

A proposal for how multitexturing should work in IFC4.3

By Dion Moulton and Ilkin Kavi

This proposal handles the topic of Multi Texturing in Ifc 4 and further formats. Currently IFC schema can support the textures with IfcSurfaceStyleWithTextures and carries textures via IfcSurfaceTexture.

In this proposal we are going to elaborate why the current way of multi texturing should be modified/enhanced and also provide some example ways of solving current issues.

Blending Modes

In IfcSurfaceStyle we have 5 attributes, these are: RepeatS, RepeatT, Mode, Texture Transform, Parameter. The "Mode" and the "Parameter" are the attributes for creating multi textures via different blending options.

In Ifc World a single texture can't exist, it is always either has to blend with the base color of the object or another texture. In this case there are different blending options that mixes RGBA colours whether it is multitexture or a texture and a colour. This attribute is carried by the "Mode" and it is determined by the modes described in ISO/IES 19775-1.2:2008 X3D Architecture and base components Edition 2, Part 1. In X3D v3.2 there are different blending types like, "MODULATE", "ADD", "SUBTRACT" or "REPLACE".

The parameter attribute is provided to control the appearance of the multi textures. This is also defined via X3D. By convention, Parameter[1] holds the source value, Parameter[2] the function value, Parameter[3] the base RGB color for select operations, and Parameter[4] the alpha value for select operations. The source value effects how the object will be rendered or opacity of the texture that we are using.

Current X3D behavior and IFC

In the current version of Ifc 4 we are inheriting the some of the attributes from the "MultiTexture Node" from X3D Part 1: Architecture and base components.

```
MultiTexture : X3DTextureNode {  
  SFFloat [in,out] alpha 1 [0,1]  
  SFCOLOR [in,out] color 1 1 1 [0,1]  
  MFString [in,out] function []  
  SFNode [in,out] metadata NULL [X3DMetadataObject]  
  MFString [in,out] mode []  
  MFString [in,out] source []  
  MFNode [in,out] texture [] [X3DTextureNode]  
}
```

The X3D *MultiTexture* Node and its values. The values IFC references are highlighted.

The MultiTexture node specifies the application of several individual textures to a 3D object to achieve a more complex visual effect. For the sake of simplicity we will only elaborate on the values we reference in IFC.

Color and Alpha: base RGB and alpha values for SELECT mode operations.

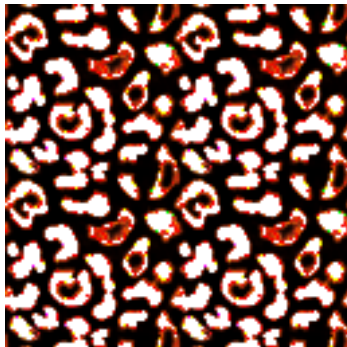
Mode: controls the type of blending operation. The available modes include MODULATE for a lit Appearance, REPLACE for an unlit Appearance, and several variations of the two. The mode field may contain an additional blending mode for the alpha channel.

Source: determines the color source for the second argument.

Function: defines an optional function to be applied to the argument after the mode has been evaluated.

To understand how these blending modes work and what's the significance of the source and function field we recreated some tests done by Michalis Kamburelis (https://castle-engine.io/x3d_multi_texturing.php) by blending two textures together and understanding how it behaves in the X3D world.

Two Textures:

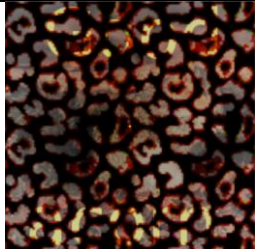




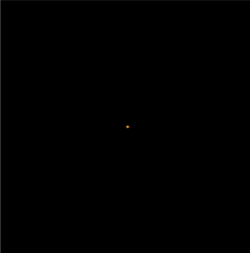
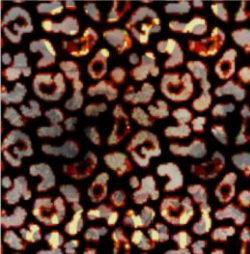
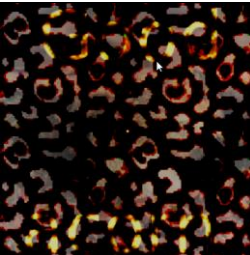
Arg2: pattern.png


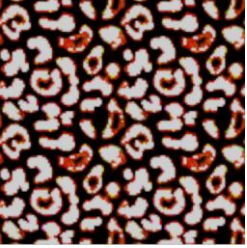
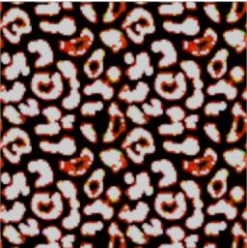





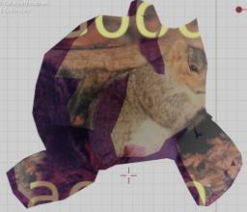

Arg1: squirrel.png

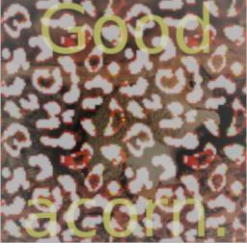
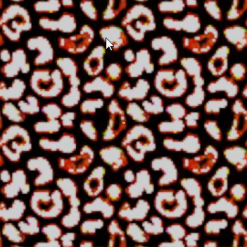

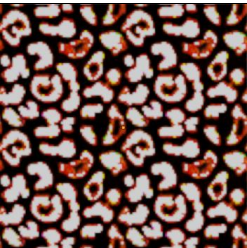
(Table1: Modes and their relations to Source field shown via examples.)

	Image	X3D Name	IFC	X3D Definition
Mixing Arg1(squirrel) and Arg2(pattern) together. Source Field = ""				
1		"MODULATE": Multiply texture color with current color Arg1 × Arg2	#1=IfcSurfaceStyleWithTextures((#2, #3)) #2=IfcImageTexture(\$,\$,'MODULATE',\$,\$,('','1 1 1','1'),'data/pattern.png') #3=IfcImageTexture(\$,\$,'MODULATE',\$,\$,('','1 1 1','1'),'data/squirrel.png')	appearance Appearance { texture MultiTexture { texture [ImageTexture { url ""data/pattern.png"" } ImageTexture { url ""data/squirrel.png"" }] } material Material { } }
2		"ADD": Add the components of the arguments Arg1 + Arg2	#1=IfcSurfaceStyleWithTextures((#2, #3)) #2=IfcImageTexture(\$,\$,'MODULATE',\$,\$,('','1 1 1','1'),'data/pattern.png') #3=IfcImageTexture(\$,\$,'ADD',\$,\$,('','1 1 1','1'),'data/squirrel.png')	appearance Appearance { texture MultiTexture { mode [""MODULATE"" ""ADD""] texture [ImageTexture { url ""data/pattern.png"" } ImageTexture { url ""data/squirrel.png"" }] } material Material { } }

3		<p>"REPLACE": Replace current color Arg2</p>	<pre>#1=IfcSurfaceStyleWithTextures((#2, #3)) #2=IfcImageTexture(\$,\$,'MODULATE ',\$,("','1 1 1','1'),'data/pattern.png') #3=IfcImageTexture(\$,\$,'REPLACE',\$, ("','1 1 1','1'),'data/squirrel.png')</pre>	<pre>appearance Appearance { texture MultiTexture { mode [""MODULATE"" ""REPLACE""] texture [ImageTexture { url ""data/pattern.png"" } ImageTexture { url ""data/squirrel.png"" }] } material Material { }</pre>
4		<p>"SUBTRACT": Subtract the components of the second argument from those of the first argument. Arg1 – Arg2</p>	<pre>#1=IfcSurfaceStyleWithTextures((#2, #3)) #2=IfcImageTexture(\$,\$,'MODULATE ',\$,("','1 1 1','1'),'data/pattern.png') #3=IfcImageTexture(\$,\$,'SUBTRACT', \$,(("','1 1 1','1'),'data/squirrel.png')</pre>	<pre>appearance Appearance { texture MultiTexture { mode [""MODULATE"" ""SUBTRACT""] texture [ImageTexture { url ""data/pattern.png"" } ImageTexture { url ""data/squirrel.png"" }] } material Material { }</pre>
5		<p>"MODULATE2X": Multiply the components of the arguments, and shift the products to the left 1 bit (effectively multiplying them by 2) for brightening.</p>	<pre>#1=IfcSurfaceStyleWithTextures((#2, #3)) #2=IfcImageTexture(\$,\$,'MODULATE ',\$,("','1 1 1','1'),'data/pattern.png') #3=IfcImageTexture(\$,\$,'MODULATE 2X',\$,("','1 1 1','1'),'data/squirrel.png')</pre>	<pre>appearance Appearance { texture MultiTexture { mode [""MODULATE"" ""MODULATE2X""] texture [ImageTexture { url ""data/pattern.png"" } ImageTexture { url ""data/squirrel.png"" }] } material Material { }</pre>
6		<p>"ADDSIGNED2X": Add the components of the arguments with a - 0.5 bias, and shift the products to the left 1 bit.</p>	<pre>#1=IfcSurfaceStyleWithTextures((#2, #3)) #2=IfcImageTexture(\$,\$,'MODULATE ',\$,("','1 1 1','1'),'data/pattern.png') #3=IfcImageTexture(\$,\$,'ADDSIGNED 2X',\$,("','1 1 1','1'),'data/squirrel.png')</pre>	<pre>appearance Appearance { texture MultiTexture { mode [""MODULATE"" ""ADDSIGNED2X""] texture [ImageTexture { url ""data/pattern.png"" } ImageTexture { url ""data/squirrel.png"" }] } material Material { }</pre>

7		<p>"SELECTARG1": Use color argument 1 Arg1</p>	<pre>#1=IfcSurfaceStyleWithTextures((#2, #3)) #2=IfcImageTexture(\$,\$,'MODULATE ',\$,("","1 1 1','1'),'data/pattern.png') #3=IfcImageTexture(\$,\$,'SELECTARG 1',\$,("","1 1 1','1'),'data/squirrel.png')</pre>	<pre>appearance Appearance { texture MultiTexture { mode [""MODULATE"" ""SELECTARG1""] texture [ImageTexture { url ""data/pattern.png"" } ImageTexture { url ""data/squirrel.png"" }] } material Material { }</pre>
8		<p>"SELECTARG2": Use color argument 2 Arg2</p>	<pre>#1=IfcSurfaceStyleWithTextures((#2, #3)) #2=IfcImageTexture(\$,\$,'MODULATE ',\$,("","1 1 1','1'),'data/pattern.png') #3=IfcImageTexture(\$,\$,'SELECTARG 2',\$,("","1 1 1','1'),'data/squirrel.png')</pre>	<pre>appearance Appearance { texture MultiTexture { mode [""MODULATE"" ""SELECTARG2""] texture [ImageTexture { url ""data/pattern.png"" } ImageTexture { url ""data/squirrel.png"" }] } material Material { }</pre>
9		<p>"OFF": Turn off the texture unit</p>	<pre>#1=IfcSurfaceStyleWithTextures((#2, #3)) #2=IfcImageTexture(\$,\$,'MODULATE ',\$,("","1 1 1','1'),'data/pattern.png') #3=IfcImageTexture(\$,\$,'OFF',\$,(""," 1 1 1','1'),'data/squirrel.png')</pre>	<pre>appearance Appearance { texture MultiTexture { mode [""MODULATE"" ""OFF""] texture [ImageTexture { url ""data/pattern.png"" } ImageTexture { url ""data/squirrel.png"" }] } material Material { }</pre>
Mixing Arg1(squirrel) with a constant yellow color together. Colours are not effected by light. Source Field = "FACTOR"				
10		<p>"MODULATE" : Multiply texture color with current color Arg1 × Arg2</p>	<pre>#1=IfcSurfaceStyleWithTextures((#3)) #2=IfcImageTexture(\$,\$,'MODULATE ',\$,("","1 1 0','1'),'data/pattern.png') #3=IfcImageTexture(\$,\$,'MODULATE ',\$,('FACTOR',"1 1 0','1'),'data/squirrel.png')</pre>	<pre>appearance Appearance { texture MultiTexture { texture [ImageTexture { url ""data/pattern.png"" } ImageTexture { url ""data/squirrel.png"" }] source ["""" ""FACTOR""] color 1 1 0 } material Material { }</pre>

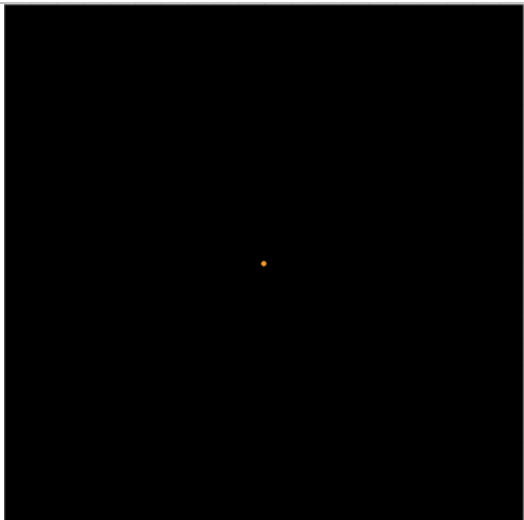
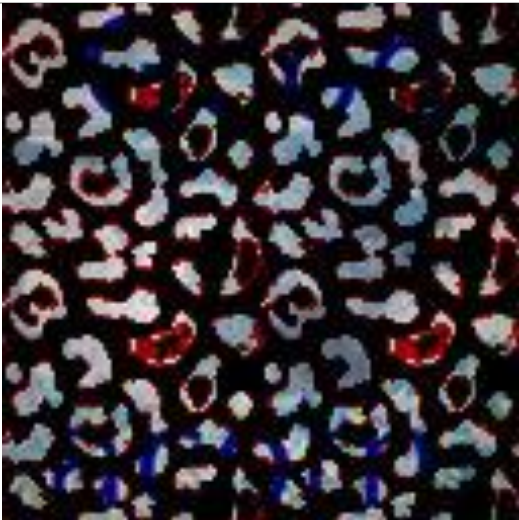
11		<p>"ADD": Add the components of the arguments Arg1 + Arg2</p>	<pre>#1=IfcSurfaceStyleWithTextures((#3)) #2=IfcImageTexture(\$,\$,'MODULATE ',\$,("','1 1 0','1'),'data/pattern.png') #3=IfcImageTexture(\$,\$,'ADD',\$,('FA CTOR','1 1 0','1'),'data/squirrel.png')</pre>	<pre>appearance Appearance { texture MultiTexture { mode [""MODULATE"" ""ADD""] texture [ImageTexture { url ""data/pattern.png"" } ImageTexture { url ""data/squirrel.png"" }] source ["""" ""FACTOR""] color 1 1 0 } material Material { } }</pre>
12		<p>"MODULATE2X": Multiply the components of the arguments, and shift the products to the left 1 bit (effectively multiplying them by 2) for brightening.</p>	<pre>#1=IfcSurfaceStyleWithTextures((#3)) #2=IfcImageTexture(\$,\$,'MODULATE ',\$,("','1 1 0','1'),'data/pattern.png') #3=IfcImageTexture(\$,\$,'MODULATE 2X',\$,('FACTOR','1 1 0','1'),'data/squirrel.png')</pre>	<pre>appearance Appearance { texture MultiTexture { mode [""MODULATE"" ""MODULATE2X""] texture [ImageTexture { url ""data/pattern.png"" } ImageTexture { url ""data/squirrel.png"" }] source ["""" ""FACTOR""] color 1 1 0 } material Material { } }</pre>
<p>Mixing Arg1(squirrel) with the diffuse Colour. Doing the lighting calculation before texturing (Gouraud shading). Mixing Arg1 (squirrel.png) with the lighting result Source Field = "DIFFUSE"</p>				
13		<p>"MODULATE" : Multiply texture color with current color Arg1 × Arg2</p>	<pre>#1=IfcSurfaceStyleWithTextures((#2, #3)) #2=IfcImageTexture(\$,\$,'MODULATE ',\$,("','1 1 1','1'),'data/pattern.png') #3=IfcImageTexture(\$,\$,'MODULATE ',\$,('DIFFUSE','1 1 1','1'),'data/squirrel.png')</pre>	<pre>appearance Appearance { texture MultiTexture { texture [ImageTexture { url ""data/pattern.png"" } ImageTexture { url ""data/squirrel.png"" }] source ["""" ""DIFFUSE""] } material Material { } }</pre>
14		<p>"MODULATE2X": Multiply the components of the arguments, and shift the products to the left 1 bit (effectively multiplying them by 2) for brightening."</p>	<pre>#1=IfcSurfaceStyleWithTextures((#2, #3)) #2=IfcImageTexture(\$,\$,'MODULATE ',\$,("','1 1 1','1'),'data/pattern.png') #3=IfcImageTexture(\$,\$,'MODULATE 2X',\$,('DIFFUSE','1 1 1','1'),'data/squirrel.png)'"</pre>	<pre>appearance Appearance { texture MultiTexture { mode [""MODULATE"" ""MODULATE2X""] texture [ImageTexture { url ""data/pattern.png"" } ImageTexture { url ""data/squirrel.png"" }] source ["""" ""DIFFUSE""] } material Material { } }</pre>

				}
BLENDXXX Mode testing				
15		"BLENDDIFFUSEALPHA": Linearly blend this texture stage, using the interpolated alpha from each vertex. $\text{Arg1} \times (\text{Alpha}) + \text{Arg2} \times (1 - \text{Alpha})$	#1=IfcSurfaceStyleWithTextures((#2, #3)) #2=IfcImageTexture(\$,\$,'REPLACE',\$, ("","'1 1 1','1'),'data/squirrel.png') #3=IfcImageTexture(\$,\$,'BLENDDIFFUSEALPHA',\$, ("","'1 1 1','1'),'data/pattern.png')	appearance Appearance { texture MultiTexture { texture [ImageTexture { url "data/squirrel.png" } ImageTexture { url "data/pattern.png" }] mode ["REPLACE" "BLENDDIFFUSEALPHA"] } material Material { transparency 0.5 } }
16		"BLENDTEXTUREALPHA": Linearly blend this texture stage, using the alpha from this stage's texture. $\text{Arg1} \times (\text{Alpha}) + \text{Arg2} \times (1 - \text{Alpha})$	#1=IfcSurfaceStyleWithTextures((#2, #3)) #2=IfcImageTexture(\$,\$,'REPLACE',\$, ("","'1 1 1','1'),'data/squirrel.png') #3=IfcImageTexture(\$,\$,'BLENDTEXTUREALPHA',\$, ("","'1 1 1','1'),'data/pattern.png')	appearance Appearance { texture MultiTexture { texture [ImageTexture { url "data/squirrel.png" } ImageTexture { url "data/pattern.png" }] mode ["REPLACE" "BLENDTEXTUREALPHA"] } material Material { } }
17		"BLENDFACTORALPHA": Linearly blend this texture stage, using the alpha factor from the MultiTexture node. $\text{Arg1} \times (\text{Alpha}) + \text{Arg2} \times (1 - \text{Alpha})$	#1=IfcSurfaceStyleWithTextures((#2, #3)) #2=IfcImageTexture(\$,\$,'REPLACE',\$, ("","'1 1 1','1'),'data/squirrel.png') #3=IfcImageTexture(\$,\$,'BLENDFACTORALPHA',\$, ("","'1 1 1','1'),'data/pattern.png')	appearance Appearance { texture MultiTexture { texture [ImageTexture { url "data/squirrel.png" } ImageTexture { url "data/pattern.png" }] mode ["REPLACE" "BLENDFACTORALPHA"] alpha 0.2 } material Material { } }
18		"BLENDCURRENTALPHA": Linearly blend this texture stage, using the alpha taken from the previous texture stage. $\text{Arg1} \times (\text{Alpha}) + \text{Arg2} \times (1 - \text{Alpha})$	#1=IfcSurfaceStyleWithTextures((#2, #3)) #2=IfcImageTexture(\$,\$,'REPLACE',\$, ("","'1 1 1','1'),'data/squirrel.png') #3=IfcImageTexture(\$,\$,'BLENDCURRENTALPHA',\$, ("","'1 1 1','1'),'data/pattern.png')	appearance Appearance { texture MultiTexture { texture [ImageTexture { url "data/squirrel.png" } ImageTexture { url "data/pattern.png" }] mode ["REPLACE" "BLENDCURRENTALPHA"] } material Material { } } geometry USE MyQuad }

Although not all, these examples cover some most used blending modes some not so common.

In X3D if a mode wasn't specified it is by default "MODULATE" which is multiplying the RGBA channels of the two textures. This, however unintuitive, isn't the biggest issue we face with the current X3D behavior.

In the default scenario of two opaque textures, the subtract mode applies to RGBA. This means that the alpha channel then becomes 0: an invisible texture. This is one of the most common mistakes in X3D implementations. This is not an ideal way of working for artists and rendering engines. Today artists expect blending modes to be applied to only RGB channel, the expected default behavior of the alpha channel is "ADD" Blending mode.

How Subtract Mode behaves in X3D	How Subtract Mode behaves in Photoshop
	

(Table2: Subtract mode in X3D vs. Photoshop. Also look Table1 Example Nr. 4)

As it is shown the result we get with X3D is far from what is desired to achieve.

We propose 3 possible solutions to this problem:

Option 1:

- By default, when a single mode such as "SUBTRACT" is applied, it only effects the RGB channels.
- Implicitly, mode of "MODULATE" is the default applies to the A channel.

This option although helpful, isn't a very good solution. It is proposed to use "MODULATE" as default mode for A channel because it is the default if a none value was given. But still it is quite unintuitive and gives unexpected results.

How this option would look like in IFC:

```
#1=IfcSurfaceStyleWithTextures((#2,#3))
#2=IfcImageTexture($,$,'MODULATE',$,('','1 1 1','1'),'data/pattern.png')
#3=IfcImageTexture($,$,'SUBTRACT',$,('','1 1 1','1'),'data/squirrel.png')
```

Option 2:

- By default, when a single mode such as "SUBTRACT" is applied, it only effects the RGB channels.
- Implicitly, mode of "REPLACE" is always applied to the A channel of the first texture.

- Subsequent A channel modes default to use “ADD” mode

By changing the default mode with “REPLACE” we create better and more expected results. Currently in other editors the default blending mode for alpha is always “ADD” and therefore it makes sense to, by default, making any subsequent textures alpha channel to “ADD” mode.

How this option would look like in IFC:

```
#1=IfcSurfaceStyleWithTextures((#2,#3))
#2=IfcImageTexture($,$,'MODULATE',$,('','1 1 1','1'),'data/pattern.png')
#3=IfcImageTexture($,$,'SUBTRACT',$,('','1 1 1','1'),'data/squirrel.png')
```

Option 3:

- By default, when a single mode such as “SUBTRACT” is applied, it only effects the RGB channels.
- Textures will always have two modes, as per the X3D spec to explicitly distinguish between RGB and A channels.
- The two modes can be separated by Whitespace.
- Just whitespace in modes are ignored (can’t leave the mode blank)
- This is basically the as option 2, but instead of implicit defaults, it must always be explicitly specified

With this option we don’t leave anything to confusion. The first mode specified will always be effecting the RGB channels and the second one A channel.

How this option would look like in IFC:

```
#1=IfcSurfaceStyleWithTextures((#2,#3))
#2=IfcImageTexture($,$,'MODULATE REPLACE',$,('','1 1 1','1'),'data/pattern.png')
#3=IfcImageTexture($,$,'SUBTRACT ADD',$,('','1 1 1','1'),'data/squirrel.png')
```

	Current Behaviour	Option 1		Option 2		Option 3	
Channel	RGBA	RGB	A	RGB	A	RGB	A
Mode	MODULATE	User def.	MODULATE	User def.	REPLACE	User def.	User def.
Subs. Mode		User def.	MODULATE	User def.	ADD	User def.	User def.
Explanation	The modes are applied to all channels. Default mode “MODULATE” can be also user defined.	User defined mode applied to RGB channels. By default “MODULATE” is applied to A channel		User defined mode applied to RGB channels. By default “REPLACE” is applied to A channel. Any subsequent mode gets the “ADD” mode as for A channel		User defined modes are applied to both RGB and A channels but separately.	

(Table3: A summary of the different options proposed and current settings in IFC)

Some common texturing situations

Scenario 1: One Diffuse texture and nothing else

In this scenario we would only have one Image Texture and the default mode would be “REPLACE”

```
#1=IfcSurfaceStyleWithTextures((#2))
```



```
#2=IfcImageTexture($,$,'REPLACE',$('','1 1 1','1'),'data/squirrel.png')
```

Scenario 2: A diffuse texture mixing with a flat colour and nothing else

In this scenario the blending mode would be “BLENDFACTORALPHA” which is one of the most common blending modes applied by artists.

```
#1=IfcSurfaceStyleWithTextures((#2))
#2=IfcImageTexture($,$,'BLENDFACTORALPHA',$('FACTOR','1 0 0','0.2'),'data/squirrel.png')
```

Scenario 3: Using a different blending mode with factor.

In scenario 2, "REPLACE" is the implicit blending mode. However artists use a variety of modes. X3D does not have enum values for these modes. We propose to add the following modes:

- *MODULATEFACTORALPHA*
- *ADDFACTORALPHA*
- *SUBTRACTFACTORALPHA*
- ... in short, the name of the *blending* mode with the suffix "FACTORALPHA" to denote the factor source

X3D only has a handful of blending modes. Most notably REPLACE, MODULATE, ADD, SUBTRACT. Artists commonly use many more blending modes. The list is quite long, however only handful are commonly used in textures. Here 's a list of common ones:

- DARKEN
- LIGHTEN
- SCREEN
- DIFFERENCE

(https://en.wikipedia.org/wiki/Blend_modes shows some relevant equations)

```
#1=IfcSurfaceStyleWithTextures((#2))
#2=IfcImageTexture($,$,'MODULATEFACTORALPHA',$('FACTOR','1 0 0','0.2'),'data/squirrel.png')
```

In this example Texture is multiplied with a base color (in this case red) with the factor of 0.2 (alpha).

Scenario 4: One diffuse texture and a bump map

The X3D v3.2 specification is not capable of storing different texture maps. A proposal was made to extend X3D v3.2 to store texture maps in its *Appearance* entity (https://castle-engine.io/x3d_implementation_texturing_extensions.php#section_ext_bump_mapping). In X3D v4, similar to the proposal, texture maps are stored in the Material entity. Unfortunately, neither *Appearance* nor *Material* have any relationship to IFC, so there is no obvious way to use this proposal in IFC. However it has a relation with the *Source* parameter, which are specified in X3D as:

- "" : This is the default mode if nothing else specified. The second argument color (ARG2) is the color from the previous rendering stage (DIFFUSE for first stage).
- "DIFFUSE": The texture argument is the diffuse color interpolated from vertex components during Gouraud shading.
- "SPECULAR": The texture argument is the specular color interpolated from vertex components during Gouraud shading.
- "FACTOR": The texture argument is the factor (color, alpha) from the MultiTexture node. This is used when a texture is mixed with a color with an alpha value.

Despite the naming similarity to things like Diffuse or Specular Map, this is not the purpose of these values. Also the fact that the purpose of Diffuse and Specular is to combine the texture with the Gouraud shading makes them near obsolete. Unless there is a specific purpose no one uses Gouraud Shading anymore. For this reason, we recommend only the value "" and "FACTOR" should be featured in the first parameter value in IfcSurfaceTexture entity.

To be able to create more photorealistic renderings we propose a new parameter, just as a new attribute in X3D, to be added to IfcSurfaceTexture. This means that there will be 5th parameter in the list and it will refer to the map type. The list of valid map types is based off IFC2x3;

- BUMP
- NORMAL (added new)
- OPACITY (before: TRANSPARENCYMAP)
- SELFILLUMINATION
- SHININESS (It should be mentioned that this is the inverse version of Roughness to avoid confusion)
- SPECULAR (before: REFLECTION)
- DIFFUSE(before: TEXTURE)
- NOTDEFINED

Some of these types were renamed to avoid confusion with the users and also to use more correct terminology to fit the other rendering engines, softwares etc.

There may be other uncommon or complex maps like; AO, ALBEDO, SSS, DISPLAMENT, METAL, etc. but probably it is not yet necessary to list every possible map.

Example:

```
#1=IfcSurfaceStyleWithTextures((#2,#3))
#2=IfcImageTexture($,$,'REPLACE',$,('','1 1 1','1','DIFFUSE'),'data/pattern.png')
#3=IfcImageTexture($,$,'REPLACE',$,('','1 1 1','1','BUMP'),'data/squirrel.png')
```

In arch viz, the shader is a combination of texture maps and a shader type with parameters. The shader type is not part of the texture maps and is not in scope for IfcSurfaceTexture. The shader type is defined in IfcSurfaceStyleRendering: FLAT, MATT, METAL, MIRROR, etc. Depending on the shader type, certain texture maps may not apply. For example, only a DIFFUSE map will affect a FLAT shader type. Therefore for a full shader, IfcSurfaceStyleRendering and IfcSurfaceStyleWithTextures must be read together. A shader may require extra parameters. E.G. a GLASS shader needs an IOR. There is no place to put these right now. Some parameters may be arbitrary and are specific to each engine. Others are common. There are only a handful of popular engines which can be catered to on a case-by-case basis.

A few common ones are listed for consideration to be added as attributes to IfcSurfaceStyleRendering:

- Anisotropy
- Roughness (distinct from Specularity)
- IOR

Scenario 5: Every single texture map

At this point, after all the proposals, in theory we can do everything, with some engine-specific caveats.

Here a normal map is used, not a bump map, since they are mutually exclusive

```
#1=IfcSurfaceStyleWithTextures((#2,#3,#4,#5,#6,#7))
#2=IfcImageTexture($,$,'REPLACE',$,('','1 1 1','1','DIFFUSE'),'data/pattern.png')
#3=IfcImageTexture($,$,'REPLACE',$,('','1 1 1','1','NORMAL'),'data/squirrel.png')
#4=IfcImageTexture($,$,'REPLACE',$,('','1 1 1','1','OPACITY'),'data/squirrel.png')
#5=IfcImageTexture($,$,'REPLACE',$,('','1 1 1','1','SELFILLUMINATION'),'data/squirrel.png')
#6=IfcImageTexture($,$,'REPLACE',$,('','1 1 1','1','SHININESS'),'data/squirrel.png')
#7=IfcImageTexture($,$,'REPLACE',$,('','1 1 1','1','SPECULAR'),'data/squirrel.png')
```

Scenario 6: Diffuse and normal, but diffuse multiplies with a 50% factor of two images.

```
#1=IfcSurfaceStyleWithTextures((#2,#3,#4))
#2=IfcImageTexture($,$,'REPLACE',$,('','1 1 1','1','DIFFUSE'),'data/pattern.png')
#3=IfcImageTexture($,$,'REPLACE',$,('','1 1 1','1','NORMAL'),'data/squirrel.png')
#4=IfcImageTexture($,$,'MODULATEFACTORALPHA',$,('','1 1 1','0.5','DIFFUSE'),'data/squirrel.png')
```