

# 3ds Max Game Interface

Alex Zadorozhny  
Software Engineer  
Autodesk Media & Entertainment Division

# What is 3ds Max Game Interface?

3ds Max Game Interface is a set of interfaces that provides a high-level API into the 3ds Max SDK

- Goals
  - Set of interfaces geared towards rapid data extraction
    - Exporters
  - Allow novice 3ds Max developers to extract data with minimal effort
  - Allow for the gradual adoption of 3ds Max Game Interface within existing pipelines
- Resources
  - Devoted R&D resources maintaining and expanding 3ds Max Game Interface

# 3ds Max Export Interface Scope

- Provides unified Scene traversal and Object enumeration
  - Well defined data containers
    - Top-level Nodes and Materials interfaces with list of their children
- Access to the following 3ds Max elements:
  - Geometry, Spline, Lights and Cameras, Helper Objects
  - Modifiers, Skin Deformers, Materials, Textures (included Bitmap)
  - Controllers, Constraints, IK chain, Biped
  - Object Parameters Containers
    - Paramblocks, Custom Attributes, MXS plugins, Custom Node Data
- Handles data conversions
  - Mirrored Geometry (considering negative scale for normals calculation)
  - Coordinate System Conversions (Max, OpenGL, DirectX, custom )

# 3ds Max Game Interface example

- Extracting key frame data from controller (MaxSDK):

```
Control * cont = node->GetTMController()->GetPositionController()
if (!cont)
    return;
IKeyControl *ikc = GetKeyControlInterface(cont);
// TCB point3
if (ikc && (cont->ClassID()== Class_ID(TCBINTERP_POINT3_CLASS_ID, 0))) {
    for (int i=0; i<ikc->GetNumKeys(); i++) {
        ITCBPoint3Key key;
        ikc->GetKey(i, &key);
        // process data
    }
}
```

- Extracting key frame data from controller (Game Interface):

```
IGameKeyTab poskeys;
IGameControl * pGameControl = gameNode->GetIGameControl();
pGameControl->GetTCBKeys(poskeys, IGAME_POS);
// process data
```

# Existing Code Integration

- Ability to fit into existing plugins
  - All the major 3ds Max Game classes provide access to the underlying 3ds Max object

```
INode * node = gameNode->GetMaxNode();  
Object * obj = gameObject->GetMaxObject();  
Mesh * m = gameMesh->GetMaxMesh();
```

- Conversion from INode\* to IGameNode\*:

```
IGameNode *gameNode =  
    GetIGameInterface()->GetIGameNode(pINode);
```

# Mesh Access

- IGameMesh class provides direct access to:
  - Vertex colors, Alpha, Illumination, Material ID, Texture coordinates
  - Normals, Binormals, Tangents
  - Vertex Normals are calculated based on the smoothing group of the face

```
IGameObject * obj = gameNode->GetIGameObject();
if(obj->GetIGameType()==IGameObject::IGAME_MESH)
{
    IGameMesh * gm = (IGameMesh*)obj;
    int numNorms = gm->GetNumberOfNormals();
    for(int i = 0;i<numNorms;i++)
    {
        Point3 n = gm->GetNormal(i);
    }
    int numFaces = gm->GetNumberOfFaces();
    for(int i = 0;i<numFaces;i++)
    {
        FaceEx * f = gm->GetFace(i);
        DebugPrint("Normals %d %d %d\n",f->norm[0],f->norm[1],
            f->norm[2]);
    }
}
```

# Parameter Access I

- Object properties are stored in different containers
  - IParamBlock, IParamBlock2, Custom Attributes, MXS plugins
- All Game entities provide an interface to extract properties

```
IGameConstraint * cnst = pGameControl->GetConstraint(IGAME_POS);
int iNumProps =
    cnst->GetIPropertyContainer()->GetNumberOfProperties();
for( int i=0; i<iNumProps; i++)
{
    IGameProperty * prop =
        cnst->GetIPropertyContainer()->GetProperty(i);
    if(prop->GetType() == IGAME_POINT3_PROP)
    {
        Point3 p;
        prop->GetPropertyValue(p);
        DebugPrint("Property value: %f %f %f \n", p.x, p.y, p.z);
    }
}
```

# Parameter Access II

- Define Custom Object Data Property
  - XML defined in IGameProp.xml

```
<ExportUserData>  
  <UserProperty>  
    <id>101</id>  
    <simplename>IGameNodeString</simplename>  
    <keyName>IGameNodeString</keyName>  
    <type>string</type>  
  </UserProperty>  
</ExportUserData>
```

- Extract Custom Property

```
IGameObject * obj = gameNode->GetIGameObject();  
IGameSupportObject * hO = (IGameSupportObject*)obj;  
IPropertyContainer * cc = hO->GetIPropertyContainer();  
IGameProperty * prop = cc->QueryProperty(101);  
if(prop)  
{  
    TCHAR * buf;  
    prop->GetPropertyValue(buf);  
}
```

# 3ds Max Game Interface Sample Exporter

- IGameExporter Sample
  - Object-level serialization into XML
  - Implemented using the 3ds Max Game Interface SDK
  - Traverses through the entire scene
  - Tags and serializes all parametric objects, modifiers, controllers, animations keys, and parameter block values
  - All of the 3ds Max Game Interface object classes are identified and exported on a per-object basis

# 3ds Max Game Interface Resources

- <http://sparks.autodesk.com>
- <http://beta.discreet.com>
- On the CD:
  - SDK.chm (maxsdk\help)
  - Game Headers (maxsdk\include\IGame)
  - Game Exporter (maxsdk\samples\impexp\IGameExporter)

End of Presentation  
Slide