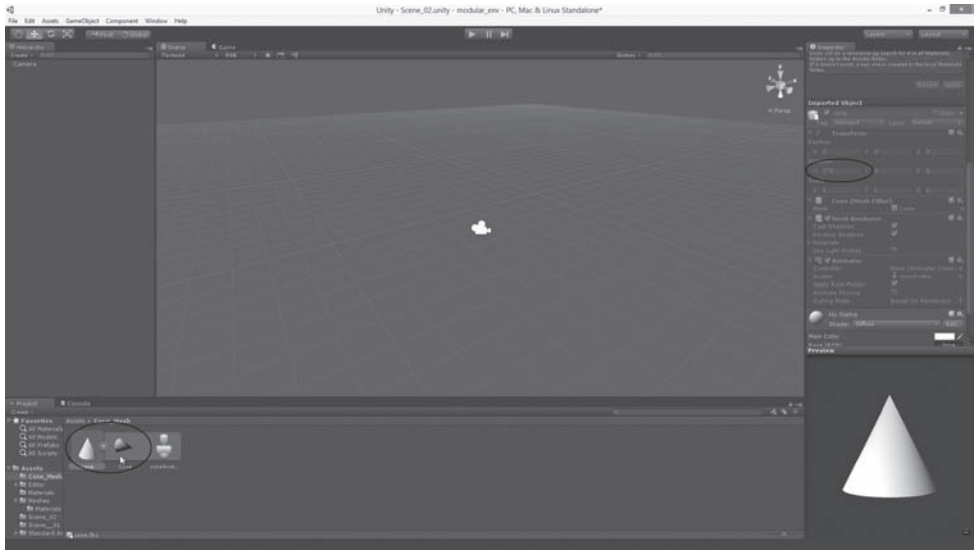
Obracanie obiektów

Niestety, to nie wszystko. Jest jeszcze kilka rzeczy do zrobienia, jeśli chcesz, aby przenoszenie siatki z Blendera do Unity przebiegło bezboleśnie i zakończyło się w sposób przewidywalny.

Aby zilustrować kolejne zadanie, jakie zostało do wykonania, rozważmy następujący scenariusz. Jeśli teraz przeniesiesz stożek z Blendera do Unity — czy to za pomocą pliku *.blend*, czy FBX — jego wygląd w panelu *Project* na pierwszy rzut oka będzie prawidłowy. Lecz jeśli przyjrzysz się uważniej, zauważysz, że coś jest nie tak z orientacją stożka. Wygląda, jakby był obrócony o 90º lub 270º. I rzeczywiście tak jest (rysunek 2.20). Zaznaczenie siatki w panelu *Project* i sprawdzenie jej transformacji w panelu *Inspector* w pełni potwierdza taką diagnozę. Podczas importu w Unity siatka została obrócona o 270º wokół osi X, chociaż w Blenderze jej obroty wokół każdej osi były zerowe (u Ciebie oś obrotu może być inna). Istnieje tutaj wyraźna rozbieżność. Mamy więc do czynienia z niedopasowaniem domyślnych orientacji obiektu w Blenderze i Unity.

Uwaga

Jeśli w tym stanie przeciągniesz siatkę z panelu *Project* i umieścisz ją w scenie Unity, najprawdopodobniej jej orientacja zostanie skorygowana automatycznie, a nawet jeśli tak się nie stanie, możesz to zrobić ręcznie za pomocą stosownego manipulatora. Niektórych takie rozwiązanie zadowala, ale mnie nie. Ja zawsze dążę do tego, by importowane siatki miały ustawienia standardowe — położenie (0,0,0), obrót (0,0,0) i skalę (1,1,1). Dzięki temu unikam kłopotów związanych z niewłaściwymi wartościami początkowymi tych trzech transformacji.



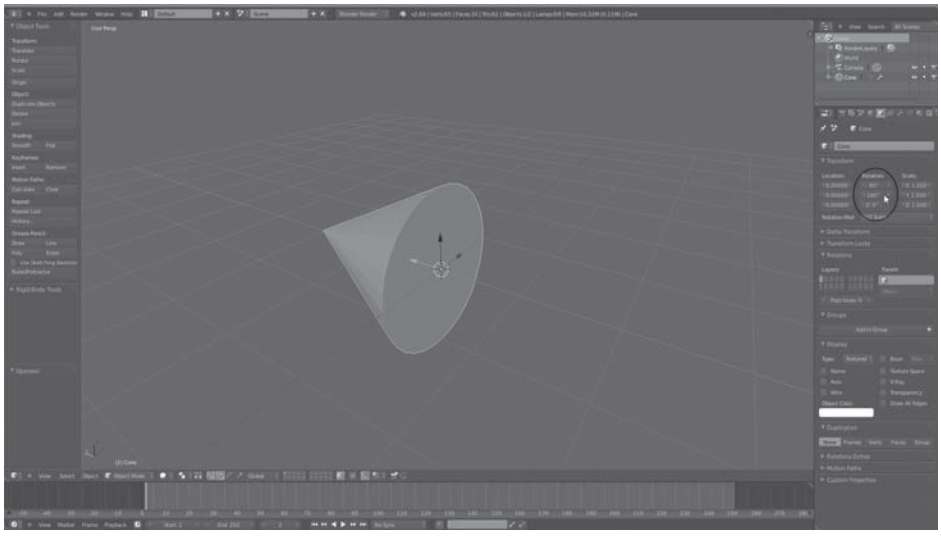
Rysunek 2.20. *Podczas przenoszenia z Blendera do Unity siatka jest obracana wokół jednej z osi o 90º lub 270º*

Źródło: Unity Technologies.

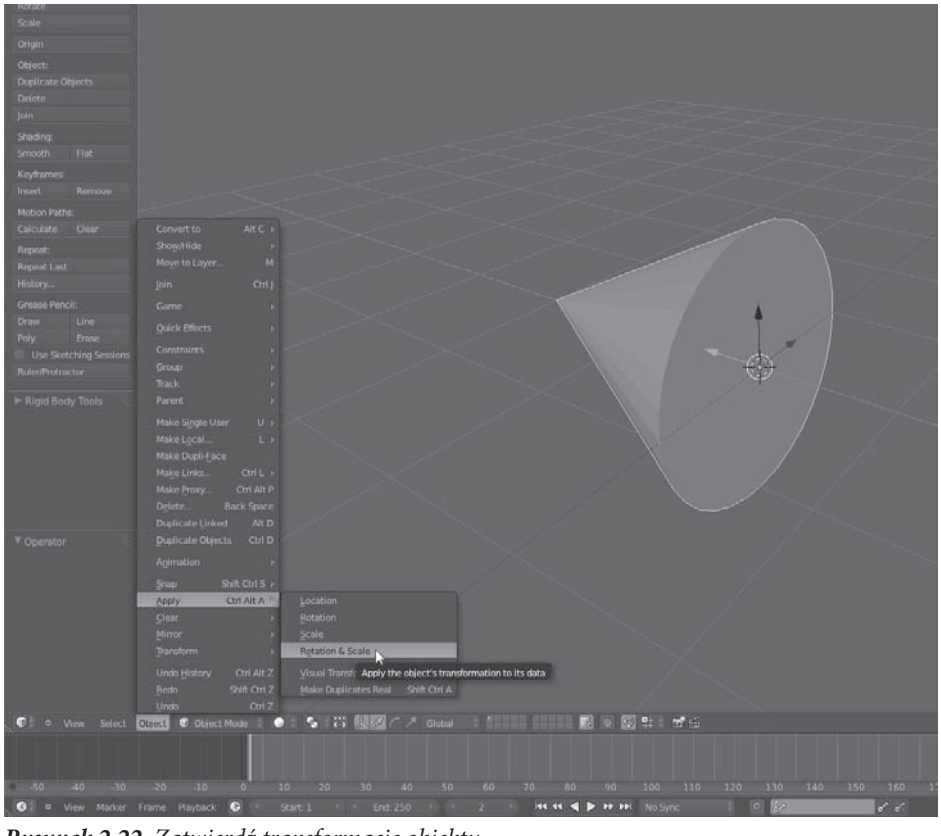
Problem z obrotem wynika z fundamentalnej rozbieżności między układami współrzędnych w Blenderze i w Unity. W Unity osią skierowaną do góry jest *Y*, a w Blenderze taką osią jest *Z*. Dlatego przy przejściu do przestrzeni o innej orientacji obiekt jest obracany o 90º lub 270º. Rozwiązanie polega na wykonaniu obrotu kompensacyjnego w programie macierzystym (Blenderze). Zabieg jest dwuetapowy. Pierwszy krok wykonamy teraz, a drugi dopiero po zapoznaniu się z blenderowym eksporterem FBX.

A zatem, jeśli razem ze mną wykonywałeś poprzednie czynności, usuń siatkę stożka w Unity i wróć do Blendera. W oknie widokowym zaznacz stożek i za pomocą odpowiedniej kontrolki w panelu *Properties* obróć go o -90º wokół osi *X*. Niektóre siatki będziesz musiał także obrócić o 180º wokół osi *Y*, aby po zaimportowaniu do Unity były zwrócone do ekranu przodem, a nie tyłem. Spójrz na rysunek 2.21.

Po wykonaniu kompensacyjnego obrotu musisz to przekształcenie zatwierdzić. Chodzi o takie „wypalenie" lub „zakodowanie" tego przekształcenia, żeby z blenderowego punktu widzenia nadal wszystkie obroty miały wartość zerową. Żeby coś takiego osiągnąć, upewnij się, że siatka jest zaznaczona, po czym wybierz polecenie *Object/Apply/Rotation & Scale* (obiekt/zastosuj/obrót i skalowanie) — rysunek 2.22. I to na razie wszystko! Wrócimy do tej kwestii, gdy będziemy eksportować model do pliku FBX.



Rysunek 2.21. *Przed wyeksportowaniem do formatu FBX nadaj siatce odpowiednią orientację*
Źródło: Blender.

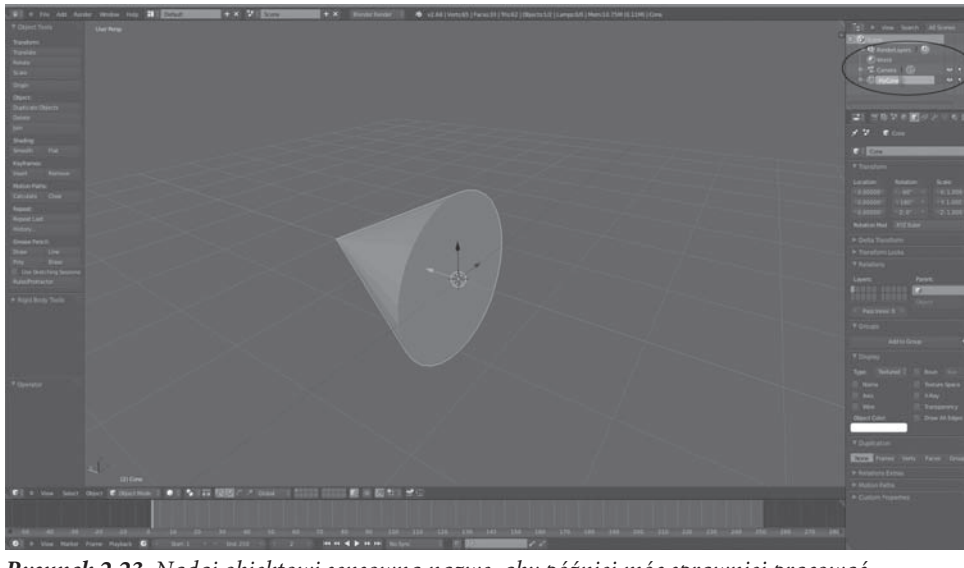


Rysunek 2.22. *Zatwierdź transformację obiektu*
Źródło: Blender.

Nazywanie obiektów

Ten etap procesu eksportowania jest opcjonalny. Przecież możesz z powodzeniem przenieść model do Unity bez nadawania mu specjalnej nazwy, a dla gracza nie będzie to miało żadnego znaczenia. Lecz jeśli gra ma zawierać dużo rozmaitych modeli, to warto poświęcić trochę czasu i ponadawać im jakieś sensowne nazwy, aby później można było łatwiej znaleźć to, czego się szuka. Nie lekceważ tej sprawy, bo bez dobrego systemu nazewnictwa szybko wkrada się bałagan i organizacja pracy zaczyna kuleć. Dlatego nadawaj nazwy modelom już w Blenderze.

Blender automatycznie przypisuje stożkowi nazwę *Cone* (stożek). Jednak na ogół będziesz chciał zmienić taką domyślną nazwę na bardziej unikatową. Spróbujmy zatem nadać stożkowi nazwę `MyCone` (mój stożek). W tym celu zaznacz bryłę w oknie widokowym, dwukrotnie kliknij jej dotychczasową nazwę w panelu *Outliner* (organizator) i zastąp ją nową (rysunek 2.23). I to wszystko! Teraz możemy rozpocząć eksportowanie siatki do pliku FBX.



Rysunek 2.23. *Nadaj obiektowi sensowną nazwę, aby później móc sprawniej pracować*
Źródło: Blender.

Eksportowanie do pliku FBX

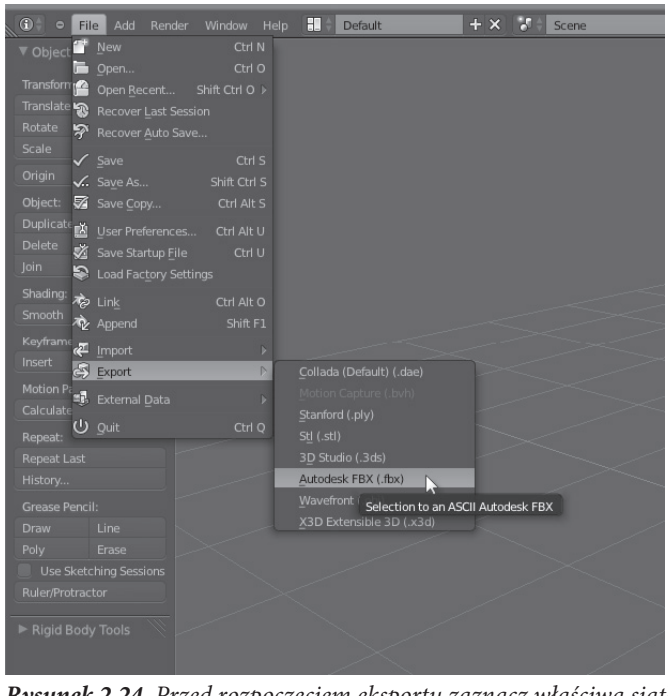
Wszystko, co robiliśmy do tej pory, było przygotowaniem do tego kroku. Teraz użyjemy blenderowego narzędzia eksportującego i za jego pomocą zapiszemy stożek w pliku FBX, aby stamtąd już bez żadnych niespodzianek mógł być zaimportowany w Unity.

Wcześniej uzasadniałem, dlaczego lepiej jest importować siatki w Unity z plików FBX, a nie z plików *.blend*, a teraz, podtrzymując wszystkie tamte stwierdzenia, chcę powiedzieć, że pliki *.blend* również powinny odgrywać ważną rolę w Twojej pracy. Nigdy nie

zapominaj o zapisywaniu swoich scen właśnie w tych plikach, bo tylko dzięki temu będziesz miał do nich pełny dostęp i w razie potrzeby będziesz mógł wprowadzać niezbędne zmiany. Jeśli zdecydujesz się na modyfikowanie siatki, nie wczytuj jej do Blendera z pliku FBX, lecz załaduj ją z pliku *.blend*. Plik FBX traktuj wyłącznie jako medium pośredniczące w przenoszeniu modeli z Blendera do Unity. Dlatego zanim wyeksportujesz siatkę do pliku FBX, zapisz ją w pliku *.blend*, wydając Blenderowi standardowe polecenie *File/Save* (plik/zapisz).

Aby zaznaczoną siatkę wyeksportować do pliku FBX, wykonaj następujące czynności:

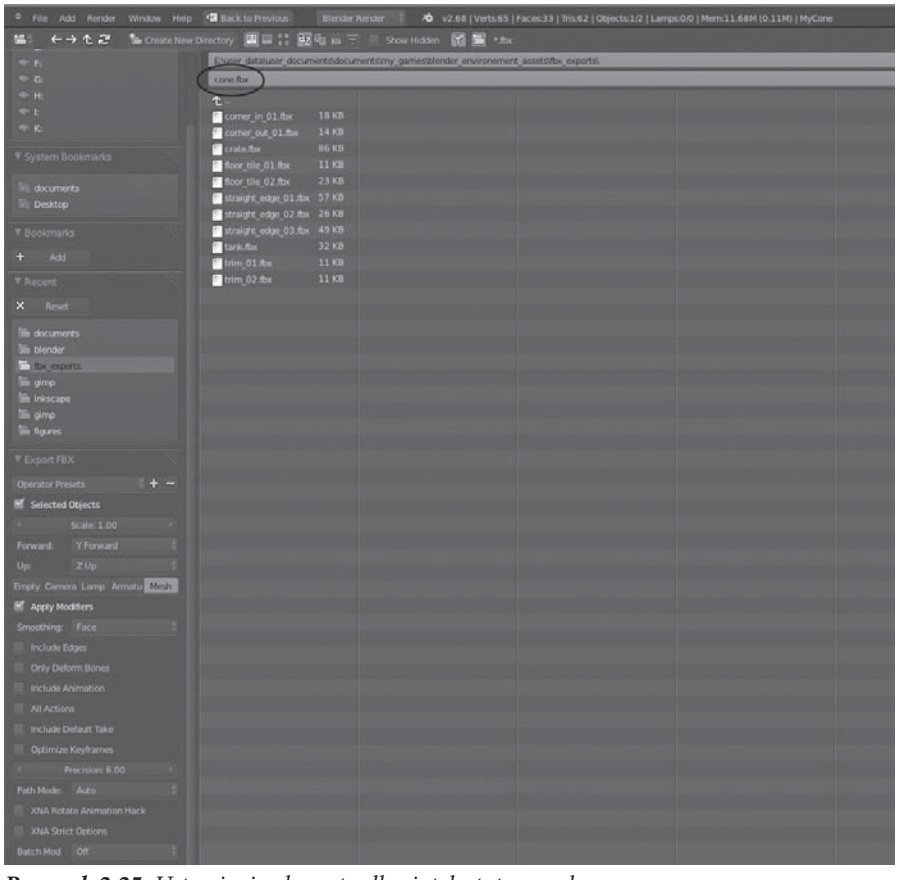
1. Zaznacz przeznaczoną do eksportu siatkę i wybierz *File/Export/Autodesk FBX (.fbx)* (plik/eksportuj/Autodesk FBX) — rysunek 2.24. Otworzy się okno narzędzia eksportującego z wieloma opcjami i parametrami eksportu. Dla siatek nieanimowanych, czyli takich jak stożek, najlepsze będą ustawienia pokazane na rysunku 2.25.



Rysunek 2.24. *Przed rozpoczęciem eksportu zaznacz właściwą siatkę*
Źródło: Blender.

2. Nadaj plikowi sensowną nazwę. Ja wpisałem *Cone.fbx*, co widać na rysunku 2.25. Każda eksportowana siatka powinna mieć własną, unikatową i opisową nazwę.

3. Na rolecie *Export FBX* (eksport FBX) zaznacz opcję *Selected Objects* (zaznaczone obiekty). Spowoduje to, że wyeksportowane zostaną tylko obiekty zaznaczone, a nie cała scena. Jeśli nie włączysz tej opcji, wszystkie siatki zostaną skomasowane w jedną grupę i wyeksportowane.



Rysunek 2.25. *Ustawienia eksportu dla siatek statycznych*
Źródło: Blender.

Uwaga

Parametr *Scale* (skala) ma domyślną wartość `1.00` i w większości przypadków byłaby ona odpowiednia. Jednakże, jak się wkrótce przekonasz, Unity zupełnie pomija ten parametr bez względu na jego wartość. W zamian stosuje własną skalę o wartości `0.01`, czy tego chcesz, czy nie. Później pokażę, jak sobie z tym problemem radzić.

4. Określ kierunki *Forward* (w przód) i *Up* (w górę). Jest to drugi etap procesu rozpoczętego w punkcie „Obracanie obiektów". Ustawienia te w połączeniu z odpowiednią orientacją obiektu w scenie gwarantują mu właściwą orientację również po zaimportowaniu w Unity. Zazwyczaj kierunki te należy ustawiać zgodnie z kierunkami osi w Blenderze, czyli w polu *Forward* należy ustawić *Y Forward*, a w polu *Up* — *Z Up*. Czasami będziesz musiał trochę pokombinować z tymi ustawieniami, zanim siatka zostanie zaimportowana ze standardowymi wartościami transformacji, czyli z położeniem (0,0,0), obrotem (0,0,0) i skalą (1,1,1), a mimo to będzie miała właściwą orientację.

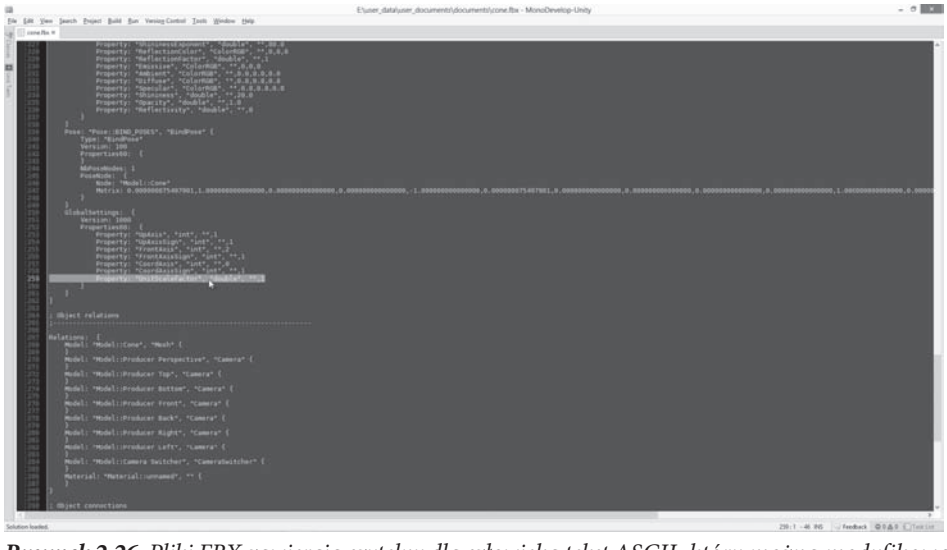
5. Upewnij się, że zaznaczona jest opcja *Apply Modifiers* (zastosuj modyfikatory). Jest to konieczne, aby wszystkie modyfikatory przypisane siatce, włącznie z *Edge Split* (opisanym w punkcie „Ostrość krawędzi i modyfikator Edge Split"), zostały zakodowane w eksportowanej siatce. (Opcja ta nie spowoduje zatwierdzenia żadnego modyfikatora w pierwotnej blenderowej scenie; ma wpływ jedynie na siatkę, która jest generowana podczas eksportu i zapisywana w pliku FBX).
6. Dla siatek nieanimowanych (statycznych) wyłącz opcje *Include Animation* (dołącz animację) i *Optimize Keyframes* (optymalizuj klatki kluczowe). O eksportowaniu siatek animowanych będzie mowa w dalszej części książki.

Uwaga

Zamiast FBX możesz użyć formatu Collada (*.dae*). Użyj wtedy polecenia *File/Export/Collada* (plik/eksportuj/Collada). Format ten obsługuje siatki zarówno statyczne, jak i animowane. Więcej informacji na jego temat znajdziesz pod adresem: *https://collada.org/*.

Zawartość pliku FBX

Podczas eksportowania Blender tworzy plik FBX możliwy do odczytania w Unity. Przyjrzyjmy mu się nieco dokładniej. Warto to zrobić, bo plik zawiera czytelne dla człowieka (tekst ASCII) i edytowalne zapisy właściwości eksportowanej siatki. W razie potrzeby można tu jeszcze coś poprawić. Do przeglądania i edycji zawartości takiego pliku można użyć dowolnego edytora tekstowego, takiego jak Notepad, Notepad++ czy nawet MonoDevelop (rysunek 2.26).



Rysunek 2.26. *Pliki FBX zawierają czytelny dla człowieka tekst ASCII, który można modyfikować w dowolnym edytorze tekstowym lub zintegrowanym środowisku programistycznym (IDE)*

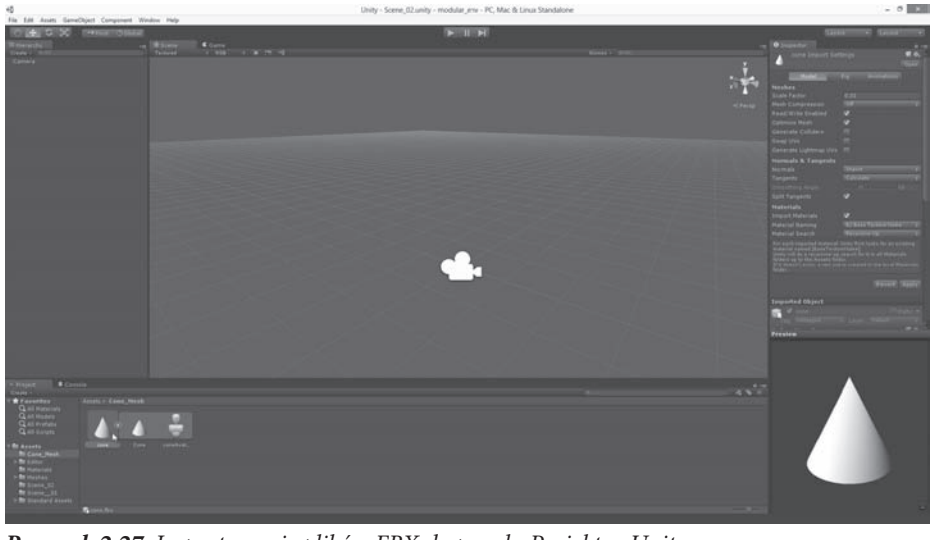
Źródło: MonoDevelop.

Uwaga

Teoretycznie ręczne manipulowanie w prawidłowo wyeksportowanych plikach FBX nie powinno być potrzebne. Ale mimo wszystko warto wiedzieć, że taka możliwość istnieje. Warto też wiedzieć, że pliki te można analizować i przetwarzać również w trybie wsadowym.

Ćwiczenie: importowanie plików FBX w Unity

Po wyeksportowaniu siatki FBX z Blendera zapewne będziesz chciał ją zaimportować w Unity, aby tam dołączyć ją do gry. W tym celu po prostu przeciągnij odpowiedni plik z Eksploratora (Windows) lub Findera (Mac OS) na panel *Project* w Unity (rysunek 2.27). Ale to jeszcze nie wszystko. Najprawdopodobniej pewne ustawienia będą wymagały dostrojenia. W tym ćwiczeniu zajmiemy się również takimi sprawami.



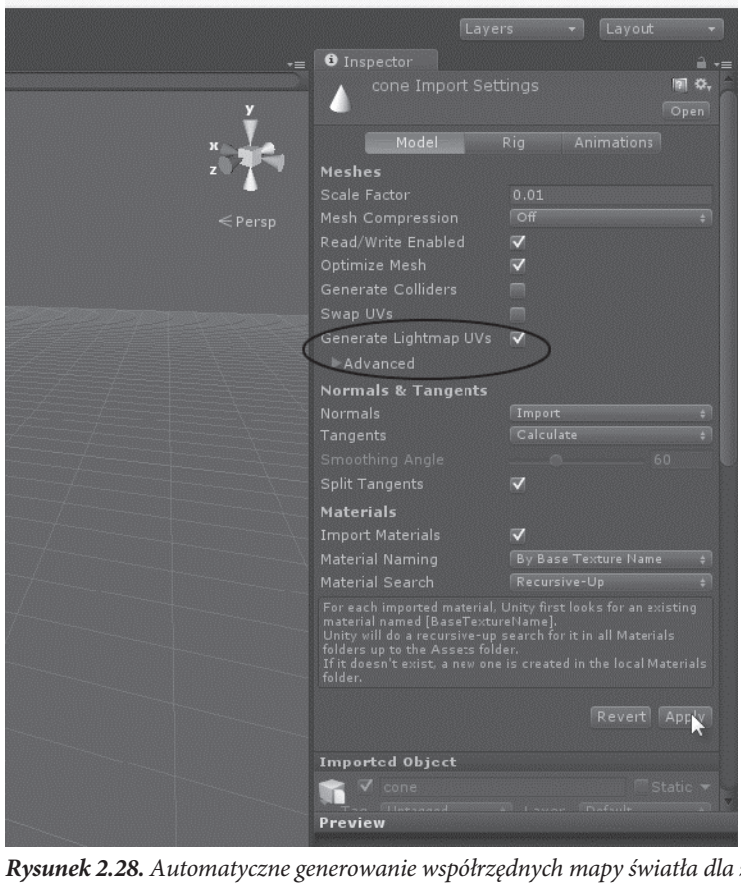
Rysunek 2.27. *Importowanie plików FBX do panelu Projekt w Unity*
Źródło: Unity Technologies.

Współrzędne UV mapy światła

Gdy będziesz importował statyczną siatkę terenu, budowli lub innego elementu środowiska gry, zapewne będziesz chciał ją oświetlić metodą mapowania światła. Do realizacji tego rodzaju oświetlenia możesz użyć narzędzia Beast Lightmapper dostępnego w Unity po wybraniu polecenia *Window/Lightmapping* (okno/mapowanie światła). **Mapowanie światła** polega na określaniu, jak dana siatka reaguje na panujące w jej otoczeniu warunki oświetleniowe — jakie rzuca cienie, jakie tworzy odblaski — i zapisywaniu takich informacji w formie specjalnej tekstury. Tekstura taka, zwana **mapą światła**, jest potem nakładana na powierzchnię siatki, co sprawia, że ta ostatnia wygląda, jakby rzeczywiście była oświetlona.

Dzięki takim zabiegom ani procesor graficzny, ani główny nie muszą na bieżąco obliczać efektów oświetleniowych. Jednak żeby coś takiego się udało, zaimportowane siatki muszą mieć przypisane współrzędne mapy światła. Współrzędne te to specjalne dane matematyczne określające, w jaki sposób tekstura z mapą światła ma być odwzorowana na powierzchni siatki. Bez nich mapowanie światła może dać wynik daleki od oczekiwanego. Wartości tych współrzędnych możesz wyznaczyć ręcznie w Blenderze, ale możesz też zlecić to zadanie odpowiednim funkcjom Unity.

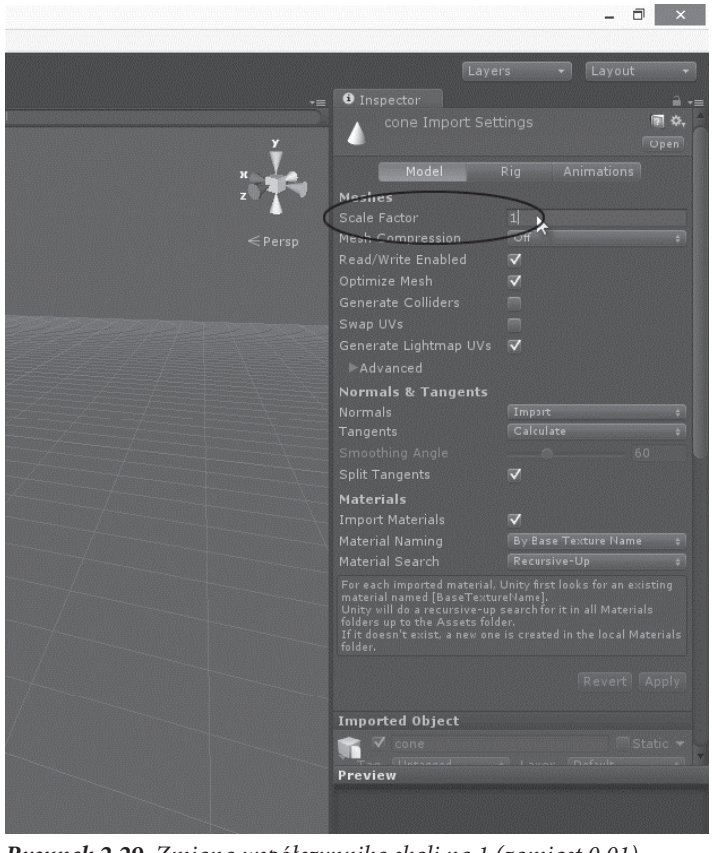
Jeśli zdecydujesz się na ten drugi sposób, zacznij od zaznaczenia siatki w panelu *Project*, aby wyświetlić jej właściwości w panelu *Inspector*. Potem po prostu włącz opcję *Generate Lightmap UVs* (generuj współrzędne mapy światła) i kliknij przycisk *Apply* (zastosuj) — tak jak na rysunku 2.28. W większości przypadków otrzymasz rezultat prawidłowy, kłopot mogą sprawić jedynie siatki o kształtach okrągłych lub organicznych i wtedy będziesz musiał sam wyznaczyć te współrzędne za pomocą odpowiednich narzędzi Blendera. Więcej informacji na ten temat znajdziesz w następnym rozdziale.



Rysunek 2.28. *Automatyczne generowanie współrzędnych mapy światła dla zaimportowanej siatki*
Źródło: Unity Technologies.

Współczynnik skali

Dla każdej importowanej siatki FBX Unity automatycznie ustala współczynnik skali o wartości 0,01 i nie ma tu znaczenia, jaka wartość jest zapisana w importowanym pliku. W rezultacie siatka przyjmuje rozmiary dużo mniejsze niż te, które powinna mieć zgodnie z projektem. Czasami będziesz musiał wykonać mocne zbliżenie, żeby ją w ogóle zobaczyć. Problem można rozwiązać przez zwiększenie wartości parametru *Scale Factor* (współczynnik skali) do poziomu 1 (rysunek 2.29). Jednakże przy większej liczbie importowanych siatek lub wielokrotnym importowaniu tej samej siatki (ze względu na częste modyfikacje) ciągłe powtarzanie tej samej czynności szybko staje się męczące.

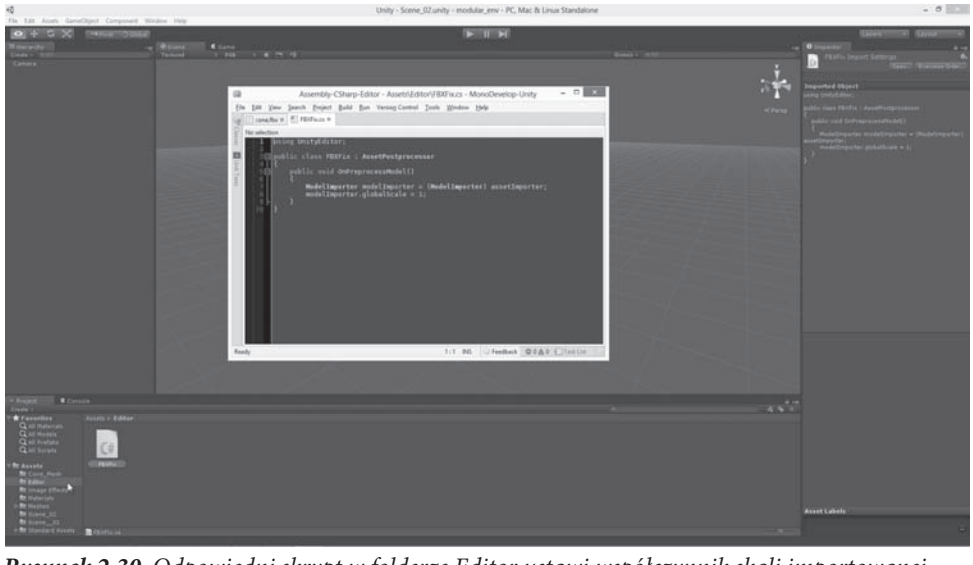


Rysunek 2.29. *Zmiana współczynnika skali na 1 (zamiast 0,01)*
Źródło: Unity Technologies.

Metoda alternatywna polega na napisaniu skryptu, który zmusi Unity do określonego zachowania podczas importowania siatek — w tym przypadku do ustawiania za każdym razem współczynnika skali o wartości 1. Skrypt będzie działał automatycznie i już nigdy nie będziesz musiał ręcznie poprawiać tego parametru. Aby to uzyskać, utwórz w panelu *Project* nowy skrypt w języku C# i nazwij go `FBXFix.cs` (poprawianie FBX). Następnie umieść w tym skrypcie następujący kod:

```
using UnityEditor;
public class FBXFix : AssetPostprocessor
{
   public void OnPreprocessModel()
   {
      ModelImporter modelImporter = (ModelImporter) assetImporter;
   modelImporter.globalScale = 1;
   }
}
```

Gotowy skrypt umieść w folderze *Editor* (edytor). (Jeśli taki folder nie istnieje w bieżącym projekcie, utwórz go). Żeby skrypt działał, musi się znajdować właśnie w takim folderze (rysunek 2.30). Aby go przetestować, zaimportuj nowy plik FBX. Współczynnik skali powinien mieć teraz wartość 1!



***Rysunek 2.30**. Odpowiedni skrypt w folderze Editor ustawi współczynnik skali importowanej siatki na 1*

Źródło: Unity Technologies.

A zatem umiesz już eksportować modele z Blendera i importować je do Unity z zachowaniem niemal pełnej wierności.

Podsumowanie

Rozdział ten poświęciłem opisowi czynności związanych z przenoszeniem modeli między Blenderem a Unity. Mówiąc krótko: gdy dwa różne programy wymieniają między sobą dane, bardzo często część tych danych jest tracona lub zmieniana. W przypadku Blendera i Unity do przenoszenia danych można użyć plików *.blend* lub FBX. W tym rozdziale starałem się wykazać, że pliki FBX nadają się do tego celu znacznie lepiej. Ale jak miałeś okazję się przekonać, temat jest znacznie szerszy niż tylko wybór

formatu plików. Żeby siatka przeniesiona z Blendera do Unity wyglądała jak najlepiej, trzeba wykonać jeszcze kilka dodatkowych czynności. Niezbędne jest oznakowanie ostrych krawędzi siatki, nadanie jej odpowiedniej orientacji, właściwe umiejscowienie środka transformacji itp. Wszystko, co tutaj napisałem, odnosi się do każdej siatki, niezależnie od jej rodzaju. Siatki animowane mają dodatkowe wymagania, ale tym zagadnieniem zajmiemy się dokładniej w jednym z dalszych rozdziałów. Natomiast tematem następnego rozdziału będzie specyficzny rodzaj siatek statycznych, mianowicie modułowe siatki środowiskowe.

Skorowidz

E

F

G

H

I

J

K

L

Ł

M

N

O

P

R

S

Z

Zaprezentuj światu swoją wizję elektronicznej rozrywki!

Unity to darmowy silnik do tworzenia zaawansowanych gier komputerowych. Blender to darmowe środowisko do tworzenia grafiki 3D. Co może powstać z połączenia tych dwóch narzędzi? Genialne środowisko, dzięki któremu przygotujesz atrakcyjną wizualnie grę o niesamowitej grywalności! Z tego duetu gromadnie korzystają małe studia opracowujące gry — studia z małym budżetem, ale ze świetnymi pomysłami!

Jeżeli Twoim marzeniem jest wejście na rynek gier komputerowych, lecz nie dysponujesz budżetem, który pozwoliłby Ci zagrozić gigantom elektronicznej rozrywki, to Unity w połą-czeniu z Blenderem jest Twoją szansą na sukces. Sięgnij po tę książkę i przekonaj się, jak zorganizować proces tworzenia gry w 10 etapach. Z kolejnych rozdziałów nauczysz się, jak przenosić modele z Blendera do Unity, tworzyć tekstury i teren oraz animować obiekty. Na sam koniec dowiesz się, w jaki sposób oświetlać przedmioty i przechowywać stan gry. Znajdziesz tu też opisy różnych narzędzi, które mogą Ci pomóc w codziennej pracy. Ta książka to obowiązkowa lektura dla wszystkich marzycieli, którzy chcą tworzyć gry komputerowe. Dzięki niej marzenia staną się rzeczywistością!

Sięgnij po tę książkę i:

- zintegruj Unity z Blenderem
- stwórz zaawansowany teren
- wykonaj atrakcyjne modele
- stwórz animację
- zbuduj grę komputerową w 10 sprawdzonych etapach

Alan Thorn
— matematyk, twórca gier komputerowych. Założyciel niezależnego studia gier, zdobywca wielu nagród branżowych, autor licznych książek poświęconych tej tematyce. Prowadzi wykłady związane z tworzeniem gier.

Helion

31292 numer katalogowy
księgarnia internetowa
http://helion.pl
zamówienia telefoniczne
0 801 339900
0 601 339900

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

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

sięgnij po WIĘCEJ
KOD KORZYŚCI

ISBN 978-83-283-0269-3
9 788328 302693

Informatyka w najlepszym wydaniu
cena: 54,90 zł