Łapanie zdobyczy w AR

Jeśli pamiętasz z naszej fabuły Foody GO, gracze musieli polować na zbiegłe eksperymentalne potwory kulinarne i je złapać. Od końca ostatniego rozdziału gracz może teraz śledzić i widzieć na mapie otaczające ich potwory. Teraz potrzebujemy, aby gracz mógł wchodzić w interakcje i próbować złapać potwory, które widzi. Aby nasza gra była głęboka, chcemy, aby gracz łapał potwory w widoku alternatywnej rzeczywistości. Dlatego chcemy również włączyć kamerę urządzenia, aby zapewnić tło dla naszej aktywności połowowej. Dodanie komponentu AR pozwoli teraz zaklasyfikować naszą grę do gatunku gier przygodowych w świecie rzeczywistym lub AR opartego na lokalizacji. W tej części dodamy do naszej gry szereg nowych funkcji, które będą wymagały od nas dotknięcia kilku nowych koncepcji. W przeciwieństwie do poprzednich części, nie będziemy zbytnio zagłębiać się w teorię, ponieważ większość tych nowych koncepcji ma fundamentalne znaczenie dla tworzenia gier i Unity. Zamiast tego przyjrzymy się, jak te rzeczy działają w Unity i zapewnimy kilka referencji dla tych, którzy chcą dowiedzieć się więcej o konkretnej koncepcji. Oto lista przedmiotów, które będziemy omawiać:

* Zarządzanie scenami

* Przedstawiamy Menedżera Gier

* Ładowanie sceny

* Aktualizacja wprowadzania dotykowego

* Zderzacze i fizyka ciała sztywnego

* Budowanie sceny przechwytywania AR

* Używanie aparatu jako tła sceny

* Dodanie piłki do łapania

* Rzucanie piłki

* Sprawdzanie kolizji

* Efekty cząsteczkowe dla opinii

* Łapanie potwora

Podobnie jak w poprzednich rczęściach, jeśli masz otwarte Unity z poprzedniej części, z załadowanym projektem gry, przejdź do następnej sekcji. W przeciwnym razie otwórz Unity i załaduj projekt gry FoodyGO lub otwórz folder href="https://remigiuszkurczab.pl/C4E.zip">Chapter_4_End z pobranego kodu źródłowego. Następnie upewnij się, że scena Mapa jest załadowana.

Wskazówka : Gdy otworzysz jeden z zapisanych plików projektu, prawdopodobnie będziesz musiał również załadować scenę początkową. Unity często tworzy nową domyślną scenę, zamiast próbować odgadnąć scenę, która powinna zostać załadowana.

Zarządzanie sceną

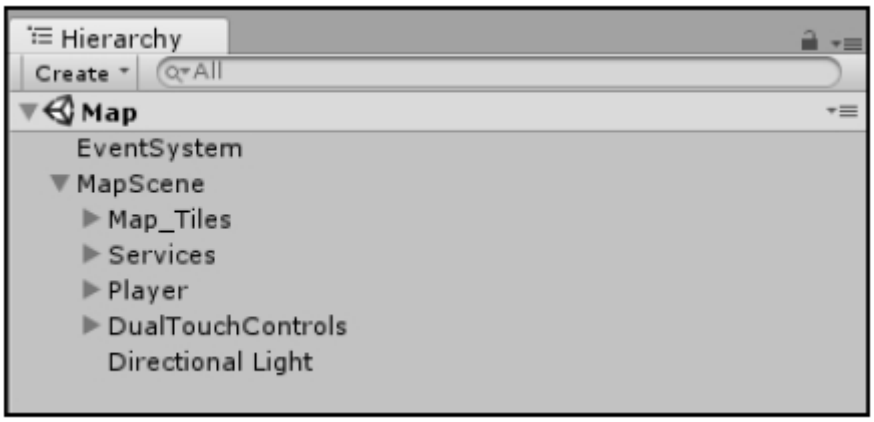
Zanim przejdziemy do dodawania nowych funkcji do gry, powinniśmy cofnąć się nieco i zająć się tym, jak będziemy przechodzić od sceny do sceny. Obecnie mamy dwie sceny opracowane dla naszej gry: sceny Splash i Map. Ponadto w tej części dodamy jeszcze dwie sceny, Game i Catch. Jednak obecnie nie mamy możliwości zarządzania przejściami scen ani czasem życia obiektów w grze. Idealnie, chcesz jakiegoś głównego obiektu i/lub skryptu, który zrobi to wszystko za nas. Właśnie to zbudujemy i nazwiemy to Game Managerem. Utrzymaj podekscytowanie przez minutę. Aby bezproblemowo

załadować i przejście między scenami, chcemy trochę posprzątać nasze obecne sceny. Otwórz Unity i wykonaj następujące instrukcje, aby wyczyścić i zreorganizować obecne sceny gry:

1. Upewnij się, że scena Mapa jest załadowana w oknie Hierarchia. Utwórz nowy pusty obiekt gry, wybierając GameObject | Utwórz puste menu.

2. Zmień nazwę nowego obiektu MapScene i zresetuj transformację do zera.

3. Przeciągnij i upuść obiekt Player na obiekt Mapa w oknie Hierarchia. To sprawi, że gracz stanie się dzieckiem Mapy. Zrób to ponownie dla Map_Tiles, Services, DualTouchControls i Directional Light, jak pokazano na poniższym zrzucie ekranu:



4. Zapisz scenę, wybierając pozycję menu Plik | Zapisz scenę.

5. Zapisz scenę jako nową scenę o nazwie Gra, wybierając pozycję menu Plik | Zapisz scenę jako... iw oknie dialogowym Zapisz scenę wpisz nazwę Gra i kliknij przycisk Zapisz.

6. Upewnij się, że dokładnie wykonałeś następną sekwencję kroków, aby uniknąć frustracji.

7. W oknie Hierarchia usuń obiekt EventSystem, zaznaczając obiekt i naciskając klawisz Delete. Scena powinna teraz zawierać tylko MapScene i elementy podrzędne.

8. Z menu wybierz Plik | Zapisz jako... i nazwij mapę sceny, a następnie kliknij przycisk Zapisz. Zostaniesz poproszony o nadpisanie sceny; kliknij Tak, w następujący sposób:

9. Otwórz nową scenę Gry, wybierając folder Zasoby w oknie Projekt, a następnie dwukrotnie klikając scenę Gry.

10. Usuń obiekt MapScene, zaznaczając go i naciskając usuń. Po usunięciu obiektu MapScene zauważysz, że okno gry staje się czarne i wyświetla komunikat Brak renderowania kamer. Staraj się nie

rozpraszać, nie panikować ani nie wchodź w tryb naprawy i spróbuj dodać nową kamerę do sceny. Wszystko jest w porządku, więc po prostu kontynuuj.

11. Scena gry powinna teraz zawierać tylko obiekt EventSystem. Moglibyśmy faktycznie usunąć ten obiekt, ponieważ Unity automatycznie doda go z powrotem, jeśli do sceny zostanie dodany komponent interfejsu użytkownika. Aby jednak lepiej zarządzać naszymi scenami i obiektami, zachowamy obiekt EventSystem bez zmian.

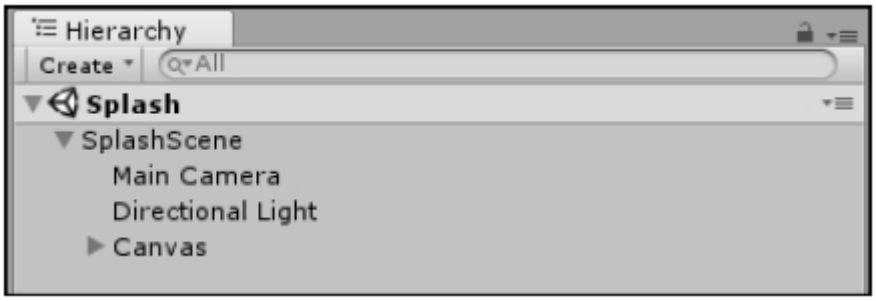
12. Zapisz scenę, wybierając pozycję menu Plik | Zapisz scenę.

13. Otwórz scenę Splash, wybierając folder Zasoby w oknie Projekt, a następnie klikając dwukrotnie scenę Splash, aby ją otworzyć.

14. Wybierz pozycję menu GameObject | Utwórz puste. Zmień nazwę nowego obiektu na SplashScene i zresetuj transformację do zera w oknie Inspektora.

15. Przeciągnij i upuść kamerę główną, światło kierunkowe i płótno do obiektu SplashScene. Wszystkie powinny być teraz dziećmi obiektu SplashScene.

16. Usuń obiekt EventSystem, zaznaczając go i naciskając Delete. Scena powitalna w oknie hierarchii nie powinna wyglądać jak następujący fragment ekranu:

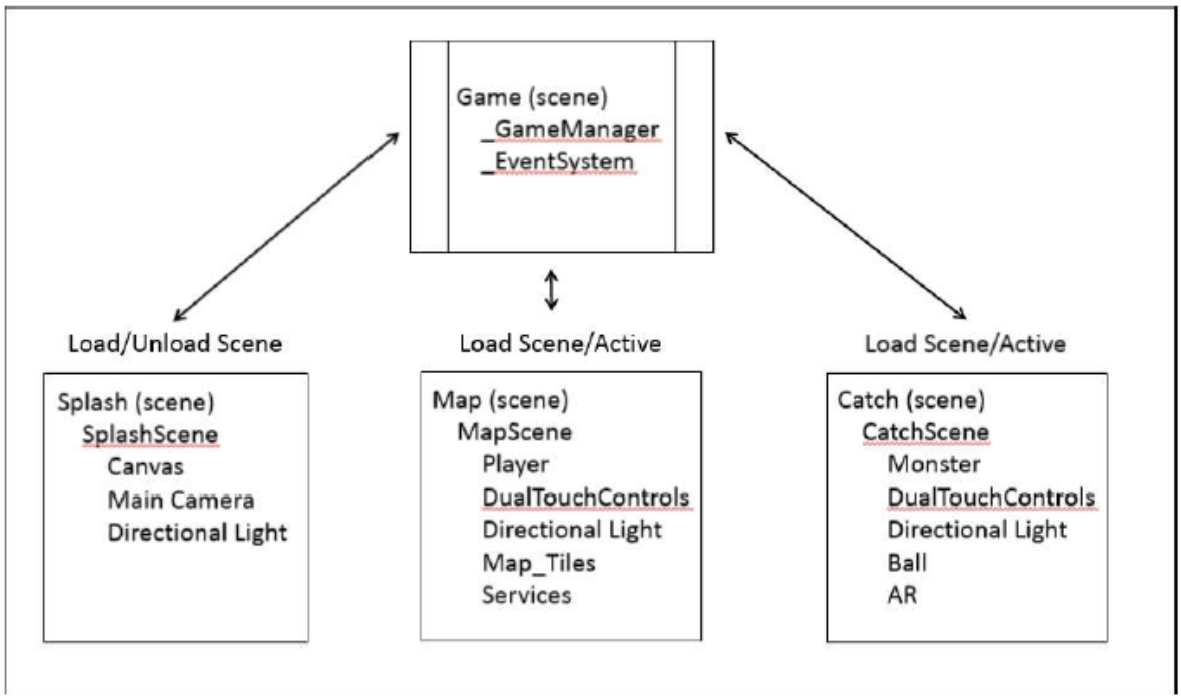


17. Zapisz scenę, wybierając pozycję menu Plik | Zapisz scenę.

Powinieneś teraz móc otwierać sceny Splash i Map i odtwarzać je bez żadnych problemów. Wypróbuj to teraz jako test, aby upewnić się, że wszystko przeniosłeś i wszystko zostało zapisane w porządku. Podczas uruchamiania scen zauważysz, że obiekt EventSystem jest dynamicznie dodawany do sceny Map; to jest w porządku i oczekiwane.

Przedstawiamy Menedżera Gry

Game Manager (GM) będzie naszym nadzorcą i kontrolerem wszystkich głównych działań w grze. GM będzie zarządzał ładowaniem lub rozładowywaniem sceny i przejściami, a także wieloma innymi wyższymi funkcjami, które omówimy później. GM będzie rezydował w scenie Gry, która będzie pierwszą załadowaną sceną. Następnie, w razie potrzeby, GM będzie zarządzać inną aktywnością pomiędzy scenami, jak pokazano na poniższym diagramie:



Otrzymamy skonfigurowany i uruchomiony obiekt i skrypt GameManagera, co wyjaśni dalej. Niestety, będzie to bardzo pracowita część i nie będziemy mieli dużo czasu na przeglądanie całego kodu. Zdecydowanie zaleca się poświęcenie trochę czasu na samodzielne przejrzenie skryptów. Teraz postępuj zgodnie z kolejnymi instrukcjami, aby zaimportować i skonfigurować skrypt GameManager:

1. Otwórz scenę Gra, wybierając folder Zasoby w oknie Projekt i klikając dwukrotnie scenę Gra.

2. Po załadowaniu sceny utwórz nowy pusty obiekt gry, wybierając pozycję menu GameObject | Utwórz puste.

3. Zmień nazwę nowego obiektu _GameManager i zresetuj transformację do zera w oknie Inspektora; zwróć uwagę na użycie podkreślenia. Użyjemy podkreślenia do oznaczenia obiektu, którego nie należy niszczyć ani dezaktywować.

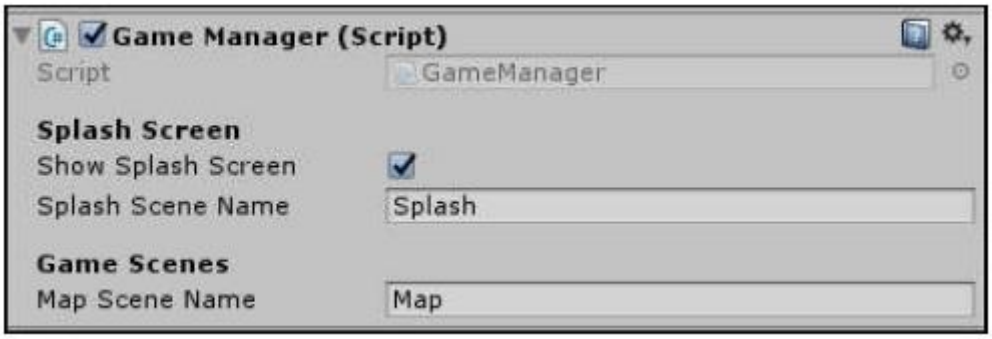
4. Wybierz i zmień nazwę obiektu EventSystem na _EventSystem w oknie Inspektora z tego samego powodu.

5. Wybierz pozycję menu Zasoby | Importuj pakiet | Pakiet niestandardowy.... Następnie w oknie dialogowym Importuj pakiet przejdź do pobranego folderu kodu źródłowego Chapter_5_Assets i wybierz opcję Chapter5_import1.unitypackage. Następnie kliknij Otwórz, aby rozpocząć import.

6. Kiedy otworzy się okno dialogowe Import Unity Package, po prostu upewnij się, że wszystkie zasoby są wybrane i kliknij Importuj.

7. Przejdź do folderu Assets/FoodyGo/Scripts/Managers. Przeciągnij i upuść skrypt GameManager na obiekt _GameManager w oknie Hierarchy.

8. Wybierz obiekt _GameManager i ustaw właściwości komponentu skryptu Game Manager w następujący sposób:



9. Wybierz pozycję menu Plik | Ustawienia budowania.... Musimy dodać sceny Gry, Splash i Mapy do ustawień budowania i ustawić je w kolejności pokazanej w następującym oknie dialogowym Ustawienia budowania:

10. Możesz dodać sceny, przeciągając scenę z folderu Zasoby w oknie Projekt i upuszczając ją na obszar sceny. Sceny można zmienić w obszarze, wybierając i przeciągając scenę w górę lub w dół, a następnie upuszczając ją w razie potrzeby. Pierwsza scena na liście będzie tą, która zostanie załadowana jako pierwsza. Upewnij się, że konfiguracja sceny jest zgodna z obrazem okna dialogowego.

11. Po dodaniu i uporządkowaniu scen uruchom grę w edytorze, naciskając Play. Zauważ, że scena Gry szybko się ładuje, a następnie wczytywana jest scena Splash, a kilka sekund później scena Mapa. Powinieneś również zauważyć, że scena Map została załadowana za sceną Splash, w oknie Scena.

12. Upewnij się, że tworzysz i wdrażasz również grę na swoim urządzeniu mobilnym. Uruchom grę na swoim urządzeniu, aby mieć pewność, że działa tak, jak w poprzedniej części.

Ładowanie sceny

Jak wspomniano, nie będziemy mieli czasu na przyjrzenie się zmianom w kodzie na poziomie szczegółowości, który zrobiliśmy wcześniej. Nie chcemy jednak całkowicie pominąć ważnych wzorców kodowania, co oznacza, że nadal będziemy przeglądać ważne sekcje lub wiersze kodu. W tym pierwszym imporcie skryptu GameManager ważną sekcją kodu, którą chcemy przejrzeć, jest sposób ładowania sceny. Wykonaj następujące instrukcje, aby przejrzeć kod:

1. Otwórz skrypt GameManagera z okna Project, odnajdując plik w folderze Assets/FoodyGo/Scripts/Managers i klikając go dwukrotnie. Spowoduje to otwarcie wybranego edytora lub domyślnego MonoDevelop.

2. Przewiń w dół do metody DisplaySplashScene. Ta sekcja kodu pokazuje ważne wzorce, które chcemy wyróżnić. Poniżej pokazano metodę dla osób, które nie mogą otworzyć edytora skryptów:

```
//display the Splash scene and then

load the game start scene

IEnumerator DisplaySplashScene()

{

SceneManager.LoadSceneAsync(MapSceneName,

LoadSceneMode.Additive);

//set a fixed amount of time to wait

before unloading splash

scene

//we could also check if the GPS

service was started and running

//or any other requirement

yield return new WaitForSeconds(5);

SceneManager.UnloadScene(SplashScene);

}
```

3. Wewnątrz współprogramu zwróć uwagę na użycie nowej klasy SceneManager. SceneManager to klasa pomocnicza, która umożliwia dynamiczne ładowanie i rozładowywanie scen w czasie wykonywania. W pierwszym wierszu SceneManager asynchronicznie ładuje scenę w trybie addytywnym, a nie w trybie zastępowania. Addytywne ładowanie scen umożliwia jednoczesne wczytanie wielu scen, a następnie wyładowanie sceny, gdy nie jest ona już potrzebna, jak pokazano w ostatnim wierszu metody.

4. Samodzielnie przejrzyj resztę kodu skryptu GameManager. Upewnij się, że śledzisz sposób ładowania i usuwania scen. Możesz zauważyć wiele innych rzeczy, które dzieje się w GameManager. Nie martw się, wkrótce omówimy kilka z tych kawałków.

Aktualizacja wprowadzania dotykowego

Teraz, gdy możemy zarządzać naszymi przejściami scen za pomocą GameManagera, musimy teraz ustawić katalizator, który wywoła zmianę sceny. W przypadku sceny Catch będzie to oznaczać, że gracz stuknie potwora, którego chce złapać, co oznacza, że będziemy musieli wyizolować wejście dotykowe, gdy jest on na potworze. Jeśli pamiętasz, nasze obecne sterowanie dotykowe działa na całym ekranie, a wejście kieruje tylko kamerą. To, co musimy zrobić, to dostosować skrypt wprowadzania dotykowego, aby obsłużyć dotyk potwora. Na szczęście, dla zwięzłości, te zmiany skryptu zostały dodane w ramach naszego ostatniego importu. Wykonaj następujące instrukcje, aby skonfigurować nowy skrypt i przejrzeć te zmiany:

1. Otwórz edytor Unity i załaduj scenę mapy.

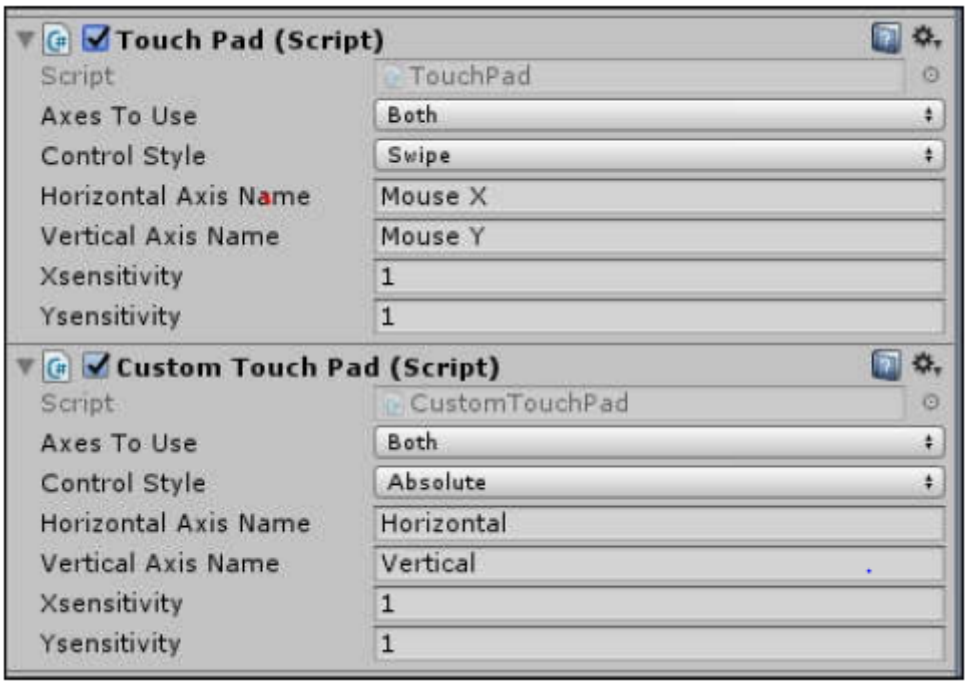
2. W oknie Hierarchia rozwiń obiekt MapScene i wybierz obiekt DualTouchControls. Zmień nazwę tego obiektu UI_Input w oknie Inspektora; będzie to bardziej opisowa nazwa funkcji tego obiektu.

Wskazówka : Dobrą praktyką programistyczną jest zmiana nazw obiektów gry, klas, skryptów lub innych komponentów w celu dopasowania do funkcji. Dobra nazwa może być równa kilku linijkom dokumentacji na temat funkcji, podczas gdy zła nazwa może powodować frustrację i koszmary związane z aktualizacją lub konserwacją.

3. Rozwiń obiekt UI_Input i wybierz TurnAndLook Touchpad.

4. Z folderu Assets/FoodyGo/Scripts/TouchInput przeciągnij skrypt CustomTouchPad i upuść go na obiekcie TurnAndLookTouchpad.

5. Spowoduje to dodanie składnika skryptu niestandardowego panelu dotykowego tuż pod składnikiem panelu dotykowego w oknie inspektora w następujący sposób:



6. Skopiuj wszystkie ustawienia z komponentu Touch Pad do niestandardowego komponentu Touch Pad. Upewnij się, że oba ustawienia są identyczne.

7. Usuń element płytki dotykowej, klikając ikonę koła zębatego i wybierając Usuń element z menu kontekstowego.

8. Skrypt CustomTouchPad jest praktycznie identyczny ze skryptem TouchPad, różni się tylko jednym wierszem. Być może myślisz, dlaczego po prostu nie zmodyfikowaliśmy oryginalnego skryptu? Powodem, dla którego stworzyliśmy nową kopię skryptu, a następnie ją zmodyfikowaliśmy, jest uczynienie go naszym własnym. W ten sposób, jeśli zasób danych wejściowych międzyplatformowych będzie musiał zostać w przyszłości uaktualniony, nasze niestandardowe zmiany w skrypcie nie zostaną nadpisane.

9. Kliknij ikonę koła zębatego w komponencie Niestandardowy panel dotykowy i wybierz Edytuj skrypt z menu kontekstowego. Spowoduje to otwarcie skryptu w wybranym przez Ciebie edytorze.

10. Przewiń w dół lub wyszukaj metodę OnPointerDown; poniżej znajduje się fragment metody i jedna linia zmienionego kodu:

```
public void

OnPointerDown(PointerEventData data)

{

if

(GameManager.Instance.RegisterHitGameObject(data)) return;
```

11. Metoda OnPointerDown jest wywoływana, gdy użytkownik po raz pierwszy dotknie ekranu i będzie początkiem przesunięcia. Nie chcemy jednak śledzić przesunięcia, jeśli dotyk był ważny obiekt. To właśnie robi nowa linia. Linia kodu wywołuje GameManager.Instance.RegisterHitGameObject z pozycją dotyku. Jeśli ważny obiekt zostanie dotknięty, zwracana jest wartość true i powracamy, nie pozwalając na rozpoczęcie akcji machnięcia. Jeśli zamiast tego nic nie zostanie trafione, przesunięcie będzie działać normalnie.

Wskazówka: GameManager.Instance oznacza wywołanie w celu uzyskania pojedynczej instancji GameManagera. Singleton to dobrze znany wzorzec używany do utrzymywania globalnej instancji pojedynczego obiektu. Singleton jest idealny dla naszego GameManagera, ponieważ będzie używany przez wiele klas do kontrolowania jednego stanu gry.

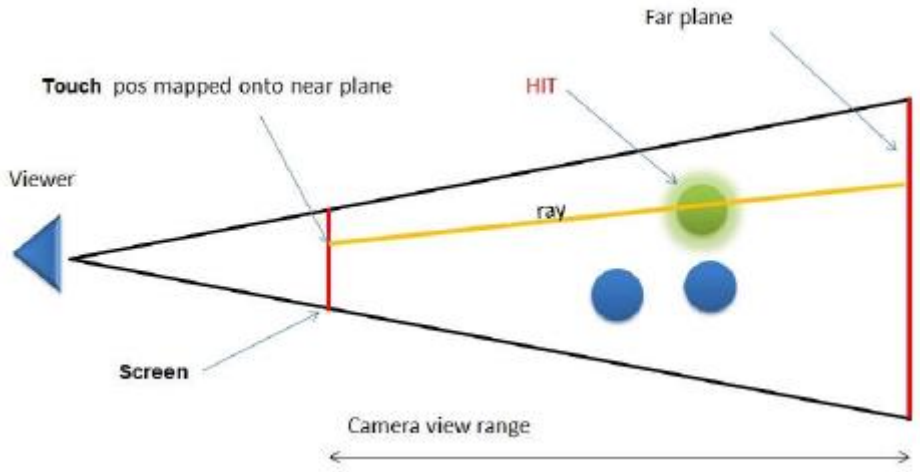
12. Teraz, gdy jesteś jeszcze w edytorze kodu, otwórz ponownie klasę GameManager.

13. Przewiń w dół do metody RegisterHitGameObject:

```
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;
}
```

14. Funkcją tej metody jest określenie, czy konkretne wejście dotykowe uderzyło w ważny obiekt na scenie. Czyni to poprzez zasadniczo rzucanie promienia z pozycji na ekranie do świata gry. Możesz myśleć o promieniu jako o wskaźniku świetlnym, a być może pomoże poniższy diagram:



15. Większość pracy jest wykonywana w ramach metody Physics.Raycast, która wykorzystuje rzucanie promienia przez interakcję dotykową, odniesienie do obiektu RaycastHit, odległość, na jaką promień powinien być testowany, a na koniec maskę warstwy do określenia czy i jak obiekt został trafiony. Dużo się tu dzieje, więc podzielmy te parametry dalej:

* Promień: jest to promień lub linia prosta, która jest rzucana i używana do testowania kolizji

* out RaycastHit: Zwraca informacje o kolizji

* Odległość: jest to maksymalny zasięg, w którym wyszukiwanie powinno być zrobione

* Maska: maska służy do określenia warstw, które należy przetestować pod kątem kolizji. Bardziej szczegółowo o zderzeniach i warstwach fizyki omówimy w następnej sekcji.

Wskazówka : W kodzie używamy Mathf.Infinity dla zakresu wyszukiwania. Dla liczby obiektów, które mamy obecnie w scenie, będzie to działać dobrze. W bardziej złożonych scenach wartość nieskończoności może być kosztowna, ponieważ możesz chcieć przetestować tylko wszystkie obiekty znajdujące się w odległości wizualnej.

16. Wróć do Unity i uruchom grę w edytorze, naciskając Play. Jak możesz zauważyć, nic nowego się nie dzieje. To dlatego, że brakuje nam jednego istotnego elementu. Physics.RayCast testuje kolizje z obiektami ze zderzaczami. Jak dotąd nasz obiekt-potwór nie ma zderzacza ani nie jest w żaden sposób skonfigurowany do korzystania z silnika fizyki, ale wkrótce to naprawimy.

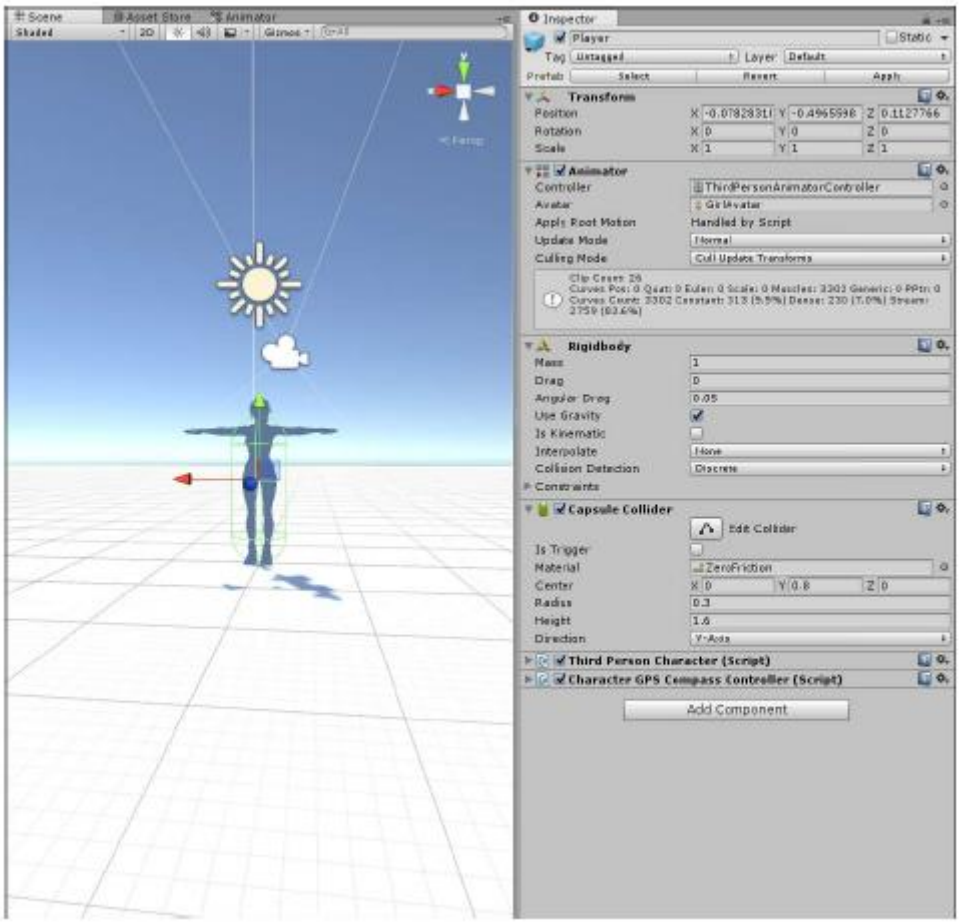
Zderzacze i fizyka ciała sztywnego

Do tej pory unikaliśmy dyskusji o fizyce, ale nasza gra korzysta z silnika fizyki Unity, odkąd dodaliśmy postać do sceny. Silnik fizyki Unity składa się z dwóch części: jednej dla 2D i bardziej złożonej 3D. Silnik fizyczny jest tym, co ożywia grę i sprawia, że środowisko gry jest bardziej naturalne. Deweloperzy mogą następnie wykorzystać ten silnik, aby szybko i łatwo dodawać nowe obiekty do świata, które automatycznie reagują w naturalny sposób. Ponieważ mamy już dobry działający przykład obiektu gry wykorzystującego silnik fizyki, przyjrzymy się temu:

1. W Unity upewnij się, że scena Map jest załadowana.

2. W oknie Hierarchia rozwiń obiekt MapScene. Następnie, wybierz obiekt Player. Kliknij dwukrotnie obiekt Player, aby wykadrować go w oknie Scena.

3. Spójrz na okno Sceny i na zieloną kapsułę owiniętą wokół twojej postaci iClone. W oknie Inspektora sprawdź komponenty Rigidbody i Capsule Collider; poniżej znajduje się zrzut ekranu obu okien:



4. Elementy ciała sztywnego i zderzacza są niezbędne do określenia fizyki obiektu; poniżej znajduje się krótkie podsumowanie tego, co robi każdy z komponentów i jak mogą być ze sobą powiązane:

* Rigidbody: Pomyśl o sztywnym ciele jako o czymś, co definiuje właściwości masy obiektu, na przykład, czy obiekt reaguje na grawitację, jaka jest jego masa, jak łatwo się obraca i tak dalej.

* Collider: Jest to ogólnie uproszczona geometria, która definiuje granice obiektu. Stosowana jest uproszczona geometria, taka jak pudełko, kula lub kapsuła, ponieważ wykrywanie kolizji można przeprowadzić bardzo szybko. Korzystanie z bardziej złożonej siatki, a nawet próba użycia rzeczywistej siatki postaci, zatrzymywałaby silnik fizyki za każdym razem, gdy testowano kolizję. W każdej klatce silnik fizyki testuje, aby sprawdzić, czy obiekty kolidują ze sobą. Jeśli obiekty zderzają się ze sobą, silnik fizyki użyje praw ruchu Newtona, aby określić efekt zderzenia. Wystarczy powiedzieć, że jeśli chcesz dowiedzieć się więcej o fizyce, istnieje wiele zasobów dostępnych za pośrednictwem Twojego przyjaciela Google. W bardziej zaawansowanych grach do owinięcia tułowia i kończyn można użyć kilku zderzaczy kapsułek. Tym samym pozwala na detekcję kolizji na poszczególnych częściach ciała. Na nasze potrzeby zderzacz kapsułek będzie odpowiadał naszym potrzebom.

Pomijając definicje fizyki, wróćmy do naszych potworów. Dodamy te elementy fizyki do prefabrykatu potworów, wykonując następujące instrukcje:

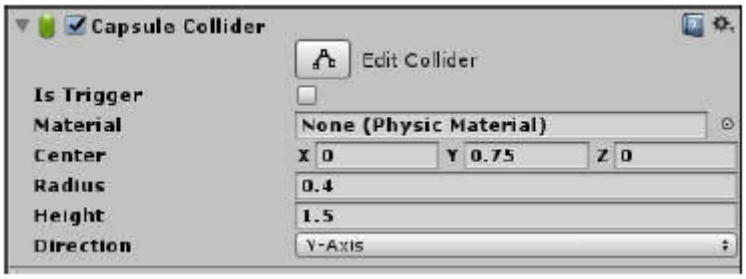
1. Otwórz folder Assets/FoodyGo/Prefabs w oknie projektu i przeciągnij i upuść prefabrykat Monster do okna Hierarchii.

2. Kliknij dwukrotnie prefabrykat potwora, aby ustawić go w oknach Sceny i Inspektora.

3. Dodaj zderzacz kapsułek, wybierając pozycję menu Komponent | Fizyka | Zderzacz kapsułek.

4. Dodaj korpus sztywny, wybierając pozycję menu Komponent | Fizyka | Sztywny korpus.

5. Jeśli przyjrzysz się teraz uważnie w oknie Sceny, zobaczysz, że zderzacz kapsuł nie owija potwora. Spróbuj samodzielnie dostosować właściwości składnika Capsule Collider lub użyj ustawień, jak pokazano w poniższym oknie dialogowym:



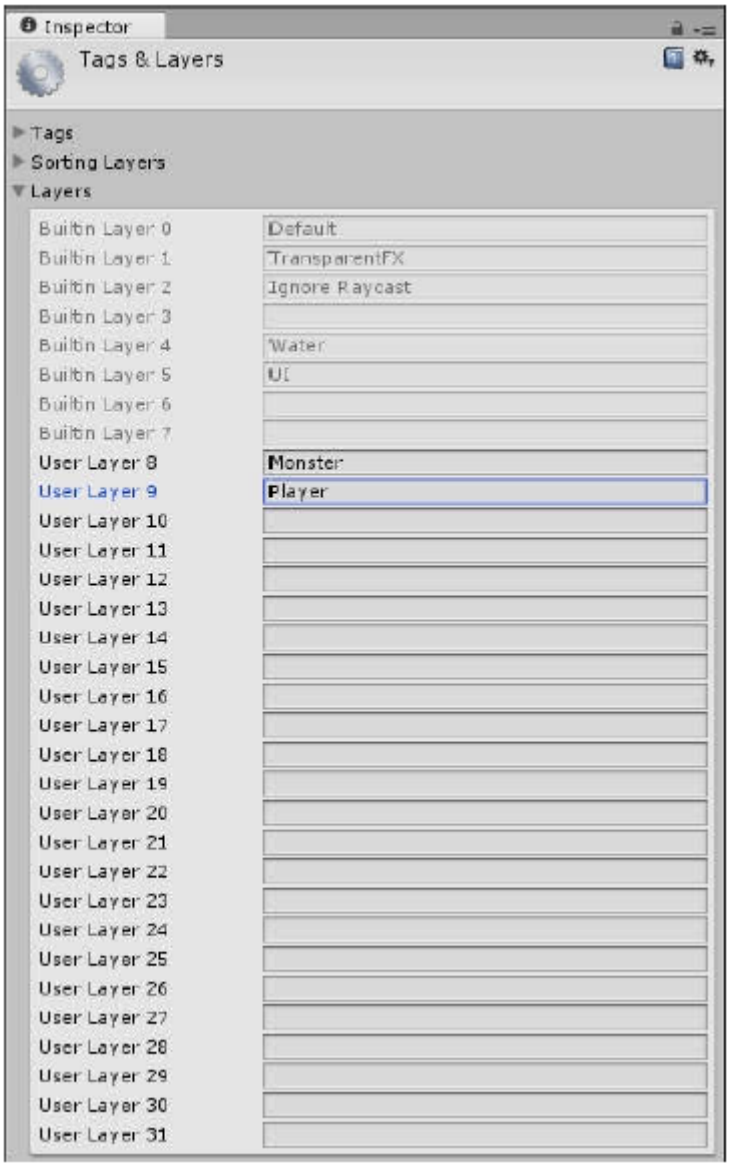
6. Po podświetleniu potwora w oknie Inspektora kliknij przycisk Zastosuj w górnej części okna pod akcjami z prefabrykatów. Spowoduje to zastosowanie zmian do prefabrykatów. Pozostaw obiekt Monster w oknie Hierarchii.

7. Naciśnij Play w edytorze, aby uruchomić grę. Obserwując przebieg gry, miejmy nadzieję, zauważysz, że postać ląduje teraz na szczycie potwora, a następnie zeskakuje. Biedny potwór niestety przewraca się, a potem po prostu toczy. Jeśli nie widzisz tego za pierwszym razem, spróbuj uruchomić grę kilka razy, aż zobaczysz to.

8. Nie dodamy funkcjonalności, aby potwór stanął z powrotem na nogi. Jednak to, co zrobimy, nie pozwoli graczowi na interakcję z obiektami potworów. Najlepszą analogią do tego w świecie rzeczywistym jest nasza gra, w której tworzymy duchy potworów; można je zobaczyć i usłyszeć, ale nie można ich dotknąć.

9. Wybierz obiekt Potwór. W oknie Inspektora wybierz menu rozwijane Warstwy, a następnie wybierz Dodaj warstwę...

10. Otworzy się panel Znaczniki i warstwy. Dodaj do listy dwie nowe warstwy o nazwie Monster i Player, jak pokazano na poniższym zrzucie ekranu:



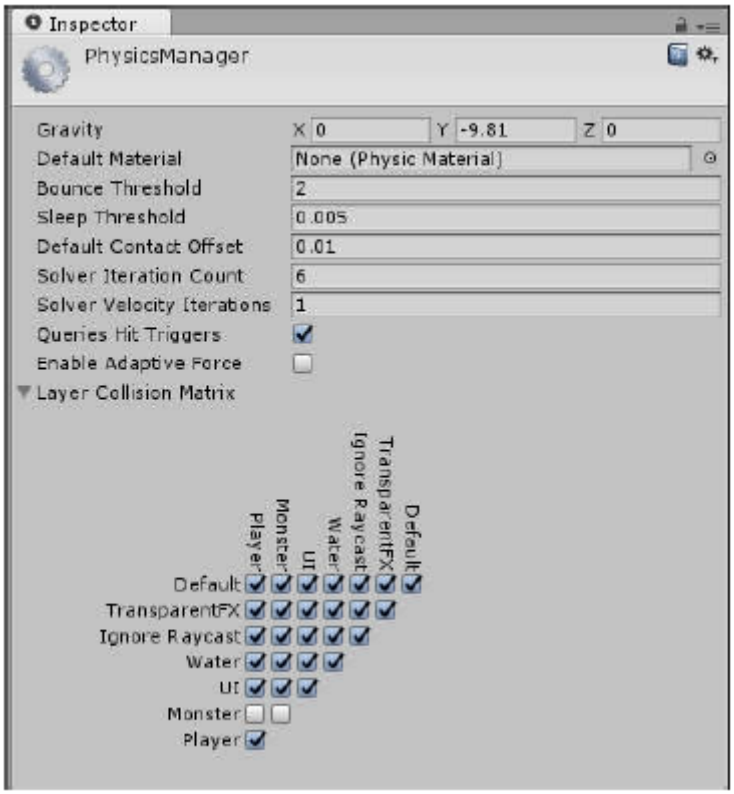
11. Ponownie wybierz obiekt Monster w oknie Hierarchy. Potwór będzie teraz znajdował się na warstwie potwora pokazanej na liście rozwijanej Warstwa.

12. Wybierz obiekt Player w oknie Hierarchia. Wybierz menu rozwijane Warstwa w oknie Inspektora i zmień warstwę na Odtwarzacz. Zostaniesz poproszony o zmianę dzieci; kliknij Tak, zmień dzieci w następujący sposób:

13. Umieszczenie potworów na nowej warstwie pozwoli nam nie tylko kontrolować interakcje fizyczne, ale także zoptymalizować testy kolizji. Jeśli pamiętasz, metoda Physics.Raycast przyjęła parametr maski warstwy. Teraz, gdy nasze potwory znajdują się na warstwie o nazwie Monster, możemy zoptymalizować test zderzeń promieni tylko do warstwy Monster.

14. Wybierz pozycję menu Edytuj | Ustawienia projektu | Fizyka. Spowoduje to otwarcie panelu PhysicsManager w oknie Inspektora.

15. Odznacz pola wyboru Monster-Player i Monster-Monster w macierzy kolizji warstw:



16. Edytując macierz kolizji warstw, aby uniemożliwić potworom kolidowanie z graczami lub innymi potworami, unikamy potencjalnych problemów.

17. Uruchom grę ponownie w edytorze, naciskając Play. Rzeczywiście, nasz gracz nie niszczy już potwora i na świecie znów wszystko jest w porządku.

Więc teraz nasze potwory mają zderzacze, co oznacza, że nasz promień dotykowy powinien być w stanie się z nimi zderzyć. Postępuj zgodnie z poniższymi instrukcjami, aby skonfigurować wybór dotykiem i przetestować wybieranie potworów dotykiem:

1. Wybierz obiekt Monster w oknie Hierarchia. Kliknij przycisk Zastosuj w ustawieniach prefabrykatów w oknie Inspektora, aby upewnić się, że wszystkie zmiany zostały zapisane.

2. Usuń prefabrykat Potwora ze sceny, wybierając go w oknie Hierarchii i naciskając Usuń.

3. Zapisz scenę mapy. Następnie otwórz scenę Gra z folderu Zasoby w oknie Projekt.

4. Wybierz _GameManager w oknie Hierarchia. W oknie Inspektora upewnij się, że nazwa warstwy potwora nazywa się potwór.

5. Uruchom grę, naciskając Play w edytorze. Kliknij na potwory. Powinieneś zobaczyć komunikat w oknie konsoli, że potwór został trafiony.

6. Zbuduj i wdróż grę na swoje urządzenie mobilne. Upewnij się, że dołączasz również okno konsoli CUDLR po uruchomieniu gry na urządzeniu. Stuknij w potwory i zanotuj komunikaty dziennika, które są przesyłane do konsoli CUDLR.

Gracz może teraz zainicjować łapanie potwora, dotykając go. Normalnie chcielibyśmy również kontrolować odległość, na jaką gracz musi znajdować się od potwora, aby go złapać. Na razie jednak założymy, że każdy potwór, którego można zobaczyć, może zostać złapany. Ułatwi to testowanie naszej gry, zwłaszcza w Trybie symulacji GPS. Później, gdy zaczniemy dodawać kolejne obiekty i miejsca na mapie, przejdziemy do ustawiania odległości interakcji.

Budowanie sceny AR Catch

Nagromadzenie się skończyło; wreszcie jesteśmy tutaj: tworzymy scenę akcji, w której gracze dzielnie łapią potwora. Będzie to również nasz wstęp do włączenia AR do naszej gry. Jest wiele rzeczy do zrobienia, aby zakończyć to wszystko do końca rozdziału, więc zaczynajmy. Zaczniemy od stworzenia nowej sceny Catch:

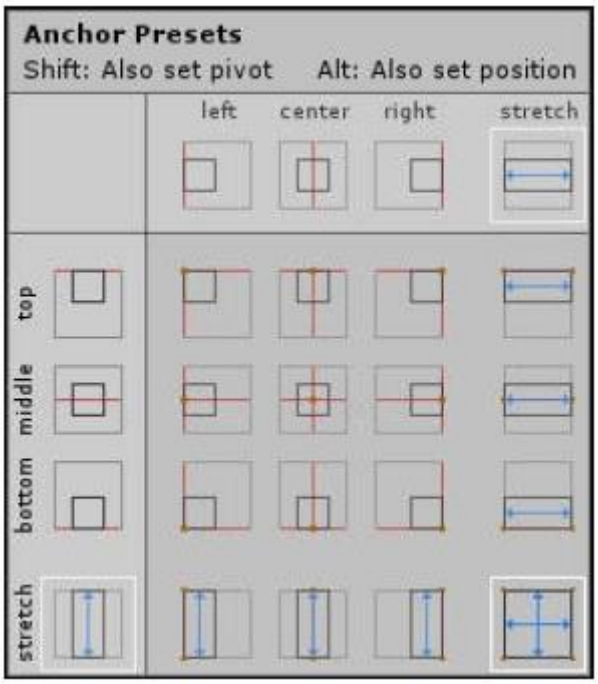
1. Utwórz nową scenę, wybierając pozycję menu Plik | Nowa scena. Następnie wybierz Plik | Zapisz scenę jako ... z menu. W oknie dialogowym Zapisz scenę wprowadź nazwę Catch i kliknij przycisk Zapisz.

2. Z menu wybierz GameObject | Utwórz puste. Zmień nazwę nowego obiektu CatchScene i zresetuj jego transformację do zera w oknie Inspektora.

3. W oknie Hierarchia przeciągnij i upuść obiekty Main Camera i Directional Light na obiekt CatchScene, aby zmienić ich rodzicielstwo.

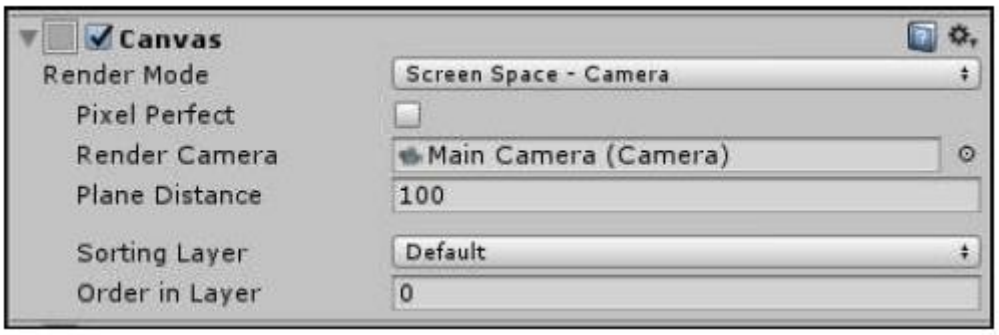
4. Z menu wybierz GameObject | interfejs użytkownika | Surowy obraz. To stworzy obiekt Canvas z Raw Image jako dziecko. Wybierz i zmień nazwę obiektu RawImage na Camera_Backdrop w oknie Inspektora.

5. Po wybraniu Camera_Backdrop w oknie Inspektora ustaw ustawienia kotwicy na rozciąganie z rozciąganiem, klikając ikonę kotwic, aby wyświetlić menu. Następnie przytrzymaj klawisze set pivot i set position podczas zaznaczania w prawym dolnym rogu w następujący sposób:



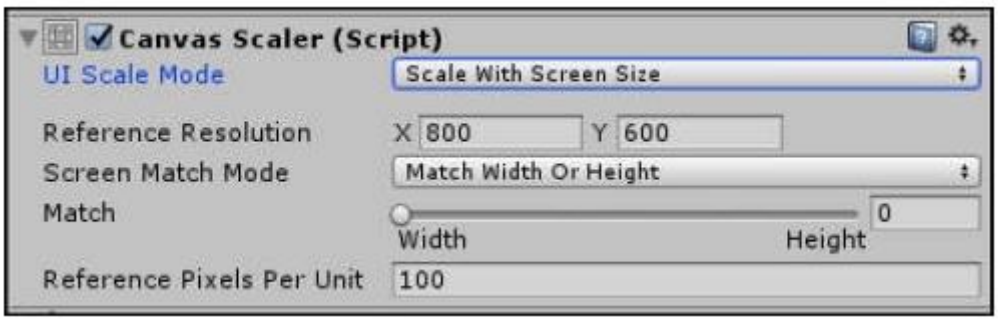
6. W oknie Hierarchia przeciągnij i upuść obiekt Canvas na obiekt CatchScene.

7. Wybierz obiekt Płótno. W oknie Inspektora zlokalizuj składnik Canvas. Zmień tryb renderowania na Przestrzeń ekranu — Kamera. Następnie przeciągnij i upuść obiekt Kamera główna z okna Hierarchii do otwartego slotu Kamera renderowania w komponencie Płótno; zapoznaj się z poniższym zrzutem ekranu, aby uzyskać prawidłowe ustawienia:



Wskazówka : Różnica między trybami renderowania Przestrzeń ekranu - Nakładka i Przestrzeń ekranu - Kamera polega na położeniu płaszczyzny renderowania. W trybie nakładki wszystkie elementy interfejsu użytkownika są renderowane przed wszystkim innym w scenie. Natomiast w trybie kamery płaszczyzna interfejsu użytkownika jest tworzona w stałej odległości od kamery. W ten sposób obiekty świata mogą być renderowane przed elementami interfejsu użytkownika.

8. Mając nadal zaznaczony obiekt Canvas, zmień tryb skalowania interfejsu użytkownika w komponencie Canvas Scaler na Scale with Screen Size w oknie Inspektora w następujący sposób:



Wskazówka: Skaler płótna ze skalowaniem z rozmiarem ekranu wymusza na aparacie utrzymywanie proporcji w miarę zmiany rozdzielczości ekranu. Jest to dla nas ważne, ponieważ nie chcemy, aby obraz, który będzie kamerą urządzenia zmieniał proporcje lub zniekształcał.

9. Wybierz i usuń obiekt EventSystem. Nie będziemy go tutaj potrzebować, a nawet jeśli to zrobimy, Unity stworzy go dla nas. Na koniec zapisz scenę. To kończy podstawową scenę Catch. Czas przenieść się do świata AR.

Używanie aparatu jako tła sceny

Jak mogłeś już zauważyć, tworząc scenę Catch, umieściliśmy już element interfejsu użytkownika, który będzie działał jako tło naszej sceny. Skonfigurowany przez nas obiekt Camera_Backdrop wyświetli kamerę urządzenia jako teksturę. Postępuj zgodnie z tymi instrukcjami, aby dodać skrypt i uzyskać widok z kamery jako tło sceny:

1. Z menu wybierz Zasoby | Importuj pakiet | Pakiet niestandardowy...

2. Po otwarciu okna dialogowego Importuj pakiet... przejdź do pobranego folderu z kodem źródłowym Chapter_5_Assets. Wybierz pakiet Chapter5_import2.unity i kliknij Otwórz, aby zaimportować pakiet.

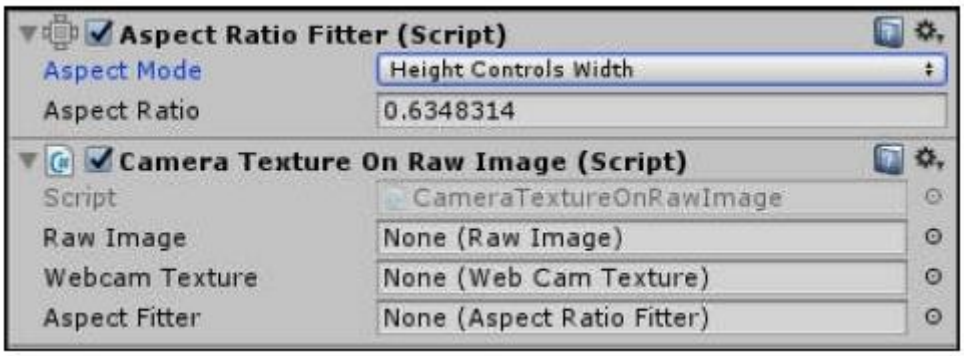
Wskazówka : ten pakiet jest pełnym importem zasobów FoodyGo i nie wszystkie skrypty mogą wymagać aktualizacji. Jeśli jednak dokonałeś własnych modyfikacji niektórych skryptów, spowoduje to nadpisanie zmian. Jeśli masz zmiany, które chcesz zachować, utwórz kopię zapasową plików w nowej lokalizacji lub wyeksportuj tylko te pliki, które chcesz zachować.

3. Po załadowaniu okna dialogowego Unity Import Package przejrzyj pliki, które chcesz zaimportować. Będzie wiele nowych plików i kilka zmienionych plików. Tylko upewnij się, że wszystkie importy są w porządku, a następnie kliknij Importuj.

4. Wybierz obiekt Camera_Backdrop w oknie Hierarchia. W w oknie Inspektora kliknij przycisk Dodaj składnik u dołu okna. Wybierz z menu komponent Dopasowanie proporcji.

5. W komponencie Aspect Ratio Fitter zmień Aspect Mode na Height Controls Width.

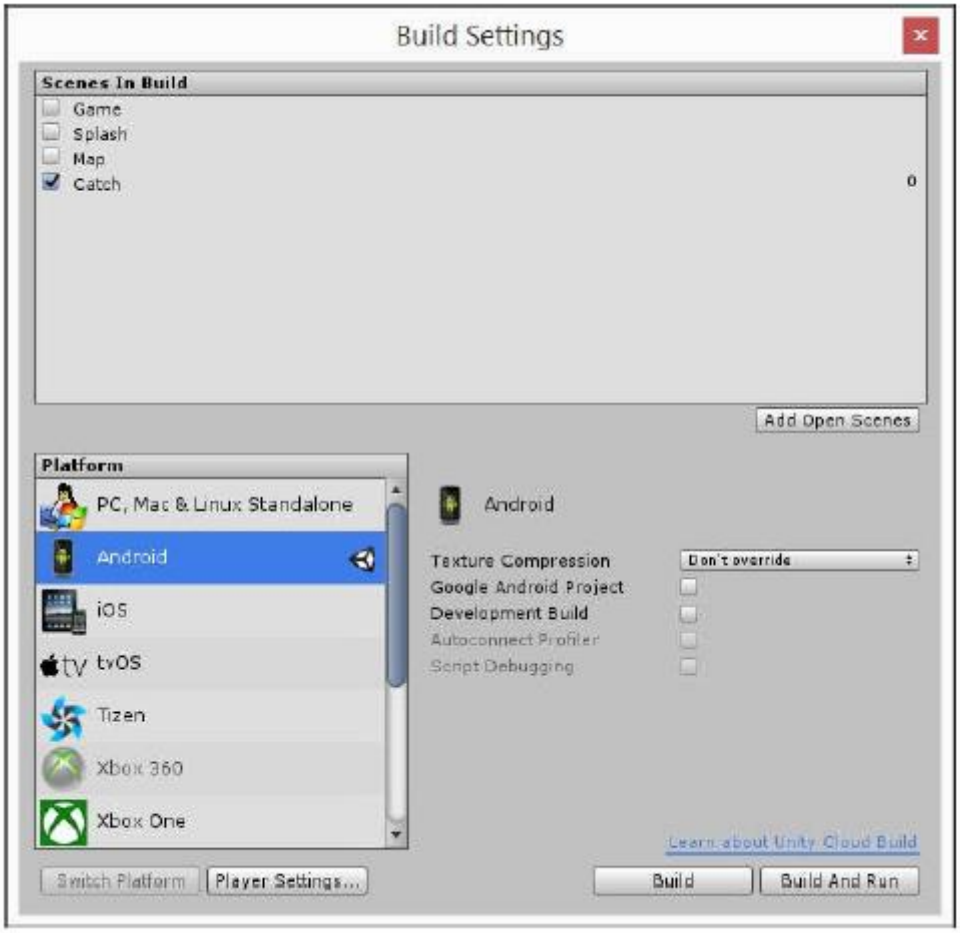
6. Kliknij ponownie przycisk Dodaj komponent i wybierz z menu komponent Tekstura kamery na obrazie surowym. Nie musisz ustawiać żadnego z pól w tym komponencie. Poniższy zrzut ekranu przedstawia prawidłowe ustawienia nowych komponentów:



7. Zapisz scenę, wybierając pozycję menu Plik | Zapisz scenę.

8. Z menu wybierz Plik | Ustawienia budowania, aby otworzyć okno dialogowe Ustawienia budowania. Kliknij przycisk Dodaj otwarte sceny w oknie dialogowym, aby dodać scenę Catch.

9. Odznacz sceny Game, Map i Splash w oknie budowania i zaznacz scenę Catch w następujący sposób:



10. Upewnij się, że urządzenie mobilne jest podłączone przez USB do maszyny deweloperskiej, a następnie kliknij przycisk Zbuduj i uruchom. Po skompilowaniu i wdrożeniu gry na urządzeniu mobilnym zwróć uwagę na tło z kamery urządzenia. Obróć urządzenie, aby zmienić orientację i zwróć uwagę na kurczenie się tła; naprawimy to w następnej sekcji. Jak widać, jesteśmy teraz na dobrej drodze, aby zapewnić graczowi doświadczenie rzeczywistości rozszerzonej w grze. Dzięki kamerze urządzenia zapewniającej tło sceny, gracz poczuje, że obiekty gry znajdują się w tym samym obszarze,

co one. Cała praca nad dodaniem tła kamery jest wykonywana w skrypcie CameraTextureOnRawImage. Przejrzyj skrypt, postępując zgodnie z tym przewodnikiem:

1. Otwórz ten skrypt w wybranym edytorze, klikając ikonę koła zębatego obok komponentu Tekstura kamery na nieprzetworzonym obrazie w oknie Inspektora. Następnie wybierz Edytuj skrypt z menu kontekstowego. Skoncentrujmy się na kodzie w metodzie Awake:

```
void Awake()

{

webcamTexture = new

WebCamTexture(Screen.width, Screen.height);

rawImage = GetComponent< RawImage >

();

aspectFitter =

GetComponent< AspectRatioFitter >();

rawImage.texture = webcamTexture;

rawImage.material.mainTexture =

webcamTexture;

webcamTexture.Play();

}
```

2. Metoda Awake jest wywoływana za każdym razem, gdy obiekt jest aktywowany, co zwykle ma miejsce zaraz po wywołaniu metody Start. Będzie to ważne później, gdy przejdziemy do przełączania scen. W przeciwnym razie kod tutaj jest dość prosty, a główne elementy są inicjowane. WebCamTexture to opakowanie Unity dla kamery internetowej lub kamery urządzenia. Po zainicjowaniu webcamTexture jest ona stosowana jako tekstura do rawImage. RawImage jest elementem tła interfejsu użytkownika w scenie. Następnie kamera jest włączana poprzez wywołanie metody Play.

3. Nie będziemy tu zagłębiać się w szczegóły metody Update w skrypcie, ponieważ zajmuje się ona niektórymi artefaktami orientacji urządzenia, które wymagają manipulacji w celu prawidłowego wyświetlania. Ważnym wnioskiem jest to, że musimy manipulować teksturą kamery, aby właściwie zorientowała się w tle sceny. Poza tym, jeśli pamiętasz, nadal mieliśmy jeden problem, gdy urządzenie zostało obrócone do orientacji poziomej.

4. Aby rozwiązać problem z orientacją poziomą, po prostu po prostu zignoruj lub zablokuj opcję. Ustalenie orientacji dla wszystkich typów urządzeń i ustawień wykracza poza zakres tej książki. W końcu scena Catch i tak nie będzie grać dobrze w orientacji poziomej, więc od teraz zmusimy grę do korzystania z orientacji pionowej.

5. Wróć do edytora Unity i z menu wybierz Edytuj | Ustawienia projektu | Gracz. Następnie wybierz kartę Rozdzielczość i prezentacja u góry. Wewnątrz tej zakładki poszukaj domyślnej

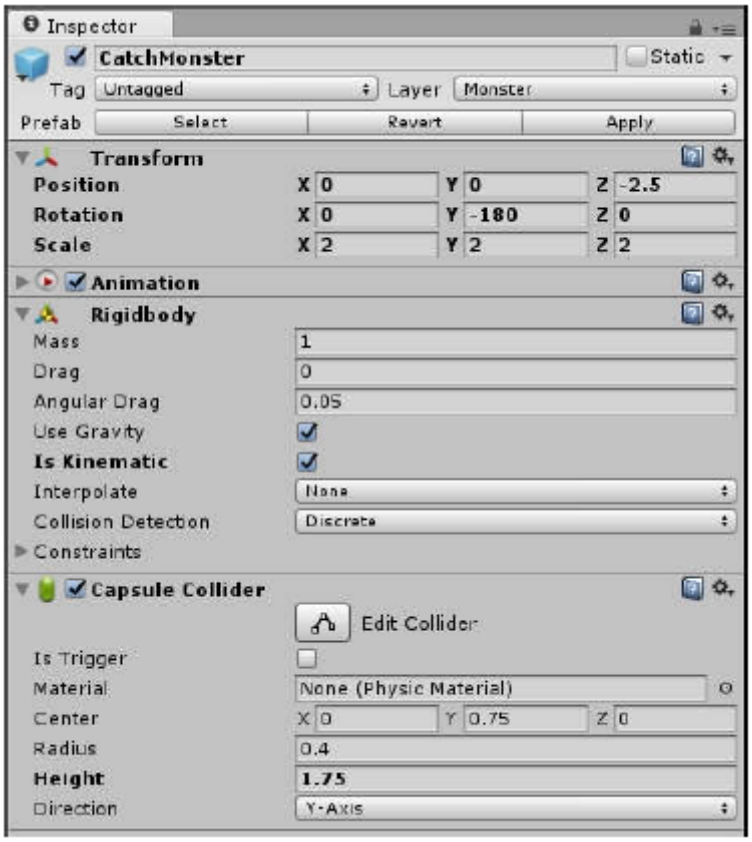
Rozwijana orientacja i zmień domyślną wartość na Pionową. W tej chwili scena jest dość nudna, w tle działa tylko kamera, więc dodajmy naszego potwora, aby było ciekawiej:

1. Z menu wybierz GameObject | 3D | Samolot. Stworzy to nową płaszczyznę na scenie. W oknie Inspektora zresetuj transformację do zera i ustaw skale X i Z na 1000. Wyłącz komponent Mesh Renderer, odznaczając pole wyboru. To sprawi, że samolot stanie się niewidzialny.

2. Przeciągnij i upuść obiekt Plane na obiekt CatchScene, aby uczynić go dzieckiem.

3. Z folderu Assets/FoodyGo/Prefabs w oknie Project przeciągnij prefabrykat Monster do okna Hierarchy i upuść go na obiekt CatchScene.

4. W oknie Inspektora zmień nazwę obiektu CatchMonster i ustaw właściwości komponentów Transform, Rigidbody i Capsule Collider, aby pasowały do wartości, jak pokazano na poniższym zrzucie ekranu:



5. Przeciągnij CatchMonster do folderu Assets/FoodyGo/Prefabs, aby stworzyć nowy prefabrykat.

6. Zbuduj i wdróż grę na swoje urządzenie mobilne i obserwuj, jak potwór pojawia się teraz w oknie gry jako sekretne okno do otaczającego Cię świata.

Kilka znanych gier przygodowych w świecie rzeczywistym wykorzystuje kamerę sterowaną żyroskopem, aby dodać dodatkowy element realizmu do doświadczenia AR. Kamera sterowana żyroskopem zmieni perspektywę na wirtualne obiekty, gdy gracz orientuje swoje urządzenie. Unikaliśmy tego w naszej grze z kilku powodów:

* Kamery żyroskopowe są trudne do zakodowania ze względu na różnice w systemie operacyjnym urządzenia i orientacji. Nawet w systemie Android różnice mogą się różnić w zależności od producenta urządzenia.

* Kamery żyroskopowe często cierpią z powodu pewnej formy dryfu, która musi być stale korygowana.

* Włączenie żyroskopu zwiększa trudność w doświadczeniu AR. W wielu innych grach w świecie rzeczywistym gracze mają możliwość wyłączenia AR i często robią to ze względu na dodatkowy poziom trudności. W naszym przypadku chcemy, aby nasi gracze cieszyli się doświadczeniem AR i jest to kolejny powód, aby nie używać kamery żyroskopowej.

Dodanie piłki do łapania

W naszej grze gracz będzie używał kul lodu do łapania potworów. Gracz musi uderzyć potwora lodowymi kulami, aby stały się chłodniejsze i wolniejsze, aby ostatecznie zamarzły w miejscu. Po błyskawicznym zamrożeniu, jak mrożony obiad, można je łatwo złapać. Kula, którą dodamy, musi być zrobiona z lodu. Ponieważ obecnie nie mamy żadnych materiałów lodowych, którymi można by teksturować piłkę, najpierw wczytamy kilka zasobów. Zasoby, które załadujemy, dotyczą efektów cząsteczkowych, których użyjemy w dalszej części rozdziału. Dogodnie, jeden z tych efektów cząsteczkowych ma również ładną teksturę lodu, której użyjemy do naszej piłki. Wykonaj poniższe instrukcje, aby zaimportować zasoby efektów cząsteczkowych:

1. Z menu wybierz Zasoby | Importuj pakiet | Systemy Cząsteczkowe. Po wyświetleniu monitu w oknie dialogowym Importuj pakiet jednostkowy kliknij przycisk Importuj. Spowoduje to zainstalowanie standardowego zasobu systemu cząstek z Unity.

Wskazówka: system cząsteczkowy Unity jest znany jako Shuriken i jest podstawą wielu zasobów efektów cząsteczkowych.

2. Otwórz okno magazynu zasobów, wybierając pozycję menu Okno | Sklep aktywów.

3. Po otwarciu okna i załadowaniu strony Asset Store wpisz elementals w polu wyszukiwania, a następnie naciśnij enter.

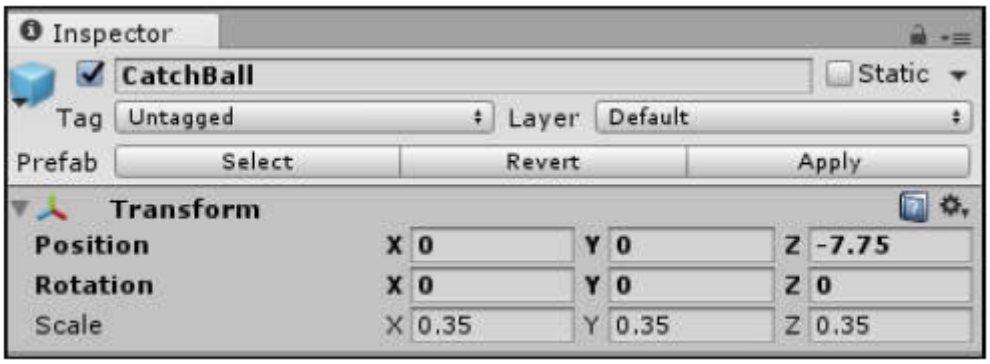
4. Poszukaj Systemów Cząsteczek Żywiołów od G.E. Zasób TeamDev na liście i wybierz ten element. Elementals to doskonały darmowy zasób, który działa dobrze na większości urządzeń mobilnych.

5. Kliknij przycisk Pobierz na stronie zasobu, aby pobrać i zaimportować zasób.

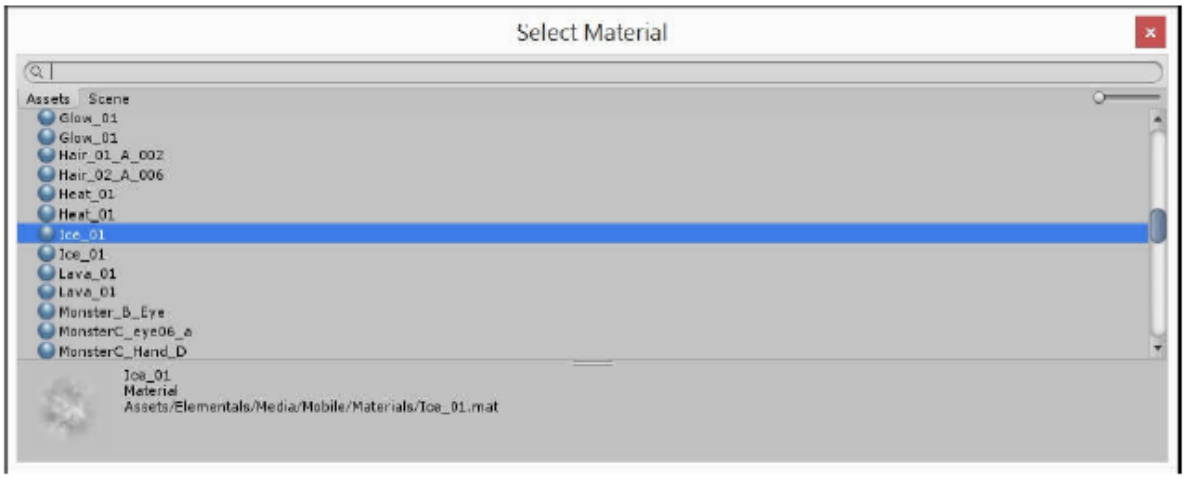
6. Po wyświetleniu monitu Import Unity Package kliknij przycisk Importuj, aby zainstalować ten zasób.

Teraz, gdy mamy zaimportowane zasoby systemu cząstek, możemy zacząć dodawać tę piłkę:

1. Z menu wybierz GameObject | 3D | Sphere, aby dodać obiekt Sphere do okna Hierarchy. Wybierz nowy obiekt, aw oknie Inspektora zmień jego nazwę CatchBall i ustaw pozycję transformacji i skalę, aby pasowały do następującego zrzutu ekranu:



2. Rozwiń listę materiałów w komponencie Mesh Renderer i wybierz ikonę dziesiątki obok opcji Default-Material. W oknie dialogowym Wybierz materiał wybierz materiał Ice_01, który wskazuje na Assets/Elementals/Media/Mobile/Materials/Ice_01.mat w następujący sposób:



3. Gdy masz otwarte okno Inspektora, kliknij przycisk Dodaj składnik i wybierz z listy Rigidbody lub wprowadź go w polu wyszukiwania.

4. Po dodaniu do piłki komponentu Rigidbody usuń zaznaczenie pola Użyj grawitacji. Za pomocą skryptu będziemy kontrolować grawitację piłki.

5. Z okna Hierarchii przeciągnij CatchBall i upuść go do folderu Assets/FoodyGo/Prefabs w oknie projektu. Spowoduje to utworzenie nowego prefabrykatu CatchBall.

6. Ponownie z okna Hierarchii przeciągnij i upuść

Rzucanie piłką

Mamy teraz świetnie wyglądającą kulę lodu tuż przed naszym chodzącym potworem. Czas wprawić piłkę w ruch, wykonując następujące kroki, aby dodać obiekty i skrypty:

1. Przeciągnij element prefabrykowany DualTouchControls z folderu Assets/Standard Assets/CrossPlatformInput/Prefabs w oknie Project do okna Hierarchy.

2. Zmień nazwę obiektu DualTouchControls na Catch_UI w oknie Inspektora.

3. Rozwiń nowy obiekt Catch_UI i usuń obiekty TurnAndLookTouchpad i Jump. Po wyświetleniu monitu o rozbicie prefabrykatu kliknij przycisk Kontynuuj.

4. Rozwiń obiekt MoveTouchpad i usuń obiekt Tekst, zaznaczając go i naciskając Delete.

5. Wybierz obiekt MoveTouchpad i zmień jego nazwę ThrowTouchpad w oknie inspektora.

6. Otwórz Anchor Presets w komponencie Rect Transform i wybierz opcję stretch-stretch w prawym dolnym rogu, przytrzymując klawisze obrotu i pozycji. Spowoduje to rozszerzenie nakładki tak, aby wypełniała ekran, tak jak robiliśmy to wcześniej w przypadku kamery o swobodnym wyglądzie.

7. Wybierz pole Kolor komponentu obrazu, aby otworzyć okno dialogowe kolorów. Ustaw kolor na #FFFFFF00 w systemie szesnastkowym.

8. Kliknij przycisk Dodaj składnik na dole okna Inspektora. Z listy rozwijanej wybierz Throw Touch Pad lub wprowadź go w polu wyszukiwania.

9. Wybierz ikonę koła zębatego obok komponentu Touchpad i wybierz Usuń komponent z menu kontekstowego.

10. Z okna Hierarchia przeciągnij obiekt CatchBall i upuść go na puste pole Throw Object w komponencie Throw Touch Pad

11. Przeciągnij obiekt Catch_UI na obiekt CatchScene, aby dodać go jako dziecko.

12. Naciśnij Play w edytorze, aby uruchomić grę. Możesz teraz kliknąć piłkę, przeciągnąć ją ruchem rzutowym, a następnie wypuścić ją, klikając myszką.

13. Zbuduj i wdróż grę na swoje urządzenie mobilne. Rzuć piłkę palcem; zobacz, jak dobrze możesz trafić potwora.

Wskazówka : Jeśli rzucanie sprawia Ci trudność, spróbuj dostosować ustawienie Szybkość rzutu w komponencie Throw Touch Pad.

Cała praca polegająca na rzucaniu piłką jest wykonywana przez skrypt ThrowTouchPad, który wykorzystuje mocno zmodyfikowany skrypt Touchpad u swojej podstawy. Przyjrzyjmy się krytycznym sekcjom kodu:

1. Otwórz skrypt ThrowTouchPad w wybranym edytorze. Oczywiście teraz wiesz, jak to zrobić.

2. Przewiń w dół do metody Start i przejrzyj inicjalizację tego skryptu. Większość inicjalizacji odbywa się wewnątrz instrukcji if, która sprawdza, czy throwObject nie ma wartości NULL. Dalsza inicjalizacja zmiennych jest wyłączona w wywołaniu ResetTarget(). Poniżej znajduje się kod do przeglądu:

```
if (throwObject != null)

{

startPosition =

throwObject.transform.position;

startRotation =

throwObject.transform.rotation;

throwObject.SetActive(false);
```

```
ResetTarget();
}
```

3. Przewiń w dół do metody OnPointerDown; kod metody jest pokazany tutaj do przeglądu:

```
public void
OnPointerDown(PointerEventData data)
{
Ray ray =
Camera.main.ScreenPointToRay(data.position);
RaycastHit hit;
if (Physics.Raycast(ray, out hit,
100f))
{
//check if target object was hit
if (hit.transform ==
target.transform)
{
//yes, start dragging the
object
m_Dragging = true;
m_Id = data.pointerId;
screenPosition =
Camera.main.WorldToScreenPoint
(target.transform.position);
offset =
target.transform.position -
Camera.main.ScreenToWorldPoint(new
Vector3(data.position.x,
data.position.y,
screenPosition.z));
}
}
```

}

4. Ten kod jest bardzo podobny do tego, którego używaliśmy wcześniej, aby wybrać potwory, które musisz złapać. Jak widać, używana jest ta sama metoda Physics.RayCast, bez maski warstwy. Jeśli wskaźnik (dotyk) w coś trafi, sprawdza, czy jest to cel, którym jest CatchBall. Jeśli trafi w cel, ustawia wartość logiczną m_Dragging na true i pobiera pozycję na ekranie dotykanego obiektu oraz przesunięcia wskaźnika lub dotyku.

5. Następnie, jeśli przewiniesz trochę w dół do metody Update, zobaczysz instrukcję if, która sprawdza, czy piłka jest przeciągana, sprawdzając wartość logiczną m_Dragging Boolean. Jeśli tak, robi migawkę bieżącej pozycji wskaźnika (dotyku) i wywołuje metodę OnDragging do przeglądu:

```
void OnDragging(Vector3 touchPos)

{

//track mouse position.

Vector3 currentScreenSpace = new

Vector3(Input.mousePosition.x,

Input.mousePosition.y,

screenPosition.z);

//convert screen position to world

position with offset

object

m_Dragging = true;

m_Id = data.pointerId;

screenPosition =

Camera.main.WorldToScreenPoint

(target.transform.position);

offset =

target.transform.position -

Camera.main.ScreenToWorldPoint(new

Vector3(data.position.x,

data.position.y,

screenPosition.z));

}

}

}
```

```
p

changes.

Vector3 currentPosition =

Camera.main.ScreenToWorldPoint(currentScreenSp

ace) + offset;

//It will update target

gameobject's current postion.

target.transform.position

= currentPosition;

}
```

6. Metoda OnDragging po prostu przesuwa obiekt docelowy (piłkę) po ekranie w oparciu o pozycję wskaźnika (dotyku).

7. Następnie przewiń w dół do metody OnPointerUp. Metoda OnPointerUp jest wywoływana po zwolnieniu przycisku myszy lub usunięciu dotyku. Kod wewnątrz metody jest dość prosty: ponownie sprawdza, czy m_Dragging jest prawdziwe; jeśli nie, to po prostu wraca. Jeśli obiekt jest przeciągany, w tym momencie wywoływana jest metoda ThrowObject, jak pokazano w poniższym kodzie:

```
void ThrowObject(Vector2 pos)

{

rb.useGravity = true; //turn on

gravity

float y = (pos.y - lastPos.y) /

Screen.height * 100;

speed = throwSpeed * y;

float x = (pos.x / Screen.width) -

(lastPos.x / Screen.width);

x = Mathf.Abs(pos.x - lastPos.x) /

Screen.width * 100 * x;

Vector3 direction = new Vector3(x,

0f, 1f);

direction =

Camera.main.transform.TransformDirection(direction);

rb.AddForce((direction * speed * 2f
```

```
) + (Vector3.up *

speed/2f));

thrown = true;

var ca =

target.GetComponent<CollisionAction>();

if(ca != null)

{

ca.disarmmed = false;

}

Invoke("ResetTarget", 5);

}
```

8. Metoda ThrowObject polega na obliczeniu pozycji startu i określeniu siły, z jaką obiekt zostanie rzucony. Obliczenia x,y określają, jak szybko obiekt poruszał się po ekranie, zanim został zwolniony. Określa to, biorąc różnicę między ostatnią znaną pozycją wskaźnika a pozycją zwolnienia. Wartość x lub pozycja wyzwolenia określa kierunek (w lewo lub w prawo) rzutu, podczas gdy ruch y lub w górę określa prędkość rzutu. Wartości te są następnie sumowane do wektora siły i stosowane do ciała sztywnego przez wywołanie rb.AddForce(). rb jest docelowym korpusem sztywnym, który został ustawiony w metodzie ResetTarget podczas inicjalizacji. Na dole metody znajduje się wywołanie GetComponent dla składnika CollisionAction. Nie będziemy się tym martwić tutaj, ale omówimy to później. Następnie ponownie wywołujemy ResetTarget przy użyciu metody Invoke, która czeka 5 sekund przed wywołaniem.

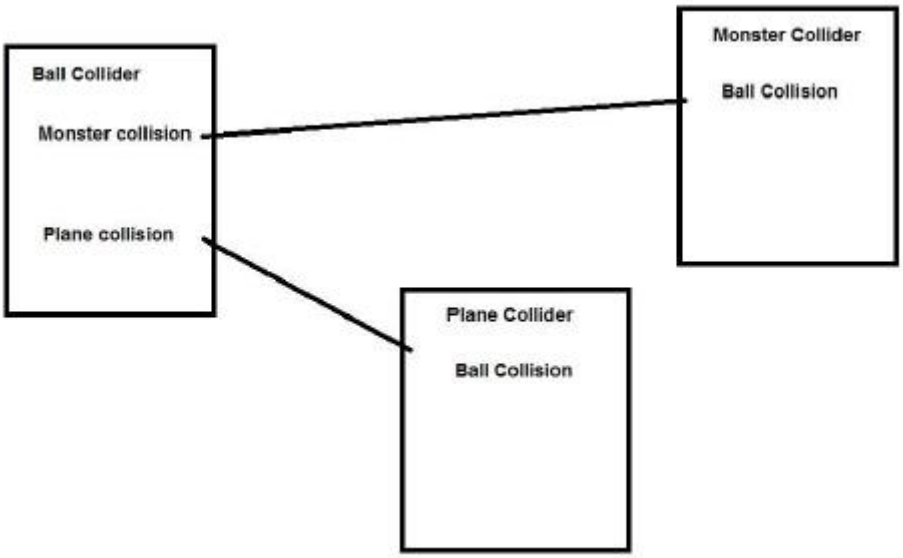
Wskazówka: Rigidbody.AddForce to jedna z najważniejszych metod do opanowania podczas tworzenia dowolnej gry z fizyką.

Sprawdzanie kolizji

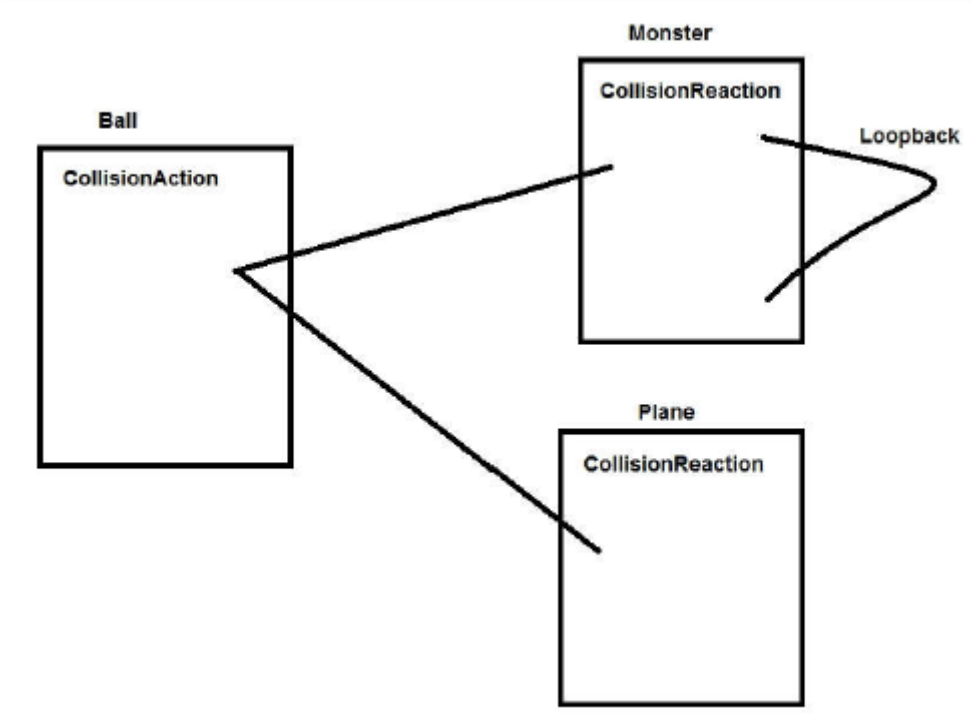
Do tej pory gracz może rzucić piłkę w potwora, z niewielkim skutkiem. Jeśli udało ci się trafić potwora podczas testów, prawdopodobnie zauważyłeś, że piłka po prostu się odbija, co z pewnością nie jest wynikiem, którego szukamy. To, czego teraz potrzebujemy, to sposób na wykrycie, kiedy piłka uderza w potwora lub w samolot. Na szczęście silnik fizyki Unity ma kilka metod określania kolizji obiektu z innym obiektem. Oto standardowe opcje:

* OnCollisionEnter: Obiekt ma zderzacz, który wchodzi w kontakt z innym obiektem gry, który również ma zderzacz. Obiekty nawiążą kontakt, a następnie odepchną się od siebie w zależności od siły zderzenia i jeśli jeden lub oba obiekty mają przymocowany sztywny korpus. Obiekt nie potrzebuje ciała sztywnego do zderzenia, ale jak widzieliśmy, potrzebuje zderzacza. OnTriggerEnter: Dzieje się tak, gdy obiekt ma zderzacz, ale zderzacz jest ustawiony jako wyzwalacz. Zderzacz ustawiony jako wyzwalacz wykryje kolizje, ale następnie pozwoli obiektowi przejść przez niego. Jest to przydatne w przypadku takich rzeczy, jak drzwi, portale lub inne obszary, które chcesz wykryć, gdy obiekt wejdzie. Jak już podejrzewałeś, będziemy używać OnCollisionEnter do określenia, kiedy nasze obiekty się zderzają. Jednak zamiast pisać pojedynczy skrypt dla każdego obiektu, który chcemy sprawdzić pod kątem kolizji, zamiast tego zaimplementujemy system zdarzeń kolizji. Problem z pisaniem skryptu kolizyjnego dla każdego obiektu polega na tym, że często otrzymujesz zduplikowany kod w różnych skryptach

dołączonych do różnych obiektów. Każdy skrypt często zarządza wtedy własnymi kolizjami, ale często ma inne zasady w zależności od tego, z jakim obiektem się koliduje. Spójrz na poniższy diagram, który pokazuje, jak to może działać:



Jak widać na diagramie, oba obiekty Monster i Plane potrzebują części tego samego kodu, aby poradzić sobie z kolizją kuli. Dodatkowo obiekt kulowy musi inaczej reagować na uderzenia potwora lub samolotu. Jeśli dodamy więcej obiektów do sceny, musielibyśmy rozszerzyć ten skrypt, aby uwzględnić każdą kolizję obiektów. Chcemy uogólnionego sposobu działania i reagowania na zderzenia, który jest również rozszerzalny. Zamiast pisać trzy niestandardowe skrypty do zarządzania kolizjami w naszej scenie, użyjemy dwóch, jednego do kolizji (akcji), a drugiego do reakcji obiektu. Skrypty są trafnie nazwane CollisionAction i CollisionReaction, a poprawiony diagram pokazujący je dołączone do obiektów sceny jest następujący:



Zanim przyjrzymy się bliżej kodowi skryptu, dodamy do sceny skrypty kolizyjne. Wykonaj poniższe instrukcje, aby dodać skrypty do obiektów:

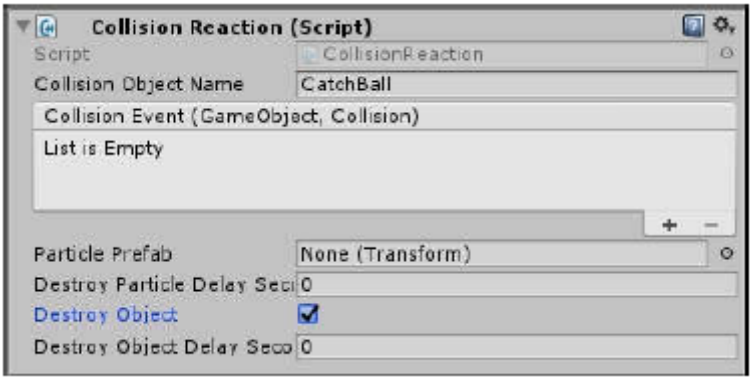
1. Wróć do edytora Unity i otwórz folder Assets/FoodyGo/Scripts/PhysicsExt w oknie projektu. Zobaczysz skrypty CollisionAction i CollisionReaction w folderze.

2. Przeciągnij i upuść skrypt CollisionAction na obiekt CatchBall w oknie Hierarchia. Jeśli nie widzisz obiektu CatchBall, po prostu rozwiń obiekt CatchScene.

3. Przeciągnij i upuść skrypt CollisionReaction na obiekty CatchMonster i Plane w oknie Hierarchy.

4. Wybierz obiekt Płaszczyzna w oknie Hierarchia.

5. W oknie Inspektora zmień reakcję na kolizję komponent pasującego do poniższego zrzutu ekranu:



6. Zauważ, że ustawiamy nazwę obiektu (CatchBall), dla którego chcemy, aby ten komponent zarządzał reakcjami. Zaznaczając opcję Destroy Object, upewniamy się, że piłka zostanie teraz zniszczona po uderzeniu w samolot. Nie martw się jeszcze o ustawianie ustawień cząsteczek; wkrótce do tego dojdziemy.

7. Wybierz obiekt CatchMonster w oknie Hierarchia i powtórz krok 4.

8. Uruchom grę w edytorze, naciskając Play. Przetestuj skrypty kolizji, rzucając piłką w potwora i samolot. Kiedy piłka uderza teraz w potwora lub samolot, natychmiast zostaje zniszczona.

Jak widzieliście, ustawienie tych skryptów kolizji było dość proste. Możesz się spodziewać, że te skrypty będą dość złożone, ale na szczęście są one dość proste. Otwórz skrypty w wybranym przez siebie edytorze, a my je przejrzymy:

* CollisionAction: Ten skrypt jest dołączony do obiektów, które mają kolidować z innymi obiektami, takimi jak nasza piłka lub pociski i tak dalej. Skrypt wykrywa kolizję, a następnie powiadamia obiekty za pomocą składnika CollisionReaction o wystąpieniu kolizji. Przyjrzyjmy się metodzie OnCollisionEnter:

```
void OnCollisionEnter(Collision

collision)

{

if (disarmed == false)

{
```

```
reactions =
collision.gameObject.GetComponents<CollisionRe
action>();
if(reactions != null &&
reactions.Length>0)
{
foreach (var reaction in reactions)
{
if (gameObject.name.StartsWith
(reaction.collisionObjectName))
{
reaction.OnCollisionReaction(gameObject, collision);
}
}
}
}
}
```

Ta metoda w rzeczywistości nie zarządza kolizją. Zamiast tego powiadamia komponenty CollisionReaction innego obiektu, z którym doszło do kolizji. Po pierwsze, upewnia się, że obiekt nie jest rozbrojony. Obiekt jest uzbrojony, gdy gracz go zaczepia i rzuca. Następnie pobiera wszystkie komponenty CollisionReaction, które mogą znajdować się w obiekcie kolizji. Obiekt może mieć wiele dołączonych komponentów CollisionReaction, które reagują inaczej na każdy obiekt. Następnie przechodzi w pętlę przez wszystkie reakcje kolizyjne i upewnia się, że obiekt chce obsłużyć kolizje, testując nazwę obiektu Collision. Jeśli tak, wywołuje metodę OnCollisionReaction.

Wskazówka: użyjemy tutaj nazw obiektów gry, aby filtrować kolizje obiektów. Lepszą implementacją byłoby użycie tagów. Od czytelnika będzie zależeć, czy będzie sumienny i sam dokona tej zmiany.

* CollisionReaction: Zajmuje się niechlujnymi szczegółami tego, co się dzieje, gdy obiekt się z nim zderzy; kod jest dość prosty:

```
public void
OnCollisionReaction(GameObject go, Collision collision)
{
ContactPoint contact =
collision.contacts[0];
Quaternion rot =
```

```
Quaternion.FromToRotation(Vector3.up,
contact.normal);
Vector3 pos = contact.point;
if (particlePrefab != null)
{
var particle =
(Transform)Instantiate(particlePrefab,
pos, rot);
Destroy(particle.gameObject,
destroyParticleDelaySeconds);
}
if (destroyObject)
{
Destroy(go,
destroyObjectDelaySeconds);
}
collisionEvent.Invoke(gameObject,
collision);
}
```

WSKAZÓWKA: Jest to prosta implementacja skryptu CollisionReaction, ale można ją rozszerzyć lub odziedziczyć w celu zastosowania innych ogólnych efektów, takich jak kalkomanie, uszkodzenia i tak dalej.

Pierwsza część kodu określa punkt uderzenia i kierunek zderzenia. Następnie sprawdza, czy ustawiono cząsteczkę Prefab. Jeśli tak, tworzy instancję prefabrykatu w punkcie uderzenia. Następnie wywołuje metodę Destroy z opóźnieniem określonym we właściwościach. Następnie sprawdza, czy kolidujący obiekt powinien zostać zniszczony. Jeśli tak, niszczy obiekt po opóźnieniu określonym w ustawieniach. Na koniec wywoływane jest zdarzenie Unity o nazwie kolizjaEvent, a kolizja jest przekazywana do wszystkich odbiorników tego zdarzenia. Dzięki temu inne niestandardowe skrypty mogą subskrybować to zdarzenie i w razie potrzeby dodatkowo obsługiwać kolizję. Użyjemy tego zdarzenia, aby później zająć się efektami zamrażania potwora.

* CollisionEvent: Na dole skryptu CollisionReaction znajduje się inna definicja klasy dla zdarzenia CollisionEvent : UnityEvent<GameObject, Collision>, która jest po prostu pusta. Jest to definicja niestandardowego zdarzenia Unity, którego używamy do powiadamiania innych skryptów lub komponentów o wystąpieniu kolizji. Zdarzenie Unity jest podobne do zdarzenia C# lub wzorca delegata, którego używaliśmy wcześniej, ale ma mniejszą wydajność. Jednak Unity Events można łatwo

połączyć między komponentami w edytorze, a nie zakodować w skrypcie, co jest niezbędne dla każdego skryptu, który chcemy uogólnić.

Efekty cząsteczkowe dla opinii

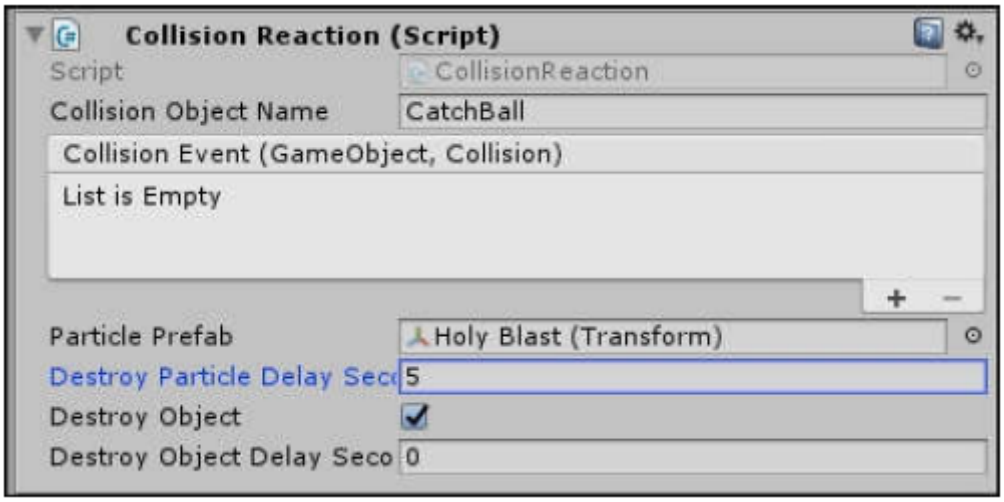
Efekty cząsteczkowe w grach są jak masło dla francuskiego szefa kuchni: niezbędne. Zapewniają one nie tylko pizzę i efekty specjalne, które wszyscy widzimy w grach, ale często są również subtelnie używane jako wskazówki do aktywności graczy. Użyjemy efektów cząsteczkowych dla niektórych pizzazz, aby ulepszyć sceny a także dostarczają wskazówek wizualnych. Nie zdążymy wejść w tło efektów cząsteczkowych, ale zostaną one omówione ponownie w Części 8, Interakcja ze światem AR. Teraz zajmijmy się dodaniem kilku efektów cząsteczkowych do naszej sceny:

1. Wróć do edytora Unity i otwórz folder Assets/Elementals/Prefabs(Mobile)/Light w oknie Projekcie.

Wskazówka: Zasób Żywiołów jest naprawdę dobrze zrobiony i możesz nawet spróbować nie używać wersji mobilnej, próbując prefabrykatów w folderze Zasoby/Żywioły/Prefabrykaty.

2. Wybierz obiekt Płaszczyzna w oknie Hierarchia. Jeśli nie widzisz obiektu Plane, pamiętaj o rozwinięciu obiektu CatchScene.

3. Przeciągnij prefabrykat Holy Blast z okna Project na pole Particle Prefab w komponencie Collision Reaction w oknie Inspektora. Zmień pole Destroy Particle Delay Seconds na 5 w następujący sposób:



4. Wybierz CatchMonster w oknie Hierarchii i powtórz krok 3, dokładnie w ten sam sposób.

5. Naciśnij Play w oknie edytora i uruchom grę, aby przetestować. Teraz z pewnością wygląda to lepiej. Zbuduj i wdróż grę na swoim urządzeniu mobilnym i tam też przetestuj.

6. Zachęcamy do wypróbowania i przetestowania innych prefabrykatów cząsteczkowych w komponentach Reakcji Zderzeniowej. Zobacz jakie ciekawe efekty są dostępne. Możesz także poeksperymentować z Unity Standard Assets Particle System znajdującym się w folderze Assets/Standard Assets/ParticleSystems/Prefabs.

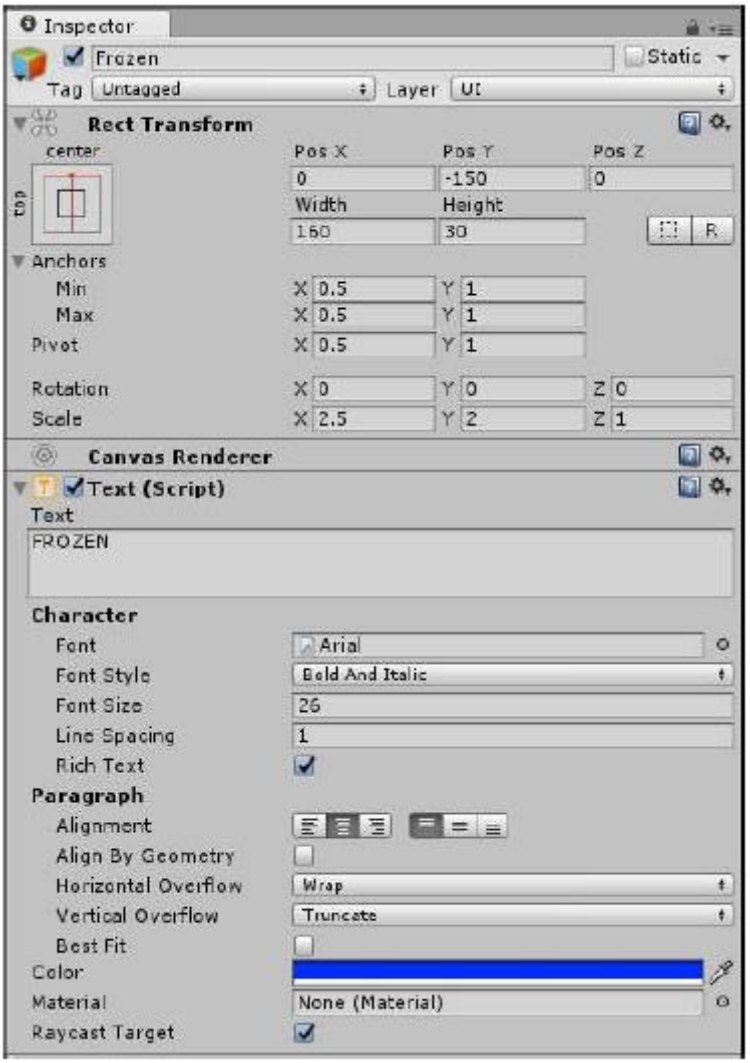
Łapanie potwora

Doszliśmy teraz do kulminacji naszej sceny: złapania potwora. Lodowe kule mogą uderzyć potwora, które eksplodują przy uderzeniu, ale potworowi nic się nie dzieje. Jeśli sobie przypominasz, gracz rzuca kulami lodu w potwora, aby je zamrozić. Następnie dodamy skrypt, który spowolni naszego potwora z

każdym uderzeniem kuli lodu, a gdy potwór zostanie trafiony wystarczająco dużo razy, zamarznie. Wykonaj następujące czynności, aby dodać i przejrzeć skrypt:

1. Z menu wybierz GameObject | interfejs użytkownika | Płótno. Spowoduje to dodanie do sceny nowych obiektów Canvas i EventSystem. Usuń EventSystem i zmień nazwę CanvasCaught_UI. Zmień obiekt Caught_UI na obiekt CatchScene.

2. Wybierz obiekt Caught_UI w oknie Hierarchy iz menu wybierz GameObject | interfejs użytkownika | Tekst. Zmień nazwę nowego obiektu Tekst zamrożony i ustaw parametry komponentu Przekształcenie prostokątne i Tekst w oknie Inspektora tak, aby odpowiadały poniższemu zrzutowi ekranu:

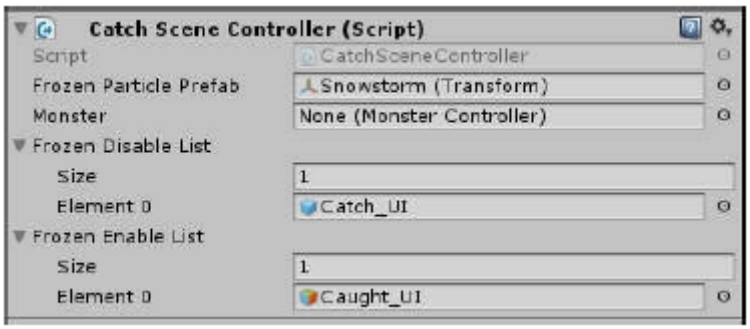


3. Wybierz obiekt Caught_UI w oknie Hierarchy i wyłącz go, odznaczając pole wyboru obok nazwy obiektu w oknie Inspektora. Tekst FROZEN stanie się niewidoczny w Oknie gry.

4. Wybierz obiekt CatchScene w oknie Hierarchia. Następnie z folderu Assets/FoodyGo/Scripts/Controllers w oknie projektu przeciągnij skrypt CatchSceneController i upuść go na obiekt CatchScene.

5. Z folderu Assets/Elementals/Prefabs(Mobile)/Ice przeciągnij prefabrykat cząsteczkowy Snowstorm do slotu Frozen Particle Prefab komponentu Catch Scene Controller w oknie Inspektora.

6. Będąc nadal w komponencie Catch Scene Controller, rozwiń pola Frozen Enable/Disable List, wprowadzając 1 w polu Size. Następnie z okna Hierarchy przeciągnij obiekt Catch_UI do Frozen Disable List, a Caught_UI do Frozen_Enable_List:



7. Wybierz CatchMonster w oknie Hierarchia. Z folderu Assets/FoodyGo/Scripts/Controllers przeciągnij skrypt MonsterController na obiekt CatchMonster w oknach hierarchii lub inspektora. Tutaj nie ma nic do skonfigurowania; wystarczy dodać skrypt.

8. Mając wciąż zaznaczony CatchMonster w oknie Inspektora, zlokalizuj komponent Collision Reaction, prawdopodobnie tuż nad właśnie dodanym skryptem. Kliknij przycisk + pod polem Collision Event, aby dodać nowy detektor zdarzeń. Z okna Hierarchia przeciągnij obiekt CatchScene na miejsce Brak (obiekt) w nowym zdarzeniu. Następnie kliknij menu rozwijane oznaczone Brak funkcji, aby otworzyć menu kontekstowe i wybierz Catch Scene Controller | Przy trafieniu potwora.

9. Właśnie okablowaliśmy wydarzenie kolizji | Procedura obsługi OnMonsterHit bez pisania kodu. Dzięki temu opracowany przez nas kod jest jeszcze bardziej rozszerzalny i wydajny. Teraz, jeśli chcemy zmienić zasady i zachowanie gry, wystarczy zmodyfikować skrypt CatchSceneController.

10. Naciśnij Play i uruchom grę z edytora. Rzucaj kulami w potwora i zauważ, że gdy każda kula uderza potwora, on zwalnia. W końcu, gdy wystarczająca liczba kulek trafi w potwora, zostanie on zamrożony na stałe, pojawi się tekst FROZEN, a cząstki śnieżnej burzy zapewnią chłodną atmosferę. Okno gry będzie wyglądać jak na poniższym zrzucie ekranu:

11. Jak zwykle, nie zapomnij przetestować na urządzeniu mobilnym, aby upewnić się, że wszystko działa tak samo. Fantastycznie, w końcu nasz potwór został złapany. Zanim zakończymy tę długą podróż, przejrzyjmy ostatni fragment kodu, który łączy to wszystko razem, CatchSceneController. Otwórz skrypt w wybranym edytorze i przejdź do metody OnMonsterHit, pokazanej tutaj w celach informacyjnych:

```
public void OnMonsterHit(GameObject go, Collision

kolizja)

{

monster = go.GetComponent<MonsterController>

();

jeśli (potwór != null)

{

print("Potworne uderzenie");

var animRedukcjaPrędkości =

Mathf.Sqrt(collision.relativeVelocity.magnitude) /

10;

monster.animationSpeed =

Mathf.Clamp01(monster.animationSpeed
```

```
- animSpeedReduction);

jeśli (monster.animationSpeed == 0)

{

print("Potwór ZAMROŻONY");

Wystąpienie (zamrożonych prefabrykatów cząsteczkowych);

foreach(var g w frozenDisableList)

{

g.SetActive(fałsz);

}

foreach(var g w frozenEnableList)

{

g.SetActive(prawda);

}

}

}

}
```

Skrypt jest stosunkowo prosty i powinien być łatwy do naśladowania, ale zidentyfikujemy najważniejsze elementy: Metoda zaczyna się od pobrania MonsterControllera z obiektu gry, który został uderzony w kolizję. Jeśli obiekt nie ma kontrolera MonsterController, to nie jest potworem i skrypt po prostu się kończy. Po pewnym logowaniu metoda oblicza współczynnik obrażeń (0-1) i stosuje go do szybkości animacji potwora. Nie będziemy tutaj recenzować MonsterControllera, ponieważ obecnie kontroluje tylko prędkość animacji potwora. Szybkość animacji jest ograniczona do wartości od 0 do 1. Wreszcie, jeśli szybkość animacji potwora wynosi zero, potwór zostaje zamrożony. Instancja zamrożonego ParticlePrefab jest tworzona, a następnie zamrożone listy wyłączania/włączania są przepuszczane przez pętlę, a obiekty na tych listach są włączane/wyłączane. To kończy materiał do tego rozdziału. Jeśli obawiasz się, że zostawiliśmy pewne rzeczy zawieszone, nie martw się; w następnym rozdziale będziemy kontynuować pracę nad sceną Catch, dodając możliwość przechowywania potworów, które łapie gracz.

Podsumowanie

To była wyjątkowo długa część, ale omówiliśmy dużo materiału i zasadniczo ukończyliśmy mini-grę w naszej grze. Najpierw omówiliśmy zarządzanie scenami, z ładowaniem i przechodzeniem między scenami. Menedżer gry został wprowadzony w celu koordynowania działań w grze. Następnie omówiliśmy wprowadzanie dotykowe, fizykę i zderzacze w ramach inicjowania przez gracza próby złapania potwora. Było to niezbędne do przejścia nas do stworzenia nowej sceny AR Catch. W ramach naszej integracji AR spędziliśmy trochę czasu, aby zrozumieć, jak zintegrować kamerę urządzenia z tłem sceny. Następnie dodaliśmy kulę lodową do naszej sceny i omówiliśmy, jak rzucać piłką za pomocą wprowadzania dotykowego i fizyki. Potem spędziliśmy trochę więcej czasu na omówienie zderzaczy i skryptów reakcji kolizji. Stamtąd dodaliśmy pizzazz do sceny z efektami artykułu, wyzwalanymi przez

reakcje kolizyjne. Na koniec dodaliśmy skrypt kontrolera catch scene, aby zarządzać reakcją potwora na trafienie. Ten skrypt zarządzał również obiektami sceny, gdy potwór został trafiony wystarczająco dużo razy i został zamrożony. W następnym rozdziale będziemy kontynuować od miejsca, w którym skończyliśmy ze sceną Catch. Nie chcemy opuszczać sceny i nie być w stanie najpierw przechować zdobyczy gracza. Przechowywanie zdobyczy gracza i innych przedmiotów będzie niezbędne w naszej grze mobilnej.