Unity Prototype

GPS based Augmented Reality

Dokomentation

Christoph Prieß
Elisabethstraße 88
24143 Kiel

Kontakt:
Telefon 0159 0530 4049
E-Mail christoph.priess@student.fh-kiel.de

Inhalt

S. 2 **1 Prototyp**

S. 2 1.1 Installation des Unity Package

Import Custom Package
Unity Settings

S. 3 1.2 Inhalte des Prototypen

Vuforia
Ordner Struktur
Sample Scene
Hierarchie der Sample Scene

S. 5 1.3 Neue Scenes

Neue Augmented Scene
Neue Unity Scene

S. 6 1.4 C# Klassen

GPS
Scene Manager
Augmented Scene
Debug

S. 8 **2 Ausblick**

S. 8 2.1 Erweiterungen

Asynchrones Laden von Resourcen
Google Maps Karte
Automatische Berechtigungsgabfrage

S. 9 **3 ANHANG**

S. 9 3.1 Scripte

Wichtiger Hinweis für die Installation der App:
Nach der Installation muss manuell die Berechtigung für Standort Abfragen gewährt werden!

1 Prototyp

1.1 Installation des Unity Package

Import Custom Package

Um die GPS Funktion und den Scene Manager in einem neuen Unity Projekt zu nutzen, kann das Unitypackage dieses Prototypen einfach importiert werden.

Wichtiger Hinweis:
Bevor das Unity Package in ein bestehendes Projekt importiert wird, sollte ein Backup des gesamten Projektorders erstellt werden! Die Sample Scene des Projekts kann durch den Import ersetzt werden.

Folgendes ist für das Importieren des Packages nötig:

Über das Menu unter „Assets - Import Package - Custom Package“ kann die Datei mit dem Namen „GPSbasedAR_UnityPackage“ importiert werden.

Im Package sind alle nötigen Scripte und auch Prefabs enthalten, die für die GPS Funktion und das Scene Management benötigt werden. Zusätzlich befindet sich die Unity Sample Scene, welche den Prototypen und verschiedene kleine AR Scenes enthält.

Nach dem Import muss zunächst bestätigt werden, dass die Scene ersetzt und neu geladen werden soll. Die leere Sample Scene wird hierbei durch die Scene des Prototypen ersetzt, so dass nach dem Inport alle Funktionen nutzbar sind.

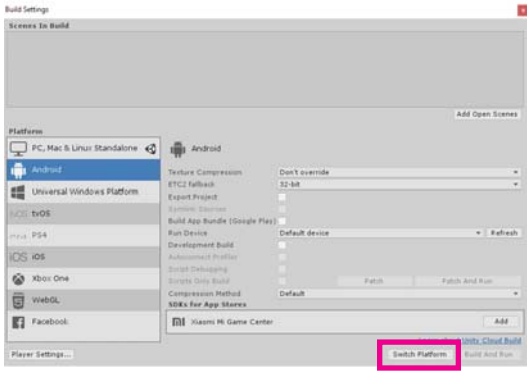
Unity Settings

Bevor das Package benutzt werden kann, muss zunächst Unity konfiguriert werden.

Folgendes ist hierfür nötig:

In den Player Settings, aufrufbar im Menu unter „Edit - Project Settings - Player“ unter dem Punkt „XR Settings“ muss der Suport für Vuforia aktiviert werden.

Im Menu unter „File - Build Settings“ muss das Unity Projekt auf die Android Plattform eingestellt werden.



Nun kann das Projekt mit der Build Funktion für Android Geräte ausgegeben werden. Auf dem Android Gerät muss für die App noch die Berechtigung für die Standortabfrage gegeben werden. Dies geht auf dem Gerät unter „Einstellungen - Apps - Berechtigungen - Standort“.

1.2 Inhalte des Prototypen

Vuforia

Um Augmented Reality Inhalte darzustellen, benutzt der Prototyp die bereits in Unity integrierten AR Lösung Vuforia. Die Funktion des GPS und des Scene Managers sind von Vuforia unanhängig nutzbar. So ließen sich die Funktionen wahrscheinlich auch auf z.B. ARCore übertragen.

Für das User Defined Image Target der Vuforia Engine wurde eine lokale Datenbank in das Projekt integriert. Über die Vuforia Webseite kann ein kostenloser Account erstellt werden, um Image Target Datenbanken zu erstellen.

Ordner Struktur

Alle Inhalte des Prototypen befinden sich im Projektordner unter „Assets - Resources - Custom“. Die allgemeinen Funktionen des Prototypen befinden sich hier im Ordner „Packages - GPSbasedAR“. Die Inhalte der AR Scenes befinden sich unter „Augmented Scenes - Kiel“.

Sample Scene

Das Image Target zur Aktivierung der Augmented Reality ist das Logo der Kultur Sphäre.

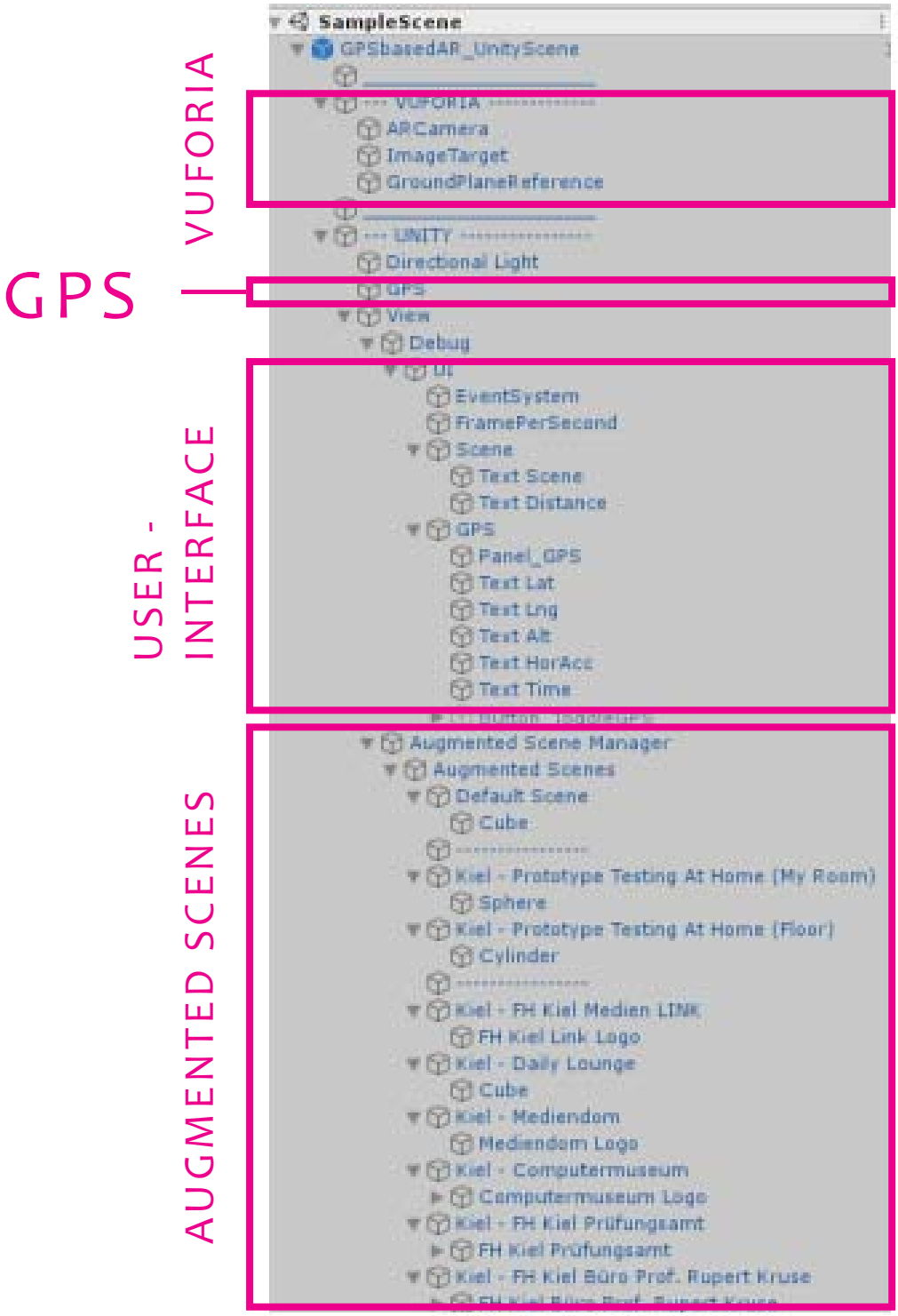
Wird von der Kamara eines AR fähigen Geräts erkannt, werden in der Augmented Reality die Logos zu einigen Locations der Fachhochschule Kiel gezeigt.

Wichtiger Hinweis:
Die Logos sind nur zu Testzwecken des Prototypen auf dem Gelände der Fachhochschule gedacht und nicht für eine Veröffentlichung!

Welche Location aktiviert wird, hängt von der ermittelten Position des Geräts ab. Ist das Gerät zu weit von den Locations entfernt, wird stattdessen eine Default Scene mit einem grauen Cube aktiviert.

Für das Aktivieren der AR Scenes wird ein Radius von 10 Metern um die Locations herum akzeptiert. Diese Einstellung kann konfiguriert werden, mehr hierzu im Kaptitel zu den Scripten.

Hierarchie der Sample Scene



1.2 Neue Scenes

Neue GPS Augmented Reality Scene

Eine neue GPS basierte AR Scene kann erzeugt werden, indem in der Hierarchie unter „View - Augmented Scene Manager - Augmented Scenes“ ein neues GameObject erstellt wird.

Dem neuen GameObject muss lediglich im Inspector das Script „AugmentedScene“ als Komponente hinzugefügt werden. Dieses GameObject muss das Parent aller GameObjects der neuen AR Scene sein, in der Hierachrie müssen die Inhalte der AR Scene also diesem GameObject untergeordnet sein.

Alternativ kann auch aus dem Projekt Fenster unter „Assets - Resources - Custom - GPSbasedAR - Prefabs“ das Prefab „AugmentedScene_Prefab“ in die Hierarchie unter „View - SceneManager - Scenes“ gezogen werden.

Im Inspector kann im „Augmented Scene“ Script die Latitude und Longitude der AR Scene eingestellt werden. Zusätzlich kann der AR Scene noch ein Name gegeben werden. Ist der Name leer, wird der Name des GameObjects übernommen.

Neue Unity Scene

Um eine neue Unity Scene mit den GPS Funktionen auszustatten, wird die MainCamera der Scene entfernt und das „GPSbasedAR_UnityScene“ Prefab in die Hierarchie gezogen.

Wie bereits im vorangehenden Abschnitt beschrieben wurde können anschließend neue GPS basierte AR Scenes hinzugefügt werden.

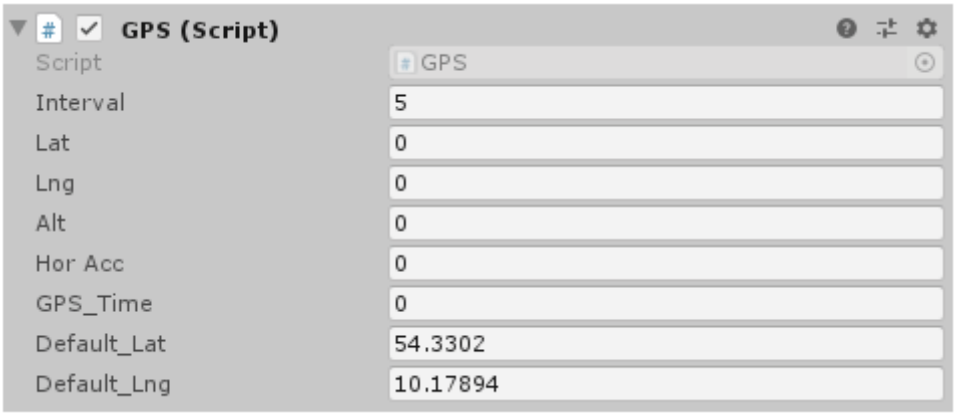
1.4 C# Klassen

GPS

Die GPS Klasse fragt in kurzen Intervallen die aktuelle Position ab. Im derzeitigen Prototyp wird die Position sehr häufig abgefragt, um die Anwendung besser testen zu können.

Im GPS Script wird keine weitere Konfiguration benötigt. Es kann aber eine Default Location eingestellt werden, diese wird verwendet, wenn in Unity der Playmode gestartet wird. Im Playmode können Latitude und Longitude beliebig verändert werden, der Scene Manager funktioniert ebenfalls im Playmode und die Aktivierung der Scenes kann in der Hierarchie verfolgt werden.

Um den Akku eines mobilen Endgeräts nicht zu sehr zu belasten, sollte hierfür im Code ein zeitliches Limit für die Abfragen implementiert werden. Dieser Schritt könnte aufgrund des Aufbaus des Code für die Abfrage kompliziert werden. Die Abfrage läuft zwar asynchron, d.h. mit einer Coroutine Funktion, über mehrere Frames, innerhalb der Coroutine wird aber eine while Schleife mit zahlreichen yields und breaks (yield = Fortsetzung im nächsten Frame, break = Abbruchbedingung) benötigt, der Code ist also relativ komplex.

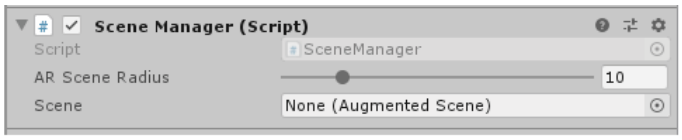


Scene Manager

Der Scene Manager de-/aktiviert die Augmented Scene GameObjects, welche ihm in der Hierarchie als Children untergeordnet sind. Hierbei wird die Distanz zwischen dem GPS des Geräts und der Augmented Scenes berechnet. Ist die Distanz kleiner als der im Scene Manager eingestellte Radius (z.B. 10 Meter), wird die Augmented Scene aktiviert.

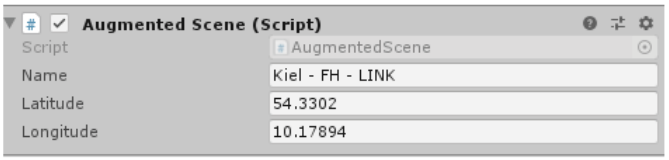
Befindet sich das Gerät nicht innerhalb eines Radius einer Augmented Scene, wird die Default Scene aktiviert. Diese zeigt einen grauen Cube.

Der Radius kann im Inspector des GameObjects „AugmentedSceneManager“ im „SceneManager“ Script eingestellt werden.



Augmented Scene

Im „Augmented Scene“ Script kann im Inspector die Latitude/Longitude und der Name einer AR Scene eingestellt werden.



Debug

Um die Anwendung auf mobilen Geräten zu testen und die Informationen über die Geräteposition zu erhalten, gibt es für GPS und SceneManager jeweils Debug Scripte. Diese sind lediglich dafür zuständig, dass die Informationen auf dem Bildschirm, d.h. in einem User Interface (Canvas) angezeigt werden.

2 Ausblick

2.1 Erweiterungen

Asynchrones Laden von Resourcen

Um die Performance der Anwendung bei vielen und komplexen Inhalten zu erhöhen, könnten die Inhalte erst zur Laufzeit geladen werden.

Im derzeitigen Prototypen werden alle Inhalte bei Start der Anwendung geladen, aber alle Inhalte der inaktiven AR Scenes wieder deaktiviert. Für einen Prototypen erscheint dies ausreichend, für eine professionelle Anwendung mit vielen speicher aufwendigen Inhalten sollten diese jedoch nur bei Bedarf in den Speicher geladen werden. Gerade im Bereich der technisch eher schwachen mobilen Endgeräte ist die Performance einer Anwendung ein entscheidender Faktor für die Usability und die User Experience.

Google Maps Karte

Um die Anwendung besser testen zu können und um dem Nutzer seinen Standort visuell zu bestätigen, könnte eine einfache, statische Google Maps Karte in das User Interface integriert werden.

Automatische Berechtigungsabfrage

Wenn die App installiert wurde und gestartet wird, fragt Vuforia nach der Berechtigung für die Nutzung der Kamera. Es müsste möglich sein, eine solche Abfrage auch für die Berechtigung von Standort Abfragen zu implementieren. Bis dahin muss die Berechtigung wie am Anfang der Dokumentation beschrieben wurde, manuell eingestellt werden.

3 Anhang

...\Assets\Resources\Custom\Packages\GPSbasedAR\GPS\GPS.cs 1

```
1  using UnityEngine;
2  using System.Collections;
3  using UnityEngine.UI;
4
5  public class GPS : MonoBehaviour
6  {
7      #region - Variables
8
9      // Request Interval
10     public float Interval = 5; // sec
11
12     // GPS data
13     public float Lat;
14     public float Lng;
15     public float Alt;
16     public float HorAcc;
17     public double GPS_Time;
18
19     // Default location if device has no GPS
20     // FH Kiel, Medien, LINK
21     public float Default_Lat = 54.3302f;
22     public float Default_Lng = 10.17894f;
23
24     // Turn GPS on/off
25     private bool Enabled = true;
26
27     // Async query
28     private IEnumerator GPSQuery;
29
30     // Time of last request
31     float TimeStamp = 0;
32
33     #endregion
34
35
36     #region - Unity Events
37
38     // Use this for initialization
39     void Start()
40     {
41         // Set default location to FH Kiel, Medien, LINK
42         Default_Lat = 54.3302f;
43         Default_Lng = 10.17894f;
44
45         // Immediately run a GPS query on application start
46         if (IsEnabled()) QueryCoordinates();
47     }
48
49     // Update is called once per frame
50     void Update()
51     {
52         // GPS is enabled
53         if (IsEnabled())
54         {
55             // Wait for Update Interval
56             if (Time.realtimeSinceStartup > TimeStamp + Interval)
```

```
57              {
58                  // If no query exists, request device location
59                  if (GPSQuery == null && Input.location.status !=
                      LocationServiceStatus.Running) QueryCoordinates();
60                  TimeStamp = Time.realtimeSinceStartup;
61              }
62          }
63          // GPS is disabled, but there still runs a query
64          else if (GPSQuery != null)
65          {
66              // Stop the last GPS request
67              StopCoroutine(GPSQuery);
68              GPSQuery = null;
69          }
70      }
71
72      #endregion
73
74
75      #region - Utility
76
77          // Calculates distance between two sets of coordinates, taking into
              account the curvature of the earth.
78          public float GetDistance(float lat1, float lon1, float lat2, float
              lon2)
79          {
80              double distance = 0;
81              var R = 6378.137; // Radius of earth in KM
82              var dLat = lat2 * Mathf.PI / 180 - lat1 * Mathf.PI / 180;
83              var dLon = lon2 * Mathf.PI / 180 - lon1 * Mathf.PI / 180;
84              float a = Mathf.Sin(dLat / 2) * Mathf.Sin(dLat / 2) +
85                Mathf.Cos(lat1 * Mathf.PI / 180) * Mathf.Cos(lat2 * Mathf.PI /
                    180) *
86                Mathf.Sin(dLon / 2) * Mathf.Sin(dLon / 2);
87              var c = 2 * Mathf.Atan2(Mathf.Sqrt(a), Mathf.Sqrt(1 - a));
88              distance = R * c;
89              distance = distance * 1000f; // meters
90              //convert distance from double to float
91              float distanceFloat = (float)distance;
92              return distanceFloat;
93          }
94
95      #endregion
96
97
98      #region - GPS
99
100     // ? Enabled/Disabled
101     public bool IsEnabled()
102     {
103         return Enabled;
104     }
105
106     // Enable/Disable GPS
107     public void ToggleGPS()
108     {
```

```
109         Enabled = !Enabled;
110         Debug.Log("GPS Enabled: " + IsEnabled().ToString());
111     }
112
113     // Start location request
114     void QueryCoordinates()
115     {
116         Debug.Log("Starting query for coordinates");
117         GPSQuery = GetCoordinates();
118         StartCoroutine(GPSQuery);
119     }
120
121     // Async function to request location of device
122     IEnumerator GetCoordinates()
123     {
124         while (IsEnabled())
125         {
126             // check if user has location service enabled
127             if (!Input.location.isEnabledByUser)
128             {
129                 print("Device location is not enabled");
130                 // Default Lat/Lng
131                 Lat = Default_Lat;
132                 Lng = Default_Lng;
133                 print("Location ist set to: Lat: " + Lat + ", Lng: " + Lng);
134                 yield break;
135             }
136
137             // Start service before querying location
138             Input.location.Start(1f, .1f);
139
140             // Wait until service initializes
141             int maxWait = 20;
142             while (Input.location.status == LocationServiceStatus.Initializing ↵
                    && maxWait > 0)
143             {
144                 yield return new WaitForSeconds(1);
145                 maxWait--;
146             }
147
148             // Service didn't initialize in 20 seconds
149             if (maxWait < 1)
150             {
151                 print("Timed out");
152                 yield break;
153             }
154
155             // Connection has failed
156             if (Input.location.status == LocationServiceStatus.Failed)
157             {
158                 print("Unable to determine device location");
159                 yield break;
160             }
161             else
162             {
163                 // Access granted and location value could be retrieved
```

```
164                 print("Location: " + Input.location.lastData.latitude + " " +
                      Input.location.lastData.longitude + " " +
                      Input.location.lastData.altitude + " " +
                      Input.location.lastData.horizontalAccuracy + " " +
                      Input.location.lastData.timestamp);
165
166                 // Overwrite current lat and lon everytime
167                 Lat = Input.location.lastData.latitude;
168                 Lng = Input.location.lastData.longitude;
169                 Alt = Input.location.lastData.altitude;
170                 HorAcc = Input.location.lastData.horizontalAccuracy;
171                 GPS_Time = Input.location.lastData.timestamp;
172             }
173             GPSQuery = null;
174             Input.location.Stop();
175         }
176     }
177
178     #endregion
179 }
```

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class AugmentedScene : MonoBehaviour
6  {
7      public string Name = "New Augmented Scene";
8      public float Latitude = 0;
9      public float Longitude = 0;
10
11     bool IsActive = false;
12
13     private void Start()
14     {
15         if (Name == null || Name == "") Name = gameObject.name;
16     }
17 }
18
```

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class SceneManager : MonoBehaviour
6  {
7      GPS GPS;
8      [Range(1, 50)]
9      public float ARSceneRadius = 1;
10
11     AugmentedScene[] Scenes;
12     public AugmentedScene Scene;
13     AugmentedScene DefaultScene;
14
15     void Start()
16     {
17         GPS = FindObjectOfType<GPS>();
18         Scenes = GetComponentsInChildren<AugmentedScene>();
19         foreach (AugmentedScene sc in Scenes)
20         {
21             if (sc.name.Equals("Default") || sc.name.Equals("Default Scene"))
                   DefaultScene = sc;
22         }
23         if (DefaultScene != null)
24         {
25             // Activate Default scene
26             ChangeScene(DefaultScene);
27         }
28         else Debug.Log("SceneManager --- Default Scene not found");
29     }
30
31     void Update()
32     {
33         if (GPS != null)
34         {
35             // Active last scene if GPS is available
36             if (GPS.IsEnabled())
37             {
38                 AugmentedScene NearScene = GetNearestScene();
39                 if (NearScene != Scene) ChangeScene(NearScene);
40             }
41         }
42     }
43
44     void ChangeScene(AugmentedScene NewScene)
45     {
46         if (gameObject.activeSelf && gameObject.activeInHierarchy)
47         {
48             // Disable all scenes
49             foreach (AugmentedScene scene in Scenes) scene.gameObject.SetActive
                   (false);
50             // Set new scene as active scene
51             Scene = NewScene;
52             Scene.gameObject.SetActive(true);
53         }
54     }
```

```
55
56      AugmentedScene GetNearestScene()
57      {
58          AugmentedScene NearScene = Scene;
59          float dist = -1;
60
61          foreach (AugmentedScene scene in Scenes)
62          {
63              if (dist == -1)
64              {
65                  dist = GPS.GetDistance(scene.Latitude, scene.Longitude,
                      GPS.Lat, GPS.Lng);
66                  NearScene = scene;
67              }
68              else
69              {
70                  float distance = GPS.GetDistance(scene.Latitude,
                      scene.Longitude, GPS.Lat, GPS.Lng);
71                  if (distance < dist)
72                  {
73                      dist = distance;
74                      NearScene = scene;
75                  }
76              }
77          }
78
79          // If near scene is not in the AR Scene Radius, we return the default
              scene
80          if (DefaultScene != null)
81          {
82              if (dist > ARSceneRadius) return DefaultScene;
83          }
84          else Debug.Log("Missing Default Scene");
85
86          return NearScene;
87      }
88  }
89
```

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.UI;
5
6  // Debug class to write GPS data to UI text elements.
7  // The purpose is to make the requested GPS data visible on mobile devices to
     test the application
8  public class GPSDebugInfo : MonoBehaviour
9  {
10     #region Variables
11
12     // Instance of GPS class
13     GPS GPS;
14
15     // UI text elements
16     Text Lat;
17     Text Lng;
18     Text Alt;
19     Text HorAcc;
20     Text Time;
21
22     // ? update process enabled/disabled
23     bool Process = false;
24
25     #endregion
26
27
28     #region Unity Events
29
30     // Use this for initialization
31     void Start()
32     {
33         if (InitGPSDebug())
34         {
35             Debug.Log("GPS Debug Info initialization successful");
36             Process = true;
37         }
38         else if (GPS != null) Debug.LogError("GPS Debug Info Error: Missing
             some UI text elements");
39         else Debug.LogError("GPS Debug Info Error: Missing GPS Object");
40     }
41
42     // Update is called once per frame
43     void Update()
44     {
45         // Continuously update GPS data on UI text elemts
46         if (Process) ShowGPSInfo();
47     }
48
49     #endregion
50
51
52     #region GPSDebugInfo
53
54     // Initialize UI text elements
```

```
55      bool InitGPSDebug()
56      {
57          GPS = FindObjectOfType<GPS>();
58          if (GPS != null)
59          {
60              Text[] Texts = GetComponentsInChildren<Text>();
61              foreach (Text text in Texts)
62              {
63                  switch (text.text)
64                  {
65                      case "Lat":
66                          Lat = text;
67                          break;
68                      case "Lng":
69                          Lng = text;
70                          break;
71                      case "Alt":
72                          Alt = text;
73                          break;
74                      case "HorAcc":
75                          HorAcc = text;
76                          break;
77                      case "Time":
78                          Time = text;
79                          break;
80                      default:
81                          // default...
82                          break;
83                  }
84              }
85              if (Lat != null && Lng != null && Alt != null && HorAcc != null &&
                    Time != null) return true;
86          }
87          return false;
88      }
89
90      // Update UI text elements with current GPS data
91      void ShowGPSInfo()
92      {
93          Lat.text = "Lat: " + GPS.Lat.ToString();
94          Lng.text = "Lng: " + GPS.Lng.ToString();
95          Alt.text = "Alt: " + GPS.Alt.ToString();
96          HorAcc.text = "HorAcc: " + GPS.HorAcc.ToString();
97          Time.text = "Time: " + GPS.GPS_Time.ToString();
98
99          // format string
100         //Lat.text = "Lat: " + GPS.Lat.ToString("0.00");
101     }
102
103     #endregion
104 }
105
```

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.UI;
5
6  public class SceneDebugInfo : MonoBehaviour
7  {
8      SceneManager SceneManager;
9      Text ScText;
10     Text DistText;
11     string ScStr;
12     string DistStr;
13
14     void Start()
15     {
16         Text[] texts = GetComponentsInChildren<Text>();
17         foreach (Text txt in texts)
18         {
19             if (txt.name == "Text Scene") ScText = txt;
20             else if (txt.name == "Text Distance") DistText = txt;
21         }
22         ScText = GetComponentInChildren<Text>();
23         SceneManager = FindObjectOfType<SceneManager>();
24     }
25
26     void Update()
27     {
28         if (ScText != null && DistText != null && SceneManager != null &&
             SceneManager.Scene != null)
29         {
30             GPS gps = FindObjectOfType<GPS>();
31
32             ScStr = "Scene: " + SceneManager.Scene.Name;
33             DistStr = "Dist: " + gps.GetDistance(SceneManager.Scene.Latitude,
                 SceneManager.Scene.Longitude, gps.Lat, gps.Lng);
34
35             if (ScText.text != ScStr) ScText.text = ScStr;
36             if (DistText.text != DistStr) DistText.text = DistStr;
37         }
38     }
39 }
40
```