ppScript Functions Lists

Table of Contents

Table of Contents

Strings

string stringNew(void);

Creates a New Empty String

string stringDup(string s);

Duplicates a string

void stringFree(string s);

Frees a string

bool stringCmp(string s1, string s2);

Returns true if the two string are equals, false otherwise

void stringCpy(string s1, string s2);

Copy a string into another one

void stringCat(string s1, string s2);

Concatenates s2 at the end of s1

string stringFromBool(bool v);

Creates a string from a boolean (’true’ or ’false’)

bool boolFromString(string str);

Creates a boolean from a string (’true’ or ’false’), false beeing the default).

string stringFromChar(char c);

Creates a string with one character

string stringChr(int i);

Creates a string with one character of code *i*

string stringFromInt(int i);

Creates a string form an int

int intFromString(string str);

Returns an int by parsing the string. Returns 0 if the string is not a number.

string stringFromFloat(float f);

Creates a string from a float

float floatFromString(string str);

Returns a float by parsing the string. Returns 0 if the string is not a number.

int stringLen(string s);

Returns the number of characters into the string s ('s.len()')

int stringGetLen(string s);

Defines the len property of the string s ('s.len')

char stringGetNthChar(string s, int pos);

Returns the pos character (0 is the first character, 's.nthChar(pos)')

void stringSetNthChar(string s, int pos, char c);

Sets the pos character ('s.nthChar(pos) = 'a'')

void stringReplace(string s, string p, string r);

Not implemented yet

void stringTrim(string s);

Removes the trailing and leading spaces ('s.trim()')

int stringFind(string s, string p, int pos);

Find the first occurence of the string p into the string s, starting at character pos ('s.find("ab", 0)')

string stringToken(string s, string separator);

Cut the string in tokens separated by separator. First, call stringToken with the string to be cutted, then with and empty string: param1 = stringToken(myString, ","); param2 =

stringToken("", ",");

string stringSub(string s, int stpos, int edpos);

Returns a part of the string s (’s.sub(0, s.len - 1)’)

Typed Iterartors

extern typedef xxxIterator;

Defines a xxx typed Iterator.

void xxxIteratorFree(xxxIterator iterator);

Releases an Iterator on type xxx.

void xxxIteratorMoveFirst(xxxIterator iterator);

Sets the current element of the Iterator to the first element.

void xxxIteratorMoveNext(xxxIterator iterator);

Moves the current element of the Iterator to the next element.

bool xxxIteratorIsLast(xxxIterator iterator);

Returns true if the current element of the Iterator is the last one.

xxx xxxIteratorGetCurrentElem(xxxIterator iterator);

Returns the current element of the Iterator.

int xxxIteratorGetNumElems(xxxIterator iterator);

Returns the number of elements in the Iterator.

xxx xxxIteratorNthElem(xxxIterator iterator, int elemNum);

Returns the nth element of the Iterator.

Typed lists

extern typedef xxxList

Defines list of element of type xxx.

xxxList xxxListNew(void);

Creates an empty list of element of type xxx.

void xxxListFree(xxxList list);

Frees a list of element of type xxx.

void xxxListAddElem(xxxList list, xxx elem);

Adds an element at the beginning of a list of element of type xxx

void xxxListDelElem(xxxList list, xxx elem);

Removes the first element equal to elem from the list.

void xxxListMoveFirst(xxxList list);

Sets the current element to the first of the list.

void xxxListMoveNext(xxxList list);

Moves the current element to the next of the list.

bool xxxListIsLast(xxxList list)

Returns true if the current element is the last of the list.

xxx xxxListCurrentElem(xxxList list);

Returns the current element of the list.

int xxxListNumElems(xxxList list);

Returns the number of elements in the list.

xxx xxxListGetNthElem(xxxList list, int elemNum);

Returns the nth element of the list.

xxx xxxListRmNthElem(xxxList list, int elemNum);

Removes the nth element of the list.

int xxxListElemPos(xxxList list, xxx elem);

Returns the position of the first occurence of elem in the list. If the element is not found, returns -1.

void xxxListInvert(xxxList list);

Inverts the order of the elements of the list.

Typed Dynamic Arrays

extern typedef xxxDArray;

Defines dynamic array of type xxx.

xxxDArray xxxDArrayNew(int numElems);

Creates a dynamic array of type xxx with numElems elements.

void xxxDArrayFree(xxxDArray array);

Frees a dynamic array of elements of type xxx.

void xxxDArrayResize(xxxDArray array, int numElems);

Resizes the dynamic array, preserving elements.

void xxxDArrayMoveFirst(xxxDArray array);

Sets the current element to the first of the dynamic array.

void xxxDArrayMoveNext(xxxDArray array);

Moves the current element to the next of the dynamic array.

bool xxxDArrayIsLast(xxxDArray array);

Returns true if the current element is the last of the dynamic array.

xxx xxxArrayGetCurrentElem(xxxDArray array);

Returns the current element of the dynamic array.

int xxxDArrayGetNumElems(xxxDArray array);

Returns the number of elements in the dynamic array.

xxx xxxDArrayGetNthElem(xxxDArray array, int elemNum);

Returns the nth element of the dynamic array.

void xxxDArraySetNthElem(xxxDArray array, int elemNum, xxx elem);

Sets the nth element of the dynamic array.

Math functions

int abs(int a);

Returns the absolute value af the integer a (’a = abs(a)’)

float fabs(float a);

Returns the absolute value af the float a (’a = fabs(a)’)

float sqrt(float a);

Returns the square root of the float a (’b = sqrt(a)’)

int ceil(float a);

Returns the integer part + 1 of the float a (’b = ceil(1.8)’

int floor(float a);

Returns the integer part of the float a (’b = ceil(1.8)’

int round(float a);

Returns the nearest integer of the float a (’b = ceil(1.8)’

float toFloat(int a);

Converts int to float.

float pi(void);

Returns the pi Constant

float cos(float a);

Returns the cosine of the angle a in radian (’b = cos(pi())’)

float sin(float a);

Returns the sine of the angle a in radian (’b = sin(pi())’)

float tan(float a);

Returns the tangent of the angle a in radian (’b = tan(pi())’)

float acos(float a);

Returns the arccosine in radian of the float a (’b = acos(a)’)

float asin(float a);

Returns the arcsine in radian of the float a (’b = asin(a)’)

float atan(float a);

Returns the arctangent in radian of the float a (’b = atan(a)’)

float atan2(float y, float x);

Returns the arctangent in radian of the float y/x (’b = atan2(y, x)’)

float exp(float a);

Returns the exponentiation of the float a (’b = exp(a)’)

float log(float a);

Returns the neperian logarithm of the float a (’b = log(a)’)

float log10(float a);

Returns the decimal logarithm of the float a (’b = log10(a)’)

float pow(float a, float b);

Returns a raised at the power b (’c = pow(a, b)’)

void srand(int seed);

Sets the seed for the random function.

int rand(void);

Returns a random number (from 1 to 2^31-2).

float frand(void);

Returns a random number (from 0 to 1).

File

Reading

typedef readFile;

Defines a read file object

string filePath(string name);

Returns the file path (without the file name).

string fileName(string name);

Returns the file without path (only the file name).

string fileSuffix(string name);

Returns the file extension (after the last '.').

string fileWithoutSuffix(string name);

Returns the file without extension (before the last '.').

bool setWorkingFolder(string path);

Set the working folder. Returns true if succeed, false otherwise.

readFile readFileTextNew(string name);

Creates a text read file object

readFile readFileBinaryNew(string name, bool bigEndian);

Defines a binary read file object

void readFileClose(readFile file);

Releases a read file object

int readFileGetSize(readFile file);

Returns size of the file in bytes. Can be trusted only with binary files.

bool readFileIsEof(readFile file);

Returns true if the end of the read file object is reached

string readFileGetLine(readFile file);

Returns a string from a read file object

int readFileGetByte(readFile file);

Returns a byte from a read file object

char readFileGetChar(readFile file);

Returns a byte from a read file object

int readFileGetWord(readFile file);

Returns a word (2 bytes) from a read file object

int readFileGetDWord(readFile file);

Returns a double word (4 bytes) from a read file object

float readFileGetFloat(readFile file);

Returns a float from a read file object

void readFileSeek(readFile file, int pos);

Skips *pos* bytes. Resets the EOF criteria.

void readFileSeekSet(readFile file, int pos);

Sets the next byte to read *pos* bytes after the beginning. Resets the EOF criteria.

int readFileGetTell(readFile file);

Returns the position of the next byte to be read.

Writing

typedef writeFile;

Defines a write file object

writeFile writeFileTextNew(string name);

Creates a text write file object

writeFile writeFileBinaryNew(string name, bool bigEndian);

Creates a binary write file object

void writeFileClose(writeFile file);

Releases a write file object

void writeFilePutLine(writeFile file, string s);

Puts a string into a write file object

void writeFilePutByte(writeFile file, int b);

Puts a byte into a write file object

void writeFilePutWord(writeFile file, int w);

Put a word (2bytes) into a write file object

void writeFilePutDWord(writeFile file, int d);

Puts a double word (4 bytes) into a write file object

void writeFilePutInt(writeFile file, int d);

Puts a double word (4 bytes) into a write file object

void writeFilePutFloat(writeFile file, float f);

Puts a float into a write file object

PPGUI Interface

RGB color Functions

typedef rgbColor;

Defines a rgb color

rgbColor rgbColorNew(void);

Returns a new rgb color (black).

int rgbColorToInt(rgbColor rgb);

Returns an integer that represent an rgb color.

rgbColor intToRgbColor(int color);

Returns a rgbColor.

int rgbColorGetRed(rgbColor rgb);

Returns the red component of an rgb color (0 -> 255).

float rgbColorGetRedFloat(rgbColor rgb);

Returns the red component of an rgb color (0.0 -> 1.0).

int rgbColorGetGreen(rgbColor rgb);

Returns the green component of an rgb color (0 -> 255).

float rgbColorGetGreenFloat(rgbColor rgb);

Returns the green component of an rgb color (0.0 -> 1.0).

int rgbColorGetBlue(rgbColor rgb);

Returns the blue component of an rgb color (0 -> 255).

float rgbColorGetBlueFloat(rgbColor rgb);

Returns the blue component of an rgb color (0.0 -> 1.0).

rgbColor rgbColorSetRed(rgbColor rgb, int red);

Sets the red component of an rgb color (0 -> 255).

rgbColor rgbColorSetRedFloat(rgbColor rgb, float red);

Sets the red component of an rgb color (0.0 ->
1.0).

rgbColor rgbColorSetGreen(rgbColor rgb, int green);

Sets the green component of an rgb color (0 ->
255).

rgbColor rgbColorSetGreenFloat(rgbColor rgb, float green);

Sets the green component of an rgb color (0.0 ->
1.0).

rgbColor rgbColorSetBlue(rgbColor rgb, int blue);

Sets the blue component of an rgb color (0 -> 255).

rgbColor rgbColorSetBlueFloat(rgbColor rgb, float blue);

Sets the blue component of an rgb color (0.0 ->
1.0).

int rgbColorGetAlpha(rgbColor rgb);

Returns the alpha component of an rgb color (0 ->
255).

rgbColor rgbColorSetAlpha(rgbColor rgb, int alpha);

Sets the alpha component of an rgb color (0 ->
255).

float rgbColorGetAlphaFloat(rgbColor rgb);

Returns the alpha component of an rgb color (0.0 ->
1.0).

rgbColor rgbColorSetAlphaFloat(rgbColor rgb, float alpha);

Sets the alpha component of an rgb color (0.0 ->
1.0).

rgbColor rgbColorBlend(rgbColor rgb1, rgbColor rgb2, float percent);

Returns a rgb color composed of *percent* of rgb1 and (1-percent) of rgb2.

string stringFromRgbColor(rgbColor rgb);

Returns a string representing the color.

rgbColor rgbColorFromString(string str);

Returns the color represented in the string str.

Raster

typedef ppgRaster;

Defines a raster. A raster is a generic image, independent from the drawing device

ppgRaster ppgRasterNew(int width, int height, rgbColor rgb);

Creates a new raster, filled with the rgb color

void ppgRasterFree(ppgRaster rst);

Frees a raster

ppgRaster ppgRasterDup(ppgRaster rst);

Duplicates a raster

int ppgRasterGetWidth(ppgRaster rst);

Returns a raster width

int ppgRasterGetHeight(ppgRaster rst);

Returns the raster height

ppgRaster ppgRasterLoad(string filName);

Reads a raster from a file. The file extention determines the used reader. Returns null if failed.

bool ppgRasterSave(ppgRaster rst, string fileName);

Writes a raster to a file. The file extention determines the format. Returns false if failed

bool ppgRasterSaveTgaAlpha(ppgRaster rst, ppgRaster alphaRst, string fileName);

Writes a raster to a tga file with an alpha channel defined in alphaRst. The 2 raster should have the same size. Returns false if failed

ppgRaster ppgRasterSub(ppgRaster rst, int left, int top, int width, int height);

Returns a part of a raster

ppgRaster ppgRasterScale(ppgRaster rst, int width, int height);

Scales a raster

void ppgRasterSetPixel(ppgRaster rst, int x, int y, rgbColor rgb);

Sets a raster pixel color

rgbColor ppgRasterGetPixel(ppgRaster rst, int x, int y);

Returns a raster pixel color

Monochrome Raster

typedef ppgMonoRaster;

Defines a monochrome raster. A monochrome raster is a generic image, independent from the drawing device

ppgMonoRaster ppgMonoRasterNew(int width, int height, rgbColor rgb);

Creates a new monochrome raster, filled with the rgb color

void ppgMonoRasterFree(ppgMonoRaster rst);

Frees a monochrome raster

ppgMonoRaster ppgMonoRasterDup(ppgMonoRaster rst);

Duplicates a monochrome raster

int ppgMonoRasterGetWidth(ppgMonoRaster rst);

Returns a monochrome raster width

int ppgMonoRasterGetHeight(ppgMonoRaster rst);

Returns the monochrome raster height

void ppgMonoRasterSetPixel(ppgMonoRaster rst, int x, int y, int level);

Sets a monochrome raster pixel value [0, 255].

int ppgMonoRasterGetPixel(ppgMonoRaster rst, int x, int y);

Returns a monochrome raster pixel value.

ppgRaster ppgMonoRasterToRaster(ppgMonoRaster rst);

Creates a raster from a monochrome raster.

ppgMonoRaster ppgRasterToMonoRaster(ppgRaster rst);

Creates a monochrome raster from a raster.

void ppgRasterIntoMonoRaster(ppgRaster rst, ppgMonoRaster mRst);

Copy the data(luminance) from raster into the monochrome raster.

Raw drawing Functions

All the drawing functions draw on the current window, with the current color, inside the current clipping rectangle (if one is defined). **void setClipRectangle(int left, int top, int right, int bottom);**

Defines the clip rectangles. No Drawing function will put something outside of this rectangle

void resetClipRectangle(void);

Removes the clip rectangle

void setColor(rgbColor rgb);

Sets the current drawing color

void putString(string str, int x, int y);

Put a string

int stringWidth(string str);

Returns the string width, in pixels

int stringHeight(string str);

Returns the string heights, in pixel

void putPixel(int x, int y);

Draws a pixel

void putLine(int x1, int y1, int x2, int y2);

Draws a line

void fillRectangle(int x1, int y1, int x2, int y2);

Draws a filled rectangle

void putEllipse(int xc, int yc, int a, int b);

Draws an ellipse

void fillEllipse(int xc, int yc, int a, int b);

Draws a filled ellipse

void putArcEllipse(int xc, int yc, int a, int b, float teta1, float teta2);

Draws an arc of ellipse

void xorLine(int x1, int y1, int x2, int y2);

Draws a line, XORing the current color with the actual colors

void xorArcEllipse(int xc, int yc, int a, int b, float teta1, float teta2);

Draws an arc of ellipse, XORing the current color with the actual colors

void putDownBox(int left, int top, int width, int height);

Draws a shaded box (borders of a pressed button)

void putUpBox(int left, int top, int width, int height);

Draws a shaded box (borders of a released button)

Raw Images Functions

typedef ppgImage;

Defines an image. An image is an abstract object (dependant of the device), that can be drawn in a window

ppgImage ppgImageGet(int left, int top, int right, int bottom);

Returns an image of a part of the current window

int ppgImageWidth(ppgImage img);

Returns the width of an image

int ppgImageHeight(ppgImage img);

Returns ther height of an images

void ppgImagePut(ppgImage img, int left, int top);

Draws an image in the current window

void ppgImageFree(ppgImage img);

Frees an image

ppgImage ppgImageFromRaster(ppgRaster rst);

Returns an image built from a raster

ppgRaster ppgRasterFromImage(ppgImage img);

Returns aa raster built from an image

ppgImage ppgImageFromRasterQuick(ppgRaster rst);

Returns an image built from a raster, but without taking care of transparencies

PPGUI Objects

typedef ppgWin;

Defines a window

typedef ppgObj;

Defines widgets (graphical objects like buttons, radio box...)

typedef ppgWinPos enum = { ppgRight, ppgTop, ppgLeft, ppgBottom, ppgCenter };

The possible locations when a window is opened

typedef bool ppgCallback(ppgObj obj);

Defines the callback function type of widgets

Predefined windows

typedef ppgFileDlgType enum = { ppgDlgLoad, ppgDlgSave };

Defines the two possible types of file selector

string selectFile(string filter, ppgFileDlgType dlgFlag);

Opens a modal file selection window

string askString(string title, string prompt, string ini);

Opens a modal window to ask a value

int displayWarningMsg(string title, string prompt, string button1Label, string button2Label, string button3Label);

Opens a modal window, with one to three buttons('displayWarningMsg("Test", "Proceed anyway ?", "Ok", "cancel", "")'

rgbColor selectColor(rgbColor rgb);

Opens the color selection modal window

void showDoneInit(int numTotal, int doneRgb, int toDoRgb, string title);

Opens and Initializes the process progres window

void showDone(int numDone);

Updates the process progres window

void showDoneEnd(void);

Closes process progres window

Windows

ppgWin ppgWinNew(void);

Creates a new window (not displayed).

void ppgWinFree(ppgWin win);

Frees a window

void ppgWinAddObj(ppgWin window, ppgObj obj);

Adds a widget to a window (’w.addObj(o)’)

void ppgWinShow(ppgWin window, ppgWinPos hPos, ppgWinPos vPos, string title, int width, int height);

Displays the window (’w.show(ppgLeft, ppgTop, "Title", 300, 200)’)

Widget Functions

int ppgObjGetLeft(ppgObj obj);

Returns the left position of a widget (’ i = w.left’)

int ppgObjGetTop(ppgObj obj);

Returns the top position of a widget (’ i = w.top’)

int ppgObjGetWidth(ppgObj obj);

Returns the width of a widget (’ i = w.width’)

int ppgObjGetHeight(ppgObj obj);

Returns the height of a widget (’ i = w.height’)

void ppgObjPos(ppgObj obj, int left, int top);

Sets the position and the size of a widget (’w.pos(10,10, 80,20)’)

void ppgObjSize(ppgObj obj, int width, int height);

Sets the size of a widget (’w.size(80, 20)’)

void ppgObjSetCaption(ppgObj obj, string val);

Sets the caption of a widget (if the widget do have a caption)

string ppgObjGetCaption(ppgObj obj);

Returns the caption of a widget, null if the widget don’t have a caption

void ppgObjEnabled(ppgObj obj, bool enabled);

Enables/Disables the widget (the widget stays visible, but inactive)

void ppgObjSetCallback(ppgObj obj, ppgCallback func);

Sets the callback function of a widget

typedef bool ppgModifyCallback(ppgObj obj);

Defines the modify callback of a widget

void ppgObjSetObjModifyCallback(ppgObj obj, ppgModifyCallback func);

Sets the modify callback of a widget

Button

typedef ppgButton ppgObj;

Defines the button widget

ppgButton ppgButtonNew(int left, int top, int width, int height, string label);

Creates a new button

Menu bar

typedef ppgMenuBar ppgObj;

DEfines a menu bar widget

ppgMenuBar ppgMenuBarNew(int delta);

Creates a new menu bar widget

void ppgMenuBarAddMenu(ppgMenuBar menuBar, ppgMenu menu);

Adds a menu to a menu bar

User defined widget

typedef ppgEvent enum = { ppgEvButtonDown, ppgEvButtonUp, ppgEvDragMouse, ppgEvKeyPressed, ppgEvLButtonDown, ppgEvLButtonUp, ppgEvResizeWindow, ppgEvMouseMove};

Defines all the events

typedef void ppgDrawCallback(ppgObj obj);

Defines the draw callback function of a user defined widget

typedef bool ppgEventCallback(ppgEvent ev, int x, int y, int key, ppgObj obj);

Defines the event callback function of a user defined widget

ppgObj ppgFreeObjNew(int left, int top, int width, int height, ppgDrawCallback draw, ppgEventCallback event);

Creates a new user defined widget

Misc

void wait(int milisec);

Waits milisec unless an event is raised

Radio button

typedef ppgRadio ppgObj;

Defines the radio button widget

ppgRadio ppgRadioNew(int left, int top, int width, int height, string label, bool state);

Creates a new radio button widget

bool ppgRadioGetState(ppgRadio obj);

Returns the state of a radio button

void ppgRadioSetState(ppgRadio obj, bool state);

Sets the state of a radio button

Input

typedef ppgInput ppgObj;

Defines an input widget (one line of free text)

ppgInput ppgInputNew(int left, int top, int width, int height, string content);

Creates a new input widget

Prompt

typedef ppgPrompt ppgObj;

Defines a prompt widget (static text)

ppgPrompt ppgPromptNew(int left, int top, string label);

Creates a new Prompt widget

Selection list

typedef ppgSelect ppgObj;

Defines a selection list widget

ppgSelect ppgSelectNew(int left, int top, int width, int height, int minSelected, int maxSelected);

Creates a new selection list widget

int ppgSelectGetNumSelected(ppgSelect obj);

Returns the number of selected items

int ppgSelectGetNthSelected(ppgSelect obj, int pos);

Returns the nth selected item

void ppgSelectSetNthItem(ppgSelect obj, int pos, string item);

Defines the nth item

string ppgSelectGetNthItem(ppgSelect obj, int pos);

Returns the nth item

void ppgSelectAddItem(ppgSelect obj, int pos, string item);

Adds an item in the list

void ppgSelectRmItem(ppgSelect obj, int pos);

Removes an item from the list

Combo box

typedef ppgCombo ppgObj;

Defines a combo box widget

ppgCombo ppgComboNew(int left, int top, int width, int height);

Creates a new combo box widget

int ppgComboGetSelected(ppgCombo obj, int pos);

Returns the selected item number

void ppgComboSetNthItem(ppgCombo obj, int pos, string item);

Sets the nth items

string ppgComboGetNthItem(ppgCombo obj, int pos);

Returns the nth items

void ppgComboAddItem(ppgCombo obj, int pos, string item);

Adds an item in the list

void ppgComboRmItem(ppgCombo obj, int pos);

Removes an item from the list

Group

typedef ppgGroup ppgObj;

Defines a group widget

ppgGroup ppgGroupNew(int left, int top, int width, int height, string title);

Creates a new group widget

Shape widget

ppgObj ppgLineNew(int left, int top, int width, int height, int rgb);

Creates a line widget

ppgObj ppgRectangleNew(int left, int top, int width, int height, int rgb);

Creates a fill rectangle widget

ppgObj ppgEllipseNew(int left, int top, int width, int height, int rgb);

Creates a fill ellipse widget

Menu

typedef ppgMenu ppgObj;

Defines a menu widget

typedef ppgItemState enum = {   ppgItemEnabled, ppgItemDisabled,   ppgItemChecked,   ppgItemMarked };

The possible menu item states

ppgMenu ppgMenuNew(int left, int top, int width, int height, string title);

Creates a new Menu widget

void ppgMenuAddItem(ppgMenu menu, string label, ppgCallback func);

Adds an item in a menu (’ menu.addItem("Load...", &menuCallback)’)

void ppgMenuRmItem(ppgMenu menu, int itemNum);

Removes an item from the menu

void ppgMenuItemState(ppgMenu menu, int itemNum, ppgItemState state);

Sets the state of an item

void ppgMenuItemSetSubMenu(ppgMenu menu, int itemNum, ppgMenu subMenu);

Sets the submenu of an item

void ppgWinPopupMenuShow(ppgWin win, ppgMenu menu, int x, int y);

Opens a popup menu in the window win

Albatross 3D Interface

The functions used by Albatross 3D are:

- **ppmScriptType ppmGetScriptType(void)** -Required- called just after loading the script to determines its type.
- **void ppmInit(void)** called when the script is loaded the script.
- **string ppmGetScriptIconRc(void)**
- **string ppmGetScriptStringRc(void)**
- **string ppmGetScriptHelpRc(void)**
- **string ppmGetScriptFileType(void)**
- **string ppmGetScriptPrefix(void)**
- **void ppmMain(void) or void ppmGenerateTexture(point3d pos, vector3d normal)** -Required- the main function of the script, called each time the script has to run
- **void ppmClose(void)** called when the script is released (closing ppModeler).

typedef ppmScriptType enum = { ppmExportScript, ppmImportScript, ppmCreateObjScript, ppmModifierScript, ppmObjectScript, ppmFaceScript, ppmEdgeScript, ppmVertexScript, ppmMaterialScript, ppmCommandScript, ppmRenderScript, ppmRenderCameraParamsScript, ppmRenderLightParamsScript, ppmRenderPointLightParamsScript, ppmRenderSpotLightParamsScript, ppmRenderMeshParamsScript, ppmRenderMaterialParamsScript, ppmRenderSceneParamsScript, ppmHeightFieldFilterScript, ppmTextureScript };

Defines the possible script types.

Global Objects

Associative Arrays

typedef ppmAssociativeArray;

Defines associative arrays.

void ppmAssociativeArrayAdd(ppmAssociativeArray array, string key, string value, bool hided);

Adds/Updates a key-value association.

void ppmAssociativeArrayDel(ppmAssociativeArray array, string key);

Removes a key-value association.

int ppmAssociativeArrayNumElems(ppmAssociativeArray array);

Returns the number of key-value associations.

bool ppmAssociativeArrayHasKey(ppmAssociativeArray array, string key);

Returns true if the associative array has the key *key*.

string ppmAssociativeArrayValue(ppmAssociativeArray array, string key);

Returns the value associated with key.

stringIterator ppmAssociativeArrayPrefixedKeys(ppmAssociativeArray array, string prefix);

Returns the all the keys that start with prefix.

bool ppmAssociativeArrayHasVisibleKey(ppmAssociativeArray array, string key);

Returns true if the associative array has the key *key*.

string ppmAssociativeArrayVisibleValue(ppmAssociativeArray array, string key);

Returns the visible value associated with key.

stringIterator ppmAssociativeArrayPrefixedVisibleKeys(ppmAssociativeArray array, string prefix);

Returns the all the keys that start with prefix.

Materials

typedef ppmMaterial;

Defines 3D Objects’ Materials

ppmMaterial ppmMaterialNew(string materialName);

Creates a new material.

string ppmMaterialGetName(ppmMaterial mat);

Returns the material’s name.

void ppmMaterialSetName(ppmMaterial mat, string name);

Sets the material’s name (do nothing if it’s one of the two predefined material).

rgbColor ppmMaterialGetAmbientColor(ppmMaterial mat);

Returns the material’s ambient color (rgb).

void ppmMaterialSetAmbientColor(ppmMaterial mat, rgbColor rgb);

Sets the material’s ambient color (rgb).

rgbColor ppmMaterialGetDiffuseColor(ppmMaterial mat);

Returns the material’s diffuse color (rgb).

void ppmMaterialSetDiffuseColor(ppmMaterial mat, rgbColor rgb);

Sets the material’s diffuse color (rgb).

rgbColor ppmMaterialGetSpecularColor(ppmMaterial mat);

Returns the material’s specular color (rgb).

void ppmMaterialSetSpecularColor(ppmMaterial mat, rgbColor rgb);

Sets the material’s specular color (rgb).

int ppmMaterialGetShininess(ppmMaterial mat);

Returns the material’s shininess (0 -> 255).

void ppmMaterialSetShininess(ppmMaterial mat, int shininess);

Sets the material’s shininess.

float ppmMaterialGetTransparency(ppmMaterial mat);

Returns the material’s transparency (0 -> 1).

void ppmMaterialSetTransparency(ppmMaterial mat, float transparency);

Sets the material’s transparency.

float ppmMaterialGetReflexivity(ppmMaterial mat);

Returns the material’s reflexivity (0 -> 1).

void ppmMaterialSetReflexivity(ppmMaterial mat, float reflexivity);

Sets the material’s reflexivity.

float ppmMaterialGetBumpScale(ppmMaterial mat);

Returns the material’s bump scale (> 0).

void ppmMaterialSetBumpScale(ppmMaterial mat, float bumpScale);

Sets the material’s bumpScale.

int ppmMaterialGetSmoothGroup(ppmMaterial mat);

Returns the material’s smooth group. 0 equals flat shading, all the other values defines a smooth group.

void ppmMaterialSetSmoothGroup(ppmMaterial mat, int smoothGroup);

Sets the material’s smooth group.

bool ppmMaterialGetNoShading(ppmMaterial mat);

Returns true if the material has no shading.

void ppmMaterialSetNoShading(ppmMaterial mat, bool noShading);

Sets the material’s shading property.

bool ppmMaterialGetDoubleSided(ppmMaterial mat);

Returns true if the material can be seen from both sides.

void ppmMaterialDoubleSided(ppmMaterial mat, bool doubleSided);

Sets the material’s double sided property.

int ppmMaterialGetData1(ppmMaterial txt);

Returns the data1 byte (0->255) associated with the material. This data can be changed by Albatross 3D, so it must be used localy.

void ppmMaterialSetData1(ppmMaterial txt, int data1);

Sets the material’s data1 byte (clamped to [0, 255]).

float ppmMaterialGetRefractionIndex(ppmMaterial txt);

Returns the refraction index associated with the material. This data can be changed by Albatross 3D, so it must be used localy.

void ppmMaterialSetRefractionIndex(ppmMaterial txt, float refractionIndex);

Sets the material’s refraction index.

ppmAssociativeArray ppmMaterialGetValues(ppmMaterial mat);

Returns the material’s associative Array.

Materials and Textures

typedef ppmTexturedMaterial;

An association of a Material and textures

ppmMaterial ppmTexturedMaterialGetMaterial(ppmTexturedMaterial txtMat);

Returns the material.

ppmTexture ppmTexturedMaterialGetTextureMap(ppmTexturedMaterial txtMat);

Returns the material texture map.

ppmTexture ppmTexturedMaterialGetBumpMap(ppmTexturedMaterial txtMat);

Returns the material texture map.

ppmTexture ppmTexturedMaterialGetGlossMap(ppmTexturedMaterial txtMat);

Returns the material texture map.

ppmTexture ppmTexturedMaterialGetAlphaMap(ppmTexturedMaterial txtMat);

Returns the material texture map.

Raster Drawing Functions

void ppgRasterPlot(ppgRaster raster, int x, int y, rgbColor rgb);

Plots the given point with the given color.

void ppgRasterLine(ppgRaster raster, int x1, int y1, int x2, int y2, rgbColor rgb);

Draws a line between the points (x1, y1) and (x2, y2) with the given color.

void ppgRasterRectangle(ppgRaster raster, int x1, int y1, int x2, int y2, rgbColor rgb);

Draws a rectangle defined the points (x1, y1) and (x2, y2) with the given color.

void ppgRasterFillRectangle(ppgRaster raster, int x1, int y1, int x2, int y2, rgbColor rgb);

Draws a filled rectangle defined the points (x1, y1) and (x2, y2) with the given color.

void ppgRasterEllipse(ppgRaster raster, int xCenter, int yCenter, int xRadius, int yRadius, rgbColor rgb);

Draws an ellipse of center (xCenter, yCenter) and radius (xRadius, yRadius) with the given color.

void ppgRasterFillEllipse(ppgRaster raster, int xCenter, int yCenter, int xRadius, int yRadius, rgbColor rgb);

Draws a filled ellipse of center (xCenter, yCenter) and radius (xRadius, yRadius) with the given color.

Spline Point

typedef splinePoint2d;

Defines a 2d spline point

splinePoint2d splinePoint2dNew(void);

Creates a new 2d spline point.

splinePoint2d splinePoint2dDup(splinePoint2d splinePoint);

Duplicates a 2d spline point.

void splinePoint2dFree(splinePoint2d splinePoint);

Frees a 2d spline point.

void splinePoint2dCopy(splinePoint2d dst, splinePoint2d src);

Copy a 2d spline point.

float splinePoint2dGetX(splinePoint2d splinePoint);

Returns the x component of a 2d spline point

float splinePoint2dGetY(splinePoint2d splinePoint);

Returns the y component of a 2d spline point

float splinePoint2dGetSize(splinePoint2d splinePoint);

Returns the size component of a 2d spline point

rgbColor splinePoint2dGetColor(splinePoint2d splinePoint);

Returns the color component of a 2d spline point

int splinePoint2dGetLevel(splinePoint2d splinePoint);

Returns the level component of a 2d spline point

int splinePoint2dGetBaseLevel(splinePoint2d splinePoint);

Returns the base level component of a 2d spline point

int splinePoint2dGetHighLevel(splinePoint2d splinePoint);

Returns the high level component of a 2d spline point

void splinePoint2dSetX(splinePoint2d splinePoint, float x);

Sets the x component of a 2d spline point

void splinePoint2dSetY(splinePoint2d splinePoint, float y);

Sets the y component of a 2d spline point

void splinePoint2dSetSize(splinePoint2d splinePoint, float size);

Sets the size component of a 2d spline point

void splinePoint2dSetColor(splinePoint2d splinePoint, rgbColor rgb);

Sets the color component of a 2d spline point

void splinePoint2dSetLevel(splinePoint2d splinePoint, int level);

Sets the level component of a 2d spline point

void splinePoint2dSetBaseLevel(splinePoint2d splinePoint, int level);

Sets the level component of a 2d spline point

void splinePoint2dSetHighLevel(splinePoint2d splinePoint, int level);

Sets the level component of a 2d spline point

void ppgRasterCatmulSpline(ppgRaster raster, splinePoint2dDArray pointArray);

Draws a thick Catmul-Rom Spline.

void ppgMonoRasterCatmulSpline(ppgMonoRaster raster, splinePoint2dDArray pointArray);

Draws a thick Catmul-Rom Spline.

void ppgMonoRasterBumpCatmulSpline(ppgMonoRaster raster, splinePoint2dDArray pointArray);

Draws a thick Catmul-Rom Spline with a bump brush.

ppgRaster ppgRasterResize(ppgRaster raster, int width, int height);

Returns a new raster of the given size.

void ppgRasterFilter(ppgRaster raster, string filterId);

Filter the raster. The available filters are:

LIGHT: Lighter**DARK**: Darker**LSOFT**: Low Soften**MSOFT**: Medium Soften**HSOFT**: High Soften**LSHARP**: Low Sharpen**MSHARP**: Medium Sharpen**SSHARP**: Soft Sharpen**HBLUR**: Horizontal Blur**VBLUR**: Vertical Blur**EDGE**: Edge**EMBOSS**: Emboss**B&W**: To Black and White**INVERT**: Invert Colors**VFLIP**: Horizontal Flip**HFLIP**: Vertical Flip**NOISE**: Add noise.

void ppgMonoRasterFilter(ppgMonoRaster raster, string filterId);

Filter the monochrome raster. The available filters are the same than in the previous function.

void ppgMonoRasterPlot(ppgMonoRaster raster, int x, int y, int level);

Plots the given point with the given level.

void ppgMonoRasterLine(ppgMonoRaster raster, int x1, int y1, int x2, int y2, int level);

Draws a line between the points (x1, y1) and (x2, y2) with the given level.

void ppgMonoRasterRectangle(ppgRaster raster, int x1, int y1, int x2, int y2, int level);

Draws a rectangle defined the points (x1, y1) and (x2, y2) with the given level.

void ppgMonoRasterFillRectangle(ppgMonoRaster raster, int x1, int y1, int x2, int y2, int level);

Draws a filled rectangle defined the points (x1, y1) and (x2, y2) with the given level.

void ppgMonoRasterEllipse(ppgMonoRaster raster, int xCenter, int yCenter, int xRadius, int yRadius, int level);

Draws an ellipse of center (xCenter, yCenter) and radius (xRadius, yRadius) with the given level.

void ppgMonoRasterFillEllipse(ppgMonoRaster raster, int xCenter, int yCenter, int xRadius, int yRadius, int level);

Draws a filled ellipse of center (xCenter, yCenter) and radius (xRadius, yRadius) with the given level.

3D Vectors

typedef vector3d;

Defines a 3D vector

vector3d vector3dNew(void);

Creates a new vector

vector3d vector3dDup(vector3d v);

Duplicates a vector

void vector3dFree(vector3d v);

Frees a vector

void vector3dCopy(vector3d dst, vector3d src);

Copy a vector

float vector3dGetX(vector3d v);

Returns the x component of a vector

float vector3dGetY(vector3d v);

Returns the y component of a vector

float vector3dGetZ(vector3d v);

Returns the z component of a vector

void vector3dSetX(vector3d v, float x);

Sets the x component of a vector

void vector3dSetY(vector3d v, float y);

Sets the y component of a vector

void vector3dSetZ(vector3d v, float z);

Sets the z component of a vector

int vector3dGetData1(vector3d v);

Returns the data1 [0->255] associated to a vector

void vector3dSetData1(vector3d v, int data);

Sets the data1 [0->255] associated to a vector

int vector3dGetData2(vector3d v);

Returns the data2 [0->255] associated to a vector

void vector3dSetData2(vector3d v, int data);

Sets the data2 [0->255] associated to a vector

int vector3dGetIntData(vector3d v);

Returns the integer data associated to a vector

void vector3dSetIntData(vector3d v, int data);

Sets the integer data associated to a vector

float vector3dLength(vector3d v);

Returns the length of a vector

void vector3dNormalise(vector3d v);

Normalizes a vector (v.length = 1)

void vector3dNeg(vector3d v);

v = -v

void vector3dScale(vector3d v, float s);

Scales a vector

float vector3dDot(vector3d v1, vector3d v2);

Returns the dot product of 2 vectors

void vector3dCross(vector3d result, vector3d v1, vector3d v2);

Returns the cross product of 2 vectors

void vector3dAdd(vector3d result, vector3d v1, vector3d v2);

Adds 2 vectors

void vector3dSub(vector3d result, vector3d v1, vector3d v2);

Substracts 2 vectors

void vector3dMin(vector3d result, vector3d v1, vector3d v2);

Sets the result vector with the minimum, on each components) of the vector v1 and v2

void vector3dMax(vector3d result, vector3d v1, vector3d v2);

Sets the result vector with the maximum, on each components) of the vector v1 and v2

string vector3dToString(vector3d v);

Returns a string with each component separeted by a ’;’.

void vector3dFromString(vector3d v, string str);

Sets the vector v with the values stored in str (created by vector3dToString).

3D Points

typedef point3d;

Defines a 3D points

point3d point3dNew(void);

Creates a new point

point3d point3dDup(point3d v);

Duplicates a point

void point3dFree(point3d p);

Frees a point

void point3dCopy(point3d dst, point3d src);

Copy a point

float point3dGetX(point3d v);

Returns the x property

float point3dGetY(point3d v);

Returns the y property

float point3dGetZ(point3d v);

Returns the z property

void point3dSetX(point3d v, float x);

Sets the x property

void point3dSetY(point3d v, float y);

Sets the y property

void point3dSetZ(point3d v, float z);

Sets the z property

int point3dGetData1(point3d v);

Returns the data1 [0->255] associated to a point

void point3dSetData1(point3d v, int data);

Sets the data1 [0->255] associated to a point

int point3dGetData2(point3d v);

Returns the data2 [0->255] associated to a point

void point3dSetData2(point3d v, int data);

Sets the data2 [0->255] associated to a point

int point3dGetIntData(point3d v);

Returns the integer data associated to a point

void point3dSetIntData(point3d v, int data);

Sets the integer data associated to a point

float point3dLength(point3d p);

Returns the distance to origin.

void point3dNeg(point3d p);

Negates a point

void point3dScale(point3d p, float s);

Scales a point

void point3dAdd(point3d result, point3d p, vector3d v);

Adds a vector and a point (move the point)

void point3dSum(point3d result, point3d p1, point3d p2);

Adds two point (for average).

void point3dIntDiv(point3d p1, int n);

Divides the coordiantes of the point by n (for average).

void point3dSub(point3d result, point3d p, vector3d v);

Substracts a vector to a point (move the point)

void point3dDisplacement(vector3d result, point3d p1, point3d p2);

Sets the result vector with the displacement form p2 to p1.)

void point3dMin(point3d result, point3d p1, point3d p2);

Sets the coordinates of result with the minimum coordinates ofp1 and p2

void point3dMax(point3d result, point3d v1, point3d v2);

Sets the coordinates of result with the maximum coordinates ofp1 and p2

float point3dDot(point3d pt, vector3d v);

Returns the dot product of a point and a vector

float point3dGetWeight(point3d v);

Returns the weight property (for sds meshes).

void point3dSetWeight(point3d v, float weight);

Sets the weight property (for sds meshes).

string point3dToString(point3d v);

Returns a string with each component separeted by a ’;’.

void point3dFromString(point3d v, string str);

Sets the point v with the values stored in str (created by point3dToString).

3D Matrixes

typedef matrix3d;

Defines a 3D Matrix

matrix3d matrix3dNew(void);

Creates a new nul matrix

matrix3d matrix3dNewEuler(int alpha, int beta, int gamma);

Creates a rotation matrix. alpha is the rotation around the x axis, beta is the rotation around the y axis, gamma is the rotation around the z axis. The rotations are applied in that order. The angles are in degree.

matrix3d matrix3dNewAxeRotate(vector3d axe, int angle);

Creates a rotation matrix. angle is in degree.

matrix3d matrix3dNewScale(float sx, float sy, float sz);

Creates a scale matrix

matrix3d matrix3dNewMove(point3d orgPos, vector3d orgDir, point3d endPos, vector3d endDir);

Creates the matrix that transforms the orgPos point into the endPos point and the orgDir vector into end Dir vector. The matrix is a combinaison of a translation matrix and a rotation matrix

matrix3d matrix3dDup(matrix3d m);

Duplicates a matrix

void matrix3dFree(matrix3d m);

Frees a matrix

float matrix3dGetElem(matrix3d m, int pos);

Returns the nth component of a matrix

void matrix3dSetElem(matrix3d m, int pos, float v);

Sets the nth component of a matrix

void matrix3dMul3x3(matrix3d result, matrix3d a, matrix3d b);

Composes the 2 vector transformations a and b. the result matrix must not be a or b.

void matrix3dMul4x4(matrix3d result, matrix3d a, matrix3d b);

Composes the 2 point transformations a and b. the result matrix must not be a or b.

void matrix3dInvert(matrix3d m);

Invert the matrix. This function works only if the matrix m is an arbitrary composition of rotations and translations

void matrix3dNormalise(matrix3d m);

Normalizes the matrix (ensure that the matrix is a composition of rotations and translations)

void vector3dTransform(vector3d result, vector3d a, matrix3d m);

Sets the vector result as the transformation of the vector a by the matrix m

void point3dTransform(point3d result, point3d a, matrix3d m);

Sets the point result as the transformation of the point a by the matrix m

Texture Coordinates

typedef uvVector;

Defines a texture coordinate

uvVector uvVectorNew(void);

Creates a new texture coordinate

uvVector uvVectorDup(uvVector txtCoord);

Duplicates a texture coordinate

void uvVectorFree(uvVector txtCoord);

Frees a texture coordinate

void uvVectorCopy(uvVector dst, uvVector src);

Copy a texture coordinate

int uvVectorGetU(uvVector txtCoord);

Returns the v component of a texture coordinate

int uvVectorGetV(uvVector txtCoord);

Returns the v component of a texture coordinate

void uvVectorSetU(uvVector txtCoord, int u);

Sets the u component of a texture coordinate

void uvVectorSetV(uvVector txtCoord, int v);

Sets the v component of a texture coordinate

2D Vectors

typedef vector2d;

Defines a 2D vector

vector2d vector2dNew(void);

Creates a new vector

vector2d vector2dDup(vector2d v);

Duplicates a vector

void vector2dFree(vector2d v);

Frees a vector

void vector2dCopy(vector2d dst, vector2d src);

Copy a vector

float vector2dGetX(vector2d v);

Returns the x component of a vector

float vector2dGetY(vector2d v);

Returns the y component of a vector

void vector2dSetX(vector2d v, float x);

Sets the x component of a vector

void vector2dSetY(vector2d v, float y);

Sets the y component of a vector

float vector2dLength(vector2d v);

Returns the length of a vector

void vector2dNormalise(vector2d v);

Normalizes a vector (v.length = 1)

float vector2dDot(vector2d v1, vector2d v2);

Returns the dot product of 2 vectors

float vector2dCross(vector2d v1, vector2d v2);

Returns the cross product of 2 vectors

void vector2dAdd(vector2d v1, vector2d v2);

Adds 2 vectors

void vector2dSub(vector2d v1, vector2d v2);

Substracts 2 vectors

void vector2dMin(vector2d result, vector2d v1, vector2d v2);

Sets the result vector with the minimum on each components of the vector v1 and v2

void vector2dMax(vector2d result, vector2d v1, vector2d v2);

Sets the result vector with the maximum on each components of the vector v1 and v2

2D Points

typedef point2d;

Defines a 2D points

point2d point2dNew(void);

Creates a new point

point2d point2dDup(point2d v);

Duplicates a point

void point2dFree(point2d p);

Frees a point

void point2dCopy(point2d dst, point2d src);

Copy a point

float point2dGetX(point2d v);

Returns the x property

float point2dGetY(point2d v);

Returns the y property

void point2dSetX(point2d v, float x);

Sets the x property

void point2dSetY(point2d v, float y);

Sets the y property

void point2dAdd(point2d p, vector2d v);

Adds a vector and a point (move the point)

void point2dSum(point2d p1, point2d p2);

Adds two point (for average).

void point2dSub(point2d p, vector2d v);

Substracts a vector to a point (move the point)

void point2dDisplacement(vector2d result, point2d p1, point2d p2);

Sets the result vector with the displacement form
p2 to p1.)

void point2dMin(point2d result, point2d p1, point2d p2);

Sets the coordinates of result with the minimum
coordinates ofp1 and p2

void point2dMax(point3d result, point2d v1, point2d v2);

Sets the coordinates of result with the maximum
coordinates ofp1 and p2

float point2dDot(point2d pt, vector2d v);

Returns the dot product of a point and a vector

2D Matrixes

typedef matrix2d;

Defines a 2D Matrix

matrix2d matrix2dNew(void);

Creates a new indentity matrix

matrix2d matrix2dNewRotate(int angle);

Creates a rotation matrix. The angle is in degree.

matrix2d matrix2dNewScale(float sx, float sy);

Creates a scale matrix

matrix2d matrix2dDup(matrix2d m);

Duplicates a matrix

void matrix2dFree(matrix2d m);

Frees a matrix

float matrix2dGetElem(matrix2d m, int pos);

Returns the nth component of a matrix

void matrix2dSetElem(matrix2d m, int pos, float v);

Sets the nth component of a matrix

void matrix2dMul(matrix2d a, matrix2d b);

Composes the 2 vector transformations a and b. the result matrix must not be a or b.

void vector2dTransform(vector2d a, matrix2d m);

Sets the vector result as the transformation of the vector a by the matrix m

void point2dTransform(point2d a, matrix2d m);

Sets the point result as the transformation of the point a by the matrix m

Textures

typedef ppmTexture;

Defines Textures

ppmTexture ppmTextureNew(ppgRaster raster, string name);

Adds a new texture.

string ppmTextureGetName(ppmTexture txt);

Returns the texture's name.

void ppmTextureSetName(ppmTexture txt, string name);

Sets the texture's name.

int ppmTextureGetData1(ppmTexture txt);

Returns the data1 byte (0->255) associated with the texture. This data can be changed by Albatross 3D, so it must be used localy.

void ppmTextureSetData1(ppmTexture txt, int data1);

Sets the texture's data1 byte (clamped to [0, 255]).

int ppmTextureGetData2(ppmTexture txt);

Returns the data2 byte (0->255) associated with the texture. This data can be changed by Albatross 3D, so it must be used localy.

void ppmTextureSetData2(ppmTexture txt, int data2);

Sets the texture's data2 byte (clamped to [0, 255]).

bool ppmTextureIsMonochrome(ppmTexture txt);

Returns true if the texture is monochrome (can be used for gloss map and bump map), false otherwise.

ppgRaster ppmTextureGetRaster(ppmTexture txt);

Returns the texture's raster.

void ppmTextureSetRaster(ppmTexture txt, ppgRaster rst);

Sets the texture's raster.

Scripted Materials

Palettes

typedef ppmPaletteColor;

Defines palette colors.

rgbColor ppmPaletteColorGetColor(ppmPaletteColor palCol);

Returns the palette color’s color.

void ppmPaletteColorSetColor(ppmPaletteColor palCol, rgbColor col);

Sets the palette color’s color.

int ppmPaletteColorGetPosition(ppmPaletteColor palCol);

Returns the palette color’s position.

void ppmPaletteColorSetPosition(ppmPaletteColor palCol, int pos);

Sets the palette color’s position.

typedef ppmPalette;

Defines Palettes

ppmPalette ppmPaletteNew(rgbColor colorStart, rgbColor colorEnd);

Creates a new palette.

void ppmPaletteFree(ppmPalette palette);

Frees a palette.

ppmPaletteColorIterator ppmPaletteGetColors(ppmPalette palette);

Returns the palette palette color.

void ppmPaletteAddColor(ppmPalette palette, rgbColor color, int position);

Adds a new palette color to the palette.

void ppmPaletteDelColor(ppmPalette palette, int colorIndex);

Removes a palette color from the palette. The first and the last palette color can’t be removed.

rgbColor ppmPaletteGetColor(ppmPalette palette, float pos);

Returns the palette color at position pos [0, 255].

string ppmPaletteToString(ppmPalette palette);

Returns a string representing the palette palette.

void ppmPaletteFromString(ppmPalette palette, string str);

Sets the palette palette with the values stored in
str (created by ppmPaletteToString).

Monochrome Palettes

typedef ppmPaletteLevel;

Defines monochrome palette level.

int ppmPaletteLevelGetLevel(ppmPaletteLevel palLev);

Returns the palette level’s level.

void ppmPaletteLevelSetLevel(ppmPaletteLevel palLev, int level);

Sets the palette level’s level.

int ppmPaletteLevelGetPosition(ppmPaletteLevel palLev);

Returns the palette level’s position.

void ppmPaletteLevelSetPosition(ppmPaletteLevel palLev, int pos);

Sets the palette level’s position.

typedef ppmMonoPalette;

Defines Monochrome Palettes

ppmMonoPalette ppmMonoPaletteNew(int levelStart, int levelEnd);

Creates a new monochrome palette.

void ppmMonoPaletteFree(ppmMonoPalette monoPalette);

Frees a monochrome palette.

ppmPaletteLevelIterator ppmPaletteGetLevels(ppmMonoPalette monoPalette);

Returns the palette levels.

void ppmMonoPaletteAddLevel(ppmMonoPalette monoPalette, int level, int position);

Adds a new palette level to the monochrome palette.

void ppmMonoPaletteDelLevel(ppmMonoPalette monoPalette, int levelIndex);

Removes a palette level from the monochrome palette. The first and the last palette levels can't be removed.

int ppmMonoPaletteGetLevel(ppmMonoPalette monoPalette, float pos);

Returns the monochrome palette level at position pos [0, 255].

string ppmMonoPaletteToString(ppmMonoPalette monoPalette);

Returns a string representing the monochrome palette monoPalette.

void ppmMonoPaletteFromString(ppmMonoPalette monoPalette, string str);

Sets the monochrome palette pmonoPalette with the values stored in str (created by ppmMonoPaletteToString).

Generated Texture

float ppmPerlinNoise(point3d pos);

Returns the noise amplitude at pos.

float ppmPerlinFrequencyNoise(point3d pos, float frequence);

Returns the noise amplitude at pos for a given noise frequency.

float ppmPerlinTurbulence(point3d pos, float frequence);

Returns the turbulence amplitude at pos.

float ppmDotArray(point3d pos, float frequence);

Returns the distance.

rgbColor ppmPerlinPerturbColor(rgbColor rgb, float amplitude, point3d pos, float frequence);

Returns a color changed randomly.

rgbColor ppmPerlinLighterColor(rgbColor rgb, float amplitude, point3d pos, float frequence);

Returns a color lighted randomly.

rgbColor ppmPerlinDarkerColor(rgbColor rgb, float amplitude, point3d pos, float frequence);

Returns a color darked randomly.

Modeling Objects

Bones

typedef ppmBone;

Defines Bones

string ppmBoneGetName(ppmBone bone);

Returns the bone’s name.

void ppmBoneSetName(ppmBone bone, string name);

Sets the bone’s name.

point3d ppmBoneGetCenter(ppmBone bone);

Returns the bone’s center.

void ppmBoneSetCenter(ppmBone bone, point3d center);

Sets the bone’s center.

float ppmBoneGetLength(ppmBone bone);

Returns the bone’s length.

void ppmBoneSetLength(ppmBone bone, float length);

Sets the bone’s length.

ppmBone ppmBoneGetFather(ppmBone bone);

Returns the bone’s father.

intIterator ppmBoneGetVertexIndexes(ppmBone bone);

Returns the vertex indexes attached to the bone.

floatIterator ppmBoneGetWeights(ppmBone bone);

Returns the weights of the vertices attached to the bone.

float ppmBoneGetMinTrans(ppmBone bone);

Returns the bone’s minimum translation for translation join.

void ppmBoneSetMinTrans(ppmBone bone, float minTrans);

Sets the bone’s join minimum translation.

float ppmBoneGetMaxTrans(ppmBone bone);

Returns the bone’s maximum translation for translation join.

void ppmBoneSetMaxTrans(ppmBone bone, float maxTrans);

Sets the bone’s join maximum translation.

float ppmBoneGetMinXRot(ppmBone bone);

Returns the bone’s minimum angle of rotation around the x axis of the bone (for rotation join).

void ppmBoneSetMinXRot(ppmBone bone, float minAngle);

Sets the bone’s join minimum angle of rotation around the bone’s x axis.

float ppmBoneGetMaxXRot(ppmBone bone);

Returns the bone’s maximum angle of rotation around the x axis of the bone (for rotation join).

void ppmBoneSetMaxXRot(ppmBone bone, float maxAngle);

Sets the bone’s join maximum angle of rotation around the bone’s x axis.

float ppmBoneGetMinYRot(ppmBone bone);

Returns the bone’s minimum angle of rotation around the y axis of the bone (for rotation join).

void ppmBoneSetMinYRot(ppmBone bone, float minAngle);

Sets the bone’s join minimum angle of rotation around the bone’s y axis.

float ppmBoneGetMaxYRot(ppmBone bone);

Returns the bone’s maximum angle of rotation around the y axis of the bone (for rotation join).

void ppmBoneSetMaxYRot(ppmBone bone, float maxAngle);

Sets the bone’s join maximum angle of rotation around the bone’s y axis.

float ppmBoneGetMinZRot(ppmBone bone);

Returns the bone’s minimum angle of rotation around the z axis of the bone (for rotation join).

void ppmBoneSetMinZRot(ppmBone bone, float minAngle);

Sets the bone’s join minimum angle of rotation around the bone’s z axis.

float ppmBoneGetMaxZRot(ppmBone bone);

Returns the bone’s maximum angle of rotation around the z axis of the bone (for rotation join).

void ppmBoneSetMaxZRot(ppmBone bone, float maxAngle);

Sets the bone’s join maximum angle of rotation around the bone’s z axis.

ppmBoneIterator ppmBoneGetSons(ppmBone bone);

Returns the bone’ sons.

Joints

typedef ppmJoint;

Defines Joints

ppmBone ppmJointGetBone(ppmJoint joint);

Returns the joint’s bone.

float ppmJointGetTrans(ppmJoint joint);

Returns the joint’s translation.

void ppmJointSetTrans(ppmJoint joint, float trans);

Sets the joint’s translation.

float ppmJointGetXRot(ppmJoint joint);

Returns the joint’s rotation angle around the x axis of the associated bone.

void ppmJointSetXRot(ppmJoint joint, float angle);

Sets the joint’s rotation angle around the x axis of the associated bone.

float ppmJointGetYRot(ppmJoint joint);

Returns the joint’s rotation angle around the y axis of the associated bone.

void ppmJointSetYRot(ppmJoint joint, float angle);

Sets the joint’s rotation angle around the y axis of the associated bone.

float ppmJointGetZRot(ppmJoint joint);

Returns the joint’s rotation angle around the z axis of the associated bone.

void ppmJointSetZRot(ppmJoint joint, float angle);

Sets the joint’s rotation angle around the z axis of the associated bone.

Keyframes

typedef ppmKeyframe;

Defines keyframes

float ppmKeyframeGetTime(ppmJoint joint);

Returns the keyframe time in second.

void ppmKeyframeSetTime(ppmJoint joint, float time);

Sets the keyframe’s time (in second).

ppmJointIterator ppmKeyframeGetJoints(ppmKeyframe joint);

Returns the keyframe’s joints.

Sequences

typedef ppmSequence;

Defines Sequences

string ppmSequenceGetName(ppmSequence sequence);

Returns the sequence’s name.

void ppmSequenceSetName(ppmSequence sequence, string name);

Sets the sequence’s name.

float ppmSequenceGetDuration(ppmSequence sequence);

Returns the sequence’s duartion in seconds.

ppmKeyframeIterator ppmSequenceGetKeyframes(ppmSequence sequence);

Returns the sequence’s keyframes.

Faces

typedef ppmFace;

Defines 3D Object’s Faces

ppmFace ppmFaceNew(int numVertices);

Creates a new face.

void ppmFaceFree(ppmFace face);

Frees a face

int ppmFaceGetNumVertices(ppmFace face);

Returns a face number of vertices.

intIterator ppmFaceGetVertexIndexes(ppmFace face);

Returns the vertex indexes that made the face.

void ppmFaceSetNthVertexIndex(ppmFace face, int index, int vertexNum);

Sets the nth vertex index of the face. Setting an index outside the range of vertices of the associated object will crash the program if the face is displayed !!

intIterator ppmFaceGetNormalIndexes(ppmFace face);

Returns the normal indexes.

vector3d ppmFaceGetFaceNormal(ppmFace face);

Returns the face's normal.

float ppmFaceGetFaceD(ppmFace face);

Returns the D component of face's plane eqation: n.x*x+n.y*y+n.z*z=D, where n is the face's normal.

void ppmFaceAddVertex(ppmFace face, int pos, int vertexIndex, int normalIndex);

Adds a new vertex into a face.

void ppmFaceRmVertex(ppmFace face, int pos);

Removes a vertex from a face.

ppmMaterial ppmFaceGetMaterial(ppmFace face);

Returns the face's material.

void ppmFaceSetMaterial(ppmFace face, ppmMaterial mat);

Sets the face's material.

int ppmFaceGetTextureMapIndex(ppmFace face);

Returns the face's diffuse texture index (in the mesh texture list). Returns '-1' if no texture is associated with the face.

void ppmFaceSetTextureMapIndex(ppmFace face, int index);

Sets the face's diffuse texture index (in the mesh texture list). index must be a valid one (between 0 and the number of element of the mesh texture list - 1), or negative to remove the texturing of the face. A default set of uv is created if necessary.

int ppmFaceGetGlossMapIndex(ppmFace face);

Returns the face's gloss texture index (in the mesh texture list). Returns '-1' if no gloss texture is associated with the face.

int ppmFaceGetBumpMapIndex(ppmFace face);

Returns the face’s bump texture index (in the mesh texture list). Returns ’-1’ if no bump texture is associated with the face.

int ppmFaceGetAlphaMapIndex(ppmFace face);

Returns the face’s alpha texture index (in the mesh texture list). Returns ’-1’ if no alpha texture is associated with the face.

uvVectorIterator ppmFaceGetTextureCoordinates(ppmFace face);

Returns the texture coordinates.

int ppmFaceGetData1(ppmFace face);

Returns the face data1 (a temporary integer that can be used to flag faces).

void ppmFaceSetData1(ppmFace face, int data1);

Sets the face data1 flag.

void ppmFaceFlip(ppmFace face);

Flips a face.

Edges

typedef ppmEdge;

Defines 3D Objects’ edge

int ppmEdgeGetVertex1(ppmEdge edge);

Returns the first vertex index of the edge.

int ppmEdgeGetVertex2(ppmEdge edge);

Returns the second vertex index of the edge.

ppmFace ppmEdgeGetFace1(ppmEdge edge);

Returns the first face of the edge.

ppmFace ppmEdgeGetFace2(ppmEdge edge);

Returns the second face of the edge.

float ppmEdgeGetWeight(ppmEdge edge);

Returns the weight of the edge (for sds objects).

void ppmEdgeSetWeight(ppmEdge edge, float weight);

Sets the weight [0, 1] of the edge (for sds objects).

Meshes

typedef ppmObject;

Defines 3D Objects

typedef ppmObjectType enum = { ppmUndefined, ppmCube, ppmCylinder, ppmSphere, ppmLath, ppmExtrude, ppmPyramid, ppmPolygon, ppmBlob, ppmGrid, ppmTerrain, ppmPatch, ppmPlant, ppmCloseLath, ppmTexturedPolygon, ppmSurface, ppmSkin, ppmCone, ppmTorus, ppmMesh, ppmNull };

Defines the possible object types.

ppmObject ppmObjectNew(string name, point3dDArray vertexArray, ppmFaceDArray faceArray);

Creates a new object with the vertices of vertexArray and the faces of faceArray.The faces of faceArray and the vertices of vertexArray must be freed if needed.

ppmObject ppmObjectNewMesh(string name, point3dDArray vertexArray, ppmFaceDArray faceArray);

Same than *ppmObjectNew* but for an object of type ppmMesh.

void ppmObjectFree(ppmObject obj);

Frees an objec.

void ppmObjectUpdate(ppmObject obj, point3dDArray vertexArray, ppmFaceDArray faceArray);

Updates an object with the vertices of vertexArray and the faces of faceArray.The faces of faceArray and the vertices of vertexArray must be freed if needed. The object stay where it is updated.

void ppmObjectUpdatePos(ppmObject obj);

Updates an object Position to the place where it was moved before.

void ppmObjectToMeshType(ppmObject obj);

Sets the type of the object to ’ppmMesh’.

ppmObjectType ppmObjectGetType(ppmObject obj);

Returns the object type.

matrix3d ppmObjectGetTransform(ppmObject obj);

Returns the object transformation.

void ppmObjectTransform(ppmObject obj, matrix3d m);

Transforms the object.

vector3d ppmObjectGetCubeSize(ppmObject obj);

Returns the size of the cube (if the object is a cube) before transformation. The returned vector must be freed.

vector3d ppmObjectGetCubeSubdivision(ppmObject obj);

Returns the subdivision on each axes of the cube (if the object is a cube) before transformation. The returned vector must be freed.

float ppmObjectGetSphereRadius(ppmObject obj);

Returns the radius of the sphere (if the object is a sphere) before transformation.

vector3d ppmObjectGetSphereSubdivision(ppmObject obj);

Returns the subdivision on each lattitude and longitude of the sphere (if the object is a sphere) before transformation. The returned vector must be freed.

float ppmObjectGetCylinderRadius(ppmObject obj);

Returns the radius of the cylinder (if the object is a cylinder) before transformation.

float ppmObjectGetCylinderHeight(ppmObject obj);

Returns the height of the cylinder (if the object is a cylinder) before transformation.

vector3d ppmObjectGetCylinderSubdivision(ppmObject obj);

Returns the subdivision on radius and height of the cylinder (if the object is a cylinder) before transformation. The returned vector must be freed.

bool ppmObjectGetCylinderHasBottomCap(ppmObject obj);

Returns true if the cylinder (if the object is a cylinder) has a bottom cap.

bool ppmObjectGetCylinderHasTopCap(ppmObject obj);

Returns true if the cylinder (if the object is a cylinder) has a bottom cap.

float ppmObjectGetConeRadius(ppmObject obj);

Returns the radius of the cone (if the object is a cone) before transformation.

float ppmObjectGetConeHeight(ppmObject obj);

Returns the height of the cone (if the object is a cone) before transformation.

int ppmObjectGetConeSubdivision(ppmObject obj);

Returns the subdivision on radius of the cone (if the object is a cone) before transformation.

bool ppmObjectGetConeHasCap(ppmObject obj);

Returns true if the cone (if the object is a cone) has a bottom cap.

float ppmObjectGetTorusMajorRadius(ppmObject obj);

Returns the major radius of the torus (if the object is a torus) before transformation.

float ppmObjectGetTorusMinorRadius(ppmObject obj);

Returns the minor radius of the torus (if the object is a torus) before transformation.

int ppmObjectGetTorusMajorSubdivision(ppmObject obj);

Returns the subdivision on major radius of the torus (if the object is a torus) before transformation.

int ppmObjectGetTorusMinorSubdivision(ppmObject obj);

Returns the subdivision on minor radius of the torus (if the object is a torus) before transformation.

bool ppmObjectIsSds(ppmObject obj);

Returns true if the object is a subdivision surface object.

bool ppmObjectIsSymetric(ppmObject obj);

Returns true if the object is a symetric object.

int ppmObjectGetSdsLevel(ppmObject obj);

Returns the current SDS level of the object.

int ppmObjectGetSdsRenderingLevel(ppmObject obj);

Returns the rendering SDS level of the object.

string ppmObjectGetName(ppmObject obj);

Returns the object’s name.

void ppmObjectSetName(ppmObject obj, string name);

Sets the object’s name.

point3d ppmObjectGetCenter(ppmObject obj);

Returns the object’s center.

void ppmObjectSetCenter(ppmObject obj, point3d center);

Sets the object’s center.

point3d ppmObjectGetPivot(ppmObject obj);

Returns the object’s pivot.

void ppmObjectSetPivot(ppmObject obj, point3d center);

Sets the object’s pivot.

bool ppmObjectGetVisible(ppmObject obj);

Returns the object’s visibilty.

void ppmObjectSetVisible(ppmObject obj, bool visible);

Sets the object’s visibilty.

bool ppmObjectGetRenderable(ppmObject obj);

Returns ’true’ if the object should be rendered, ’false’ otherwise.

void ppmObjectSetRenderable(ppmObject obj, bool renderable);

’true’ if the object should be rendered, ’false’ otherwise.

bool ppmObjectGetLocked(ppmObject obj);

Returns ’true’ if the object is locked, ’false’ otherwise.

void ppmObjectSetLocked(ppmObject obj, bool locked);

’true’ if the object is locked, ’false’ otherwise.

point3dIterator ppmObjectGetVertices(ppmObject obj);

Returns the vertices of the object.

vector3dIterator ppmObjectGetNormals(ppmObject obj);

Returns the normals of the object.

ppmEdgeIterator ppmObjectGetEdges(ppmObject obj);

Returns the edges of the object.

ppmFaceIterator ppmObjectGetFaces(ppmObject obj);

Returns the faces of the object.

ppmTextureIterator ppmObjectGetTextures(ppmObject obj);

Returns the textures of the object.

int ppmObjectAddTexture(ppmObject obj, ppgRaster raster, string textureName);

Adds a new texture to the object. Returns the texture index, -1 if failed.

bool ppmTexturedMaterialIsAppliedToFace(ppmTexturedMaterial txtMat, ppmObject obj, ppmFace face);

Returns true if the face material and textures are the one of the textured material.

ppmBone ppmObjectGetRootBone(ppmObject obj);

Returns the root bone of the object.

int ppmObjectGetNumBones(ppmObject obj);

Returns the number of bones of the object.

int ppmObjectGetVertexMaxBones(ppmObject obj);

Returns the maximum number of bones link to a vertex of the object.

int ppmObjectGetFaceMaxBones(ppmObject obj);

Returns the maximum number of bones link to a face of the object.

ppmSequenceIterator ppmObjectGetSequences(ppmObject obj);

Returns the sequences of the object.

void ppmObjectComputeFaceNormals(ppmObject obj);

Computes the faces' normals so that the object is displayed properly.

void ppmObjectSplitFaceMiddle(ppmObject obj, int faceNum);

Adds a vertex into the middle of the specified face, then creates triangles between each edge of the face and the center vertex. The new faces are added after the current faces of the object.

void ppmObjectSmallFaceMiddle(ppmObject obj, int faceNum, float percent);

Adds a small version of the specified face into this face.

void ppmObjectRemoveFace(ppmObject obj, int faceNum);

Removes the specified face.

void ppmObjectTriangulate(ppmObject obj);

Triangulates all the faces of the object obj.

void ppmObjectSubdivide(ppmObject obj);

Subdivides all the faces of the object obj.

ppmObject ppmObjectToMesh(ppmObject obj, int sdsLevel);

Returns an object ready to be exported (symmetry applied, sds performed, concave faces splitted in convex faces).

ppmAssociativeArray ppmObjectGetValues(ppmObject obj);

Returns the object's associative array.

intIterator ppmObjectGetConnectedVertices(ppmObject obj, int vertexNum);

Returns the the indexes of all vertices connected to the specified vertex.

Cameras

typedef ppmCamera;

Defines Cameras

ppmCamera ppmCameraNew(void);

Creates a new camera and add it to the current scene.

string ppmCameraGetName(ppmCamera camera);

Returns the camera's name.

void ppmCameraSetName(ppmCamera camera, string name);

Sets the camera's name.

point3d ppmCameraGetPos(ppmCamera camera);

Returns the camera's position.

void ppmCameraSetPos(ppmCamera camera, point3d pos);

Sets the camera’s position.

float ppmCameraGetFov(ppmCamera camera);

Returns the camera’s field of view.

void ppmCameraSetFov(ppmCamera camera, float fov);

Sets the camera’s field of view.

vector3d ppmCameraGetUp(ppmCamera camera);

Returns the camera’s up vector.

void ppmCameraSetUp(ppmCamera camera, vector3d up);

Sets the camera’s up vector.

bool ppmCameraGetVisible(ppmCamera camera);

Returns the camera’s visibilty.

void ppmCameraSetVisible(ppmCamera camera, bool visible);

Sets the camera’s visibilty.

bool ppmCameraGetLocked(ppmCamera camera);

Returns ’true’ if the camera is locked, ’false’ otherwise.

void ppmCameraSetLocked(ppmCamera camera, bool locked);

’true’ if the camera is locked, ’false’ otherwise.

point3d ppmCameraGetPointTo(ppmCamera camera);

Returns the camera’s spot camera target.

vector3d ppmCameraGetDir(ppmCamera camera);

Returns the camera’s direction.

void ppmCameraSetPointTo(ppmCamera camera,point3d target);

Sets the camera’s target (and set the camera to be a spot camera).

ppmObject ppmCameraGetPointedObject(ppmCamera camera);

Returns the camera’s pointed object (null if none, or if it’s not a spot camera).

void ppmCameraSetPointedObj(ppmCamera camera, ppmObject obj);

Sets the camera’s pointed object (null to remove the link). The camera is set to be a spot camera, and to target the object center.

int ppmCameraGetPointedVertexNum(ppmCamera camera);

Returns the camera’s pointed vertex number (-1 if it is the pointed object center, or if it’s not a spot camera).

void ppmCameraSetPointedVertexNum(ppmCamera camera, int vertexNum);

Sets the camera’s pointed vertex number (-1 to the poined object center).

vector3d ppmCameraGetPointedDelta(ppmCamera camera);

Returns the camera’s displacement from the pointed point.

void ppmCameraSetPointedDelta(ppmCamera camera, vector3d delta);

Sets the camera’s displacement to the pointed point.

ppmAssociativeArray ppmCameraGetValues(ppmCamera camera);

Returns the camera’s associative array.

Lights

typedef ppmLight;

Defines Lights

ppmLight ppmLightNew(void);

Creates a new light and add it to the current scene.

string ppmLightGetName(ppmLight light);

Returns the light’s name.

void ppmLightSetName(ppmLight light, string name);

Sets the light’s name.

point3d ppmLightGetPos(ppmLight light);

Returns the light’s position.

void ppmLightSetPos(ppmLight light, point3d pos);

Sets the light’s position.

float ppmLightGetIntensity(ppmLight light);

Returns the light’s intensity.

void ppmLightSetIntensity(ppmLight light, float intensity);

Sets the light’s intensity.

rgbColor ppmLightGetColor(ppmLight light);

Returns the light’s color (rgb).

void ppmLightSetColor(ppmLight light, rgbColor rgb);

Sets the light’s color (rgb).

bool ppmLightGetUseSpecular(ppmLight light);

Returns true if the light uses a specular color
differrent from the diffuse color.

void ppmLightSetUseSpecular(ppmLight light, bool use);

Sets if the light uses a specular color diffirent
of the diffuse color.

rgbColor ppmLightGetSpecularColor(ppmLight light);

Returns the light’s specular color (rgb).

void ppmLightSetSpecularColor(ppmLight light, rgbColor rgb);

Sets the light’s specular color (works only if the
light use a specular color different from the
diffuse color).

bool ppmLightGetCastShadows(ppmLight light);

Returns true if the light casts shadows.

void ppmLightSetCastShadows(ppmLight light, bool use);

Sets if the light casts shadows.

bool ppmLightGetVisible(ppmLight light);

Returns the light’s visibilty.

void ppmLightSetVisible(ppmLight light, bool visible);

Sets the light’s visibilty.

bool ppmLightGetRenderable(ppmLight light);

Returns ’true’ if the light should be used for rendering, ’false’ otherwise.

void ppmLightSetRenderable(ppmLight light, bool renderable);

’true’ if the light should be used for rendering, ’false’ otherwise.

bool ppmLightGetLocked(ppmLight light);

Returns ’true’ if the light is locked, ’false’ otherwise.

void ppmLightSetLocked(ppmLight light, bool locked);

’true’ if the light is locked, ’false’ otherwise.

float ppmLightGetFallOffBegin(ppmLight light);

Returns the light’s full light radius (0 if none defined).

void ppmLightSetFallOffBegin(ppmLight light, float fallOff);

Sets the light’s full light radius.

float ppmLightGetFallOffEnd(ppmLight light);

Returns the light’s radius (no lightning further).

void ppmLightSetFallOffEnd(ppmLight light, float fallOff);

Sets the light’s radius.

bool ppmLightIsSpotLight(ppmLight ligh);

Returns true if the light is a spot light.

point3d ppmLightGetPointTo(ppmLight light);

Returns the light’s spot light target.

vector3d ppmLightGetDir(ppmLight light);

Returns the light’s direction.

void ppmLightSetPointTo(ppmLight light,point3d target);

Sets the light’s target (and set the light to be a spot light).

float ppmLightGetFullLightAngle(ppmLight light);

Returns the light’s full light cone angle in degree.

void ppmLightSetFullLightAngle(ppmLight light, float angle);

Sets the light’s full light cone angle in degree (do nothing if the light is not a spot light).

float ppmLightGetOffLightAngle(ppmLight light);

Returns the light’s off light cone angle in degree.

void ppmLightSetOffLightAngle(ppmLight light, float angle);

Sets the light’s off light cone angle in degree (do nothing if the light is not a spot light).

ppmObject ppmLightGetPointedObject(ppmLight light);

Returns the light’s pointed object (null if none, or if it’s not a spot light).

void ppmLightSetPointedObj(ppmLight light, ppmObject obj);

Sets the light’s pointed object (null to remove the link). The light is set to be a spot light, and to target the object center.

int ppmLightGetPointedVertexNum(ppmLight light);

Returns the light’s pointed vertex number (-1 if it is the pointed object center, or if it’s not a spot light).

void ppmLightSetPointedVertexNum(ppmLight light, int vertexNum);

Sets the light’s pointed vertex number (-1 to the poined object center).

vector3d ppmLightGetPointedDelta(ppmLight light);

Returns the light’s displacement from the pointed point.

void ppmLightSetPointedDelta(ppmLight light, vector3d delta);

Sets the light’s displacement to the pointed point.

ppmAssociativeArray ppmLightGetValues(ppmLight light);

Returns the light’s associative array.

Scene

ppmObjectIterator ppmGetObjects(void);

Returns the not null objects of the current scene.

ppmObjectIterator ppmGetNullObjects(void);

Returns the null objects of the current scene.

ppmLightIterator ppmGetLights(void);

Returns the lights of the current scene.

ppmCameraIterator ppmGetCameras(void);

Returns the cameras of the current scene.

ppmCameraIterator ppmGetViewCameras(void);

Returns the cameras associated with each currently diplayed views.

ppmMaterialIterator ppmGetMaterials(void);

Returns the materials of the current scene.

ppmTextureIterator ppmGetTextures(void);

Returns the textures of the current scene.

typedef ppmSelectionType enum = { ppmSelObject, ppmSelFace, ppmSelEdge, ppmSelVertex, ppmSelLight, ppmSelCamera, ppmSelUndefined };

Defines the possible selection types.

ppmSelectionType ppmGetSelectionType(void);

Returns the type of selection in the current scene.

ppmObjectDArray ppmGetSelectedObjects(void);

Returns the selected objects of the current scene.
If the selection type is ppmSelLight, ppmSelCamera
or ppmSelUndefined, the returned dynamic array is
empty. If the selection type is ppmSelFace,
ppmSelEdge or ppmSelVertex, the returned iterator
contains only one element, the last selected
object. If the selection type is ppmSelObject, the
returned dynamic array contains the selected
objects.

intDArray ppmGetSelectedFaces(void);

Returns the selected faces' indexes. If the
selection type is not ppmSelObject then the
returned dynamic array contains all the faces'
indexes of the last selected object. If the
selection type is ppmSelFace, the returned dynamic
array contains the selected faces' indexes.
Otherwise, the returned dynamic array is empty.

intDArray ppmGetSelectedEdges(void);

Returns the selected edges' indexes. If the
selection type is not ppmSelObject then the
returned dynamic array contains all the edges'
indexes of the last selected object. If the
selection type is ppmSelFace, the returned dynamic
array contains all the edges' indexes of the
selected faces. If the selection type is
ppmSelEdge, the returned dynamic array contains the
selected edges' indexes. Otherwise, the returned
dynamic array is empty.

intDArray ppmGetSelectedVertices(void);

Returns the selected vertices' indexes. If the
selection type is not ppmSelObject then the
returned dynamic array contains all the vertices'
indexes of the last selected object. If the
selection type is ppmSelFace, the returned dynamic
array contains all the vertices' indexes of the
selected faces. If the selection type is
ppmSelEdge, the returned dynamic array contains all

the vertices’ indexes of the selected edges. If the selection type is not ppmSelVertex then the returned dynamic array contains the selected vertices’ indexes. Otherwise, the returned dynamic array is empty.

ppmCameraDArray ppmGetSelectedCamera(void);

Returns the selected camera. If the selection type is not ppmSelCamera then a null object is returned.

ppmLightDArray ppmGetSelectedLight(void);

Returns the selected lights. If the selection type is not ppmSelLight then a null object is returned.

ppmMaterialDArray ppmGetSelectedMaterials(void);

Returns the selected materials. If the material window is not opened then a null object is returned.

ppmTexturedMaterialIterator ppmGetTexturedMaterials(void);

Returns the all the textured materialof the current scene.

ppmAssociativeArray ppmGetValues(void);

Returns the scene’s associative array.

Height Field

typedef ppmHeightField;

Defines a Height Field

int ppmHeightFieldGetWidth(ppmHeightField hf);

Returns the height field width.

int ppmHeightFieldGetHeight(ppmHeightField hf);

Returns the height field height.

float ppmHeightFieldGetMinAltitude(ppmHeightField hf);

Returns the height field minimum altitude.

float ppmHeightFieldGetMaxAltitude(ppmHeightField hf);

Returns the height field maximum altitude.

float ppmHeightFieldGetAltitude(ppmHeightField hf, int x, int y);

Returns the height field altitude at pos (x, y).

void ppmHeightFieldSetAltitude(ppmHeightField hf, int x, int y, float altitude);

Sets the height field altitude at pos (x, y).

void ppmHeightFieldUpdateExtent(ppmHeightField hf);

Update the min and max altitudes based on height field values. Must be call after changing the height field values.

void ppmHeightFieldStackRun(string commands);

Run a Height field list of commands. The possible commands are described in Albatross 3D manual.

int ppmHeightFieldStackNumHeightField(void);

Returns the numer of height fields in the height field stack.

ppmHeightField ppmHeightFieldStackGetHeightField(int hfNum);

Returns the nth height field from the height field stack. Returns null if hfNum is inavalid.

void ppmHeightFieldStackDropHeightField(int hfNum);

Drops the nth height field from the height field stack. Does nothing if hfNum is inavalid.

float ppmHeightFieldEvaluateParam(string param);

Returns the float value corresponding to the evaluation of the equation in param. The possible variables are:

w: the width h: the height a: the maximum height i: the minimum height

Console

void print(string str);

Prints str in the console.

void printInt(string str, int i);

Prints str in the console. ’%d’ marks the place where the integer will be printed into the string *str*.

void printFloat(string str, float f);

Prints str in the console. ’%g’ or ’%f’ marks the place where the float *f* will be printed into the string *str*.

void printRgbColor(string str, rgbColor rgb);

Prints str in the console. ’#%06x’ marks the place where the color *rgb* will be printed into the string *str*.

void printString(string str, string s);

Prints str in the console.’%s’ marks the place where the color *rgb* will be printed into the string *str*.

void printVector3d(string str, vector3d v);

Prints str in the console. The vector values are printed after the string *str*.

void printPoint3d(string str, point3d p);

Prints str in the console. The point values are printed after the string *str*

void printPallette(string str, ppmPalette pal);

Prints str in the console. The palette colors and positions are printed after the string *str*.

void printMonoPalette(string str, ppmMonoPalette mPal);

Prints str in the console. The palette levels and position are printed after the string *str*.

Albatross 3D Script parameters

typedef ppmParamType enum = { ppmParamBoolType, ppmParamIntType, ppmParamFloatType, ppmParamStringType,

ppmParamChoiceType, ppmParamComboType, ppmParamColorType, ppmParamPaletteType, ppmParamMonoPaletteType, ppmParamVectorType, ppmParamFileType, ppmParamImageType};

Defines the possible parameter types.

typedef ppmParam struct { ppmParamType type; string name; string selectOptions; bool boolValue; int intValue; float floatValue; string stringValue; int selectedValue; rgbColor colorValue; ppmPalette paletteObject; ppmMonoPalette monoPaletteObject; vector3d vectorObject; string fileFilter; ppgFileDlgType fileOpenType; };

Defines the parameter structure. the type field defines the parameter type. According to the type, one of the xxValue fields stores the value. This value can be use to process the selected objects.

int ppmOpenParamsWindow(string title, string label, ppmParam &params, int numParams);

Opens a window with the defined parameters. Returns true if the user press ’Ok’, false if the user press ’Cancel’

External Program

int ppmRunCommand(string command);

Returns the execution returned value.