uniqtem

Hellgate Framework Version 1.4.0

- Download Asset Store
- Scripts Reference
- Basic Unity project framework
- Requires Unity 5.0 or higher.
- Requires NGUI: Next-Gen UI kit.
- Requires SS-System 3.0: UI and Scene Manager.
- External MiniJSON.
- Scripts in c#

---

uniqtem

Contents

uniqtem

1. Introduce

Hellgate Framework synthesize the essential part of general as Application and Game Development with Unity is angry Framework project. Powerful and able to develop at a rapid pace and will ensure ease of development and easy use.

Hellgate Framework is providing an example and Reference documentation for all functions. And give assistance in rapid understanding and use through a variety of sample code. In fact, by making sure that the sample code to help the eye to understand and use that run is doubled. It is currently available for download from the sample to the Application Google Play is the biggest advantage.

Hellgate Framework is a body of experience While progress has been reflected in the actual project, and feel the results identified in the current project. Therefore, the specifications of the project's progress as iterative development and change that can respond quickly and accurately. Most projects can be seen that part of the development of non-combat Scene formats and methods are nearly identical. A project to address this area.

Hellgate Framework project is in progress. And the quick response and bug updates.

Features Hellgate Framework is to be proud of
Loading Job Controller : AssetBundle load, Http Request / Response, Intent, including the processing and Job Scene Switch, Loading Bar representation, Data transfer.
Reflection Convert : Dictionary, List is produced by the Instance simple using Reflection. Of course, the reverse is possible.
Sqlite ORM : Simple and powerful means only enable needed. OM is also available.
Scene Manager/Controller : Call using the SS-System Scene/Delete/Enable/Disable
AssetBundle : Build, Initial Download, Manager.
Http : Requrt / Response, UI integration.
Encrypt : The Unity of PlayerPrefs (Game Sessions) to be used.

In addition, there are many functions.

2. Install

Hellgate Framework is NGUI and SS-System, you need to download. NGUI has been used in reference to the UI, You may use the UGUI but you can not use some of the functions and examples. SS-System is free of charge and are based to the Scene Manager.

2.1. Asset Download

NGUI : https://www.assetstore.Unity.com/en/#!/content/2413
SS-System : https://www.assetstore.Unity.com/en/#!/content/12947

2.2. Modify SS-System

After the download you need to modify the code in the SS-System.
1. SSSystem/Scripts/SSRoot.cs

```
protected virtual void Update()
{
    #if UNITY_EDITOR
    Rename();
//      FindCameras();
    FindEventSystems();
    #endif
}
```

2. SSSystem/Scripts/Compatibility/nGUI/SSAutoAddUICamera.cs

```
private void Awake()
{
    gameObject.AddComponent<UICamera> ();
//      UnityEngineInternal.APIUpdaterRuntimeServices.AddComponent(gameObject,
"Assets/SSSystem/Scripts/Compatibility/nGUI/SSAutoAddUICamera.cs (8,3)", "UICamera");
}
```

Ready to proceed with this project has been completed. now

```
using Hellgate;
```

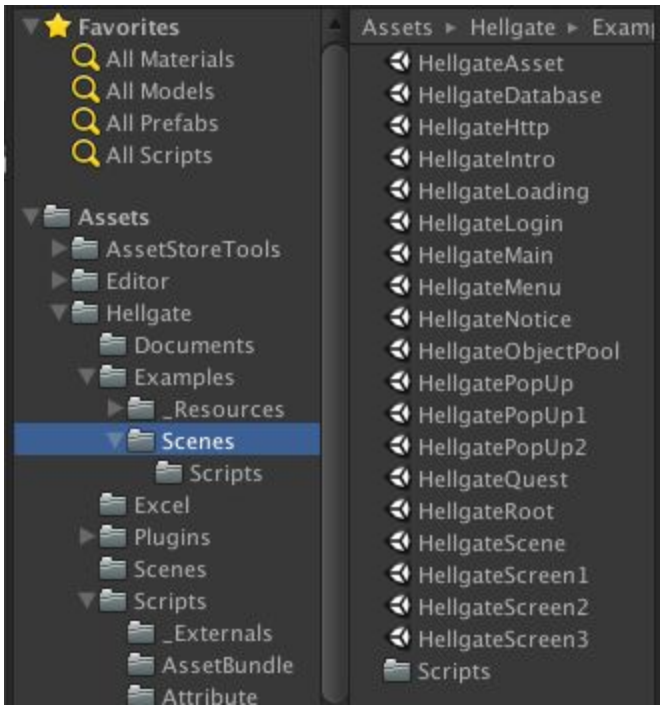
is set, You can use all the functions of **Hellgate Framework**.

3. Hello World!!

Looking return the sample code provided in **Hellgate Framework**, we can try the functions identified. There example code for all functions and can also be downloaded from Google Play.
Google Play : https://play.google.com/store/apps/details?id=com.uniqtem.hellgate

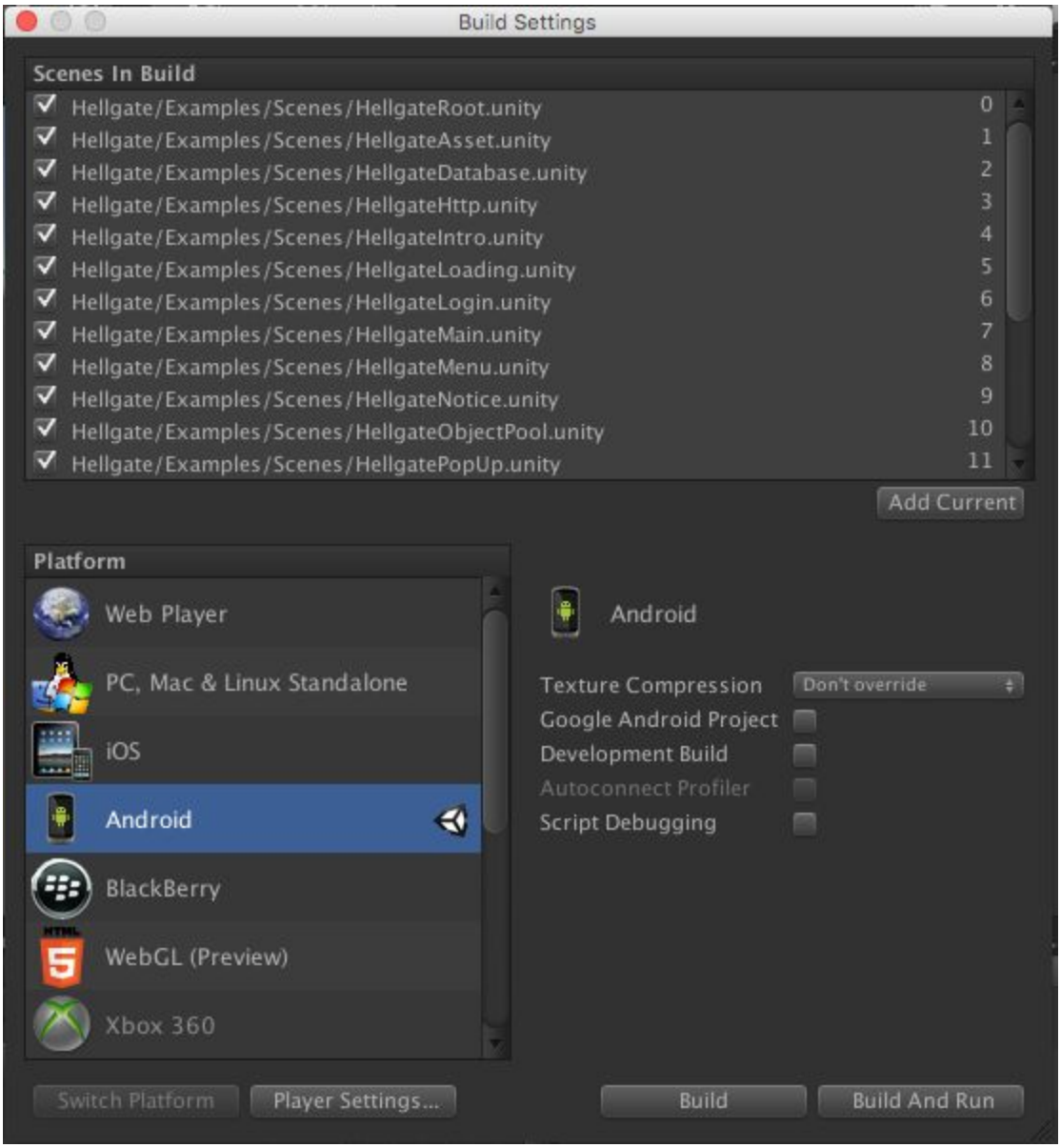
3.1. Set Scenes in build

Unity Editor/File/Build Settings에 Assets/Hellgate/Examples/Scenes of Scene set all ([Screenshot 1] refer)



Screenshot 1.

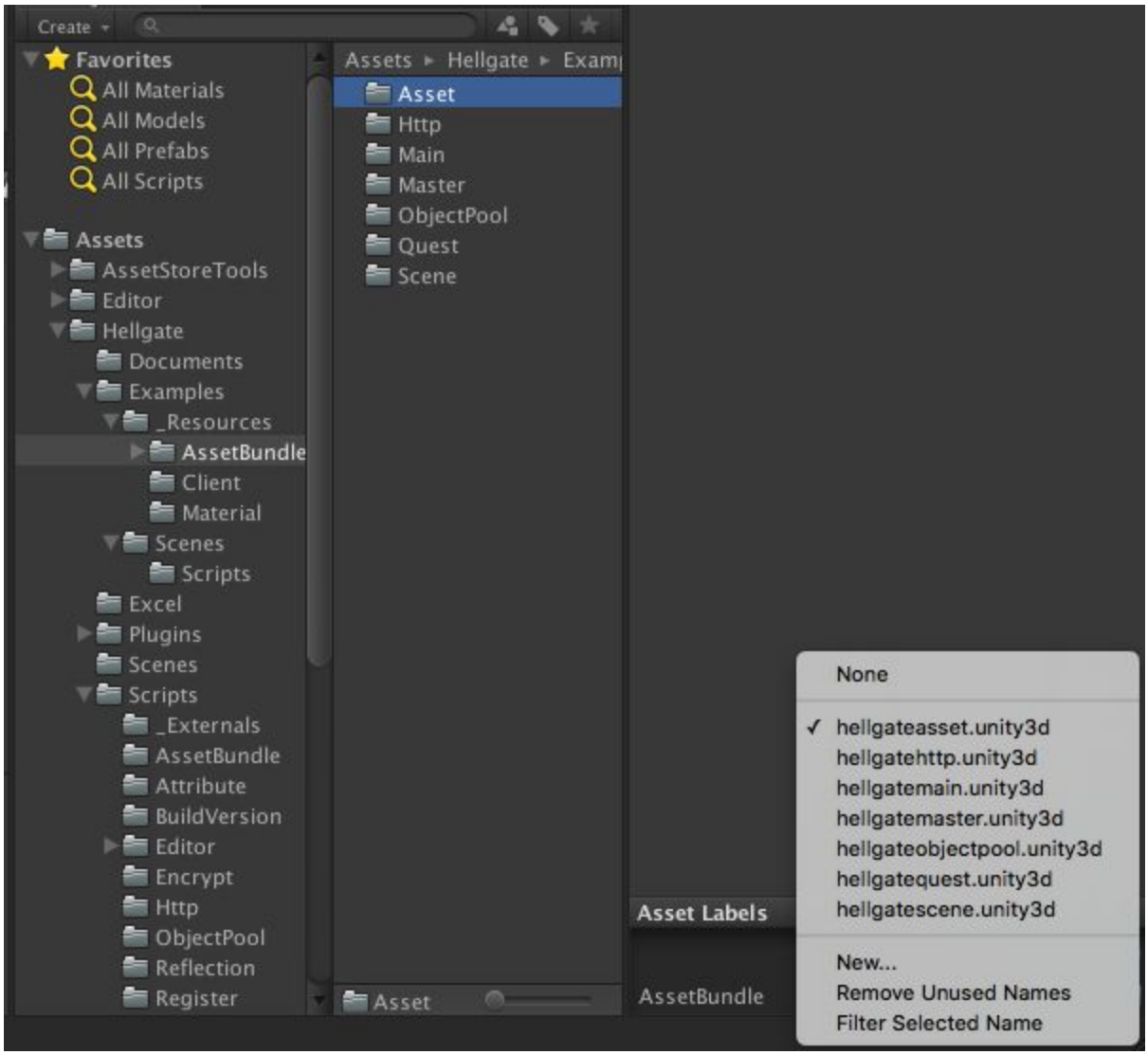
When displayed as [Screenshot 2] The Scenes in build setting is complete.



Screenshot 2.

3.2. Set Asset Labels

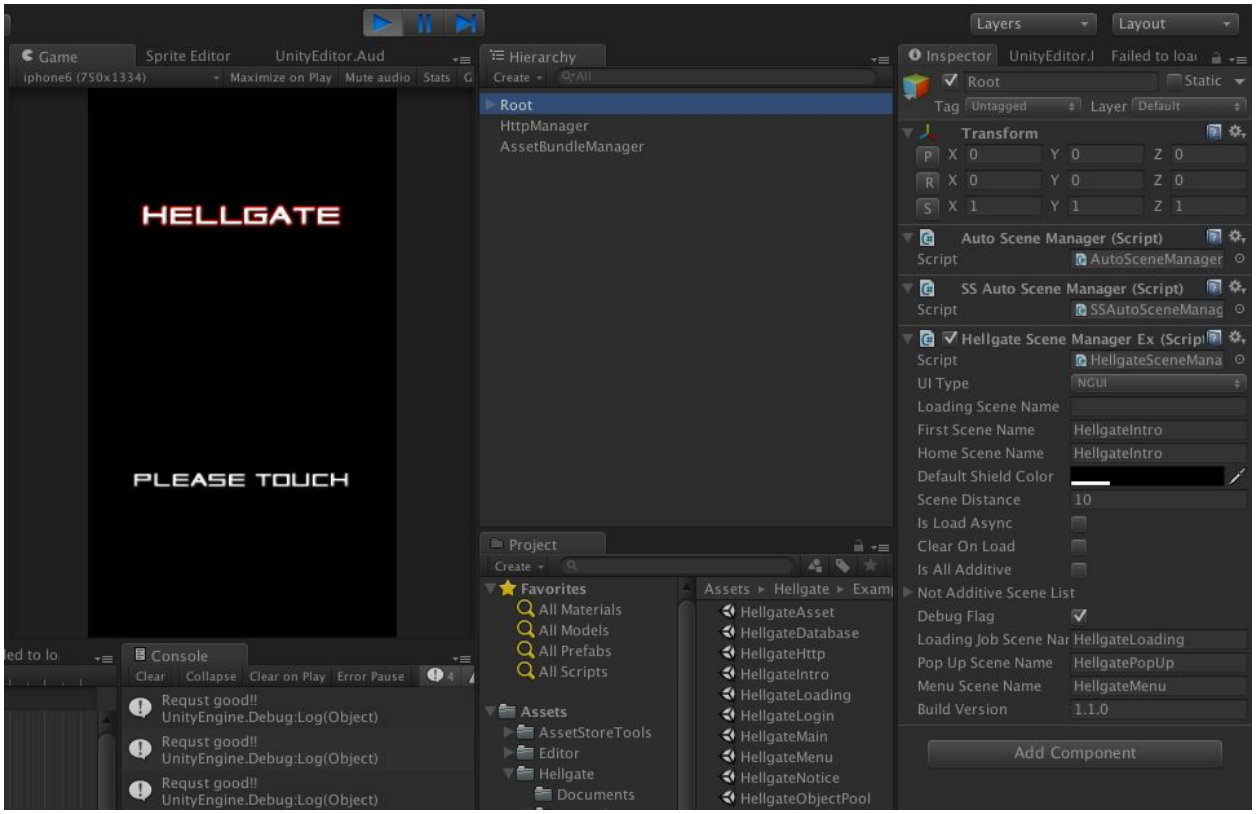
Assets/Hellgate/Examples/AssetBundle folder on the lets you set up Asset Labels.
- Asset -> hellgateasset.Unity
- Http -> hellgateasset.Unity
- Main -> hellgatemain.Unity
- Master -> hellgatemaster.Unity
- ObjectPool -> hellgateobjectpool.Unity
- Quest -> hellgatequest.Unity
- Scene -> hellgatescene.Unity



Screenshot 3.

3.3. Run

Assets/Hellgate/Examples/Scenes/Hellgate/Root.unity complete If you run. [Screenshot 4] is an run screen. Sometimes it can take a Time Over to the Dropbox server problems.



Screenshot 4.

4. AssetBundles

Hellgate Framework Resource download, load, management that is using the Unity AssetBundles. If the development Unity just seems to be part and to go beyond comprehension once. And Unity 5.0 which conforms to specifications less than AssetBundles version is not supported. And support Unity 5.0 or more AssetBundles, Less version is not supported.

4.1. Initial download

Application After running is the ability to download the Resource list. Download and Update resources the first run in many games.

Managing the Resource list Json from https://dl.dropboxusercontent.com/u/95277951/hellgate/8118ea697fed5008b03c4a67c4eeb5bb33ae2775/pc/resource.json

```
{
        "resource": {
           "major": 1,
           "minor": 1,
           "version": 1,
           "name": "resource"
        },
        "assetbundle": [
           {
              "name": "hellgatemaster",
              "version": 1
           },
           {
              "name": "hellgatemain",
              "version": 1
           },  ...
        ]
}
```

"major" : Resource all delete after downloading the begin.
"minor" : "assetbundle": [] when changing to a higher version of the sub-array version check and download / update

Ex)

```
// downloader instance
// response json from
AssetBundleInitialDownloader aDownloader = new AssetBundleInitialDownloader ("URL");
aDownloader.aEvent = CallbackDownloader;
aDownloader.Download ();

/// <summary>
/// Callbacks the downloader.
/// </summary>
/// <param name="status">Status.</param>
private void CallbackDownloader (AssetBundleInitalStatus status)
{
    if (status == AssetBundleInitalStatus.Start) {
        // download start
    } else if (status == AssetBundleInitalStatus.Over) {
        // not download list
    } else if (status == AssetBundleInitalStatus.HttpTimeover) {
        // request time over
    } else if (status == AssetBundleInitalStatus.HttpOver) {
        // request over
    } else if (status == AssetBundleInitalStatus.HttpError) {
        // request error
    } else if (status == AssetBundleInitalStatus.DownloadOver) {
        // download over
    }
}

/// <summary>
/// Update this instance.
/// Initial download progress bar and status
/// </summary>
void Update ()
{
    // AssetBundle status.
    if (aStatus == AssetBundleInitalStatus.Start && aDownloader != null) {
        slider.value = aDownloader.Progress;
        pLabel.text = aDownloader.SProgress;
        cLabel.text = aDownloader.CurretIndex + " / " + aDownloader.DownloadCount;
    }
}
```

4.2. AssetBundle Manager

Resource Load is simply possible through the AssetBundle Manager.In the Editor is run, the renewal without being downloaded again because the Load directly in its path without load in AssetBundle. Therefore, the Editor easy test.

Ex)

```
// sprite load(Single)
AssetBundleData data = new AssetBundleData ("AssetBundle Name");
data.objName = "Name";
data.type = typeof(Sprite);

AssetBundleManager.Instance.LoadAssetBundle (data, delegate(object obj) {
   Sprite temp = obj as Sprite;
   sprite.sprite2D = temp;
});

// prefabs load(Multi)
string[] loads = new string[] {
   "CubeGreen", "CubeYellow"
};

List<AssetBundleData> datas = new List<AssetBundleData> ();
for (int i = 0; i < loads.Length; i++) {
   AssetBundleData data = new AssetBundleData ("AssetBundle Name", loads [i]);
   datas.Add (data);
}

AssetBundleManager.Instance.LoadAssetBundle (datas, delegate(object obj) {
   List<object> objs = obj as List<object>;
   cubes = Util.GetListObjects<GameObject> (objs);

   Instantiate (cubes [ran.Next (0, loads.Length)]);
});
```

5. Http

Hellgate Framework provides the WWW utilities module. Simple UI makes it possible to work together.

5.1. Http Manager

Request/response processing is available through the Http Manager.

Ex)

```
        // Set base url(Static).
        HttpData.BASE_URL = "base url";

        // Set base url(Static).
        HttpData http = new HttpData ("uri");
//        http.popUp = true; // loading popup ui on/off
//        http.retry = 3; // retry count
//        http.timeout = 10; // time out second
        http.finishedDelegate = delegate (WWW www) {
            if (www == null) { // time over
            } else if (www.error != null) { // error
            } else {
                // done
            }
        };
        HttpManager.Instance.GET (http);
```

6. Object Pooling

Hellgate Framework provides the Object Pool. Object necessary to create and destroy non-way, This is a preview generated by an appropriate number of Active true/false that way.

6.1. Object Pool Manager

Through the Object Pool Manager creates and recycling Object.

Ex)

```
public GameObject prefab;

void Awake ()
{
    // if you want to create in advance
    ObjectPoolManager.Init (prefab);
}

 // object set active(true)
GameObject temp = ObjectPoolManager.Spawn (prefab);
// object delay 2 second set active(false)
ObjectPoolManager.DelayDespawn (temp, 2f);
// object set active(false)
 ObjectPoolManager.Despawn (temp);
```

7. Reflection

Hellgate Framework using Reflection can be a single line of code that creates the List, Dictionary Class instance. The reverse is also possible, of course.

7.1. Data Convert

Data Convert It is especially useful to utilize the Json data.
Json :
https://dl.dropboxusercontent.com/u/95277951/hellgate/8118ea697fed5008b03c4a67c4eeb5bb33ae2775/reflection.json

Ex)

```
HttpData http = new HttpData ("reflection", "json");
      http.finishedDelegate = delegate (WWW www) {
         if (www == null) { // time over
         } else if (www.error != null) { // error
         } else {
            // class reference
            // dictoinary -> HellgateReflectionDataEx instance
            HellgateReflectionDataEx data = Reflection.Convert<HellgateReflectionDataEx>
((IDictionary)MiniJSON.Json.Deserialize (www.text));

            // HellgateReflectionDataEx instance -> dictoinary
            IDictionary iDic = Reflection.Convert<HellgateReflectionDataEx> (data);
         }
      };
      HttpManager.Instance.GET (http);
```

8. Register/Encrypt

Hellgate Framework can be used to encrypt the PlayerPrefs (Game Session) function of the Unity. Encryption 3DES, MD5, and SHA1 algorithms use.

8.1. Register

PlayerPrefs class instead of using the Register class. Use the same method.

Ex)

```
Register.SetInt ("KeyInt", 10);
Register.SetFloat ("KeyFloat", 20.20f);
Register.SetString ("KeyString", "Hellgate Framework");

int num = Register.GetInt ("KeyInt", 0);
float f = Register.GetFloat ("keyFloat", 0);
string str = Register.GetString ("KeyString", "defaultValue");
```

8.2. Encrypt

When encryption, or post and url Http Request is helpful to security.

Ex)

```
// create manifest url
string encrypt = Encrypt.SHA1Key (BuildVersionBindings.GetBuildVersion () + "Hellgate");
//    Debug.Log (encrypt);
List<string> param = new List<string> ();
param.Add (BASE_URL);
param.Add (encrypt);
param.Add (Util.GetDevice ());
param.Add ("manifest");

string url = Http.CreateURL (param, "json");
HttpData hD = new HttpData (url);
hD.popUp = false;
hD.finishedDelegate = CallbackManifest;
HttpManager.Instance.GET (hD);
```

9. Scene Manager & Controller

Hellgate Framework, it can easily manage the large number of Scene by using the SS-System. Screen, PopUp, Menu, Loading Job management such as the Scene in various forms, and can be expressed, It boasts a simple usage.

SS-System Advance : http://anh-pham.appspot.com/sssystem/en/ssadvance.pdf

9.1. Scene Manager

Through the Scene Manager static instance, you can call Scene/Delete/Activate/Deactivate the like.

Ex)

```
// load scene
SceneManager.Instance.Screen ("scene name");

// load popup
SceneManager.Instance.PopUp ("popup name");

// close popup (stack)
SceneManager.Instance.Close ();

// load menu
SceneManager.Instance.LoadMenu ("menu name");

 // load message popup
SceneManager.Instance.PopUp ("Yes and No.", PopUpType.YesAndNo);
SceneManager.Instance.PopUp ("Okay.", PopUpType.Ok);

// load main menu
SceneManager.Instance.LoadMainMenu ();

// show main menu
SceneManager.Instance.ShowMainMenu ();

// hide main menu
SceneManager.Instance.HideMainMenu ();
```

9.2. Scene Controller

Scene Controller corresponds to the load through the scene Scene Manager.

Ex)

```
public class HellgateSceneController : SceneController
{
   public override void OnSet (object data)
   {
      base.OnSet (data);
      // init
   }

   public override void OnShow ()
   {
      base.OnShow ();
      // show
   }

   /// <summary>
   /// Android back key
   /// </summary>
   public override void OnKeyBack ()
   {
      // message popup
      base.Quit ("Exit ?");
   }

   ….
}
```

9.3. Loading Job Controller

Loading Job Controller will move Scene AssetBundles, Http, Intent and then automatically process the form, such as the Job Controller is sending Scene movement and Event.

Ex)

```
	// http request list
	List<HttpData> https = new List<HttpData> ();
	https.Add (new HttpData ("URL1"));
	https.Add (new HttpData ("URL2"));

	// load asset bundle list
	List<AssetBundleData> assetBundles = new List<AssetBundleData> ();
	assetBundles.Add (new AssetBundleData ("AssetBundle Name", "Name1", typeof (Sprite)));
	assetBundles.Add (new AssetBundleData ("AssetBundle Name", "Name2", typeof
(GameObject)));

	LoadingJobData data = new LoadingJobData ("Next Scene");
	data.https = https;
	data.assetBundles = assetBundles;

	// send intent
	data.PutExtra ("title", "Scene");
	data.PutExtra ("int", 10);
	data.PutExtra ("float", 20.20);

//	data.assetBundleasync = true; // load async assetbundle
//	data.popUp = true; // loading progress
//	data.nextScenePopUp = false; // the next scene is not the popup

	SceneManager.Instance.LoadingJob (data);
```

10. Sqlite3

Hellgate Framework supports the Sqlite3 OM and ORM. Query also available.

10.1. Sqlite3 ORM(Object-Relational Mapping)

Set the Class-based and uses a simple Attribute.
Wiki : https://en.wikipedia.org/wiki/Object-relational_mapping

Ex)

```
/// <summary>
/// table class.
/// </summary>
[Table]
public class HellgateTableName
{
    [Ignore]
    private int idx = 0;
    private int column1 = 0;
    private string column2 = "";

    public int Idx {
        get {
            return idx;
        }
    }

    public int Column1 {
        get {
            return column1;
        } set {
            column1 = value;
        }
    }

    public string Column2 {
        get {
            return column2;
        } set {
            column2 = value;
        }
```

```
}
    Query query = new Query ("DB name.db");

    // insert
    HellgateTableName data = new HellgateTableName ();
    data.Column1 = 10;
    data.Column2 = "test";
    query.INSERT<HellgateTableName> (data);

    // insert batch
    List<HellgateTableName> list = new List<HellgateTableName> ();

    HellgateTableName data1 = new HellgateTableName ();
    data1.Column1 = 10;
    data1.Column2 = "test1";

    HellgateTableName data2 = new HellgateTableName ();
    data2.Column1 = 20;
    data2.Column2 = "test2";

    list.Add (data1);
    list.Add (data2);
    query.INSERT_BATCH<HellgateTableName> (list);

    // Update
    HellgateTableName data = new HellgateTableName ();
    data.Column1 = 10;
    data.Column2 = "test";
//      query.UPDATE<HellgateTableName> (data, "add query");
    query.UPDATE<HellgateTableName> (data, "key", 1);

    // all select
    HellgateTableName[] list = query.SELECT<HellgateTableName> ();

    // select
//      HellgateTableName[] list = query.SELECT<HellgateTableName> ("add query");
    HellgateTableName[] list = query.SELECT<HellgateTableName> ("key", 1);

    // all delete
    query.DELETE<HellgateTableName> ();

    // delete
//      query.DELETE<HellgateTableName> ("add query");
    query.DELETE<HellgateTableName> ("key", 1);
    ....
```

10.2. Sqlite3 OM(Object Mapping)

Use based Dictionary and List

Ex)

```
Query query = new Query ("DB name.db");

// insert
Dictionary<string, object> data = new Dictionary<string, object> ();
data.Add ("column1", 1);
data.Add ("column2", "test");
query.INSERT ("table name", data);

// update
Dictionary<string, object> data = new Dictionary<string, object> ();
data.Add ("column1", 1);
data.Add ("column2", "test");
//    query.UPDATE ("table name", data, "add query");
query.UPDATE ("table name", data, "key", 1);

// all select
DataTable data = query.SELECT ("table name");

// select
//    DataTable data = query.SELECT ("table name", "add query");
DataTable data = query.SELECT ("table name", "key", 1);

// delete
//    query.DELETE ("table name", "add query");
query.DELETE ("table name", "key", 1);

// all delete
query.DELETE ("table name");

....
```

11. Runtime Build Version

Hellgate Framework, Get in a Build Version Runtime. Version Information You do not need to be managed as Config.

Ex)

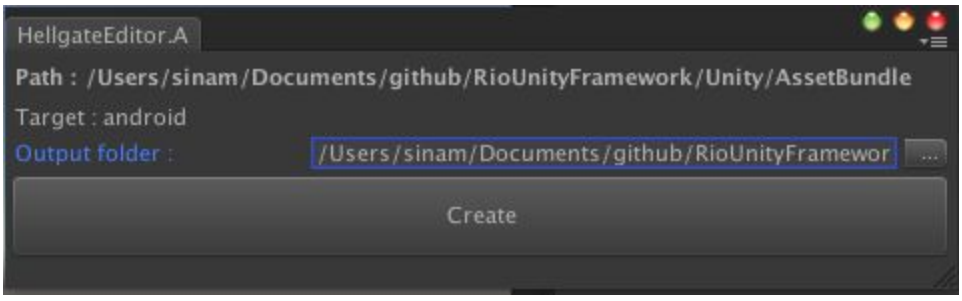
```
string version = BuildVersionBindings.GetBuildVersion ();
```

12. Editor

There are a number of the **Hellgate Framework** Editor**.** The path Unity Editor/Window/Hellgate

12.1. Build AssetBundles

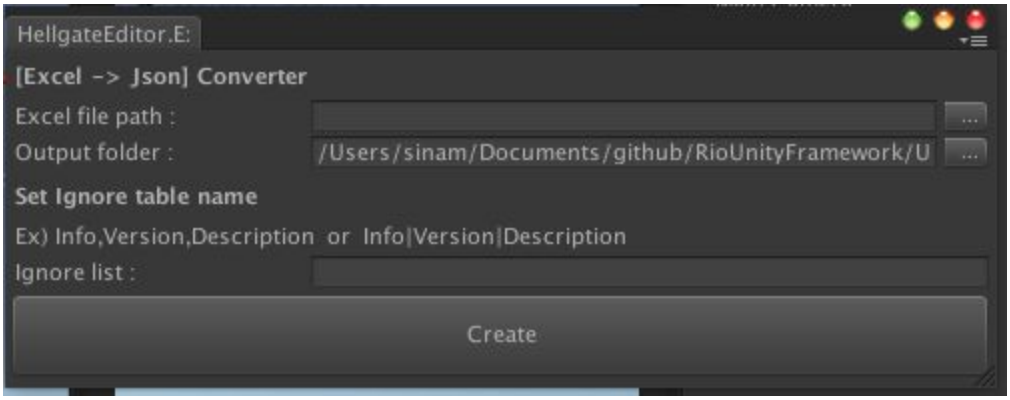
The AssetBundle as actually creating Editor The Path Unity Editor/Window/Hellgate/Build AssetBundles. After setting the Asset Labels, If you specify the Output folder, as shown in [Screenshot 5], AssetBundle will also be created. If the Target is set in accordance with the Editor of the Build Settings Platform only supports PC / Android / iOS.



Screenshot 5.

12.2. Json Converter for Excel

The Editor creating a Json using an Excel file. Useful for creating Master data. The Path Unity Editor/Window/Hellgate/Json Converter for Excel. Run the screen is the same as [Screenshot 6]. If you set only the Excel file path and Json Output folder path it is created. And [Screenshot 7] is a Excel Sheet form. Ignore setting is also available in the Sheet do can be utilized.



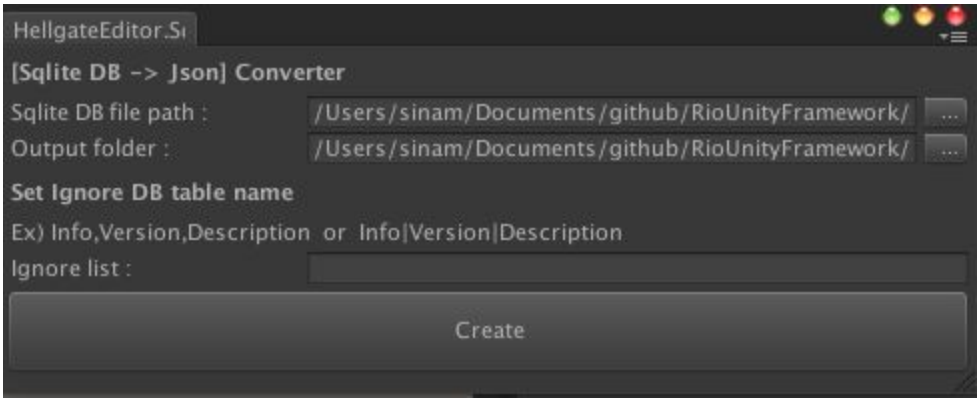
Screenshot 6.

idx	name	description	attack	defence	speed
1	aaa	aaaaa	10	10	1
2	bbb	bbbbb	20	20	2
3	ccc	ccccc	30	30	3
4	ddd	ddddd	40	40	4
5	eee	eeeee	50	50	5

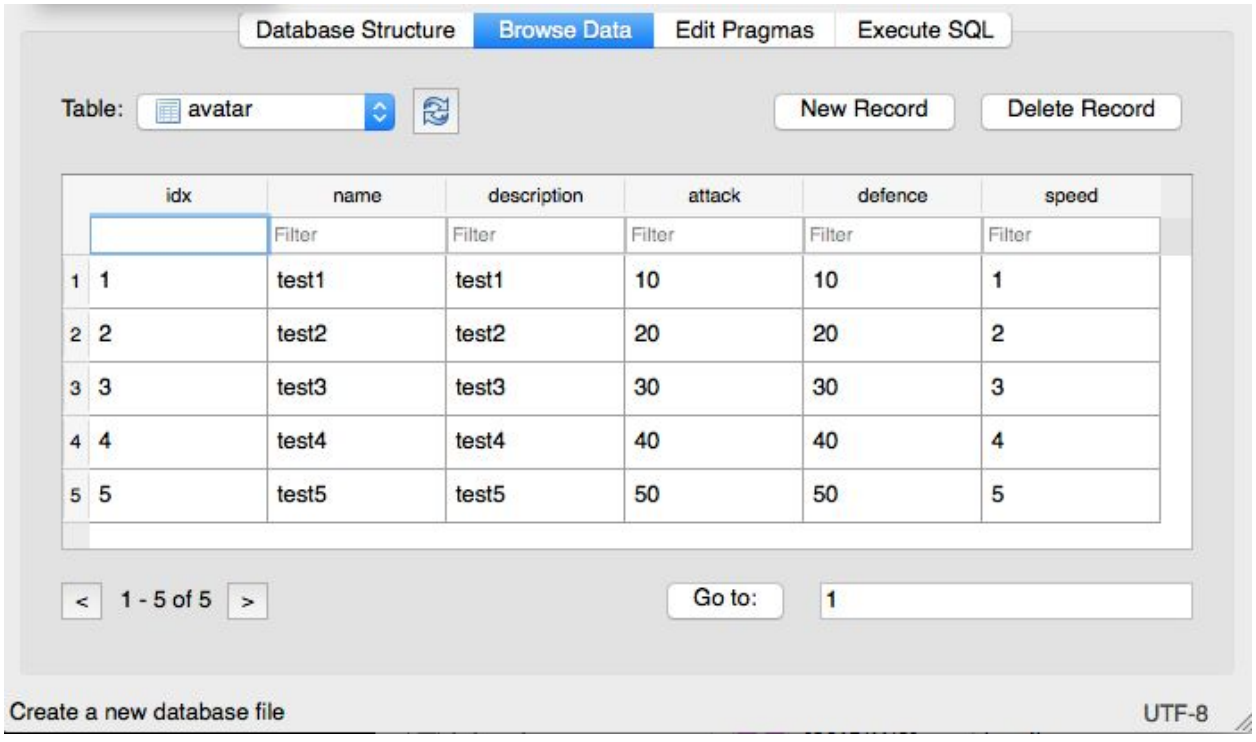
Screenshot 7.

12.3. Json Converter for Sqlite DB

The Editor creating a Json using an Sqlite DB file. Useful for creating Master data. The Path Unity Editor/Window/Hellgate/Json Converter for Sqlite DB. Run the screen is the same as [Screenshot 8]. If you set only the Sqlite DB file path and Json Output folder path it is created. And [Screenshot 7] is a Table form. Ignore setting is also available in the Table do can be utilized.



Screenshot 8.



Screenshot 9.

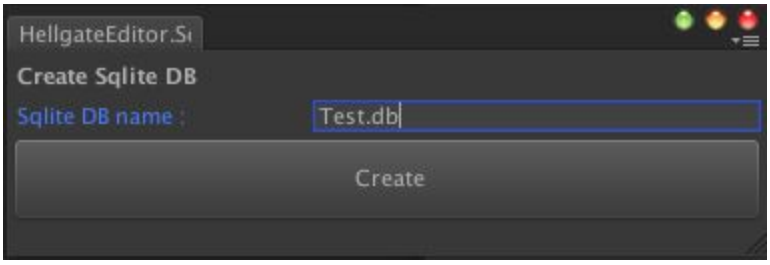
12.3. Create Sqlite DB

Table Auto Generated Attribute Class is set to read Sqlite DB Editor to create a Assets/StreamingAssets. (See Figure 10, Figure 11)

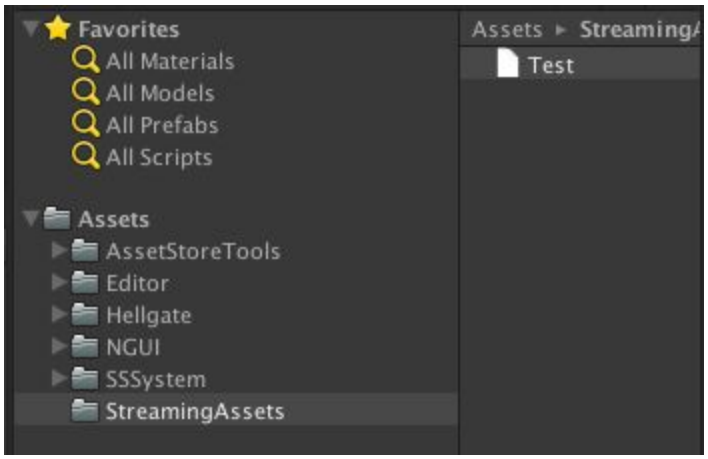
Ex)

```
[Table ("table_sample1", true)]
public class HellgateSampleTableEx
{
    [Column (SqliteDataConstraints.AI)]
    private int idx;
    [Column (new SqliteDataConstraints[] {
        SqliteDataConstraints.NOTNULL, SqliteDataConstraints.UNIQUE
    })]
    private string column1;
    protected float column2;
    // public will not be added to this column.
    public string temp;
}

[Table (true)]
public class Table_sample2
{
    [Column (SqliteDataConstraints.AI)]
    private int idx;
    private string column1;
    private float column2;
    private bool column3;
}
```



Screenshot 10.



Screenshot 11.