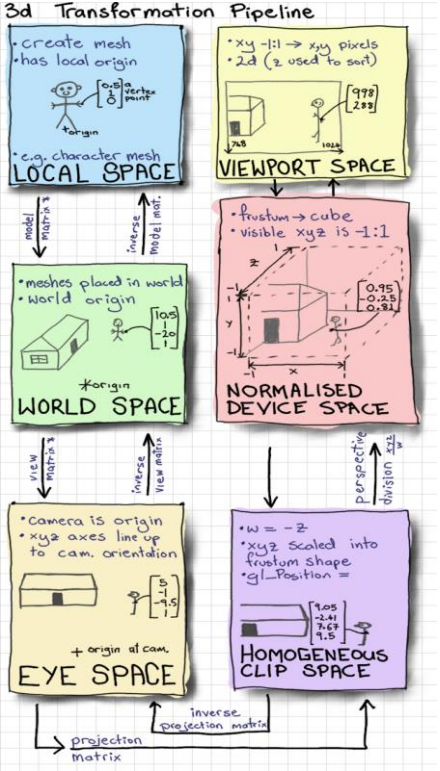
Introduction to Tools Scripting

C++ SDK

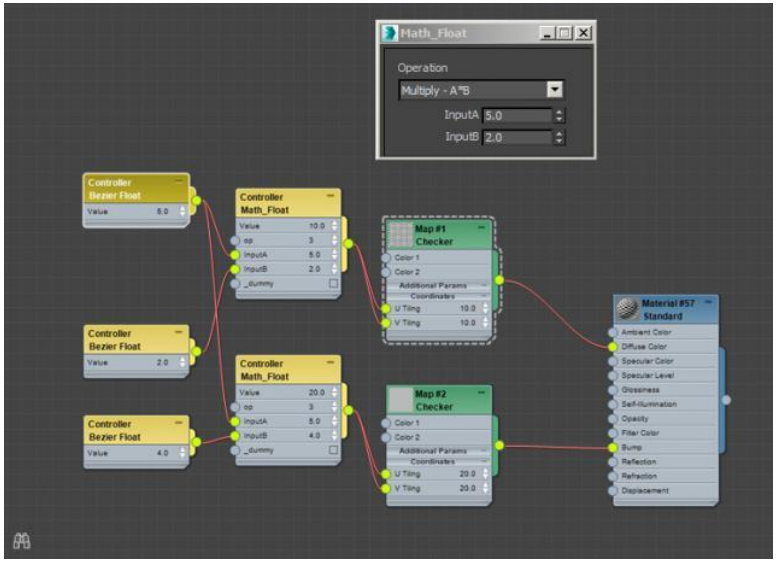
Samples

- Mouse ray picking explained
  - Viewport coordinate to homogeneous
  - Homogeneous to eye coordinate
  - Eye coordinate to world
  - World, simply cast ray using camera front and it’s origin point



Controllers

- Works like blueprints (e.g Unreal)
- Acts the same way like scripts.
- It’s more visual and widely used by artists nowadays.
- Doesn’t need any kind of programming knowledge.

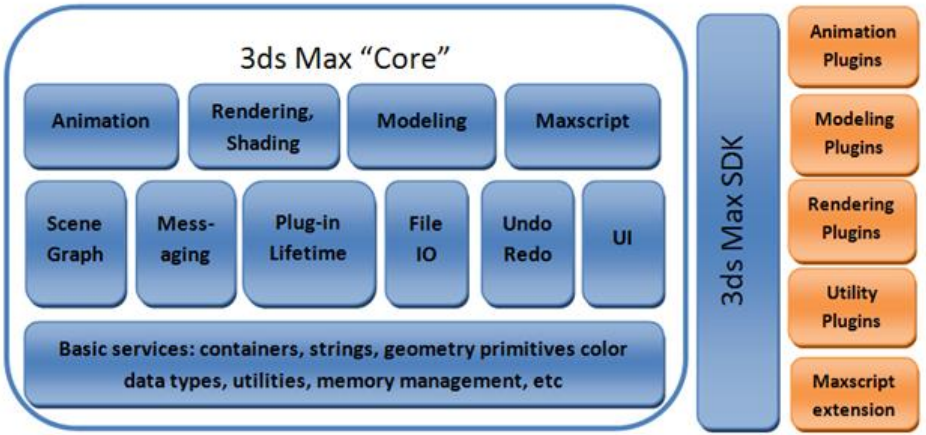


3DS Max SDK

C++

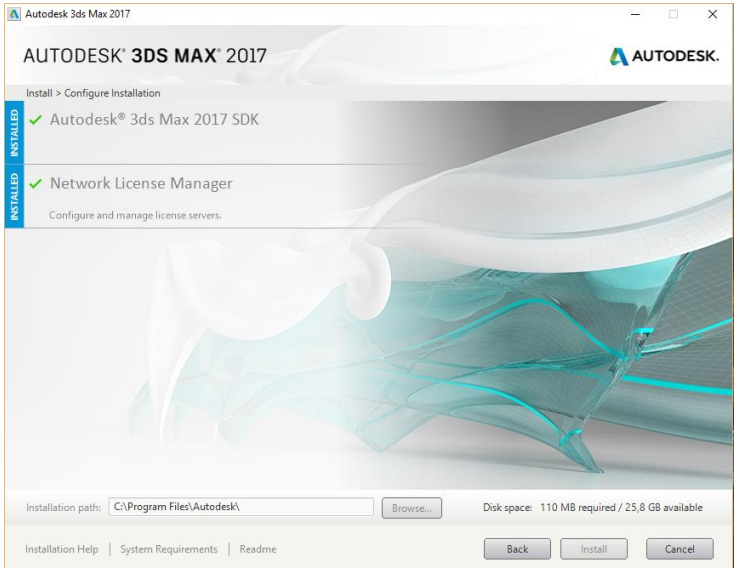
3DSMax Architecture

- Extend functionality by third party developers.
- Different layers below it that allow us to customize the software
    - SDK C++
    - .NET API
    - Python API
    - Maxscript



3DSMax SDK

- Installation
  - Utilities: Under 3DS Max installer
  - Visual studio 2015/2017
  - Visual C++ package
  - Windows SDK 10.1

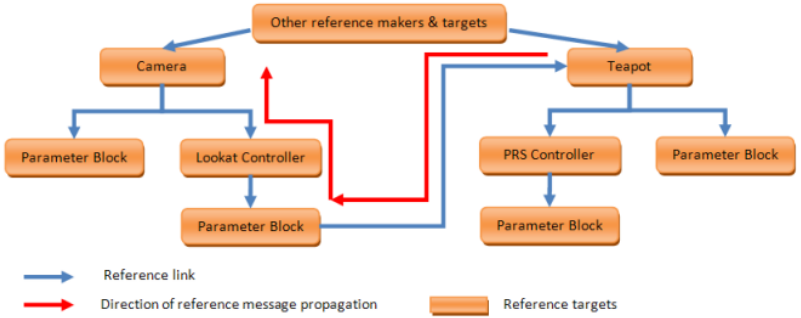


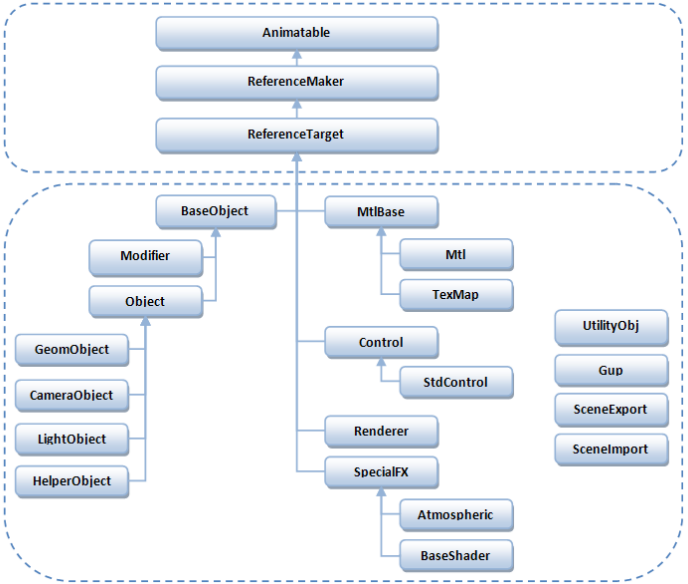
3DSMax SDK

- **List of useful directories:**
  - **assemblies:** This folder contains the 64-bit .NET assembly (.DLL) files.
  - **help:** gencid.exe file, which is a program that generates a random Class ID for your plug-in.
  - **howto:** basic samples containing visual studio project examples.
  - **include:** This folder contains the header files for the core of 3ds Max
  - **projectsettings:** This folder contains the compiler and linker settings in the form of property sheets.
  - **samples:** This folder contains the complete code for some of the 3ds Max standard plug-ins.
  - **plugin:** Where the plug-ins in the \howto folder put their 64-bit binaries.
  - **pdb:** The symbols are generated to this folder.
  - **lib:** This folder contains the library files (.LIB) needed for the 64-bit plug-ins.
  - **tools:** It contains a tool that extracts resources into a .mui file (related to language pack) automatically.

Object	These plug-ins that represent objects the user can create and manipulate .	.dlo
Modifier	These are plug-ins that take as input a procedural object and apply a transformation to it.	.dlm
Animation	Store and interpolate between animation keyframe values of basic data types such as integers and floats.	.dlc
Rendering	These plug-ins can add effects such as glare, shadows, etc to a scene, or render the entire scene.	.dlv
Materials and Texture	These plug-ins allow for defining the look, or appearance of objects when displayed in the viewport and rendered.	.dlt
Scene data	These plug-ins either import or export 3D scene data such as geometry, lights, camera, materials, textures, etc.	.dle
Image File IO	These plug-ins allow for importing and saving images.	.bmi
Utility	These plug-ins usually perform some operation on scene entities, such as display information about the selected objects, or provide other miscellaneous services.	.dlu
MaxScript	These plug-ins customize how the MAXScript engine works or extend it with new functionality, such as C++ implemented functions.	

3DSMax SDK


Other reference makers & targets
Camera
Teapot
Parameter Block
Lookat Controller
PRS Controller
Parameter Block
Parameter Block
Parameter Block
Reference link
Direction of reference message propagation
Reference targets


Animatable
ReferenceMaker
ReferenceTarget
BaseObject
MtlBase
Modifier
Object
Mtl
TexMap
GeomObject
CameraObject
LightObject
HelperObject
Control
StdControl
Renderer
SpecialFX
Atmospheric
BaseShader
UtilityObj
Gup
SceneExport
SceneImport

3DSMax SDK

Examples:

- Layer example demo
- Export to obj demo
- QT demos, mention these demos.
- Mesh creation demo (widget)
- Modifier demo (bending)
- Snap modifier (mention)
- Maxscript publishing
- Maxscript function publishing

Found at samples folder and howto folder under maxSDK directory.

Maxscript break 1

- Monochrome effect

```
plugin RenderEffect MonoChrome
name:"MonoChrome"
classID:#(0x9e6e9e77, 0xbe815df4)
(
    on apply r_image progressCB: do
    (
        bmp_w = r_image.width
        bmp_h = r_image.height
        for y = 0 to bmp_h-1 do
        (
            pixel_line = getPixels r_image [0,y] bmp_w
            for x = 1 to bmp_w do
            (
                p_v = pixel_line[x].value
                pixel_line[x] = color p_v p_v p_v pixel_line[x].alpha
            )--end x loop
            setPixels r_image [0,y] pixel_line
        )--end y loop
    )--end on apply
)--end plugin
```

Maxscript break 2

- Depth voxelize

```
delete $VoxelBox*
rbmp = render outputsize:[128,128] channels:#(#zdepth) vfb:off
z_d = getchannelasmask rbmp #zdepth
progressstart "Rendering Voxels..."

for y = 1 to rbmp.height do
(
    progressupdate (100.0 * y / rbmp.height)
    pixel_line = getpixels rbmp [0,y-1] rbmp.width
    z_line = getpixels z_d [0,y-1] rbmp.width
    for x = 1 to rbmp.width do
    (
        b = box width:10 length:10 height:(z_line[x].value/2)
        b.pos = [x*10,-y*10,0]
        b.wirecolor = pixel_line[x]
        b.name = uniquename "VoxelBox"
    )
)

progressend()
```

3DSMax SDK

- Publishing c++ functions into maxscript

```
#include <maxscript\maxscript.h>
#include <maxscript\macros\define_instantiation_functions.h>
def_visible_primitive(IntervalArray, "IntervalArray");

Value* IntervalArray_cf(Value **arg_list, int count)
{
   return &ok;
}
```

- Publishing maxscript functions into C++

```
ExecuteMAXScriptScript(MCHAR *s, BOOL quietErrors = FALSE, FPValue *fpv = NULL);
```

3DSMax SDK

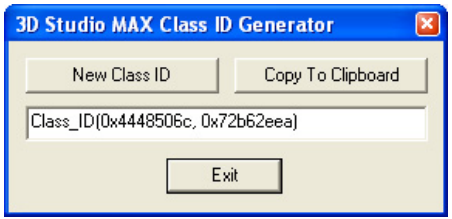
- Project Initialization
  - DllMain: Main entry point of the .dll
  - LibNumberClasses: the number of plugins that the classes in the dll use.
  - LibClassDesc: class pointer returning the class exported by the .dll.
  - LibDescription: string describing the .dll
  - LibVersion: string returns the version of the .dll
  - LibInitialize: initialize the plugin at max initialization.
  - LibShutdown: shut down the plugin on max shutdown.
  - Class descriptors...

- Class Id: identifier for each plugin, must be generated from gencid.exe

3DSMax SDK

- 3DSMAX Plugin wizard: not recommended
- Visual 2015 or 2017: Visual C++ package needed

- Manual way
  - DEF file
  - DLL entry
  - Resource file
  - Include directories (maxsdk/include)
  - Library directories
  - Dynamic .dll, unicode character
  - Output plugin folder
  - Resource header
  - Resource script



https://getcoreinterface.typepad.com/

3DSMax SDK: Exercise

- Steps done
  - Configure the visual studio project
  - Add def, resource and any other necessary file (use how-to samples)
  - Create the dllentry with default parameters
  - Create the main class
  - Add the menu to the main class
  - Add the scene exporter to the main class (create new class)
  - Modify the text of the panel to fit your needs
  - Swap configuration from files if necessary.
  - Extern for class desc.

http://docs.autodesk.com/3DSMAX/16/ENU/3ds-Max-SDK-Programmer-Guide/index.html?url=files/GUID-511D17BA-88AE-42EE-9ADD-AED27495EC83.htm,topicNumber=d30e29251

3DSMax SDK: Plugin reload

- Hot reload of plugins, deferred way.
- Not implemented, contact me if needed!

- https://codelab.wordpress.com/2014/03/25/plugin-reloading-for-3dsmax-exporters/

3DSMax SDK: Exercise

- Build a plugin in C++ to export the current scene.
- Nlohmann json library
- Export the scene only using

```
rollout rolloutExport "Untitled" width:237 height:130
(
    button 'btn1' "Export" pos:[18,75] width:200 height:22 align:#left
    button 'btn2' "..." pos:[200,42] width:19 height:17 align:#left
    editText 'edt1' "" pos:[14,41] width:178 height:18 align:#left

    on btn2 pressed do (

        dir = getSavePath caption:"Select Folder" initialDir:#images
        edt1.text = dir
    )

    on btn1 pressed do (

         ExportScene edt1.text
    )
)
```