Tworzenie świata AR

Jak dotąd nasza gra skupiała się na interakcji gracza z eksperymentalnymi potworami do gotowania. Chociaż mogą śledzić potwory na mapie, nic wokół nich nie jest częścią tej alternatywnej rzeczywistości. Cóż, oczywiście chcemy naprawić tę wadę. Chcemy stworzyć bogaty świat rzeczywistości rozszerzonej wokół gracza, z którym mogą również wchodzić w interakcje, znajdować/uzupełniać przedmioty, szkolić potwory i wysyłać je na misje kulinarne. W tym celu musimy zaludnić świat wokół gracza alternatywnymi lokalizacjami rzeczywistości gry. W tej części wrócimy do mapy i zaczniemy zaludniać świat wokół gracza nowymi lokalizacjami w alternatywnej rzeczywistości. Te lokacje nie będą w całości oparte na naszej rzeczywistości w grze. W rzeczywistości użyjemy lokalizacji w świecie rzeczywistym jako podstawy naszego alternatywnego świata gry. Lokalizacje, które zapełniamy w świecie AR, będą pobierane z usługi internetowej opartej na lokalizacji. Poniżej znajduje się podsumowanie tego, co omówimy tu:

* Powrót do mapy

* Singleton

* Przedstawiamy interfejs API Miejsc Google

* Korzystanie z JSON

* Konfigurowanie usługi API Miejsc Google

* Tworzenie znaczników

* Optymalizacja wyszukiwania

To będzie krótka, ale intensywna część i szybko omówimy kilka rzeczy. Jeśli skoczyłeś do przodu z poprzedniego rozdziału w książce, upewnij się, że jesteś doświadczonym programistą Unity lub po prostu przeglądasz zawartość książki.

Powrót do mapy

Oczywiście w pewnym momencie trzeba było zdać sobie sprawę, że wrócimy do mapy. Mapa jest podstawowym i fundamentalnym elementem naszej gry opartej na lokalizacji. Zapewnia okno do alternatywnej rzeczywistości gry, a jednocześnie zapewnia graczowi odniesienie do świata rzeczywistego. Do tej pory jedyną wskazówką, jaką dawaliśmy temu alternatywnemu światu, było śledzenie i wizualizacja potworów. Jednak nasze potwory są całkowicie losowe i nie przylegają do żadnego z otaczającego ich świata rzeczywistego. W pewien sposób niszczy to doświadczenie, ale naprawienie zachowania potwora samo w sobie może być inną książką. Aby przywrócić nasze wirtualne obiekty/miejsca z powrotem do rzeczywistości, użyjemy interfejsu API Map Google do wypełnienia mapy wokół odtwarzacza. Zanim przejdziemy do dodawania nowych funkcji do gry, poświęćmy trochę czasu na naprawienie kilku problemów, które zignorowaliśmy pod koniec ostatniej części. Są szanse, że zauważyłeś problem, jeśli grałeś w grę przez jakiś czas. Jeśli tego nie zrobiłeś, problem polegał na tym, że usługa GPS przestawała działać, a mapa przestawała się aktualizować po powrocie użytkownika ze scen Catch lub Inventory. W poprzedniej części zabrakło nam czasu na naprawienie tego problemu, a poza tym wiedzieliśmy, że w kolejnym rozdziale będziemy mieli okazję naprawić problem. Jeśli jesteś profesjonalnym programistą, doskonale wiesz, że łamanie, refaktoryzacja i naprawianie kodu polega na ewolucji oprogramowania i gier. Wielu nowych programistów spędza zbyt dużo czasu na pisaniu idealnego fragmentu kodu, a następnie robieniu wszystkiego, aby go zachować. Kod ma zostać zmieniony, przepisany iz pewnością usunięty. Im szybciej to zrozumiesz, tym lepsze nastawienie będziesz miał jako programista. Oczywiście jest czas i miejsce na refaktoryzację, a na pewno nie jest to

dzień przed wysłaniem gry lub produktu. W celu naprawienia problemu z GPS, klasa GPSLocationService została przepisana jako Singleton. Ponadto wszystkie klasy zależne korzystające z tej usługi również musiały zostać zaktualizowane. Najpierw zaimportujemy zaktualizowane skrypty, a następnie przeniesiemy niektóre usługi w następujący sposób:

1. Otwórz Unity na jeden z projektów, tak jak go zostawiliśmy na końcu ostatniej części.

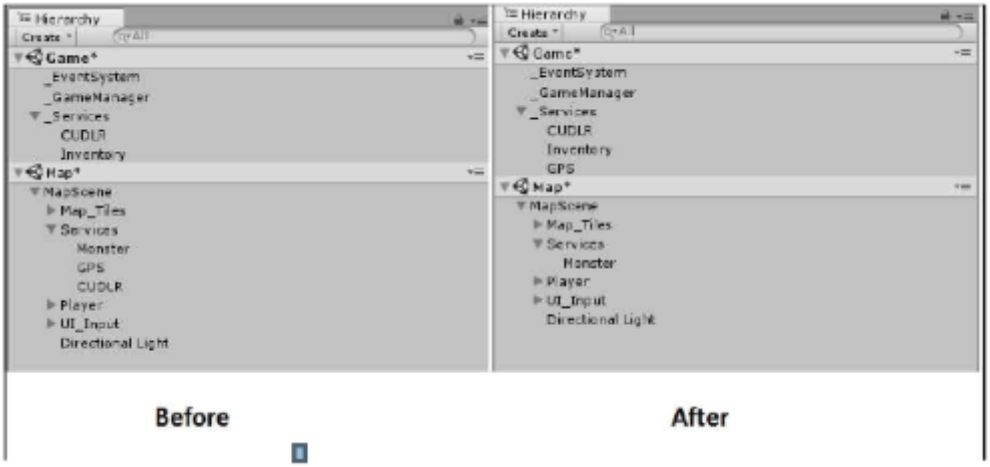
2. Z menu wybierz Zasoby | Importuj pakiet | Pakiet niestandardowy..., a po otwarciu okna dialogowego Importuj pakiet... przejdź do folderu pobranego kodu źródłowego.

3. Zaimportuj pakiet tak jak wcześniej.

4. Otwórz scenę Gra z folderu Zasoby w oknie Projekt, klikając ją dwukrotnie.

5. Przeciągnij scenę Mapa do okna Hierarchia z folderu Zasoby w oknie Projekt.

6. Rozwiń mapę | Obiekt MapScene w oknie Hierarchia. Następnie rozwiń również obiekt Usługi. Rozwiń także obiekt _Services w scenie Game. Okno Hierarchii powinno teraz przypominać poniższy zrzut ekranu:



7. Wybierz mapę | MapScene | Usługi | obiekt CUDLR i naciśnij klawisz Delete, aby go usunąć. Nasza główna scena Game ma CUDLR i nie potrzebujemy dwóch instancji.

8. Przeciągnij mapę | MapScene | Usługi | obiekt GPS i upuść go w grze | Usługi obiektu czynią z niego dziecko. Dzięki temu nasza usługa GPS stanie się teraz usługą główną. Usługa Monster będzie jedyną usługą pozostałą na naszej scenie mapy, jak pokazano na poprzednim zrzucie ekranu.

9. Kliknij prawym przyciskiem myszy (naciśnij klawisz Ctrl i kliknij komputer Mac) na scenie mapy w oknie hierarchii, aby otworzyć menu kontekstowe. Wybierz Usuń scenę z menu i po wyświetleniu monitu upewnij się, że ją zapisałeś.

10. Zapisz scenę gry, wpisując Ctrl + S (polecenie + S na Macu).

11. Naciśnij Play i ponownie uruchom grę. Upewnij się, że tryb symulacji GPS jest uruchomiony (patrz Rozdział 2, Mapowanie lokalizacji odtwarzacza, jeśli nie masz pewności, jak to ustawić). Zmieniaj się kilka razy z mapy na scenę ekwipunku iz powrotem. Zauważ teraz, jak działa GPS, tak jak powinien.

Może się więc wydawać, że wszystko, co musieliśmy zrobić, to przenieść kilka usług, ale to było dalekie od zakresu wymaganych zmian. Jak wspomniano, usługa GPSLocationService została przekonwertowana na Singleton. Wzorzec Singleton stosowaliśmy wcześniej również dla klas GameManager i InventoryService. Jednak nigdy nie doszliśmy do szczegółów, jak działa Singleton. W następnej sekcji skupimy się na formacji Singleton.

Singleton

Kiedy zaczynaliśmy tworzenie naszej gry, zarządzaliśmy wszystkimi naszymi obiektami lokalnie na scenie. Nie chcieliśmy ani nie musieliśmy się martwić o żywotność naszych usług/menedżerów. Jednak, jak prawie zawsze, nasza gra dojrzała i zaczęliśmy używać wielu scen. Teraz potrzebowaliśmy, aby nasze usługi lub klasy menedżerskie były łatwo dostępne dla scen potomnych i prawie w każdym miejscu naszego kodu. W starych grach stworzylibyśmy po prostu zmienną globalną lub statyczną do śledzenia stanu gry w scenach lub skryptach. Globalna klasa statyczna może działać, ale ma szereg ograniczeń, takich jak:

* Klasy statyczne są leniwie ładowane i mogą być szczególnie kruche w Unity.

* Klasy statyczne nie mogą implementować interfejsu.

* Klasy statyczne można wyprowadzić tylko z obiektu. Nie mogą dziedziczyć po MonoBehaviour i dlatego mogą być używane jako komponenty w Unity, co oznacza, że nie mogą również używać Unity Coroutines ani innych metod podstawowych, takich jak Start, Update i tak dalej.

Przyjrzyjmy się różnicy w deklarowaniu standardowego obiektu gry MonoBehaviour i takiego, który używa Singletona ze zaktualizowanym skryptem GPSLocationService.

Wskazówka : możesz swobodnie otwierać zaktualizowane skrypty w wybranym edytorze podczas śledzenia.

Poprzednia implementacja usługi GPSLocationService została zadeklarowana w następujący sposób:

public class GPSLocationService : MonoBehaviour

Jak widzieliśmy już wiele razy, jest to standardowy sposób deklarowania komponentu Unity. Porównaj to z nową deklaracją klasy GPSLocationService:

public class GPSLocationService :

Singleton<GPSLocationService>

Uwaga : Przejrzyj i zrozum, jak działa definicja klasy Singleton. Skrypt znajduje się w folderze Assets/FoodyGo/Scripts/Managers.

Może to wyglądać dziwnie: deklarowanie obiektu do dziedziczenia z typu ogólnego o nazwie Singleton z samym sobą jako typem. Pomyśl o klasie Singleton jako o opakowaniu, które konwertuje instancję na globalną zmienną statyczną, która jest dostępna w dowolnym miejscu w kodzie. Oto przykład tego, w jaki sposób usługa GPS była dostępna wcześniej i jest teraz:

//before, GPS service object was set as a field of

the class which had to be set in the editor

public GPSLocationService gpsLocationService;

gpsLocationService.OnMapRedraw +=

gpsLocationService_OnMapRedraw;

//after, the GPS service is now accessible

anywhere as a Singleton

GPSLocationService.Instance.OnMapRedraw +=

GpsLocationService_OnMapRedraw;

Jest jeden krytyczny element, o którym powinieneś wiedzieć podczas pracy z naszą implementacją Singletona. Zawsze upewnimy się, że tworzymy instancję menedżera lub usługi Singletona jako obiektu gry w scenie. Odbywa się to, abyśmy mogli skorzystać z metod Start, Awake, Update i innych. Jeśli jednak nie dodasz tych obiektów do sceny, a następnie spróbujesz uzyskać do nich bezpośredni dostęp w kodzie, będą one dostępne, ale prawdopodobnie będie brakować ważnych inicjalizacji ustawionych w metodach Awake lub Start. Spójrz na klasy, które wykorzystują GPSLocationService: MonsterService, CharacterGPSCompassController i GoogleMapTile. Jak docenisz, różnice są subtelne, ale znaczące. W tym rozdziale dodamy kolejnego konsumenta usługi GPSLocationService, a także zmienimy nową usługę jako Singleton. Ta nowa usługa będzie usługą GooglePlacesAPIService i omówimy ją w następnej sekcji.

Przedstawiamy interfejs API Miejsc Google

Wykorzystamy interfejs API Miejsc Google, aby wypełnić wirtualny świat wokół gracza odniesieniami do rzeczywistych lokalizacji lub miejsc. Ponieważ korzystaliśmy już z Google Static Maps API, dodanie kolejnej usługi powinno być proste. Jednak w przeciwieństwie do API map, API miejsc jest znacznie bardziej restrykcyjne w użyciu. Oznacza to, że będziemy musieli podjąć dodatkowe kroki konfiguracyjne i zmodyfikować sposób, w jaki uzyskujemy dostęp do usługi. Nie wspominając o tym, że wysyłka będzie miała bezpośredni wpływ na nasz model biznesowy.

Uwaga : kolejnym bezpośrednim konkurentem interfejsu API Miejsc Google jest Foursquare. Foursquare ma znacznie mniej restrykcyjnych limitów użytkowania, ale wymaga dodatkowych mechanizmów uwierzytelniania.

Aby zacząć korzystać z Google Places API, będziemy musieli się zarejestrować i utworzyć nowy klucz API. Ten klucz pozwoli Twojej aplikacji/grze na wykonanie 1000 zapytań dziennie, co nie jest zbyt wiele, jeśli jest rozłożone na wielu graczy. Na szczęście, jeśli zarejestrujesz się do rozliczeń w Google, podniosą Twój limit do 150 000 żądań dziennie. Kod, którego użyjemy, będzie próbował zoptymalizować najmniejszą liczbę żądań, aby podczas testowania nie przekroczyć limitu 1000 żądań.

Uwaga : interfejs API Google Static Maps ma również limit 2500 żądań na adres IP. Jest to znacznie mniej restrykcyjne i jest powodem, dla którego nie zarejestrowaliśmy się. Co więcej, składamy prośbę o nową mapę tylko wtedy, gdy gracz wyszedł poza granice kafelka mapy.

Otwórz swoją ulubioną przeglądarkę internetową i wykonaj następujące instrukcje, aby wygenerować klucz API Miejsc Google:

1. Kliknij na: https://developers.google.com/places/web-service/getapi-key lub skopiuj go do przeglądarki.

2. Kliknij niebieski przycisk z napisem GET A KEY mniej więcej na środku strony

3. Zaloguj się na swoje konto Google lub utwórz, jeśli go nie masz.

4. Po zalogowaniu pojawi się okno dialogowe z prośbą o wybranie lub utworzenie nowego projektu. Wybierz, utwórz nowy projekt i nazwij go Foody GO lub cokolwiek innego, co uznasz za stosowne.

5. Kliknij łącze UTWÓRZ I WŁĄCZ API. Spowoduje to otwarcie okna dialogowego z Twoim kluczem API i linkami do materiałów początkowych. Upewnij się, że skopiowałeś klucz.

Teraz, gdy masz już klucz API, przetestujmy usługę REST, czyli interfejs API Miejsc Google. To ćwiczenie nie tylko pokaże nam, jakie informacje powracają podczas wyszukiwania, ale także pomoże nam zrozumieć interfejs API. Postępuj zgodnie z instrukcjami ćwiczeń tutaj:

1. Kliknij lub skopiuj podany adres URL: https://www.hurl.it/ do swojej przeglądarki. Hurl.it umożliwia szybkie i łatwe testowanie wywołań API REST w oknie przeglądarki.

2. U góry formularza wprowadź podstawowy adres URL interfejsu Google Places API w polu tekstowym twojaapihere.com w następujący sposób:

https://maps.googleapis.com/maps/api/place/nearbysearch/json.

3. Następnie kliknij link Dodaj parametr i wpisz nazwę jako typ i wartość jako żywność. W poniższej tabeli zdefiniowano parametry i wartości, które powinny być użyte w tym ćwiczeniu:

Nazwa : Wartość : Opis

type : jedzenie : rodzaj miejsca, które chcesz wyszukać. Oczywiście będziemy używać terminu jedzenie.

location : -33.8670,151.1957 : Współrzędne szerokości i długości geograficznej oddzielone przecinkiem.

radius : 500 : Jest to wyrażony w metrach promień obszaru, który ma być przeszukiwany od położenia centralnego.

key : YOUR KEY : Użyj klucza API, który wygenerowałeś w powyższym kroku.

4. Uzupełnij parametry zgodnie z odniesieniami w tabeli i upewnij się, że pasują.

5. Po zakończeniu dodawania parametrów potwierdź, że nie jesteś robotem, klikając pole wyboru, a następnie kliknij polecenie Uruchom.

6. Zakładając, że parametry zostały wprowadzone poprawnie, powinieneś teraz zobaczyć zmianę w: komunikat odpowiedzi pod formularzem. Wiadomość będzie dość długa i może wyglądać dość obco, w zależności od twoich wcześniejszych doświadczeń z JSON.

Pozostaw przeglądarkę otwartą i na stronie wyników. Wrócimy do zbadania wyników w następnej sekcji.

Korzystanie z JSON

JSON oznacza JavaScript Object Notation i jest definicją bardzo lekkiego formatu do przesyłania danych poprzez serializację obiektów. Oznacza to, że wiadomość, którą otrzymaliśmy z interfejsu Google Places API, jest w rzeczywistości zbiorem obiektów. Wszystko, co musimy zrobić, to odpowiednio przeanalizować te obiekty, a zrozumienie wyników wyszukiwania będzie proste. Unity faktycznie ma wbudowaną bibliotekę JSON. Ponieważ silnik Unity nie mógł skutecznie przeanalizować odpowiedzi, zdecydowano się na użycie biblioteki o nazwie TinyJson. TinyJson to kolejna biblioteka open source wyciągnięta z GitHub, ale części musiały zostać przepisane, aby obsługiwać platformę iOS. Pozostało jednak kilka wywołań przestrzeni nazw System.Linq. Jeśli planujesz uruchomić ten kod na urządzeniu z systemem iOS, upewnij się, że backend skryptów jest ustawiony na IL2CPP.

Uwaga : Jak wspomnieliśmy wcześniej, musisz uważać, jakich przestrzeni nazw C# używasz podczas programowania z myślą o iOS. Zwykle staramy się unikać przestrzeni nazw System.Linq, ponieważ może to być problematyczne podczas wdrażania na iOS.

Teraz, gdy mamy już pełny obraz, przyjrzymy się przykładowemu fragmentowi kodu, który pokazuje, w jaki sposób wykonamy żądanie wyszukiwania do interfejsu API:

```
//this code is run as part of a coroutine

var req = new

WWW("https://maps.googleapis.com/maps/api/place/ne

arbysearch/json?

location=-33.8670,151.1957&type=food&radius=500&ke

y={yourkeyhere}");

//yield until the service responds

yield return req;

//extract the JSON from the response

var json = req.text;

//use the TinyJson library JSONParser to deserialize

the result into

//an object called SearchResult

var searchResult =

TinyJson.JSONParser.FromJson<SearchResult>(json);
```

Ten kod jest bardzo podobny do kodu, którego użyliśmy do pobrania obrazów z API Google Static Maps, z tą różnicą, że teraz używamy metody WWW, aby zwrócić ciąg tekstowy JSON i przetworzyć go na obiekt o nazwie SearchResult. SearchResult jest definiowany przez odczytanie JSON i wyodrębnienie właściwości i hierarchii obiektów do definicji klas. Niestety musi to być proces ręczny, ponieważ nie możemy użyć dynamicznego generowania kodu, jeśli chcemy obsługiwać iOS. Na szczęście istnieje jednak wiele dostępnych narzędzi, które pozwalają nam przekonwertować JSON na wymagane definicje klas. Abyś mógł zobaczyć cały proces i magię JSON, użyjemy narzędzia online do skonstruowania naszej hierarchii klas SearchResult. Wykonaj następujące instrukcje, aby wykonać ćwiczenie:

1. Wróć do strony Hurl.it i skopiuj odpowiedź JSON. Upewnij się, że zawierasz wszystko, od początkowego nawiasu klamrowego ({) do końcowego nawiasu klamrowego (}) na dole.

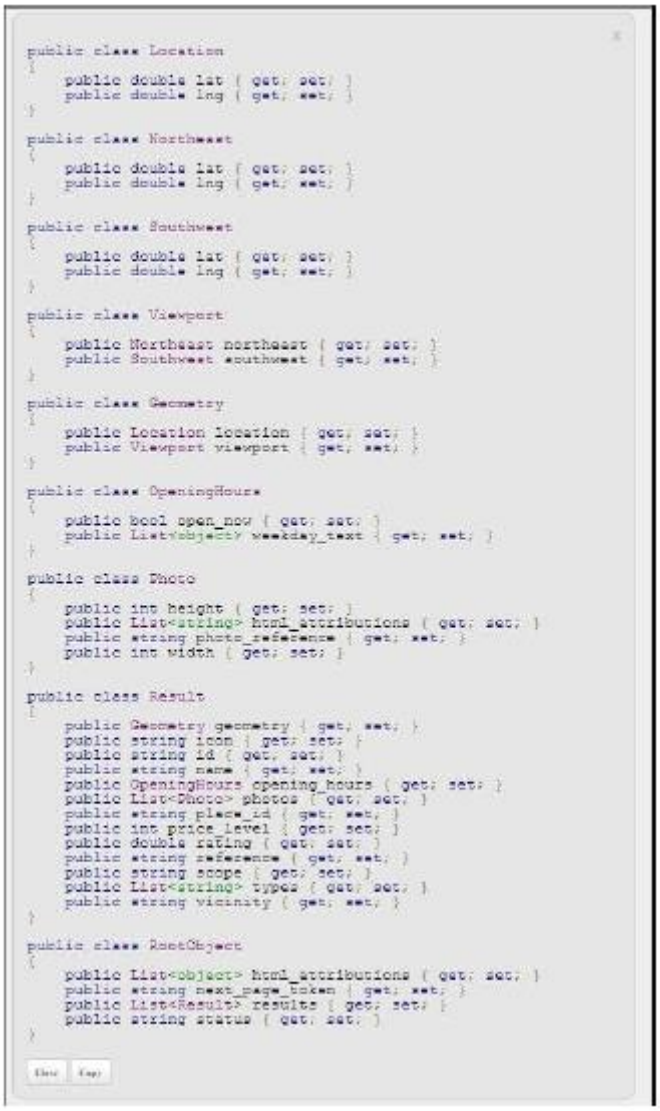
2. Skopiuj tekst JSON do schowka, wpisując Ctrl + C (

polecenie + C na Macu)

3. Otwórz inną kartę przeglądarki na http://json2csharp.com/.

4. Wklej skopiowany wcześniej plik JSON do pola JSON, wpisując Ctrl + V (polecenie + V na komputerze Mac).

5. Kliknij przycisk Generuj, aby utworzyć klasy C# w następujący sposób:



```
public class Location
{
    public double lat { get; set; }
    public double lng { get; set; }
}

public class Northeast
{
    public double lat { get; set; }
    public double lng { get; set; }
}

public class Southwest
{
    public double lat { get; set; }
    public double lng { get; set; }
}

public class Viewport
{
    public Northeast northeast { get; set; }
    public Southwest southwest { get; set; }
}

public class Geometry
{
    public Location location { get; set; }
    public Viewport viewport { get; set; }
}

public class OpeningHours
{
    public bool open_now { get; set; }
    public List<object> weekday_text { get; set; }
}

public class Photo
{
    public int height { get; set; }
    public List<string> html_attributions { get; set; }
    public string photo_reference { get; set; }
    public int width { get; set; }
}

public class Result
{
    public Geometry geometry { get; set; }
    public string icon { get; set; }
    public string id { get; set; }
    public string name { get; set; }
    public OpeningHours opening_hours { get; set; }
    public List<Photo> photos { get; set; }
    public string place_id { get; set; }
    public int price_level { get; set; }
    public double rating { get; set; }
    public string reference { get; set; }
    public string scope { get; set; }
    public List<string> types { get; set; }
    public string vicinity { get; set; }
}

public class RootObject
{
    public List<object> html_attributions { get; set; }
    public string next_page_token { get; set; }
    public List<Result> results { get; set; }
    public string status { get; set; }
}
```

6. Następnie skopiujemy ten kod i wkleimy go do naszego edytora kodu jako część skryptu i zmienimy nazwę klasy RootObject na naszą klasę SearchResult. RootObject to po prostu nazwa przypisana do głównego lub nienazwanego obiektu najwyższego poziomu w odpowiedzi. Przykładowy kod, który pokazaliśmy powyżej, oraz wygenerowana hierarchia klas zostały użyte do zbudowania usługi GooglePlacesAPIService. Sama usługa ma kilka innych funkcji, ale teraz przynajmniej rozumiesz, jak została zbudowana usługa i jak budować inne takie usługi, które wykorzystują JSON. W następnej sekcji skonfigurujemy nową usługę.

Konfigurowanie usługi Google Places API

Ponieważ zaimportowaliśmy już zaktualizowane skrypty, skonfigurowanie tej nowej usługi powinno być dla nas teraz proste. Wykonaj następujące instrukcje, aby skonfigurować i przetestować usługę GooglePlacesAPIService:

1. Wróć do edytora Unity. Przeciągnij scenę Mapa do okna Hierarchia z folderu Zasoby w oknie Projekt.

2. Rozwiń obiekt MapScene i usługi w oknie Hierarchia.

3. Kliknij prawym przyciskiem myszy (naciśnij klawisz Ctrl i kliknij na komputerze Mac) obiekt Usługi iz menu kontekstowego wybierz Utwórz pusty. Zmień nazwę nowego obiektu GooglePlacesAPI.

4. Przeciągnij skrypt GooglePlacesAPIService z folderu Assets/FoodyGo/Scripts/Services i upuść go na obiekt GooglePlacesAPI w oknach hierarchii lub inspektora.

5. Kliknij prawym przyciskiem myszy (naciśnij klawisz Ctrl i kliknij na komputerze Mac) na obiekcie MapScene w oknie Hierarchia iz menu kontekstowego wybierz Utwórz pusty. Zmień nazwę nowego obiektu PlaceMarker.

6. Kliknij prawym przyciskiem myszy (naciśnij Ctrl i kliknij na Mac) na obiekcie PlaceMarker w oknie Hierarchy iz menu kontekstowego wybierz Obiekt 3D | Cylinder.

7. Przeciągnij nowy obiekt PlaceMarker do folderu Assets/FoodyGo/Prefabs, aby uczynić go nowym prefabrykatem. Wyjedź oryginalnym obiektem w scenę, ale dezaktywuj go, odznaczając pole wyboru obok nazwy obiektu w oknie Inspektora.

8. Wybierz obiekt GooglePlacesAPI. Przeciągnij gotowy prefabrykat znacznika miejsca, który właśnie utworzyłeś, na puste miejsce na miejsce prefabrykatu znacznika miejsca.

9. Mając nadal wybrany obiekt GooglePlacesAPI, wypełnij właściwości, jak pokazano na poniższym zrzucie ekranu:

10. Upewnij się, że wpisałeś klucz, który wygenerowałeś dla klucza API w poprzedniej sekcji.

11. Kliknij prawym przyciskiem myszy (naciśnij klawisz Ctrl i kliknij na komputerze Mac) na scenie mapy i z menu kontekstowego wybierz opcję Usuń scenę. Upewnij się, że zapisałeś scenę po wyświetleniu monitu.

12. Naciśnij Play, aby uruchomić grę w edytorze i upewnij się, że usługa GPS jest ustawiona na symulację. Jeśli nadal używasz siedziby Google (37.62814, -122.4265) jako miejsca rozpoczęcia symulacji, powinieneś zobaczyć wiele obiektów cylindrycznych znaczników miejsc pojawiających się wokół postaci.

Notatka : Jeśli nie używasz współrzędnych Google w symulacji GPS i nie widzisz żadnych lokalizacji, sprawdź lokalizację w pobliżu wielu restauracji, sklepów spożywczych lub innych miejsc związanych z jedzeniem. Oczywiście, jeśli nadal będziesz mieć problemy, zapoznaj się z rozdziałem 10, Rozwiązywanie problemów.

Dzięki uruchomionej usłudze miejsc, gracz może teraz zobaczyć nowe obiekty wokół siebie. Nie pozwolimy jeszcze graczowi na interakcję z tymi obiektami. Jednak z pewnością nie chcemy też zwykłego starego cylindra reprezentującego te znaczniki. To, co musimy zrobić, to stworzyć znacznie lepiej wyglądający znacznik i zrobimy to w następnej sekcji.

Tworzenie znaczników

Generalnie, kiedy deweloperzy tworzą prototyp gry lub sceny w grze, ignorują szczegóły estetyczne i po prostu używają brzydkiego markera. To zazwyczaj działa, dopóki zespół projektowy nie dostarczy czegoś lepszego. Cóż, ponieważ nie mamy teraz zespołu projektowego, zamiast tego stworzymy lepiej wyglądający znacznik. Będzie to szczególnie dobre ćwiczenie, gdy w dalszej części książki będziemy tworzyć inne obiekty. Przeciągnij scenę mapy z powrotem do okna Hierarchia i wykonaj następujące instrukcje, aby utworzyć zaktualizowany znacznik miejsca:

1. Zlokalizuj i wybierz obiekt PlaceMarker, który zostawiliśmy w scenie Mapy i aktywuj go ponownie, zaznaczając pole obok nazwy w oknie Inspektora.

2. Wybierz obiekt Cylinder i zmień jego nazwę Base. W oknie Inspektora ustaw obiekty Transform | Skaluj do X=0,4, Y=0,1, Z=0,4 i przekształć | Pozycja do X=0, Y=-0,5, Z=0. Usuń komponent Capsule Collider, klikając ikonę koła zębatego obok komponentu i wybierając Usuń komponent.

3. Kliknij prawym przyciskiem myszy (naciśnij Ctrl i kliknij Mac) na PlaceMarker iz menu kontekstowego wybierz Obiekt 3D | Cylinder. Powtórz ten proces, aby utworzyć obiekty podrzędne Kula i Kostka.

4. Ustaw właściwości dla każdego z nowych obiektów podrzędnych, jak pokazano w poniższej tabeli:

Game object	Property/component	Value
Cylinder	Name	Pole
	Transform Position	(0, .5, 0)
	Transform Scale	(.05, 1, .05)
	Capsule Collider	Remove
Sphere	Name	Holder
	Transform Position	(0, 1.5, 0)
	Transform Scale	(.2, .2, .2)
	Sphere Collider	Remove
Cube	Name	Sign
	Transform Position	(0, 2, 0)
	Transform Scale	(1, 1, .1)
	Box Collider	Remove

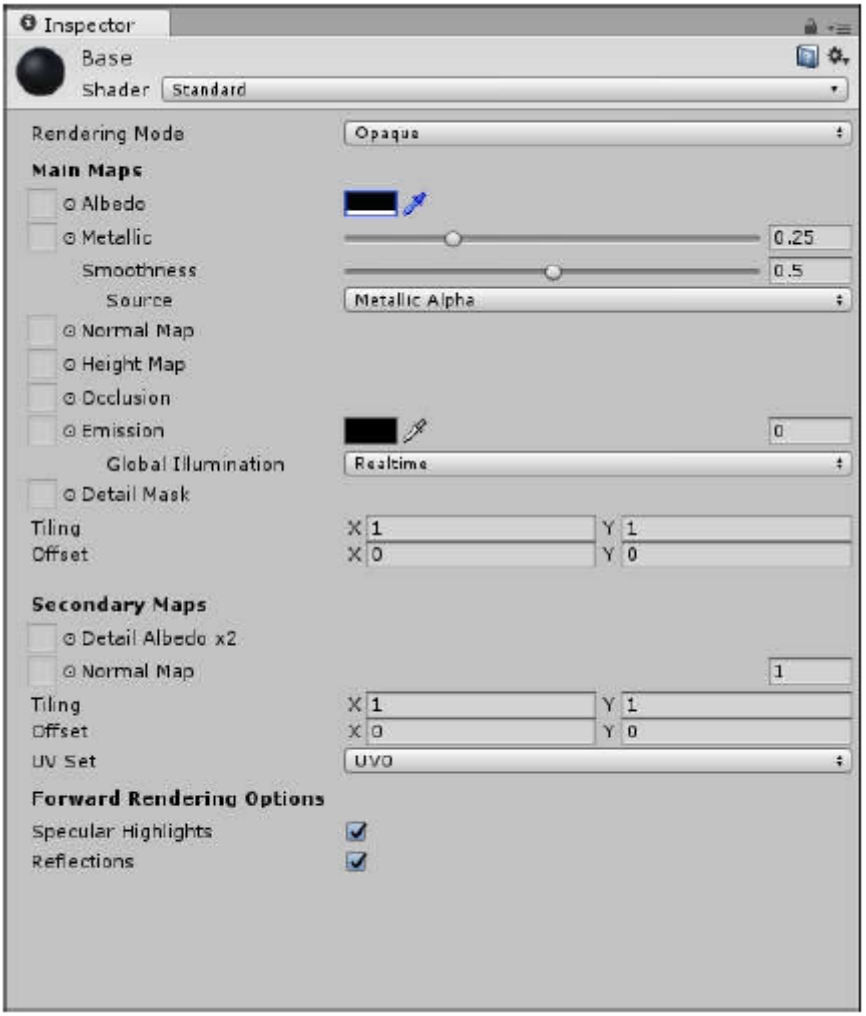
5. Kliknij prawym przyciskiem myszy (naciśnij Ctrl i kliknij Mac) na folder Assets/FoodyGo w oknie Projekt i z menu kontekstowego wybierz Utwórz | Teczka. Zmień nazwę nowego folderu Materiały.

6. Wybierz nowy folder Assets/FoodyGo/Material w oknie Projekt. Następnie kliknij prawym przyciskiem myszy (naciśnij Ctrl i kliknij Mac) pusty folder iz menu kontekstowego wybierz Utwórz | Materiał. Zmień nazwę materiału Baza. Powtórz ten proces i utwórz jeszcze dwa materiały o nazwie Highlight and Board.

7. Przeciągnij materiał bazowy z okna Projekt na obiekt bazowy w oknie Hierarchia. Powtórz ten proces dla materiału Highlight na słupie i uchwycie. Na koniec przeciągnij materiał tablicy na obiekt Sign.

8. Nowo utworzone materiały są nadal domyślnie białe, więc wizualnie nic się nie zmieni. Jednak posiadanie materiałów na obiekcie pozwoli nam zobaczyć zmiany bezpośrednio na obiekcie podczas edycji materiałów.

9. Wybierz materiał bazowy z okna projektu i przejdź do okna inspektora. Typowe okno właściwości zmieniło się w edytor właściwości cieniowania. Jeśli nie jesteś pewien, czym jest shader, nie martw się, omówimy to w następnym rozdziale. Na razie jednak edytuj właściwości w oknie, aby pasowały do następnego zrzutu ekranu:



10. Jeśli nie masz pewności, kolor Albedo dla materiału podstawowego powinien być czarny (#00000000). Zwróć także uwagę, jak zmienia się podstawa właściwości materiału PlaceMarker podczas edycji materiału.

11. Edytuj właściwości materiału shadera dla każdego z materiałów, jak pokazano w tabeli:

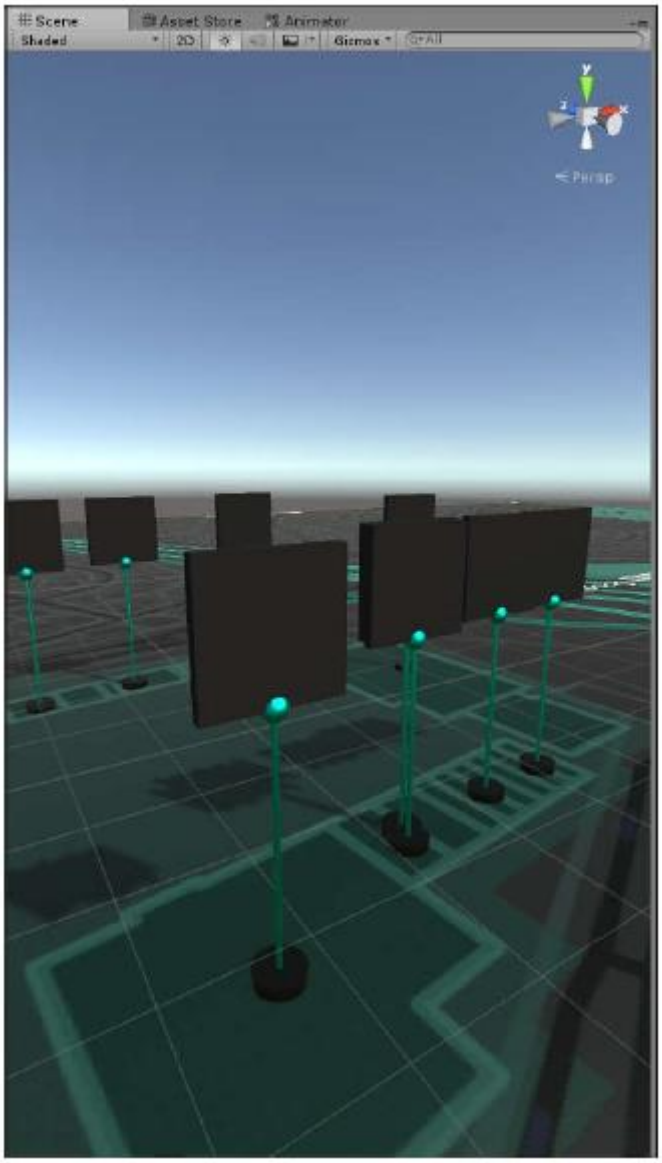
Material	Property	Value
Base	Albedo Color	#00000000
	Metallic	.25
Highlight	Albedo Color	# 00FFE9FF
	Metallic	1
Board	Albedo Color	#090909FF
	Metallic	0
	Smoothness	0

12. Wybierz prefabrykat PlaceMarker w Hierarchii, a następnie kliknij przycisk Zastosuj pod opcjami Prefabrykatów na górze okna Inspektora. To zaktualizuje prefabrykat ze wszystkimi zmianami.

13. Następnie dezaktywuj obiekt w scenie, odznaczając pole wyboru obok nazwy.

14. Zapisz i usuń scenę mapy z okna Hierarchia.

15. Naciśnij Play w edytorze, aby przetestować grę. Powinieneś teraz widzieć nowy znacznik miejsca wypełniający mapę, jak pokazano w oknie sceny:



Skonstruowane przez nas znaczniki miejsc mają przypominać połączenie znacznika na stole w restauracji i menu na tablicy. Pozostawimy tę część menu pustą, dopóki nie pozwolimy graczowi na interakcję ze znacznikami miejsc, w następnym rozdziale. Na razie jednak nadal musimy rozwiązać kilka problemów z naszym obecnym wyszukiwaniem interfejsu API Miejsc Google, które omówimy w następnej sekcji.

Optymalizacja wyszukiwania

Obecnie nasza usługa wyszukiwania Google Places API przeprowadza wyszukiwanie w pobliżu i zwraca stronę (20) szczegółowych wyników na przerysowanej mapie. Oczywiście, gdybyśmy mieli więcej niż 20 lokalizacji, które pasowały do naszych wyszukiwań w okolicy, brakowałoby nam kilku lokalizacji. Moglibyśmy zaktualizować nasze prośby, aby przeglądać wyniki, ale wtedy zaczniemy składać liczne prośby o przerysowanie każdej mapy. Pamiętaj, że używany przez nas interfejs API nie jest darmowy i ma ścisłe ograniczenia dotyczące liczby wniosków. Na szczęście istnieje inna opcja wyszukiwania, której możemy użyć, aby zwrócić lokalizacje wokół gracza. Interfejs API obsługuje również opcję wyszukiwania radaru. Wyszukiwanie radarowe zwróci do 200 lokalizacji dla przeszukiwanego obszaru, ale wyniki będą zawierały tylko geometrię i identyfikatory. To zadziała dla nas dobrze, więc edytujmy kod, aby wprowadzić tę zmianę i przetestuj ją:

1. Kliknij dwukrotnie skrypt GooglePlacesAPIService w folderze Assets/FoodyGo/Scripts/Services w oknie projektu, aby otworzyć skrypt w wybranym edytorze.

2. Przewiń w dół do metody IEnumerator SearchPlaces() i zmień następujący wiersz kodu na to, co jest tutaj:

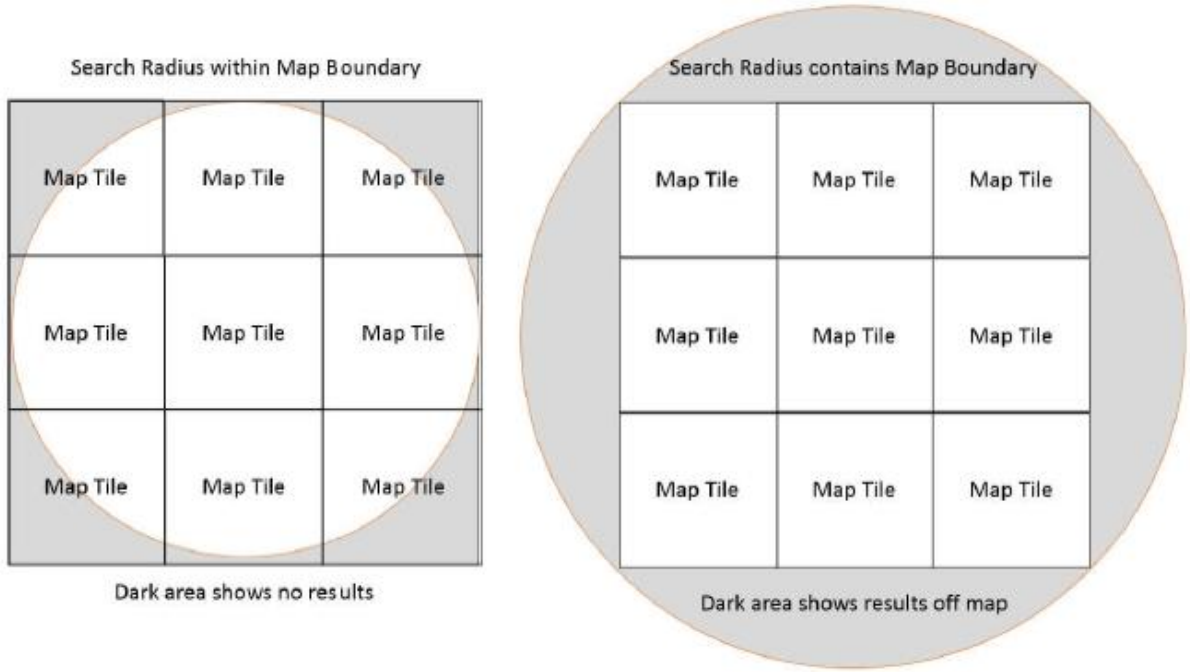
```
//line to change

var req = new

WWW(GOOGLE_PLACES_NEARBY_SEARCH_URL + "?" +

queryString);

//change to RADAR

var req = new

WWW(GOOGLE_PLACES_RADAR_SEARCH_URL + "?" +

queryString);
```

3. Zapisz zmiany w pliku i wróć do edytora Unity; poczekaj, aż skrypty się skompilują.

4. Naciśnij Play, aby ponownie uruchomić grę i zauważ, że wszystko wygląda mniej więcej tak samo jak wcześniej.

5. Teraz załaduj scenę Map do okna Hierarchy, a następnie przejdź do MapScene | Usługi | Obiekt GooglePlacesAPI. Zmień wartość właściwości Odległość wizualna na 2000. Usuń i zapisz scenę mapy.

6. Ponownie przetestuj grę w edytorze i zauważ, że tym razem w oddali znajduje się kilka znaczników miejsc, a nawet nie na mapie. Oczywiście wartość 2000 jest zbyt duża dla naszej odległości wizualnej lub promienia wyszukiwania. Jakiej wartości powinniśmy użyć, aby wypełnić tylko mapę wynikami wyszukiwania? Krótka odpowiedź brzmi: to zależy. Być może, jeśli spojrzymy na nasz problem wizualnie, możemy określić kilka możliwych rozwiązań naszego problemu odległości wyszukiwania. Spójrz na poniższy diagram:



Jak pokazuje diagram, problemem, przed którym stoimy, jest próba zmniejszenia promienia wyszukiwania tak, aby znajdował się w granicach mapy lub zawierał granicę mapy. Jeśli promień wyszukiwania znajduje się w granicach naszej mapy, części naszej mapy nie będą miały pokazanej lokalizacji (ciemne obszary). Alternatywnie, jeśli promień zawiera granicę, wyszukiwanie zwróci liczbę lokalizacji, których nie będziemy mogli zmapować (ciemne obszary). W idealnym przypadku chcielibyśmy mieć możliwość wyszukiwania według granic naszej mapy, ale niestety interfejs API Miejsc Google nie obsługuje tego. Musimy więc zdecydować, czy przeprowadzać wyszukiwanie wewnątrz, czy zawiera.

Uwaga : Większość innych konwencjonalnych usług map zapewnia zapytanie o granice mapy w postaci ramki ograniczającej. To pole jest zwykle definiowane przez ustawienie północno-zachodniego i południowo-wschodniego punktu naszej mapy jako parametrów wyszukiwania. Bardziej niezawodne i zaawansowane usługi mapowe będą również obsługiwać wyszukiwanie wielokątów.

Aby przeprowadzić filtrowanie wyszukiwania, będziemy musieli wykonać kilka zmian kodu w klasach GPSLocationService i GooglePlacesAPIService. Wykonaj następujące instrukcje, aby przejrzeć te zmiany w skrypcie:

1. Z menu wybierz Zasoby | Importuj pakiet | Pakiet niestandardowy..., a gdy otworzy się okno dialogowe Importuj pakiet, przejdź do folderu pobranego kodu źródłowego C7A i wybierz plik Chapter7_import2.unitypackage. Importujemy zaktualizowane skrypty nie dlatego, że jest ich dużo, ale dlatego, że w dużych plikach jest tylko kilka zmian.

2. Otwórz skrypt GPSLocationService w wybranym przez siebie edytorze. Następnie znajdź metodę CenterMap. Ta metoda przelicza różne ważne parametry mapy po każdym przerysowaniu. Na dole tego pliku dodano obliczenia dla nowej zmiennej o nazwie mapBounds, jak następuje:

lon1 =

GoogleMapUtils.adjustLonByPixels(Longitude, -

MapTileSizePixels*3/2 , MapTileZoomLevel);

```
lat1 =

GoogleMapUtils.adjustLatByPixels(Latitude,

MapTileSizePixels*3/2 , MapTileZoomLevel);

lon2 =

GoogleMapUtils.adjustLonByPixels(Longitude,

MapTileSizePixels*3/2 , MapTileZoomLevel);

lat2 =

GoogleMapUtils.adjustLatByPixels(Latitude, -

MapTileSizePixels*3/2 , MapTileZoomLevel);

mapBounds = new MapEnvelope(lon1,

lat1, lon2, lat2);
```

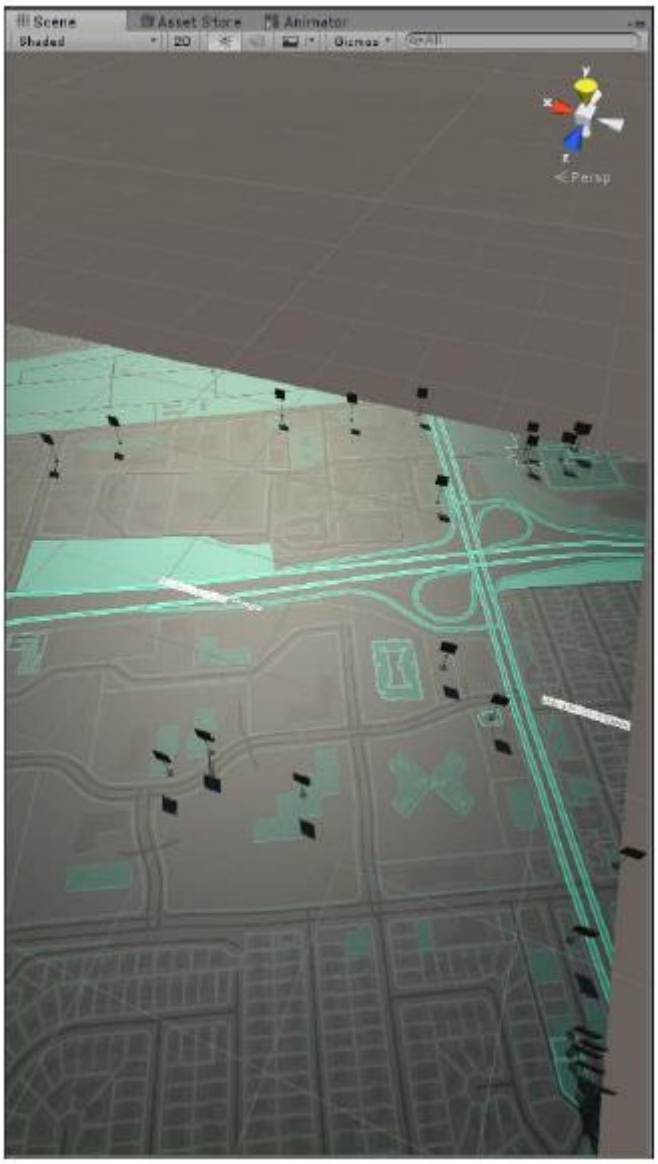
3. Wszystko, co robi ten kod, to obliczanie granic szerokości i długości geograficznej mapy. Zachęcamy do zapoznania się z sekcją poprzedzającą ten kod, ponieważ minęło trochę czasu (Rozdział 2, Mapowanie lokalizacji gracza) od ostatniego sprawdzania tego kodu.

4. Następnie otwórz skrypt GooglePlacesAPIService w swoim edytorze. Znajdź metodę UpdatePlaces w górnej części pliku. Metoda UpdatePlaces to miejsce, w którym aktualizujemy lokalizacje na mapie. Metoda iteruje po nowych wynikach wyszukiwania i umieszcza obiekty na mapie. W tej metodzie znajdziesz następujący kod:

```
if(GPSLocationService.Instance.mapBounds.

Con tains(new MapLocation((float)lon,

(float)lat))==false)

{

continue;

}
```

5. Ten fragment kodu został dodany i używa nowego pola mapBounds w usłudze GPSLocationService, aby określić, czy szerokość i długość geograficzna wyniku wyszukiwania znajdują się w granicach mapy. Jeśli wynik wyszukiwania znajduje się poza granicą mapy, wynik wyszukiwania jest ignorowany, a pętla jest kontynuowana za pomocą instrukcji continue.

6. Naciśnij Play, aby uruchomić grę. Tym razem przejdź do okna Scena i zbliż kamerę daleko ponad mapę i spójrz w dół. Zauważ teraz, że wszystkie znaczniki miejsc znajdują się w granicach mapy, jak pokazano na poniższym zrzucie ekranu:



Widok sceny mapy ze wszystkimi znacznikami miejsc w granicach mapy Usługa GooglePlacesAPIService teraz zwraca wyniki obejmujące całą mapę, a my rozwiązaliśmy nasz problem z wyszukiwaniem. Upewnij się, że poświęcisz trochę czasu na przejrzenie reszty skryptu, aby zrozumieć, jak wszystko idzie w parze.

Podsumowanie

W tej części przeskoczyliśmy z powrotem do mapy, aby dodać do gry kilka nowych funkcji zlokalizowanych w świecie rzeczywistym. Zanim dodaliśmy nowe funkcje, musieliśmy naprawić niektóre problemy, które wynikły ze zmian, które wprowadziliśmy pod koniec ostatniego rozdziału. Wymagało to od nas przekonwertowania naszej usługi GPS na użycie wzorca Singleton. W ramach konwersji skorzystaliśmy z okazji, aby zrozumieć, jak działał Singleton. Następnie poświęciliśmy trochę czasu na zapoznanie się z interfejsem Google Places API, czyli usługą internetową, której używamy do lokalizowania interesujących miejsc w pobliżu odtwarzacza. Wymagało to od nas stworzenia klucza API i zrozumienia, jak wysyłać żądania przeciwko usłudze za pomocą Hurl.it. Wykorzystaliśmy Hurl.it do przetestowania naszych zapytań, a następnie zrozumieliśmy, jak wyniki zwracane w formacie JSON mogą być konwertowane na obiekty C# w czasie wykonywania przy użyciu TinyJson. Po zaimportowaniu i przygotowaniu naszego skryptu konfigurujemy nową usługę w scenie mapy.

Następnie zbudowaliśmy lepszy prototyp naszego znacznika miejsca, używając prymitywnych obiektów 3D i niestandardowych materiałów. Na koniec ustaliliśmy, że musimy rozwiązać niektóre problemy z naszym wyszukiwaniem, co zrobiliśmy za pomocą innego prostego importu skryptu. Następnie dokonaliśmy przeglądu zmiany i byliśmy zadowoleni z wyników końcowych. Ponieważ znaczniki miejsc są teraz wyświetlane na mapie, wykorzystamy następny rozdział, aby umożliwić graczowi interakcję z tymi znacznikami poprzez zbieranie obiektów i umieszczanie potworów. Będzie to wymagało od nas ulepszenia ekranów ekwipunku, które opracowaliśmy wcześniej. Ponadto spędzimy dodatkowy czas na ulepszaniu gry za pomocą efektów cząsteczkowych i wizualnych.