blindGUI Documentation

Installation

Just copy blindGUI library to assets/Plugins folder.

Prepare Graphics

To have perfect GUI with pixel-perfect elements you should set Texture Type to GUI and set Truecolor as Format. It will set real texture size so auto-size of controls will work.

Texure type for tiles of blindGUITiledTexturedContainer should be set to Advanced with check of Read/Write Enabled property.

Quick Start

Step 1. Dialog window

Add quick start materials and blindGUI library to project.
Set Texture type to GUI for all textures except QSBackground.
Set Texture type to Advanced and check Read/Write Enabled for QSBackground.
Start with creation of GameObject at root of your project with name GUIcontroller.
Add blindGUIController component to it.
Next create first GUI layer. Add new GameObject as sub-element of GUIcontroller. Name it HelloLayer.
Add blindGUILayer component to it.
Continue with new GameObject creation. It should be a sub-element of HelloLayer with name Window.
Add blindGUITexturedContainer to it and set:

- Vertical and horizontal alignment to center.
- Set Background Texture property to QSWindow
- Set Auto Size to Texture to true.

Step 2. Window label

Add GameObject to this window with name HelloLabel.

Add blindGUIText component to it.
Set HelloLabel parameters:

- Set Font Color to (87,87,87,255)
- Set Font to FuturaBK
- Set Font Size for FuturaBK font to 30
- Type "Hello world." as Text
- Set Text Anchor to Middle Center
- Set Vertical Align to Top
- Set Horizontal Align to Stretch
- Set Size to (0,32)
- Set Offset to (0,50)

Step 3. OK button

Add GameObject to this window with name OKButton.
Add blindGUIButton component to it.
Set OKButton parameters:

- Set Idle Image to QSButtinIdle image
- Set Hover Image to QSButtonHover image
- Set Press Image to QSButtonPress image
- Set Horizontal Align to Center
- Set Vertical Align to Bottom
- Set Offset to (0,50)
- Set Auto Size to Texture to True

Step 4. Background

Now add QuickStartMainScript...(where ... is CS, JS or Boo - your favorite language) to HelloLayer.
Add new layer called BackgroundLayer with Z value of -1
Add sub-element with blindGUITiledTextureContainer called "Background"

- Set Vertical and horizontal alignment to Stretch
- Set QSBackground as Background Texture for it

Performance Questions

BlindGUI is based on Unity3D GUI engine with all pros and cons.

Desktop systems can use up to 300 objects without performance drop.

Mobile systems can use 5-7 objects without performance drop during dynamic graphics rendering. Pure GUI screens, which allows fps drop to 10-15, can hold 15-50 objects depending on device.

Custom element creation

Creation of custom elements is easy.

Create class and set blindGUIParentElement(or other blindGUI element if you need some existent functions) as parent class of control.

If new component needs initialization, add Start method:

```
01.    override public void Start() {
02.        // Your initialization code
03.        base.Start();
04.    }
```

Add Draw method:

```
01.    override public void Draw(
02.        blingGUILayout parentLayout,
03.            bool enabled
04.        ) {
05.            Rect contentFrame = GetFrame( parentLayout );
06.            // Your draw code
07.            base.Draw( parentLayout, enabled );
08.    }
```

It's mandatory to add call of GetFrame method, because it setups rotation, translation and scale of GUI. Also parent Draw method call is important, because it will call draw for all children elements.

blindGUIParentElement

Methods

public Rect **GetFrame (**
blingGUILayout **parentLayout**
)
Returns rectangle offset and size for current element.

parentLayout *Information about parent's element size, anchorPoint, scale, etc.*

public void **Draw (**
blingGUILayout **parentLayout** ,
bool **enabled**
)
Draws element and all it's children.

parentLayout *information about parent's element size, anchorPoint, scale, etc.*
enabled *flag show if parent element is enabled.*

public void **UpdateLayout (**
)
Find all children elements and add them to elements draw list. Must be called after dynamic children elements creation.

public void **AnimateTo (**
blindGUIAnimationState **targetState** ,
float **animationTime** ,
AnimationCompleteDelegate **animationCompleteDelegate** ,
iTweenInBlindGUI.EaseType **easeType** ,
float **delay**
)
Starts Animation from objects current state to new

targetState *Target animation state*

animationTime *Duration of animation in seconds*
animationCompleteDelegate *This function will be called after animation is finished*
*void **AnimationComplete(***
*blindGUIParentElement **sender***
)
***sender** - element with ended animation.*
easeType *Animation ease type*
delay *Delay of animation in seconds*

public void **AnimateTo (**
blindGUIAnimationState **targetState** ,
float **animationTime** ,
AnimationCompleteDelegate **animationCompleteDelegate**

)

Starts Animation from objects current state to new

targetState	*Target animation state*
animationTime	*Duration of animation in seconds*
animationCompleteDelegate	*This function will be called after animation is finished* *void **AnimationComplete(*** *blindGUIParentElement **sender*** ***)*** ***sender** - element with ended animation.*

public void **AnimateTo (**

blindGUIAnimationState	**targetState** ,
float	**animationTime** ,
AnimationCompleteDelegate	**animationCompleteDelegate** ,
iTweenInBlindGUI.EaseType	**easeType**

)

Starts Animation from objects current state to new

targetState	*Target animation state*
animationTime	*Duration of animation in seconds*
animationCompleteDelegate	*This function will be called after animation is finished* *void **AnimationComplete(*** *blindGUIParentElement **sender*** ***)*** ***sender** - element with ended animation.*
easeType	*Animation ease type*

public void **AnimateTo (**

blindGUIAnimationState	**targetState** ,
float	**animationTime** ,
iTweenInBlindGUI.EaseType	**easeType**

)

Starts Animation from objects current state to new

targetState	*Target animation state*
animationTime	*Duration of animation in seconds*
easeType	*Animation ease type*

public void **AnimateTo (**

blindGUIAnimationState	**targetState** ,
float	**animationTime**

)

Starts Animation from objects current state to new

targetState	*Target animation state*
animationTime	*Duration of animation in seconds*

Properties

public HALIGN **m_horizontalAlign**
Horizontal alignment of element

left	element's left edge and container's left edge has distance equal to m_offset.x
center	centers element horizontally. m_offset.x moves element right for positive and left for negative values
right	element's right edge and container's right edge has distance equal to m_offset.x
stretch	element width is equal to containers width
free	element's anchor point has horizontal offset from container's anchor point equal to m_offset.x

public Vector2 **m_offset**
Meaning of this parameter depends on align values. See #id1 and m_verticalAlign.

public VALIGN **m_verticalAlign**
Vertical alignment of element

top	element's top edge and container's top edge has distance equal to m_offset.y
center	centers element vertically. m_offset.y moves element bottom for positive and top for negative values
bottom	element's bottom edge and container's bottom edge has distance equal to m_offset.y
stretch	element height is equal to containers height
free	element's anchor point has vertical offset from container's anchor point equal to m_offset.y

public Vector2 **m_size**
Size of element in elements' scale. Has no effect when stretch align selected.

public int **m_z**
z position of control in this container.

public float **m_scale**
Scale of element. Affects children

public float **m_angle**
Angle in degrees of control rotation. Effects only if both m_hAlign and m_vAlign is set to free

public Vector2 **m_anchorPoint**
Anchor point of element. This is center of rotation. This is relative value with (0,0) at element's center and (1,1) is width and height of control. left top corner is (-0.5, -0.5); Right bottom corner is (0.5, 0.5).

public float **m_alpha**
alpha value of element and children

public bool **m_enabled**
enables element and it's children.

public bool **m_clipping**
enables clipping of element's children. Effects only if non of m_hAlign and m_vAlign is set to free

blindGUIController

Methods

public void **UpdateLayout (**
)
Find all children elements and add them to elements draw list. Must be called after dynamic children elements creation.

Properties

public ScaleType **m_scaleType**
If screen resolution differs from target resolution, blindGUI components will be scaled depending on this parameter.

NoScale	Layers scale is always 1
FitAll	Whole target screen fits to current. Without crop.
FillAll	Fills screen with target screen. Overflow is cropped.
FitWidth	Target screen width fits on screen. Height may crop.
FitHeight	Target screen height fits on screen. Width may crop.

public Vector2 **m_targetScreenSize**
Target screen resolution, used to scale whole GUI system.

public ScaleDirection **m_scaleDirection**
Using this flag you can choose direction of scaling

Both	It will shrink or grow layers if needed.
Shrink	It will shrink layers if needed.
Grow	It will grow layers if needed.

blindGUILayer inherited from blindGUIParentElement

Properties

public int **id**
Identification of layer. This is assigned by blindGUIController.

blindGUIButton inherited from blindGUIParentElement

Properties

public bool **m_pushed**
Indicates state of toggle and radio button. Can be used to set button state.

public Texture2D **m_idleImage**
Button image in idle state

public Texture2D **m_pressedImage**
Button image in pressed state

public Texture2D **m_hoverImage**
Button image in hover state

public Texture2D **m_disabledImage**
Button image in disabled state

public bool **m_autosizeToTexture**
Automatically adjust size of button to m_idleImage texture.

public string **m_groupTag**
Radio buttons group tag. Only one radio button in group can be in on state.

public ButtonClickDelegate **m_buttonClickDelegate**
Button click event handler function delegate
void ***ButtonClicked(***
blindGUIButton ***sender***
)
sender *- clicked button*

public ButtonStateChangeDelegate **m_buttonStateChangedDelegate**
Button state change event handler function delegate
void ***ButtonClicked(***
blindGUIButton ***sender***
bool ***state***
)

sender *- clicked button*
state *- new state of button*

blindGUIText inherited from blindGUIParentElement

Properties

public string **m_text**
Text

public string **m_font**
Text field font

public Color **m_fontColor**
Text field font color

public int **m_maxLength**
Max text length. -1 for unlimited text length.

public TextAnchor **m_textAnchor**
Text anchor point. Sides, corners and center of element.

blindGUIPasswordField inherited from blindGUIParentElement

Properties

public string **m_text**
Text

public string **m_font**
Text field font

public Color **m_fontColor**
Text field font color

public int **m_maxLength**
Max text length. -1 for unlimited text length.

public char **m_passwordChar**
Character replaces input symbols

blindGUITextArea inherited from blindGUIParentElement

Properties

public string **m_text**
Text

public string **m_font**
Text field font

public Color **m_fontColor**
Text field font color

public int **m_maxLength**
Max text length. -1 for unlimited text length.

blindGUITextField inherited from blindGUIParentElement

Properties

public string **m_text**
Text

public string **m_font**
Text field font

public Color **m_fontColor**
Text field font color

public int **m_maxLength**
Max text length. -1 for unlimited text length.

blindGUITexturedContainer inherited from blindGUIParentElement

Properties

public Texture **m_backgroundTexture**
Background texture for container

public bool **m_autoSizeToTexture**
Container size adjusts automatically to size of texture.

public ScaleMode **m_textureScaleMode**
Scale mode of texture in container.

StretchToFill	Stretches the texture to fill the complete component rectangle
ScaleAndCrop	Scales the texture, maintaining aspect ratio, so it completely covers component rectangle. If components aspect ratio differs from texture, texture is cropped
ScaleToFit	Scales the texture, maintaining aspect ratio, so it completely fits within component rectangle

blindGUIMovieTexturedContainer inherited from blindGUITexturedContainer

Methods

public void **Play (**
)
Starts movie texture play and resumes from pause.

public void **Pause (**
)
Pauses movie texture.

public void **Stop (**
)
Stops movie texture. Rewinds at play.

blindGUITiledTexturedContainer inherited from blindGUITexturedContainer

Methods

public void **ReassignTexture (**
)
Creates tiled texture from m_backgroundTexture. It must be called at every m_backgroundTexture set.

blindGUIColorTexturedContainer inherited from blindGUITiledTexturedContainer

Properties

public Color **m_textureColor**
Background color of container.

blindGUIAnimationState

Properties

public Vector2 **size**
Size of element

public Vector2 **anchorPoint**
Anchor point of element. This is center of rotation. This is relative value with (0,0) at element's center and (1,1) is width and height of control. left top corner is (-0.5, -0.5); Right bottom corner is (0.5, 0.5).

public Vector2 **offset**
Meaning of this parameter depends on align values. See #id1 and m_verticalAlign.

public float **angle**
Angle in degrees of control rotation. Effects only if both m_hAlign and m_vAlign is set to free

public float **alpha**
alpha value of element and children

public float **scale**
Scale of element. Affects children