Rozwiązywanie problemów

Każdy programista w pewnym momencie napotka nieoczekiwany problem lub problemy, który blokuje jego postęp w dalszym rozwoju. Może to być coś tak złośliwego jak ukryty błąd składniowy lub znacznie poważniejszy. Tak czy inaczej, programista musi skorzystać z dostępnych mu narzędzi i rozwiązać problem. Tu przyjrzymy się wprowadzeniu lub zapoznaniu się z różnymi narzędziami do rozwiązywania problemów dostępnymi dla mobilnych deweloperów Unity. Następnie przyjrzymy się bardziej zaawansowanym narzędziom specjalistycznym, których można użyć, gdy problem wydaje się szczególnie trudny. Oczywiście będziemy też chcieli omówić opcje śledzenia problemów, a nawet zapobiegania im. Następnie, na końcu tego rozdziału, zostanie zamieszczona tabela pomocna w rozwiązywaniu problemów, które możesz napotkać podczas przechodzenia przez rozdziały tej książki. Oto lista głównych tematów, które omówimy:

* Okno konsoli

* Błędy i ostrzeżenia kompilatora

* Debugowanie

* Zdalne debugowanie

* Zaawansowane debugowanie

* Logowanie

* CUDLR

* Analiza jedności

* Problemy i rozwiązania według rozdziałów

Okno konsoli

Okno konsoli powinno być Twoim miejscem startowym, gdy napotkasz problem. Dostęp do niego można uzyskać z menu, wybierając Okno | Konsola, która otworzy okno konsoli, które możesz zadokować w edytorze według własnego uznania. W zależności od preferencji i doświadczenia możesz chcieć, aby konsola była zawsze widoczna. Tak czy inaczej, jak tylko coś pójdzie nie tak, z pewnością powinno to być pierwsze miejsce, które sprawdzisz. Przyjrzyjmy się szczegółowo oknu konsoli, ponieważ będzie ono kluczowe dla kilku innych elementów:

* Upewnij się, że edytor Unity jest otwarty. Jeśli okno konsoli nie jest otwarte, otwórz je z menu, wybierając Okno | Konsola.

* Spójrz na okno i zapoznaj się z przyciskami i menu kontekstowym. Poniżej znajduje się zrzut ekranu okna konsoli z typową konfiguracją:

Co powiesz na to, że przyjrzymy się, co robi każdy z przycisków i kilka pomocnych wskazówek, których możesz nie znaleźć w dokumentacji Unity:

* Wyczyść: Czyści bieżące okno ze wszystkich wiadomości. Przydatne może być wyczyszczenie dziennika podczas długich sesji testowych.

* Zwiń: spowoduje zwinięcie wszystkich tych samych komunikatów dziennika i oznaczenie etykiety numerem. Jest to szczególnie przydatne, jeśli chcesz śledzić powtarzającą się wiadomość, ale nie chcesz, aby okno było przeładowane wiadomościami.

* Clear on Play: Czyści log przy każdej nowej sesji uruchomionej w edytorze. Jest to bardzo przydatne i coś, co będziesz chciał mieć cały czas.

* Error Pause: Powoduje zatrzymanie edytora, gdy podczas gry wystąpi błąd. To kolejna świetna funkcja, która pozwala wykryć te błędy, gdy się pojawią. Niestety nie możesz wyizolować typu błędu, który chcesz śledzić.

Ikony po prawej stronie okna są opisane w następujący sposób:

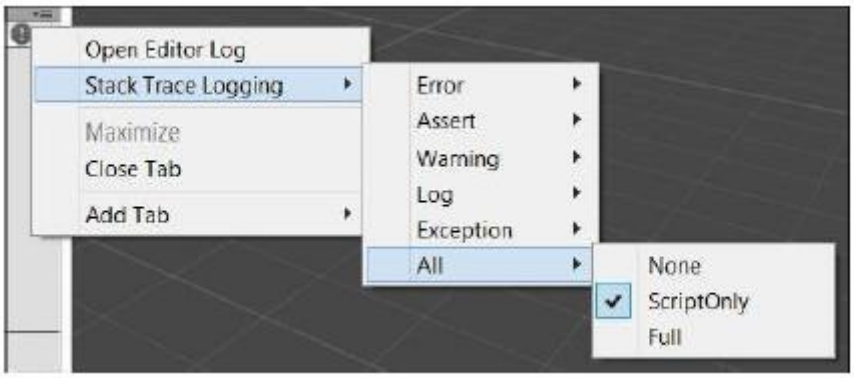
* Filtr informacji: jest to pierwszy przycisk po prawej stronie. Spowoduje to odfiltrowanie wszystkich informacji lub ogólnych komunikatów debugowania wysyłanych do konsoli. Jest to przydatny sposób na zablokowanie hałasu.

* Filtr ostrzegawczy: Jest to przycisk ze znakiem wydajności — filtr, którego można użyć do włączania/wyłączania ostrzeżeń, ponownie w celu zmniejszenia hałasu w oknie. Unikaj impulsu wyłączania ostrzeżeń; są one pomocnymi przypomnieniami, których będziesz chciał zmniejszyć.

* Filtr błędów: jest to ostatni przycisk po prawej stronie i jest filtrem do włączania/wyłączania komunikatów o błędach. Ogólnie rzecz biorąc, dobrze jest zawsze włączać ten filtr. Czasami jednak przydatne może być zignorowanie hałaśliwych błędów, które wypełniają okno, aby śledzić brakujący komunikat debugowania.

Menu kontekstowe jest wyjaśnione w następujący sposób:

1. Kliknij ikonę menu kontekstowego okna Konsola, znajdującą się w prawym górnym rogu okna. Spowoduje to otwarcie menu kontekstowego, jak pokazano:



2. Aby być dokładnym, wyjaśnijmy elementy menu dziennika w następujący sposób:

* Otwórz dziennik edytora: otworzy szczegółowy dziennik edytora. Przyjrzymy się temu logowi bardziej szczegółowo, gdy przejdziemy do sekcji o logowaniu.

* Rejestrowanie śladu stosu: Ustawia ilość śladu stosu, który ma być uwzględniony w dzienniku konsoli. Generalnie ustawisz to tylko na ScriptOnly, jak pokazano na fragmencie ekranu. Ponownie omówimy tę opcję bardziej szczegółowo w sekcjach dotyczących debugowania i rejestrowania.

Dlatego następnym razem, gdy coś pójdzie nie tak, zacznij od otwarcia konsoli. W następnej sekcji przyjrzymy się zestawowi komunikatów kompilatora wysyłanych do konsoli.

Błędy i ostrzeżenia kompilatora

Komunikaty kompilatora wypełnią okno konsoli za każdym razem, gdy skrypty gry projektu zostaną ponownie skompilowane, co może się zdarzyć w wyniku importu zasobów lub zmiany skryptu. Poważniejsze błędy kompilatora zablokują możliwość uruchomienia projektu w edytorze. Ostrzeżenia są łagodniejsze i mniej krytyczne, ale równie ważne jest obserwowanie, kiedy się pojawiają. Poniższa lista zawiera niektóre z najczęstszych błędów i ostrzeżeń, które możesz napotkać:

* Błędy kompilatora: będą wyświetlane jako czerwony tekst na pasku stanu lub z ikoną błędu w konsoli:

- Błąd składni: jest to prawdopodobnie najczęstszy błąd, jaki napotkasz. Kliknij dwukrotnie problem, aby otworzyć skrypt do punktu błędu w edytorze. Po prostu edytuj skrypt, aby naprawić problem ze składnią.

– Brakujący skrypt: jest to nieprzyjemny problem, który może wystąpić podczas importowania zasobów — skrypt mógł zostać przeniesiony lub mógł wystąpić konflikt nazw. Brakujący lub uszkodzony skrypt zostanie usunięty jako komponent z dowolnego obiektu gry. Możesz rozwiązać ten problem, ponownie importując uszkodzone zasoby lub zarządzając konfliktem nazw.

- Wewnętrzny błąd kompilatora: to kolejny paskudny błąd, który może być trudny do zdiagnozowania. Jest to bardziej powszechne, jeśli używasz wtyczek, ale może również wystąpić, jeśli zmienisz sygnatury metod. Spróbuj określić, gdzie występuje problem i sprawdź, w jaki sposób korzystasz z metod lub parametrów.

* Ostrzeżenia kompilatora: pojawiają się jako żółtawy tekst na pasku stanu lub z ikoną ostrzeżenia w konsoli. Kliknij dwukrotnie dowolne ostrzeżenie, aby przejść do obraźliwego kodu w wybranym edytorze:

- Przestarzały kod: Unity oznaczy kod, który używa właściwości lub metod, które zostały przestarzałe. Może to być częsty problem, jeśli używasz starszych zasobów, które nie zostały wydane dla Twojej wersji Unity. Aby usunąć ostrzeżenie, musisz zaktualizować kod, aby korzystać z nowej metody.

- Niespójne zakończenia wierszy: to irytujące ostrzeżenie, które może pojawić się podczas przełączania edytorów lub importowania kodu. Napraw problem w edytorze kodu, ustawiając spójne zakończenia wierszy, przechodząc do MonoDevelop: Project | Opcje rozwiązania | Kod źródłowy | Formatowanie kodu, Visual Studio: plik | Zaawansowane ustawienia.

- Ogólne ostrzeżenia: To jest jak nieużywane pola lub zmienne, nie krytyczne, ale coś, co należy wyczyścić, gdy gra lub skrypty będą gotowe do wydania.

Uwaga : Istnieje sposób na przekształcenie ostrzeżeń kompilatora w błędy, a tym samym wymuszenie naprawienia wszystkich ostrzeżeń. Chociaż może to być preferowane przez niektórych programistów, nie jest to zalecane jako najlepsza praktyka.

Dobrym nawykiem jest wyczyszczenie okna konsoli przed edycją skryptu lub importem zasobów. W ten sposób łatwiej jest śledzić i filtrować problemy z kompilatorem. Po naprawieniu tych problemów z kompilatorem nadszedł czas na wyśledzenie wszelkich błędów lub ostrzeżeń w czasie wykonywania, które omówimy w następnej sekcji.

Debugowanie

Unity zapewnia świetny interfejs w edytorze do oglądania i edytowania stanu obiektów i komponentów gry, gdy gra jest uruchomiona. Chociaż prawdopodobnie w ten sposób możesz debugować większość swojej gry, nadal możesz napotkać sytuacje, w których logika w skrypcie wymaga dokładniejszego spojrzenia. Na szczęście Unity zapewnia również doskonały zestaw narzędzi, które umożliwiają debugowanie skryptów w edytorze, gdy gra jest uruchomiona. Spójrzmy, jak możesz rozpocząć sesję debugowania:

1. Otwórz Unity z pustym projektem.

Uwaga : w przypadku korzystania z programu Visual Studio w tym ćwiczeniu założono, że zainstalowano już rozszerzenie narzędzi i skonfigurowano preferencje edytora.

2. Z menu wybierz Zasoby | Importuj pakiet | Pakiet niestandardowy... i przejdź do folderu C10A w pobranym kodzie źródłowym książki. Wybierz Chapter10.unitypackage i kliknij Otwórz.

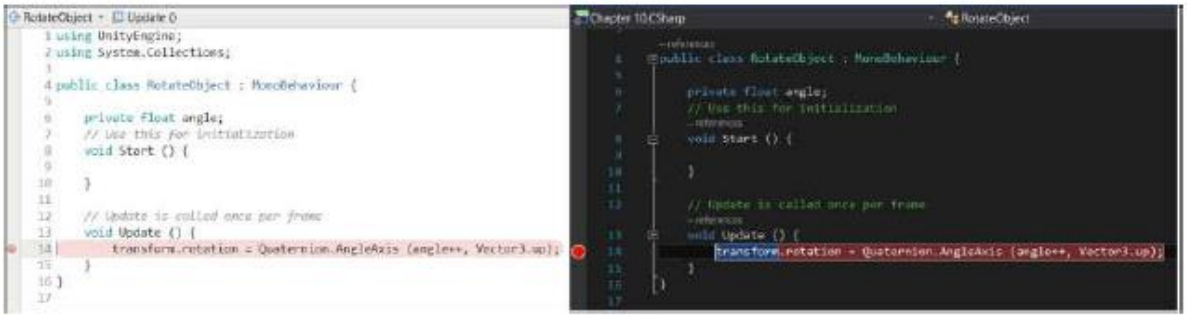
3. Paczka jest mała, więc powinna się szybko zaimportować. Kliknij przycisk Importuj w oknie dialogowym Importuj pakiet Unity, aby kontynuować.

4. Zlokalizuj scenę główną w folderze Zasoby/Rozdział 10/Sceny w oknie Projekt i kliknij dwukrotnie scenę, aby ją otworzyć.

5. Zlokalizuj skrypt RotateObject w folderze Assets/Chapter 10/Scripts w oknie projektu i kliknij dwukrotnie skrypt, aby otworzyć go w wybranym edytorze.

Uwaga : zademonstrujemy użycie debugera dla MonoDevelop i Visual Studio. Jeśli używasz innego edytora kodu, proces może być inny.

6. Ustaw punkt przerwania w jednym wierszu kodu w metodzie Update, jak pokazano na poniższym zrzucie ekranu:



7. Kolejny krok będzie zależał od używanego edytora. Postępuj zgodnie ze wskazówkami dla wybranego edytora, aby rozpocząć debugowanie:

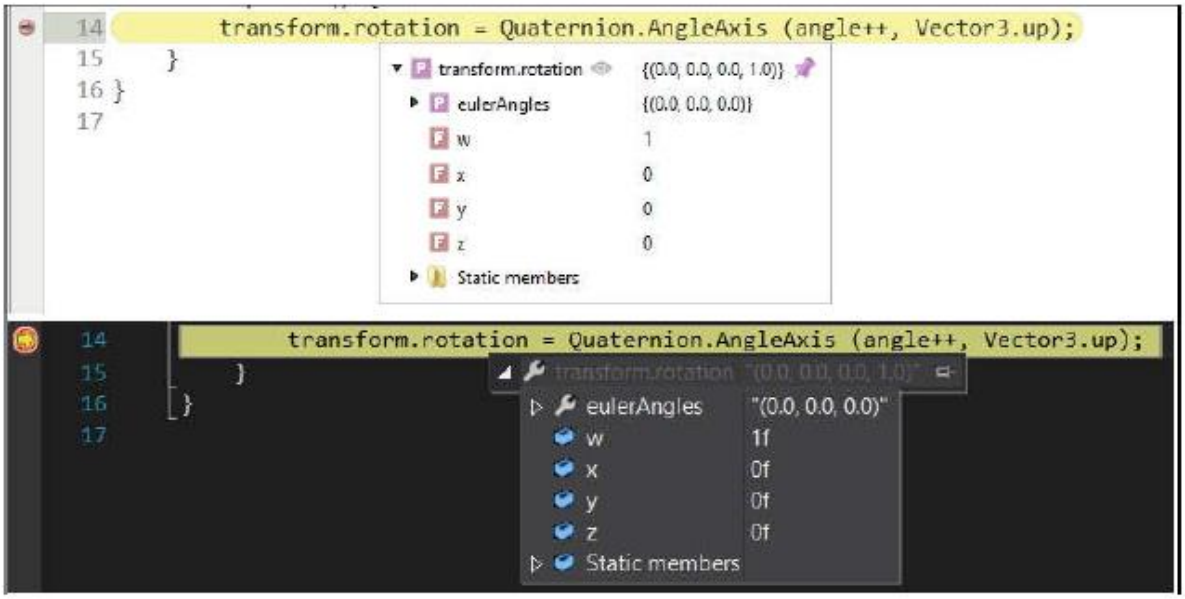
* MonoDevelop: Na pasku narzędzi naciśnij przycisk Odtwórz, aby rozpocząć debugowanie. Wyświetlone zostanie okno dialogowe Dołącz do procesu; teraz wybierz proces Unity, w którym Twój projekt jest załadowany i kliknij Dołącz.

Wskazówka: jeśli masz uruchomionych wiele instancji edytora Unity, znalezienie odpowiedniego procesu do dołączenia może wymagać trochę prób i błędów. Zwróć szczególną uwagę na identyfikator procesu po dołączeniu do prawidłowego procesu. Oczywiście alternatywą jest uruchomienie tylko jednej instancji Unity.

* Visual Studio: na pasku narzędzi naciśnij przycisk Odtwórz (Dołącz do Unity), aby rozpocząć debugowanie. Program Visual Studio jest wystarczająco inteligentny, aby samodzielnie dołączyć do edytora Unity.

8. Wróć do Unity i uruchom projekt, naciskając przycisk Odtwórz.

9. W bardzo krótkim czasie zostaniesz przeniesiony do edytora skryptów, a ustawiony wcześniej punkt przerwania zostanie podświetlony. W tym momencie użyj myszy, aby najechać kursorem na tekst i sprawdź właściwości zmiennych, jak pokazano na poniższym zrzucie ekranu:



Istnieje wiele innych opcji debugowania, które możesz zastosować, takich jak zegarki, ale poprzednie ćwiczenie pozwoli Ci zacząć. Oczywiście zdarzają się sytuacje, gdy wdrażasz na platformie mobilnej, gdzie chcesz debugować bezpośrednio na urządzeniu; omówimy to w następnej sekcji.

Zdalne debugowanie

Możliwość debugowania skryptów projektu w edytorze to świetna funkcja. Oczywiście możliwość debugowania skryptu podczas wdrażania na platformę docelową jest jak Święty Graal dla programistów. Już nie masz do czynienia z niewiadomymi dotyczącymi działania twojego skryptu/kodu na urządzeniu, kiedy możesz faktycznie oglądać uruchamiany kod za pomocą zdalnego debugowania. Zdalne debugowanie to potężna funkcja, która istnieje od jakiegoś czasu, ale ma ograniczenia i może powodować własne problemy z łącznością. Przed próbą zdalnego debugowania aplikacji upewnij się, że możesz ją bezproblemowo wdrożyć na urządzeniu mobilnym. Jeśli masz problemy z wdrożeniem projektu na urządzeniu, ta sekcja nie pomoże. Zamiast tego zapoznaj się z sekcją Problemy i rozwiązania według rozdziałów na końcu tego rozdziału. Postępuj zgodnie z podanymi instrukcjami, aby skonfigurować zdalne debugowanie w edytorze:

1. Aby rozpocząć, użyj projektu z poprzedniego ćwiczenia debugowania. Jeśli skoczyłeś tutaj, wykonaj kroki 1-4, aby otworzyć projekt i ustawić scenę.

2. Z menu wybierz Plik | Ustawienia kompilacji. Gdy otworzy się okno dialogowe Ustawienia kompilacji, kliknij przycisk Dodaj otwarte sceny, aby dodać scenę główną do kompilacji. Następnie ustaw wybraną platformę wdrażania i zaznacz pole wyboru Development Build, jak pokazano na poniższym zrzucie ekranu:

3. W tym momencie może być konieczne ustawienie dodatkowych ustawień odtwarzacza w zależności od konkretnej platformy. Kliknij przycisk Ustawienia odtwarzacza..., aby otworzyć panel Ustawienia odtwarzacza w oknie Inspektora. Następnie ustaw wymagane ustawienia dla wybranej platformy, iOS lub Android.

4. Następny krok będzie zależał od platformy, na której wdrażasz swoją grę:

* iOS:

- Upewnij się, że Twoje urządzenie mobilne jest podłączone do tej samej sieci (Wi-Fi), co komputer programisty.

- Po wdrożeniu gry możesz odłączyć kabel USB.

* Android:

- Otwórz monit terminala/CMD i przejdź do folderu Android SDK/platform-tools

- Wpisz następujące polecenie:

adb tcpip 5555

- Powinno to spowodować wyświetlenie następującego komunikatu:

ponowne uruchomienie w TCP

port trybu: 5555

- Otwórz urządzenie i znajdź adres IP, otwierając Ustawienia | O | Status Zapisz lub zapamiętaj adres IP, a następnie wprowadź polecenie, zastępując adres IP urządzenia za pomocą następującego polecenia:

połączenie adb

ADRESIPURZĄDZENIA

- Powinno to odpowiedzieć następującym komunikatem, zastąpionym Twoim adresem IP:

połączony z

ADRES IP URZĄDZENIA: 5555

- Potwierdź, że urządzenie jest zainstalowane, uruchamiając następujące polecenie:

urządzenia adb

- Powinno to wyglądać następująco (zwróć uwagę, że możesz zobaczyć wpis dla swojego urządzenia według nazwy):

Lista załączonych urządzeń

ADRES IP URZĄDZENIA: 5555 urządzenie

5. Kliknij przycisk Zbuduj i uruchom w oknie dialogowym i zapisz plik zgodnie z nazwą ustawioną w identyfikatorze pakietu.

6. Po załadowaniu i uruchomieniu prostej wersji demonstracyjnej na urządzeniu odłącz kabel USB. Następnie wróć do edytora skryptów. Skorzystaj z instrukcji dla swojego edytora, które znajdziesz w poniższej tabeli:

* MonoDevelop:

* Naciśnij przycisk Odtwórz lub F5, aby rozpocząć debugowanie

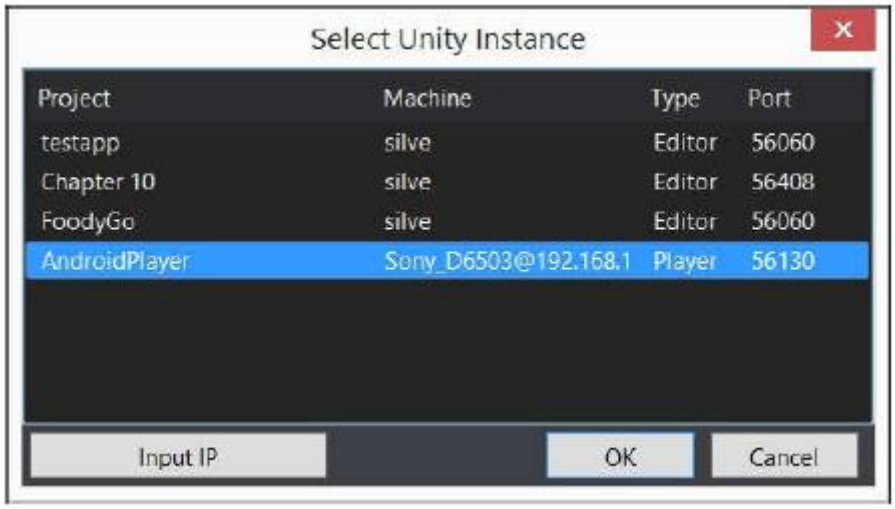
* Wybierz wpis, który pasuje do Twojej platformy i urządzenia, a następnie kliknij przycisk Dołącz

Uwaga : gra na Twoim urządzeniu zostanie zatrzymana po osiągnięciu punktu przerwania i możesz ją debugować tak samo, jak lokalnie.

* Visual Studio (2015):

* Z menu wybierz Debuguj | Dołącz debuger Unity

* Zostanie wyświetlone okno dialogowe Select Unity Instance zawierające listę instancji, z którymi możesz się połączyć, pokazane w następujący sposób:



* Wybierz instancję pasującą do Twojego urządzenia (Type=Player) i kliknij OK

* Możesz teraz używać debugera tak, jakbyś miał lokalnie

7. Pamiętaj, że podczas zdalnego debugowania może być konieczne poczekanie na pewne działania, takie jak inspekcja zmiennych, więc bądź cierpliwy. Dzieje się tak, ponieważ edytor musi pobrać stan ze zdalnego urządzenia, a nie lokalnie.

8. Po zakończeniu debugowania kliknij przycisk Zatrzymaj w MonoDevelop lub Visual Studio. Powinno to usunąć debuger i pozwolić grze kontynuować działanie. Czasami, jeśli edytor (zazwyczaj MonoDevelop) zostanie zablokowany, musisz go zamknąć.

Zdalne debugowanie to świetna opcja do debugowania kodu podczas programowania, ale w niektórych sytuacjach możesz potrzebować kilku bardziej zaawansowanych narzędzi do swojej dyspozycji. W następnej sekcji omówimy kilka zaawansowanych narzędzi do debugowania aplikacji mobilnych.

Zaawansowane debugowanie

Bez względu na to, ile czasu spędzasz w edytorze Unity, zawsze pomocne jest posiadanie bardziej zaawansowanych opcji debugowania, zwłaszcza tych, które można uruchomić całkowicie odłączony od komputera deweloperskiego. Chociaż debugowanie jest z pewnością narzędziem, którego można użyć do rozwiązywania problemów, nie jest to coś, co chcesz robić przez cały czas. Lepszym sposobem na

zrozumienie, jak działa Twoja gra, jest dodanie funkcji rejestrowania, które omówimy w następnej sekcji.

Logowanie

Jeśli omówiłeś już kilka rozdziałów tej książki, możesz docenić, jak cenne jest rejestrowanie, aby Twoja gra działała zgodnie z oczekiwaniami. W Unity wszystkie komunikaty dziennika są wyprowadzane do konsoli, chyba że utworzysz niestandardowy rejestrator, który zapisuje do pliku lub usługi. W dalszej części tej sekcji utworzymy niestandardowy rejestrator. Na razie spójrzmy na opcje logowania, które Unity zapewnia nam poza pudełko, jak wyszczególniono:

* print: Jest to skrócony odpowiednik Debug.log.

* Debug.Log, Debug.LogFormat: wyprowadza standardowy komunikat informacyjny w postaci niesformatowanego lub sformatowanego tekstu. Komunikaty są wyświetlane z ikoną informacji w oknie konsoli.

* Debug.LogError, Debug.LogErrorFormat: wyświetla niesformatowane i sformatowane komunikaty o błędach. Komunikaty są wyświetlane z ikoną błędu w oknie konsoli.

* Debug.LogException: wypisuje wyjątek do konsoli

z ikoną błędu.

* Debug.LogWarning, Debug.LogWarningFormat : wyświetla niesformatowane i sformatowane komunikaty ostrzegawcze. Komunikaty są wyświetlane z ikoną ostrzeżenia w oknie konsoli.

* Debug.LogAssertion, Debug.LogFormatAssertion: to dane wyjściowe

niesformatowane i sformatowane komunikaty potwierdzenia testu.

Aby korzystać z tych możliwości rejestrowania, należy dodać te instrukcje do wejścia, wyjścia lub innych punktów w skrypcie. Poniżej znajduje się przykład skryptu Unity, który pokazuje, w jaki sposób można użyć każdego z tych typów rejestrowania:

```
using UnityEngine;

using System.Collections;

using System;

public class LoggingExample : MonoBehaviour {

public GameObject target;

public float iterations = 1000;

i t fl t t t

private float start;

// Use this for initialization

void Start () {

Debug.Log("Start");

if (target == null)
```

```
{
Debug.LogWarning("target object not
set");
}
if (iterations < 1)
{
Debug.LogWarningFormat("interations:
{0} < 1", iterations);
}
Debug.LogFormat("{0} iterations set",
iterations);
start = iterations;
}
// Update is called once per frame
void Update () {
//try/catch used for demo
//never use in an update method
try
{
iterations--;
Debug.LogFormat("Progress {0}%", (100
- iterations / start * 100));
}
catch (Exception ex)
{
Debug.LogError("Error encountered " +
ex.Message);
Debug.LogErrorFormat("Error at {0}
iterations, msg = {1}", iterations, ex.Message);
Debug.LogException(ex);
}
```

```
}
}
```

Klasa LoggingExample jest przykładem użycia różnych

rodzaje logowania dostępne w Unity. Co się stanie w klasie LoggingExample, jeśli początkowe iteracje zostaną ustawione na 0? Jaką zmianę możesz wprowadzić, aby próbka zgłosiła wyjątek? Oczywiście w większości przypadków wyprowadzanie komunikatów dziennika do okna konsoli będzie w porządku, zwłaszcza w fazie rozwoju. Jednak w innych sytuacjach, po wdrożeniu gry, w celach testowych lub komercyjnych, nadal możesz chcieć śledzić te wiadomości. Na szczęście istnieje bardzo łatwa możliwość obsługi niestandardowego wyjścia dziennika, jak pokazano w następującej klasie:

```
using System;
using System.IO;
using UnityEngine;
public class CustomLogHandler : MonoBehaviour
{
public string logFile = "log.txt";
private string rootDirectory =
@"Assets/StreamingAssets";
private string filepath;
void Awake()
{
Application.logMessageReceived +=
Application_logMessageReceived;
#if UNITY_EDITOR
filepath = string.Format(rootDirectory +
@"/{0}", logFile);
if(Directory.Exists(rootDirectory)==false)
{
Directory.CreateDirectory(rootDirectory);
}
#else
// check if file exists in
Application.persistentDataPath
filepath = string.Format("{0}/{1}",
```

```
Application.persistentDataPath, logFile);
#endif
}
private void
Application_logMessageReceived(string condition,
string stackTrace, LogType type)
{
var level = type.ToString();
var time =
DateTime.Now.ToShortTimeString();
var newLine = Environment.NewLine;
var log = string.Format("{0}:[{1}]:{2}{3}"
, level,time, condition, newLine);
try
{
File.AppendAllText(filepath, log);
}
catch (Exception ex)
{
var msg = ex.Message;
}
}
}
```

CustomLogHandler działa, dołączając się do zdarzenia Application.logMessageReceived w metodzie Awake. To zdarzenie jest wywoływane za każdym razem, gdy zawartość jest rejestrowana w Unity. Reszta klasy dotyczy skonfigurowania poprawnej ścieżki do pliku, tworzenia folderów w razie potrzeby, a następnie formatowania danych wyjściowych, gdzie rzeczywiste rejestrowanie odbywa się w metodzie Application_logMessageReceived. Chociaż ta klasa może nie być dobrze dostosowana do platform mobilnych, jest idealna do śledzenia komunikatów dziennika uruchamiania w edytorze lub na wdrożeniu platformy komputerowej. W następnych kilku sekcjach przyjrzymy się, jak ta obsługa dzienników może być używana do debugowania i/lub zwalniania wdrożeń.

CUDLR

Jeśli omówiłeś już kilka rozdziałów tej książki, być może znasz CUDLR, który jest doskonałym narzędziem zdalnego logowania, debugowania i inspekcji. Dla tych z was, którzy przegapili konfigurację

CUDLR w rozdziale 2, Mapowanie lokalizacji gracza, nie martwcie się, ponieważ dokonamy przeglądu konfiguracji tutaj. Oczywiście, jeśli jesteś tutaj z powodu problemów z CUDLR, zapoznaj się z ostatnią sekcją tego rozdziału, Problemy i rozwiązania według rozdziałów. CUDLR to zdalna konsola, która działa przez wewnętrzny serwer WWW utworzony w twojej grze. Wykorzystuje tę samą technikę, której używaliśmy do wyprowadzania danych wyjściowych dziennika, ale zapewnia również kontrolę obiektów, a nawet dostosowywanie. Skorzystaj z poniższych instrukcji, aby skonfigurować CUDLR (jeśli już to zrobiłeś, możesz po prostu przejrzeć tę sekcję):

1. Jeśli jeszcze nie skonfigurowałeś projektu, otwórz nowy projekt Unity i zaimportuj Chapter10.unitypackage.

2. Otwórz główną scenę z folderu Zasoby/Rozdział 10/Sceny.

3. Z menu wybierz GameObject | Utwórz puste. Zmień nazwę obiektu CUDLR i zresetuj jego transformację do zera.

4. Przeciągnij skrypt serwera z folderu Assets/CUDLR/Scripts w oknie projektu na nowy obiekt CUDLR GameObject.

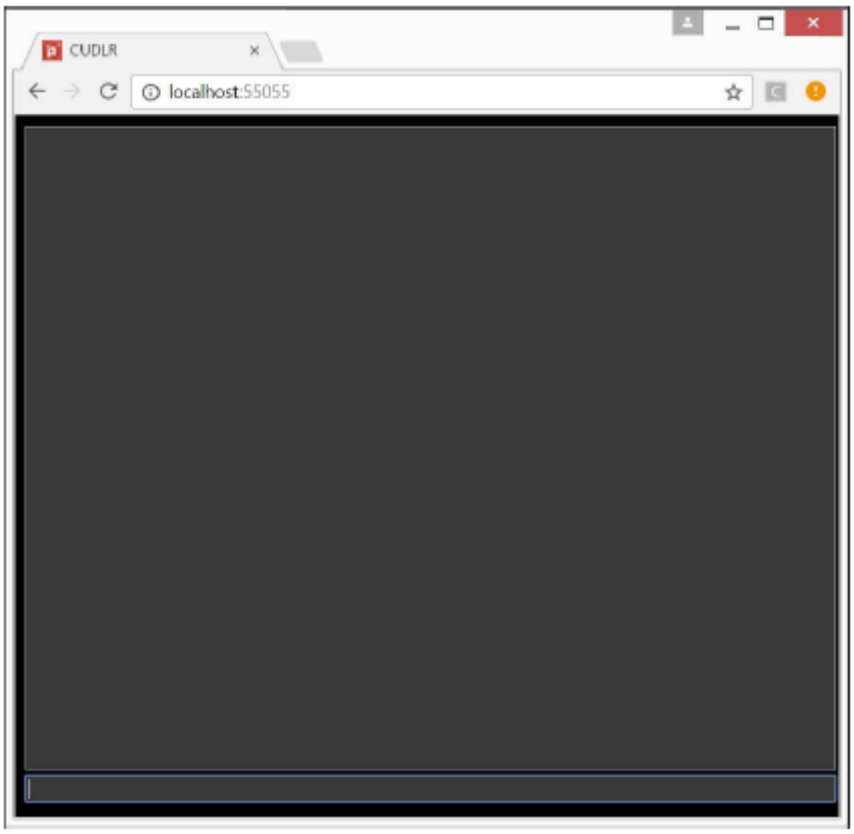
5. Domyślnie CUDLR uruchamia się na porcie 55055. To ustawienie można zmienić, sprawdzając GameObject CUDLR w oknie Inspektora. Na razie jednak pozostaw to na wartości domyślnej.

6. Naciśnij Play w edytorze, aby uruchomić scenę. Utrzymuj scenę w ruchu.

7. Otwórz wybraną przeglądarkę i wprowadź następujący adres URL:

http://host lokalny:550555

8. W przeglądarce otworzy się okno CUDLR, jak pokazano na następującym zrzucie ekranu:



9. W dolnej części okna wpisz następujące polecenie:

help

10. Zapewni to następujące wyniki:

Commands:

object list : lists all the game objects in

the scene

object print : lists properties of the object

clear : clears console output

help : prints commands

11. Wpisz następującą komendę:

object list

12. Daje to następujący wynik:

CUDLR

Directional Light

Cube

Main Camera

13. Wpisz następującą komendę:

object print Cube

14. Daje to następujący wynik:

Game Object : Cube

Component : UnityEngine.Transform

Component : UnityEngine.MeshFilter

Component : UnityEngine.BoxCollider

Component : UnityEngine.MeshRenderer

Component : RotateObject

Jak pokazaliśmy w kilku sekcjach, CUDLR jest również przydatny do rejestrowania aktywności rejestrowania w grze uruchomionej na urządzeniu mobilnym. Dzięki CUDLR możesz jednocześnie śledzić dane wyjściowe logowania na kilku urządzeniach jednocześnie. Oczywiście CUDLR jest przeznaczony tylko do przechwytywania dzienników podczas debugowania lub testowania. Wewnętrzny serwer sieciowy, który tworzy CUDLR, nie jest czymś, co chcielibyśmy dołączyć do naszej gry z wielu powodów. Co jeśli chcemy śledzić logi błędów krytycznych lub wyjątków po wydaniu naszej gry? Na szczęście w Unity dostępnych jest wiele opcji, a jedną z nich przyjrzymy się w następnej sekcji.

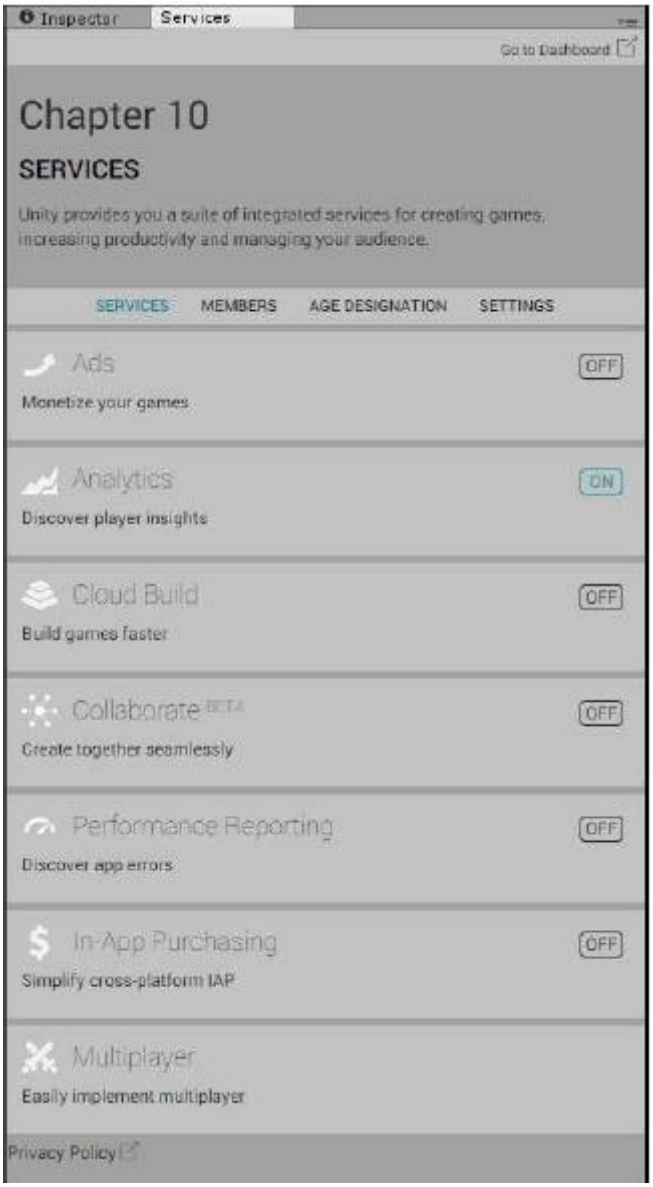
Analiza jedności

Po wyjęciu z pudełka Unity Analytics to coś, co powinieneś mieć za każdym razem, gdy wydajesz grę. Niezbędne jest przekazanie opinii na temat graczy, dystrybucji gry i wielu innych wskaźników. W tej sekcji omówimy niektóre z podstaw Unity Analytics, a następnie zagłębimy się w korzystanie z narzędzia do śledzenia krytycznych błędów i wyjątków. Dostęp do tych informacji zapewni lepszą obsługę gry podczas testów poza witryną lub wydania. Postępuj zgodnie z podanymi instrukcjami, aby włączyć Unity Analytics w swoim projekcie:

1. Z menu wybierz Okno | Usługi. Otworzy się okno Usługi, zwykle nad oknem Inspektora.

Wskazówka : Raportowanie skuteczności będzie również zgłaszać błędy, ale wymaga to uaktualnionego konta Unity.

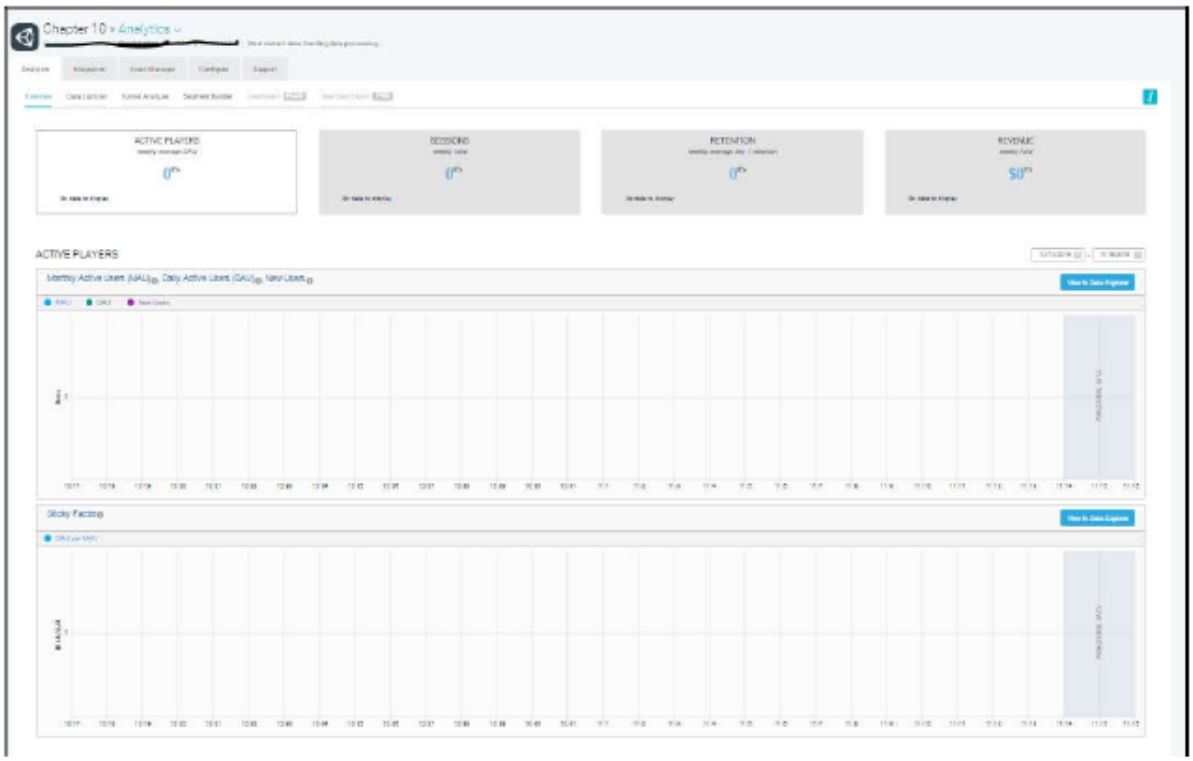
2. Znajdź grupę Analytics na liście i ustaw przełącznik po prawej stronie w pozycji On., jak pokazano na poniższym zrzucie ekranu:



3. Kliknij panel Analityka w oknie Usługi. Możesz zostać poproszony o potwierdzenie wytycznych dotyczących wieku Twojej gry. Jeśli tak, po prostu wybierz wiek powyżej 13 lat i kontynuuj.

4. Strona Analytics zostanie załadowana w oknie Usługi. Kliknij przycisk Przejdź do pulpitu nawigacyjnego tuż pod głównym tekstem. Spowoduje to otwarcie domyślnej przeglądarki i przejście do witryny Unity Analytics. Być może będziesz musiał zalogować się na swoje konto Unity, po prostu zrób to, gdy zostaniesz o to poproszony.

5. Po załadowaniu strony powinieneś zobaczyć coś zbliżonego do następującego zrzutu ekranu:



W tej chwili Twoi gracze, sesje i inne wskaźniki prawdopodobnie będą miały wartość 0. Jest to oczekiwane, ponieważ prawdopodobnie właśnie włączyłeś analitykę w swoim projekcie. Teraz sam Unity Analytics nie będzie śledzić komunikatów rejestrowania, ale możemy użyć niektórych niestandardowych mechanizmów zdarzeń do śledzenia błędów lub dzienników wyjątków. Na szczęście cała praca nad tym została zamknięta w skrypcie o nazwie AnalyticsLogHandler.

Uwaga : Unity Analytics nie działa w czasie rzeczywistym, co oznacza, że na wyświetlenie wyników zazwyczaj trzeba poczekać od 12 do 14 godzin. Chociaż sprawia to, że jest on zasadniczo bezużyteczny do debugowania, może być przydatnym wskaźnikiem, gdy zostanie wdrożony dla graczy na całym świecie.

Postępuj zgodnie z podanymi wskazówkami, aby skonfigurować ten skrypt:

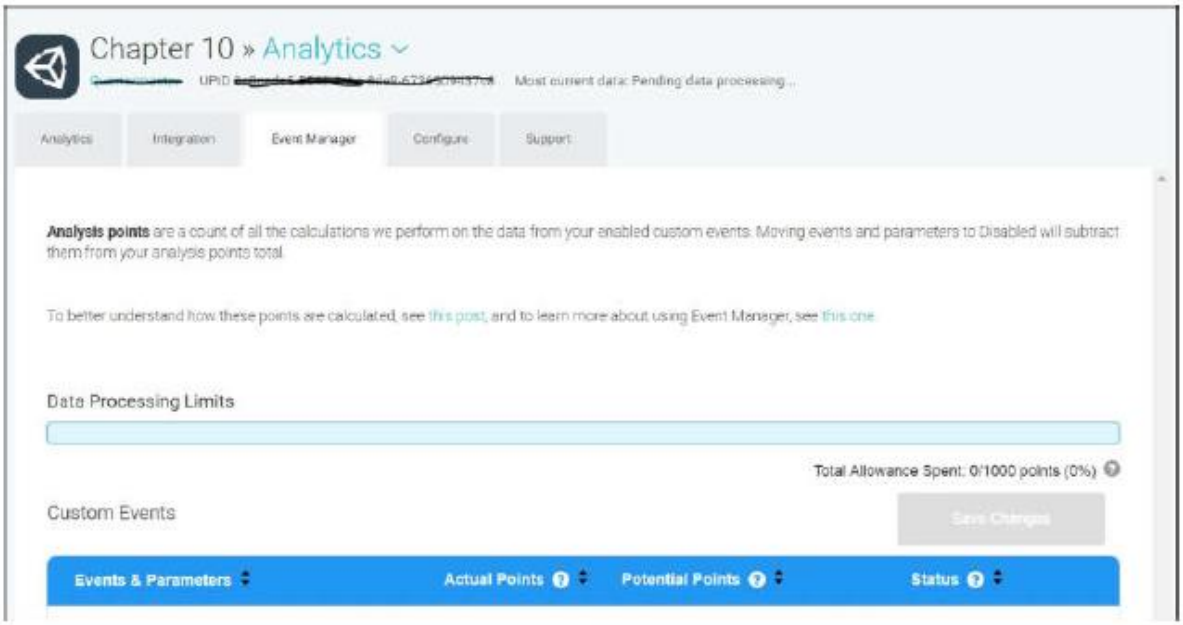
1. Z menu wybierz GameObject | Utwórz puste. Zmień nazwę obiektu na AnalyticsLogHandler i zresetuj transformację do zera.

2. Przeciągnij skrypt AnalyticsLogHandler z folderu Assets/Chapter 10/Scripts w oknie Project na obiekt AnalyticsLogHandler.

3. Skrypt RotateObject, który jest dołączony do kostki w przykładowej scenie Main, rejestruje komunikat o błędzie za każdym razem, gdy obiekt kończy cały obrót.

4. Naciśnij Play, aby uruchomić scenę w edytorze. Poczekaj, aż kilka z tych komunikatów o błędach obracania zostanie zalogowanych na pasku stanu, w oknie konsoli lub oknie CUDLR. Niestety, będziesz musiał poczekać dodatkowe 12-14 godzin, zanim komunikat pojawi się na stronie pulpitu nawigacyjnego Unity Analytics.

5. Po 14 godzinach wróć do pulpitu analitycznego i przejrzyj komunikaty przeglądarki zdarzeń, wybierając kartę Menedżer zdarzeń, jak pokazano na poniższym zrzucie ekranu:



Przyjrzyjmy się skryptowi AnalyticsLogHandler, jak pokazano na ilustracji:

```
using UnityEngine;
using UnityEngine.Analytics;
public class AnlyticsLogHandler : MonoBehaviour
{
public LogType logLevel = LogType.Error;
void Awake()
{
Application.logMessageReceived +=
Application_logMessageReceived;
}
private void
Application_logMessageReceived(string condition,
string stackTrace, LogType type)
{
```

```
if (type == logLevel)
{
Analytics.CustomEvent("LOG", new
Dictionary<string, object>
{
{ "msg", condition },
{ "type", type.ToString() }
});
}
}
}
```

Ten skrypt jest prostszy, ale implementacja jest podobna do skryptu CustomLogHandler, który omówiliśmy wcześniej. Istnieje jedna główna różnica, a mianowicie użycie obiektu Analytics do utworzenia zdarzenia niestandardowego, które następnie zostanie automatycznie wysłane do Unity Analytics. W w tym przypadku używamy metody do śledzenia komunikatów rejestrowania lub innych warunków błędów. Oczywiście możesz dodać ten fragment kodu w dowolnym miejscu, w którym chcesz śledzić niestandardowe zdarzenie w swojej grze.

Uwaga: Menedżer zdarzeń wykorzystuje system punktów, aby określić, ile zdarzeń można śledzić i przeprowadzać inne analizy. Każdy projekt zaczyna się od 1000 punktów, które powinny pokryć większość Twoich potrzeb w zakresie śledzenia.

Gdy zaczniesz zbierać zdarzenia niestandardowe, czy to z błędów, czy z innej zwykłej aktywności w grze, możesz powiązać te wskaźniki z innymi wskaźnikami. Na przykład możesz wykreślić liczbę błędów na użytkownika lub liczbę sesji gry, co powie Ci, jak krytyczny lub niegroźny jest błąd i jak szybko może być konieczne zareagowanie. Ułożenie tych fabuł wykracza poza zakres tej książki, ale warto zbadać je na stronie Unity, kiedy będziesz miał na to czas.

Jak zauważyłeś, analityka może być świetnym narzędziem nie tylko do śledzenia aktywności graczy, ale także innych działań, takich jak błędy lub inne niestandardowe zdarzenia. W następnej i ostatniej części książki przyjrzymy się innym problemom, które mogłeś napotkać w trakcie tej książki, oczywiście z możliwymi rozwiązaniami.

Zagadnienia i rozwiązania według Części

Poniższa tabela przedstawia, według rozdziałów, potencjalne błędy, które możesz napotkać, oraz listę możliwych rozwiązań dla każdego problemu:

Część/sekcja: Problem: Rozwiązanie

1. Konfigurowanie Androida do programowania: Nie można zlokalizować urządzenia za pomocą polecenia adb:

* Upewnij się, że urządzenie jest podłączone przez USB

* Upewnij się, że kabel działa z innymi urządzeniami

* Upewnij się, że port USB działa poprawnie; wypróbuj inne urządzenie USB, takie jak pendrive

* Potwierdź, że urządzenie ma włączone debugowanie USB

* Sprawdź, czy sterownik urządzenia jest zainstalowany

* Odłącz i podłącz kabel USB kilka razy, czekając kilka sekund między połączeniami

1. Budowanie i wdrażanie gry: Projekt się nie kompiluje (Android):

* Upewnij się, że ścieżki SDK i JDK są ustawione poprawnie

* Upewnij się, że identyfikator pakietu pasuje do nazwy kompilacji apk

* Potwierdź, że zainstalowałeś poprawną platformę SDK; jeśli nie, otwórz Eclipse i zainstaluj inne wersje platformy

1...8. Konstruowanie i wdrażanie gry: Kompilacja blokuje się w połowie i zawiesza się:

* Jest to częsty problem, który najczęściej pojawia się po podłączeniu urządzenia; po prostu bądź cierpliwy lub spróbuj anulować kompilację

* Jeśli tak się stanie, zamknij i ponownie otwórz Unity

2. Konfiguracja CUDLR : Nie można połączyć się z CUDLR :

* Spróbuj zmienić port używany przez CUDLR z 55055 na inny (1024 - 65535)

* Potwierdź, że adres IP urządzenia jest poprawny

* Sprawdź, czy składnia adresu URL jest poprawna — http://IPADDRESS:PORT

* Upewnij się, że na Twoim komputerze lub urządzeniu nie ma zapory, która może blokować połączenie; jeśli tak, dodaj wyjątek dla portu zgodnie z wymaganiami

* Upewnij się, że gra jest uruchomiona na urządzeniu

* Uruchom grę w edytorze Unity i spróbuj połączyć się przez

localhost -- http://localhost:55055

2. Konfiguracja usługi GPS: obrazy ze znakami zapytania:

* Jeśli korzystasz z urządzenia mobilnego, potwierdź, że usługi lokalizacyjne są włączone

* Jeśli korzystasz z urządzenia mobilnego, potwierdź, że usługa GPS działa, instalując aplikację testową lub korzystając z Google Maps

* Sprawdź, czy adresy URL wysyłane dla żądań kafelków zwracają obrazy, kopiując adres URL z okna konsoli lub CUDLR do okna przeglądarki

* Poczekaj kilka godzin; mogłeś przekroczyć limity użytkowania IP

6. Baza danych: Baza danych nie działa po wdrożeniu na urządzeniu:

* Potwierdź, że ustawienia wtyczki są prawidłowe dla Twojej platformy docelowej

* Upewnij się, że wersja bazy danych ma postać #.#.# i domyślnie 1.0.0

* Jeśli wdrażasz na iOS, potwierdź, że używasz IL2CPP

* Zatrzymaj grę, odinstaluj ją z urządzenia i ponownie wdroż grę

7. Konfigurowanie usługi Google Places API : Miejsca nie są wyświetlane

na mapie lub nie pasują do lokalizacji:

* Upewnij się, że usługa GPS działa poprawnie, a tryb symulacji jest wyłączony/włączony, zgodnie z wymaganiami

* Potwierdź, że lokalizacja jest włączona na Twoim urządzeniu

8. Aktualizacja bazy danych: Nie można sprzedać potworów do lokalizacji:

* Zatrzymaj edytor Unity i usuń plik bazy danych z folderu Assets/StreamingAssets

* Użyj narzędzia SQLite, aby bezpośrednio edytować lub weryfikować zawartość bazy danych. Świetnym narzędziem jest DB Browser for SQLite z http://sqlitebrowser.org/

* Odinstaluj grę i zainstaluj ją ponownie, to zresetuje bazę danych

* Złap potwory niższego poziomu

Podsumowanie

Ta Część dotyczył naprawiania problemów i rozwiązywania problemów, które możesz napotkać podczas typowego programowania. Zaczęliśmy od przyjrzenia się pierwszemu miejscu, na które należy zawsze zwracać uwagę, gdy coś pójdzie nie tak w Unity, czyli Konsoli. Od tego momentu przeszliśmy do przyjrzenia się typowym błędom i ostrzeżeniom, które mogą zatrzymać cię w rozwoju. To zabrało nas do debugowania, a następnie zdalnego debugowania skryptów z edytora kodu. Następnie przyjrzeliśmy się bliżej możliwościom rejestrowania w Unity, gdzie zakończyliśmy na przykładzie niestandardowego programu obsługi dziennika. To zaprowadziło nas z powrotem do przeglądu CUDLR jako naszej lokalnej/zdalnej konsoli, którą możemy połączyć z dowolną platformą i śledzić komunikaty dziennika, a nawet sprawdzać obiekty. Po spędzeniu tak dużo czasu na przeglądaniu narzędzi do tworzenia logów, przyjrzeliśmy się również wykorzystaniu Unity Analytics do przechwytywania komunikatów o błędach/wyjątkach po wydaniu gry. Na koniec przejrzeliśmy listę potencjalnych problemów i rozwiązań, które możesz napotkać podczas tworzenia gry demonstracyjnej zawartej w tej książce. To kończy naszą podróż do tworzenia gier AR opartych na lokalizacji. Mamy nadzieję, że dzięki Unity uzyskałeś wgląd w wiele obszarów tworzenia gier. Stąd zachęcamy do odkrywania rozwoju w rzeczywistości rozszerzonej, GIS i innych zaawansowanych funkcji rozwoju Unity. Ponadto mamy nadzieję, że podejmiesz decyzję i rozwiniesz własną grę AR opartą na lokalizacji.