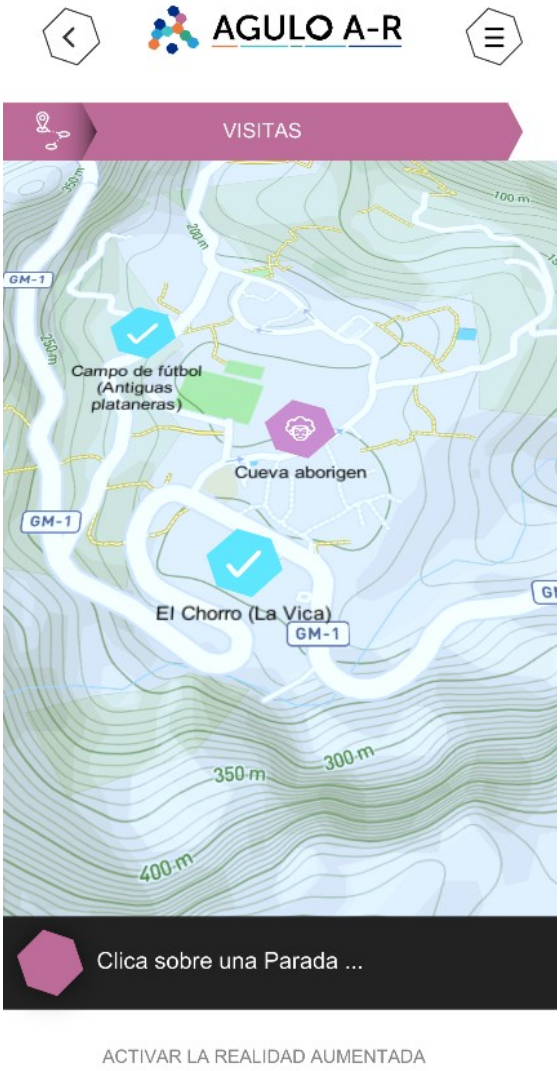
Documentación de la codificación



ACTIVAR LA REALIDAD AUMENTADA

Darel Martínez Caballero

Indice

Introducción:

En este documento se detallaran cada una de las clases y métodos usados para la construcción de la APP de AguloAR.

Al haber desarrollado la aplicación en Unity, me he visto obligado a distribuir todo en pequeños componentes. Dando un Resultado total de unas 35 clases. Cada una con un pequeño propósito. Además han habido clases que se comparten con diferentes escenas de Unity que han sido tratadas de una forma especial usando un patrón llamado Singleton (Instancia única).

Además para el desarrollo de AguloAR se han usado múltiples librerías para poder llevar acabo todo el proyecto. Las librerías usadas:

- JsonDotNet (Para trabajar con JSON)
- Firebase (Para el uso de la base de datos)
- MapBox (Para el tratamiento del mapa y el uso de puntos de interés)
- EasyAR (Para el uso de realidad aumentada y reconocimiento de imágenes)

Creo que es de importancia ver de forma general todos los scripts usados para el tratamiento y desarrollo de la APP de AguloAR.

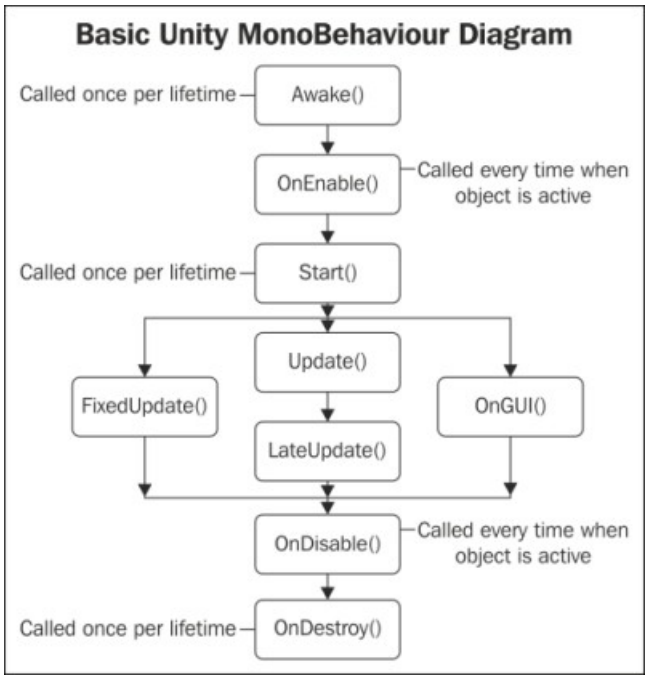
También mencionar que la APP se ha desarrollado única y exclusivamente en C#.

Desarrollo

En Unity existen diferentes métodos por defecto que se usarán según que se vaya a hacer o se necesite. Estos métodos tienen un ciclo de vida que afectará a como deben de usarse. Algunos de estos métodos son:

- Start()
- Update()
- Awake()
- OnEnable()
- OnDestroy()

Para que quede mas claro:



CacheFirstExecution

Esta clase se ejecuta en la primera escena, y se encarga de guardar las preferencias de idiomas para saltar en su segunda ejecución directamente a la escena primaria y no pasar por la de idiomas. Estas preferencias se podrán cambiar.

- Método Start()
  - En su método Start() verificamos si existe el archivo “preference.txt”, si existe cargamos el idioma guardado. Si no existe el contenedor de idiomas oculto lo pasamos a visible para que el usuario elija el idioma de su preferencia.

- Método Update()
  - Este método se encargará de actualizar el estado del contenedor de idiomas haciendo que se visualice el mismo.

- Método saveData(string idioma)
  - Este método es llamado cada vez que el usuario selecciona un idioma, cuando el usuario realiza tal acción se guarda el idioma (es, en, de) en el archivo preference.txt.

CheckBoxState

Esta clase es un molde con los valores booleanos para cada uno de los chekboxs contenidos en la escena “visitScene”

- Método Awake()
  - Contiene un patrón Singleton para quedarnos con una única instancia de este objeto y poder hacer referencia siempre al mismo.

Además contiene sus métodos getter y setter siguientes:

- public bool Lugares
- public bool Personajes
- public bool Arquitectura
- public bool Tradiciones
- public bool HistoriaAborigen
- public bool Iniciado

CheckStateTracking

Esta clase se encarga de comprobar continuamente si la imagen que se trata de reconocer coincide con la parada seleccionada, si es a si se procederá a descargar el video, imagen o animación perteneciente a esa parada.

- Método Awake()
  - Se encarga de ejecutar el método makeRequest().
- Método Update()
  - Este método se encarga continuamente de comprobar si coincide la imagen a reconocer y la parada que se trata de visualizar. Si es a si, el componente con el contenido se hará visible.
- Método makeRequest()
  - Se encarga de realizar la petición a la base de datos para proceder a la descarga del video o animación de esa parada.
- Método getText(IEnumerable<string> lineas)
  - Se encarga de recorrer el string con el nombre de la parada si esta es muy grande se añadirán saltos de lineas.

Close

Esta clase se encarga de comprobar si el gps está activo, si no es a si mostrará un aviso que al darle click te redirigirá a la escena de visitas.

- Método closeWindow(GameObject aviso)
  - Se le pasa un objeto y este lo pondrá en oculto.
- Método Update()
  - Comprobará continuamente si el GPS está activo, si no lo está mostrará un aviso.

ComponentInfo

Esta clase se encarga de estar pendiente si el usuario hace click sobre alguna de las paradas, en ese caso se activará el botón para ir a la escena de realidad aumentada.

- Método Awake()
  - Lleva un patrón singleton para quedarnos con una sola instancia del objeto y poder trabajar los datos que contiene.

- Método Update()

    - Se encarga de actualizar el estado del botón de AR, además de actualizar el color, texto del banner con el nombre de las paradas y el botón de AR.

GetJsonClimaticoAndFunciones

Esta clase se encarga de conectarse a la base de datos Firebase, la cuál nos devuelve un JSON con el que poder trabajar. En este caso nos traemos toda la información perteneciente a la escena de CambioClimatico y la escena de AppFunc. Comparten el mismo formato de JSON donde se descarga: una imagen y un texto.

- Método Start()
    - Se encarga de llamar al método getTextCambioClimaticoAndFunc(), además contiene la conexión a la base de datos, donde si el resultado es correcto llamará al método login().
- Método login()
    - Se encarga del login hacia la base de datos de Firebase.
- Método getTextCambioClimaticoAndFunc()
    - Este método se encarga de llamar al método loadData() además de inicializar una co-rutina con el método StartCorutine(makeRequestImage())
- Método loadData()
    - Se encarga de la llamada al método valueChange() con los datos que queremos traernos de la base de datos.
- Método valueChange()
    - Este método es un método especial es el que se encarga de traer el JSON de la base de datos Firebase además los datos como el texto son tratados y parseados, ya que el texto viene con un formato preestablecido con etiquetas HTML y se debe parsear para que se interpretada con las etiquetas richText.
- Método makeRequestImage()
    - Una vez obtenido la URL de la imagen esta para ser usada en un objeto 2D o 3D de Unity debe tratarse de una forma especial, se debe pasar como textura.

GetJsonVisitasInfo

Esta clase se encarga de la conexión a la base de datos Firebase trayendo el JSON y tratándolo para obtener los datos perteneciente a la parada de la cuál se quiere saber información.

- Método Start()
    - Se encarga de llamar al método getTextVisitasInfo(), además se encarga de intentar hacer la conexión a la base de datos, si la conexión es correcta llamará al método login().
- Método login()
    - Se encarga de realizar el login en la base de datos a la que tratamos de acceder.
- Método getTextVisitasInfo()
    - Se encarga de llamar al método loadData() y además se encarga de inicializar una co-rutina con el método StartCorutine(makeRequestImage()).
- Método loadData()
    - Se encarga de la llamada al método valueChange() con los datos que queremos traernos de la base de datos.
- Método valueChange()
    - Este método es un método especial es el que se encarga de traer el JSON de la base de datos Firebase además los datos como el texto son tratados y parseados, ya que el texto

viene con un formato preestablecido con etiquetas HTML y se debe parsear para que sea interpretada con las etiquetas richText.

- Método makeRequestImage()
  - Se encarga de convertir la imagen de la URL proporcionada en una textura y posteriormente pasarla a un objeto tipo imagen.

GpsLocation

Esta clase se encarga de comprobar continuamente el estado del GPS.

- Método Update()
  - Ejecuta constantemente una co-rutina contenida en el método GPSLoc().
- Método GPSLoc()
  - Este método se encarga de comprobar el estado actual del gps del usuario.

Lenguage

Esta clase se encarga de devolver dos valores que serán usados por las demás clases. Estos valores son el idioma (“es”, “de”,”en”) y la posicion del indice del idioma actual (0,1,2).

- Método Awake()
  - Contiene un patrón singleton para quedarnos con una única instancia de este objeto y poder trabajar con los valores que tienen sus variables.
- Método Update()
  - Se encarga constantemente de verificar la posición del indice del idioma y actualizarlo si el idioma cambia.
- Método setIdioma(string leng)
  - Este método se encarga de setear el valor del idioma, valores admitidos → es, de, en

Menu

Esta clase se encarga de la apertura y cierre de la escena MenuBlackScene.

- Método Awake()
  - Contiene un patrón singleton para quedarnos con una instancia única de este objeto.
- Método openMenu()
  - Se encarga de abrir el MenuBlackScene guardando el número de escena desde la que se está abriendo el Menú, ya que este es accesible desde todas las escenas.
- Método closeMenu()
  - Cierra la escena MenuBlackScene y te devuelve a la escena desde la que lo abriste.

ObjectImgAndText

Esta clase es un molde que será usado por la clase getJsonClimaticoAndFunciones, este molde es el necesario para tratar el JSON descargado de la base de datos FIREBASE.

No contiene ningún método, sus variables usadas para dar forma al JSON son:

```
public int id { get; set;}
public string titulo { get; set;}
public string descripcion { get; set;}
```

public string imagen { get; set;}

ObjectInfoParadas

Esta clase es un molde que será usado por la clase getJsonVisitasInfo, este molde es el necesario para tratar el JSON descargado de la base de datos FIREBASE.

No contiene ningún método, sus variables usadas para dar forma el JSON son:

public int id { get; set;}
public string id_parada { get; set;}
public string titulo { get; set;}
public string imagen { get; set;}
public string descripcion { get; set;}
public string saludo { get; set;}
public string extra { get; set;}
public string slug { get; set;}

ObjectVid

Esta clase es un molde que será usado por la clase checkStateTracking, este molde es el necesario para tratar el JSON descargado de la base de datos FIREBASE.

public int id { get; set;}
public string id_modelo { get; set;}
public string titulo { get; set;}
public string video { get; set;}
public string modelo { get; set;}
public string slug { get; set;}

ObjectVidList

Esta clase tan solo contiene una lista de tipo ObjectVid, se ha hecho a si para poder compartir de una forma simple esta clase atreves de un objeto prefab de Unity.

OpenInfo

Esta clase se encarga de contener ciertos datos que deben ser compartidos con diferentes escenas MapaScene e InfoScene.

- Método Awake()
  - Contiene un patrón singleton para quedarnos con una única instancia del objeto.
- Método getNameParada()
  - Obtiene el nombre de la parada a partir del GameObject que se le pasa por parámetro. Y además se encarga de setear el nombre en un objeto tipo Text, además de cambiar el color del banner de Ar también el texto de AR y de poner en visible la flecha de AR.
- Método Name
  - Es un getter y setter de la variable Name que contiene el nombre de la parada seleccionada.

OpenLink

Esta clase contiene un único método que nos permite pasar un texto con formato URL y lo transforma en un texto tipo LINK el cuál puede ser clickeado.

- Método Open(string URL)
  - Nos permite convertir un texto en un link.

Parada

Esta clase es un molde que será usado por la clase SpawnOnMapMod. Este molde contiene las coordenadas, nombre, tipo y visibilidad de la parada.

Paradas

Esta clase contiene una lista de tipo Parada.

- Método Paradas()
  - Se trata del constructor de la clase y en ella se encuentran las múltiples paradas usadas y su información
- Método Awake()
  - Contiene un patrón singleton para quedarnos con una instancia única de este objeto y que sea compartida por múltiples objetos.
- Método Instance()
  - Devuelve la instancia
- Método listaParadas()
  - Devuelve la lista de paradas.

ResetStateCheckBox

Esta clase se encarga limpiar los valores establecidos en los checkboxs de la escena VisitSceneAR devolviendo sus valores a false.

- Método Reset()
  - Este método limpia los valores de los chekboxs y la parada seleccionada.
- Método ResetParada()
  - Este método se encarga de “deseleccionar” la parada que se haya elegido.

SelectParada

Esta clase contendrá los objetos checkboxs (Toggle) para poder trabajar con ellos.

- Método Awake()
  - Se encarga de poner a false la visibilidad del botón que te lleva al mapa con los tipos de paradas seleccionados.
- Método addOrRemoveParada()
  - Se encarga simplemente de establecer un idioma, si este se encuentra a null será ES por defecto.

- Método paradasAmostrar()
  - Se encarga de recorrer cada una de las paradas y cambiar su visibilidad a true o false si estas pertenecen al grupo de paradas seleccionadas.
- Método setToggle(Toggle toggle)
  - Se encarga de añadir a una lista de tipos de paradas activas cada uno de los checkboxs (Toggle) que se encuentren seleccionados.
- Método list()
  - Se encarga de setear los valores de la clase CheckBoxState según el valor de los checkbox de la escena VisitScene
- Método listener(bool state, Toggle toggle)
  - Se encarga de estar atento a los cambios que se produzcan en los valores de los checkboxs
- Método Update()
  - Se encarga de comprobar si hay algún checkbox con valor true, si es a si se visualiza la imagen que permite cambiar a la escena mapa. Además llamamos al método listener continuamente para comprobar continuamente cada checkbox y el valor de cada unos de los valores booleanos de la clase CheckBoxState.

SetNameVisitasAr

Se encarga de setear el nombre de la parada seleccionada al banner con su texto.

- Método Update()
  - Se encarga de actualizar el texto del banner con el nombre de la parada seleccionada.

SpawnOnMapMod

Esta clase se encarga de poner en el mapa las paradas (puntos de interés) que pertenezcan al grupo de paradas seleccionados en la escena de VisitScene.

- Método Start()
  - Se encarga de recorrer un array con las paradas seleccionadas. Estas contienen unas coordenadas GPS y este es el dato usado para saber la posición donde deben estar colocadas en el mapa. Además desde aquí se llamará al método comprueba para comprobar si esta parada ha sido visitada o no.
- Método comprueba()
  - Se encarga de comprobar si el nombre de la parada existe en el fichero generado “paradasVisitadas” si es a si, el icono de la parada será el icono de parada ya visitada (color verde).
- Método addSaltoLinea()
  - Agregará un salto de linea al nombre de la parada si este supera los 14 caracteres.
- Método Update()
  - Se encarga de spawnear cada objeto parada en el mapa. Además ocultará la imagen de carga.

SwitchScene

Esta clase se encarga del cambio de escena entre las múltiples escenas existentes.

- Método switchScene(int scene)
  - Se le pasa el numero de escena a la que se quiere cambiar, y cambia la escena.

- Método SwitchSceneActivarAr()
  - Comprueba si está activo el botón de activarAr, si es a si te cambia a la escena de AR.
- Método switchSceneIdiomas()
  - Se encarga de llevarte a la escena de idiomas desde cualquier escena.

TextActivarAr

Esta clase se encarga de cambiar los textos de la escena AR.

- Método Update()
  - Este método se encarga de actualizar los textos de estas escena según el idioma que se haya seleccionado en la APP.

TextEstadisticas

Esta clase se encarga de cambiar los textos de la escena estadisticas.

- Método Update()
  - Este método se encarga de actualizar los textos de esta escena según el idioma que se haya seleccionado en la APP.

TextCarga

Esta clase se encarga de cambiar los textos del objeto con la imagen de carga.

- Método Update()
  - Este método se encarga de actualizar los textos de este objeto según el idioma que se haya seleccionado en la APP.

TextInfoCanvas

Esta clase se encarga de cambiar los textos pertenecientes a la escena InfoScene.

- Método Update()
  - Este método se encarga de actualizar los textos de esta escena según el idioma que se haya seleccionado en la APP.

TextInfoScene

Esta clase se encarga de cambiar los textos pertenecientes a la escena InfoScene.

- Método Update()
  - Este método se encarga de actualizar los textos de esta escena según el idioma que se haya seleccionado en la APP.

TextMapaScene

Esta clase se encarga de cambiar los textos pertenecientes a la escena MapaScene.

- Método Update()
  - Este método se encarga de actualizar los textos de esta escena según el idioma que se haya seleccionado en la APP.

TextMenuBlack

Esta clase se encarga de cambiar los textos pertenecientes a la escena MenuBlackScene.

- Método Update()
  - Este método se encarga de actualizar los textos de esta escena según el idioma que se haya seleccionado en la APP.

TextParadaVisitada

Esta clase se encarga de cambiar los textos pertenecientes al saludo de cada parada.

- Método Update()
  - Este método se encarga de actualizar los textos de este objeto según el idioma que se haya seleccionado en la APP.

TextPrimaryScene

Esta clase se encarga de cambiar los textos pertenecientes a la escena PrimaryScene.

- Método Update()
  - Este método se encarga de actualizar los textos de esta escena según el idioma que se haya seleccionado en la APP.

TextVisitas

Esta clase se encarga de cambiar los textos pertenecientes a la escena VisitasScene.

- Método Update()
  - Este método se encarga de actualizar los textos de esta escena según el idioma que se haya seleccionado en la APP.

TextVisitSceneAr

Esta clase se encarga de cambiar los textos pertenecientes a la escena VisitSceneAr.

- Método Update()
  - Este método se encarga de actualizar los textos de esta escena según el idioma que se haya seleccionado en la APP.

TranslateAnimationZ

Esta clase se encarga del control de movimiento en el eje Z de la animación spoetnik AnimationZ.

- Método Update()

- Se encarga de mover el objeto con el globo spoetnik hacia arriba y resetear su estado devolviéndolo a su punto de partida, y vuelta a empezar.

Conclusiones

Como se a comentado al principio del documento Unity te obliga a componentizar absolutamente todo. Repartiendo cada acción de cada clase en múltiples objetos y o clases. Haciendo más fácil el poder reparar o realizar cambios en cada clase o método.

Bibliografía

Unity Technologies. (s/f). *Unity user manual 2021.3 (LTS)*. Unity3d.Com. Recuperado el 27 de mayo de 2022, de https://docs.unity3d.com/Manual/index.html

Métrica v.3. (s/f). Gob.es. Recuperado el 27 de mayo de 2022, de https://administracionelectronica.gob.es/pae_Home/pae_Documentacion/pae_Metodolog/pae_Metrica_v3.html