///original design by GrayMatter Tutorials

<https://www.youtube.com/watch?v=-zDZ5H47eH8&list=PL1talji0WeRPO7yOLTONufPFmQUz5ivfi>

Class Test

using UnityEngine;

using database;

public class test : MonoBehaviour {

void Start () {

GameData.Load();

GameData.actors.Add(new ActorData("Declan", "english man", 100f) { id = 1 });

GameData.actors.Add(new ActorData("Dean", "american man", 100f) { id = 2 });

GameData.actors.Add(new ActorData("Lee", "japanese man", 100f) { id = 3 });

GameData.items.Add(new ItemData("item 1", "this is item 1", 20) { id = 1 });

GameData.weapons.Add(new WeaponData("weapon 1", "this is weapon 1", 20,5) { id = 1 });

GameData.armor.Add(new ArmorData("armor 1", "this is armor 1", 20,20,20) { id = 1 });

for(int ac = 0;ac < GameData.actors.Length; ac++)

Debug.LogFormat("{0}: {1}",GameData.actors.GetAt(ac).name, GameData.actors.GetAt(ac).description);

for (int ac = 0; ac < GameData.items.Length; ac++)

Debug.LogFormat("{0}: {1}", GameData.items.GetAt(ac).name, GameData.items.GetAt(ac).description);

for (int ac = 0; ac < GameData.weapons.Length; ac++)

Debug.LogFormat("{0}: {1}", GameData.weapons.GetAt(ac).name, GameData.weapons.GetAt(ac).description);

for (int ac = 0; ac < GameData.armor.Length; ac++)

Debug.LogFormat("{0}: {1}", GameData.armor.GetAt(ac).name, GameData.armor.GetAt(ac).description);

GameData.Save();

}

}

Class ActorData

namespace database

{

public struct ActorData : IIdentity {

private int \_id;

private string \_name;

private string \_description;

private float \_curHP;

private float \_maxHP;

public int id { get { return \_id; } set { \_id = value; }}

public string name { get { return \_name; } set { \_name = value; }}

public string description { get { return \_description; } set { \_description = value; }}

public float curHP { get { return \_curHP; } set { \_curHP = value; }}

public float maxHP { get { return \_maxHP; } set { \_maxHP = value; }}

public ActorData(string name, string description, float maxHealth) {

\_id = -1;

\_name = name;

\_description = description;

\_curHP = \_maxHP = maxHealth;

}

public void ShowEditable(){

EditorGUILayout.LabelField(“ID:”, \_id.ToString());

\_name = EditorGUILayout.TextField(“Name:”, \_name);

\_description = EditorGUILayout.TextField(“Description:”, \_description);

\_curHP = EditorGUILayout.FloatField(“Current HP:”, \_curHP.ToString());

\_maxHP = EditorGUILayout.FloattField(“Max HP:”, \_maxHP.ToString());

}

}

}

Class ItemData

namespace database {

public struct ItemData : IIdentity {

private int \_id;

private string \_name;

private string \_description;

private double \_price;

public int id { get { return \_id; } set { \_id = value; } }

public string name { get { return \_name; } set { \_name = value; } }

public string description { get { return \_description; } set { \_description = value; } }

public double price { get { return \_price; } set { \_price = value; } }

public ItemData(string name, string description, double price) {

\_id = -1;

\_name = name;

\_description = description;

\_price = price;

}

public void ShowEditable(){

EditorGUILayout.LabelField(“ID:”, \_id.ToString());

\_name = EditorGUILayout.TextField(“Name:”, \_name);

\_description = EditorGUILayout.TextField(“Description:”, \_description);

\_price = EditorGUILayout.DoubleField(“Price:”, \_price.ToString());

}

}

}

Class WeaponData

namespace database{

public struct WeaponData : IIdentity {

private int \_id;

private string \_name;

private string \_description;

private double \_price;

private int \_damage;

public int id { get { return \_id; } set { \_id = value; } }

public string name { get { return \_name; } set { \_name = value; } }

public string description { get { return \_description; } set { \_description = value; } }

public double price { get { return \_price; } set { \_price = value; } }

public int damage { get { return \_damage; } set { \_damage = value; } }

public WeaponData(string name, string description, double price, int damage) {

\_id = -1;

\_name = name;

\_description = description;

\_price = price;

\_damage = damage;

}

public void ShowEditable(){

EditorGUILayout.LabelField(“ID:”, \_id.ToString());

\_name = EditorGUILayout.TextField(“Name:”, \_name);

\_description = EditorGUILayout.TextField(“Description:”, \_description);

\_price = EditorGUILayout.DoubleField(“Price:”, \_price.ToString());

\_damage = EditorGUILayout.IntField(“Damage:”, \_damage.ToString());

}

}

}

Class ArmorData

namespace database {

public struct ArmorData : IIdentity {

private int \_id;

private string \_name;

private string \_description;

private double \_price;

private int \_armorRate;

private int \_durability;

public int id { get { return \_id; } set { \_id = value; } }

public string name { get { return \_name; } set { \_name = value; } }

public string description { get { return \_description; } set { \_description = value; } }

public double price { get { return \_price; } set { \_price = value; } }

public int armorRate { get { return \_armorRate; } set { \_armorRate = value; } }

public int durability { get { return \_durability; } set { \_durability = value; } }

public ArmorData(string name, string description, int price, int armorRate, int durability) {

\_id = -1;

\_name = name;

\_description = description;

\_price = price;

\_armorRate = armorRate;

\_durability = durability;

}

public void ShowEditable(){

EditorGUILayout.LabelField(“ID:”, \_id.ToString());

\_name = EditorGUILayout.TextField(“Name:”, \_name);

\_description = EditorGUILayout.TextField(“Description:”, \_description);

\_price = EditorGUILayout.DoubleField(“Price:”, \_price.ToString());

\_armorRate = EditorGUILayout.IntField(“Armor Rate:”, \_armorRate.ToString());

\_durability = EditorGUILayout.IntField(“Durability:”, \_durability.ToString());

}

}

}

Interface IIdentity

namespace database

{

public interface IIdentity : IEditable {

string name { get; set; }

int id { get; set; }

}

}

Interface IEditable

namespace database

{

public interface IEditable {

void ShowEditable();

}

}

Class Database

using System.Collections.Generic;

using System.IO;

using System.Xml.Serialization;

namespace database {

[XmlRoot("Database")]

public class Database<T> where T : IIdentity {

private List<T> \_elements;

[XmlArray("Elements"), XmlArrayItem("Element")]

public List<T> elements { get { return \_elements; } set { \_elements = value; } }

public Database()

{

\_elements = new List<T>();

}

public void Add(T element) {

if (!Contains(element)) {

element.id = DetermineID();

\_elements.Add(element);

}

}

public void Clear() {

\_elements.Clear();

}

public int IndexOf(T element) {

return FindElement(element.id);

}

public int Length {

get { return \_elements.Count; } }

public int IndexOf(int id) {

return FindElement(id);

}

public int IndexOf(string name) {

return FindElement(name);

}

public T Get(int id) {

int i = FindElement(id);

if (i >= 0)

return \_elements[i];

return default(T);

}

public T Get(string name) {

int i = FindElement(name);

if (i >= 0)

return \_elements[i];

return default(T);

}

public T GetAt(int index) {

if (index >= 0 && index < \_elements.Count)

return \_elements[index];

return default(T);

}

public void Remove(T element) {

\_elements.Remove(element);

}

public void Remove(int id) {

int i = FindElement(id);

if (i >= 0)

\_elements.RemoveAt(i);

}

public void Remove(string name) {

int i = FindElement(name);

if (i >= 0)

\_elements.RemoveAt(i);

}

public void RemoveAt(int index) {

\_elements.RemoveAt(index);

}

public void Replace(int index, T element) {

if (index >= 0 && index < \_elements.Count)

\_elements[index] = element;

Add(element);

}

Private int DetermineID() {

int id=0;

for (int i = 0;i< \_elements.Count; i++) {

if(\_elements[i].id > id)

Id = \_elements[i].id;

}

return id+1;

}

private bool Contains(T element) {

return FindElement(element.id) >= 0;

}

private int FindElement(int id) {

for (int i = 0; i < \_elements.Count; i++)

{

if (\_elements[i].id == id)

return i;

}

return -1;

}

private int FindElement(string name) {

for (int i = 0; i < \_elements.Count; i++) {

if (elements[i].name == name) return i;

}

return -1;

}

public void Save<U>(string path) where U : Database<T> {

var serializer = new XmlSerializer(typeof(U));

using (var stream = new FileStream(path, FileMode.Create)) {

serializer.Serialize(stream, this); }

}

public static U Load<U>(string path) where U : Database<T>, new() {

if (File.Exists(path)) {

var serializer = new XmlSerializer(typeof(U));

using (var stream = new FileStream(path, FileMode.Open)) {

return serializer.Deserialize(stream) as U; }

}

return new U();

}

}

}

Class GameData

using System.IO;

using UnityEngine;

namespace database {

public static class GameData {

public static readonly string ACTOR\_PATH = Path.Combine(Application.dataPath + "/data", "actors.xml");

public static readonly string ITEM\_PATH = Path.Combine(Application.dataPath + "/data", "items.xml");

public static readonly string WEAPON\_PATH = Path.Combine(Application.dataPath + "/data", "weapons.xml");

public static readonly string ARMOR\_PATH = Path.Combine(Application.dataPath + "/data", "armor.xml");

public static Database<ActorData> actors;

public static Database<ItemData> items;

public static Database<WeaponData> weapons;

public static Database<ArmorData> armor;

public static void Save() {

actors.Save<Database<ActorData>>(ACTOR\_PATH);

items.Save<Database<ItemData>>(ITEM\_PATH);

weapons.Save<Database<WeaponData>>(WEAPON\_PATH);

armor.Save<Database<ArmorData>>(ARMOR\_PATH);

}

public static void Load() {

actors = Database<ActorData>.Load<Database<ActorData>>(ACTOR\_PATH);

items = Database<ItemData>.Load<Database<ItemData>>(ITEM\_PATH);

weapons = Database<WeaponData>.Load<Database<WeaponData>>(WEAPON\_PATH);

armor = Database<ArmorData>.Load<Database<ArmorData>>(ARMOR\_PATH);

}

}

}

Class DatabaseEditor

using UnityEditor;

using unityEngine;

namespace database {

public class DatabaseEditor: EditorWindow {

private const float WINDOW\_MIN\_WIDTH = 600f;

private const float WINDOW\_MIN\_HEIGHT = 800f;

private enum Tab { Actors, Items, Weapons, Armors }

private Tab \_tab;

private Vector2 \_selectionScrollPos;

private Vector2 \_editScrollView;

private bool \_editMode;

private int \_selectedIndex;

private IIdentity \_data;

[MenuItem("Window/Data Editor")]

public static void GetWindow() {

DatabaseEditor window = GetWindow<DatabaseEditor>("DataEditor", true);

window.minSize = new Vector2(WINDOW\_MIN\_WIDTH, WINDOW\_MIN\_HEIGHT);

}

private void OnEnable() {

\_tab = Tab.Actor;

Game.Load();

}

private void OnGUI() {

ShowTabs();

ShowContent();

}

private void Reset() {

\_selectedScrollPos = Vector2.zero;

\_editScrollView = Vector2.zero;

\_editMode = false;

\_selectedIndex = -1;

GUI.FocusControl(null);

}

private void ShowTabs() {

if (\_editMode)

GUILayout.Toolbar((int) \_tab, Enum.GetNames(typeof(Tab)));

else

\_tab = (Tab)GUILayout.Toolbar((int)\_tab, System.Enum.GetNames(typeof(Tab)));

}

private void ShowContent() {

EditorGUILayout.BeginHorizontal(GUILayout.ExpandHeight(true), GUILayout.ExpandWidth(true));

switch (\_tab) {

case Tab.Actors:

ShowSelection<Database< ActorData >, ActorData >(GameData.actors);

ShowEditView<Database < ActorData >, ActorData >(GameData.actors);

break;

case Tab.Items:

ShowSelection<Database< ItemData >, ItemData >(GameData.Items);

ShowEditView<Database < ItemData >, ItemData >( GameData.Items);

break;

case Tab.Weapons:

ShowSelection<Database< WeaponData >, WeaponData >( GameData.Weapons);

ShowEditView<Database < WeaponData >, WeaponData >( GameData.Weapons);

break;

case Tab.Armors:

ShowSelection<Database< ArmorData >, ArmorData >( GameData.Armors);

ShowEditView<Database < ArmorData >, ArmorData >( GameData.Armors);

break;

}

EditorGUILayout.EndHorizontal();

}

private void ShowSelection<T, U>(T database) where T: database<U> where U : IIdentity, new() {

EditorGUILayout.BeginVertical("Box", GUILayout.ExpandHeight(true), GUILayout.ExpandWidth(WINDOW\_MIN\_WIDTH/3));

\_selectionScrollPos = EditorGUILayout.BeginScrollView(\_selectionScrollPos);

for (int i = 0; i < database.Length; i++) {

EditorGUILayount.BeginHorizontal(GUILayout.ExpandWidth(true), GUILayout.ExpandHeight(20f));

GUILayout.Label(string.Format(“[{0}] {1}”, database.GetAt(i).id, database.GetAt(i).name));

if (!\_editMode) {

if (GUILayout.Button(“Edit”, GUILayout.Width(40f))) {

\_editMode = true;

\_selectedIndex = i;

\_data =database.GetAt(i);

}

if (GUILayout.Button(“Delete”, GUILayout.Width(50f))) {

database.RemoveAt(i);

SaveChanges();

break;

}

}

EditorGUILayout.EndHorizontal();

}

EditorGUILayout.EndScrollView();

if (GUILayout.Button(“Add New”, GUILayout.ExpandWidth(true))) {

\_selectedIndex = -1;

\_editMode = true;

\_data = new U();

}

EditorGUILayout.EndVertical();

}

private void ShowEditView<T, U>(T database) where T: database<U> where U : IIdentity {

EditorGUILayout.BeginVertical("Box", GUILayout.ExpandHeight(true), GUILayout.ExpandWidth(true));

if (\_editMode) {

\_editScrollView = EditorGUILayout.BeginScrollView(\_editScrollView, GUILayout.ExpandWidth(true), GUILayout.ExpandHeight(true));

EditorGUILayout.EndScrollView();

\_data.ShowEditable();

EditorGUILayout.BeginHorizontal(GUILayout.ExpandWidth(true));

if (GUILayout.Button(“Save”)) {

if(\_selectedIndex == -1)

database.Add((U)\_data);

else

database.Replace(\_selectedIndex, (U)\_data);

SaveChanges();

}

if (GUILayout.Button(“Cancel”) {

Reset();

}

EditorGUILayout.EndHorizontal();

}

EditorGUILayout.EndVertical();

}

private void SaveChanges() {

GameData.Save();

Reset();

}

}