Writing scripts in 3D-Coat

Scripts in 3D-Coat are based on Angelscript that is very close to C++ syntax. Using scripts you may do virtually everything that is possible within UI. Scripts are intended to perform batch actions, make own macro actions, create interactive tutorials. At first you may look angelscript documentation to see general features of the language and difference with C++:

http:angelcode.com/angelscript/sdk/docs/manual/doc_script.html
http:angelcode.com/angelscript/sdk/docs/manual/doc_understanding_as.html

Arrays:

http:angelcode.com/angelscript/sdk/docs/manual/doc_datatypes_arrays.html

Strings:

http:angelcode.com/angelscript/sdk/docs/manual/doc_datatypes_strings.html

It is easy to start writing scripts in 3D-Coat. Just click **Scripts->Create your script** and select template that is closest to what you want to do. File will be opened in text editor, after edit you may save it and execute via **Scripts->Run script** or from recent list in **Scripts** menu. Read carefully comments in a file that will be opened in text editor and follow the recommendations.

Simple examples

This is example of a simple script to demonstate how to handle modal message boxes.

```
//declare any global variables if need, don't write commands ritght there, do it in main() of any other function.
//...
//This function will be called once when script will be run
//insert any commands there. Start from cmd as described above.
int n=0;
void main(){
	// next command is optional, just to keep control of execution in modal message boxes (see
description	//of ModalDialogCallback below), if you are new you may delete or ignore next line
	SetModalDialogCallback("ModalDialogCallback");
	// write your script right there
	ModalDialog("This message will disappear soon"," ");
}
//Next  function is mostly for advanced users. When you are calling or getting some modal message box you
are loosing control until user will press some button there.
//But each time when modal dialog will be show this function will be called and you may do some action inside
modal message box
void ModalDialogCallback(){
	// Example of command - press first button in message box (uncomment next line if need)
	if(n++>200)cmd("$DialogButton#1");/*press OK*/
}
```

Second example - processing all files in some folder. After running the script description dialog will be shown, then user will be asked to choose files. At the end the number of processed files will be shown.

```
//declare any global variables if need, don't write commands ritght there, do it in main() of any other function.
//...
int nfiles;
//This function will be called once when script will be run
//insert any commands there
void main(){
	nfiles=0;
	//shows message box if need
	if(ModalDialogOkCancel("Just write description there"," ")){
		//This function will call void ForAnyFile(string &in FileName) for each file in user defined folder - jpg and png are just for example
		ForEachFileInFolder("","*.png;*.jpg","ForAnyFile");
		ModalDialog(formatInt(nfiles,'l')+" files processed."," ");
	}
}
//This function will be called for each file in user-defined folder
void ForAnyFile(string &in FileName){
	//print there just for example, use Scripts->View execution log
	print(FileName + "\n");
	nfiles++;
}
```

A bit more advanced example - select files in folder, import, voxelize, smooth, export back with decimation:

```
//this is callback for dialog that asks if user needs to save scene after New command
void ModalDialogCallbackDontSave(){
	cmd("$DialogButton#2");/*press Don't save - second button*/
}
//this is callback to press OK when user is asked to keep scale or not
void ModalDialogCallbackOk(){
	cmd("$DialogButton#1");/*press OK*/

}
//this is callback for decimation dialog
void ModalDialogCallbackDecimation(){
	SetSliderValue("$DecimationParams::ReductionPercent",80.0);
	cmd("$DialogButton#1");/*press Ok*/
}
int nfiles;
void main(){
	nfiles=0;
	//This function will call void ForAnyFile(string &in FileName) for each file in user defined folder - stl and obj are just for example
	ForEachFileInFolder("","*.stl;*.obj","ForAnyFile");
	ModalDialog(formatInt(nfiles,'l')+" files processed."," ");
}

void ForAnyFile(string &in FileName){
	nfiles++;
	//During the New command 3D-Coat will ask what to do with the scene - save or not, this dialog intended to skip it
	SetModalDialogCallback("ModalDialogCallbackDontSave");
	cmd("$CLEARSCENE");/*New*/
	//Substitute filename for the next file dialog
	SetFileForFileDialogs(FileName);
```

```
        //Import model
        cmd("$ImportForVoxelizing");/*Import mesh for voxelizing*/
        //This is intended to press OK when user will be asked if scale should be kept
        SetModalDialogCallback("ModalDialogCallbackOk");
        //ensure that model will be voxelized
        SetBoolField("$VoxelSculptTool::MergeWithoutVoxelizing",false);
        //press apply button
        apply();
        //smooth voxelized mesh
        cmd("$[Page4]Smooth all");/*Smooth all*/
        string filename;
        filename=RemoveExtension(FileName);
        filename+="_smoothed.stl";
        SetFileForFileDialogs(filename);
        //set output filename
        SetModalDialogCallback("ModalDialogCallbackDecimation");
        //and finally export
        cmd("$ExportScene");/*Export scene*/
}
```

Next example – trivial tutorial example. It just asks you to click File->New and checks if you done it. Pay attention to Step function in cycle – it performs one step of rendering cycle in 3D-Coat.

```
void main(){
        ModalDialog("Just some description there. Click OK to start."," ");
        do{
                Step(1); //Rendering cycle
                ShowFloatingMessage("Please click File->New",1); //Show non-modal message
                if(WasRecentlyPressed("$CLEARSCENE",15)){ //Check if New was pressed in last 15 sec
                        ModalDialog("Right, thanks!"," ");
                        return;
                }
        }while(GetTimeSinceStart()<40);
        ModalDialog("Excuse, I can't wait anymore. Bye!"," ");
}
```

Script functions specific to 3D-Coat
Most important functions

bool cmd(string &in ID);

This is main command that you will use. It performs any action from user interface. To get the command ID hover your mouse cursor over required item and press MMB+RMB simultaneously. The command ID will be copied to clipboard. Pay attention that if a command is not present in current UV layout it will not be performed.

void Step(int n);

do n steps in 3D-Coat rendering process. Important if after performing the command you need UI refresh.

void InstallToMenu(string& path,string& ItemName)

Creates permanent new menu item that will run this script. Path is just path to menu like File.Export or Voxels or Retopo. Script will be copied to Scripts/ExtraMenuItems/

Example:

InstallToMenu("File","Do some stuff");

Dialog related functions

void ModalDialog(string &in ID,string &in Caption);

Show model dialog with text identified ID. The ID is used to take translation from language .xml or just may be shown directly if translation is not found.

---

bool ModalDialogOkCancel(string &in ID,string &in Caption);

Show dialog with text identified ID and two buttons - Ok and Cancel. Returns true if OK pressed, false if Cancel pressed. The ID is used to take translation from language .xml or just may be shown directly if translation is not found.

---

bool ModalDialogYesNo(string &in ID,string &in Caption);

Show dialog with text identified ID and two buttons - Yes and No. Returns true if OK pressed, false if Cancel pressed. ID used to take translation from language xml or just may be shown directly if translation not found.

Important! All dialogs may expose list of parameters and you are able to change value of any local or global variable via dialogs. This is set of functions that allows to control additional parameters:

---

void AddTranslation(string& ID, string& Text);

Each function that adds control passes variable name or other ID. Of course names of variables are are not always obvious to end-user. So you may translate in on normal language and make correspondence between ID (name of variable or any other ID in UI) and displayed text.

---

void AddFloatSlider(string &in VariableName,float Min,float Max);
Add slider for floating variable. Most of functions have VariableName parameter. It is the name of variable (local or global) you want to edit in the dialog. Look example a bit later. Min, Max – range if the value. Variable shiuld be declared as float VariableName; in global or local scope.

---

void AddIntSlider(string &in VariableName,int Min,int Max);

Add slider for integer vaiable. Variable shiuld be declared as int VariableName; in global or local scope.

---

void AddFloatInput(string &in VariableName,bool EmptyName);

Add input box for floating variable.

---

void AddIntInput(string &in VariableName,bool EmptyName);

Add input box for integer variable.

---

void AddStringInput(string &in VariableName,bool EmptyName);

Add input box for string variable.

---

void AddTextField(string &in TextID,int Align);//1-center,0-left

Insert the text among the list of variables.

---

void AddDelimiter();

Insert delimiter

---

void AddButton(string &in FuncName);

Insert button that will call function

---

void Columns(int nc);

Place next controls in nc columns
Example:

Columns(2);
AddButton("Function1");
AddButton("Function2");

In this case 2 buttons will be aligned horizontally in the dialog.

---

void AddCheckBox(string &in BoolVarRef);

Add checkbox. BoolVarRef shoud refer existing boolean variable declared as bool VariableName;

---

void AddDroplist(string &in IntVarRef,string &in CaseList);

Add droplist with several cases. Case index will be stored in IntVarRef variable. CaseList is list of possible values for droplist, delimiters are ,;|
Example

Int Case=0;
…
AddDroplist("Case","Case1,Case2,Case3");

---

void UICondition(string& function);

void StopUICondition();

That functions allow to control appearance of UI elements. Example explains all:

```
bool ShowSlider1and2;
float Slider1;
float Slider2;
float Slider3;
bool CheckUI(){
        return ShowSlider1and2;
}
void main(){
        AddCheckBox("ShowSlider1and2");
```

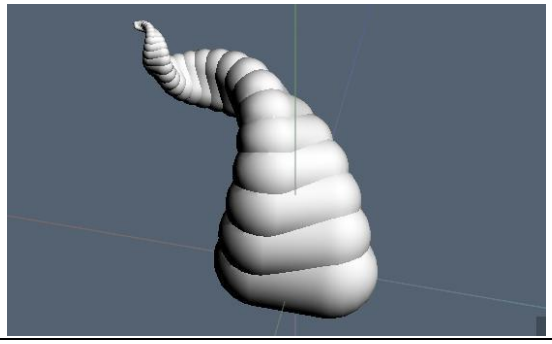
```
        UICondition("CheckUI");//function CheckUI should return true if elements below should be visible
        AddFloatSlider("Slider1",0,123);
        AddFloatSlider("Slider2",0,123);
        StopUICondition();//This function ends scope of previous UICondition
        AddFloatSlider("Slider3",0,123);//This slider is always visible
        ModalDialogOkCancel("","");
}
```

General example of usage:

```
//Making horn
float Angle=10;
int NumberOfChunks=20;

void main(){
        AddTranslation("NumberOfChunks"," Number of chunks in horn");
        AddFloatSlider("Angle",0,90);
        AddIntSlider("NumberOfChunks",1,40);
        if(ModalDialogOkCancel("Please enter horn parameters","Making horn")){
                ResetPrimTransform();
                PrimDensity(0.3);
                for(int i=0;i<NumberOfChunks;i++){
                        capsule(-15,0,0,15,0,0,20,20,0);
                        PrimRotateY(0,0,0,Angle);
                        PrimRotateX(0,0,0,Angle);
                        PrimTranslate(0,15,0);
                        PrimScaleAt(10,20,30,0.9,0.9,0.9);
                        ProgressBar("Please wait",(i*100)/NumberOfChunks);
                }
        }
}
```

Result:



void PressInNextModalDialogs(int ButtonIndex);

This function should be called before you call any of modal dialogs function if you want to press button number ButtonIndex (first button is 1, second is 2).

For example, if you have *Yes* and *No* buttons and you call PressInNextDialogs(1) before showing dialog then *Yes* will be pressed automatically.

You should call PressInNextDialogs(-1) to stop automatical pressing of buttons.

void SetModalDialogCallback(string &in name);

When you call any modal dialog the execution of the script will be stopped until user press Ok or other button in the dialog.

Thus you are loosing control over dialog execution. If you want to do some action in the dialog, change some field you need to setup routine.

that will be called constantly when dialog will be active. The routine may change fields in dialog, press buttons, and do other things.

example:

```
int idx=0;
void DialogCallback(){
if(idx++>100)cmd("$DialogButton#1");/*press OK*/
}
void main(){
SetModalDialogCallback("DialogCallback");
ModalDialog("Hello!");
}
```

You may get the name of a current dialog name using next function.

---

void RemoveModalDialogCallbacks();

remove all dialog callbacks

---

void ShowFloatingMessage(string &in ID,float Time);

show non-modal message on screen that will be hidden after some period of time will pass (Time, sec).

---

bool GetCurrentDialog(string &out ID,string &out Caption);

returns true if you are in dialog now, also it retuns text and caption in a current dialog to identify it

---

int GetLastButtonIndex();

get index of button pressed in last dialog. Usually OK=1, Cancel=2

File dialogs

bool *OpenDialog(string &in extensions,string &out result);*

show open file dialog. List extensions like in example - *.tga;*.jpg;

The resulting file name will be placed in result

function returns true if file is successfully chosen

---

bool SaveDialog(string &in extensions,string &out result);

show save file dialog. List extensions like in example - *.tga;*.jpg;

The resulting file name will be placed in result

function returns true if file is successfully chosen

---

bool FolderDialog(string &out result);

show selecting folder dialog. The resulting file name will be placed in result

function returns true if file is successfully chosen

---

void SetFileForFileDialogs(string &in name);

some functions in 3D-Coat may engage file dialog that will wait user's input. You may skip that dialogs and provide file name automatically so that file dialog will return specified file name without waiting for user's input.

Use empty name "" to stop automatic file dialogs skipping othervice all next file dialogs will be skipped and it may lead to loosing data.

---

bool FileDialogCancelPressed();

returns true if Cancel pressed in the last file dialog

File path management routines

string RemoveFilePath(string &in s);

string RemoveExtension(string &in s);

string EnsureAbsolutePth(string &in s);

string GetFilePath(string &in s);

string GetFileExtension(string &in s);

string GetFileName(string &in s);

bool CheckIfExists(string &in s);

void CreatePath(string &in s);

void ForEachFileInFolder(string &in Folder,string &in ExtList,string &in Callback);

Call Callback for each file in folder. If Folder is empty used will be asked to choose file in folder

Callback should be declared as void Callback(string &in Path);

Ext list contains list of extensions like "*.jpg;*.png"

UI elements getting and setting

bool FieldExists(string &in ID);

Check if element exists in UI. ID has same meaning as in cmd

Example – how to get to know amount of lights in Render room?

```
void main(){
	int n=0;
	do{
		string s="$ExtraLight::Color["+formatInt(n,"l")+"]";
		if(FieldExists(s)){
			n++;
		}else break;
	}while(true);
	string s="Amount of lights = "+formatInt(n,"l");
	ModalDialog(s,"");
}
```

bool GetBoolField(string &in ID);

Get bool field from UI. ID has same meaning as in cmd

bool SetBoolField(string &in ID,bool val);

set value of the boolean field in UI. ID has same meaning as in cmd

returns true if set value successfuly

int GetColorField(string &in ID);

Get color field from UI as integer value. ID has same meaning as in cmd

bool SetColorField(string &in ID,bool val);

set value of the color field in UI. ID has same meaning as in cmd

returns true if set value successfully.

Example of colors – 0xFF0000 – red, 0x00FF00 – green, 0x0000FF - blue

float GetSliderValue(string &in ID);

Get value of slider in UI. ID has same meaning as in cmd

bool SetSliderValue(string &in ID,float val);

Set value of slider in UI. ID has same meaning as in cmd

returns true if set value successfuly

float GetEditBoxValue(string &in ID);

Get value of edit box in UI. ID has same meaning as in cmd

returns 0 if field not found

bool GetEditBoxValue(string &out ID,string &out value);

Get value of edit box in UI. ID has same meaning as in cmd

returns true if field found

---

bool SetEditBoxValue(string &in ID,string &out val);

bool SetEditBoxValue(string &in ID,float val);

Set value of edit box in UI. ID has same meaning as in cmd

returns true if field found

---

void SubstituteInputText(const string & val);

void SubstituteInputText(float val);

Substitute string or value to the next input text dialog. You need this command if there is button that triggers input dialog to enter some text or value. Example - transform tool, ScaleY button. Code to scane object twice along Y axis:

```
SubstituteInputText(200.0);
cmd("$CubPrim::ScaleY");
Step(1);
```

Checking tool and recent commad

bool WasRecentlyPressed(string &in ID,float Time);

Was widget with identifier ID recently (within last Time sec) pressed?

---

bool WasRecentlyRMBPressed(string &in ID,float Time);

Was widget with identifier ID recently (within last Time sec) pressed via RMB?

---

bool IsInTool(string &in ToolID);

Is user in tool identified as ID? To get current tool identifier press RMB+MMB over empty field

---

string GetCurrentToolID ();

Get active tool ID

---

float GetTimeSinceStart();

Get time (sec) since script started

Voxels management

bool IsSurface();

returns true if current volume is in surface mode

bool IsInCache();

Check if volume cached

void ToCache();

move current volume to cache

void FromCache();

restore current volume from cache

string GetCurVolume();

get current volume name

void RenameCurVolume(string &in Name);

rename current volume

void SetCurVolumeMode(bool Surf,bool Silent);

Set surface/voxel mode for volume Surf=true - set surface mode, false - volume. It is same as click on S/V icon. If silent=true then no dialogs will be shown, all will be done by default

bool SetCurVolume(string &in name);

set current volume by name, returns true if succeed

void SelectFirstVolume(bool OnlyVisible);

select first volume in scene, if OnlyVisible==true then first visible volume will be selected

bool SelectNextVolume(bool OnlyVisible);

select next volume after current in tree, if OnlyVisible==true then next visible volume will be selected. Returns false if current volume is last in list.

Example of walking through all volumes:

```
void main(){
        string s=GetCurVolume();//keep current selection
        SelectFirstVolume(true);
        do{
                //insert your action
        }while( SelectNextVolume(true));
         SetCurVolume(s);//restore initial selection
}
```

---

bool CurVolumeIsEmpty();

checks if volume is empty

---

int GetCurVolumePolycount();

Get current volume polycount

---

int GetVoxSceneVisiblePolycount();

Get polycount of whole voxel scene

---

int GetVoxScenePolycount ();

Get polycount of visible volumes

---

string GetCurVolumeShader();

Get current volume shader name

---

void SetVolumeVisibility(bool vis)

Set volume visibility (like with eye icon)

---

bool GetVolumeVisibility()

Returns current volume visibility.

---

void SetVolumeGhosting(bool Ghosting)

Sets Ghost property to volume.

---

bool GetVolumeGhosting()

Get Ghost property from the volume.

---

void SetVolumeOpacity(float Opacity)

Set opacity of the current volume if the shader has corresponding property.

---

void SetVolumeColor(int Color)

Set current volume color if this property of shader is available.

---

void SetShaderProperty(string &in ID,string &in val)

Assign val to current object shader property field.

3D primitives management

You may create composition of 3D primitives in scene using functions below. It allows to create hardsurface scene in non-destructive way. You may define some variables to define paraneters of the scene and get different variations of your scene without complete re-sculpting. There are typical parameters of primitivesthat are self-obvious from the names like x,y,z (coordinates), radius, height etc. Some primitives like cone and tube have only vertical alignment (along Y) by default. To resolve such problems you may use transforms to rotate, spale and translate primitives in space before merging. All transforms are additive and next transform will be applied in addition to all previous. If you need to start from scratch with transform just use ResetPrimTransform(). For example, you need a cone placed at point (10,20,30) and aligned aling X axis.

```
void main(){
    ResetPrimTransform();
    PrimRotateZ(10,20,30,90);
    cone(10,20,30,10,20,0);
}
```

Each primitive has Mode parameter. It allows to perform different boolean operations

0 – add to scene, 1 – subtract from scene, 2 – intersect with scene.

Example of primitive drawing (horn-like figure)

```
void main(){
    ResetPrimTransform();
    PrimDensity(0.3);
    for(int i=0;i<20;i++){
        capsule(-15,0,0,15,0,0,20,20,0);
```

```
            PrimRotateY(0,0,0,10);
            PrimRotateX(0,0,0,10);
            PrimTranslate(0,15,0);
            PrimScaleAt(10,20,30,0.9,0.9,0.9);
            ProgressBar("Please wait",(i*100)/36);
        }
}
```

The list of related commands:

`void ResetPrimTransform();`

Reset additional transformation for low-level primitives.

`void PrimTranslate(float dx,float dy,float dz);`

Translate all further low-level primitives.

`void PrimScaleAt(float x,float y,float z,float scalex,float scaley,float scalez);`

Scale all further low-level primitives using pivot point x,y,z.

`void PrimRotateX(float x,float y,float z,float Angle);`

Rotate further primitives at xyz point around X-axis.

`void PrimRotateY(float x,float y,float z,float Angle);`

Rotate further primitives at xyz point around Y-axis

`void PrimRotateZ(float x,float y,float z,float Angle);`

Rotate further primitives at xyz point around Z-axis

---

void PrimRotateAroundRode(float x,float y,float z,float xend,float yend,float zend,float Angle);

Rotate further primitives around rode (x,y,z)->(xend,yend,zend) on given Angle

---

void PrimStretchBetweenPoints(float x,float y,float z,float xend,float yend,float zend);

Set special transform for further primitives. If you will apply primitive that is placed between points (0,-1,0) and (0,1,0) it will be actually applied as primitive stretched between points (x,y,z) and (xend,yend,zend)

Example:

```
void main(){
    ResetPrimTransform();
    PrimStretchBetweenPoints(10,20,30,40,50,60);//stretch between  (10,20,30) and (40,50,60)
    cylinder(0,-1,0,10,10,2,0);//starts from point (0,-1,0), radius=10, height=2
}
```

This code will create cylinder of radius 10 between points (10,20,30) and (40,50,60)

---

void PrimDensity(float density);

Set additional density factor for low-level primitives. 1 means default density.

---

string GetPrimTransform();

Store current transform for primitives as string to be kept for future usage

---

void SetPrimTransform(string& in M);

Restore current primitives transform from string that was previously kept using GetPrimTransform

---

void sphere(float x,float y,float z,float r,int mode);

Create sphere of radius R

---

void ellipse(float x,float y,float z,float rx,float ry,float rz,int mode);

Create ellipse

---

void cube(float x,float y,float z,float sizex,float sizey,float sizez,int mode);

Create parallelepiped.

---

void cylinder(float x,float y,float z,float topradius,float bottomradius,float height,int mode);

Create cylinder.

---

void cone(float x,float y,float z,float radius,float height,int mode);

Create cone.

---

void ngon(float x,float y,float z,int sides,float topradius,float bottomradius,float height,int mode);

Create N-gon

---

void tube(float x,float y,float z,float topradius,float bottomradius,float height,float wallthickness,int mode);

Create tube

---

void ngontube(float x,float y,float z,int sides,float topradius,float bottomradius,float height,float wallthickness,int mode);

Create n-gonal tube

---

void capsule(float x,float y,float z,float xend,float yend,float zend,float startradius,float endradius,int mode);

Creates capsule between points

2D paint layers management

void AddPaintLayer(string& ID);

Add new paint layer

---

string GetCurrentPaintLayerName();

Get current paint layer name

---

void RenameCurrentPaintLayer(string& Name);

Rename current paint layer

---

bool SelectPaintLayer(string& in ID)

Select paint layer with name ID. Returns false if layer not found.

---

void SelectFirstPaintLayer(bool visible)

Select lowest paint layer. If visible=true, first visible paint layer will be selected

---

bool SelectNextPaintLayer(bool visible)

Select next paint layer. Returns false if current layer is last and no new layer was selected.

---

bool GetPaintLayerVisibility()

Get visibility of the current paint layer

---

void SetPaintLayerVisibility(bool visible)

Set visibility of current paint layer

---

void SetPaintLayerOpacity(float Percents)

Set opacity of the current paint layer (0..100)

---

void SetPaintLayerDepthOpacity(float Percents)

Set opacity of the current paint layer (0..100)

---

void SetPaintLayerBlendingMode(string& in ID)

Set blending mode of the current paint layer. ID-s are same as in English version

---

void SetPaintLayerDepthBlendingMode(string &in ID)

Set depth blending mode of the current paint layer. ID-s are same as in English version

---

Retopo layers management

void AddPaintLayer(string& ID);

Add new Retopo layer

---

string GetCurrentRetopoLayerName();

Get current Retopo layer name

---

void RenameCurrentRetopoLayer(string& Name);

Rename current Retopo layer

---

bool SelectRetopoLayer(string& in ID)

Select Retopo layer with name ID. Returns false if layer not found.

---

void SelectFirstRetopoLayer(bool visible)

Select lowest Retopo layer. If visible=true, first visible Retopo layer will be selected

---

bool SelectNextRetopoLayer(bool visible)

Select next Retopo layer. Returns false if current layer is last and no new layer was selected.

---

bool GetRetopoLayerVisibility()

Get visibility of the current Retopo layer

---

void SetRetopoLayerVisibility(bool visible)

Set visibility of current Retopo layer

Objects/materials/UV-sets management

int GetMaterialsCount()

Get amount of materials in scene.

---

string GetMaterialName(int Index)

Get name of the material. Index is in range 0..GetMaterialsCount()-1

---

void RenameMaterial(int Index, string& Name);

Rename material

---

void DeleteMaterial(int Index)

Delete material

---

void LockMaterial(int Index,bool Locked)

Lock/unlock material

---

void SetMaterialVisibility(int Index,bool Visibility)

Set material visibility.

---

int AddMaterial(string& Name);

int GetMaterialIndex(string& Name);

int GetObjectsCount()

Get objects amount in scene.

---

string GetObjectName(int Index)

Get object name. Index is in range 0.. GetObjectsCount()-1

---

void RenameObject(int Index, string& Name);

Rename object

---

void DeleteObject(int Index)

Delete object in scene

---

void LockObject(int Index,bool Locked)

Lock/unlock object

---

void SetObjectVisibility(int Index,bool Visibility)

Set object visibility

---

int GetUVSetsCount();

Returns amount of UV sets. Pay attention that UV-set – related functions are room dependent. In retopo mesh it operates with retopo mesh, in other rooms – with paint mesh.

string GetUVSetName(int Index);

Returns name of the UV -set

void RenameUVSet(int Index,const cstring& Name);

Rename UV-set

void SelectAllFacesInCurrentUVSet ();

Select all faces in current UV set in retopo or uv room.

void SelectAllFacesInCurrentUVSetAndGroup ();

Select all faces in current UV set and in current retopo group in room.

void SelectAllVisibleFaces ();

Select all visible faces in retopo or uv room.

Miscellaneous

string getCommandLine();

returns whole command line.

void SetGlobalVar(string& Name,string& Value)

Stores some string as global value that may be read later in the session. The value will be stored in 3B file and you will be able to read in further work with this scene.

string GetGlobalVar (string& Name)

Returns value previously stored using SetGlobalVar

string GetSceneFileName()

returns scene filename (last saved or opened as 3B file)

void SetSceneFileName(string& Name)

Sets scene filename for further saving.

void HighlightUIElement(string &in ID,float time);

highlight element with red rectangle

void back(int steps=1);

Goes to one of previous dialogs in call stack

---

void open(string &in Path);

Opens window described by xml-file pointed by Path. If Path contains .3b file will be opened as 3B file.

---

void ppp(string &in path);

Opens model for PPP, if path is empty, shows open dialog

---

void mv(string &in path);

Opens model for MV painting, if path is empty, shows open dialog

---

void ptex(string &in path);

Opens model for Ptex, if path is empty, shows open dialog

---

void imagemesh();

Import image as mesh, dialog will be shown

---

void refmesh(string &in path);

Import mesh as reference, if path is empty dialog will be shown

---

void vertexpaint(string &in path);

Import mesh for vertex painting, if path is empty dialog will be shown

---

void autopo(string &in path);

Perform autopo over the mesh chosen in dialog

---

void repair(string &in id);

Opens mesh for repairing. If id contains "vox" then model will be voxelized, if there is substring "shell" then mesh will be imported as thin shell. Mesh Opening dialog will be shown.

---

void bass();

Activate bas-relief tool

---

void undercut();

Activale remove undercuts mode

---

void activate(string &in id);

Activate special voxel tool. id may be found in English.xml between <ID>...</ID> if you will find name of tool between <Text>...</Text> tags

---

void retopo();

Activate retopo tool

---

void retopopen();

Open mesh using dialog and merge as retopo mesh

---

void ToRoom(string &in name);

Activate any room - name is one of "Paint", "Tweak", "UV", "Retopo", "Render"

---

bool IsInRoom(string &in name);

check if you are in specified room - name is one of "Paint", "Tweak", "UV", "Retopo", "Render"

---

void AddNewVolume(string &in name);

add new volume in voxel room. If name is empty name will be assigned automatically.

---

void uv();

Activate UV room

---

void vox();

Activate voxel room and add new volume

---

void sphere(float x,float y,float z,float r,int mode);

Create sphere of radius R in voxel room in current object

---

void cube(float x,float y,float z,float sizex,float sizey,float sizez,int mode);

Create cube in voxel room in current object

mode==0 - add, 1 - subtract, 2 - intersect

---

void surf();

Turn all volumes to surface mode

---

void cursurf();

Turn current volume to the surface mode

---

void voxelize();

Turn current volume to voxel mode, voxelize if need

---

void mergeopt(string &in opt);

Sets merging options in voxel room. opt is just set of subbstring s with different options. Possible values are:

[voxelize=true]

[voxelize=false]

[separate=true]

[separate=false]

[respectneg=true]

[respectneg=false]

[as_skin=true]

[as_skin=false]

[skin=....] - to set skin thickness

example: mergeopt("[voxelize=true][as_skin=true][skin=4.5]");

---

void merge(string &in model);

Merge model in voxel room. Empty string means that dialog will be shown.

---

void prim(string &in id);

Activate voxel primitives tool. Possible primitives:

cube, cylinder, sphere, tube, cone, ellipse, n-gon, gear

---

void apply();

Apply in current tool (same as press enter)

---

void ApplyAndKeepScale();

Apply in Merge tool without asking "Keep scale?". Scale will not be kept and scene scale will not be changed

---

void mapply();

Apply in current tool (same as press enter) wint one difference - in Merge tool scale of merged object sill be automatically kept and scene scale changed if this merge is first.

---

void recent3b();

open recent 3B file

---

void Log(string &in line);

print text to MyDocuments/3D-CoatV4/log.txt

---

int rand(int min,int max);

generate integer random number min..max

---

float randF(float min,float max);

generate floating random number min..max

---

void seed(int val);

set random generator seed

---

void ProgressBar(const string& message,int pos);

show progress bar pos = 0..100

---

void SetOrthoMode(bool value);

Set orthogonal (true) or perspective (false) view mode

---

void Log(string &in line)

Ping text to MyDocuments/3D-CoatV4/log.txt

Stroke control functions

With that functions set you may do artificial strokes in any tool.

float GetMouseX()

float GetMouseY()

Get current mouse coordinates.

---

float GetPressure()

Get pen pressure.

---

bool LMBPressed()

Check if LMB pressed.

---

bool RMBPressed()

Check if RMB pressed.

---

bool MMBPressed();

Check if MMB pressed.

---

float GetVisiblePenRadius();

Get visible (in screen projection) radius of the brush.

---

void StartStroke(float x,float y,float Pressure)

Start drawing stroke at point (x,y) with some pressure.

---

void DrawStrokeTo(float x,float y,float Pressure)

Draw stroke to point (x,y)

---

void EndStroke()

After that command stroke will be actually drawn. Set of commands StartStroke/ DrawStrokeTo should be terminated by EndStroke

---

bool ScreenRayPicksObject(float x,float y)

The function check if object is under the screen point (x,y)

Example: the script will switch to retopo/strokes and draw 3 closed circles. It is better to assign hotkey for script execution for convenience of checking.

```
void main(){
	ToRoom("Retopo");
	cmd("$[Page2]Strokes");/*Strokes*/
	float x=GetMouseX();
	float y=GetMouseY();
	float r=100;
	for(int p=0;p<3;p++){
		for(int i=0;i<=32;i++){
			float a=i*3.1415*2.0/32.0;
			float dx=x+r*cos(a);
			float dy=y+r*sin(a);
			if(i==0)StartStroke(dx,dy,1);
			else DrawStrokeTo(dx,dy,1);
		}
		EndStroke();
		r+=30;
	}
}
```