using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using Microsoft.Xna.Framework;

using Microsoft.Xna.Framework.Graphics;

using ManhattanMorning.Misc;

namespace ManhattanMorning.Model.MenuObject.MenuComponents

{

/// <summary>

/// The painted Background, while being in a Menu-GameState.

/// </summary>

public class MenuBackground : DrawableObject

{

#region Properties

#endregion

#region Members

#endregion

#region Initialization

/// <summary>

/// Intialization of a Game Objects.

/// </summary>

/// <param name="name">Name of the Object. use variable name, if not sure what to choose.</param>

/// <param name="visible">Shall the object be visible? only set visible if object either contains a texture or a animation</param>

/// <param name="texture">The texture. if this object has no texture, set to null</param>

/// <param name="animation">animation, if this object has no animation, set to null</param>

/// <param name="Size">The size. depending on the measurementUnit either in Meters or in percent of screen</param>

/// <param name="Position">The MIDDLE position. depending on the measurementUnit either in Meters or in percent of screen</param>

/// <param name="layer">In which layer should the object be drawn?</param>

/// <param name="measurementUnit">The measurement unit. </param>

public MenuBackground(String name, bool visible, Texture2D texture,Animation animation, Vector2 size, Vector2 position, int layer,MeasurementUnit measurementUnit)

: base(name, visible, texture,animation, size, position, layer,measurementUnit)

{

}

#endregion

#region Methods

//place for Methods; remove, when adding some.

#endregion

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using Microsoft.Xna.Framework;

using Microsoft.Xna.Framework.Graphics;

using ManhattanMorning.Misc;

namespace ManhattanMorning.Model.MenuObject.MenuComponents

{

/// <summary>

/// The MenuHeadline is the shown Headline, when the menu is parentMenu( ? :) ).

/// </summary>

public class MenuHeadline : DrawableObject

{

#region Properties

#endregion

#region Members

#endregion

#region Initalization

/// <summary>

/// Constructs the headline of a menu

/// </summary>

/// <param name="visible">Is the headline visible?</param>

/// <param name="texture">The picture drawn as headline.</param>

/// <param name="size">The size of the headline picture</param>

/// <param name="position">Upper left point of the headline</param>

/// <param name="layer">Layer of the headline. In the case of the background it shall be 0.</param>

public MenuHeadline(String name,bool visible, Texture2D texture,Animation animation, Vector2 size, Vector2 position, int layer,MeasurementUnit measurementUnit)

: base(name, visible, texture, animation, size, position, layer, measurementUnit)

{

}

#endregion

#region Methods

#endregion

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using Microsoft.Xna.Framework;

using Microsoft.Xna.Framework.Graphics;

using ManhattanMorning.Misc;

namespace ManhattanMorning.Model.MenuObject.SpecificControls

{

/// <summary>

/// A Button represent a SpecificControl.

/// </summary>

public class Button : Controls

{

/// <summary>

/// Constructs a Button. A Button is a ControlObject in the Menu.

/// </summary>

/// <param name="name">Name of the Object. use variable name, if not sure what to choose.</param>

/// <param name="visible">Shall the object be visible? only set visible if object either contains a texture or a animation</param>

/// <param name="texture">The texture. if this object has no texture, set to null</param>

/// <param name="animation">animation, if this object has no animation, set to null</param>

/// <param name="size">The size. depending on the measurementUnit either in Meters or in percent of screen</param>

/// <param name="position">The UpperLeft position. depending on the measurementUnit either in Meters or in Pixel </param>

/// <param name="layer">In which layer should the object be drawn?</param>

/// <param name="measurementUnit">The measurement unit. </param>

/// <param name="highlighted">Indicates, whether the object is highlighted. </param>

public Button(String name, bool visible, Texture2D texture, Animation animation, Vector2 size, Vector2 position, int layer, MeasurementUnit measurementUnit, bool highlighted, typeOfControl controltype)

: base(name, visible, texture, animation, size, position, layer, measurementUnit, highlighted, controltype)

{

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using Microsoft.Xna.Framework;

using Microsoft.Xna.Framework.Graphics;

using ManhattanMorning.Misc;

namespace ManhattanMorning.Model.MenuObject.SpecificControls

{

public class PicOption : DrawableObject

{

#region Properties

public bool Highlighted { get { return this.highlighted; } set { this.highlighted = value; } }

#endregion

#region Members

public bool highlighted;

#endregion

#region Initialization

public PicOption(String name, bool visible, Texture2D texture, Animation animation, Vector2 size, Vector2 position, int layer, MeasurementUnit measurementUnit, bool highlighted)

: base(name, visible, texture, animation, size, position, layer, measurementUnit)

{

this.highlighted = highlighted;

}

#endregion

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using Microsoft.Xna.Framework;

using Microsoft.Xna.Framework.Graphics;

using ManhattanMorning.Misc;

namespace ManhattanMorning.Model.MenuObject.SpecificControls

{

/// <summary>

/// PictureOptions shall be a specificControl

/// </summary>

class PictureOption : Controls

{

#region Properties

/// <summary>

/// A List of the PicturesOptions to choose from.

/// </summary>

public LayerList<PicOption> PictureOptions { get { return this.pictureOption; } set { this.pictureOption = value; } }

/// <summary>

/// Which PictureOption is selected?

/// </summary>

public int Selected { get { return this.selected; } set { this.selected = value;} }

#endregion

#region Members

/// <summary>

/// A List of the PicturesOptions to choose from.

/// </summary>

private LayerList<PicOption> pictureOption;

/// <summary>

/// Which PictureOption is selected?

/// </summary>

private int selected;

#endregion

#region Initialistion

/// <summary>

///

/// </summary>

/// <summary>

/// Constructs a PictureOption. A PictureOption is a ControlObject in the Menu.

/// </summary>

/// <param name="name">Name of the Object. use variable name, if not sure what to choose.</param>

/// <param name="visible">Shall the object be visible? only set visible if object either contains a texture or a animation</param>

/// <param name="texture">The texture. if this object has no texture, set to null</param>

/// <param name="animation">animation, if this object has no animation, set to null</param>

/// <param name="size">The size. depending on the measurementUnit either in Meters or in percent of screen</param>

/// <param name="position">The UpperLeft position. depending on the measurementUnit either in Meters or in Pixel </param>

/// <param name="layer">In which layer should the object be drawn?</param>

/// <param name="measurementUnit">The measurement unit. </param>

/// <param name="highlighted">Is the ObjectHighlighted</param>

/// <param name="controltype">Set it to Picture!</param>

/// <param name="pictureOptions">List of the options</param>

/// <param name="selected">which option is selected</param>

public PictureOption(String name, bool visible, Texture2D texture, Animation animation, Vector2 size, Vector2 position, int layer, MeasurementUnit measurementUnit, bool highlighted, typeOfControl controltype, LayerList<PicOption> pictureOption, int selected)

: base(name, visible, texture, animation, size, position, layer, measurementUnit, highlighted, typeOfControl.Pictures)

{

this.pictureOption = pictureOption;

this.selected = selected;

}

#endregion

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using Microsoft.Xna.Framework;

using Microsoft.Xna.Framework.Graphics;

using ManhattanMorning.Misc;

namespace ManhattanMorning.Model.MenuObject.SpecificControls

{

/// <summary>

/// Slider. Has to be implemented!

/// </summary>

public class Slider : Controls

#region Properties

#endregion

#region Member

#endregion

#region Initialization

{

public Slider(String name, bool visible, Texture2D texture, Animation animation, Vector2 size, Vector2 position, int layer, MeasurementUnit measurementUnit, bool highlighted, typeOfControl controltype)

: base(name, visible, texture, animation, size, position, layer, measurementUnit, highlighted, controltype)

{

}

#endregion

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using Microsoft.Xna.Framework;

using Microsoft.Xna.Framework.Graphics;

using ManhattanMorning.Misc;

namespace ManhattanMorning.Model.MenuObject.SpecificControls

{

/// <summary>

/// A specific Control, where you can choose between to opportunities

/// </summary>

public class Switch : Controls

{

#region Properties

/// <summary>

/// SwitchButton1 represents a Button the first of the two opportunities of Switch.

/// </summary>

public SwitchButtons SwitchButton1 { get {return this.switchButton1;} set{ this.switchButton1 = value;} }

/// <summary>

/// SwitchButton2 represents a Button the second of the two opportunities of Switch.

/// </summary>

public SwitchButtons SwitchButton2 { get { return this.switchButton2; } set { this.switchButton2 = value; } }

/// <summary>

/// This integer determinates, which of the 2 Buttons is selected.

/// </summary>

public int Selected { get { return this.selected; } set { this.selected = value; } }

#endregion

#region Member

/// <summary>

/// SwitchButton1 represents a Button the first of the two opportunities of Switch.

/// </summary>

private SwitchButtons switchButton1;

/// <summary>

/// SwitchButton2 represents a Button the second of the two opportunities of Switch.

/// </summary>

private SwitchButtons switchButton2;

/// <summary>

/// This integer determinates, which of the 2 Buttons is selected.

/// </summary>

private int selected;

#endregion

#region Initialization

/// <summary>

/// Constructs a Switch. A Switch is a Object Menu contains two opportunities to choose from

/// </summary>

/// <param name="name">Name of the Object. use variable name, if not sure what to choose.</param>

/// <param name="visible">Shall the object be visible? only set visible if object either contains a texture or a animation</param>

/// <param name="texture">The texture. if this object has no texture, set to null</param>

/// <param name="animation">animation, if this object has no animation, set to null</param>

/// <param name="Size">The size. depending on the measurementUnit either in Meters or in percent of screen</param>

/// <param name="Position">The UpperLeft position. depending on the measurementUnit either in Meters or in Pixel </param>

/// <param name="layer">In which layer should the object be drawn?</param>

/// <param name="measurementUnit">The measurement unit. </param>

/// <param name="switchButtons1">The button representing the first option of the switch, set a SwitchButton, null will be not accepted</param>

/// <param name="switchButtons2">The button representing the second option of the switch, set a SwitchButton, null will be not accepted</param>

/// <param name="selected"></param>

public Switch(String name, bool visible, Texture2D texture, Animation animation, Vector2 size, Vector2 position, int layer, MeasurementUnit measurementUnit, bool highlighted, typeOfControl controltype, SwitchButtons switchButtons1, SwitchButtons switchButtons2, int selected)

: base(name, visible, texture, animation, size, position, layer, measurementUnit, highlighted, controltype)

{

if (visible && (switchButtons1 == null || switchButtons2 == null))

{

throw new ArgumentException("A Switch need two Button as opportunities!");

}

if( selected <= 0)

{

this.selected = 0;

}

else

{

this.selected = 1;

}

this.switchButton1 = switchButtons1;

this.switchButton2 = switchButtons2;

}

#endregion

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using Microsoft.Xna.Framework;

using Microsoft.Xna.Framework.Graphics;

using ManhattanMorning.Misc;

namespace ManhattanMorning.Model.MenuObject.SpecificControls

{

/// <summary>

/// The options of a SwitchButton

/// </summary>

public class SwitchButtons : DrawableObject

{

#region Properties

public bool Highlighted { get { return this.highlighted; } set { this.highlighted = value; } }

#endregion

#region Members

public bool highlighted;

#endregion

/// <summary>

/// The SwitchButtons are the two opportunities at a Switch

/// </summary>

/// <param name="name">NameOftheSwitchButton</param>

/// <param name="visible">Is the SwitchButton visible?</param>

/// <param name="texture">A Texture for the switchButton</param>

/// <param name="animation">A Animation for the switchButton</param>

/// <param name="size">The Size of the Switch Button</param>

/// <param name="position">The UpperLeft Position of the SwitchButton</param>

/// <param name="layer">In which layer is the switchButton?</param>

/// <param name="measurementUnit">The measurement.unit.</param>

public SwitchButtons(String name, bool visible, Texture2D texture, Animation animation, Vector2 size, Vector2 position, int layer, MeasurementUnit measurementUnit, bool highlighted)

: base(name, visible, texture, animation, size, position, layer, measurementUnit)

{

this.highlighted = highlighted;

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using Microsoft.Xna.Framework;

using Microsoft.Xna.Framework.Graphics;

using ManhattanMorning.Misc;

namespace ManhattanMorning.Model.MenuObject

{

public enum typeOfControl

{

Button,

Pictures,

Switch,

Slider

};

public class Controls : DrawableObject

{

#region Properties

public typeOfControl Controltype { get { return this.controltype; } set { this.controltype = value; } }

public bool Highlighted { get { return this.highlighted; } set { this.highlighted = value; } }

#endregion

#region Member

private typeOfControl controltype;

private bool highlighted;

#endregion

#region Initialization

public Controls(String name, bool visible, Texture2D texture, Animation animation, Vector2 size, Vector2 position, int layer, MeasurementUnit measurementUnit, bool highlighted, typeOfControl controltype)

: base(name, visible, texture, animation, size, position, layer, measurementUnit)

{

this.highlighted = highlighted;

this.controltype = controltype;

}

#endregion

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using Microsoft.Xna.Framework;

using Microsoft.Xna.Framework.Graphics;

using ManhattanMorning.Model.MenuObject.MenuComponents;

using ManhattanMorning.Misc;

namespace ManhattanMorning.Model.MenuObject

{

/// <summary>

/// Defines a MenuObject as Ingame- or

/// MainMenu-Object

/// </summary>

public enum menuType

{

Main,

Ingame

};

/// <summary>

/// Parent class for all menu objects which are drawn onscreen

/// when a menu is visible

/// </summary>

public class MenuObject : DrawableObject

{

#region Properties

#region ProbsOfMenuAsFatherMenu

/// <summary>

/// Headline of this Headline

/// </summary>

public MenuHeadline Headline

{

get { return this.headline; }

set { this.headline = value; }

}

/// <summary>

/// Background of this MenuObject

/// </summary>

public MenuBackground Background

{

get { return this.background; }

set { this.background = value; }

}

/// <summary>

/// All submenues of this MenuObject

/// </summary>

public LayerList<MenuObject> Submenues

{

get { return this.submenues; }

set { this.submenues = value; }

}

/// <summary>

/// Number of the selected MenuItem

/// </summary>

public int SelectedItem

{

get { return this.selectedItem; }

set { this.selectedItem = value; }

}

/// <summary>

/// List Of Control, that should be displayed in a specificMenu

/// </summary>

public LayerList<Controls> Controls

{

get { return this.controls; }

set { this.controls = value; }

}

public LayerList<DrawableObject> ControlsAndSubmenues

{

get { return this.controlsAndsubmenues; }

set { this.controlsAndsubmenues = value; }

}

#endregion

#region ProbsOfMenuAsMenuItem

/// <summary>

/// Indicates, if the menu object is currently selected

/// </summary>

public bool Highlighted

{

get { return this.highlighted; }

set { this.highlighted = value; }

}

#endregion

#endregion

#region Members

#region MenuAsParentMenu

/// <summary>

/// properties of the menu, when the menu is the "parent-menu"

/// </summary>

// the headline of the menu

private MenuHeadline headline;

//the background of the menu

private MenuBackground background;

// the Menu-Items

private LayerList<MenuObject> submenues;

// what kind of menu do we have? Main or Ingame

private menuType type;

// number of the currently selected item in the

private int selectedItem;

// MenuElemet of this MenuItem (slider, switch, button)

private LayerList<Controls> controls;

// All Menu Item of the Menu (controls + submenus)

private LayerList<DrawableObject> controlsAndsubmenues;

#endregion

#region MenuAsMenuItem

/// <summary>

/// properties of the menu as a menu item

/// </summary>

// is the MenuItem selected

private bool highlighted;

#endregion

#endregion

#region Initialization

/// <summary>

/// Constructs a menu object, using all attributes from DrawableObject plus the attributes

/// for 2 cases:

/// 1. Menu as ParentMenu:

/// Title, Headline, Background, Submenues, SelctedItem,Type

/// 2. Menu as MenuItem:

/// Highlighted, Control, SelectedAnimation

/// </summary>

/// <param name="name">Name Of the DrawableObject</param>

/// <param name="visible">Indictates if MenuObject is visible</param>

/// <param name="texture">Texture of the MenuObject as MenuItem</param>

/// <param name="animation"> Animation of the MenuObject as MenuItem</param>

/// <param name="size">Sets size of the Texture</param>

/// <param name="position">Sets the UpperLeft position in the given measurement unit</param>

/// <param name="layer"> Name of the menu object./ ParentMenu</param>

/// <param name="Headline">The Headline/ ParentMenu</param>

/// <param name="Background">The Background/ ParentMenu</param>

/// <param name="Submenues">List of Submenues/ ParentMenu</param>

/// <param name="selctedItem">The Number of the selected Item/ ParentMenu</param>

/// <param name="Type">Type of the Menu. InGame Or MainMenu / ParentMenu</param>

/// <param name="Highlighted">Indicates the menu object that is currently selected.</param>

/// <param name="controls">ControlItems of the/ MenuItem</param>

/// <param name="measurementUnit">measurement unit</param>

public MenuObject(String name, bool visible, Texture2D texture, Animation animation, Vector2 size, Vector2 position, int layer, MenuHeadline headline,

MenuBackground background, LayerList<MenuObject> submenues, int numberOfselctedItem, menuType Type, bool highlighted, LayerList<Controls> controls

, MeasurementUnit measurementUnit, LayerList<DrawableObject> controlsAndSubmenues)

: base(name, visible, texture, animation, size, position, layer, measurementUnit)

{

this.headline = headline;

this.background = background;

this.submenues = submenues;

this.controls = controls;

this.selectedItem = numberOfselctedItem;

this.highlighted = highlighted;

this.type = Type;

this.controlsAndsubmenues = controlsAndSubmenues;

#endregion

#region Methods

#endregion

}

}

}