Interakcja ze światem AR

Nasz wirtualny świat gry jest teraz wypełniony miejscami z prawdziwego świata, do których chcemy, aby gracz podróżował i wchodził w interakcję. W tej wersji gry pozwolimy graczowi odwiedzić otaczające go miejsca i spróbować sprzedać jego gotujące potwory. Jeśli miejsce lub restauracja lubi któryś z kucharzy gracza, zaoferuje płatność. Gracz może wtedy zaakceptować płatność lub przejść dalej. Jeśli jednak gracz zaakceptuje płatność, kucharz zostanie sprzedany do restauracji i zostanie jej szefem kuchni. Restauracja z szefem-potworem nie kupi więcej kucharzy, dopóki obecny szef nie odejdzie. Szef kuchni wyjdzie po upływie określonego czasu. W tym rozdziale skupimy się na ukończeniu interakcji gracza eksplorującego otaczające go miejsca. W tym celu będziemy musieli zbudować nową scenę Miejsca. Ta scena pozwoli graczowi sprzedać swoje potwory, aby zebrać nowe przedmioty, takie jak kule zamrażające i doświadczenie. Będzie to wymagało od nas wprowadzenia nowych tabel do bazy danych, obsługi nowych elementów ekwipunku, doświadczenia gracza lub poziomu oraz śledzenia historii miejsca. Po drodze wprowadzimy również kilka subtelnych ulepszeń. Więc przygotuj się; ta część omówi te materiały bardzo szybko i obejmie następujące tematy:

* Scena Miejsc

* Google Street View jako tło

* Pokaz slajdów ze zdjęciami Google Places API

* Dodanie interakcji z interfejsem użytkownika do sprzedaży

* Mechanika sprzedaży w grze

* Aktualizacja bazy danych

* Łączenie elementów

W tej części zostanie wprowadzonych kilka nowych elementów, ale większość zawartości dodanej w tym momencie do gry to rozszerzenie koncepcji omówionych w kilku poprzednich rozdziałach. Jak zawsze będziemy kontynuować projekt od miejsca, w którym zostawiliśmy go z poprzedniej części.

Scena Miejsc

Ta scena będzie kolejnym punktem, w którym chcemy mieszać zawartość naszego alternatywnego świata gry ze światem rzeczywistym. Podobnie jak w scenie Catch, gdzie nasza interakcja AR była światem gry na szczycie świata rzeczywistego, zrobimy coś podobnego, ale tym razem z użyciem tła z Google Street View. Wykorzystamy również zdjęcia miejsc w naszych znacznikach z prawdziwego świata, aby poszerzyć naszą rzeczywistość w grze. Zacznijmy od stworzenia nowej sceny i umieszczenia głównych elementów bazowych w następujący sposób:

1. Otwórz projekt FoodyGo, z którego wyszliśmy w poprzedniej części. Jeśli skoczyłeś do przodu, będziesz musiał zapoznać się z sekcją dotyczącą przywracania projektu z pobranego kodu źródłowego.

2. Wpisz Ctrl + N (polecenie + N na Macu), aby utworzyć nową scenę.

3. Wpisz Ctrl + S (polecenie + S na Macu), aby zapisać scenę. Zapisz scenę jako Miejsca.

4. Wpisz Ctrl + Shift + N (polecenie + shift + N na Macu), aby utworzyć nowy pusty obiekt gry. Zmień nazwę obiektu PlacesScene i zresetuj transformację do zera.

5. Przeciągnij obiekty kamery głównej i światła kierunkowego na scenę PlacesScene, aby uczynić je obiektami potomnymi.

6. Wybierz obiekt Kamera główna w oknie Hierarchia. Następnie w oknie Inspektora ustaw pozycję Transformuj Y = 2 i Z=-2. Chcemy podnieść kamerę do góry i przesunąć ją do przodu.

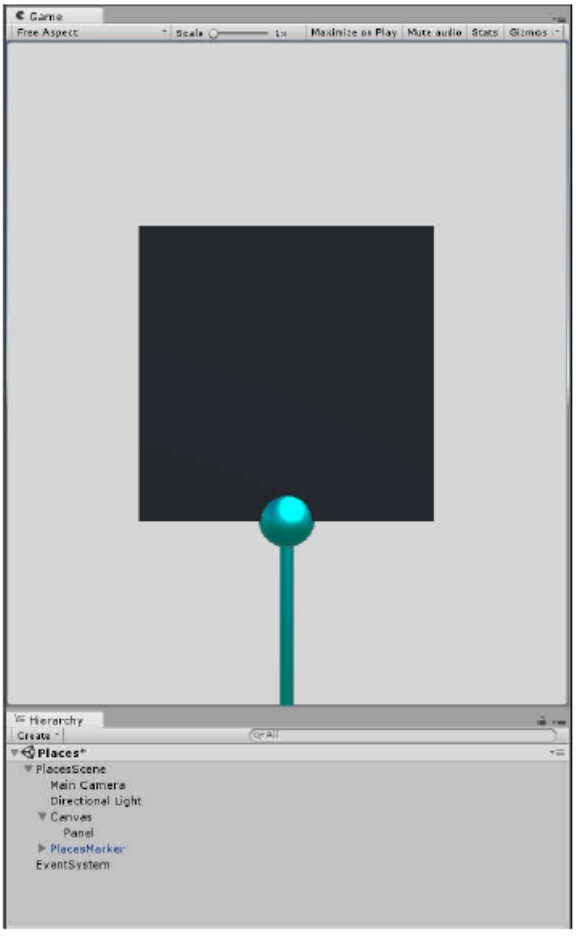
7. Z menu wybierz GameObject | interfejs użytkownika | Płyta. Spowoduje to dodanie nowego płótna z panelem podrzędnym w oknie hierarchii. Wybierz obiekt Płótno.

8. Zlokalizuj komponent Canvas w oknie Inspektora i ustaw Tryb renderowania na Przestrzeń ekranu - Kamera. Następnie przeciągnij obiekt Kamera główna z okna Hierarchia do pustego gniazda Kamera renderowania w komponencie Canvas.

9. Wybierz obiekt Panel w oknie Hierarchia. Następnie w oknie Inspektora zmień Kolor składnika obrazu na #FFFFFFFF, klikając gniazdo koloru, aby otworzyć okno dialogowe Kolor i wprowadzić wartość szesnastkową.

10. Zlokalizuj prefabrykat PlaceMarker w folderze Assets/FoodyGo/Prefabs w oknie projektu. Wybierz prefabrykat i naciśnij Ctrl + D (polecenie + D na Macu), aby zduplikować prefabrykat. Zmień nazwę prefabrykatu PlacesMarker i przeciągnij nowy prefabrykat do okna Hierarchii.

11. Na koniec przeciągnij obiekty Canvas i PlacesMarker na obiekt PlacesScene w oknie Hierarchy, aby uczynić je dziećmi. Twoje okna gry i hierarchii powinny pasować do poniższego zrzutu ekranu:



To tworzy naszą podstawową scenę Miejsc, a teraz chcemy dodać inne elementy AR, takie jak tło, i zalogować się w następnych sekcjach.

Google Street View jako tło

Podobnie jak w przypadku sceny Catch, którą stworzyliśmy wcześniej, w której jako tło wykorzystano kamerę, chcemy użyć jako tła elementu ze świata rzeczywistego. Zamiast aparatu w tej scenie zostanie wykorzystany Google Street View. Street View będzie w stanie zapewnić ciekawe tło sceny bez użycia aparatu. Zanim zaczniemy, będziesz musiał utworzyć klucz API Map Google podobny do tego, co zrobiłeś w poprzednim rozdziale. Przejdź pod ten adres URL do interfejsu Google Street View Image API, aby wygenerować klucz programisty, tak jak poprzednio:

https://developers.google.com/maps/documentation/streetview/.

Kliknij przycisk GET A KEY u góry strony i postępuj zgodnie z procesem, tak jak poprzednio, aby wygenerować klucz. Po wygenerowaniu klucza programisty wykonaj następujące czynności, aby skonfigurować tło za pomocą interfejsu Google Street View Image API:

1. Z menu wybierz Zasoby | Pakiet importowy | Pakiet niestandardowy..., aby otworzyć okno dialogowe Importuj pakiet. W oknie dialogowym przejdź do folderu z pobranym kodem źródłowym C8A. Wybierz pakiet Chapter8_import1.unity i kliknij Otwórz.

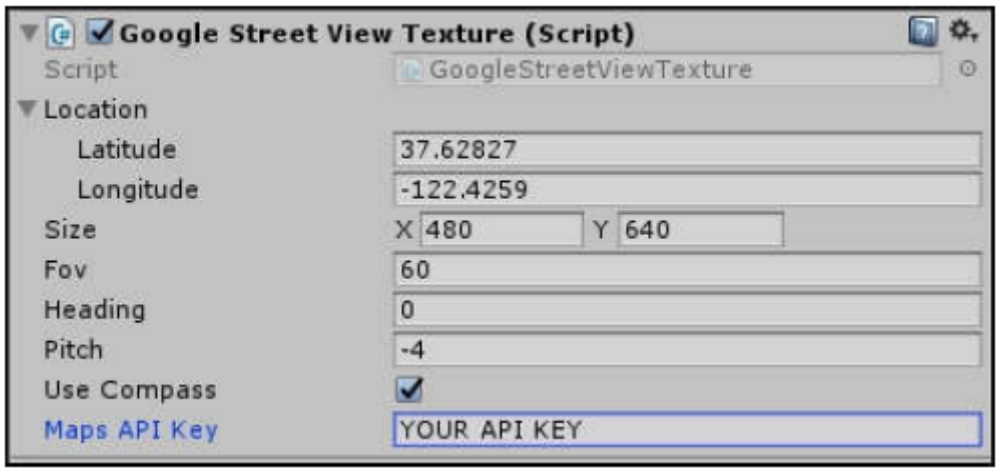
2. Po otwarciu okna dialogowego Import Unity Package upewnij się, że wszystko jest zaznaczone i kliknij Importuj.

3. Wybierz obiekt Panel w oknie Hierarchia. Następnie w oknie Inspektora usuń komponent Obraz, klikając ikonę koła zębatego obok komponentu i wybierając Usuń komponent z menu kontekstowego.

4. Z menu wybierz Komponent | interfejs użytkownika | Surowy obraz. Spowoduje to dodanie nowego komponentu Raw Image do Panelu. Zmień nazwę panelu StreeViewTexturePanel.

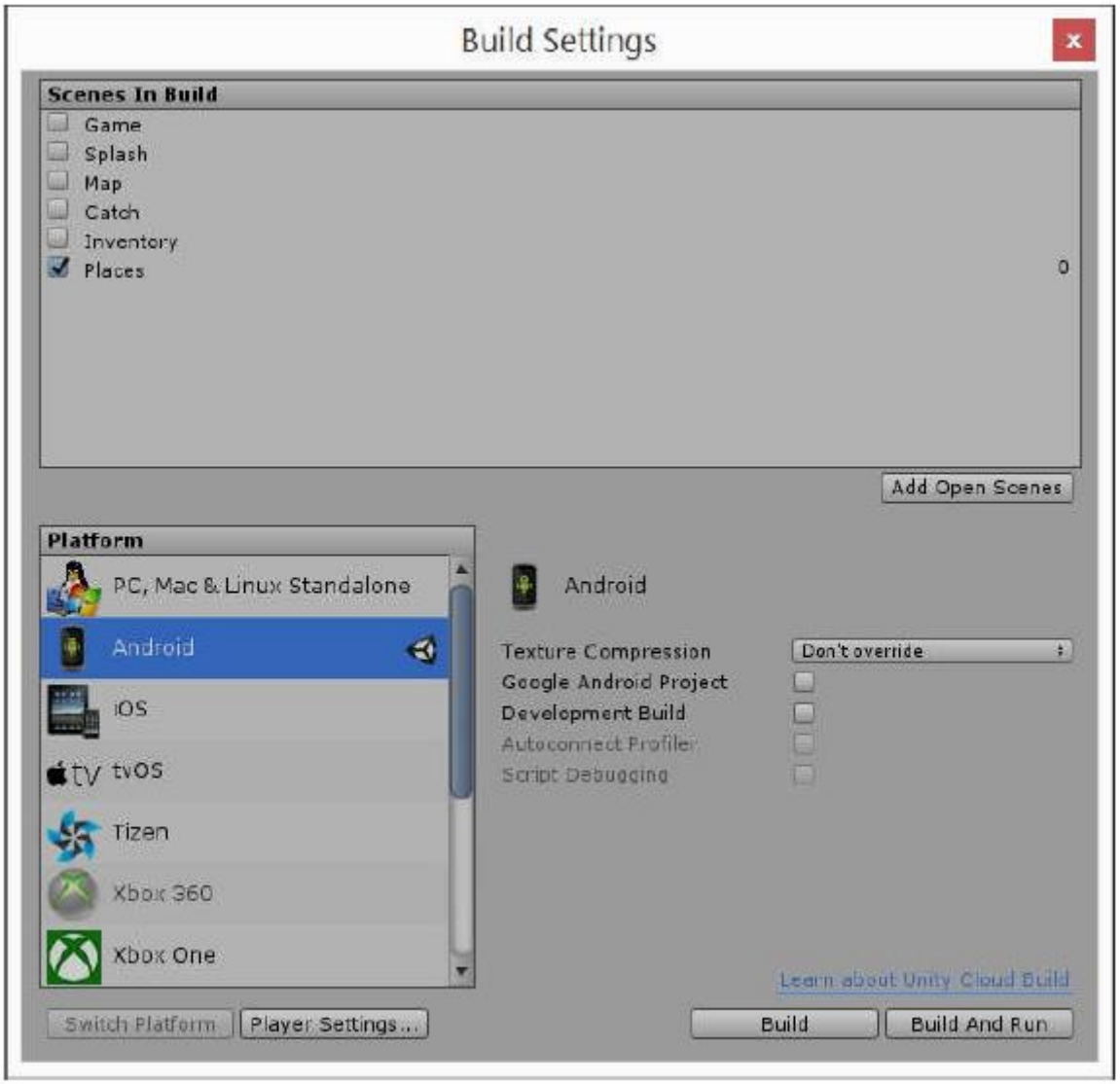
5. Z folderu Assets/FoodyGo/Scripts/Mapping w oknie Project przeciągnij GoogleStreetViewTexture na obiekt StreetViewTexturePanel w oknie Hierarchy lub Inspector.

6. W oknie Inspektora ustaw właściwości komponentu Google Street View Texture na wartości w następujący sposób:



7. Naciśnij Play, aby uruchomić grę w edytorze i zwróć uwagę, że tłem jest teraz ulica poza biurem Google.

8. Z menu wybierz Plik | Ustawienia budowania... i kliknij przycisk Dodaj otwarte sceny, aby dodać scenę Miejsca do scen w kompilacji. Następnie usuń zaznaczenie innych scen, jak pokazano w oknie dialogowym Ustawienia kompilacji:



9. Zbuduj i wdróż grę na swoje urządzenie mobilne. Jak scena działa, obróć urządzenie i obserwuj, jak zmienia się tło. Tło zmieni się zgodnie z nagłówkiem w telefonie.

Korzystanie z interfejsu Google Street View Image API doskonale sprawdza się jako tekstura tła, gdy gracz angażuje się w obszary mapowane przez Street View. Jednak nie wszystkie lokalizacje (szerokość/długość geograficzna) będą miały zdjęcia Street View. Można to naprawić, zmieniając tło dla lokalizacji bez zdjęć, ale na razie zostawimy to bez zmian. Skrypt GoogleStreetViewTexture działa podobnie jak skrypty Google Places i Google Maps API, które opracowaliśmy wcześniej. Otwórzmy skrypt i spójrzmy na główną metodę wykonującą zapytanie w następujący sposób:

```
IEnumerator LoadTexture()

{

var queryString = string.Format("location=

{0}&fov={1}&heading={2}&key={3}&size=

{4}x{5}&pitch={6}", location.LatLong, fov,

heading, MapsAPIKey, size.x, size.y, pitch);
```

```
var req = new WWW(GOOGLE_STREET_VIEW_URL + "?"

+ queryString);

//yield until the service responds

yield return req;

//first destroy the old texture first

Destroy(GetComponent<RawImage>

().material.mainTexture);

//when the image returns set it as the tile

texture

GetComponent<RawImage>().texture =

req.texture;

GetComponent<RawImage>().material.mainTexture

= req.texture;

}
```

Jak wspomniano, wygląda to bardzo podobnie do innego kodu, którego używaliśmy wcześniej, aby uzyskać dostęp do innych interfejsów API Google. Najważniejszą rzeczą, na której się skupimy, są parametry zapytania przesyłane do API:

* lokalizacja: są to współrzędne szerokości i długości geograficznej oddzielone przecinkiem. Później, gdy połączymy ten skrypt z usługą GPS, automatycznie wypełni te wartości.

* fov: Jest to pole widzenia w stopniach obrazu, które zasadniczo odpowiada powiększenia. Używamy wartości 60, która reprezentuje rozsądne zawężenie naszego świata w celu dopasowania do wymiarów ekranu urządzenia.

* kurs: jest to kurs kompasu w stopniach północnych. Zezwalamy, aby kurs kompasu urządzenia ustawiał tę wartość, jeśli opcja jest włączona, w przeciwnym razie 0.

* rozmiar: jest to rozmiar żądanego obrazu. Maksymalny rozmiar to 640 x 640. Używamy wartości 430 x 640 ze względu na widok pionowy.

* pitch: jest to kąt powyżej lub poniżej horyzontu. Wartości mogą mieścić się w zakresie od -90 do 90. Używamy wartości -4, aby uzyskać trochę ziemi na zdjęciach Street View.

* klucz: jest to klucz API wygenerowany dla Google Street View API.

Zachęcamy do samodzielnego przejrzenia reszty skryptu GoogleStreetViewTexture. Resztę kodu znajdziesz podobnie do tego, co zrobiliśmy w skryptach Miejsc i Map Google. W następnej sekcji omówimy dodatkowe zapytania dotyczące interfejsu Google Places API.

Pokaz slajdów ze zdjęciami Google Places API

W poprzedniej części wykorzystaliśmy Google Places API w celu wypełnienia znaczników miejsc na mapie wyszukiwaniem radarowym. Teraz, gdy gracz wchodzi w interakcję ze znacznikiem, chcemy, aby

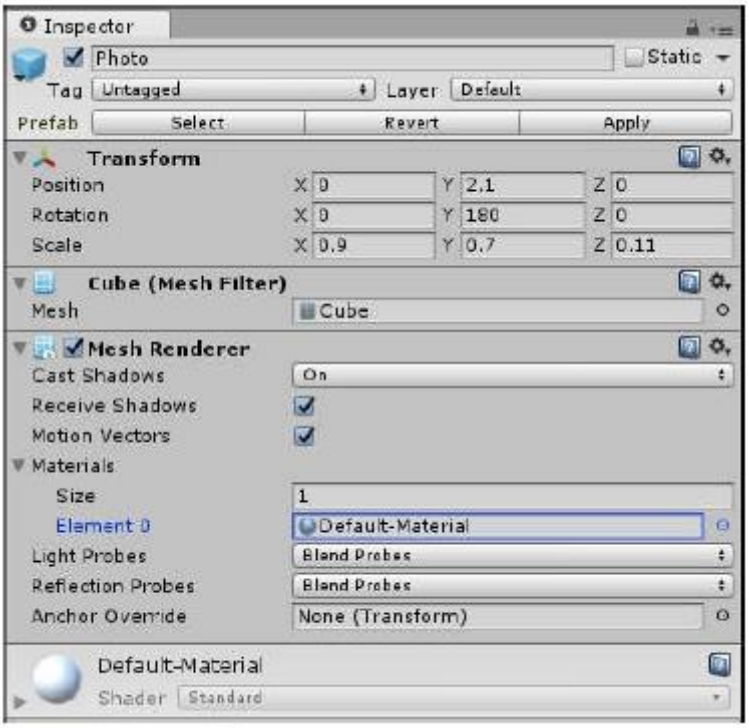
mógł wizualizować informacje o lokalizacji. Możemy to zrobić wykonując szczegółowe zapytanie o miejsce, które wróci dodatkowe informacje oraz załączone zdjęcia. Te zdjęcia posłużą do stworzenia pokazu slajdów dla gracza podczas wizyty w tym miejscu. Wraz z pokazem slajdów dodamy również tekst o nazwie miejsca i aktualnej ocenie. Później użyjemy oceny, aby określić, jakie potwory gotujące dane miejsce będzie chciało kupić i za ile. Będziesz potrzebować klucza Google Places API, który wygenerowaliśmy w poprzednim rozdziale, aby dokończyć konfigurację w następnej sekcji. Jeśli przeskoczyłeś wcześniej z poprzedniej części, przejrzyj proces generowania klucza API w poprzedniej części. Zaczniemy od dodania nowych elementów potrzebnych do sceny, postępując zgodnie z instrukcjami tutaj:

1. Wróć do sceny Places i zlokalizuj PlacesMarker w oknie Hierarchy.

2. Rozwiń PlacesMarker i wybierz obiekt Sign w oknie Hierarchia. Naciśnij Ctrl + D (polecenie + D na Macu), aby zduplikować obiekt. Zmień nazwę nowego obiektu Zdjęcie.

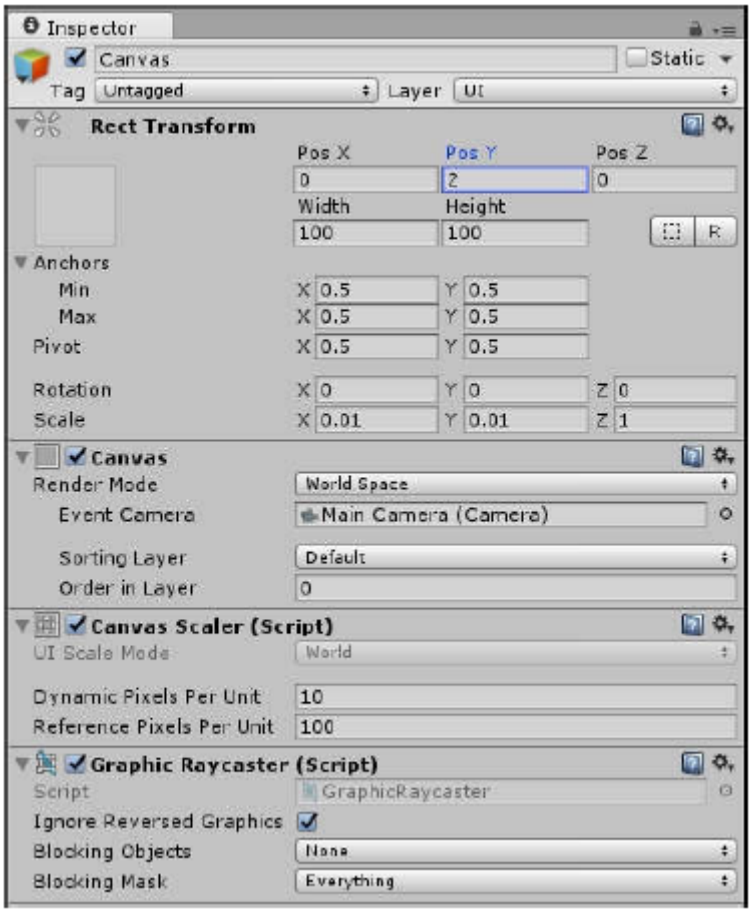
3. Po wybraniu nowego obiektu Photo w oknie Inspektora zmień Renderer siatki | Materiał, klikając ikonę tarczy, aby otworzyć okno dialogowe Wybierz materiał. Następnie przewiń w dół listy i wybierz opcję Domyślny materiał. Po dokonaniu wyboru zamknij okno dialogowe i potwierdź, że materiał został wymieniony.

4. Zmień właściwości Transform obiektu Photo, aby pasowały do poniższego zrzutu ekranu:



5. Z menu wybierz GameObject | interfejs użytkownika | Canvas, aby utworzyć nowy obiekt Canvas. Przeciągnij obiekt Canvas na obiekt PlacesMarker w oknie Hierarchia. Dodaliśmy kanwę do obiektu, dzięki czemu możemy dodać tekst bezpośrednio do obiektu.

6. Po wybraniu nowego płótna w oknie Inspektora zmień płótno | Renderuj tryb do przestrzeni świata, a następnie zmień inne właściwości, aby dopasować wartości, jak pokazano na poniższym zrzucie ekranu:



Uwaga: po zmianie kanwy | . konieczne będzie ręczne ustawienie wszystkich właściwości w przekształceniu prostokąta Tryb renderowania do przestrzeni świata.

7. Kliknij prawym przyciskiem myszy (naciśnij Ctrl i kliknij Mac) na obiekcie Canvas iz menu kontekstowego wybierz UI | Tekst. Zmień nazwę nowego nagłówka elementu.

8. W oknie Inspektora zmień właściwości obiektu Nagłówek, aby odpowiadały wartościom:

Przekształcenie prostopadłe: Poz Z = -.1

Przekształcenie prostokąta: szerokość = 90, wysokość = 70

Tekst: Rozmiar czcionki = 10

Tekst: Wyrównanie akapitu = do lewej i do dołu

Tekst: Kolor = #FFFFFFFF (biały)

9. Wybierz obiekt Header w oknie Hierarchy i naciśnij Ctrl + D (polecenie + D na Macu), aby zduplikować obiekt. Zmień nazwę obiektu Ocena.

10. Wybierz nowy obiekt Ocena iw oknie Inspektora zmień właściwości tak, aby odpowiadały wartościom w następujący sposób:

Transformacja prostokątna: wysokość = 90

Tekst: Rozmiar czcionki = 7

Tekst: Czcionka = fontawsome-webfont

Tekst: Wyrównanie akapitu = do prawej i do dołu

Tekst: Kolor = #00FFFFFF (cyjan)

11. Wybierz obiekt Ocena i naciśnij Ctrl + D (polecenie + D na Macu), aby zduplikować obiekt. Zmień nazwę obiektu Price w oknie Inspektora i ustaw składnik Tekst Wyrównanie akapitu na Lewo — do dołu.

12. Przeciągnij skrypt GooglePlacesDetailInfo z folderu Assets/FoodyGo/Scripts/UI w oknie Project na PlacesMarker w oknie Hierarchy.

13. Wybierz obiekt PlacesMarker w oknie Hierarchia. Następnie przeciągnij następujące komponenty do każdego miejsca w komponencie Informacje szczegółowe Miejsc Google w oknie Inspektora w następujący sposób:

Photo Panel: Obiekt fotograficzny z PlacesMarker | zdjęcie

Header: Obiekt tekstowy nagłówka z PlacesMarker | Płótno | nagłówki

Ocena: Ocena obiektu tekstowego z PlacesMarker | Płótno | ocena

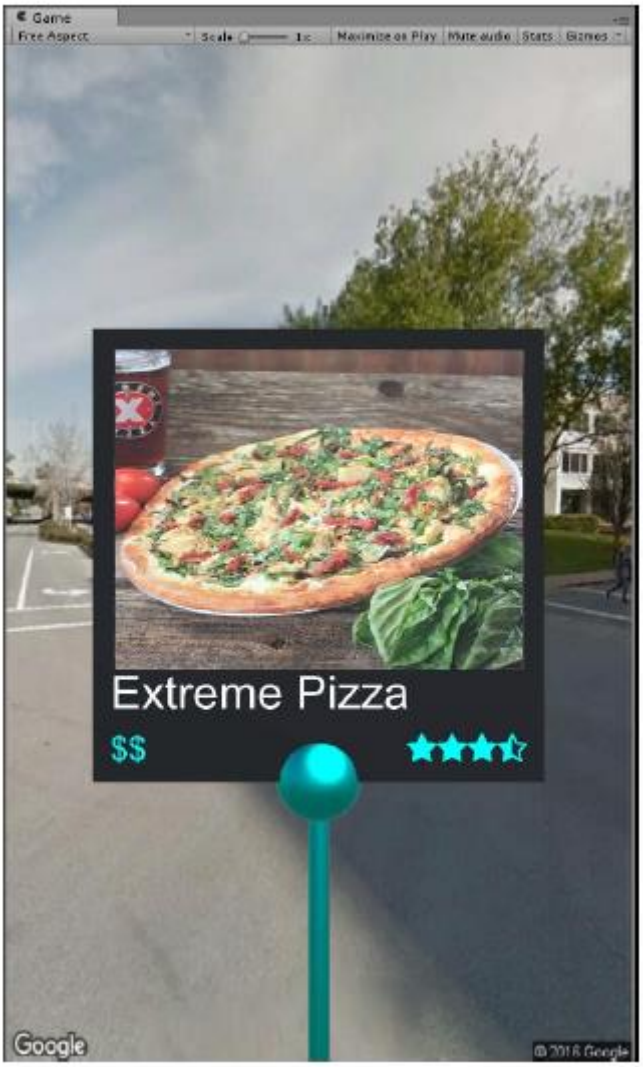
Cena: obiekt tekstowy ceny z PlacesMarker | Płótno | Cena £

14. Ustaw inne właściwości komponentu na wartości, jak pokazano na poniższym zrzucie ekranu:

Wskazówka: wartość identyfikatora miejsca użyta w tym przykładzie to (ChIJ7cdd3ed5j4AR7NfUycQnKvg). Możesz również użyć innych wartości identyfikatora miejsca, wykonując ręczne wyszukiwanie w pobliżu Google Places API.

15. Na koniec zapisz prefabrykat PlacesMarker, klikając przycisk Zastosuj obok opcji Prefabrykatów w oknie Inspektora.

16. Naciśnij Play, aby uruchomić scenę. Scena jest nieco sfabrykowana, ale pokaże Ci dodaną przez nas funkcjonalność. Poczekaj chwilę i obserwuj, jak zdjęcia na znaku zmieniają się w pokaz slajdów. Poniżej znajduje się przykładowy zrzut ekranu ze sceny uruchomionej w oknie gry:



Jak widać, nasza scena ładnie się rozwija. Przyjrzyjmy się szybko kilku głównym metodom w skrypcie GooglePlacesDetailInfo. Zaczniemy od metody LoadPlacesDetail:

```
IEnumerator LoadPlacesDetail()

{

var queryString = string.Format("placeid=

{0}&key={1}"

, placeId, PlacesAPIKey);

var req = new WWW(GOOGLE_PLACES_DETAIL_URL +

"?" + queryString);

//yield until the service responds

yield return req;

var json = req.text;
```

```
ParseSearchResult(json);

}
```

Do tej pory kod w ramach tych współprogramów powinien być starym kapeluszem. Ten kod jest bardzo podobny do tego, którego użyliśmy do wyszukiwania radaru Miejsc Google w poprzednim rozdziale. Jak widać w pierwszym wierszu, gdzie tworzona jest zmienna QueryString, parametry adresu URL to placeid i key, gdzie placeid to id lokalizacji, a key to oczywiście Twój klucz Google Places API. Po tym, jak skrypt zażąda szczegółowych informacji o miejscu, ustawia tekst nagłówka i ocenę. Jeśli miejsce ma jakieś zdjęcia, uruchamia nowe zapytanie względem punktu końcowego Google Places API Photos, jak pokazano w metodzie LoadPhotoTexture:

```
private IEnumerator LoadPhotoTexture(Photo photo)

{

var queryString =

string.Format("photoreference={0}&key=

{1}&maxwidth=800"

, photo.photo_reference,

PlacesAPIKey);

var url = GOOGLE_DETAIL_PHOTO_URL + "?" +

queryString;

var req = new WWW(url);

//yield until the service responds

yield return req;

//first destroy the old texture first

Destroy(photoRenderer.material.mainTexture);

//when the image returns set it as the tile

texture

photoRenderer.material.mainTexture =

req.texture;

}
```

Ten kod jest bardzo zbliżony do ładowania tekstur Street View, które zrobiliśmy wcześniej w tym rozdziale. Jedyną główną różnicą są parametry, które przekazujemy. Ten punkt końcowy wymaga tylko trzech parametrów: fotoreferencji (zasadniczo identyfikatora), klucza (ponownie tego klucza API) i maxwidth (lub maxheight). Wartość fotoreferencji jest pobierana z głównego zapytania, które zostało wykonane wcześniej. Ostatnią metodą, której się przyjrzymy, jest współprogram uruchamiający pokaz slajdów, zwany SlideShow:

```
private IEnumerator SlideShow(Result result)
```

```
{
while(doSlideShow && idx < result.photos.Count
- 1)
{
yield return new
WaitForSeconds(showSlideTimeSeconds);
idx++;
StartCoroutine(LoadPhotoTexture(result.photos[idx]
));
}
}
```

Parametr Result przekazany do tej metody jest wynikiem obiektu, który opisuje miejsce. Wewnątrz tej metody znajduje się ciągła pętla while, która sprawdza, czy doSlideShow jest prawdziwe i czy wynik zawiera jakieś zdjęcia do pokazania. idx jest indeksem do aktualnie wyświetlanego zdjęcia, zaczynając od 0. Jak widać, zmienna wynikowa ma właściwość Array o nazwie photos, która przechowuje wszystkie wartości fotoreferencji zdjęć zrobionych dla tego miejsca. W pętli while porównujemy bieżący indeks (idx) z liczbą zdjęć.

Jeśli jest więcej zdjęć, wykonywana jest pętla. Pierwsza instrukcja pętli powoduje, że podprogram działa przez określony czas zdefiniowany przez showSlideTimeSeconds. Następnie zwiększa idx, a następnie ładuje kolejne zdjęcie. Gdy nasze tło i znak miejsca działają, możemy teraz dodać interakcję ze sprzedażą potworów w następnej sekcji.

Dodanie interakcji interfejsu użytkownika do sprzedaży

Jak opisano na początku, gracze sprzedają swoje potwory, udając się do miejsca, a następnie inicjując sprzedaż, klikając przycisk. Miejsce lub restauracja następnie zbada potwory gracza i zaoferuje cenę za to, co uważa za najlepsze. Gracz może następnie zaakceptować ofertę, klikając przycisk Tak lub odrzucić, klikając przycisk Nie. Nawet przy tym bardzo prostym przepływie pracy nadal musimy połączyć wiele elementów, z których większość będzie interfejsami użytkownika. Podzielimy budowanie interfejsu użytkownika na sekcje, aby łatwiej było go śledzić. Pierwszą częścią, nad którą będziemy pracować, jest dodanie płótna interfejsu użytkownika i przycisków podstawowych, wykonując następujące czynności:

1. Z menu wybierz GameObject | interfejs użytkownika | płótno. Zmień nazwę nowej kanwy UI_Places. Jeśli liczyłeś, to jest nasze trzecie płótno na scenie, z każdym renderowaniem do innej przestrzeni (przestrzeń ekranu-nakładka, przestrzeń ekranu-kamera i przestrzeń świata).

2. Wybierz nowe płótno iw oknie Inspektora zmień właściwości komponentu Canvas Scaler w następujący sposób:

* Skaler płótna: tryb skalowania interfejsu użytkownika = skalowanie z rozmiarem ekranu

* Skaler płótna: Rozdzielczość referencyjna: X=500, Y=900

3. Kliknij prawym przyciskiem myszy (naciśnij klawisz Ctrl i kliknij komputer Mac) obiekt UI_Places iz menu kontekstowego wybierz UI | przycisk. Zmień nazwę nowego przycisku SellButton.

4. Rozwiń przycisk SellButton w oknie Hierarchia i wybierz obiekt Tekst. Wpisz usuń, aby usunąć obiekt Tekst z przycisku.

5. Wybierz przycisk SellButton w oknie Hierarchia i wpisz Ctrl + D (polecenie + D na komputerze Mac), aby zduplikować przycisk. Zmień nazwę nowego przycisku ExitButton.

6. Dla każdego z nowych przycisków zmień właściwości komponentu zgodnie z poniższą tabelą:

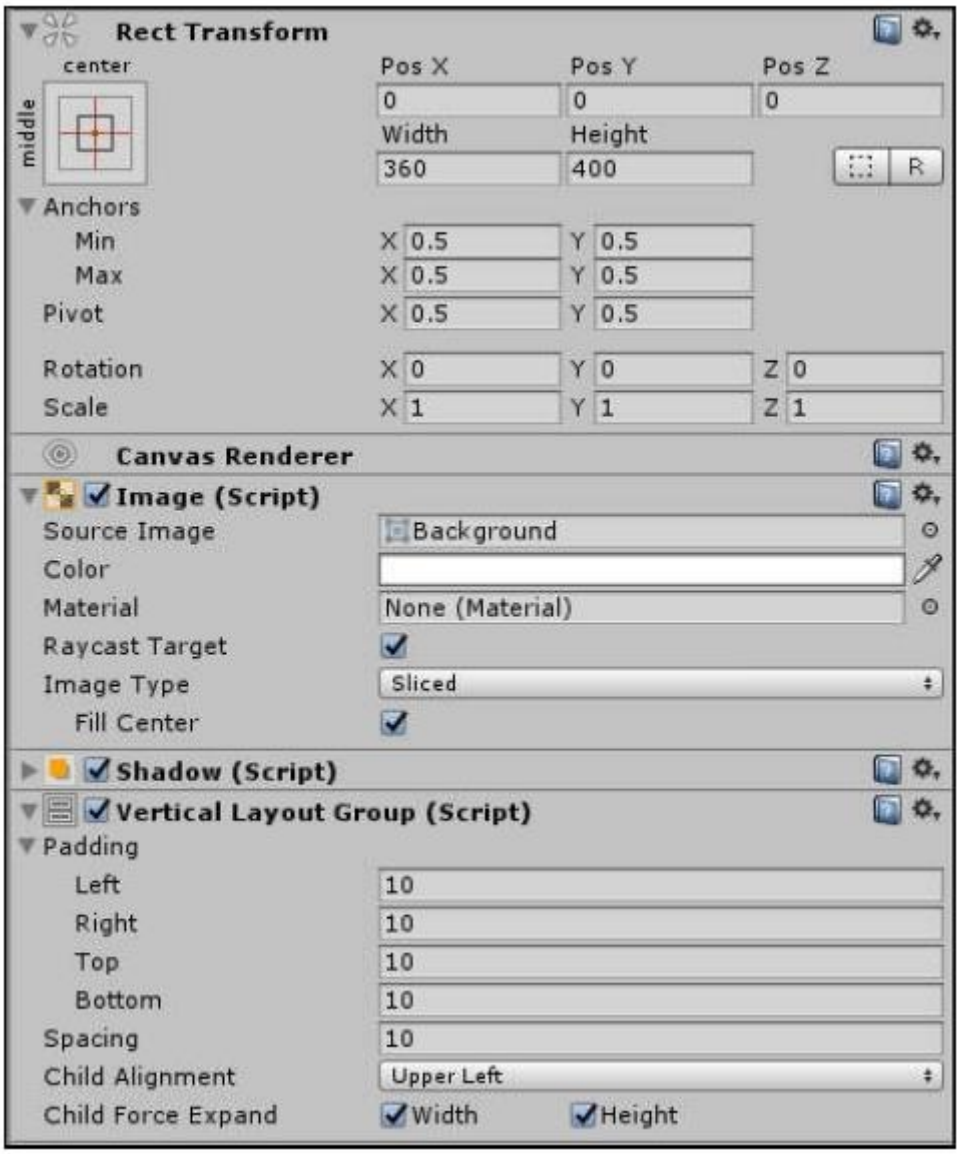
Button	Property	Value
SellButton	Rect Transform - Anchors	Top-Center pivot and position
	Rect Transform – Pos Y	-130
	Rect Transform – Width, Height	100
	Image – Source Image	button_set03_a
ExitButton	Rect Transform - Anchors	Bottom-Center pivot and position
	Rect Transform – Pos Y	10
	Rect Transform – Width, Height	75
	Image – Source Image	Button_set11_b

To kończy dodawanie przycisków do interfejsu. Nie martw się na razie okablowaniem przycisków. Zamiast tego będziemy pracować nad uzupełnieniem pozostałych elementów interfejsu potrzebnych w scenie. Następnie zbudujemy OfferDialog:

1. Kliknij prawym przyciskiem myszy (naciśnij klawisz Ctrl i kliknij, na komputerze Mac) na kanwie UI_Places w oknie Hierarchia iz menu kontekstowego wybierz UI | płyta. Zmień nazwę panelu OfferDialog. Będzie to obiekt nadrzędny dla wszystkich naszych komponentów okna dialogowego.

2. Wybierz panel OfferDialog w oknie Hierarchia, a następnie z menu wybierz Komponent | Układ | Grupa układu pionowego. Następnie z menu wybierz Komponent | interfejs użytkownika | Efekty | cień.

3. W oknie Inspektora ustaw właściwości komponentu Przekształcenie prostokąta, Obraz i Pionowa grupa układu, jak pokazano na następnym zrzucie ekranu:



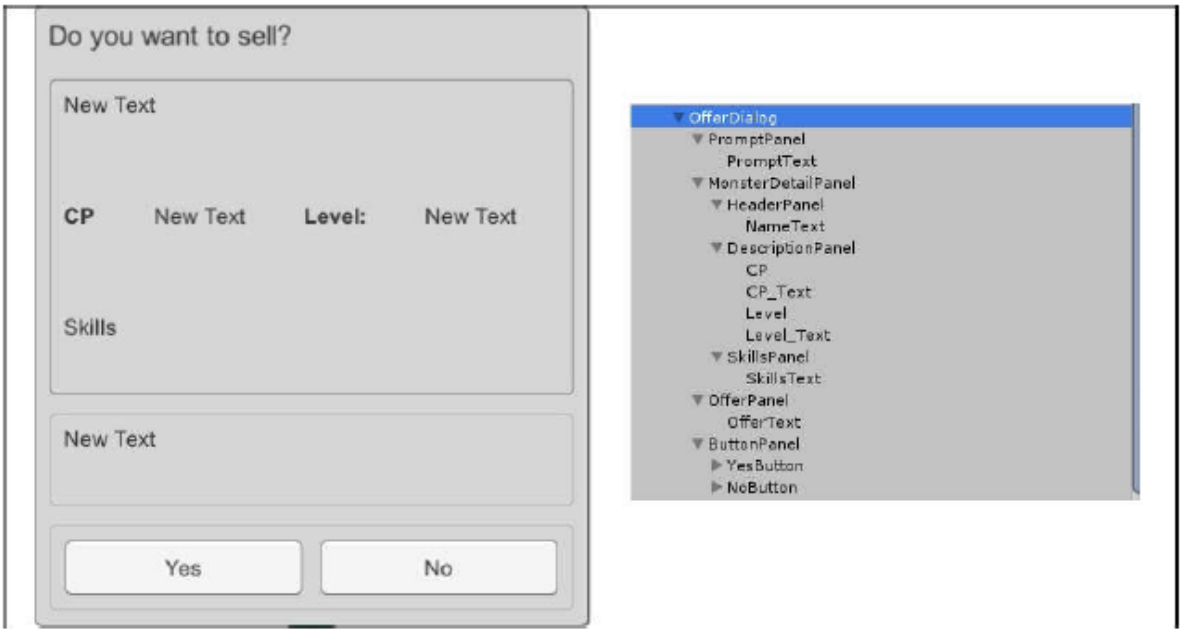
4. Kliknij prawym przyciskiem myszy (naciśnij klawisz Ctrl i kliknij komputer Mac) w oknie OfferDialog w oknie hierarchii iz menu kontekstowego wybierz opcję UI | płyta. Zmień nazwę panelu PromptPanel.

5. Wybierz PromptPanel w oknie Hierarchia i naciśnij Ctrl + D (polecenie + D na Macu), aby zduplikować obiekt. Zrób to jeszcze dwa razy, aby w sumie pojawiły się trzy nowe panele. Zmień nazwy nowych paneli w następujący sposób: MonsterDetailPanel, OfferPanel i ButtonPanel.

6. Zwróć uwagę, jak panele wewnątrz grupy układów dopasowują się, aby równomiernie dopasować przestrzeń. Teraz, gdy masz już pomysł, jak dodać układy i zbudować hierarchię interfejsu użytkownika, kontynuuj budowanie okna dialogowego, postępując zgodnie z tabelą:

Panel	UI Hierarchy			Component	Property	Value
PromptPanel				Image : remove		
	PromptText [Text]			Text	Text	Do you want to sell?
				Text	Font Size	18
MonsterDetailPanel				Vertical Layout Group	Spacing	10
	HeaderPanel [Panel]			Horizontal Layout Group	Left, Right, Top, Bottom, Spacing	10
		NameText [Text]				
	DescriptionPanel [Panel]			Horizontal Layout Group	Left, Right, Top, Bottom, Spacing	10
		CP [Text]		Text	Text	CP:
				Text	Font Style	Bold
		CPText [Text]				
		Level [Text]		Text	Text	Level:
				Text	Font Style	Bold
		LevelText [Text]				
	SkillsPanel [Panel]			Horizontal Layout Group	Left, Right, Top, Bottom, Spacing	10
		SkillsText [Text]		Text	Text	Skills
OfferPanel				Vertical Layout Group	Left, Right, Top, Bottom, Spacing	10
	OfferText [Text]					
ButtonPanel				Horizontal Layout Group		
	YesButton [Button]					
		Text		Text	Text	Yes
	NoButton [Button]					
		Text		Text	Text	No

7. Sprawdź, czy ostateczna konstrukcja pasuje do okna dialogowego i hierarchii:



Po zakończeniu OfferDialog możemy szybko dodać RefuseDialog, postępując zgodnie z następującymi wskazówkami:

1. Wybierz panel OfferDialog w oknie Hierarchia i naciśnij Ctrl + D (polecenie + D na Macu), aby zduplikować okno dialogowe. Zmień nazwę panelu RefuseDialog.

2. W oknie Inspektora zmień Rect Transform - Height = 150.

3. Rozwiń RefuseDialog i usuń MonsterDetailPanel i OfferPanel.

4. Rozwiń PromptPanel, a następnie wybierz PromptText w oknie Hierarchia. W oknie Inspektora zmień Tekst - Tekst na To miejsce nie będzie składać oferty na żaden z Twoich aktualnych akcji!.

5. Rozwiń ButtonPanel w oknie Hierarchii i usuń jeden z przycisków, nie ma znaczenia który. Zmień nazwę pozostałego przycisku na przycisk OK. Rozwiń przycisk i wybierz podrzędny obiekt Tekst. W oknie inspektora zmień tekst - tekst na OK.

6. Na koniec przeciągnij obiekt UI_Places na obiekt PlacesScene w oknie Hierarchy, aby uczynić go dzieckiem korzenia sceny.

Po zbudowaniu wszystkich elementów interfejsu możemy teraz połączyć wszystkie skrypty, których potrzebujemy, aby ukończyć interakcję ze sprzedażą scen. Wykonaj następujące instrukcje, aby dodać różne skrypty składowe:

1. Przeciągnij skrypt PlacesSceneUIController z folderu Assets/FoodyGo/Scripts/Controllers w oknie Project na obiekt UI_Places w oknie Hierarchy.

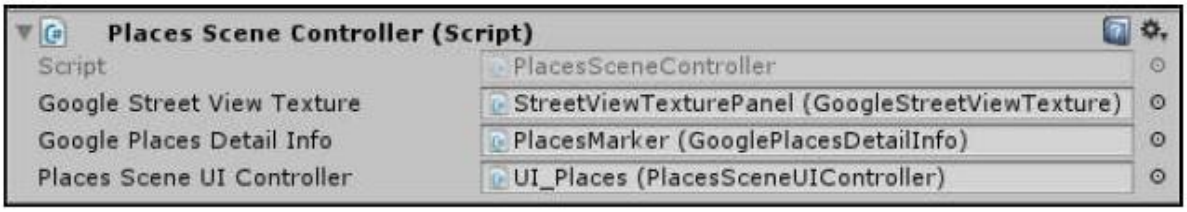
2. Wybierz obiekt UI_Places i przeciągnij SellButton, OfferDialog i RefuseDialog do każdego z odpowiednich slotów w komponencie Places Scene UI Controller w oknie Inspektora.

3. Przeciągnij skrypt MonsterOfferPresenter z folderu Assets/FoodyGo/Scripts/UI w oknie Project na obiekt OfferDialog w oknie Hierarchy.

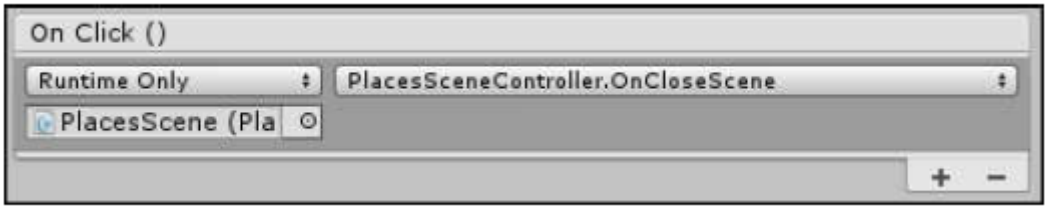
4. Wybierz obiekt OfferDialog i przeciągnij obiekty NameText, CPText, LevelText, SkillsText i OfferText do każdego z odpowiednich miejsc w komponencie Monster Offer Presenter w oknie Inspektora.

5. Przeciągnij skrypt PlacesSceneController z folderu Assets/FoodyGo/Scripts/Controllers w oknie projektu na obiekt główny PlacesScene.

6. Wybierz obiekt PlacesScene i przeciągnij obiekty StreeViewTexturePanel, PlacesMarker i UI_Places do odpowiedniego miejsca w komponencie Places Scene Controller w oknie Inspektora, jak pokazano na poniższym zrzucie ekranu:



7. Wybierz przycisk ExitButton w oknie Hierarchy, a następnie w oknie Inspektora dodaj nowy program obsługi zdarzeń, naciskając plus. Następnie przeciągnij obiekt PlacesScene na miejsce na obiekt i otwórz menu rozwijane funkcji. Wybierz funkcję PlacesSceneController.OnCloseScene w następujący sposób:



8. Powtórz proces dla SellButton, ale tym razem wybierz funkcję PlacesSceneController.OnClickSell z menu rozwijanego funkcji.

9. Wybierz YesButton i dodaj nową obsługę zdarzeń. Przeciągnij obiekt UI_Places do miejsca obiektu i wybierz PlacesSceneUIController.AcceptOffer() dla funkcji.

10. Wybierz NoButton i dodaj nową obsługę zdarzeń. Przeciągnij obiekt UI_Places do miejsca obiektu i wybierz PlacesSceneUIController.RefuseOffer() dla funkcji.

11. Wybierz przycisk OK i dodaj nową obsługę zdarzeń. Przeciągnij Obiekt UI_Places w gnieździe obiektu i wybierz PlacesSceneUIController.OK() dla funkcji.

To kończy okablowanie sceny. Teraz wystarczy dodać kilka usług, aby uruchomić scenę Miejsca samodzielnie. Wykonaj następujące instrukcje, aby dodać usługi:

1. Utwórz w scenie nowy pusty obiekt gry o nazwie Usługi. Przeciągnij go na obiekt PlacesScene, aby uczynić go dzieckiem głównego obiektu sceny.

2. Dodaj dwa nowe obiekty podrzędne do obiektów usług o nazwie Inventory i MonsterExchange.

3. Przeciągnij skrypt InventoryService na obiekt Inventory, a skrypt MonsterExchangeService na obiekt MonsterExchange z folderu Assets/FoodyGo/Scripts/Services w oknie projektu.

4. Naciśnij Play, aby uruchomić scenę i spróbuj kliknąć przycisk Sprzedaj i różne inne przyciski. Jak zauważysz, nie możesz jeszcze sprzedać potwora, ale interfejs użytkownika powinien działać zgodnie z oczekiwaniami, z wyjątkiem przycisku Exit, oczywiście.

To uzupełnia wszystkie wymagane elementy interfejsu użytkownika dla sceny, co, jak widzieliście, jest dość dużo jak na prostą scenę. Interfejs użytkownika jest nadal bardzo prosty i deweloper będzie mógł go później rozszerzyć i stworzyć własny. Zapraszam do odkrywania i zmieniania okien dialogowych interfejsu użytkownika tak, jak lubisz. Możesz również zauważyć, że nie przekazujemy opinii graczowi, jeśli sprzedaje on przedmioty. Deweloper musi wprowadzić nowe okno dialogowe lub dźwięk według własnego uznania. W następnej sekcji dowiemy się, jak potwory są oceniane, przyglądając się bliżej usłudze MonsterExchangeService.

Mechanika sprzedaży w grze

Spośród wielu rzeczy, z których twórcy gier mogą czerpać przyjemność, z pewnością tworzenie mechaniki i algorytmów gry musi znajdować się na szczycie listy. Nie ma lepszego sposobu na odróżnienie twojej gry lub wirtualnego świata niż od mechaniki gry, która ją napędza. Jako twórca i projektant gier masz wiele możliwości definiowania mechaniki gry, czy to tworzenia potężnej broni lub stworzeń, fizyki świata, czy nawet sprzedawania gotujących się potworów. Ogólnie rzecz biorąc, najlepiej jest zachować prostotę i trzymaliśmy się tej filozofii podczas budowania mechaniki sprzedaży gry. Zamiast patrzeć na rzeczywisty kod źródłowy usługi MonsterExchangeService, która jest usługą oceniającą ekwipunek potworów gracza. Przyjrzymy się bardziej podstawowemu pseudokodowi, który będzie łatwiejszy do naśladowania:

```
MonsterOffer PriceMonsters(PlaceResult result)
{
//get the players monsters from Inventory
var monsters = InventoryService.ReadMonsters();
//loop through all the reviews of the place
//and append into a single text string
var reviews = string.empty;
foreach(var r in result.reviews)
{
reviews += r.text + " ";
}
//loop through the monsters and place a value
to each
List<MonsterOffer> offers = new
List<MonsterOffer>();
//calculate the places budget to spend on
cooking monsters
var budget = result.rating * result.price *
result.types * 100;
foreach(var m in monsters)
{
var value = 0;
//add a bonus value based on every skill
word
//that matches a word in the review
//ex. skill=pizza, reviews may mention the
word pizza 5 times = 500
value += CountMatches(m.skills,
reviews)*100;
//add the cooking power (CP)
value += m.power * m.level * 100;
```

```
if(budget > value)

//add a new offer for the monster

offers.Add(new MonsterOffer{ Monster =

m, Value = value });

}

offers.sort(); //sort the offers from lowest

to highest

offers.reverse(); //reverse the order so

highest on top

return offer[0]; //return the top offer

}
```

Algorytm, którego tu używamy, jest dość prosty z jednym zwrotem, w którym faktycznie staramy się dać premię potworom, które mogą mieć umiejętności wymienione w recenzjach miejsc. Jeśli otworzysz skrypt MonsterExchangeService, zauważysz, że kod jest zgodny z tym samym wzorcem. Oczywiście możesz dowolnie modyfikować tę mechanikę gry. e mechanik w dowolny sposób. Po utworzeniu wartości liczbowej dla gotującego potwora nie przejmujemy się konwersją tej liczby na warunki gry, dopóki oferta nie zostanie przedstawiona graczowi. Kiedy oferta jest przedstawiana graczowi, liczba jest następnie konwertowana na doświadczenie i przedmioty przez usługę MonsterExchangeService przy użyciu metody ConvertOffer. Nie będziemy tutaj wchodzić w szczegóły metody ConvertOffer, ponieważ kod po prostu rozbija wartość o granice, podobnie jak może działać algorytm wymiany monet. Poniżej znajduje się lista pokazująca granice zdefiniowane w skrypcie:

1000: kula nitro

250: Kula suchego lodu

100: Kula Lodu

Na przykład, jeśli oferta została wyceniona na 2600, to ConvertOffer

podzieli tę wartość na:

2: Kulki nitro (2000)

2: Kulki suchego lodu (500)

1: Kula lodowa (100)

260 Doświadczenie: doświadczenie jest obliczane na 1/10 wartości

Ponownie możesz otworzyć skrypt MonsterExchangeService i dodać nowe elementy lub zmienić wartości. Teraz, gdy rozumiemy, jak ceni się mechanikę gry, jak ceni się gotującego potwora, musimy przejść do faktycznej wymiany, która jest omówiona w dalszej części.

Aktualizacja bazy danych

Jak zapewne zauważyłeś podczas testowania interfejsu użytkownika sceny Miejsca, nie możesz jeszcze sprzedać potwora. To dlatego, że musimy dodać kilka nowych tabel do naszej bazy danych (Inwentarz). Na szczęście, jak być może pamiętasz, skoro nasza baza danych to ORM (object relacyjne mapowanie), praca nad tworzeniem nowych tabel jest dość bezbolesna. Otwórz więc skrypt InventoryService znajdujący się w folderze Assets/FoodyGo/Scripts/Services w wybranym edytorze. Przewiń w dół do metody CreateDB i poszukaj nowej sekcji następującej po kodzie po podświetlonym wierszu, aby utworzyć nowe tabele w bazie danych:

```
Debug.Log("DatabaseVersion table created.");

//look for this line to start

//create the InventoryItem table

var iinfo =

_connection.GetTableInfo("InventoryItem");

if (iinfo.Count > 0)

_connection.DropTable<InventoryItem>();

_connection.CreateTable<InventoryItem>();

//create the Player table

var pinfo = _connection.GetTableInfo("Player");

if (pinfo.Count > 0)

_connection.DropTable<Player>();

_connection.CreateTable<Player>();
```

Ten kod po prostu dodaje nowe tabele, InventoryItem i Player do bazy danych. Następnie przewiń w dół nieco dalej tą samą metodą i spójrz na kod wymagany do utworzenia nowego gracza startowego za podświetloną linią:

```
Debug.Log("Database version updated to " +

DatabaseVersion);

//start here

_connection.Insert(new Player

{

Experience = 0,

Level =1

});
```

Po dodaniu nowych tabel w CreateDB i wypełnieniu odtwarzaczem początkowym, zaktualizowano również metodę UpgradeDB. Stary UpgadeDB został zastąpiony następującym kodem:

```
private void UpgradeDB()
```

```
{
var monsters = _connection.Table<Monster>
().ToList();
var player = _connection.Table<Player>
().ToList();
var items = _connection.Table<InventoryItem>
().ToList();
CreateDB();
Debug.Log("Replacing data.");
_connection.InsertAll(monsters);
_connection.InsertAll(items);
_connection.UpdateAll(player);
Debug.Log("Upgrade successful!");
}
```

Jeśli zostanie wyzwolone uaktualnienie bazy danych, zostanie uruchomiona metoda UpgradeDB. Najpierw zapisze wartości w bieżących tabelach w zmiennych tymczasowych, następnie uruchomi się metodę CreateDB w celu utworzenia nowych tabel bazy danych, a na końcu dane zostaną wstawione lub zaktualizowane z powrotem w bazie danych. Możliwość aktualizacji bazy danych w ten sposób pozwala nam dodawać nowe właściwości do obiektu. Jednak nigdy nie możemy usunąć ani zmienić nazwy istniejących właściwości, ponieważ prawdopodobnie zniszczyłoby to stare obiekty. Teraz, gdy stoły są gotowe, możemy dodać metody CRUD dla naszych nowych stołów, InventoryItem i Player. Przewiń do dołu pliku InventoryService i spójrz na następujące nowe metody:

```
//CRUD for InventoryItem
public InventoryItem
CreateInventoryItem(InventoryItem ii)
{
var id = _connection.Insert(ii);
ii.Id = id;
return ii;
}
public InventoryItem ReadInventoryItem(int id)
{
return _connection.Table<InventoryItem>()
.Where(w => w.Id ==
```

```
id).FirstOrDefault();
}
public IEnumerable<InventoryItem>
ReadInventoryItems()
{
return _connection.Table<InventoryItem>();
}
public int UpdateInventoryItem(InventoryItem ii)
{
return _connection.Update(ii);
}
public int DeleteInventoryItem(InventoryItem ii)
{
return _connection.Delete(ii);
}
```

The code is almost the same as the Monster CRUD code we wrote in a previous chapter. Now, look at the CRUD code for the Player:

```
//CRUD for Player
public Player CreatePlayer(Player p)
{
var id = _connection.Insert(p);
p.Id = id;
return p;
}
public Player ReadPlayer(int id)
{
return _connection.Table<Player>()
.Where(w => w.Id ==
id).FirstOrDefault();
}
public IEnumerable<Player> ReadPlayers()
```

```
public IEnumerable<Player> ReadPlayers()

{

return _connection.Table<Player>();

}

public int UpdatePlayer(Player p)

{

return _connection.Update(p);

}

public int DeletePlayer(Player p)

{

return _connection.Delete(p);

}
```

Być może zastanawiasz się, dlaczego potrzebujemy wszystkich metod CRUD dla Gracza. Umieściliśmy je tam, aby nasza gra mogła być później używana do obsługi wielu graczy, pozwalając użytkownikowi wybrać różne postacie do gry lub nawet pozwalając wielu graczom grać razem. Następnie chcemy wrócić do sceny Miejsca i przeprowadzić aktualizację do naszej bazy danych, aby wszystkie nowe tabele były tam. Upewnij się, że zapisałeś wszystkie swoje skrypty i wróć do edytora Unity. Znajdź obiekt InventoryService i wybierz go w oknie Hierarchia. W oknie Inspektora zmień opcję Inventory Service — Database Version na 1.0.1 (przy założeniu, że aktualna wartość to 1.0.0). Uruchom scenę Miejsca w edytorze i sprawdź okno Konsola. Poszukaj komunikatu w dzienniku Uaktualnienie zakończone pomyślnie, aby upewnić się, że baza danych jest uaktualniana poprawnie.

Wskazówka: za każdym razem, gdy chcesz zaktualizować bazę danych w przyszłości, po prostu zwiększ wersję do wyższej wartości. Wersja 1.0.3 jest wyższa niż 1.0.1, a wersja 2.0.0 jest wyższa niż 1.0.22.

Po zaktualizowaniu kodu InventoryService i bazy danych musimy teraz dokonać wymiany gracza, w którym odda on potwora na miejsce i otrzyma w zamian swoje doświadczenie i przedmioty. Otwórz skrypt PlacesSceneUIController w swoim edytorze i przewiń w dół do metody AcceptOffer:

```
public void AcceptOffer()

{

OfferDialog.SetActive(false);

SellButton.SetActive(true);

var offer = CurrentOffer;

InventoryService.Instance.DeleteMonster(offer.Mons

ter);

var player =

InventoryService.Instance.ReadPlayer(1);
```

```
player.Experience += offer.Experience;

InventoryService.Instance.UpdatePlayer(player);

foreach(var i in offer.Items)

{

InventoryService.Instance.CreateInventoryItem(i);

}

}
```

Dwie górne linie tej metody po prostu włączają / wyłączają okno dialogowe i przycisk. Następnie używamy tych metod CRUD, aby usunąć potwora z ekwipunku. Następnie aktualizujemy wrażenia gracza, a następnie dodajemy przedmioty do nowej tabeli InventoryItem. Nie będziemy używać tych nowych przedmiotów w tym rozdziale, ale mamy teraz miejsce do przechowywania nowego ekwipunku. Wróć do Unity i uruchom scenę Places. Teraz spróbuj sprzedać potwory. Powinieneś zauważyć, że na miejsce wyprzedawane są potwory. Sprawdź to, wracając i stale sprzedając więcej potworów. Za każdym razem, gdy sprzedajesz nowego potwora, prawdopodobnie będzie miał nowe imię i inne umiejętności.

Uwaga: gdy uruchamiasz grę z edytora, usługa ekwipunku upewni się, że w ekwipunku zawsze znajduje się losowy potwór. Odbywa się to w celach debugowania. Jeśli jednak napotkasz, że miejsce nie jest w stanie kupić twojego ostatniego potwora, może to opóźnić twoje testy.

Nasza scena Miejsc jest na razie gotowa. Możesz dowolnie rozszerzać elementy w scenie, jak tylko chcesz. W następnej sekcji dodamy scenę Miejsca z powrotem do gry i pozwolimy graczowi odwiedzić miejsce.

Łączenie elementów

Po ukończeniu sceny Places musimy teraz ponownie zintegrować scenę z Gamescene. Wykonaj poniższe instrukcje, aby skonfigurować scenę Miejsca, tak aby mogła zostać załadowana przez GameManagera:

1. Z folderu Assets w oknie Project przeciągnij scenę Game do okna Hierarchy.

2. Wybierz usługę Inventory w scenie Places i zanotuj, co ustawiłeś dla usługi Inventory Service - Database Version w oknie Inspektora. Następnie wybierz usługę Inventory w scenie gry i zmień usługę Inventory — wersja bazy danych, aby była zgodna.

Uwaga: jeśli nie zrobimy tego kroku, baza danych może zostać uszkodzona i po prostu wyczyścić się za pomocą zaktualizowanych obiektów. W tym momencie nie jest to tak krytyczne, ale byłoby bardzo niewygodne dla wszystkich użytkowników, gdybyś tworzył aktualizację gry.

3. Ponieważ potrzebujemy tylko jednej usługi Inventory, wybierz obiekt usługi Inventory w scenie Places i naciśnij Delete, aby go usunąć. Wybierz i usuń również obiekt System zdarzeń w scenie Miejsca.

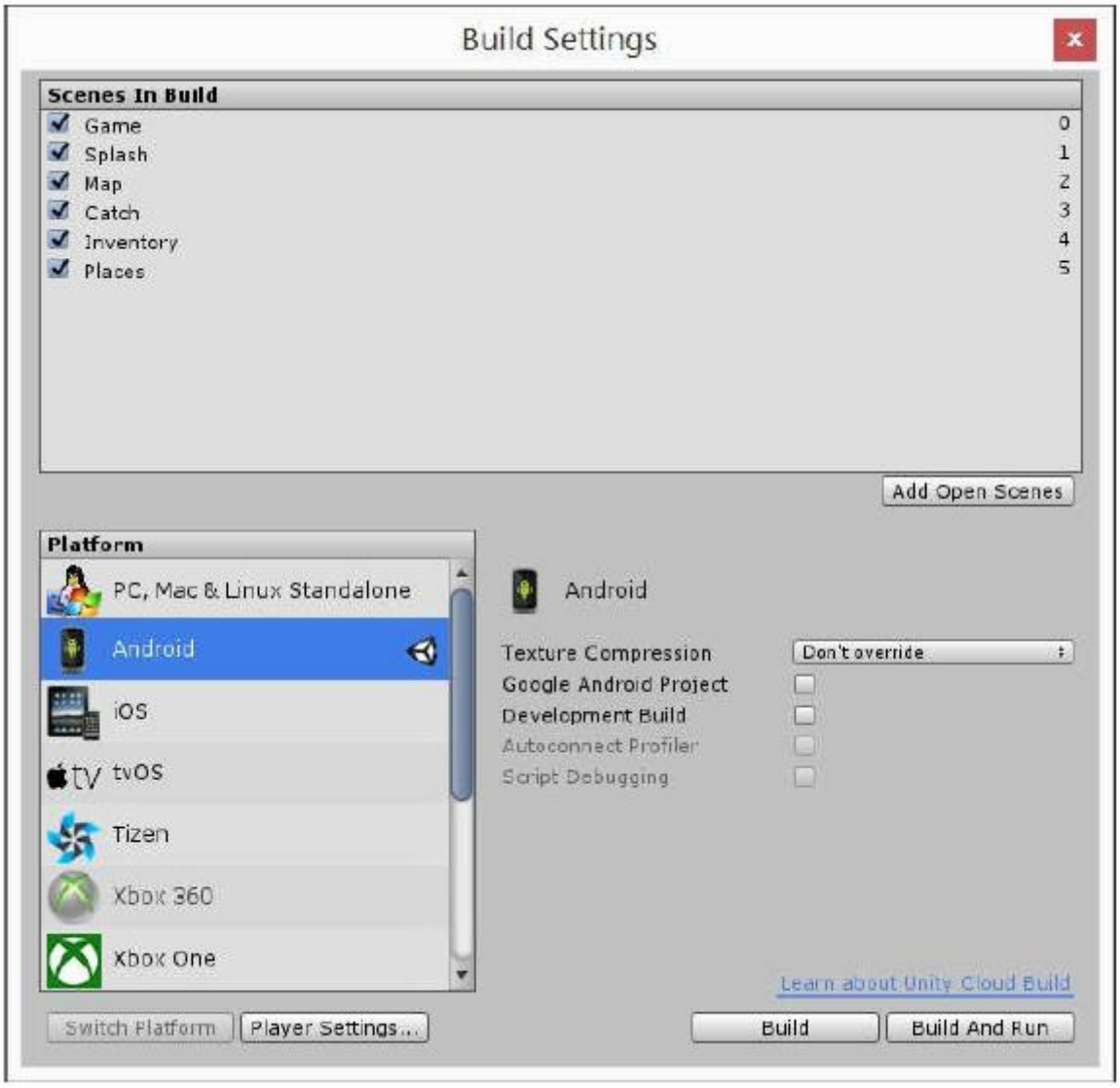
4. Wybierz obiekt PlacesScene w oknie Hierarchia i dezaktywuj go, odznaczając pole wyboru obok nazwy obiektu w oknie Inspektora. Wyłączymy obiekt główny, ponieważ będziemy ładować scenę na starcie.

5. Kliknij prawym przyciskiem myszy (naciśnij klawisz Ctrl i kliknij na komputerze Mac) na scenie Miejsca i wybierz Usuń scenę z menu kontekstowego. Upewnij się, że zapisałeś zmiany, które właśnie wprowadziliśmy w scenie.

6. Wybierz obiekt _GameManager w oknie Hierarchia. Ustaw

Menedżer gier — umieszcza właściwość Nazwa sceny w polu Miejsca w oknie Inspektora.

7. Z menu wybierz Plik | Ustawienia budowania, aby otworzyć okno dialogowe Ustawienia budowania. Po prostu potwierdź, że wszystkie sceny zostały dodane i są sprawdzane pod kątem uwzględnienia, jak pokazano na poniższym zrzucie ekranu:



Jeśli uruchomisz grę w edytorze, niestety nadal nie będziesz mógł kliknąć miejsca, aby otworzyć scenę Miejsca. Dzieje się tak, ponieważ musimy pozwolić naszemu PlaceMarkerowi działać jako zderzacz, coś, co pominęliśmy w poprzednim rozdziale. Wykonaj następujące instrukcje, aby dodać zderzacz i inne ustawienia do PlaceMarker:

1. Przeciągnij prefabrykat PlaceMarker, który zbudowaliśmy w poprzednim rozdziale, z folderu Assets/FoodyGo/Prefabs do okna Hierarchia. Zauważ, że zmieniamy PlaceMarker, a nie prefabrykaty PlacesMarker, które zmodyfikowaliśmy w tym rozdziale.

2. Wybierz PlaceMarker w oknie Hierarchy, a następnie z menu wybierz Component | Fizyka | Zderzacz pudełek.

3. Kliknij dwukrotnie PlaceMarker, aby ustawić ostrość obiektu w oknie Sceny. Zobaczysz zieloną ramkę u podstawy obiektu; to jest zderzacz.

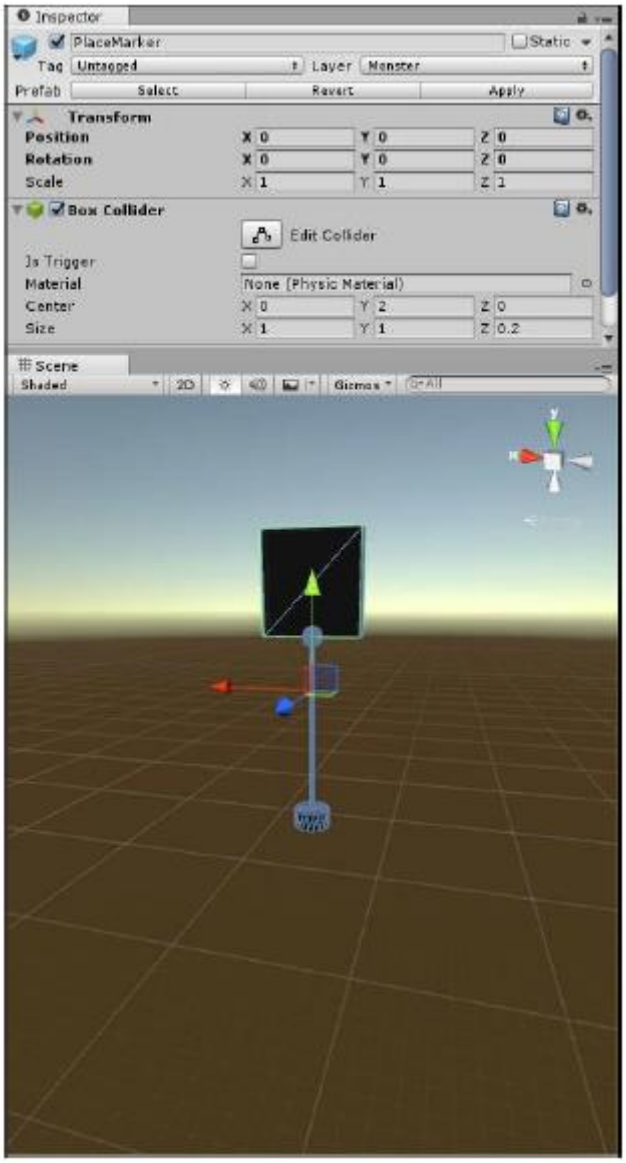
4. W oknie Inspektora zmień Box Collider - Center i Size na wartości pokazane na poniższej liście:

Box Collider: Środek: X = 0, Y = 2, Z = 0

Box Collider: Rozmiar: X = 1, Y = 1, Z = 0,2

5. Teraz zmień warstwę obiektów na Monster. Gdy pojawi się monit, nie martw się o zmianę dzieci, ponieważ zderzacz znajduje się tylko na górnym obiekcie. Prawdopodobnie powinniśmy utworzyć nową warstwę dla naszego wykrywania kolizji, ale na razie będzie to działać.

6. Sprawdź poniższy zrzut ekranu, aby upewnić się, że właściwości obiektu w oknie Inspektora są zgodne, a okno Scena wygląda podobnie do następującego:



7. Kliknij przycisk Zastosuj obok opcji Prefabrykatów w oknie Inspektora, aby zapisać zmiany. Następnie usuń znacznik miejsca z okna hierarchii.

8. Naciśnij Play, aby uruchomić grę w edytorze. Upewnij się, że usługa GPS działa w trybie symulacji. Kliknij dowolny znak PlaceMarker na scenie, a powinieneś zostać przeniesiony do sceny Places.

9. Oczywiście, jak zawsze, upewnij się, że tworzysz i wdrażasz swoją grę na urządzeniu mobilnym w celu przetestowania, upewniając się, że wszystko przechodzi i wygląda zgodnie z oczekiwaniami.

Konfiguracja była stosunkowo prosta, więc spójrzmy szybko na kod, który łączy kolizję. Otwórz skrypt GameManager w wybranym edytorze i przewiń w dół do metody HandleHitGameObject. Na szczycie metody znajduje się nasz oryginalny kod, który obsługuje, gdy gracz kliknie potwora, a następnie zostanie przeniesiony do sceny Catch. Tuż poniżej tego zajmujemy się, gdy nasz PlaceMarker zostanie trafiony (wybrany):

```
if (go.GetComponent<PlacesController>() != null)

{

print("Places hit, need to open places

scene ");

//check if the scene has already been run

if (PlacesScene == null)

{

SceneManager.LoadSceneAsync(PlacesSceneName,

LoadSceneMode.Additive);

}

else

{

//the scene has run before, reactivate

it

var psc =

PlacesScene.RootGameObject.GetComponent<PlacesScen

eController>();

if (psc != null)

{

var pc =

go.GetComponent<PlacesController>();

psc.ResetScene(pc.placeId,

pc.location);
```

```
}

PlacesScene.RootGameObject.SetActive(true);

}

MapScene.RootGameObject.SetActive(false);

}
```

Ten blok kodu jest bardzo podobny do tego, którego używaliśmy, gdy obiekt potwora (MonsterController) został trafiony lub wybrany. Jeśli podążasz za komentarzami, wszystko jest dość oczywiste. Jak widać, postępując zgodnie z tym samym wzorem, moglibyśmy łatwo dodać inne obiekty, z którymi możesz chcieć szybko wejść w interakcję. Możesz również zauważyć, że nie ma znaczenia, jak daleko miejsca są od gracza, nadal mogą z nimi wchodzić w interakcje. W tym momencie jest to celowe, ale może to być coś bardzo łatwego do naprawienia, po prostu ustawiając długość promienia podczas testowania kolizji. Jeśli przewiniesz kod w górę do metody RegisterHitGameObject, zobaczysz, że promień jest ustawiony na nieskończoność:

```
public bool RegisterHitGameObject(PointerEventData

data)

{

int mask = BuildLayerMask();

Ray ray =

Camera.main.ScreenPointToRay(data.position);

RaycastHit hitInfo;

if (Physics.Raycast(ray, out hitInfo,

Mathf.Infinity, mask))

{

print("Object hit " +

hitInfo.collider.gameObject.name);

var go = hitInfo.collider.gameObject;

HandleHitGameObject(go);

return true;

}

return false;

}
```

Możesz zmienić tę wartość na wartość zakodowaną na stałe, na przykład 100, lub wrócić do naszej dyskusji na temat dokładności GPS i ustawić tę wartość na podstawie aktualnej dokładności urządzenia. W obu przypadkach jest to kolejna mechanika gry, którą można łatwo ustawić, ale może mieć duże różnice w sposobie rozgrywania twojej gry.

Podsumowanie

To była intensywna część poświęcona budowaniu sceny Miejsca jako miejsca interakcji gracza ze światem gry zmieszanym z częścią świata wirtualnego. Zaczęliśmy od zbudowania fundamentów sceny. Następnie przeszliśmy do włączenia zdjęć Google Street View jako tła sceny, aby zapewnić głębsze poczucie interakcji z grą w świecie rzeczywistym. Stamtąd ulepszyliśmy znak znacznika pokazem slajdów dostarczonym przez Zdjęcia z Miejsc Google i innymi elementami, takimi jak nazwa, ocena i cena. Teraz, gdy zebraliśmy szczegółowe informacje o miejscu, dodaliśmy przyciski interakcji i dialogów. Dało nam to duże doświadczenie w pracy z elementami uGUI Unity, w tym używanie układów do układania elementów. Po umieszczeniu elementów interaktywnych przeszliśmy do zrozumienia, jak działała mechanika sprzedaży gry. To skłoniło nas do ulepszenia bazy danych ekwipunku o nowe tabele i metody CRUD dla przedmiotów ekwipunku i gracza. Następnie połączyliśmy wszystkie elementy razem i dodaliśmy do gry scenę Miejsca. Następnie, aby wszystko działało razem, dodaliśmy do znacznika zderzacz pudełek, aby gracz mógł wchodzić z nim w interakcję na scenie mapy. Zbliżamy się do zakończenia gry, a następna część zajmie się dodaniem obsługi sieci dla wielu graczy. Ponadto spędzimy dodatkowy czas na ulepszaniu gry za pomocą efektów cząsteczkowych i wizualnych.