helper_imgui.hps

ImGui_ClearStates

```
void ImGui_ClearStates()
```

Clears all states, timers, variables, etc.

ImGui_InFocus

```
bool ImGui_InFocus()
```

If the current ImGui is in focus.

ImGui_BecameInFocus

```
bool ImGui_BecameInFocus()
```

If the current ImGui just got focus

ImGui_WasInFocus

```
bool ImGui_WasInFocus()
```

If the current ImGui had focus the previous update

ImGui_Exit

```
void ImGui_Exit()
```

If am in-game gui exit, then the current gui is exited. Do not need to be called in a OnGui function.

ImGui_ClickedMouseOutside

```
bool ImGui_ClickedMouseOutside()
```

Returns: true if the mouse is clicked outside the current ImGui

---

ImGui_SetTransCategory

```
void ImGui_SetTransCategory(const tString &in asCat)
```

This sets the current translation category, if “” then all text params will used as text, else as entry names in a Translate(cat, entry)

ImGui_GetTransCategory

```
tString ImGui_GetTransCategory()
```

This gets the current translation category.

ImGui_Translate

```
tWString ImGui_Translate(const tString &in asEntry)
```

Translate using the currently set trans category

ImGui_SetTextOverride

```
void ImGui_SetTextOverride(const tWString &in asText)
```

This sets a literal text to use in place of any empty strings.

ImGui_ClearTextOverride

```
void ImGui_ClearTextOverride()
```

Just clears the currently set text override.

_ImGui_GetTextString

```
tWString _ImGui_GetTextString(const tString &in asText)
```

Internal

ImGui_GetSize

```
cVector2f ImGui_GetSize()
```

Get the size (in pixels) of the GUI screen

ImGui_NrmPos

```
cVector3f ImGui_NrmPos(const cVector3f &in avPos)
```

Input should be normalized, 0-1, and a position in screen space is returned.

ImGui_NrmPos

```
cVector3f ImGui_NrmPos(float afX,
                       float afY,
                       float afZ)
```

Input should be normalized, 0-1, and a position in screen space is returned.

ImGui_NrmPos2

```
cVector2f ImGui_NrmPos2(float afX,
                        float afY)
```

Input should be normalized, 0-1, and a position in screen space is returned.

ImGui_NrmSize

```
cVector2f ImGui_NrmSize(const cVector2f &in avSize)
```

Input should be normalized, 0-1, and a size in screen space is returned.

ImGui_NrmSize

```
cVector2f ImGui_NrmSize(float afX,
                        float afY)
```

Input should be normalized, 0-1, and a size in screen space is returned.

ImGui_NrmPosGroup

```
cVector3f ImGui_NrmPosGroup(const cVector3f &in avPos,
                            bool abOffsetInsideGroup=false)
```

Input should be normalized, 0-1, and a position in the current group space is returned.

ImGui_NrmPosGroup

```
cVector3f ImGui_NrmPosGroup(float afX,
                            float afY,
                            float afZ,
                            bool abOffsetInsideGroup=false)
```

Input should be normalized, 0-1, and a size in the current group space is returned.

ImGui_NrmPos2Group

```
cVector2f ImGui_NrmPos2Group(float afX,
                             float afY,
                             bool abOffsetInsideGroup=false)
```

Input should be normalized, 0-1, and a size in the current group space is returned.

ImGui_NrmSizeGroup

```
cVector2f ImGui_NrmSizeGroup(const cVector2f &in avSize)
```

Input should be normalized, 0-1, and a size in the current group space is returned.

ImGui_NrmSizeGroup

```
cVector2f ImGui_NrmSizeGroup(float afX,
                             float afY)
```

Input should be normalized, 0-1, and a size in the current group space is returned.

ImGui_NrmSizeKeepRatio

```
cVector2f ImGui_NrmSizeKeepRatio(const cVector2f &in avSize)
```

Input should be normalized, 0-1, and a size in screen space is returned.

ImGui_NrmSizeKeepRatio

```
cVector2f ImGui_NrmSizeKeepRatio(float afX,
                                 float afY)
```

Input should be normalized, 0-1, and a size in screen space is returned.

ImGui_NrmSizeGroupKeepRatio

```
cVector2f ImGui_NrmSizeGroupKeepRatio(const cVector2f &in avSize)
```

Input should be normalized, 0-1, and a size in screen space is returned.

ImGui_NrmSizeGroupKeepRatio

```
cVector2f ImGui_NrmSizeGroupKeepRatio(float afX,
                                      float afY)
```

Input should be normalized, 0-1, and a size in screen space is returned.

ImGui_GetRatioCorrectSizeByWidth

```
cVector2f ImGui_GetRatioCorrectSizeByWidth(const cImGuiGfx &in aGfx,
                                           float afWidth)
```

Given an input width and a Gfx object, a size in screen space is returned, being aspect ratio correct according to input Gfx object.

ImGui_GetRatioCorrectSizeByNrmWidth

```
cVector2f ImGui_GetRatioCorrectSizeByNrmWidth(const cImGuiGfx &in aGfx,
                                              float afWidth)
```

Input width should be normalized, 0-1, and a size in screen space is returned, being aspect ratio correct according to input Gfx object.

ImGui_GetRatioCorrectSizeByHeight

```
cVector2f ImGui_GetRatioCorrectSizeByHeight(const cImGuiGfx &in aGfx,
                                            float afHeight)
```

Given an input height and a Gfx object, a size in screen space is returned, being aspect ratio correct according to input Gfx object.

ImGui_GetRatioCorrectSizeByNrmHeight

```
cVector2f ImGui_GetRatioCorrectSizeByNrmHeight(const cImGuiGfx &in aGfx,
                                               float afHeight)
```

Input height should be normalized, 0-1, and a size in screen space is returned, being aspect ratio correct according to input Gfx object.

ImGui_GetRatioCorrectSizeByRect

```
cVector2f ImGui_GetRatioCorrectSizeByRect(const cImGuiGfx &in aGfx,
                                          const cVector2f &in aRect,
                                          bool abCompensateForUV=false)
```

Given an input rect and a Gfx object, the maximum size in screen space to fit within said rect is returned, being aspect ratio correct according to input Gfx object.

ImGui_SetupWidgetRect

```
void ImGui_SetupWidgetRect(const cVector3f &in avInPos,
                           const cVector2f &in avInSize,
                           cVector3f &out avOutPos,
                           cVector2f &out avOutSize,
                           const cVector2f &in avDefaultSize,
                           const cImGuiGfx &in aGfx)
```

ImGui_ActionTriggered

```
bool ImGui_ActionTriggered(eImGuiAction aAction,
                           bool abCheckIfUsed=false)
```

See if an action (sent to ImGui by game) has been triggered (just became down).

- **aAction**: The action type.
- **abCheckIfUsed**: If true, then false is returned if any other class for this action have been made.

ImGui_ActionIsDown

```
bool ImGui_ActionIsDown(eImGuiAction aAction,
                        bool abCheckIfUsed=false)
```

See if an action (sent to ImGui by game) is down.

- **aAction**: The action type.
- **abCheckIfUsed**: If true, then false is returned if any other class for this action have been made.

ImGui_GetMouseRel

```
cVector2f ImGui_GetMouseRel()
```

the relative mouse movement.

ImGui_GetMouseRel3D

```
cVector3f ImGui_GetMouseRel3D()
```

The relative mouse movement as 3D vector (z=0)

ImGui_GetMousePosition

```
cVector2f ImGui_GetMousePosition()
```

The absolute mouse position

ImGui_GetMousePosition3D

```
cVector3f ImGui_GetMousePosition3D()
```

The absolute mouse position as 3D vector (z=0)

ImGui_CheckMouseHasMoved

```
bool ImGui_CheckMouseHasMoved()
```

Returns true if the mouse has been moved.

ImGui_SetFocus

```
void ImGui_SetFocus(const tString &in asWidgetName)
```

Sets the widget that should be in focus.

ImGui_SetAlignment

```
void ImGui_SetAlignment(eImGuiAlign aAlign)
```

Sets the alignment of the widget regarding the postion (if positon is top-left corner, center, etc). Applies to any widget declared after this call.

ImGui_SetUIMoveGroupFlags

```
void ImGui_SetUIMoveGroupFlags(int alGroupFlags)
```

This sets what move group(s) the widget will belong to. When using keyboard/gamepad movement, it is only possible to move between widget that share the same group. This is a bitflag, that it is not the number, but binary bits that matter. To set it, use one or more, eFlagBit_* combined with the bitwise-or-opterator “pipe” TODO: fix pipe char. Example: SetUIMoveGroupFlags(eFlagBit_0 | eFlagBit_2); This will make widgets belong to group 0 and 2. Applies to any widget declared after this call.

ImGui_SetUIMoveWrapMode

```
void ImGui_SetUIMoveWrapMode(eImGuiWrap aWrap)
```

Sets the keyboard/gamepad movement wrap mode, ie what happens when “off the edge”. This is takes effect to any widgets that are defined after it is called. Default is always XY at the start of an OnGui call!

ImGui_GetStateInt

```
int ImGui_GetStateInt(const tString &in asVarName,
                      int alDefault=)
```

ImGui_GetStateFloat

```
float ImGui_GetStateFloat(const tString &in asVarName,
                          float afDefault=0.0f)
```

ImGui_GetStateVector3f

```
cVector3f ImGui_GetStateVector3f(const tString &in asVarName,
                                 const cVector3f &in avDefault)
```

ImGui_GetStateColor

```
cColor ImGui_GetStateColor(const tString &in asVarName,
                           const cColor &in aDefault)
```

ImGui_GetStateBool

```
bool ImGui_GetStateBool(const tString &in asVarName,
                        bool abDefault)
```

ImGui_SetStateInt

```
void ImGui_SetStateInt(const tString &in asVarName,
                       int alVal)
```

ImGui_SetStateFloat

```
void ImGui_SetStateFloat(const tString &in asVarName,
                         float afVal)
```

ImGui_SetStateVector3f

```
void ImGui_SetStateVector3f(const tString &in asVarName,
                            const cVector3f &in avVal)
```

ImGui_SetStateColor

```
void ImGui_SetStateColor(const tString &in asVarName,
                         const cColor &in aVal)
```

ImGui_SetStateBool

```
void ImGui_SetStateBool(const tString &in asVarName,
                        bool abVal)
```

ImGui_IncStateInt

```
void ImGui_IncStateInt(const tString &in asVarName,
                       int alVal)
```

ImGui_IncStateFloat

```
void ImGui_IncStateFloat(const tString &in asVarName,
                         float afVal)
```

ImGui_IncStateVector3f

```
void ImGui_IncStateVector3f(const tString &in asVarName,
                            const cVector3f &in avVal)
```

ImGui_IncStateColor

```
void ImGui_IncStateColor(const tString &in asVarName,
                         const cColor &in aVal)
```

ImGui_FadeStateFloat

```
void ImGui_FadeStateFloat(const tString &in asVarName,
                          float afGoalVal,
                          float afTime,
                          eEasing aType=eEasing_QuadInOut,
                          bool abReplaceIfExist=true)
```

Adds a fade that will fade a float variable from its current value to goal over time.

- **asVarName**: The name of the state var.
- **afGoalVal**: The goal value to fade to.

- **afTime**: How long the fade will take.
- **aType**: the type of fade, meaning in what way the transiation will be. Sigmoid is default as this means the starts and ends smoothly.
- **abReplaceIfExist**: if the variable is currently fading, that fade is replaced if true, else the function call will have no effect.

ImGui_FadeStateVector3f

```
void ImGui_FadeStateVector3f(const tString &in asVarName,
                             const cVector3f &in avGoalVal,
                             float afTime,
                             eEasing aType=eEasing_QuadInOut,
                             bool abReplaceIfExist=true)
```

Adds a fade that will fade a Vector3f variable from its current value to goal over time.

- **asVarName**: The name of the state var.
- **avGoalVal**: The goal value to fade to.
- **afTime**: How long the fade will take.
- **aType**: the type of fade, meaning in what way the transiation will be. Sigmoid is default as this means the starts and ends smoothly.
- **abReplaceIfExist**: if the variable is currently fading, that fade is replaced if true, else the function call will have no effect.

ImGui_FadeStateColor

```
void ImGui_FadeStateColor(const tString &in asVarName,
                          const cColor &in aGoalVal,
                          float afTime,
                          eEasing aType=eEasing_QuadInOut,
                          bool abReplaceIfExist=true)
```

Adds a fade that will fade a Color variable from its current value to goal over time.

- **asVarName**: The name of the state var.
- **aGoalVal**: The goal value to fade to.
- **afTime**: How long the fade will take.
- **aType**: the type of fade, meaning in what way the transiation will be. Sigmoid is default as this means the starts and ends smoothly.
- **abReplaceIfExist**: if the variable is currently fading, that fade is replaced if true, else the function call will have no effect.

ImGui_StopFade

```
void ImGui_StopFade(const tString &in asVarName)
```

Stops the fading in the fader.

ImGui_IsFading

```
bool ImGui_IsFading(const tString &in asVarName)
```

Checks if the fader is fading.

ImGui_FadeOver

```
bool ImGui_FadeOver(const tString &in asName)
```

See if a fader ended this update. Mainly to do what ever event happens when FadeState is over.

ImGui_FadeOscillateFloat

```
float ImGui_FadeOscillateFloat(const tString &in asVarName,
                               float afStart,
                               float afGoal,
                               float afTime,
                               eEasing aType=eEasing_QuadInOut)
```

This will fade a float state var value back and forth between two values and returns the current value.

- **asVarName**: The name of the state var.
- **afStart**: The start value
- **afGoal**: The goal value
- **afTime**: the time it takes to go from Start to Goal (or the opposite)
- **aType**: The type of fade between the values.

ImGui_FadeOscillateVector3f

```
cVector3f ImGui_FadeOscillateVector3f(const tString &in asVarName,
                                      const cVector3f &in avStart,
                                      const cVector3f &in avGoal,
                                      float afTime,
```

```
                                                eEasing aType=eEasing_QuadInOut)
```

This will fade a float

- **asVarName**: The name of the state var.
- **avStart**: The start value
- **avGoal**: The goal value
- **afTime**: the time it takes to go from Start to Goal (or the opposite)
- **aType**: The type of fade between the values.

ImGui_FadeOscillateColor

```
cColor ImGui_FadeOscillateColor(const tString &in asVarName,
                                const cColor &in aStart,
                                const cColor &in aGoal,
                                float afTime,
                                eEasing aType=eEasing_QuadInOut)
```

This will fade a float

- **asVarName**: The name of the state var.
- **aStart**: The start value
- **aGoal**: The goal value
- **afTime**: the time it takes to go from Start to Goal (or the opposite)
- **aType**: The type of fade between the values.

ImGui_AddTimer

```
void ImGui_AddTimer(const tString &in asName,
                    float afTime)
```

Adds a timer, that ends of afTime time

ImGui_RepeatTimer

```
bool ImGui_RepeatTimer(const tString &in asName,
                       float afTime)
```

This will create a timer that is recreated when over. It returns true whenver the timer ends. Example: if(ImGui_RepeatTimer(“mytime”, 2)){EventThatHappensEvery2Secs();}

ImGui_StopTimer

```
void ImGui_StopTimer(const tString &in asName)
```

Stops an active timer.

ImGui_TimerOver

```
bool ImGui_TimerOver(const tString &in asName)
```

See if a timer ended this update. Mainly to do what ever event happens when AddTimer is over.

ImGui_TimerExists

```
bool ImGui_TimerExists(const tString &in asName)
```

See if a timer is running.

ImGui_SetModColorMul

```
void ImGui_SetModColorMul(const cColor &in aCol)
```

Sets a color modifier to any widgets defined below. Default: (1,1,1,1)

ImGui_SetModTextColorMul

```
void ImGui_SetModTextColorMul(const cColor &in aCol)
```

Sets a text color modifier to any widgets defined below. Default: (1,1,1,1)

ImGui_SetModUseUIPos

```
void ImGui_SetModUseUIPos(bool abX)
```

Sets if the widgets below should be possible to navigate to using keyboard/gamepad movement. Default: true

---

ImGui_SetModUISizeHoriExpansion

```
void ImGui_SetModUISizeHoriExpansion(float afNeg,
                                     float afPos)
```

Expands (or contracts, use negative values) the size of the UI Pos size horitzontally. Use this to make a small UI element reachable with keyboard/gamepad movement, but by making the UI box line up with the other widgets. Turn on the debug feature Draw GUI debug to see the boxes.

ImGui_SetModUISizeVertExpansion

```
void ImGui_SetModUISizeVertExpansion(float afNeg,
                                     float afPos)
```

Expands (or contracts, use negative values) the size of the UI Pos size vertically. Use this to make a small UI element reachable with keyboard/gamepad movement, but by making the UI box line up with the other widgets. Turn on the debug feature Draw GUI debug to see the boxes.

ImGui_SetModUseInput

```
void ImGui_SetModUseInput(bool abX)
```

Sets if the widgets below should use any input. Default: true

---

ImGui_SetModRotateAngle

```
void ImGui_SetModRotateAngle(float afX)
```

Sets the rotatation angle for any widgets below. Default: 0. Applies to ImGui_DrawGfx too

---

ImGui_SetModRotateCustomPivot

```
void ImGui_SetModRotateCustomPivot(bool abX)
```

Sets if a custom pivot should be used for widget below when rotating. Default: false. Applies to ImGui_DrawGfx too

---

ImGui_SetModRotatePivot

```
void ImGui_SetModRotatePivot(const cVector2f &in avPivot)
```

Sets the custom pivot used for widget below when rotating. Default: false. Applies to ImGui_DrawGfx too ImGui_SetModRotateCustomPivot must be true for it to have effect.

ImGui_SetModGfx

```
void ImGui_SetModGfx(const cImGuiGfx &in aGfx)
```

Sets an additional image to widgets below, only used by Buttons so far. Default “”.

ImGui_ResetModifiers

```
void ImGui_ResetModifiers()
```

Resets all modifiers to default values.

ImGui_PushModifiers

```
void ImGui_PushModifiers()
```

Saves the currently set modifiers on a stack.

ImGui_PopModifiers

```
void ImGui_PopModifiers()
```

Reverts all SetMod* settings to the latest one pushed on the stack.

ImGui_PrevPressed

```
bool ImGui_PrevPressed()
```

Checks if the previously defined widget is being pressed.

ImGui_PrevBecamePressed

```
bool ImGui_PrevBecamePressed()
```

Checks if the previously defined widget just became pressed.

ImGui_PrevInFocus

```
bool ImGui_PrevInFocus()
```

Checks if the previously defined widget is in focus.

ImGui_PrevPosition

```
cVector3f ImGui_PrevPosition()
```

Gets the previously defined widget final position (might not be what is supplied as argument)

ImGui_PrevSize

```
cVector2f ImGui_PrevSize()
```

Gets the previously defined widget final size (might not be what is supplied as argument)

ImGui_PrevBecameInFocus

```
bool ImGui_PrevBecameInFocus()
```

Checks if the previously defined widget just became in focus.

ImGui_PrevWasInFocus

```
bool ImGui_PrevWasInFocus()
```

Checks if the previously defined widget just lost focus

ImGui_PrevMouseOver

```
bool ImGui_PrevMouseOver()
```

Checks if the previously defined widget has the mouse over it.

ImGui_PrevUpdated

```
bool ImGui_PrevUpdated()
```

Checks if the previously defined widget's internal values was updated somehow.

ImGui_ClearPrevData

```
void ImGui_ClearPrevData()
```

This resets any data retrieved by ImGui_Prev* to the default values. Mostly used when creating custom Widget types.

ImGui_GroupBegin

```
void ImGui_GroupBegin(const cVector3f &in avPos,
                      const cVector2f &in avSize=cVector2f_MinusOne,
                      bool abClip=false)
```

Begins the definition of a group which all subsequent Widgets will be relative to. Must be ended with ImGui_GroupEnd Nested groups are possible

- **avPos**: the position of the group (relative to any groups and layout already defined)
- **avSize**: The size of the group. Mostly useful when adding groups in a layout or when clipping.
- **abClip**: if the widgets inside the group should be clipped.

ImGui_GroupEnd

```
void ImGui_GroupEnd()
```

Ends a group started with ImGui_GroupBegin

ImGui_GetCurrentGroupPos

```
cVector3f ImGui_GetCurrentGroupPos()
```

ImGui_GetCurrentGroupSize

```
cVector2f ImGui_GetCurrentGroupSize()
```

ImGui_LayoutBegin

```
void ImGui_LayoutBegin(eImGuiLayout aType,
                       const cVector3f &in avPos=cVector3f_Zero,
                       const cVector2f &in avSize=cVector2f_MinusOne,
                       const cVector2f &in avSpacing=cVector2f_Zero)
```

Begins the definition of a layout which will affect the placement of all subsequence Widgets. Must be ended with ImGui_LayoutEnd Nested groups are possible

- **aType**: the type of the layout. X=Align horizontally, Y=Align Vertically, XY=Align left to right and go down when widget does not fit width
- **avPos**: the position of the layout. (relative to any groups and layout already defined)
- **avSize**: the size of the group. Mostly useful for XY-type or when inside another layout.
- **avSpacing**: spacing between the widgets in the layout.

ImGui_LayoutEnd

```
void ImGui_LayoutEnd()
```

Ends a layout started with ImGui_LayoutBegin

ImGui_AddLayoutHorizontalSpace

```
void ImGui_AddLayoutHorizontalSpace(float afWidth,
                                    float afHeight=)
```

Adds a horizontal space in a layout. Height is useful in XY-type of layout.

ImGui_AddLayoutVerticalSpace

```
void ImGui_AddLayoutVerticalSpace(float afHeight)
```

Adds a horizontal space in a layout

ImGui_AddItemStringW

```
void ImGui_AddItemStringW(const tWString &in asStr)
```

Adds a string item to a list. Used by some widgets like MultiToggle and MultiSelect

---

ImGui_AddItemString

```
void ImGui_AddItemString(const tString &in asStr)
```

Adds a string item to a list. Used by some widgets like MultiToggle and MultiSelect

- **asStr**: The text to use, if ImGui_SetTransCategory “” the actual string is used, else it is taken form the current tranlation file

---

ImGui_AddItemStringList

```
void ImGui_AddItemStringList(const tWString &in asStrList)
```

Adds many strings as single list. Seperate items by space or comma.

ImGui_AddItemGfx

```
void ImGui_AddItemGfx(const cImGuiGfx &in aGfx)
```

Adds a gfx item to list. Used by some widgets like MultiToggle

---

ImGui_ClearItems

```
void ImGui_ClearItems()
```

Clears all item list. Done automatically by widgets, so should mostly be used for custom widgets only.

ImGui_DoMouse

```
void ImGui_DoMouse(const cImGuiGfx &in aGfx,
                   const cVector3f &in avOffset=,
                   const cVector2f &in avSize=cVector2f_MinusOne)
```

Draws the mouse. If not called and ShowMouseAutomatically the default mouse will be shown.

- **aGfx**: the image to draw
- **avOffset**: the offset in regards to the current mouse postion.
- **avSize**: size of the mouse, if negative then size of image is used.

ImGui_DoButtonExt

```
bool ImGui_DoButtonExt(const tString &in asName,
                       const tString &in asText,
                       const cImGuiButtonData &in aData,
                       const cVector3f &in avPos=cVector3f_Zero,
                       const cVector2f &in avSize=cVector2f_MinusOne)
```

Draw and do logic for a Button Widget. Returns true if it became pressed.

- **asName**: name of widger, must be unique.
- **asText**: The text to display, if ImGui_SetTransCategory “” the actual string is used, else it is taken form the current tranlation file
- **aData**: properties of the button
- **avPos**: Position of the widget (will be relative if gruoup or layout is declared)
- **avSize**: Size of widget, if negative the default size declared in data is used.

ImGui_DoButton

```
bool ImGui_DoButton(const tString &in asName,
                    const tString &in asText,
                    const cVector3f &in avPos=cVector3f_Zero,
                    const cVector2f &in avSize=cVector2f_MinusOne)
```

Draw and do logic for a Button Widget using default data. Returns true if it became pressed.

- **asName**: name of widger, must be unique.
- **asText**: The text to display, if ImGui_SetTransCategory “” the actual string is used, else it is taken form the current tranlation file
- **avPos**: Position of the widget (will be relative if gruoup or layout is declared)
- **avSize**: Size of widget, if negative the default size declared in data is used.

ImGui_DoRepeatButtonExt

```
bool ImGui_DoRepeatButtonExt(const tString &in asName,
                             const tString &in asText,
                             const cImGuiButtonData &in aData,
                             const cVector3f &in avPos=cVector3f_Zero,
                             const cVector2f &in avSize=cVector2f_MinusOne)
```

Draw and do logic for a Repeat Button Widget. Returns true while the button is pressed.

- **asName**: name of widger, must be unique.
- **asText**: The text to display, if ImGui_SetTransCategory “” the actual string is used, else it is taken form the current tranlation file
- **aData**: properties of the button
- **avPos**: Position of the widget (will be relative if gruoup or layout is declared)
- **avSize**: Size of widget, if negative the default size declared in data is used.

ImGui_DoRepeatButton

```
bool ImGui_DoRepeatButton(const tString &in asName,
                          const tString &in asText,
                          const cVector3f &in avPos=cVector3f_Zero,
                          const cVector2f &in avSize=cVector2f_MinusOne)
```

Draw and do logic for a Repeat Button Widget using default data. Returns true while the button is pressed.

- **asName**: name of widger, must be unique.

- **asText**: The text to display, if ImGui_SetTransCategory "" the actual string is used, else it is taken form the current tranlation file
- **avPos**: Position of the widget (will be relative if gruoup or layout is declared)
- **avSize**: Size of widget, if negative the default size declared in data is used.

ImGui_DoToggleButtonExt

```
bool ImGui_DoToggleButtonExt(const tString &in asName,
                             const tString &in asText,
                             bool abDefaultChecked,
                             const cImGuiButtonData &in aData,
                             const cVector3f &in avPos=cVector3f_Zero,
                             const cVector2f &in avSize=cVector2f_MinusOne)
```

Draw and does logic for a Toggle Button Widget. Returns true if button is toggled on else false.

- **asName**: name of widger, must be unique.
- **asText**: The text to display, if ImGui_SetTransCategory "" the actual string is used, else it is taken form the current tranlation file
- **abDefaultChecked**: The default toggle value
- **aData**: properties of the widget
- **avPos**: Position of the widget (will be relative if gruoup or layout is declared)
- **avSize**: Size of widget, if negative the default size declared in data is used.

ImGui_DoToggleButton

```
bool ImGui_DoToggleButton(const tString &in asName,
                          const tString &in asText,
                          bool abDefaultChecked,
                          const cVector3f &in avPos=cVector3f_Zero,
                          const cVector2f &in avSize=cVector2f_MinusOne)
```

Draw and do logic for a Toggle Button Widget using default data. Returns true if button is toggled on else false.

- **asName**: name of widger, must be unique.
- **asText**: The text to display, if ImGui_SetTransCategory "" the actual string is used, else it is taken form the current tranlation file
- **abDefaultChecked**: The default toggle value
- **avPos**: Position of the widget (will be relative if gruoup or layout is declared)
- **avSize**: Size of widget, if negative the default size declared in data is used.

ImGui_DoSliderHorizontalExt

```
float ImGui_DoSliderHorizontalExt(const tString &in asName,
                                  float afDefaultValue,
                                  float afMin,
                                  float afMax,
                                  float afStepSize,
                                  const cImGuiSliderData &in aData,
                                  const cVector3f &in avPos=cVector3f_Zero,
                                  const cVector2f &in
avSize=cVector2f_MinusOne)
```

Draw and do logic for a Horizontal Slider Widget. Returns the current value.

- **asName**: name of widger, must be unique.
- **afDefaultValue**: the default value used on the slider.
- **afMin**: The default toggle value
- **afMax**: The max value of the slider.
- **afStepSize**: the steps the values will change if using keyboard/gamepad.
- **aData**: properties for the widget.
- **avPos**: Position of the widget (will be relative if gruoup or layout is declared)
- **avSize**: Size of widget, if negative the default size declared in data is used.

ImGui_DoSliderHorizontal

```
float ImGui_DoSliderHorizontal(const tString &in asName,
                               float afDefaultValue,
                               float afMin,
                               float afMax,
                               float afStepSize=-1,
                               const cVector3f &in avPos=cVector3f_Zero,
                               const cVector2f &in
avSize=cVector2f_MinusOne)
```

Draw and do logic for a Horizontal Slider Widget using default data. Returns the current value.

- **asName**: name of widger, must be unique.
- **afDefaultValue**: the default value used on the slider.
- **afMin**: The default toggle value
- **afMax**: The max value of the slider.
- **afStepSize**: the steps the values will change if using keyboard/gamepad.
- **avPos**: Position of the widget (will be relative if gruoup or layout is declared)
- **avSize**: Size of widget, if negative the default size declared in data is used.

ImGui_DoSliderVerticalExt

```
float ImGui_DoSliderVerticalExt(const tString &in asName,
                                float afDefaultValue,
                                float afMin,
                                float afMax,
                                float afStepSize,
                                const cImGuiSliderData &in aData,
                                const cVector3f &in avPos=cVector3f_Zero,
                                const cVector2f &in
avSize=cVector2f_MinusOne)
```

Draw and do logic for a Vertical Slider Widget. Returns the current value.

- **asName**: name of widger, must be unique.
- **afDefaultValue**: the default value used on the slider.
- **afMin**: The default toggle value
- **afMax**: The max value of the slider.
- **afStepSize**: the steps the values will change if using keyboard/gamepad.
- **aData**: properties for the widget.
- **avPos**: Position of the widget (will be relative if gruoup or layout is declared)
- **avSize**: Size of widget, if negative the default size declared in data is used.

ImGui_DoSliderVertical

```
float ImGui_DoSliderVertical(const tString &in asName,
                             float afDefaultValue,
                             float afMin,
                             float afMax,
                             float afStepSize=-1,
                             const cVector3f &in avPos=cVector3f_Zero,
                             const cVector2f &in avSize=cVector2f_MinusOne)
```

Draw and do logic for a Vertical Slider Widget using default data. Returns the current value.

- **asName**: name of widger, must be unique.
- **afDefaultValue**: the default value used on the slider.
- **afMin**: The default toggle value
- **afMax**: The max value of the slider.
- **afStepSize**: the steps the values will change if using keyboard/gamepad.
- **avPos**: Position of the widget (will be relative if gruoup or layout is declared)
- **avSize**: Size of widget, if negative the default size declared in data is used.

ImGui_DoLabelExt

```
void ImGui_DoLabelExt(const tString &in asText,
                      const cImGuiLabelData &in aData,
```

```
                        const cVector3f &in avPos=cVector3f_Zero,
                        const cVector2f &in avSize=cVector2f_MinusOne,
                        float afFontSizeMul=1)
```

Draw and do logic for a Label Widget.

- **asText**: The text to display, if ImGui_SetTransCategory “” the actual string is used, else the text string is taken from the translation file
- **aData**: properties for the widget
- **avPos**: Position of the widget (will be relative if gruoup or layout is declared)
- **avSize**: Size of widget, if negative the default size declared in data is used.
- **afFontSizeMul**: Multiplier for font size

ImGui_DoLabel

```
void ImGui_DoLabel(const tString &in asText,
                   const cVector3f &in avPos=cVector3f_Zero,
                   const cVector2f &in avSize=cVector2f_MinusOne,
                   float afFontSizeMul=1)
```

Draw and do logic for a Label Widget using default data.

- **asText**: The text to display, if ImGui_SetTransCategory “” the actual string is used, else it is taken form the current tranlation file
- **avPos**: Position of the widget (will be relative if gruoup or layout is declared)
- **avSize**: Size of widget, if negative the default size declared in data is used.
- **afFontSizeMul**: Multiplier for font size

ImGui_DoImage

```
void ImGui_DoImage(const cImGuiGfx &in aGfxImage,
                   const cVector3f &in avPos=cVector3f_Zero,
                   const cVector2f &in avSize=cVector2f_MinusOne)
```

Draw and do logic for an Image Widget

- **aGfxImage**: graphics image displayed.
- **avPos**: Position of the widget (will be relative if gruoup or layout is declared)
- **avSize**: Size of widget, if negative the size of aGfxImage is used.

ImGui_DoImageCorrectAspect

```
void ImGui_DoImageCorrectAspect(const cImGuiGfx &in aGfxImage,
                                const cVector3f &in avPos=cVector3f_Zero,
```

```
                                    const cVector2f &in
avSize=cVector2f_MinusOne)
```

Draw and do logic for an Image Widget with the image's native aspect ratio, fitting it into the specified size.

- **aGfxImage**: graphics image displayed.
- **avPos**: Position of the rectangle that the image sgould fit into. Image will be centered in the rectangle.
- **avSize**: Size of rectangle that the image should fit within.

ImGui_DoMultiToggleExt

```
int ImGui_DoMultiToggleExt(const tString &in asName,
                           int alDefaultSelectedItem,
                           uint alColumnNum,
                           const cVector2f &in avSpacing,
                           const cImGuiButtonData &in aData,
                           const cVector3f &in avPos,
                           const cVector2f &in avSize)
```

Draw and do logic for a Multi Toggle Widget. Returns currently selected button. use ImGui_AddItem and/or ImGui_AddGfx to add text and/or images to the buttons in the grid.

- **asName**: name of widger, must be unique.
- **alDefaultSelectedItem**: default button to be active.
- **alColumnNum**: number of buttons per row.
- **avSpacing**: the spacing between the buttons
- **aData**: properties for the buttons that make up the grid.
- **avPos**: Position of the widget (will be relative if gruoup or layout is declared)
- **avSize**: Size of widget (negative not allowed)

ImGui_DoMultiToggle

```
int ImGui_DoMultiToggle(const tString &in asName,
                        int alDefaultSelectedItem,
                        uint alColumnNum,
                        const cVector2f &in avSpacing,
                        const cVector3f &in avPos,
                        const cVector2f &in avSize)
```

Draw and do logic for a Multi Toggle Widget using default button data. Returns currently selected button. use ImGui_AddItem and/or ImGui_AddGfx to add text and/or images to the buttons in the grid.

- **asName**: name of widger, must be unique.

- **alDefaultSelectedItem**: default button to be active.
- **alColumnNum**: number of buttons per row.
- **avSpacing**: the spacing between the buttons
- **avPos**: Position of the widget (will be relative if gruoup or layout is declared)
- **avSize**: Size of widget (negative not allowed)

ImGui_DoCheckBoxExt

```
bool ImGui_DoCheckBoxExt(const tString &in asName,
                         const tString &in asText,
                         bool abDefaultChecked,
                         const cImGuiCheckBoxData &in aData,
                         const cVector3f &in avPos=cVector3f_Zero,
                         const cVector2f &in avSize=cVector2f_MinusOne)
```

Draw and do logic for a Check Box Widget. Return if checked or not.

- **asName**: name of widger, must be unique.
- **asText**: The text to display, if ImGui_SetTransCategory “” the actual string is used, else it is taken form the current tranlation file
- **abDefaultChecked**: the default state of the widget
- **aData**: properties for the widget
- **avPos**: Position of the widget (will be relative if gruoup or layout is declared)
- **avSize**: Size of widget, if negative the default size declared in data is used.

ImGui_DoCheckBox

```
bool ImGui_DoCheckBox(const tString &in asName,
                      const tString &in asText,
                      bool abDefaultChecked,
                      const cVector3f &in avPos=cVector3f_Zero,
                      const cVector2f &in avSize=cVector2f_MinusOne)
```

Draw and do logic for a Check Box Widget using default data. Return if checked or not.

- **asName**: name of widger, must be unique.
- **asText**: The text to display, if ImGui_SetTransCategory “” the actual string is used, else it is taken form the current tranlation file
- **abDefaultChecked**: the default state of the widget
- **avPos**: Position of the widget (will be relative if gruoup or layout is declared)
- **avSize**: Size of widget, if negative the default size declared in data is used.

ImGui_DoTextFrameExt

```
float ImGui_DoTextFrameExt(const tString &in asText,
                           const cVector2f &in avEdgeSpacing,
                           float afRowSpace,
                           float afStartRowOffset,
                           const cImGuiTextFrameData &in aData,
                           const cVector3f &in avPos,
                           const cVector2f &in avSize)
```

Draw and do logic for a Text Frame Widget. Returns the number of rows (might be fractional) that does not fit the frame, useful when making scroll slider.

- **asText**: The text to display, if ImGui_SetTransCategory “” the actual string is used, else it is taken form the current tranlation file
- **avEdgeSpacing**: The spacing around the edges of the frame
- **afRowSpace**: Space between the row
- **afStartRowOffset**: The offset for when the first row begins (can be negative, which is useful for scrolling text)
- **aData**: properties for the widget
- **avPos**: Position of the widget (will be relative if gruoup or layout is declared)
- **avSize**: Size of widget, (must be postive)

ImGui_DoTextFrame

```
float ImGui_DoTextFrame(const tString &in asText,
                        const cVector2f &in avEdgeSpacing,
                        float afRowSpace,
                        float afStartRowOffset,
                        const cVector3f &in avPos,
                        const cVector2f &in avSize)
```

Draw and do logic for a Text Frame Widget using default data. Returns the number of rows (might be fractional) that does not fit the frame, useful when making scroll slider.

- **asText**: The text to display, if ImGui_SetTransCategory “” the actual string is used, else it is taken form the current tranlation file
- **avEdgeSpacing**: The spacing around the edges of the frame
- **afRowSpace**: Space between the row
- **afStartRowOffset**: The offset for when the first row begins (can be negative, which is useful for scrolling text)
- **avPos**: Position of the widget (will be relative if gruoup or layout is declared)
- **avSize**: Size of widget, (must be postive)

ImGui_DoMultiSelectExt

```
int ImGui_DoMultiSelectExt(const tString &in asName,
```

```
                            int alDefaultSelectedItem,
                            const cImGuiMultiSelectData &in aData,
                            const cVector3f &in avPos=cVector3f_Zero,
                            const cVector2f &in avSize=cVector2f_MinusOne)
```

Draw and do logic for a Multi Select Widget. Returns currently selected item. use ImGui_AddItem* to add selection options.

- **asName**: name of widger, must be unique.
- **alDefaultSelectedItem**: the default state of the widget
- **aData**: properties for the widget
- **avPos**: Position of the widget (will be relative if gruoup or layout is declared)
- **avSize**: Size of widget, if negative the default size declared in data is used.

ImGui_DoMultiSelect

```
int ImGui_DoMultiSelect(const tString &in asName,
                        int alDefaultSelectedItem,
                        const cVector3f &in avPos=cVector3f_Zero,
                        const cVector2f &in avSize=cVector2f_MinusOne)
```

Draw and do logic for a Multi Select Widget using default data. Returns currently selected item. use ImGui_AddItem* to add selection options.

- **asName**: name of widger, must be unique.
- **alDefaultSelectedItem**: the default state of the widget
- **aData**: properties for the widget
- **avPos**: Position of the widget (will be relative if gruoup or layout is declared)
- **avSize**: Size of widget, if negative the default size declared in data is used.

ImGui_DoWindowStart

```
void ImGui_DoWindowStart(const tString &in asCaption,
                         const cImGuiWindowData &in aData,
                         const cVector3f &in avPos=cVector3f_Zero,
                         const cVector2f &in avSize=cVector2f_MinusOne,
                         bool abClip=true,
                         bool abForceCaptionFit=true)
```

Begins the definition of a Window which will affect the placement of all subsequence Widgets. Must be ended with ImGui_DoWindowEnd

- **asCaption**: the window caption.
- **aData**: properties for the widget.
- **avPos**: position of the widget (will be relative if group or layout is declared).
- **avSize**: Size of widget, if negative the default size declared in data is used.

- **abClip**: true = the window will clip any widgets declared between the window start and end that lie outside its boundaries - false = every widget will be drawn.
- **abForceCaptionFit**: (?).

ImGui_DoWindowEnd

```
void ImGui_DoWindowEnd()
```

Draw and do logic for a Window. Must be previously started with ImGui_DoWindowStart.

ImGui_DoGauge

```
void ImGui_DoGauge(const cImGuiGaugeData &in aData,
                   float afFillAmount,
                   const cVector3f &in avPos=cVector3f_Zero,
                   const cVector2f &in avSize=cVector2f_MinusOne)
```

Draw and do logic for a gauge (progress bar).

- **aData**: properties for the widget.
- **afFillAmount**: Value in the [0,1] range that sets how full the bar will be drawn
- **avPos**: position of the widget (will be relative if group or layout is declared).
- **avSize**: Size of widget, if negative the default size declared in data is used.

ImGui_SetDefaultButton

```
void ImGui_SetDefaultButton(const cImGuiButtonData &in aData)
```

ImGui_GetDefaultButton

```
cImGuiButtonData ImGui_GetDefaultButton()
```

ImGui_SetDefaultSliderHorizontal

```
void ImGui_SetDefaultSliderHorizontal(const cImGuiSliderData &in aData)
```

ImGui_GetDefaultSliderHorizontal

```
cImGuiSliderData ImGui_GetDefaultSliderHorizontal()
```

ImGui_SetDefaultSliderVertical

```
void ImGui_SetDefaultSliderVertical(const cImGuiSliderData &in aData)
```

ImGui_GetDefaultSliderVertical

```
cImGuiSliderData ImGui_GetDefaultSliderVertical()
```

ImGui_SetDefaultLabel

```
void ImGui_SetDefaultLabel(const cImGuiLabelData &in aData)
```

ImGui_GetDefaultLabel

```
cImGuiLabelData ImGui_GetDefaultLabel()
```

ImGui_SetDefaultCheckBox

```
void ImGui_SetDefaultCheckBox(const cImGuiCheckBoxData &in aData)
```

ImGui_GetDefaultCheckBox

```
cImGuiCheckBoxData ImGui_GetDefaultCheckBox()
```

ImGui_SetDefaultTextFrame

```
void ImGui_SetDefaultTextFrame(const cImGuiTextFrameData &in aData)
```

ImGui_GetDefaultTextFrame

```
cImGuiTextFrameData ImGui_GetDefaultTextFrame()
```

ImGui_SetDefaultMultiSelect

```
void ImGui_SetDefaultMultiSelect(const cImGuiMultiSelectData &in aData)
```

ImGui_GetDefaultMultiSelect

```
cImGuiMultiSelectData ImGui_GetDefaultMultiSelect()
```

ImGui_SetDefaultMouse

```
void ImGui_SetDefaultMouse(const cImGuiGfx &in aGfx)
```

ImGui_SetDefaultFont

```
void ImGui_SetDefaultFont(const cImGuiFont &in aFont)
```

ImGui_GetName

```
tString ImGui_GetName()
```

Get name of the ImGui

ImGui_SetShowMouseAutomatically

```
void ImGui_SetShowMouseAutomatically(bool abX)
```

Set if default mouse should be drawn if not rendered with DoMouse

---

ImGui_GetShowMouseAutomatically

```
bool ImGui_GetShowMouseAutomatically()
```

Get if default mouse should be drawn if not rendered with DoMouse

---

ImGui_IsFirstRun

```
bool ImGui_IsFirstRun()
```

Get if this is the first update for the ImGui

ImGui_GetTimeStep

```
float ImGui_GetTimeStep()
```

Get the current timestep

ImGui_GetTimeCount

```
float ImGui_GetTimeCount()
```

Get the time that has passed since first run.

ImGui_GetMouseVisible

```
bool ImGui_GetMouseVisible()
```

Get if the mouse is currently visible

ImGui_DrawLine

```
void ImGui_DrawLine(const cVector2f &in avStart,
                    const cVector2f &in avEnd,
                    float afZ,
                    float afThickness=1.0f,
                    const cColor &in aCol=cColor_White)
```

Draw a line between two positions. Is not affected by Layout or Group.

- **avStart**: start position.
- **avEnd**: end position.
- **afZ**: the z value
- **afThickness**: the thicnkess of the line
- **aCol**: color of the line

ImGui_DrawLineExt

```
void ImGui_DrawLineExt(const cVector2f &in avStart,
                       const cVector2f &in avEnd,
                       float afZ,
                       float afThickness,
                       const cColor &in aCol,
                       const cImGuiGfx &in aGfx)
```

Draw a line between two positions. Is not affected by Layout or Group.

- **avStart**: start position.
- **avEnd**: end position.
- **afZ**: the z value
- **afThickness**: the thicnkess of the line
- **aCol**: color of the line
- **aGfx**: an Image streched across the line. It is aligned horizontally.

ImGui_AddLineStripVertex

```
void ImGui_AddLineStripVertex(const cVector2f &in avVertex)
```

Add a vertex to a line strip for drawing. Is not affected by Layout or Group.

- **avVertex**: line vertex.

ImGui_AddLineStripNrmVertex

```
void ImGui_AddLineStripNrmVertex(const cVector2f &in avVertex)
```

Add a vertex to a line strip for drawing in normalized coords. Is not affected by Layout or Group.

- **avVertex**: line vertex.

ImGui_DrawAndClearLineStrip

```
void ImGui_DrawAndClearLineStrip(float afZ,
                                 float afThickness,
                                 const cColor &in aCol=cColor_White,
                                 const cImGuiGfx &in aGfx=cImGuiGfx())
```

Draws and clears the current line strip set up through AddLineStripVertex. Is not affected by Layout or Group.

- **afZ**: Z coord for the line.
- **afThickness**: the thicnkess of the line
- **aCol**: Color multiplier
- **aGfx**: the gfx used for a line segment

ImGui_DrawGfx

```
void ImGui_DrawGfx(const cImGuiGfx &in aGfx,
                   const cVector3f &in avPos,
                   const cVector2f &in avSize=cVector2f_MinusOne,
                   const cColor &in aCol=cColor_White)
```

Draw an image. Is not affected by Alignment, Layout or Group.

- **aGfx**: the image to be displayed
- **avPos**: The position of the images upper left corner.
- **avSize**: The size of the image, if negative then size of aGfx is used.
- **aCol**: Color multiplier

ImGui_DrawAlignedGfx

```
void ImGui_DrawAlignedGfx(const cImGuiGfx &in aGfx,
                          const cVector3f &in avPos,
```

```
                                eImGuiAlign aAlignment,
                                const cVector2f &in avSize=cVector2f_MinusOne,
                                const cColor &in aCol=cColor_White)
```

Draw an aligned image. Is not affected by Alignment, Layout or Group.

- **aGfx**: the image to be displayed
- **avPos**: The position of the pivot the image will be aligned to.
- **aAlignment**: Alignment mode
- **avSize**: The size of the image, if negative then size of aGfx is used.
- **aCol**: Color multiplier

ImGui_DrawFontW

```
void ImGui_DrawFontW(const tWString &in asText,
                     const cImGuiFont &in aFont,
                     const cVector3f &in avPos,
                     eFontAlign aAlign,
                     const cVector2f &in avSizeMul=1,
                     const cColor &in aColMul=1.0f)
```

Draw font. Is not affected by Alignment, Layout or Group.

- **asText**: the text to be displayed.
- **aFont**: the font to displayed the text with
- **avPos**: The position of the images upper right corner.
- **aAlign**: The alignment of the text.
- **avSizeMul**: A size multiplier for the text.
- **aColMul**: Color multiplier

ImGui_DrawFontW

```
void ImGui_DrawFontW(const tString &in asText,
                     const cImGuiFont &in aFont,
                     const cVector3f &in avPos,
                     eFontAlign aAlign,
                     const cVector2f &in avSizeMul=1,
                     const cColor &in aColMul=1.0f)
```

Draw font. Is not affected by Alignment, Layout or Group.

- **asText**: the text to be displayed.
- **aFont**: the font to displayed the text with
- **avPos**: The position of the images upper right corner.
- **aAlign**: The alignment of the text.

- **avSizeMul**: A size multiplier for the text.
- **aColMul**: Color multiplier

ImGui_DrawFrame

```
void ImGui_DrawFrame(const cImGuiFrameGfx &in aGfx,
                     const cVector3f &in avPos,
                     const cVector2f &in avSize=cVector2f_MinusOne,
                     const cColor &in aCol=cColor_White)
```

Draw frame. Is not affected by Alignment, Layout or Group.

- **aGfx**: The
- **avPos**: The position of the frame's upper left corner.
- **avSize**: Size of the frame to draw.
- **aCol**: Color multiplier

CheckCurrentWidgetBecamePressed

```
bool CheckCurrentWidgetBecamePressed(const tString &in asName,
                                     bool abCheckConfirm,
                                     bool abCheckMouseLeft)
```

Check if a widget has been just pressed by confirm and/or mouse, sets internal vars for this too

---

CheckCurrentWidgetIsPressed

```
bool CheckCurrentWidgetIsPressed(const tString &in asName,
                                 bool abCheckConfirm,
                                 bool abCheckMouseLeft)
```

Check if a widget is being pressed by confirm and/or mouse, sets internal vars for this too

---

CheckIsPressedAction

```
bool CheckIsPressedAction(bool abCheckConfirm,
                          bool abCheckMouseLeft)
```

Quick way to check if confirm and/or mouse was just pressed.

CheckBecamePressedAction

```
bool CheckBecamePressedAction(bool abCheckConfirm,
                              bool abCheckMouseLeft)
```

Quick way to check if confirm and/or mouse is being pressed.

GetDefaultOrCurrentInt

```
int GetDefaultOrCurrentInt(uint64 alDefaultVarId,
                           uint64 alCurrentVarId,
                           int alDefaultValue)
```

Gets the int value from the CurrentVar, if the value in default equals alDefaultVarId, else alDefaultVarId is returned. Used as a helper to have default param values.

GetDefaultOrCurrentFloat

```
float GetDefaultOrCurrentFloat(uint64 alDefaultVarId,
                               uint64 alCurrentVarId,
                               float afDefaultValue)
```

Gets the float value from the CurrentVar, if the value in default equals alDefaultVarId, else alDefaultVarId is returned. Used as a helper to have default param values.

LockMouseFocus

```
void LockMouseFocus()
```

When this is called, the focus will not change during this or the next update.

MouseFocusIsLocked

```
bool MouseFocusIsLocked()
```

See if mouse is locked

CalcWidgetSize

```
cVector2f CalcWidgetSize(const cVector2f &in avArgSize,
                         const cVector2f &in avDefaultSize)
```

If an element in ArgSize is negative, the output for the element will have the value of DefaultSize instead.

SetupWidgetRect

```
void SetupWidgetRect(const cVector3f &in avInPos,
                     const cVector2f &in avInSize,
                     cVector3f &out avOutPos,
                     cVector2f &out avOutSize,
                     const cVector2f &in avDefaultSize,
                     const cImGuiGfx &in aGfx)
```

Gets the final postion and size for a widget taking into account aligntment, groups, layout, etc.

GetUsedGfxSize

```
cVector2f GetUsedGfxSize(const cImGuiGfx &in aGfx,
                         const cVector2f &in avCustomSize)
```

Get the size of a graphics by using axis value of aGfx instead of custom, if the custom value is negative.

GetGfxSize

```
cVector2f GetGfxSize(const cImGuiGfx &in aGfx)
```

Get the size of a gfx.

GetFontLengthW

```
float GetFontLengthW(const cImGuiFont &in aFont,
                     float afSizeMul,
```

```
                        const tWString &in asText)
```

Get the length that a string will be given a certain font.

GetFontLengthW

```
float GetFontLengthW(const cImGuiFont &in aFont,
                     float afSizeMul,
                     const tString &in asText)
```

Get the length that a string will be given a certain font.

GetFontWordWrapRows

```
void GetFontWordWrapRows(const cImGuiFont &in aFont,
                         float afSizeMul,
                         const tString &in asText,
                         float afLineWidth,
                         array< tWString > &out avLines)
```

Get an array containing each line of a text using word wrap.

GetFontWordWrapRowsW

```
void GetFontWordWrapRowsW(const cImGuiFont &in aFont,
                          float afSizeMul,
                          const tWString &in asText,
                          float afLineWidth,
                          array< tWString > &out avLines)
```

Get an array containing each line of a text using word wrap.

ImGui_ResizeFontToFit

```
float ImGui_ResizeFontToFit(const tString &in asText,
                            cImGuiFont &aFont,
                            float afMaxWidth=-1,
                            float afSizeMultiplier=1.0f)
```

Modifies the size of the font so that it will fit within the specified width.

ImGui_ResizeFontToFit

```
float ImGui_ResizeFontToFit(const tWString &in asText,
                            cImGuiFont &aFont,
                            float afMaxWidth=-1,
                            float afSizeMultiplier=1.0f)
```

Modifies the size of the font so that it will fit within the specified width.

CheckMouseOver

```
bool CheckMouseOver(const cVector3f &in avPos,
                    const cVector2f &in avSize)
```

Check if mouse if over a certain rect. This will take rotation into account.

GetIdFromNameAndCheckCollision

```
uint64 GetIdFromNameAndCheckCollision(const tString &in asName,
                                      int alTableIdx)
```

This gets the id for a name while also checking for hash collision. Used when you want get ids directly. Table index is the type of id you want: 0: gfx, 1: font, 2: widget, 3: state var

---

ImGui_DrawReadableBackground

```
void ImGui_DrawReadableBackground(cVector3f vPos,
                                  cVector2f vSize,
                                  float fAlpha)
```

ImGui_DrawReadableLabel

```
void ImGui_DrawReadableLabel(const tString &in asText,
                             cVector3f vPos,
                             cVector2f vSize,
```

```
                                float fAlpha,
                                cColor aColor=cColor(1,
                                1,
                                1),
                                float afFontSize=20.0f)
```

ImGui_DrawReadableTextFrame

```
void ImGui_DrawReadableTextFrame(const tString &in asText,
                                 cVector3f vPos,
                                 cVector2f vSize,
                                 float fAlpha,
                                 cColor aColor=cColor(1,
                                 1,
                                 1),
                                 float afFontSize=20.0f)
```

ImGui_PreloadImage

```
void ImGui_PreloadImage(const tString &in asFile,
                        eImGuiGfx aType=eImGuiGfx_Image)
```

Preloads an image

- **asFile**: image file to preload
- **aType**: Type of the image

Gui_CreateCameraTexture

```
void Gui_CreateCameraTexture(tString asName,
                             cVector2l avSize,
                             uint alFrameRate,
                             float afFOV,
                             float afNearPlane,
                             float afFarPlane)
```

asName, Name of the special texture avSize, Resolution of the camera alFrameRatem, Frame rate of the camera, never higher than the games frame rate afFOV, Field of view afNearPlane, Near plane used for culling afFarPlane, Max distance the camera can render, lower = faster!

Gui_DestroyCameraTexture

```
void Gui_DestroyCameraTexture(tString asName)
```

Destroys camera texture

Gui_SetCameraTextureMatrix

```
void Gui_SetCameraTextureMatrix(tString asName,
                                cMatrixf a_mtxCamera)
```

Use a matrix to set the location and rotation of the camera

Gui_AttachCameraTextureToEntity

```
void Gui_AttachCameraTextureToEntity(tString asName,
                                     tString asEntity)
```

asName, Name of the texture asEntity, Name of the entity to attach to, works with CameraAnimationAreas!
