OBA Unity Business App Template

9/29/22

OBA's basic Unity app architecture for beginning a new app. Includes basic navigation, app services, features, prefabs, scripts, and scenes.

Prefabs, Plugins, Packages

Scene Transition Manager (STM)

- **Function**: Animate scene changes ("Fade" only for now)
- **Name**: SceneTransitionManager
- **Animator**:
    - TransitionImage
    - TriggerTransition (SceneNav_in, SceneNav_out)
- **Resources**: https://weeklyhow.com/how-to-make-transitions-in-unity/

Navigation Button (NB)

- **Function**: Button to navigate between scenes
- **Name**: navButton
- **Script**: appNavigationHandler

Database Manager (DbM)

- **Function**: Define the app's database, initialize it and handle updates to the database structure.
- **Name**: DatabaseManager
- **Script**: databaseManager
- **Uses**: "SceneParametersHandler" for database name, connection string, and file path
- **Resources**:
    - Database is stored in the app's "StreamingAssets" folder
    - Download "64-bit DLL (x64) for SQLite" from https://sqlite.org/download.html
        - Add "sqlite3.dll" and "sqlite3.def" to Assets / Plugins folder
        - More Info: https://forum.unity.com/threads/installing-sqlite-on-macos-and-probably-for-windows-unity-2021-1-16.1179430/

- From "Unity.app/Contents/MonoBleedingEdge/lib/mono/unityjit-macos" add "Mono.Data.Sqlite.dll" to Assets / Plugins folder
    - Note: whenever updating the version of Unity grab the new "Mono.Data.Sqlite.dll".
- Download Code and Example: https://github.com/robertohuertasm/SQLite4Unity3d
    - Note: The important bit are the plugins and configuring them and the addition of the "StreamingAssets" folder
    - Note: Only need a small portion of the database path code to work in the editor and on iOS and Android.
    - UDemy Setup Tutorial: (see free "Downloading and Adding SQLite Plugin"): https://www.udemy.com/course/appnimi-unity-sqlite/
    - Configuring the plugins: https://www.youtube.com/watch?v=oPEl0mzeYoQ

Table Cell (TC)

- **Function**: Simple panel with text and an arrow button.
- **Name**: tableCell
- **Buttons**: "cellArrowButton" that calls the "appNavigationHandler" script
- **Note**: technically the "Navigation Button" prefab could have been nested inside this prefab

Get GPS Location (GGPSL)

- **Function**: Gets the device's current latitude, longitude, altitude, horizontal accuracy. Includes methods to start and stop location services. The data gathered by this prefab is handled separately from any display of the data so the prefab can be used in multiple scenes and multiple purposes in the app as needed.
- **Name**: GetGPSLocation
- **Uses**:
    - "getGPSLocation" script
- **Resources**:
    - https://www.youtube.com/watch?v=ZdyfjJbVvDk
    - https://docs.unity3d.com/ScriptReference/LocationService.Start.html

Get Orientation (GO)

- **Function**: Gets the device's current magnetometer readings. Includes methods to start and stop magnetometer services. The data gathered by this prefab is handled separately from any display of the data so the prefab can be used in multiple scenes and multiple purposes in the app as needed.

- **Name**: getOrientation
- **Uses**:
    - "getMagnetometer" script

Get Weather File (GWF)

- **Function**: Downloads a file of weather forecasts from graphical.weather.gov.
- **Name**: getWeatherFile
- **Uses**:
    - "getWeather" script
- **Resources**:
    - Download a File and store locally: https://docs.unity3d.com/Manual/UnityWebRequest-CreatingDownloadHandlers.html
    - Weather service: graphical.weather.gov

Weather Parser (WP)

- **Function**: Parse the weather forecast XML file from graphical.weather.gov. Store the forecast data in the local database.
- **Name**: WeatherParser
- **Uses**:
    - "weatherParser" script
- **Resources**:
    - Parse an XML file: https://www.youtube.com/c/KindsonTheTechPro (search his tutorials for "XML")

Added Packages & Plugins:

Before you run the app template be sure you have to following packages and plugins installed

- **SQLite** (plugin): This plugin and the necessary resources are free and are pre-installed in this template. However, see the "Database Manager (DbM)" description above to see where to get updates and current versions.

- **Native Gallery for Android & iOS (NG)** (plugin): This is a free plugin and is pre-installed in this template. You can find the current version of this plugin in the Unity Asset Store: https://assetstore.unity.com/packages/tools/integration/native-gallery-for-android-ios-112630

- **Native Share for Android & iOS (NS)** (plugin): This is a free plugin and is pre-installed in this template. You can find the current version of this plugin in the Unity Asset Store: https://assetstore.unity.com/packages/tools/integration/native-share-for-android-ios-112731

- **UniWebView (UWV)** (plugin): If you want to use the "Web View Scene" or the "Map View Scene" in this template you will need to purchase this plugin from the Unity Asset Store: https://assetstore.unity.com/packages/tools/network/uniwebview-5-229334

Scenes

Home Scene

- **Function**: App's main scene. Presents the app's UI including scaling and positioning.
- **Name**: homeScene
- **Uses**:
  - "startMeUp" object with the app's main startup script
  - "SceneTransitionManager" prefab to fade-in scene
  - "DatabaseManager" prefab to initialize the app's database
- **Buttons**:
  - InfoButton: uses "appNavigationHandler" script to load the "appInfoScene"
  - FeedbackButton: uses "appNavigationHandler" script to load the "appFeedbackScene"
  - ListButton: uses "appNavigationHandler" script to load the "listScene"
  - WebSceneButton: uses "appNavigationHandler" script to load the "webViewScene"
  - CameraViewButton: uses "appNavigationHandler" script to load the "cameraViewScene"
  - GPSButton: uses "appNavigationHandler" script to load the "gpsLocationScene"

- CompassSceneButton: uses "appNavigationHandler" script to load the "orientationViewScene"
- MapSceneButton: uses "appNavigationHandler" script to load the "mapViewScene"
- WeatherSceneButton: uses "appNavigationHandler" script to load the "weatherScene"
- PhotosSceneButton: uses "appNavigationHandler" script to load the "photoGalleryScene"
- ShareButton: uses "appNavigationHandler" script to load the "ShareScene"

App Info Scene

- **Function**: Display a simple scrollable view of text information about the app
- **Name**: appInfoScene
- **Uses**:
    - "SceneTransitionManager" prefab to fade-in scene
    - A scroll view
- **Buttons**: BackButton: uses "appNavigationHandler" script to load the "homeScene"
- **Resources**: Scroll View how to: https://www.youtube.com/watch?v=Q-G-W93jhYc

App Feedback Scene

- **Function**: A simple scene with an email button and a web site button
- **Name**: appFeedbackScene
- **Uses**: "SceneTransitionManager" prefab to fade-in scene
- **Buttons**:
    - BackButton: uses "appNavigationHandler" script to load the "homeScene"
    - Email Button: uses "getInTouchHandler" script to launch the devices mail app
    - Website Button: uses "getInTouchHandler" script to launch the device's browser with a specific URL

List Scene

- **Function**: Display a simple table of rows with cells with text and button leading to a data detail screen.
- **Name**: listScene
- **Uses**:
    - "SceneTransitionManager" prefab to fade-in scene
    - "listTableHandler" game object with script component "scrollScript"
    - List of table rows created using the "tableCell" prefab
- **Buttons**:

    - Back Button: calls "appNavigationHandler" script
    - Table row detail (arrow) buttons: use "appNavigationHandler" script to load the "rowDetailScene"
- **Resources**: Build table of rows with data: https://www.youtube.com/watch?v=hlNaNtApIMk

Row Detail Scene

- **Function**: Display detailed record data given data passed to the scene or retrieved using a data record key passed to the scene. This scene currently only displays the "key" (number) stored by the previous scene.
- **Name**: rowDetailScene
- **Uses**:
    - "SceneTransitionManager" prefab to fade-in scene
    - "handleSceneParameters" game object with script component "displayPassedParameters" to display passed data
- **Buttons**:
    - Back Button: calls "appNavigationHandler" script

Web View Scene

- **Function**: Display a remote web page.
- **Name**: webViewScene
- **Uses**:
    - "SceneTransitionManager" prefab to fade-in scene
    - "UniWebView" package
    - "UniWebView" script inside the package
- **Buttons**:
    - Back Button: calls "appNavigationHandler" script
- **Resources**:
    - UniWebView from Asset Store: https://assetstore.unity.com/packages/tools/network/uniwebview-4-175993
    - Guide: https://docs.uniwebview.com/guide/

Camera View Scene

- **Function**: Display a camera view with overlayed object. Camera view displayed by setting "RawImage" texture.
- **Name**: cameraViewScene
- **Uses**:
    - "SceneTransitionManager" prefab to fade-in scene

    - "getCameraView" script attached to "CameraView" raw image
- **Buttons**:
    - Back Button: calls "appNavigationHandler" script
- **Resources**:
    - https://csharp.hotexamples.com/examples/UnityEngine/WebCamTexture/-/php-webcamtexture-class-examples.html
    - https://answers.unity.com/questions/773464/webcamtexture-correct-resolution-and-ratio.html?childToView=1155328#answer-1155328

GPS Location Scene

- **Function**: Get and display the device's latitude, longitude, altitude, accuracy and time.
- **Name**: gpsLocationScene
- **Uses**:
    - "SceneTransitionManager" prefab to fade-in scene
    - "GetGPSLocation" prefab
    - "displayGPSLocation" game object and script
- **Buttons**:
    - Back Button: calls "appNavigationHandler" script
    - Start Button: calls "displayGPSLocation" object's "startGPSRequest" method, which calls "GetGPSLocation" prefab's "StartLocation" method
    - Stop Button: calls "displayGPSLocation" object's "stopGPSRequest" method, which calls "GetGPSLocation" prefab's "StopLocation" method
- **Resources**:
    - https://www.youtube.com/watch?v=ZdyfjJbVvDk
    - https://docs.unity3d.com/ScriptReference/LocationService.Start.html

Orientation View Scene

- **Function**: Using the device's magnetometer get and display compass directions.
- **Name**: orientationViewScene
- **Uses**:
    - "SceneTransitionManager" prefab to fade-in scene
    - "getOrientation" prefab
    - "DisplayOrientation" game object and script
- **Buttons**:
    - Back Button: calls "appNavigationHandler" script
    - Start Button: calls "displayOrientation" object's "startCompassRequest" method, which calls "getOrientation" prefab's "StartReadings" method
    - Stop Button: calls "displayOrientation" object's "stopCompassRequest" method, which calls "getOrientation" prefab's "StopReadings" method

- **Resources**:
    - https://tutorialsforar.com/create-an-ar-compass-app-using-ar-foundation-and-unity/

Map View Scene

- **Function**: Display a map via web view/page.
- **Name**: mapViewScene
- **Uses**:
    - "SceneTransitionManager" prefab to fade-in scene
    - "UniWebView" package
    - "loadMap" script to display maps in UniWebView
- **Buttons**:
    - Back Button: calls "appNavigationHandler" script
    - Terrain Map Button: calls "loadMap" script with map URL
    - Topo Map Button: calls "loadMap" script with map URL
- **Resources**:
    - Using "UniWebView" instead of the Nineva plugin.
    - *Nineva: https://docs.ninevastudios.com/#/unity-plugins/google-maps*
    - *Nineva Discord: https://discord.com/channels/512252380071460885/727086330068336700*
    - *Google Maps View (Nineva) Asset Store: https://assetstore.unity.com/packages/tools/integration/google-maps-view-82542*

Weather Scene

- **Function**: Retrieve and display the weather forecast. Display the latitude and longitude. Display the current date.
- **Name**: weatherScene
- **Uses**:
    - "SceneTransitionManager" prefab to fade-in scene
    - "getGPSLocation" prefab
    - "getWeatherFile" prefab
    - "WeatherParser" prefab
    - "displayWeather" script
- **Buttons**:
    - Back Button: calls "appNavigationHandler" script
    - Current Location Button: calls the "getCurrentGPS" method in the "getCurrentLocationForecast" script
    - Multiple Locations Button: calls the "initMultiForecast" method in the "getMultiLocationForecast" script

Photo Gallery Scene

- **Function**: Allow a photo from the device's photo library to be selected, displayed on the screen and saved as a file.
- **Name**: photoGalleryScene
- **Uses**:
  - "SceneTransitionManager" prefab to fade-in scene
  - "NativeGallery" plugin. Provides access to the device's photo library
  - "selectPhotoFromGallery" script
- **Buttons**:
  - Back Button: calls "appNavigationHandler" script
  - Pick A Photo Button: calls "SelectPhotoFromGallery" script. Display's the device's photo library.
- **Resources**:
  - NativeGallery Github: https://github.com/yasirkula/UnityNativeGallery
  - Unity Asset Store: https://assetstore.unity.com/packages/tools/integration/native-gallery-for-android-ios-112630

Share Scene

- **Function**: Allow the user to select an app (ie. Mail, Facebook) to share a screenshot and text (works on an iOS and Android device).
- **Name**: ShareScene
- **Uses**:
  - "SceneTransitionManager" prefab to fade-in scene
  - "NativeShare" plugin
  - "socialShare" script
- **Buttons**:
  - Back Button: calls "appNavigationHandler" script
  - Share Button: calls "socialShare" script.
- **Resources**:
  - NativeShare Github: https://github.com/yasirkula/UnityNativeShare
  - Unity Asset Store: https://assetstore.unity.com/packages/tools/integration/native-share-for-android-ios-112731

Scripts

Start Me Up (SMU)

- **Function**: Handle initial app setup and initialization
- **Name**: startMeUp
- **Called From**: First scene in the app ("HomeScene")

Database Manager

- **Function**: Define the app's database, initialize it and handle updates to the database structure.
- **Name**: databaseManager
- **Called From**: is a component of "DatabaseManager" prefab
- **Input**:
    1. The database version required by the app
- **Output**:
    1. A "Control" table
    2. Tables to store the weather forecast data

Scroll Script (SS)

- **Function**: Create a simple table of rows with cells including a number and button leading to a data detail screen. Each cell is added to a content area inside a scrollable area.
- **Name**: scrollScript
- **Called From**: is a component of "listTableHandler" game object
- **Uses**: the "tableCell" prefab to populate each row in the table.

App Navigation Handler

- **Function**: Handle all the scene navigation button requests in one place. If navigating to a scene from a table row then store data to be retrieved by the next scene.
- **Input**:
    1. Name of the scene to navigate to
    2. How to transition to the scene ("fade" only for now)
- **Name**: appNavigationHandler
- **Called From**:
    - UI navigation buttons (prefab)
    - Table cell button
- **Uses**: the "SceneParametersHandler" static class to store data needed by the scene to be loaded.

Scene Parameters Handler (SPH)

- **Function**: Define and hold data to be passed from one scene to another. Individual scenes needing passed data will find it here. This is a static class that does not need to be attached to any game object.
- **Name**: SceneParametersHandler
- **Called From**:
  - "databaseManager" to retrieve database name, file path, and connection string
  - "appNavigationHandler" to store data to be passed
  - "displayPassedParameters" to retrieve passed data
- **Resources**:
  - Passing Data: https://stackoverflow.com/questions/42393259/load-scene-with-param-variable-unity
  - https://stackoverflow.com/questions/54030622/how-do-i-carry-over-data-between-scenes-in-unity

Get In Touch Handler (GITH)

- **Function**: Ways the app user can get in touch for help and more info.
- **Name**: getInTouchHandler
- **Called From**:
  - "EmailButton" in the "appFeedbackScene" scene
  - "WebsiteButton"in the "appFeedbackScene" scene

Display Passed Parameters (DPP)

- **Function**: Displays parameters that were stored using "SceneParametersHandler".
- **Name**: displayPassedParameters
- **Called From**:
  - "handleSceneParameters" game object in "rowDetailScene"

Get Camera View (GCV)

- **Function**: Displays the device's camera view on a canvas' selected "RawImage" by setting its texture.
- **Name**: getCameraView
- **Called From**:
  - "cameraViewScene" scene

Get GPS Location (GGPSL)

- **Function**: Constantly queries the device's GPS location and includes public methods to to start and stop the device's location services. GPS data is publicly exposed.

- **Name**: getGPSLocation
- **Called From**:
    - "GetGPSLocation" prefab

Display GPS Location (DGPSL)

- **Function**: Display the device's current GPS Location data gathered by the "GetGPSLocation" prefab.
- **Name**: displayGPSLocation
- **Called From**:
    - "displayGPSLocation" game object in the "gpsLocationScene"

Get Magnetometer (GM)

- **Function**: Gets the device's current magnetometer readings. Includes methods to start and stop magnetometer services. Magnetometer data is publicly exposed.
- **Name**: getMagnetometer
- **Called From**:
    - "getOrientation" prefab

Display Orientation (DO)

- **Function**: Display the device's current magnetometer data (current compass direction) gathered by the "getOrientation" prefab.
- **Name**: DisplayOrientation
- **Called From**:
    - "DisplayOrientation" game object in the "orientationViewScene"

Load Map (LM)

- **Function**: Load maps into the UniWebView object.
- **Name**: loadMap
- **Called From**:
    - "TerrainMapButton" button in the "mapViewScene"
    - "TopoMapButton" button in the "mapViewScene"

Get Current Location Forecast (GCLF)

- **Function**: Get the weather forecast at the device's current location.
- **Name**: getCurrentLocationForecast
- **Called From**:
    - "Current Location" button on the "weatherScene" scene
- **Calls**:

    - Note successful permissions and results from each of the following leads to the next:
    - The "StartLocation" method in the "getGPSLocation" prefab
    - The "StopLocation" method in the "getGPSLocation" prefab
    - The "StartFileDownload" method in the "getWeatherFile" prefab
    - The "StartParser" method in the "WeatherParser" prefab
    - The "displayWeatherData" method in the "DisplayWeather" game object / script

Get Multi Location Forecast (GMLF)

- **Function**: Get the weather forecast given a list of locations (latitudes and longitudes).
- **Name**: getMultiLocationForecast
- **Called From**:
    - "Multiple Locations" button on the "weatherScene" scene
- **Calls**:
    - Note successful permissions and results from each of the following leads to the next:
    - The "StartFileDownload" method in the "getWeatherFile" prefab
    - The "StartParser" method in the "WeatherParser" prefab
    - The "displayWeatherData" method in the "DisplayWeather" game object / script

Get Weather

- **Function**: Download an XML weather file for a given start date, list of latitude and longitude locations for a given number of days from "graphical.weather.gov".
- **Name**: getWeather
- **Called From**:
    - "getWeatherFile" prefab
- **Input**:
    1. Longitude, latitude list of locations
    2. Forecast start date
    3. Number of days to forecast
    4. Name for the downloaded file
- **Output**:
    1. Status of the file download
    2. Path to the downloaded file

Weather Parser

- **Function**: Parse the XML weather file downloaded from "graphical.weather.gov". The weather will be stored in a database so the last recorded forecast can be retrieved even without an internet connection.
- **Name**: weatherParser
- **Called From**:
  - "WeatherParser" prefab
- **Uses**: "DATABASE_CONN" from "SceneParametersHandler" script
- **Input**:
  1. The file name of the XML weather forecast file
- **Output**:
  1. Status of the parsing

Display Weather (DW)

- **Function**: Display the weather forecast stored in the local database.
- **Name**: displayWeather
- **Called From**:
  - "DisplayWeather" game object
- **Uses**: "DATABASE_CONN" from "SceneParametersHandler" script

Select Photo From Gallery (SPFG)

- **Function**: Select a photo from the device's photo gallery, display it and save it to a file.
- **Name**: selectPhotoFromGallery
- **Called From**:
  - "Pick A Photo" button on the Photo Gallery Scene
- **Resources**:
  - https://support.unity.com/hc/en-us/articles/206486626-How-can-I-get-pixels-from-unreadable-textures-

Social Share (SocS)

- **Function**: Create a screenshot and present the OS's share dialog to the user to share the screenshot and text.
- **Name**: socialShare
- **Called From**:
  - "Share" button on the Social Share Scene

Device Builds

Build for Android: https://www.youtube.com/watch?v=Nb62z3J4A_A

Build for iOS: https://www.youtube.com/watch?v=80-nE7ichvk

- iOS build: set the following in Unity Build Settings:
  - Camera Usage Description: "Camera is used to identify surrounding peaks"
  - Microphone Usage Description: "Microphone is not used"
  - Location Usage Description: "Location services are used to get the current weather forecast"

- Xcode:
  - For Signing: use the "Signing & Capabilities" tab to set "Automatically Manage Signing", which is different from the video above.