Gaming buttons documentation

Gaming buttons library gives you access to 2 classes to create and control buttons. The first class is ***GameButtons.*** Is a custom view and you will use it in your xml files as well as in your fragment or activity. The second class is the ***GameButtonsController*** which will help you control multiple buttons with simple commands.

GameButtons class

Create a button using xml code:

```
<com.app.gamebuttons.main.GameButtons
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:style="gb_Bubbly"
        app:soundEffect="gb_Booing"
        app:onTouchAnimation="gb_Pushdown" />
```

Using the parameter app: gives you access to 40 attributes to customize the look and functionality of the button. Let's have a look at them.

Button style

1. style
2. shape
3. buttonColor
4. pressedButtonColor
5. opacity
6. customButton
7. customPressedButton
8. buttonMargin
9. behindButtonImage
10. customBehindButtonImage

Icon style

11. buttonIcon
12. pressedButtonIcon
13. customButtonIcon
14. customPressedButtonIcon
15. buttonIconColor
16. pressedButtonIconColor
17. iconPosition
18. iconSize
19. iconLeftMargin
20. iconTopMargin
21. iconRightMargin
22. iconBottomMargin

Animations

23. onTouchAnimation
24. enterAnimation
25. enterAnimationDuration
26. enterAnimationDelay
27. exitAnimation
28. exitAnimationDuration
29. exitAnimationDelay
30. animateBehindButtonImage
31. behindImageAnimInterpolator
32. behindImageAnimDuration

Sound

33. soundEffect
34. soundVolume
35. customSound

Controls

36. touchFeedback
37. showPressedButtonFirst
38. disableSound
39. lockButton
40. touchAnimationEnabled

GameButtons XML attributes

1. app:style

Description: Choose among 5 different styles of buttons
Values: `gb_Cutie, gb_Bubbly, gb_Shiny, gb_Curvy, gb_Woody`

2. app:shape

Description: Choose among 4 different shapes
Values: `gb_Circle, gb_Square, gb_Rectangle, gb_Oval`

3. app:buttonColor

Description: Choose among 12 different colors
Values: `gb_Blue, gb_Blue_light, gb_Gray, gb_Gray_light, gb_Green, gb_Green_light, gb_Orange, gb_Orange_light, gb_Purple, gb_Purple_light, gb_Red, gb_Red_light`

4. app:pressedButtonColor

Description: Set the color of the button when it's pressed
Values: `gb_Blue, gb_Blue_light, gb_Gray, gb_Gray_light, gb_Green, gb_Green_light, gb_Orange, gb_Orange_light, gb_Purple, gb_Purple_light, gb_Red, gb_Red_light`

5. app:opacity

Description: Set the opacity of the button
Values: `float`

6. app:customButton

Description: Set an image from drawable folder for the button
Values: `Drawable resource`

7. app:customPressedButton

Description: Set an image from drawable folder for the pressed button
Values: `Drawable resource`

8. app:buttonMargin

Description: Set margin for button's image
Values: `dp`

9. app:behindButtonImage

Description: Add an image behind button
Values: `gb_Stars, gb_Light, gb_Light_1, gb_Light_2`

10. app:customBehindButtonImage

Description: Set an image from drawable folder for the image behind the button
Values: `Drawable resource`

11. app:buttonIcon

Description: Choose among 212 button icons
Values: `gb_Sound, gb_Back, gb_Game, gb_Help, gb_Music ...`

12. app:pressedButtonIcon

Description: Choose among 212 icons when pressed button is enabled
Values: `gb_Sound, gb_Back, gb_Game, gb_Help, gb_Music ...`

13. app:customButtonIcon

Description: Set an image from drawable folder for icon
Values: `Drawable resource`

14. app:customPressedButtonIcon

Description: Set an image from drawable folder for icon when pressed button is enabled
Values: `Drawable resource`

15. app:buttonIconColor

Description: Set the color of the icon
Values: `color`

16. app:pressedButtonIconColor

Description: Set the color of the pressed button icon
Values: `color`

17. app:iconPosition

Description: Set the position of the icon
Values: `gb_Center, gb_Left, gb_Right`

18. app:iconSize

Description: Set the size of the icon in dp
Values: `dp`

19. app:iconLeftMargin

Description: Set the left margin of the icon in dp
Values: `dp`

20. app:iconTopMargin

Description: Set the top margin of the icon in dp
Values: `dp`

21. app:iconRightMargin

Description: Set the right margin of the icon in dp
Values: `dp`

22. app:iconBottomMargin

Description: Set the bottom margin of the icon in dp
Values: `dp`

23. app:onTouchAnimation

Description: Set the animation type when the button is touched
Values: `gb_Shake, gb_Bounce, gb_Pushdown, gb_Explode, gb_Squeeze`

24. app:enterAnimation

Description: Set an animation which will play when the button is initially displayed
Values: `gb_FadeIn, gb_SlideInLeft, gb_SlideInRight, gb_SlideInUp , gb_SlideInDown, gb_ZoomIn, gb_RollIn, gb_RotateIn, gb_FlipInX, gb_FlipInY`

25. app:enterAnimationDuration

Description: How long the animation will last in milliseconds
Values: `long`

26. app:enterAnimationDelay

Description: Play the animation after a delay (milliseconds)
Values: `long`

27. app:exitAnimation

Description: Set the animation of the button to play when the method `playExitAnimation() is called`
Values: `gb_FadeOut, gb_SlideOutLeft, gb_SlideOutRight, gb_SlideOutUp , gb_SlideOutDown, gb_ZoomOut, gb_RollOut, gb_RotateOut, gb_FlipOutX, gb_FlipOutY`

28. app:exitAnimationDuration

Description: How long the animation will last in milliseconds
Values: `long`

29. app:exitAnimationDelay

Description: Play the animation after a delay (milliseconds)
Values: `long`

30. app:animateBehindButtonImage

Description: Enable rotational animation for the image behind button
Values: `boolean`

31. app:behindImageAnimInterpolator

Description: Set an animation interpolator for the rotational animation
Values: `gb_Linear, gb_AccelerateDecelerate,gb_Overshoot, gb_FastOutSlowIn, gb_Bounce, gb_AnticipateOvershoot`

32. app:behindImageAnimDuration

Description: The duration of one rotation in milliseconds
Values: `long`

33. app:soundEffect

Description: Choose among 30 sound effects to play when button is clicked
Values: `gb_Pop, gb_Beep, gb_Duck, gb_GolfBallHit, ...`

34. app:soundVolume

Description: Set sound effect volume
Values: `float`

35. app:customSound

Description: Set your sound from raw folder
Values: `Raw resource`

36. app:touchFeedback

Description: Enable or disable visual feedback when button is touched
Values: `boolean`

37. app:showPressedButtonFirst

Description: Display the pressed button first
Values: `boolean`

38. app:disableSound

Description: Mute sound effects
Values: `boolean`

39. app:lockButton

Description: Disable click functionality of the button
Values: `boolean`

40. app:touchAnimationEnabled

Description: Disable touch animation of the button
Values: `boolean`

GameButtons Public Methods

Method name: `pauseBehindImageAnimation()`
Return type: void
Description: Pause the rotational animation of the image behind button

Method name: `resumeBehindImageAnimation()`
Return type: void
Description: Resume the rotational animation of the image behind button

Method name: `playAnimation(int animation,int animationDuration, int animationDelay)`
Return type: void
Description: Play a button animation. The first parameter *(animation)* specify the type of the animation. The second parameter *(animationDuration)* specify the duration of the animation in milliseconds and the third parameter *(animationDelay)* specify the delay before the animation begin, also in milliseconds.
Example: `playAnimation(Animations.BOUNCE,500,300);`

Method name: `playExitAnimation()`
Return type: void
Description: Play the exit animation which is declared as an xml attribute.

Method name: `playEnterAnimation()`
Return type: void
Description: Play the enter animation which is declared as an xml attribute.

Method name: `setEnableTouchAnimations()`
Return type: void
Description: Enable touch animation of the button which is declared as an xml attribute.

Method name: `setDisableTouchAnimations()`
Return type: void
Description: Disable touch animation of the button which is declared as an xml attribute.

Method name: `setSoundVolume(float volume)`
Return type: void
Description: Set the volume of sound effect.

Method name: `isLocked()`
Return type: Boolean
Description: Return true if button is locked or false if button is unlocked.

Method name: `setLockButton()`
Return type: void
Description: Lock the button. This will disable all events that are triggered when the button is touched. For example sound effect, animation, visual feedback and other.

Method name: `setUnLockButton()`
Return type: void
Description: Unlock the button to restore touch events functionalities.

Method name: `enableSound()`
Return type: void
Description: Enables on touch button sound effect.

Method name: `disableSound()`
Return type: void
Description: Disables on touch button sound effect.

Method name: `unLinkButton()`
Return type: void
Description: Remove link functionality if the button is linked with other buttons.

Method name: **isLinkedModeEnabled()**
Return type: Boolean
Description: Return true if the button is linked with other buttons.

Method name: **setPressedEnabled(boolean enable)**
Return type: void
Description: Enable or disable the pressed button.

Method name: **isButtonPressed()**
Return type: Boolean
Description: Returns true if the pressed button is shown.

Method name: **hasPressedButton()**
Return type: Boolean
Description: Returns true if pressed button is specified as an xml attribute.

Method name: **release()**
Return type: void
Description: Releases the object that handles the animation of the image behind the button. This method must be called in the **onDestroy()** method of your Fragment or Activity if an animation for the *image behind the button* is specified in xml.

GameButtonsController class

This class has 2 main functionalities. The first is to link buttons state in a way that when a button get clicked, all the other buttons get unclicked. The second functionality is to perform an action to multiple buttons using only one command.

Example code:

```
GameButtonsController mainButtonsController;

//Get views
GameButtons easyBtn = view.findViewById(R.id.easyBtn);
GameButtons normalBtn = view.findViewById(R.id.normalBtn);
GameButtons hardBtn = view.findViewById(R.id.hardBtn);
GameButtons playBtn = view.findViewById(R.id.playBtn);


//Set the controller to all buttons
mainButtonsController = new GameButtonsController(easyBtn,normalBtn,hardBtn,playBtn);
mainButtonsController.muteAll();
mainButtonsController.lockButtons();

//Link buttons (static method)
GameButtonsController.Link(easyBtn,normalBtn,hardBtn);
```

To link buttons together you must specify in *xml* the attribute **app:pressedButtonColor** for all buttons you want to link, otherwise you will get a *LinkedButtonException*. Please keep in mind that the *Link* method is *statically allocated* so you won't need a *GameButtonsController* instance to use it.

Tip: You can create multiple *GameButtonsController* instances to control different button groups.

GameButtonsController public methods

Method name: **Link()**
Return type: void
Description: Static method. Link buttons pressed state.

Method name: **release()**
Return type: void
Description: Releases the object that handles the animation of the image behind the button. This method must be called in the **onDestroy()** method of your Fragment or Activity if an animation for the *image behind the button* is specified in **xml.**

Method name: **playAnimation(int animation,int animationDuration, int animationDelay)**
Return type: void
Description: Play buttons animation. The first parameter *(animation)* specify the type of the animation. The second parameter *(animationDuration)* specify the duration of the animation in milliseconds and the third parameter *(animationDelay)* specify the delay before the animation begin, also in milliseconds.
Example: **playAnimation(Animations.BOUNCE,500,500);**

Method name: **playExitAnimation()**
Return type: void
Description: Play the exit animation which is declared as an xml attribute.

Method name: **playEnterAnimation()**
Return type: void
Description: Play the enter animation which is declared as an xml attribute.

Method name: `enableAllTouchAnimations()`
Return type: void
Description: Enables all touch animations which are declared as an xml attribute for all buttons.

Method name: `disableAllTouchAnimations()`
Return type: void
Description: Disable all touch animations which are declared as an xml attribute for all buttons.

Method name: `unLinkAllButtons()`
Return type: void
Description: Remove link functionality.

Method name: `setButtonsVolume(float volume)`
Return type: void
Description: Set the volume of sound effects for all buttons.

Method name: `lockButtons()`
Return type: void
Description: Lock the buttons. This will disable all events that are triggered when the buttons are touched. For example: sound effect, animation, visual feedback and other.

Method name: `unlockButtons()`
Return type: void
Description: Unlock the buttons to restore touch events functionalities.

Method name: `unMuteAll()`
Return type: void
Description: Enables on touch buttons sound effects.

Method name: `muteAll()`
Return type: void
Description: Disables on touch buttons sound effects.