using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using ManhattanMorning.Model;

using ManhattanMorning.Controller;

using ManhattanMorning.Misc;

using ManhattanMorning.Model.MenuObject;

using ManhattanMorning.Model.MenuObject.SpecificControls;

namespace ManhattanMorning.Misc

{

/// <summary>

/// Stores all the content for a menu (MainMenu or Ingame-Menu)

/// </summary>

public class MenuInstance

{

#region Properties

/// <summary>

/// A Menu Object, that is the actual RecentMenu, that should be drawn.

/// </summary>

public MenuObject RecentMenu { get {return this.recentMenu;} set {this.recentMenu = value;}}

/// <summary>

/// A List Of DrawableObjects. This is the list, that has to be drawn by the RecentMenu

/// </summary>

public LayerList<LayerInterface> MenuObjectList { get { return this.menuObjectList; } set { this.menuObjectList = value; } }

#endregion

#region Members

/// <summary>

/// A Menu Object, that is the actual RecentMenu, that should be drawn.

/// </summary>

private MenuObject recentMenu;

/// <summary>

/// A List Of DrawableObjects. This is the list, that has to be drawn by the RecentMenu

/// </summary>

private LayerList<LayerInterface> menuObjectList;

#endregion

#region Initialization

/// <summary>

/// Creates a MenuInstance.

/// </summary>

/// <param name="menuObjectList"></param>

public MenuInstance()

{

// Loading the Menu. You want to draw on the screen

recentMenu = StorageManager.Instance.LoadMenu("MainMenu");

// Creating a list, that will contain the items to be drawn

menuObjectList = new LayerList<LayerInterface>();

addDrawableMenuObjectsToList(recentMenu, menuObjectList);

}

#endregion

#region Methods

public void addDrawableMenuObjectsToList(MenuObject recentMenu, LayerList<LayerInterface> menuObjectsList)

{

// Adding the Headline from the recentMenu to the list

//if (!menuObjectList.Contains(recentMenu.Headline))

menuObjectList.Add(recentMenu.Headline);

// Adding the Background

//if (!menuObjectList.Contains(recentMenu.Background))

menuObjectList.Add(recentMenu.Background);

// The 3 differnt controls, to distinguish the different control in menu.

Switch controlSwitch = null;

Button controlButton = null;

Slider controlSlider = null;

// For every Item in the List of Controls of the recentMenu, we decide wether it is a Button, a Switch or a Slider

// For the specific Controls, we add different Item in the List

for (int i = 0; i < recentMenu.ControlsAndSubmenues.Count(); i++)

{

// We decide, whether the recent item of the ControlsAndSubmenues is a Control item

if (recentMenu.ControlsAndSubmenues.ElementAt(i) is Controls)

{

if (((Controls)recentMenu.ControlsAndSubmenues.ElementAt(i)).Controltype == typeOfControl.Button)

{

controlButton = (Button)recentMenu.ControlsAndSubmenues.ElementAt(i);

menuObjectList.Add(controlButton);

}

if (((Controls)recentMenu.ControlsAndSubmenues.ElementAt(i)).Controltype == typeOfControl.Switch)

{

controlSwitch = (Switch)recentMenu.ControlsAndSubmenues.ElementAt(i);

menuObjectList.Add(controlSwitch);

menuObjectList.Add(controlSwitch.SwitchButton1);

menuObjectList.Add(controlSwitch.SwitchButton2);

}

if (((Controls)recentMenu.ControlsAndSubmenues.ElementAt(i)).Controltype == typeOfControl.Slider)

{

controlSlider = (Slider)recentMenu.ControlsAndSubmenues.ElementAt(i);

menuObjectList.Add(controlSlider);

}

}

else

{

if(!menuObjectList.Contains(RecentMenu))

menuObjectList.Add((MenuObject)recentMenu);

if (!(menuObjectList.Contains((MenuObject)recentMenu.ControlsAndSubmenues.ElementAt(i))))

{

menuObjectList.Add((MenuObject)recentMenu.ControlsAndSubmenues.ElementAt(i));

addDrawableMenuObjectsToList((MenuObject)recentMenu.ControlsAndSubmenues.ElementAt(i), menuObjectList);

}

}

}

}

#endregion

}

}