**HỆ THỐNG ĐA NGÔN NGỮ: ANIMAL RESCUE ADVENTURE**

**TỔNG QUAN HỆ THỐNG**

Hệ thống đa ngôn ngữ sẽ cho phép:

1. **Hỗ trợ nhiều ngôn ngữ** với tiếng Việt là mặc định
2. **Chuyển đổi ngôn ngữ nhanh chóng** mà không cần khởi động lại game
3. **Dễ dàng thêm ngôn ngữ mới** trong các bản cập nhật
4. **Phù hợp với từng vùng địa lý** dựa trên cài đặt thiết bị

**DANH SÁCH NGÔN NGỮ HỖ TRỢ**

Phiên bản đầu tiên sẽ hỗ trợ các ngôn ngữ sau:

1. Tiếng Việt (mặc định)
2. Tiếng Anh
3. Tiếng Trung Quốc (Giản thể)
4. Tiếng Thái

Các ngôn ngữ tiềm năng để bổ sung trong tương lai:

* Tiếng Indonesia
* Tiếng Nhật
* Tiếng Hàn
* Tiếng Tây Ban Nha
* Tiếng Pháp

**KIẾN TRÚC HỆ THỐNG ĐA NGÔN NGỮ**

**1. Cấu trúc file**

Assets/

|-- Resources/

| |-- Localization/

| |-- vi.json (Tiếng Việt)

| |-- en.json (Tiếng Anh)

| |-- zh-Hans.json (Tiếng Trung Giản thể)

| |-- th.json (Tiếng Thái)

|

|-- Scripts/

|-- Localization/

|-- LocalizationManager.cs

|-- LocalizedText.cs

|-- LocalizedImage.cs

|-- LocalizedAudio.cs

|-- LocalizationSettings.cs

**2. Ví dụ nội dung file ngôn ngữ (JSON)**

**vi.json (Tiếng Việt)**

{

"metadata": {

"languageCode": "vi",

"languageName": "Tiếng Việt",

"version": "1.0",

"fontSettings": {

"defaultFont": "NotoSansVi",

"fontSize": 1.0

}

},

"ui": {

"mainMenu": {

"play": "Chơi Ngay",

"settings": "Cài Đặt",

"sanctuary": "Khu Bảo Tồn",

"shop": "Cửa Hàng",

"exit": "Thoát"

},

"gameHud": {

"distance": "Khoảng cách: {0}m",

"score": "Điểm: {0}",

"coins": "Xu: {0}"

},

"pauseMenu": {

"resume": "Tiếp Tục",

"restart": "Chơi Lại",

"settings": "Cài Đặt",

"quit": "Thoát"

},

"settings": {

"music": "Âm Nhạc",

"sfx": "Hiệu Ứng Âm Thanh",

"language": "Ngôn Ngữ",

"parentalControls": "Kiểm Soát Của Phụ Huynh"

}

},

"gameplay": {

"tutorial": {

"swipeUp": "Vuốt lên để nhảy",

"swipeDown": "Vuốt xuống để trượt",

"swipeLeft": "Vuốt trái để di chuyển sang trái",

"swipeRight": "Vuốt phải để di chuyển sang phải",

"tap": "Chạm để tương tác"

},

"powerups": {

"shield": "Lá Chắn Bảo Vệ",

"magnet": "Nam Châm Xu",

"speedBoost": "Tăng Tốc",

"doubleScore": "Điểm Đôi"

},

"messages": {

"gameOver": "Kết Thúc!",

"newHighScore": "Điểm Cao Mới!",

"animalRescued": "Đã Cứu {0}!",

"levelUp": "Lên Cấp! Ranger Cấp {0}",

"dailyReward": "Phần Thưởng Hàng Ngày! Ngày {0}"

}

},

"animals": {

"monkey": {

"name": "Khỉ",

"description": "Khỉ sống trong rừng nhiệt đới và ăn quả cây.",

"funFact": "Khỉ có thể sử dụng công cụ đơn giản!",

"education\_4\_7": "Khỉ thích leo trèo trên cây và ăn chuối.",

"education\_8\_11": "Khỉ là động vật có vú sống theo đàn và có ngón tay cái đối diện.",

"education\_12\_15": "Khỉ thuộc bộ Linh trưởng, có bộ não phát triển và khả năng giải quyết vấn đề."

},

"dolphin": {

"name": "Cá Heo",

"description": "Cá heo là động vật thông minh sống trong đại dương.",

"funFact": "Cá heo ngủ với một nửa não bộ tỉnh táo!",

"education\_4\_7": "Cá heo bơi rất nhanh và thích chơi đùa trong nước.",

"education\_8\_11": "Cá heo là động vật có vú biển, giao tiếp bằng âm thanh và sóng âm.",

"education\_12\_15": "Cá heo sử dụng định vị bằng âm thanh (echolocation) để tìm thức ăn và điều hướng trong nước."

}

},

"biomes": {

"jungle": "Rừng Nhiệt Đới",

"ocean": "Đại Dương",

"savanna": "Savanna",

"arctic": "Vùng Cực",

"mountain": "Núi Cao"

},

"quests": {

"daily": {

"runDistance": "Chạy {0} mét",

"collectCoins": "Thu thập {0} xu",

"rescueAnimals": "Cứu {0} động vật",

"completePuzzles": "Hoàn thành {0} câu đố"

},

"weekly": {

"rescueSpecies": "Cứu tất cả loài {0}",

"reachScore": "Đạt {0} điểm trong {1}",

"upgradeHabitat": "Nâng cấp môi trường {0}"

}

}

}

**en.json (Tiếng Anh)**

{

"metadata": {

"languageCode": "en",

"languageName": "English",

"version": "1.0",

"fontSettings": {

"defaultFont": "NotoSansEn",

"fontSize": 1.0

}

},

"ui": {

"mainMenu": {

"play": "Play Now",

"settings": "Settings",

"sanctuary": "Sanctuary",

"shop": "Shop",

"exit": "Exit"

},

"gameHud": {

"distance": "Distance: {0}m",

"score": "Score: {0}",

"coins": "Coins: {0}"

},

"pauseMenu": {

"resume": "Resume",

"restart": "Restart",

"settings": "Settings",

"quit": "Quit"

},

"settings": {

"music": "Music",

"sfx": "Sound Effects",

"language": "Language",

"parentalControls": "Parental Controls"

}

},

"gameplay": {

"tutorial": {

"swipeUp": "Swipe up to jump",

"swipeDown": "Swipe down to slide",

"swipeLeft": "Swipe left to move left",

"swipeRight": "Swipe right to move right",

"tap": "Tap to interact"

},

"powerups": {

"shield": "Shield",

"magnet": "Coin Magnet",

"speedBoost": "Speed Boost",

"doubleScore": "Double Score"

},

"messages": {

"gameOver": "Game Over!",

"newHighScore": "New High Score!",

"animalRescued": "{0} Rescued!",

"levelUp": "Level Up! Ranger Level {0}",

"dailyReward": "Daily Reward! Day {0}"

}

},

"animals": {

"monkey": {

"name": "Monkey",

"description": "Monkeys live in tropical forests and eat fruits.",

"funFact": "Monkeys can use simple tools!",

"education\_4\_7": "Monkeys like to climb trees and eat bananas.",

"education\_8\_11": "Monkeys are mammals that live in groups and have opposable thumbs.",

"education\_12\_15": "Monkeys belong to the order Primates, have developed brains and problem-solving abilities."

},

"dolphin": {

"name": "Dolphin",

"description": "Dolphins are intelligent animals living in the ocean.",

"funFact": "Dolphins sleep with half of their brain awake!",

"education\_4\_7": "Dolphins swim very fast and like to play in the water.",

"education\_8\_11": "Dolphins are marine mammals that communicate using sounds and sound waves.",

"education\_12\_15": "Dolphins use echolocation to find food and navigate in water."

}

},

"biomes": {

"jungle": "Jungle",

"ocean": "Ocean",

"savanna": "Savanna",

"arctic": "Arctic",

"mountain": "Mountain"

},

"quests": {

"daily": {

"runDistance": "Run {0} meters",

"collectCoins": "Collect {0} coins",

"rescueAnimals": "Rescue {0} animals",

"completePuzzles": "Complete {0} puzzles"

},

"weekly": {

"rescueSpecies": "Rescue all {0} species",

"reachScore": "Reach {0} points in {1}",

"upgradeHabitat": "Upgrade {0} habitat"

}

}

}

**TRIỂN KHAI HỆ THỐNG QUẢN LÝ ĐA NGÔN NGỮ**

**1. LocalizationManager**

using System.Collections.Generic;

using UnityEngine;

using System.IO;

using Newtonsoft.Json;

using System;

public class LocalizationManager : MonoBehaviour

{

// Singleton instance

private static LocalizationManager \_instance;

public static LocalizationManager Instance => \_instance;

// Event được kích hoạt khi ngôn ngữ thay đổi

public event Action OnLanguageChanged;

// Dictionary lưu dữ liệu ngôn ngữ

private Dictionary<string, Dictionary<string, object>> \_localizedData;

// Ngôn ngữ hiện tại

private string \_currentLanguage = "vi"; // Mặc định là tiếng Việt

// Danh sách các ngôn ngữ đã được tải

private List<string> \_availableLanguages = new List<string>();

// Thông tin meta của ngôn ngữ hiện tại

private Dictionary<string, object> \_currentMetadata;

private void Awake()

{

// Singleton setup

if (\_instance == null)

{

\_instance = this;

DontDestroyOnLoad(gameObject);

InitializeLocalization();

}

else

{

Destroy(gameObject);

}

}

private void InitializeLocalization()

{

\_localizedData = new Dictionary<string, Dictionary<string, object>>();

// Tìm tất cả các file ngôn ngữ có sẵn

TextAsset[] languageFiles = Resources.LoadAll<TextAsset>("Localization");

foreach (TextAsset file in languageFiles)

{

try

{

// Parse JSON file

Dictionary<string, object> langData = JsonConvert.DeserializeObject<Dictionary<string, object>>(file.text);

// Lấy mã ngôn ngữ từ metadata

Dictionary<string, object> metadata = JsonConvert.DeserializeObject<Dictionary<string, object>>(langData["metadata"].ToString());

string languageCode = metadata["languageCode"].ToString();

\_localizedData.Add(languageCode, langData);

\_availableLanguages.Add(languageCode);

Debug.Log($"Loaded language: {languageCode}");

}

catch (Exception e)

{

Debug.LogError($"Error loading language file {file.name}: {e.Message}");

}

}

// Thiết lập ngôn ngữ mặc định từ cài đặt hoặc ngôn ngữ hệ thống

string savedLanguage = PlayerPrefs.GetString("SelectedLanguage", "");

if (!string.IsNullOrEmpty(savedLanguage) && \_availableLanguages.Contains(savedLanguage))

{

SetLanguage(savedLanguage);

}

else

{

// Xác định ngôn ngữ hệ thống

string systemLanguage = GetSystemLanguageCode();

// Kiểm tra xem có hỗ trợ ngôn ngữ hệ thống không

if (\_availableLanguages.Contains(systemLanguage))

{

SetLanguage(systemLanguage);

}

else

{

// Mặc định sử dụng tiếng Việt

SetLanguage("vi");

}

}

}

private string GetSystemLanguageCode()

{

// Chuyển đổi từ SystemLanguage của Unity sang mã ISO 639

switch (Application.systemLanguage)

{

case SystemLanguage.Vietnamese:

return "vi";

case SystemLanguage.English:

return "en";

case SystemLanguage.Chinese:

case SystemLanguage.ChineseSimplified:

return "zh-Hans";

case SystemLanguage.Thai:

return "th";

default:

return "vi"; // Mặc định tiếng Việt

}

}

public void SetLanguage(string languageCode)

{

if (!\_availableLanguages.Contains(languageCode))

{

Debug.LogWarning($"Language {languageCode} not available. Using default language.");

languageCode = "vi"; // Mặc định là tiếng Việt nếu không tìm thấy

}

\_currentLanguage = languageCode;

// Lưu cài đặt ngôn ngữ

PlayerPrefs.SetString("SelectedLanguage", languageCode);

PlayerPrefs.Save();

// Cập nhật metadata

\_currentMetadata = JsonConvert.DeserializeObject<Dictionary<string, object>>(

\_localizedData[\_currentLanguage]["metadata"].ToString());

// Áp dụng cài đặt font

ApplyFontSettings();

// Thông báo cho các component rằng ngôn ngữ đã thay đổi

OnLanguageChanged?.Invoke();

Debug.Log($"Language set to: {\_currentMetadata["languageName"]}");

}

private void ApplyFontSettings()

{

// Lấy cài đặt font từ metadata

Dictionary<string, object> fontSettings = JsonConvert.DeserializeObject<Dictionary<string, object>>(

\_currentMetadata["fontSettings"].ToString());

string fontName = fontSettings["defaultFont"].ToString();

float fontSize = Convert.ToSingle(fontSettings["fontSize"]);

// Áp dụng font cho toàn bộ UI - thực hiện trong các component LocalizedText

}

public string GetText(string key)

{

// Phân tích key theo cấu trúc (vd: "ui.mainMenu.play")

string[] keyParts = key.Split('.');

Dictionary<string, object> currentData = \_localizedData[\_currentLanguage];

// Duyệt qua các phần của key để tìm giá trị

for (int i = 0; i < keyParts.Length - 1; i++)

{

if (currentData.ContainsKey(keyParts[i]))

{

currentData = JsonConvert.DeserializeObject<Dictionary<string, object>>(

currentData[keyParts[i]].ToString());

}

else

{

Debug.LogWarning($"Localization key not found: {key}");

return key; // Trả về key nếu không tìm thấy

}

}

// Lấy giá trị cuối cùng

string finalKey = keyParts[keyParts.Length - 1];

if (currentData.ContainsKey(finalKey))

{

return currentData[finalKey].ToString();

}

else

{

Debug.LogWarning($"Localization key not found: {key}");

return key; // Trả về key nếu không tìm thấy

}

}

public string GetText(string key, params object[] args)

{

// Lấy chuỗi và thay thế các tham số

string text = GetText(key);

if (args != null && args.Length > 0)

{

try

{

return string.Format(text, args);

}

catch (Exception e)

{

Debug.LogError($"Error formatting localized text: {key} - {e.Message}");

return text;

}

}

return text;

}

public string GetAnimalInfo(string animalId, string infoType, string ageGroup = null)

{

string key = $"animals.{animalId}.{infoType}";

if (!string.IsNullOrEmpty(ageGroup))

{

key = $"animals.{animalId}.education\_{ageGroup}";

}

return GetText(key);

}

public List<KeyValuePair<string, string>> GetAvailableLanguages()

{

List<KeyValuePair<string, string>> languages = new List<KeyValuePair<string, string>>();

foreach (string langCode in \_availableLanguages)

{

Dictionary<string, object> metadata = JsonConvert.DeserializeObject<Dictionary<string, object>>(

\_localizedData[langCode]["metadata"].ToString());

string langName = metadata["languageName"].ToString();

languages.Add(new KeyValuePair<string, string>(langCode, langName));

}

return languages;

}

public string GetCurrentLanguageCode()

{

return \_currentLanguage;

}

public string GetCurrentLanguageName()

{

return \_currentMetadata["languageName"].ToString();

}

}

**2. LocalizedText (Component cho UI Text)**

using UnityEngine;

using TMPro;

using UnityEngine.UI;

[RequireComponent(typeof(TextMeshProUGUI))]

public class LocalizedText : MonoBehaviour

{

public string localizationKey;

public object[] formatParams;

private TextMeshProUGUI \_textComponent;

private void Awake()

{

\_textComponent = GetComponent<TextMeshProUGUI>();

}

private void OnEnable()

{

// Đăng ký sự kiện khi ngôn ngữ thay đổi

if (LocalizationManager.Instance != null)

{

LocalizationManager.Instance.OnLanguageChanged += UpdateText;

}

UpdateText();

}

private void OnDisable()

{

// Hủy đăng ký sự kiện

if (LocalizationManager.Instance != null)

{

LocalizationManager.Instance.OnLanguageChanged -= UpdateText;

}

}

public void UpdateText()

{

if (string.IsNullOrEmpty(localizationKey) || LocalizationManager.Instance == null)

return;

// Cập nhật text

if (formatParams != null && formatParams.Length > 0)

{

\_textComponent.text = LocalizationManager.Instance.GetText(localizationKey, formatParams);

}

else

{

\_textComponent.text = LocalizationManager.Instance.GetText(localizationKey);

}

// Áp dụng font nếu cần

ApplyFontSettings();

}

private void ApplyFontSettings()

{

// Nếu có font riêng cho ngôn ngữ hiện tại

string languageCode = LocalizationManager.Instance.GetCurrentLanguageCode();

TMP\_FontAsset font = Resources.Load<TMP\_FontAsset>($"Fonts/{languageCode}");

if (font != null)

{

\_textComponent.font = font;

}

}

public void SetFormatParams(params object[] args)

{

formatParams = args;

UpdateText();

}

}

**3. LocalizedImage (Cho các hình ảnh theo ngôn ngữ)**

using UnityEngine;

using UnityEngine.UI;

[RequireComponent(typeof(Image))]

public class LocalizedImage : MonoBehaviour

{

public string imageKey;

private Image \_imageComponent;

private void Awake()

{

\_imageComponent = GetComponent<Image>();

}

private void OnEnable()

{

if (LocalizationManager.Instance != null)

{

LocalizationManager.Instance.OnLanguageChanged += UpdateImage;

}

UpdateImage();

}

private void OnDisable()

{

if (LocalizationManager.Instance != null)

{

LocalizationManager.Instance.OnLanguageChanged -= UpdateImage;

}

}

public void UpdateImage()

{

if (string.IsNullOrEmpty(imageKey) || LocalizationManager.Instance == null)

return;

string languageCode = LocalizationManager.Instance.GetCurrentLanguageCode();

Sprite localizedSprite = Resources.Load<Sprite>($"Localization/Images/{languageCode}/{imageKey}");

if (localizedSprite != null)

{

\_imageComponent.sprite = localizedSprite;

}

else

{

// Sử dụng sprite mặc định nếu không tìm thấy

Sprite defaultSprite = Resources.Load<Sprite>($"Localization/Images/default/{imageKey}");

if (defaultSprite != null)

{

\_imageComponent.sprite = defaultSprite;

}

}

}

}

**4. LocalizedAudio (Cho âm thanh theo ngôn ngữ)**

using UnityEngine;

[RequireComponent(typeof(AudioSource))]

public class LocalizedAudio : MonoBehaviour

{

public string audioKey;

private AudioSource \_audioSource;

private void Awake()

{

\_audioSource = GetComponent<AudioSource>();

}

private void OnEnable()

{

if (LocalizationManager.Instance != null)

{

LocalizationManager.Instance.OnLanguageChanged += UpdateAudio;

}

UpdateAudio();

}

private void OnDisable()

{

if (LocalizationManager.Instance != null)

{

LocalizationManager.Instance.OnLanguageChanged -= UpdateAudio;

}

}

public void UpdateAudio()

{

if (string.IsNullOrEmpty(audioKey) || LocalizationManager.Instance == null)

return;

string languageCode = LocalizationManager.Instance.GetCurrentLanguageCode();

AudioClip localizedClip = Resources.Load<AudioClip>($"Localization/Audio/{languageCode}/{audioKey}");

if (localizedClip != null)

{

\_audioSource.clip = localizedClip;

}

else

{

// Sử dụng clip mặc định nếu không tìm thấy

AudioClip defaultClip = Resources.Load<AudioClip>($"Localization/Audio/default/{audioKey}");

if (defaultClip != null)

{

\_audioSource.clip = defaultClip;

}

}

}

public void Play()

{

if (\_audioSource.clip != null)

{

\_audioSource.Play();

}

}

}

**GIAO DIỆN CÀI ĐẶT NGÔN NGỮ**

// LanguageSettingsUI.cs

using UnityEngine;

using UnityEngine.UI;

using TMPro;

using System.Collections.Generic;

public class LanguageSettingsUI : MonoBehaviour

{

[SerializeField] private Transform languageButtonContainer;

[SerializeField] private Button languageButtonPrefab;

private List<KeyValuePair<string, string>> availableLanguages;

private void OnEnable()

{

PopulateLanguageOptions();

}

private void PopulateLanguageOptions()

{

// Xóa các button cũ

foreach (Transform child in languageButtonContainer)

{

Destroy(child.gameObject);

}

// Lấy danh sách ngôn ngữ khả dụng

availableLanguages = LocalizationManager.Instance.GetAvailableLanguages();

// Lấy mã ngôn ngữ hiện tại

string currentLanguageCode = LocalizationManager.Instance.GetCurrentLanguageCode();

// Tạo button cho mỗi ngôn ngữ

foreach (var language in availableLanguages)

{

Button languageButton = Instantiate(languageButtonPrefab, languageButtonContainer);

// Thiết lập text

TMP\_Text buttonText = languageButton.GetComponentInChildren<TMP\_Text>();

if (buttonText != null)

{

buttonText.text = language.Value; // Tên ngôn ngữ hiển thị

}

// Thiết lập hình ảnh cờ quốc gia (nếu có)

Image flagImage = languageButton.transform.Find("FlagImage")?.GetComponent<Image>();

if (flagImage != null)

{

Sprite flagSprite = Resources.Load<Sprite>($"Localization/Flags/{language.Key}");

if (flagSprite != null)

{

flagImage.sprite = flagSprite;

flagImage.gameObject.SetActive(true);

}

else

{

flagImage.gameObject.SetActive(false);

}

}

// Đánh dấu ngôn ngữ hiện tại

GameObject selectionIndicator = languageButton.transform.Find("SelectionIndicator")?.gameObject;

if (selectionIndicator != null)

{

selectionIndicator.SetActive(language.Key == currentLanguageCode);

}

// Đăng ký sự kiện click

string languageCode = language.Key; // Cần biến cục bộ để capture đúng giá trị

languageButton.onClick.AddListener(() => SelectLanguage(languageCode));

}

}

private void SelectLanguage(string languageCode)

{

// Thiết lập ngôn ngữ mới

LocalizationManager.Instance.SetLanguage(languageCode);

// Cập nhật UI

PopulateLanguageOptions();

}

}

**XỬ LÝ FONT CHO CÁC NGÔN NGỮ ĐẶC BIỆT**

// FontManager.cs

using UnityEngine;

using TMPro;

using System.Collections.Generic;

public class FontManager : MonoBehaviour

{

[System.Serializable]

public class LanguageFont

{

public string languageCode;

public TMP\_FontAsset font;

public float defaultSize = 1.0f;

public float lineSpacing = 1.0f;

}

[SerializeField] private List<LanguageFont> languageFonts = new List<LanguageFont>();

[SerializeField] private TMP\_FontAsset defaultFont;

private static FontManager \_instance;

public static FontManager Instance => \_instance;