

Hammered to the Max

A Hammer-User's Guide to 3ds Max

Shawn Olson



Hammered to the Max

A Hammer User's Guide to 3ds Max

By Shawn Olson



Hammered to The Max: A Hammer User's Guide to 3ds Max

Copyright © 2016-2018 by Shawn Olson

All Rights Reserved

<http://www.shawnolson.net>

Cover Image of Joris Ceoen's *Space Marble Unlimited*.

Limit of Liability/Disclaimer of Warranty: The author and publisher make no claims, guarantees or warranties pertaining to the accuracy or completeness of this book. There are no claims that the contents in this book are fit for a particular purpose, and no guarantee or warranty is extended. This work is distributed with the understanding that the author and publisher are not extending legal or any other professional service. Neither the publisher nor author can be held liable for damages arising from the contents of this publication. The fact that any third-party is referenced in this document does not imply a guarantee or endorsement of that third-party application. Furthermore, the reader should be aware that Hyperlinks in this document may point to documents or resources that have changed since publication.

TRADEMARKS: Wall Worm, Wall Worm Pro, Wall Worm Model Tools, Angry Teapot, Brushify, Wall Worm CorVex, ShellVex, PropLine and the Wall Worm logo are trademarks of Wall Worm and affiliates and may not be used without written permission. Autodesk 3ds Max, Gmax and Autodesk Vault are trademarks of Autodesk, Inc. Half-Life, Goldsource, Half-Life 2, Left for Dead 2, Counter-Strike, Counter-Strike Source, Counter-Strike: Global Offensive, Worldcraft, Hammer and the Source Engine are trademarks of Valve Corporation. All other trademarks are the property of their respective owners.

Table of Contents

Hammered to the Max.....	2
A Hammer User's Guide to 3ds Max.....	2

Acknowledgment.....	25
Chapter 1 Introduction.....	27
Who Should Read this Publication.....	29
What you Need.....	29
About Source 2.....	30
About Wall Worm Pro	30
Chapter 2 Preparing for 3ds Max.....	31
You are not going to learn everything overnight.....	31
The same principles of BSP level design apply in Max as they do in Hammer.....	32
The way you approach the design is not the same as in Hammer.....	32
Constant Innovation.....	33
Who is Using Wall Worm?.....	34
Chapter 3 Introduction to The 3ds Max User Interface.....	37
Create Tab.....	37
Your First Box.....	38

Modify Tab.....	39
 Modify Your Box.....	39
Display Tab.....	39
Navigation.....	40
 Flying Through the Scene.....	41
 Zoom Extents Selected.....	41
 Zooming.....	41
 Orbit Selected.....	41
Getting Familiar with the UI.....	42
Hammer and Max Function Chart.....	43
Chapter 4 Essential Tools.....	44
 Main Toolbar.....	44
 Transformation Tools.....	45
 Coordinate Systems.....	47
 The Grid and Snaps.....	49

The Grid and System Units	49
Standard Snaps.....	50
Angle, Percent and Spinner Snaps.....	52
Other Main Toolbar Buttons.....	52
Chapter 5 Basic Editing Concepts.....	54
 Editable Poly Objects.....	54
 Slicing Objects (Clipping Objects).....	57
 Carve and Make Hollow.....	58
Chapter 6 Welcome to Wall Worm.....	60
 A Brief History of Wall Worm.....	60
 Getting Started with Wall Worm.....	61
 Download Wall Worm.....	61
 Installing Wall Worm.....	62
 Run the Installer Script.....	63
 Configure Wall Worm.....	63

<u>Import Settings.....</u>	<u>65</u>
<u>Tips on Settings.....</u>	<u>65</u>
<u>Understanding and Using Wall Worm.....</u>	<u>67</u>
<u>Wall Worm Model Tools.....</u>	<u>68</u>
<u>Wall Worm Level Design Tools & Anvil.....</u>	<u>69</u>
<u>Wall Worm Materials.....</u>	<u>69</u>
<u>Utilities, Importers, Exporters and Extras.....</u>	<u>69</u>
<u>Wall Worm Conventions.....</u>	<u>70</u>
<u>Wall Worm's Focus is about Creating Custom Content.....</u>	<u>70</u>
<u>Wall Worm Team Work.....</u>	<u>71</u>
<u>Extra Wall Worm Functions.....</u>	<u>71</u>
<u>Adding Common Hammer Functions in Max.....</u>	<u>71</u>
<u>Chapter 7 Source Engine Level Design in Max.....</u>	<u>72</u>
<u>World Geometry.....</u>	<u>72</u>
<u>The Difference Between Delete and Remove Functions.....</u>	<u>73</u>

<u>Avoid Coplanar Polygons on World Geometry.....</u>	<u>74</u>
<u>Fixing Coplanar Polygons Manually.....</u>	<u>74</u>
<u>Fixing Coplanar Polygons Automatically.....</u>	<u>75</u>
<u>Keep Your Polygons Planar (Not all Convex Objects are Valid Because of Non-Planar Polygons).....</u>	<u>76</u>
<u>Fixing Non-Planar Polygons Manually.....</u>	<u>77</u>
<u>Connecting Vertices.....</u>	<u>77</u>
<u>Cutting Polygons.....</u>	<u>77</u>
<u>Fixing Non-Planar Polygons Automatically.....</u>	<u>78</u>
<u>The Brushify Modifier.....</u>	<u>79</u>
<u>Assigning World Geometry.....</u>	<u>79</u>
<u>Tags.....</u>	<u>80</u>
<u>Layers.....</u>	<u>80</u>
<u>Native Brushes.....</u>	<u>81</u>
<u>Brush Mode.....</u>	<u>81</u>

<u>Concave Brushes.....</u>	<u>81</u>
<u>Brush Explorer.....</u>	<u>83</u>
<u>Entities.....</u>	<u>84</u>
<u>Chapter 8 Source Engine Models.....</u>	<u>85</u>
<u> Make Your First Model in Source.....</u>	<u>85</u>
<u> The WWMT Helper.....</u>	<u>87</u>
<u> Wall Worm Proxies.....</u>	<u>87</u>
<u> Make Skins from Proxies.....</u>	<u>89</u>
<u> Tips on WWMT Helpers and Proxies.....</u>	<u>91</u>
<u> More Tips on Proxies.....</u>	<u>91</u>
<u> Controlling Prop Types of WWMT and Proxies.....</u>	<u>92</u>
<u> Working with Multiple Models and Proxies.....</u>	<u>92</u>
<u> Multiple WWMT Helpers.....</u>	<u>93</u>
<u> Quick WWMT.....</u>	<u>93</u>
<u> Wall Worm Proxy Tools.....</u>	<u>94</u>

<u>Changes that Require Re-Compile.....</u>	<u>94</u>
<u>Converting Scenes to Models.....</u>	<u>95</u>
<u>Clustering Models.....</u>	<u>96</u>
<u>Clustering Setup.....</u>	<u>96</u>
<u>Clustering Your Props.....</u>	<u>97</u>
<u>Chapter 9 Materials and Textures.....</u>	<u>99</u>
<u>Dropping the Hammer Approach</u>	<u>100</u>
<u>Defining Materials and Textures.....</u>	<u>100</u>
<u>Setting Up Materials and Maps for Source.....</u>	<u>102</u>
<u>Render Map.....</u>	<u>104</u>
<u>Render to Texture.....</u>	<u>104</u>
<u>Available Materials.....</u>	<u>104</u>
<u>Material and Map Conventions.....</u>	<u>105</u>
<u>The Materials and Source Shader Parameters.....</u>	<u>106</u>
<u>Advanced Source Material Controls.....</u>	<u>107</u>

<u>Apply to Materials in Material Editor.....</u>	<u>107</u>
<u>Apply to Materials of Selected Objects.....</u>	<u>108</u>
<u>Blends, WorldVertexTransition and LightMapped_4wayBlend Shader</u>	<u>108</u>
<u>Convert Blend/Composite to DX Shader.....</u>	<u>110</u>
<u>Model Materials VS World Materials.....</u>	<u>110</u>
<u>Export Model Materials.....</u>	<u>111</u>
<u>Export Brush and Displacement Materials.....</u>	<u>111</u>
<u>Worm Face (Hammer-like Texture Application).....</u>	<u>112</u>
<u>Set Worm Face Material.....</u>	<u>112</u>
<u>Drop Material on a Face.....</u>	<u>112</u>
<u>Working With Textures.....</u>	<u>113</u>
<u>Advanced Settings (Wall Worm Pro).....</u>	<u>113</u>
<u>Bitmaps to Materials.....</u>	<u>114</u>
<u>Tri-Planar Mapping inside Max.....</u>	<u>114</u>
<u>Chapter 10 Displacements.....</u>	<u>116</u>

<u>Understanding Wall Worm Displacements.....</u>	<u>116</u>
<u>The three Displacement objects WW creates are:.....</u>	<u>117</u>
<u>Creating Displacements.....</u>	<u>118</u>
<u>The Sculpt Mesh.....</u>	<u>119</u>
<u>Create a Sculpt Mesh.....</u>	<u>119</u>
<u>Sculpt Mesh Functions.....</u>	<u>120</u>
<u>Auto Hide Displacements.....</u>	<u>121</u>
<u>Display Walkable.....</u>	<u>121</u>
<u>Vertex Paint for Blends.....</u>	<u>121</u>
<u>Selection Functions.....</u>	<u>122</u>
<u>Assign Functions.....</u>	<u>123</u>
<u>Sew Selected Vertices:.....</u>	<u>123</u>
<u>Commit.....</u>	<u>124</u>
<u>Show Commit Status.....</u>	<u>124</u>
<u>Collapse Displacements.....</u>	<u>124</u>

<u>Commit UVs.....</u>	<u>124</u>
<u>Commit Multiblends.....</u>	<u>125</u>
<u>Commit Multiblends Colors.....</u>	<u>125</u>
<u>Commit All Changes.....</u>	<u>125</u>
<u>Commit Selected Changes.....</u>	<u>125</u>
<u>Revert to Pieces.....</u>	<u>126</u>
<u>Sculpt Mesh Modify Functions.....</u>	<u>126</u>
<u>Add Displacement.....</u>	<u>126</u>
<u>Remove Selected (Displacement).....</u>	<u>126</u>
<u>Remove and Delete.....</u>	<u>126</u>
<u>Remove to New Sculpt.....</u>	<u>127</u>
<u>Sculpt Mesh Miscellaneous Functions.....</u>	<u>127</u>
<u>Launch Displacement Floater.....</u>	<u>127</u>
<u>Bake Info.....</u>	<u>127</u>
<u>Clear Bake.....</u>	<u>127</u>

<u>Send Material to Pieces.....</u>	<u>128</u>
<u>Fix Alphas of Displacements.....</u>	<u>128</u>
<u>Collapse on Quadrify.....</u>	<u>128</u>
<u>Quadrify Me.....</u>	<u>128</u>
<u>Triangulate Me.....</u>	<u>129</u>
<u>Displacement Examples.....</u>	<u>129</u>
<u>Changing Power of Displacements.....</u>	<u>131</u>
<u>Displacement UVs.....</u>	<u>132</u>
<u>Clipping Displacements.....</u>	<u>132</u>
<u>Controlling the Brushes Behind Displacements.....</u>	<u>132</u>
<u>Manually Assign Brush Face to Displacement.....</u>	<u>133</u>
<u>Advanced Displacement Discussions.....</u>	<u>133</u>
<u>Parametric Displacements.....</u>	<u>133</u>
<u>Use Height Map to Control Displacements and Blending.....</u>	<u>134</u>
<u>Important Changes in Displacements (WW 2.82+).....</u>	<u>135</u>

<u>Displacement Explorer.....</u>	<u>135</u>
<u>Chapter 11 Skies.....</u>	<u>137</u>
<u>Getting Started with your Sky.....</u>	<u>137</u>
<u>Initial Setup.....</u>	<u>138</u>
<u>Lighting.....</u>	<u>139</u>
<u>3D Sky Layout.....</u>	<u>140</u>
<u>Making your 3D Skybox.....</u>	<u>141</u>
<u>Your 2D Sky and Sky Writer.....</u>	<u>142</u>
<u>Common Tools for Making Skies.....</u>	<u>143</u>
<u>Common Tools for Populating your Landscape.....</u>	<u>143</u>
<u>Finishing Your Sky.....</u>	<u>144</u>
<u>Chapter 12 Working With Entities.....</u>	<u>146</u>
<u>Access to Entities.....</u>	<u>146</u>
<u>Point Entities.....</u>	<u>147</u>
<u>Objects that Automatically Export as Point Entities.....</u>	<u>147</u>

<u>Placing other Entities.....</u>	<u>150</u>
<u>Appending to Brush Entity.....</u>	<u>150</u>
<u>Entity Outputs.....</u>	<u>150</u>
<u>Entity Manager.....</u>	<u>152</u>
<u>Worldspawn.....</u>	<u>153</u>
<u>Entity Explorer.....</u>	<u>153</u>
<u>Edit Selected Entity.....</u>	<u>154</u>
<u>Chapter 13 Importing Your VMF and MDLs into Max.....</u>	<u>155</u>
<u>Import a VMF.....</u>	<u>155</u>
<u>Working with Imported VMF.....</u>	<u>156</u>
<u>Importing Models.....</u>	<u>157</u>
<u>Import Single MDL.....</u>	<u>157</u>
<u>Import Prop to Modify.....</u>	<u>158</u>
<u>Create Prop Zoo From VMF.....</u>	<u>158</u>
<u>Load Multiple Props at Once.....</u>	<u>159</u>

<u>Creating WallWormMDL Nodes.....</u>	<u>159</u>
<u>MDL Node Functions.....</u>	<u>160</u>
<u>Chapter 14 Exporting Your Level into Source.....</u>	<u>162</u>
<u>Light Settings.....</u>	<u>163</u>
<u>RAD Files for Radiant Materials and Models with Transparency.....</u>	<u>163</u>
<u>PAK Assets.....</u>	<u>164</u>
<u>Working with PAK and RES files.....</u>	<u>165</u>
<u>Using the Res Maker.....</u>	<u>165</u>
<u>More Info.....</u>	<u>166</u>
<u>Include Options.....</u>	<u>167</u>
<u>WWMT Models.....</u>	<u>168</u>
<u>WWMT Textures.....</u>	<u>168</u>
<u>Sky Textures.....</u>	<u>168</u>
<u>Entities / CVX Props.....</u>	<u>168</u>
<u>World Textures.....</u>	<u>169</u>

Soundscapes.....	169
Distributing Your Level.....	169
Chapter 15 Troubleshooting Export and Compile Problems.....	170
Scene is Blank or Objects Missing in Game or Hammer.....	170
Invalid Geometry when Scene Opened in Hammer	171
Finding the Invalid Brushes in Max.....	171
Finding Leaks.....	171
Viewing Portal Files.....	172
An Error Freezes the Max UI.....	172
Chapter 16 Wall Worm Scene Manager.....	173
Advanced Scene Manager Topics.....	174
Scene Manager Collections.....	174
Scene Manager Methods.....	175
Chapter 17 Introductory Summation.....	176
Special, Personal Tips.....	177

Part II.....	179
Digging a Little Deeper.....	179
Chapter 18 UVW and Texture Mapping.....	180
Different UVW Strategies.....	180
Case Study: Texturing a Simple Scene.....	183
Acquiring UVs.....	186
Layering UV Modifiers.....	186
Apply Materials to Object.....	188
Reusing Your Modifiers.....	190
Fine-Tuning Your UVs.....	191
Better Method than Unwrap UVW: PolygonMap.....	194
MapScaler Modifier.....	196
Case Study: Texturing Models.....	197
Design Setup.....	197
Adding the Materials.....	202

<u>Learn More About Texturing.....</u>	<u>208</u>
<u>Chapter 19 CorVex and Parametric Level Design</u>	<u>210</u>
<u>Introduction to Using CorVex.....</u>	<u>210</u>
<u>Your First CorVex-driven Geometry.....</u>	<u>211</u>
<u>Rebuild the Wall With CorVex.....</u>	<u>212</u>
<u>Using Instanced Splines & Wire Parameters.....</u>	<u>214</u>
<u>Instancing Objects.....</u>	<u>215</u>
<u>Wiring Parameters.....</u>	<u>215</u>
<u>Modify an Existing CorVex Spline Base.....</u>	<u>218</u>
<u>Sealing the Level With CorVex.....</u>	<u>219</u>
<u>Planning Your CorVex Layout with Displacements in Mind.....</u>	<u>222</u>
<u>Make Displacements from CorVex.....</u>	<u>223</u>
<u>Create Displacements from Selected Faces.....</u>	<u>224</u>
<u>Some Tools helpful for Sculpt Mesh.....</u>	<u>226</u>
<u>Using CorVex with PropLine.....</u>	<u>227</u>

<u>Chapter 20 Scattering Props and Using Forest.....</u>	<u>228</u>
<u> Scattering Props with VBSP vs Using Forest.....</u>	<u>228</u>
<u> Flaws of using Detail.VBSP Files.....</u>	<u>228</u>
<u> Benefits of Using Forest.....</u>	<u>229</u>
<u> Wall Worm and Forest.....</u>	<u>230</u>
<u> Prepare to Scatter Props.....</u>	<u>230</u>
<u>Chapter 21 Anatomy of a Design Team.....</u>	<u>232</u>
<u> Project Setup.....</u>	<u>232</u>
<u> Setting Up Wall Worm with this Project.....</u>	<u>233</u>
<u> Paths.....</u>	<u>234</u>
<u> Models.....</u>	<u>234</u>
<u> Materials.....</u>	<u>235</u>
<u> Miscellaneous.....</u>	<u>235</u>
<u> Save and Store Preset.....</u>	<u>235</u>
<u> Create a MAXSTART file.....</u>	<u>236</u>

<u>Vault Setup.....</u>	<u>237</u>
<u>Create Vault.....</u>	<u>238</u>
<u>Create Users.....</u>	<u>238</u>
<u>Set Vault Settings.....</u>	<u>239</u>
<u>Add Folders and Files to the Vault.....</u>	<u>239</u>
<u>Team Member Setup.....</u>	<u>241</u>
<u>Vault Client Login.....</u>	<u>241</u>
<u>Download (Get) Files.....</u>	<u>241</u>
<u>Vault in 3ds Max.....</u>	<u>242</u>
<u>Installing the Project Presets.....</u>	<u>243</u>
<u>Pipeline Overview.....</u>	<u>244</u>
<u>File and Project Structure.....</u>	<u>244</u>
<u>Check In/Out Files.....</u>	<u>245</u>
<u>Model and Scene Referencing.....</u>	<u>247</u>
<u>Referencing Understood.....</u>	<u>248</u>

Exporting Assets.....	252
Collaboration.....	253
User Roles.....	253
Team Leaders.....	253
Level Designers.....	254
Artists.....	254
Animators.....	254
Audio Technicians.....	254
Programmers.....	255
Level Setups.....	256
Conclusion.....	258
Chapter 22 Useful Tools for the Source Engine.....	259
General Source Engine Tools.....	261
More 3ds Max Resources.....	262
3ds Max Tools and Developers.....	262

Source Engine Resources.....	262
More Game-related forums.....	263
Chapter 23 Conclusion.....	264

Acknowledgment

Many people have helped me as I've been developing the Wall Worm tools for Source over the last several years. Their contributions, advice and time have been immeasurable. For various 3ds Max and MAXScript help, the following people have been a great aid in person or in public CGI forums: Steve Curley, Denis “denisT” Trofimov, Vojtěch “swordslayer” Čada, Kostadin “miauu” Kotev. Important contributors of code and functions in Wall Worm include Orvid King, Dave Brennan, Fabian Groß and Corey “MarioLart64” Nguyen.

Other developers that have shared tools with me or graciously enhanced their tools (at my request with Wall Worm in mind) include Vladislav Gavrilov, Neil “Jed” Jedrzejewski, Michael Little and Cannonfodder.

Some of my close friends have also been immensely valuable to my adventures in developing Wall Worm and writing this book. Gulliver “K@rt” Thoday is often pummeled with half-finished tools and my need for front-line tests. He's repeatedly endured sitting through my evangelical discussions on Max and Wall Worm. His feedback has always helped improve Wall Worm.

Joris Ceoen has also played a significant role in providing testing and feedback, especially relating to the updates developed for WW 2.0. The cover features one of his levels made with Wall Worm.

Andrew Penry is both a close friend and my former business partner. Andrew has repeatedly helped me solve various problems that vexed me. Some of the key features in Wall Worm were only possible because Andrew helped solve them—especially mathematical problems dealing with transformations.

Also, I want to share appreciation for all of the great motivational emails from the Source

Acknowledgment

modding community and all the Wall Worm users. The creativity, excitement and kind words have always energized me. And those who have donated financially have only made it nicer to share Wall Worm. The most substantial donor who requires mention for repeatedly making helpful donations is Rick Underhill. Thank You!

I want to thank Robert Briscoe for single-handedly boosting my spirits at a low time by hiring me to add some tools in Wall Worm for *Dear Esther*. And I have ongoing thanks to *Black Mesa* for contracting me for helping in the historic remake of the classic *Half-Life*. Other *Black Mesa* employees that have helped work on Wall Worm itself include Chetan Jaggi and Richard Geslot.

There are others who should probably also be on this page. Know that I value all people who have shared time, ideas, donations and encouragement.

Chapter 1 Introduction

When *Half-Life* and *Counter-Strike* were introduced in the late '90s, many people were excited to learn about mapping or level design—thanks to a program called Worldcraft. Worldcraft was a free level editor that unleashed a creative surge from countless people.

In time Worldcraft was re-branded as Hammer. By now, it has been used to create a large collection of memorable and time-honored games. The list of games is too long to be listed here, but includes *Half-Life 2*, *Left 4 Dead 2*, *Counter-Strike: Global Offensive*, *Dear Esther*, *Black Mesa*, etc.

Hammer has had many merits. First, it is easy to learn. Its simplicity makes it accessible to many people dipping their toes into 3D design. Second, it has always been free for anyone who owns any of Valve's games made with the Source Game Engine.

That being said, Hammer is limited in many ways.

Many years ago I made a level for *Counter-Strike* that invoked a critique that I didn't understand at the time: “Those trees... Oh My God those trees.” I was too new at level design to understand the criticism. The trees were simply two cylinders, one narrow one for the trunk base and a larger one signifying the bulk of the tree. In today's world, they were about as good looking as trees in *Minecraft* (or worse).

What I did not yet know was that fine details (trees, plants, cars, etc) should be models instead of large generic shapes. Of course I didn't know the distinction yet. And this also begins a discussion that will remain an important aspect of this entire publication.

Hammer cannot create models as defined in the Source Engine. Instead, Hammer can only

create simple geometry like cubes, cylinders and other basic objects. And although those shapes can be manipulated somewhat, they must always remain convex. This excludes any kind of organic or complex shapes like humans, dogs or trees. Have you ever noticed how many crates (cubes) populate levels in community maps, or how often the same models are reused countless times? It has because of Hammer and the traditional complexity of the Source asset pipeline.

To accommodate the shortcomings of Hammer, the process has always been to create models (or props) in an external 3D application, such as 3ds Max, and place them into the Hammer scene. This has always been how it works for Source. This limitation in Hammer created, over time, an implied way that you are *supposed to* design levels and import props into Source. The limitations of Hammer created a false belief about how level design has to happen in Source.

There never was a technical reason in Source that levels and models had to be designed in different environments. The reason was always the scope of Hammer and its implementation. But the reality generated a consensus in the Source mapping world that this was not only the only way to do it, but also the only way you should do it.

Early on, I started getting into 3ds Max and its educational offspring Gmax. I was amazed with all of the design possibilities in 3ds Max, and I started looking for ways to use Max as my level editor instead of Hammer. But everywhere I looked, I found people saying things like, “You can’t do it.” Furthermore, there was a very strong sentiment from the mapping community that mapping for Source in Max is just dumb.

There has been a lot of confusion in the Source community about the nature of 3ds Max. Common definitions of 3ds Max are that it is a modeling program and not a level editor. Unlike Hammer, 3ds Max is an application designed to solve any 3D task. The point of this publication is to shed light on the topic of using 3ds Max as your Source Engine level editor. It is an important subject to me, as the Source Engine has proven to be a fascinating and personal

creative outlet, and 3ds Max is such an expansive artistic arsenal for generating any kind of art for Source, as well as movies, VFX, motion graphics and more.

The fact is this: you *can* use 3ds Max as your level editor for the Source Engine. The rest of this book is going to show you how... and by doing so I think it will become clear to you why 3ds Max is a much more powerful tool for your creativity than is Hammer. If you want to watch the process before reading, watch the [Wall Worm Level Design Start-to-Finish Series](#) currently being produced.

Who Should Read this Publication

This publication is intended for anyone who wants to create assets for the Source Game Engine. It is primarily directed at level design, environmental art and props. The publication is for both beginners and expert users. If you are already familiar with the basic 3ds Max user interface, you can skip Chapters 3-5.

What you Need

You'll need 3ds Max 2015+ (preferably 3ds Max 2018.4+) and Wall Worm 3.731+ in order to follow *all* of the lessons in this publication. You may be able to acquire a free educational license of 3ds Max from <http://students.autodesk.com>. You can get the latest version of Wall Worm for free at <http://dev.wallworm.com> or Wall Worm Pro for advanced features. Some of the other optional tools used in this book are referenced at the end, including Wall Worm's [CorVex](#) and [ShellVex](#) plugins.

You also need creativity and an open mind. Don't expect this book to answer every question or explain all aspects of Max or Wall Worm. You may need to experiment some or search the Max or Wall Worm documentation for some answers and definitions. Don't be afraid of using Google.

There are other third-party plugins that are referenced in this publication. Those plugins are not essential for the majority of this book. Some third-party plugins are referenced because I have found them to be extremely useful with my own projects and work flow.

About Source 2

This book is currently targeting the Source Engine, not Source 2. At this point in time, Valve has released the updated version of Hammer for some of its games and the Source 2 pipeline. Those updated tools are far more advanced and user-friendly than the Source Engine era of tools. At this time, this guide does not cover the methods, principles and work-flow of Source 2. When Source 2 is officially released, Wall Worm will likely be upgraded and expanded to accommodate the new tools. Depending on those changes, this guide will either be updated or a new guide written. Refer to the Wall Worm forums for user discussions on working with games like Dota 2.

About Wall Worm Pro

There are now two flavors of Wall Worm: the general free version and a commercial version called Wall Worm Pro. For the most part, this document details the general methods of using the free version of Wall Worm. You do not need WW Pro to export assets for Source—however, WW Pro does add extra functions and faster exporting.

Chapter 2 Preparing for 3ds Max

One of the most common challenges to anyone first entering the world of 3D design is the daunting complexity that is a 3D environment. Hammer can seem complex to a new user, and 3ds Max is infinitely more complex than Hammer. Even experts at Hammer can quickly become overwhelmed with the sea of endless options in a full-fledged 3D design tool like Max.

When coming to 3ds Max with a Hammer background, it is important to keep a few principles in mind:

1. You are not going to learn everything overnight.
2. The same principles of BSP level design apply in Max as they do in Hammer.
3. The way you approach the design is not the same as in Hammer.

You are not going to learn everything overnight

An enterprising person could probably learn every single part of the Hammer user interface in a single day. This is simply not possible in Max. I've spent a lot of time with Hammer users trying to use Max who suffer needless frustration that is based on an unrealistic expectation about learning Max too fast. Going from an environment where you know every single button to an environment where the tools seem endless can take anyone out of their comfort zone.

I think there are many paths to learning Max (like anything else). But if you feel overwhelmed, I think a good way to look at it is this. To start with, imagine that 3ds Max is like a souped-up Hammer editor. You still have primitives (brushes) that work just like they do in Hammer. The

Box primitive is the equivalent to the Block primitive in Hammer. Spheres are equivalent.

When starting with Max, you can limit yourself to the same kinds of objects you would make in Hammer. This might help you, initially, see how the tools are similar.

You don't need to know every aspect of Max to start using the same basic tools you'd use in Hammer, especially when it comes to layout. Start small and simple. Remake the first hollow cube level you probably started with in Hammer. Forget about the million tools in Max; focus on using a few basic primitives to create a frame of reference from which you can build on.

The same principles of BSP level design apply in Max as they do in Hammer

When using 3ds Max as a level editor for Source, all you have to remember is that the same principles apply between the two applications: all world geometry must remain convex; world geometry should contain no coplanar polygons; the world must be sealed; brush limits must be honored; and world geometry vertices should align to the world grid.

This is true of everything you make in Max *that you can make in Hammer*. So when setting up the *world geometry* and *entities*, you have to follow any *rule* that you would in Hammer.

The way you approach the design is not the same as in Hammer

Even though you will always need to follow the actual specifications for world geometry and entities that you would in Hammer, the way you approach design in 3ds Max should not follow the same pattern as in Hammer! *Keep your brushes convex, but don't limit your methods of making them to the Hammer methods.*

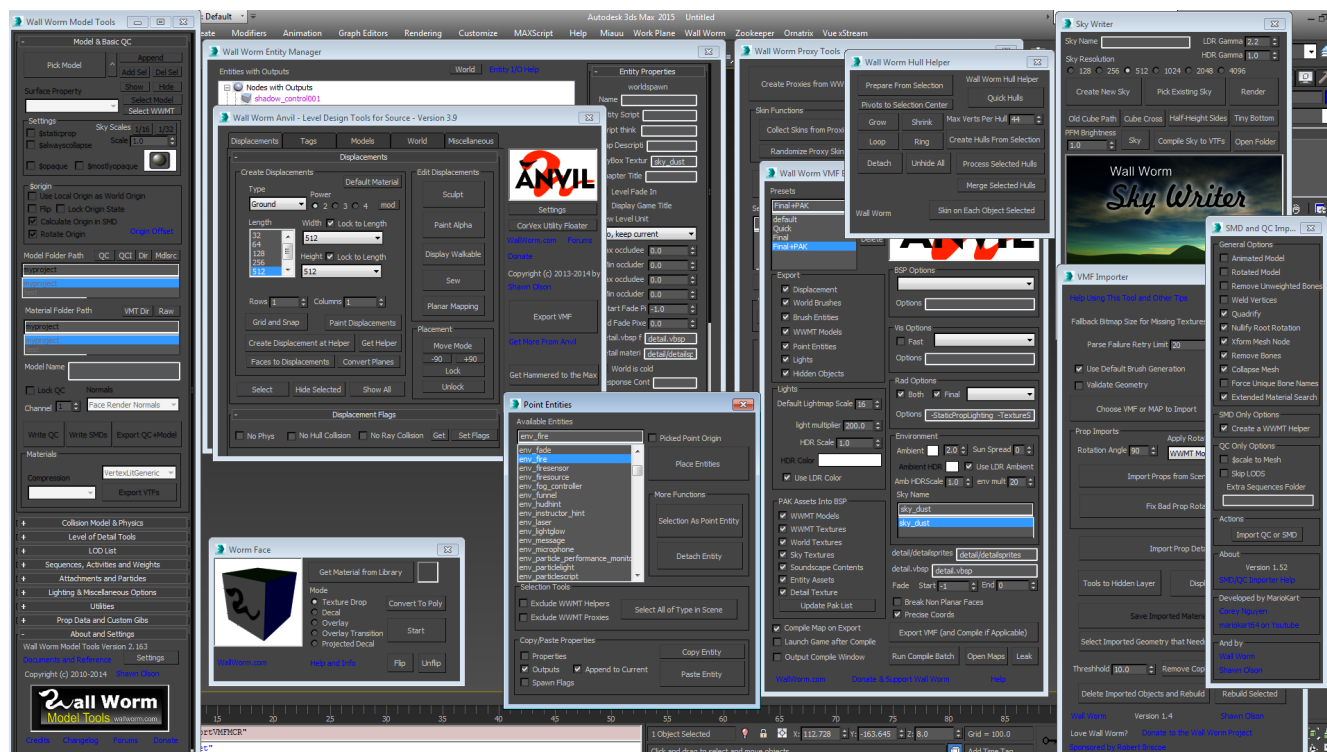
An example is Slicing World Geometry (or Clipping, as it is called in Hammer). While you will find times for slicing geometry in Max, you should not resort to this method first in Max. The reason is that there are better, more effective ways, at designing things in Max that just don't require slicing/clipping. More is discussed later in Chapter 5: Basic Editing Concepts.

One of the main things to understand about 3ds Max is that it offers many parametric design principles that just don't exist in Hammer; in 3ds Max you can control many things at once with simple controls. For example, you can use the height of one Box primitive to control the height of several other Box primitives—in this way, simply changing the height of one box will change the height of all others. Another example is how the CorVex plugin works: a single line or collection of lines can drive an entire layout system, and changing a single vertex on one of those lines can automatically rearrange the brushes all at once, or changing the CorVex width spinner will immediately make all walls in it increase or decrease width. Such an alteration in Hammer could be a catastrophically time-consuming task where the vertices must all be edited one by one across the objects.

Constant Innovation

What makes 3ds Max ideal for level design is that it has been a tried and tested 3D design environment that has been evolving for decades in several fields and industries, from movies to games. New tools are developed daily, both natively and by third parties. Design tasks can always be improved with innovative new approaches—whereas Hammer design principles have remained essentially static for over a decade.

Preparing for 3ds Max



When it comes to Wall Worm, new tools are often added weekly. This has been true for several years now, and may continue for several more. If there is some work-flow that could speed up producing content for Source, you will likely see it in Wall Worm. And if you have an idea, Wall Worm is open to all suggestions from artists and developers.

Level designers have been clamoring for a better Hammer editor for a long time... but wait no longer—it's already here inside 3ds Max. So let's get started!

Who is Using Wall Worm?

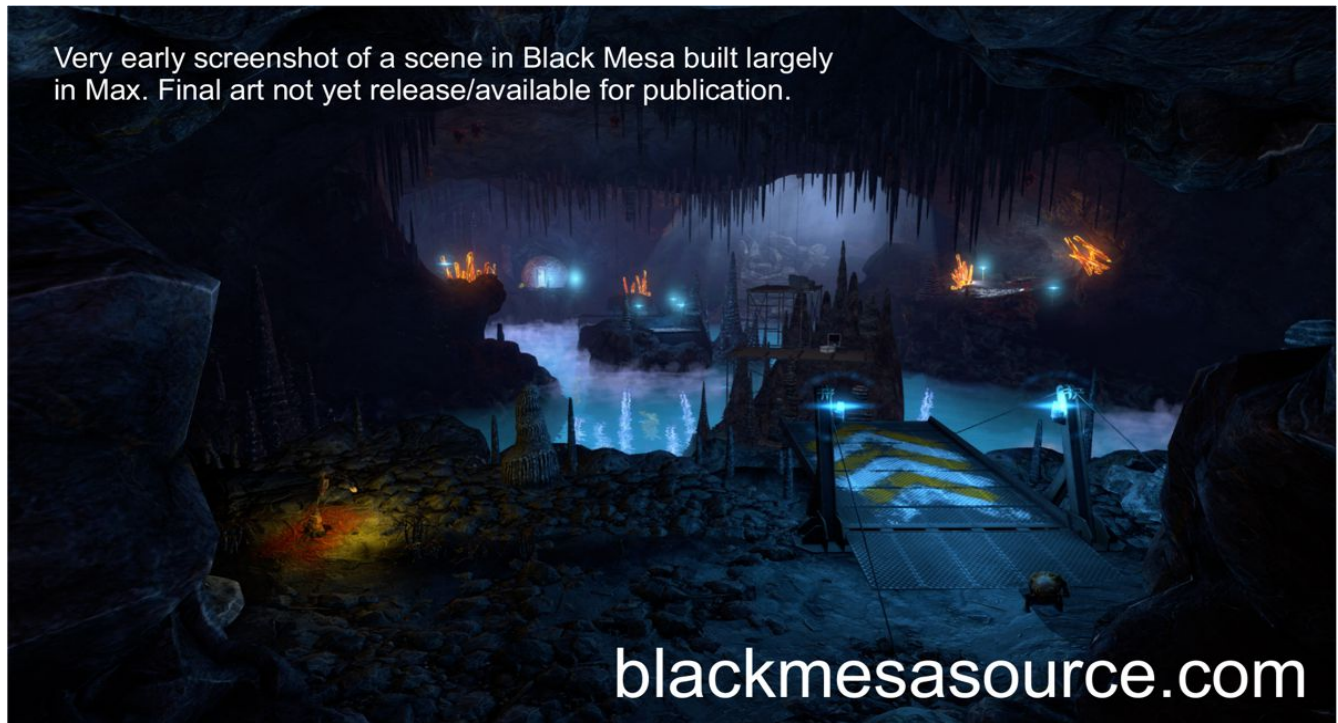
Before you dive into using Wall Worm, you are probably wondering who else is using it and what they have done with it. Wall Worm tools have been used by many thousands of artists since 2010. In large part, this has primarily been in the model creation pipeline. To this point,

the number of artists and studios using the level design tools as a replacement for Hammer has been limited (which is part of why this document was created—to help spread better information on how to use Max as a Source Level Editor). Most users have only been aware of the modeling side.

To varying degrees, Wall Worm's level design tools have participated in several Source projects that you may know of. Robert Briscoe commissioned some of the detail sprite and VMF importers in Wall Worm for his port of *Dear Esther* to Unity. Minh Le, co-creator of *Counter-Strike*, said, “I used WW extensively when I worked on maps for *Tactical Intervention*. In fact, when I was making our highway driving map, WW saved me countless hours.” Since first publishing this book, I have started working for *Black Mesa*. Enhancing and streamlining the displacement work-flow is part of my chore in some of the final phases of bringing that game to release. Since 2015 a large percentage of the updates and improvements in Wall Worm have been level design related tools built for the Black Mesa team to bring Xen to life.



And now I've started some level design contests on the Wall Worm forums that require users to use Max entirely to win commercial prizes. To the right are examples of BSP-only levels created entirely inside Max that were part of the first Old School Level Design Contest that challenged artists by not allowing models and displacements.



For the latest contests and prizes, see the Angry Teapot Level Design Awards at <http://angryteapot.com>. There you can participate in contests sponsored by some of the most prominent 3D plugin developers in the world; sponsors have included Exlevel, Boomer Labs, nPower Software, Ephere, enRichPro, JokerMartini, Miauu, Monotone Minimal; game sponsors include Black Mesa and Aperture Tag. Winners get commercial licenses to powerful plugins.

Chapter 3 Introduction to The 3ds Max User Interface

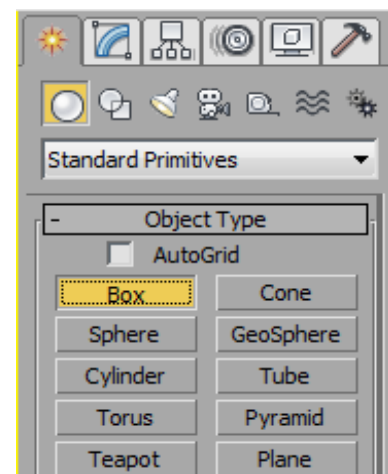
While this chapter gives some of the basics about Max, it is limited to broad aspects that are important to know and that seem to be commonly needed bits of information for new Max users coming from Hammer. The best resource for brand new users is probably this [Autodesk Youtube Channel on being New to 3ds Max](#). *Be aware that 3ds Max 2017+ has a new icon set that is different than those in most existing videos and documentation (including this book).*

Like in Hammer, 3ds Max has a default 4-view layout: top, left, front and perspective. Toolbars (containing buttons) can be customized and docked anywhere in the window. Most of the buttons in toolbars have tooltips when you hover your mouse over them—and some have different functions depending on whether you left-click or right-click. Any button with a little arrow in the bottom-right corner can be held down to produce extra options.

An important part of the Max UI is the command panel. In the default Max UI, the command panel is usually docked on the right side of the window. The command panel is a logical partition of functions divided into six tabs; you can hover over each tab to get its title. The three tabs you will likely use most for level design are Create, Modify and Display.

Create Tab

You can think of the Create Panel as your warehouse. It is where all your goods are stored: Boxes, Cones, Lights, Cameras, Shapes, etc. Whenever you want to add something new into your scene, this is where you go.

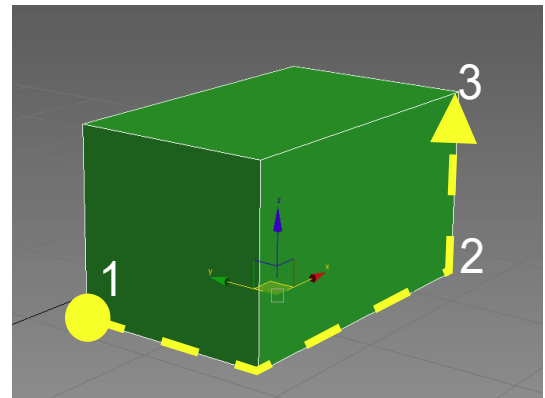


When you open the create panel, you'll notice that it has icons that represent different major groups of object types. All geometry objects are located in the Geometry group. You can tell which group is currently active by the highlight color behind the active group. To make a light in the scene, choose the Light button and pick the kind of light you want to add, etc.

Your First Box

Open the create panel in the command panel and click the Box button. Now, when you mouse over the viewport windows, you'll notice that the cursor is a cross hair. This indicates that you can create a box object, starting at the point you first click.

If you click and let go, nothing happens. But if you keep the button down and drag (from point 1 to 2 in image), you'll see the base of your box match the area between the initial click and current location. Let go of the mouse and start moving the mouse up or down—you'll see the box height change as you move the mouse (point 3 in image).



Now click again. This click ends the creation process of this box.

You can immediately start clicking new locations to continue adding boxes.

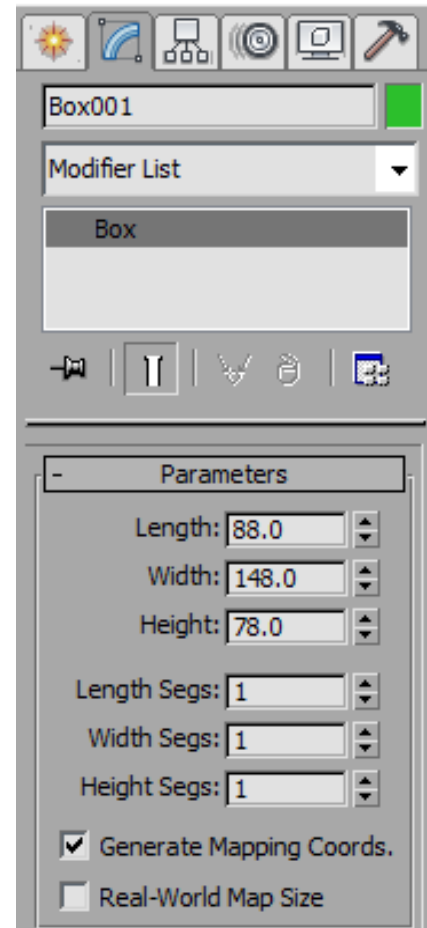
During the creation process of any object in Max, right-clicking cancels the creation. In fact, right-click is the cancel for many functions in Max.

Modify Tab

The modify tab is a powerful tool inside Max. It allows you to change the parameters available to an object and to work with the Modifier Stack. The modify tab only works if you have one or more objects selected in the scene. While the modifier stack is available when multiple objects are selected, the modify parameters are generally only available if a single object is selected in the scene.

Modify Your Box

Select the Box you created above and open the Modify tab of the command panel. Notice that you can change the values of this Box primitive, including all of its dimensions and some other parameters. If you click the height spinner and drag up, notice the box get taller in the viewport. You can type an exact value in here.



As long as this box remains a Box primitive, you will always have access to these basic parameters. What does that mean? Well, we can collapse any object into a polymesh object, at which point we will lose access to these parameters and must edit the object via its sub-elements, like vertices, faces, edges, etc. We will return to this subject later.

Display Tab

The Display Tab is a general visibility tool to quickly show/hide classes of objects in the

Navigation

Your best bet to learn the navigation is Max is to watch this [Autodesk Video on Navigating Scenes](#). Note that the video above is using 3ds Max 2011, and later versions have some slightly different default options (and some Mouse controls are now configured in the Mouse tab of Customize User Interface.)

40

Flying Through the Scene

To fly through the scene (as you would in Hammer by pressing **Z**) is the Walk Through Mode. You can activate this mode by pressing the **UP ARROW** key or the Footstep Icon in the Navigation tools (see above graphic). When active, the **ASDW** keys will propel you through 3D space and holding the left mouse button while moving the mouse will alter your view angle. To increase the speed of movement, hit the **]** (right bracket) key; to decrease speed, hit the **[** (left bracket) key. Right-click the viewport to exit Walk Through Mode.

Zoom Extents Selected

Click the **Z** key in Max to run the Zoom Extents Selected command. This will force the currently active viewport to zoom to a level such that the currently selected object(s) will fill up the entire view. If no objects are selected, the zoom level will make the entire visible scene fit in the view.

This command also resets the zoom scale, which affects how the mouse-scroll zoom works.

Zooming

While you can use the scroll wheel on your mouse for scrolling, you may also find that the **Ctrl+Alt+MMB** (Middle Mouse Button) will give you a smoother zooming experience.

Orbit Selected

To Orbit around the currently selected object(s), hold down the **Alt+MMB** while moving the mouse. If you have 3ds Max 2017+ installed, you can utilize the Orbit Point of Interest mode

that allows you to select orbit points in the viewport without changing the selection. See [Orbit Point of Interest documentation](#).

Getting Familiar with the UI

Getting comfortable with the Max user interface is probably not something that will happen in a single day with most users. Becoming familiar with various shortcuts will likely come down to you finding a general work flow that accommodates your own style and preferences—those things you do most you'll likely search for shortcuts. Finding your personal method can only come with experience, and this book does not intend to tell you how everything should be done. The purpose here is to give you a framework. Be aware that the UI/Icons in Max may change from version to version. As noted earlier, the icons in 3ds Max 2017+ are very different than those in previous versions of Max.

Almost every command can be assigned to a shortcut key on your keyboard. There are already many default shortcuts. Those functions that are available for assigning to buttons and keyboard shortcuts are available through special scripts called **MacroScripts**. These are command sequences that have been made available to assigning to the UI. If there is a command you wish to add to the keyboard shortcuts or to the right-click menu, all you need to do is click **Customize > Customize User Interface** and browse for the appropriate category. All macrosript functions provided by Wall Worm are located under the **wallworm.com category**.

This book won't explain how to make your own MacroScripts; understanding what they are and how to make your own will greatly enhance your capabilities. You can open the MAXScript listener (F11) and simply watch commands happen as you do actions in Max. You can actually select a section of previous actions and drag that sequence of action history into a brand new button that will repeat those commands any time you press it. See the documentation on the [Macro Recorder](#) for more information. See the MAXScript documentation for

Introduction to The 3ds Max User Interface

more advanced topics on making your own functions.

One final piece of introductory advice: learn all the names and functions of all buttons that appear in the default Max UI, including the main toolbar at the top and the various utilities and navigation tools across the bottom of the Max window. Becoming familiar with these are essential to becoming familiar with Max. Some of the main items in the main toolbar are discussed in the next chapter.

Hammer and Max Function Chart

This chart is not an exhaustive list of functions for either application. But it is a list of common Hammer shortcuts and the Max/WW equivalents. Items with an asterisk can be assigned to shortcuts with the wallworm.com category in Customize > Customize User Interface.

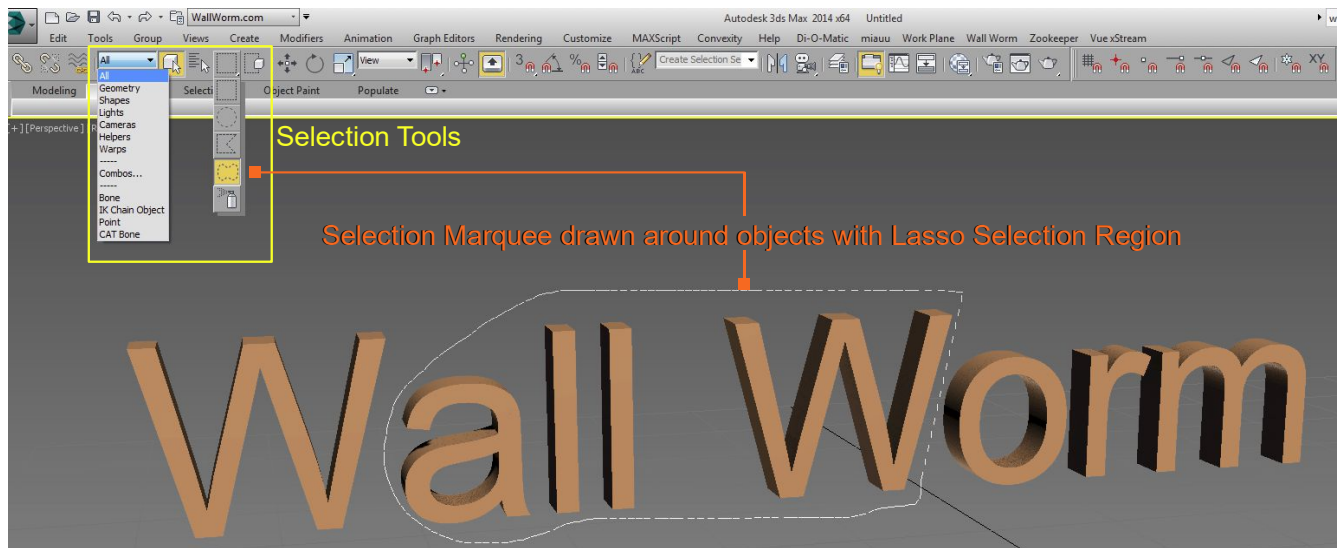
Function	Hammer Shortcut	Hammer Menu	Max Shortcut	Max Menu
Transformations				
Select	SHIFT + S		Q	
Select and Move			W	
Select and Rotate			E	
Select and Scale			R	
Transform Dialog	CONTROL + M		F12 / Right-click Transform Icons in Menu	Edit > Transform Toolbox
Nudging	SHIFT+ARROW Keys		*	Wall Worm > Wall Worm Extras > Nudge UI
Viewports				
Maximize/Restore Active Viewport	SHIFT + Z		ALT + W	
Pan View	Arrow Keys		MMB + Mouse Move	
Zoom All Viewports	CONTROL + A		CONTROL + SHIFT + Z	
Zoom Current Viewport to Selection	CONTROL + SHIFT + E		Z	
Top View	F2		T	
Side View	F3		L	
Front View	F4		F	
Bottom View			B	
Perspective View	F5 / SHIFT + F5		P	
Quick Hide Unselected (Isolate Selection)	CONTROL + H		ALT+Q	
Grids and Snapping				
Show/Hide Grid (in active viewport)	SHIFT + R		G	
Toggle Snapping	SHIFT + W		S	
Toggle Grid Snapping	SHIFT + W		ALT + F11	
Decrease Grid Spacing	[*	Tools > Grids and Snaps > Grid and Snap Settings > Home Grid
Increase Grid Spacing]		*	Tools > Grids and Snaps > Grid and Snap Settings > Home Grid
Entities				
Tie to Entity	CONTROL + T	Tools > Tie to Entity	*	Wall Worm > Wall Worm Level Design > Point Entities Brush Entities
Move to World	CONTROL + SHIFT + W	Tools > Move to World	*	Wall Worm > Wall Worm Level Design > Move to World
Edit Entity Properties	ALT + ENTER	Edit > Properties	*	Select > Modify Tab
Open Find Entities Dialog	CONTROL + SHIFT + F			Wall Worm > Wall Worm Level Design > Entity Manager
Camera/Navigation				
Create Camera	SHIFT + LMB + DRAG		CONTROL + C	Create > Cameras
Camera Tool (to navigate via camera)	SHIFT + C / Z		UPARROW	
Miscellaneous				
Go to Brush Number	CONTROL + SHIFT + G		*	Wall Worm > Wall Worm Level Design > Wall Worm Map Compile Tools > Go to Brush Number
Carve	CONTROL + SHIFT + C	Tools > Carve		
Make Hollow		Tools > Make Hollow		Wall Worm > Wall Worm Level Design > ShellVex: Create Corridor

Chapter 4 Essential Tools

Max has many tools to accomplish almost any kind of task, and those tasks that Max can't accomplish by default can usually be accomplished with Scripts and Plugins. For now we are going to stick with the basic, vanilla Max.

Main Toolbar

Now that you have a basic idea of how to make and edit a box, it is time to talk about the main buttons present in the Main Toolbar that is at the top of the Max viewport by default. This toolbar contains access to all of the essential functions common to all the objects you will create.



In the main toolbar are your tools for selecting objects, transforming objects (moving, rotating, scaling), toggling various snap modes, aligning objects and more. You should familiarize yourself with these buttons, as they are generally the kinds of commands that you will use every time

you sit down to design or edit a scene. Selection Tools

The selection tools in Max are very powerful, far more robust than you are used to if you have only used Hammer. The **Select Object** button is the button with the white box and a pointer or the shortcut **Q**. You can select individual objects with a single click of the **Left Mouse Button (LMB)**; you can add to the current selection with **Ctrl+LMB**; you can remove individual objects from the selection with **Alt+LMB**. And to select a bunch of objects at once, click and drag in the viewport.

The image above shows some of these tools. The first drop-down menu that defaults to “All” is your Selection Filter. It allows you to limit the selection tools to specific kinds of objects in the scene. For example, if you only want to select **Shape** objects (2D splines) then you select shapes in the Selection Filter and you can only select shapes in the scene; this helps you from accidentally selecting objects you don't want to include in an action.

The dotted **Rectangular Selection Region** button is the default region type. Clicking **Q** will cycle through the available region options. You can also hold down the Selection Region button to get a drop-down of the available regions. In the previous image, you see the **Lasso Selection Region** highlighted. That option allows you to draw the selection boundary, which will select the encompassed objects that match the selection filter type. In the image, this would select the four individual objects that are surrounded.

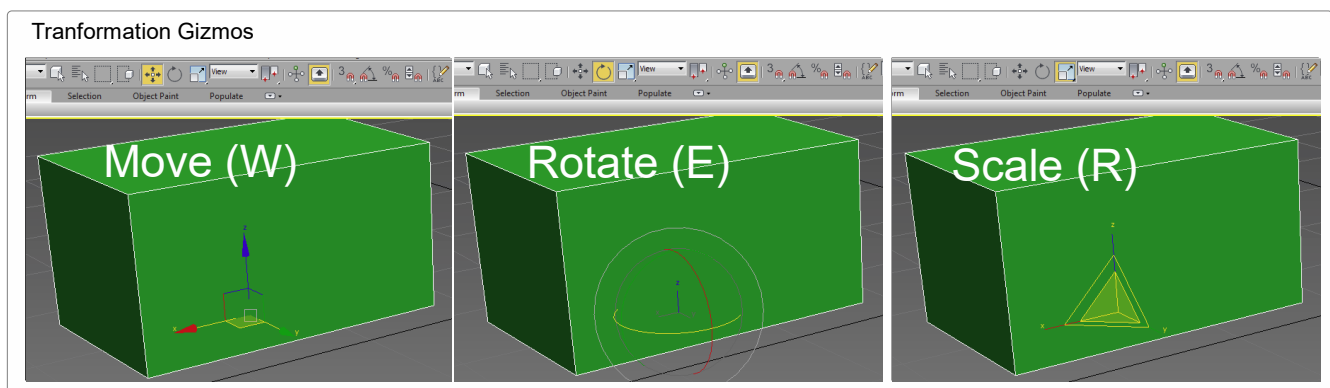
Transformation Tools

The transformation tools are for moving, rotating and scaling your objects in the scene. From the multitude of people I've mentored, I know that these tools are a challenge for many Hammer users. Although the difference can be frustrating the first time you open Max, it will not take you long to realize how much better these tools are inside 3ds Max.



These buttons from left-to-right are **Select and Move (W)**, **Select and Rotate (E)**, **Select and Scale (R)**, **Reference Coordinate System** and **Pivot Point Options**.

You should get used to the **Move (W)** and **Rotate (E)** shortcuts. But before you add the **Scale (R)** shortcut, take heed to this very important warning. *Do not scale objects at the object level!* What does that mean, exactly? It means that you should general not select an object in the scene and scale it. Instead, you should scale it at sub-object level (vertices, edges, polygons and elements). The reason behind this is beyond the scope of this chapter, but actually implementing this policy is discussed in the next chapter on basic editing.



Manipulating the transform of your objects can be accomplished visually via transform gizmos or manually with type-in tools. *When using the gizmos, you will be constrained by any relevant snap that is turned on.* Snaps are discussed a little later, but this is important to realize. For example, when you have Snap to Grid enabled and are in the perspective view, you may have trouble dragging an object in the Z axis if the grid spacing is too small—because the object will keep snapping down to the ground level. Understanding snaps is fundamental to effectively transforming objects in the Max viewport!

While in any of the transformation modes above, type-in spinners activate at the bottom of the window. This will display the current values of the position, rotation or scale (depending on the

mode) and allow you to manually change the values. Note that the values and orientation of the gizmos are determined by the current working coordinate system. This is an important feature of 3ds Max that is not relevant to the Hammer environment. In Hammer, all transformations are in World Space. But 3ds Max offers an arbitrary number of coordinate systems.

Coordinate Systems

Coordinate systems are integral to multiple fields, including physics, mathematics, 3D animation and engineering. For level design, you don't necessarily need to get too deep into them. But you need to be aware of what a coordinate system is and why you might use different systems.

If you watched the movie *Gravity* or *Ender's Game*, you may have noticed the topic of coordinate systems because the characters experienced some psychological trauma in space that is not natural to a human on the ground. Humans in a strong gravitational field have an innate sense of a special coordinate system where we always know the World's up and down. We don't have that in space, which is why the characters in the movies had moments of panic.

In terms of 3D, your environment is called the World. The world defines the coordinate system of Up, Down, Left, Right, Forward and Back. All of those directions are calculated in three dimensions of space, which is where the X, Y and Z come in. The Origin of the World is the center of the universe, so-to-speak. Anything at the world's origin is at a location of [0,0,0] *in world space*.

In the real universe, there is no meaningful, absolute origin. That means that a coordinate system is always arbitrary. This is also *mostly* true in 3ds Max*. If you rotate an object in 3ds Max 90 degrees around the Z axis, it has a rotation *in world space* of 90 degrees. But in *local space*, it has no rotation!

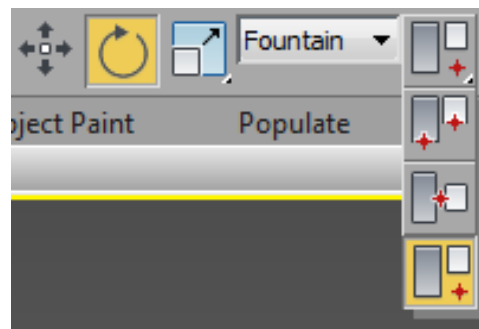
****Mostly** true because, in the end, all transformations are stored in world coordinates. This means that in Max there actually is a mathematical origin that is absolute—however, this is not the only frame of reference that you will use.*

Think of it like this: if you walk into the room and turn to the right 90 degrees, people looking at you will think of you as having oriented your body in a new angle. But from your viewpoint (in terms of what is directly in front of you), what front or back *is*, in your own definition, has not changed—instead, *the things that occupy your front and back have changed*. That is because you view the world from your own local coordinate system.

This has a lot of implications for you because it can simplify some problems. You can choose to move an object a specific number of units in world space if that is best, or you can move an object in its local space if that is best, or any other arbitrary coordinate system.

Let's use a practical example. Imagine you have a scene with a fountain and a bench at some arbitrary location in the scene where the bench faces your fountain. What if you want to move and rotate the bench so that it is exactly on the opposite side of the fountain. Doing this in Hammer would require you to calculate distances and rotations, which could be tedious. In 3ds

Max, all you do is set the fountain as the coordinate system, set the **Use Center** flyout to **Use Transform Coordinates Center**, and type 180 into the Z axis of the transform type in.



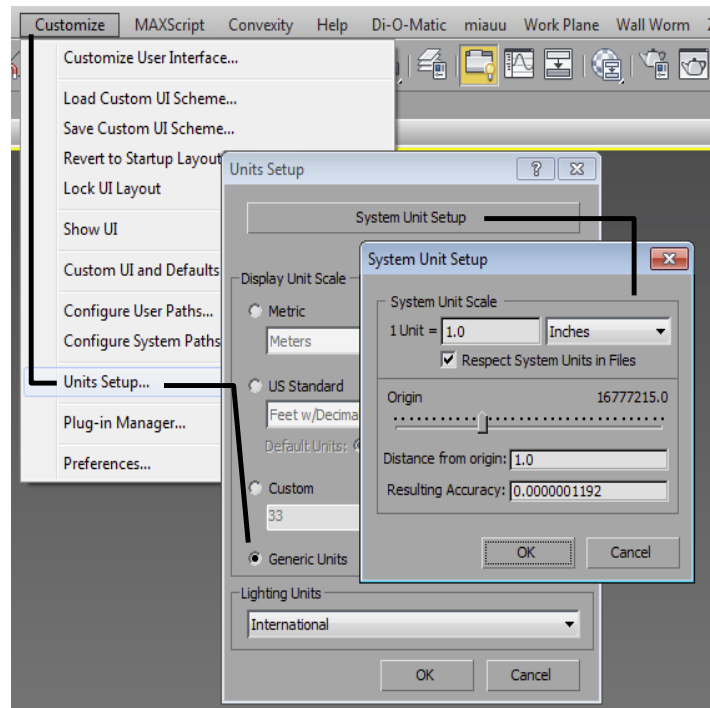
Understanding the nature of coordinate systems can be helpful. The more you delve into animating models, the more important coordinate systems become. For now, be aware of them to help simplify transformation problems. When your problem is solved, you may want to make a habit of falling back to World coordinates, especially for your world geometry and layout phases.

The Grid and Snaps

One of the most common misconceptions in the Source modding community is that you shouldn't use 3ds Max for level design because it is too easy to go off the grid. This is, again, rooted in the traditional pipeline where only models were made in 3ds Max—and models do not need all their vertices aligned to the world grid. But this is something that needs emphasis: **Just because you can make off-grid models in max, it does not mean you can't make on-grid geometry in 3ds max.** For a deeper discussion and video tutorials on this, see [On the Grid](#).

The Grid and System Units

The grid is like virtual graph paper that marks of distances in an any two axis of a coordinate system. The grid in Hammer is equivalent to the **Home Grid** in 3ds Max, which is a grid aligned to the world. Measurements on this grid are determined, in Max, by the current **Unit and Scale**. Display properties of the **Home Grid** are in the **Home Grid** tab of the **Grid and Snap Settings** floater, which you can access by clicking **Tools > Grid and Snaps > Grid and Snap Settings...** or by right-clicking any of the snap toggles. The **Home Grid** options are very similar to the 2D grid options in Hammer.



You should set the Grid Spacing in the Home Grid tab to a power of 2. For initial layout, you may find larger values like 128, 256 more convenient. At the bottom of the Max window to the

right of the transform type-in boxes you will always see the scene's current Home Grid spacing.

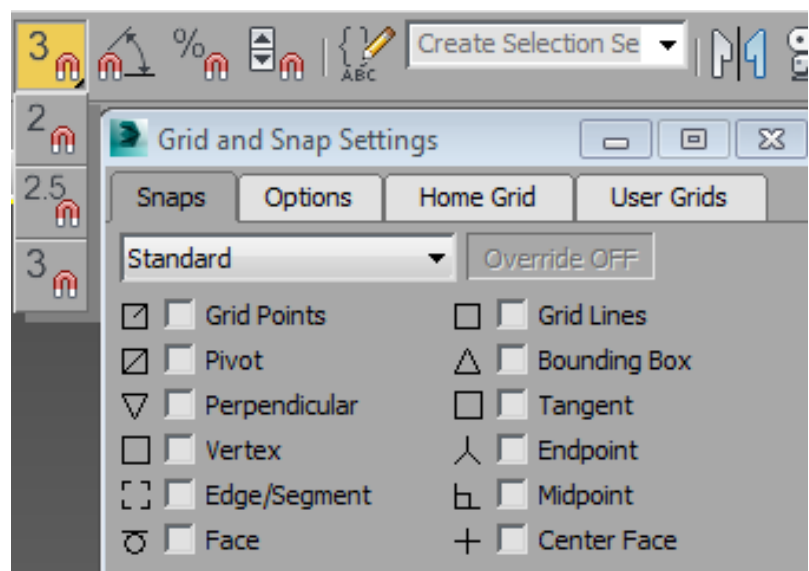
64	Large crate
128	Standard wall height
256	Good for initial blocking with CorVex
512	Good for initial blocking with large outdoor scenes

As noted earlier, the scale of the Home Grid is determined by the System Units. When working in 3ds Max for the Source Game Engine, for both level design and prop design, you should always set the system to use **Generic Units** where **1 Unit = 1.0 Inches**. You can change the units under **Customize > Units Setup...** as demonstrated in the graphic to the right.

If you do not use this scale, you will find that your geometry and models will be either too big or too small once you send them into the game.

Standard Snaps

There are four categories of snaps in the Max toolbar. The first is the **Standard Snap**, which allows you to snap elements of objects to other elements in the scene. These snaps are ex-



tremely valuable tools when manipulating object or sub-object positions.

Generally speaking, you should always start your world geometry layout with **Grid Points** and **Snap** turned on. Once your scene populates with geometry, other snap points become more relevant. The two snap points that should be used almost exclusively for world geometry are **Grid Points** and **Vertex**. By sticking with these two snaps, you can generally assure that all your vertices will remain on the grid. Later, when building and distributing props, you will find the other snap points more useful.

For a reference of all snap points, see here: <http://docs.autodesk.com/3DSMAX/16/ENU/3ds-Max-Help/files/GUID-C18F1E36-4FFC-43CF-A811-0ED6D5545C40.htm>. As much as possible, you should make a habit of visiting the 3ds Max documentation to understand features about which you have questions.

To manage the current active snaps, you can right-click the Snaps toggle and use the **Snap** tab of the **Grid**



and **Snap Settings** floater. You can also activate the **Snap Toolbar** by right-clicking the border of your main toolbar and checking the **Snap** option... which will add the snaps toolbar. The snaps toolbar will show you some of the most common snaps as well as their current state (on or off).

Understanding how to use the snaps in 3ds Max is extremely important. As such, you should take a few minutes to watch this [Youtube Video on Using the Snaps Tools by Autodesk](#). Understanding the interface of snaps will allow you to create, move and align elements of your scene with a level of precision and speed that is simply far beyond the capabilities of Hammer. Watch the video linked above to see how you can use the grid and vertex snapping in a way that is perfect for your level design tasks.

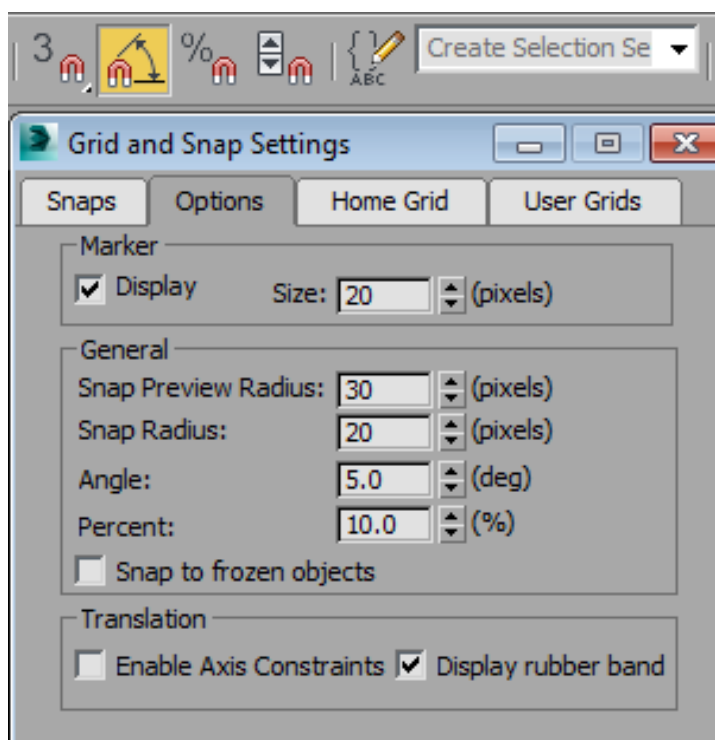
Angle, Percent and Spinner Snaps

The **Angle Snap** allows you to force the rotation gizmo to snap to increments designated by the general **Angle Snap** setting. For example, if you set the Angle Snap to 45 degrees, your rotations will always jump by 45 degree increments as you rotate an object in an axis with the rotation gizmo.

The **Percent Snap** forces the scale gizmo to increment at the set percent in an axis as you scale.

Both the angle and percent settings can be accessed by right-clicking the **Angle Snap** or **Percent Snap** in the main toolbar.

The **Spinner Snap** controls the increment snapping in parameter spinners in the command panel and other floaters. The spinner increment is controlled in the general tab of the **Preferences** floater, unlike the other snapping functions. You can **right-click** the **Spinner Snap** toggle button to access that menu.



Other Main Toolbar Buttons

The other items in the main toolbar are also important tools. You'll find **Selection Sets** a powerful way of customizing arbitrary sets of objects for easy selection; **Alignment** functions that

give you fast and powerful ways to transform and orient objects; the **Layer Explorer**, **Graph Editors**, **Material Editors** and **Rendering** functions. All of these play important roles in creating new worlds. There will be a chapter on materials later in this book. For the other options, you should refer to the on-line documentation for 3ds Max. You can always access this by clicking F1 while hovering over a button, menu or feature in Max (like in many other programs). Some items will go straight to the help on the button under the cursor.

Chapter 5 Basic Editing Concepts

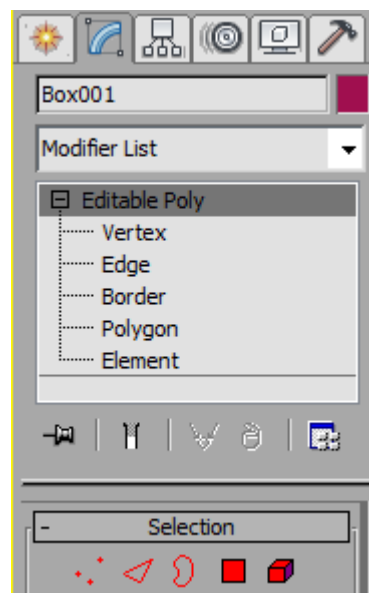
In 3ds Max, there are different types of parameters for different types of nodes. Some types of parameters are common to all objects—for example, the position in the world or visibility. Other parameters are dependent on the type of object. For example, a Box primitive has a height parameter while an Omni light doesn't; conversely, an Omni light has a light color parameter, whereas the Box doesn't.

Knowing what parameters are available to what objects is generally a result of experience. For the rest of this book, we will generally discuss objects that are called Editable Poly objects. These objects are similar in nature to brushes in Hammer, although much easier to manipulate. Other objects are valid for Source, but, for simplicity, this discussion will focus on the Editable Poly.

Editable Poly Objects

All geometry objects can be converted to an Editable Poly. You have two options for working with your object as an Editable Poly: convert it to an Editable Poly, or add an Edit Poly modifier. Both have pros and cons. You can experiment with using the modifier method, but for simplicity, we are going to discuss, at present, collapsing to Editable Poly only.

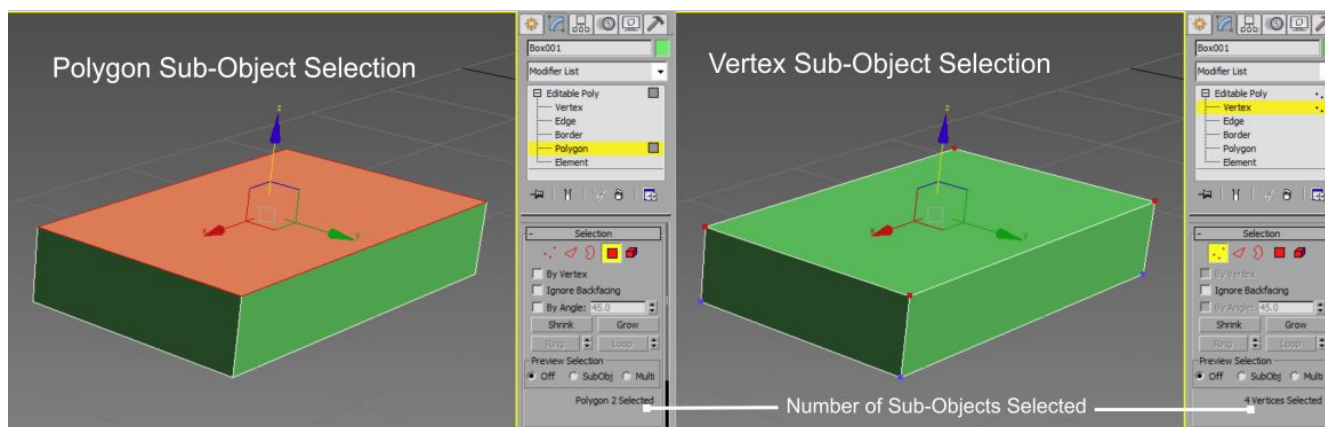
To convert a Box primitive into a an Editable Poly, right-click the box in the viewport and choose **Convert To... > Convert to Editable Poly**. Do not use any of the other types



available in the convert menu.

Now, select the Box and switch the Command Panel to the Modify tab. Notice that the parameters for height, width and length are all gone. Instead, there is a very long list of rollouts with a lot of buttons, settings and options. Welcome to the glorious world of the Editable Poly.

Notice the little plus (+) icon to the left of the Editable Poly label in the modifier list. If you click that, you see all of the available Sub-Object levels: Vertex, Edge, Border, Polygon and Element. If you select any one of those, you can edit the object at that specific level. For example, if you click the Vertex sub-object level, you can manipulate the object in the scene the same way that you would manipulate an object in Hammer with the Vertex Manipulation tool.

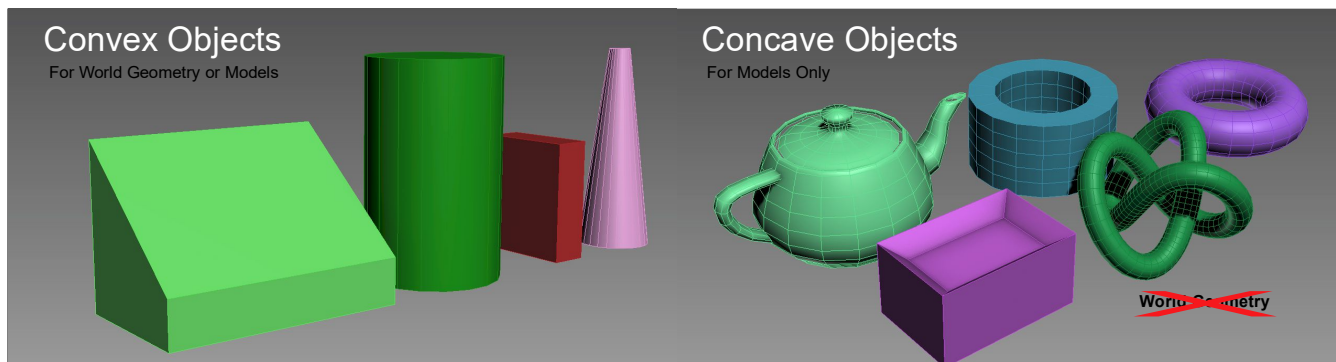


In Max, you have more convenient tools for manipulating your geometry. For example, instead of selecting the four vertices that make up a side of the box, you can click the Polygon sub-object mode, pick a single side, and then move that side around instead of four vertices.

The sub-objects can be transformed just like the objects with the transform tools. Above, the image shows a box where a single polygon (highlighted in red) could be dragged in 3D space to move that polygon. The right image is the same object in vertex sub-object mode with the vertices selected (in red) that make up the same polygon as in the left image.

Basic Editing Concepts

You can go back and forth between the different modes to modify your object however you want. However, you must always keep in mind the kind of object you are making. If you are making a prop (that will be converted to a model), you can be fairly carefree in the manipulation of vertices, etc. *For world geometry, however, you must always remember to keep each element of your object convex!* Any world geometry object that is not convex will have an error when the scene is sent to the level compiler.



Each of the different sub-object modes has its own set of tools. Some of the options are shared between different sub-objects, some are not. For example, in Polygon sub-element mode there is a function called Outline that allows you to scale the selected Polygon's in a coordinate system based on the polygon's face normal. This function is meaningless in other modes. Vertex, Edge and Border modes have a function called Chamfer that is not available to the Polygon mode.

Becoming familiar with these different sub-object modes and the tools in each will help you design and model new objects. At this level, there is no fundamental difference between the tools you use for making brushes and models; the only difference is understanding the limitations you must impose on anything that is world geometry by keeping it convex.

There are many shortcut features in Max for quickly selecting common sections of objects. For example, in Polygon sub-object mode, you can select a loop by selecting a quad polygon, holding down **CTRL**, and clicking an adjacent quad polygon. Another useful shortcut is in

Edge sub-object mode—if you double-click an edge, all other edges that form a continuous loop will get selected. Referring to the documentation will help you learn all of the selection shortcuts.

Slicing Objects (Clipping Objects)

In Max there are two general ways for slicing objects like you do in Hammer with the clipping tool. Inside an Editable Poly, you can use the built-in Slice functions. If slicing multiple objects in a scene (or non-Editable Poly objects), you can use a Slice Modifier.

To slice similarly to Hammer, go to the Element sub-object mode and scroll down to the Edit Geometry rollout in the Modify tab. Look for a button labeled QuickSlice. Click that button and make sure the Split check box is enabled right above the QuickSlice button. Making sure that you have the appropriate snaps (snap to grid or snap to vertex), click and drag the pointer from the start point all the way to the end point of your slice. After you are done, go to Border sub-Object mode, select all borders (CTR+A) and hit the Cap button in the modify tab.

If you need to slice the object multiple times, you should cap the borders between each slice that happens to cross an already cut opening. Failure to do this will create Non-Planar Polygons since the Cap function creates a single polygon from each contiguous border—even if it's non-planar.

While using the slice functions is similar to the Clipping Tool in Hammer, you should rethink your design strategy if you find that you are needing to slice world geometry too much. Max offers better approaches to designing your world geometry that do not often require the Slice tool. Also, many of the tasks where you might use the Hammer clipping tool are simply easier with Editable Poly functions. For example, instead of clipping away and deleting the tops of a wall to create a beveled effect, you can select the edges and use the Chamfer tool or select the top polygons and use the Bevel function.

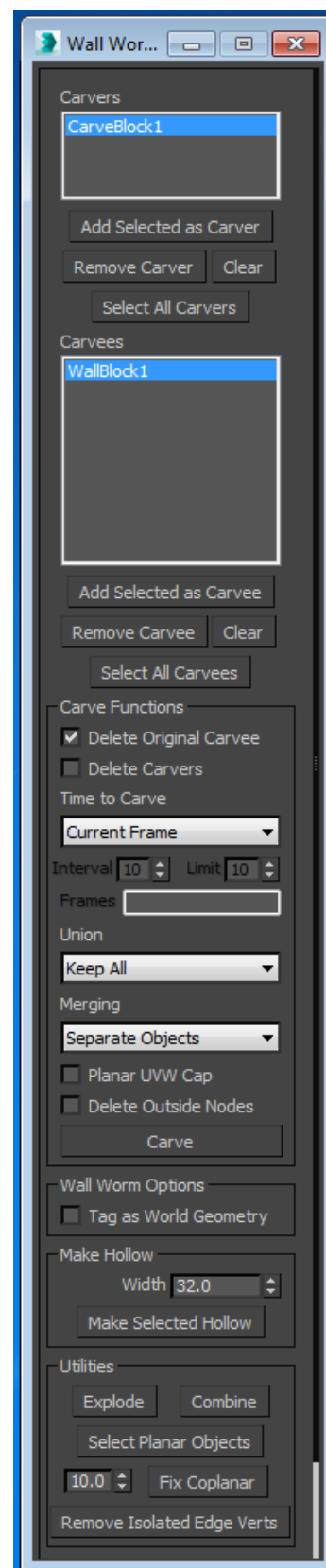
For geometry you are cutting up to “model”, you should simply abandon this principle and realize that you can simply create actual models in the scene. This does not imply that you should never use the Slice functions in Max, as Slice is a very effective tool. The point is that, unlike in Hammer (where clipping is really a requirement for many situations), Max offers multiple more appropriate tools for varying situations. Knowing when to use the correct tool is a result of experience.

Carve and Make Hollow

3ds Max and the free Wall Worm tools do not have an equivalent to the Carve function in Hammer nor the Make Hollow function. Although the Carve function is often frowned upon in the Hammer communities, it does sometimes have a utility.

To utilize similar tools in 3ds Max, you need to get the commercial (but inexpensive) [Carver](#) tool also produced by Wall Worm. Carver gives more control over your results than the similar function in Hammer. It also has a function to quickly fix [coplanar polygons](#), which is valuable for cleaning up world geometry. Visit the link above to learn more about Carver.

For the most part, the functions in Carver are one-off functions. This means you run the functions and the object is modified and collapsed to editable poly. There are other tools in the Wall Worm tool bag that can often solve some solutions non-destructively. One example is instead of using the Make Hollow function in Carver, you can use [ShellVex](#) to generate brushes around oth-



er geometry without collapsing the geometry. Also, the [Brushify Modifier](#) has tools for removing coplanar polygons without collapsing the geometry.

Please note that there is a difference between using Carver and the ProBoolean/Boolean functions inside Max. Some readers have misunderstood how Carver works and suggest using ProBoolean instead. The fact is that ProBoolean does not break up objects into convex sub-pieces. ProBoolean is an effective tool for many cases, but if you intend to break up *brush geometry* with ProBoolean, you will find that you have a lot of work ahead to make the geometry valid as *brushes*.

Chapter 6 Welcome to Wall Worm

Wall Worm is the 3ds Max-to-Source pipeline. It is a collection of tools, scripts and plugins that make your life easier. The bulk of the tools are for getting assets from Max into Source, but there are also a few tools for getting assets from Source into Max.

A Brief History of Wall Worm

When I started my journey into making levels for Goldsource (The Source Engine predecessor), I was only building levels in Worldcraft. I was already experimenting with 3ds Max and Gmax, but I did not use them for the levels I was designing. At that time, I did not understand the difference of world geometry and models as defined by the Goldsource Engine. I didn't even make a single model during those years.

By the time I moved to the Source Engine, I had a large collection of experience with 3ds Max under my belt, so I learned how to make custom models and textures. What I discovered was that the process was tedious, time-consuming and a complete buzz-kill. I almost gave up on Source altogether at that point.

But I enjoyed building environments so much, and I wanted to continue to make levels for those games I played the most: Counter-Strike: Source and Hidden: Source. So I started the Wall Worm project.

Originally, the goal was simply to build a system that made it easy for me, the artist, to send a model and texture to Source with no interaction with text files and compilers. But over time, the project has expanded into almost every aspect of Source asset production. Aside from sound, you can generally create any kind of asset used in Source entirely inside 3ds Max us-

ing Wall Worm: props, animated models, materials, textures, game levels, etc.

Now Wall Worm is used in the daily lives of amateurs and professionals around the world. The amount of time and effort it saves is a revolution in the Source Engine world.

Getting Started with Wall Worm

The home of Wall Worm for Source is <http://dev.wallworm.com>. That site contains articles, samples, videos, documentation and downloads. It is kept somewhat up-to-date, but not all information is always valid. Take one minute to scroll through the change log to realize how challenging it would be to keep the docs absolutely up-to-date; as such, do not be surprised if some information becomes obsolete. Some of the information in this publication comes directly from that website and some is new.

Whenever you have a question, you can use the search function on the documentation website to find an answer. If you don't find a solution, you can also refer to the Wall Worm Forums located at <http://www.wallworm.net>. You may find that someone has already asked/answered your question. Feel free to login and ask your own questions.

Download Wall Worm

There are always three versions of Wall Worm. The public release is linked from every page of the documentation site linked above. This is the **stable release**, and is always located at http://wallworm.com/store/index.php?route=product/product&product_id=55 .

There is also a commercial version of Wall Worm called [Wall Worm Pro](#). Wall Worm Pro has several enhancements not included in the general free Wall Worm, including a DMX exporter, faster VMF Exporter, expanded VTF Export options and a native VTF bitmap loader, among

others.

Installing Wall Worm

When you first download the Wall Worm zip file, you should immediately right-click the download file, click Properties and look for a button (Windows 7) or a check box (Windows 8+) that says “Unblock”. Click this and then hit OK. This is necessary because Windows now automatically blocks DLL files that are downloaded from the Internet—and there are now some DLL libraries in all flavors of Wall Worm that some functions require. Wall Worm Pro simply won't work at all if these files are not unblocked before unzipping, and some features in standard Wall Worm will also fail to run if blocked.

The Wall Worm zip file contains:

- WallWorm.com – Folder with all of the tools.
- changelog.txt – File listing all the version changes.
- readme.txt – File about Wall Worm, including credits, contact info, copyright, etc.

To install Wall Worm, simply copy the WallWorm.com folder from the zip file to the Scripts folder located in your version of 3ds Max. If you want to, you can use Winzip or WinRar for this, but you should be able to open it directly via Windows Explorer and drag the WallWorm.com folder from the zip into the correct directory. The most common location for 3ds Max 2015's scripts folder is: **C:\Program Files\Autodesk\3ds Max 2016\scripts**.

Note that in some settings in Windows, you may not see the extensions of file names. Although the WallWorm.com folder is not a file, Windows can interpret the name of the folder as a file and sometimes hides the ending. If you only see a folder called WallWorm inside of the

zip file you downloaded, then you should copy that folder instead (as it is actually the same folder).

Run the Installer Script

Once you have copied that folder correctly, the next step is dependent on the version of Wall Worm. The steps below are for the free version of Wall Worm. *For Wall Worm Pro, use the instructions that come with Wall Worm Pro instead.*

1. In 3ds Max, click **MAXScript > Run Script**.
2. In the dialog that opens, browse for **C:\Program Files\Autodesk\3ds Max 20##\scripts \WallWorm.com\install.ms**.

Now you should see a prompt to configure Wall Worm and have a new menu in Max called Wall Worm.

Configure Wall Worm

In order for Wall Worm to work correctly, you have to configure some settings. The settings are required because they point to the Source Engine compilers that convert your 3ds Max assets into game-compatible files and to folders that store intermediate files.

When you first install Wall Worm, you are immediately prompted to set up Wall Worm. You can also get to the settings any time by clicking **Wall Worm > Wall Worm Settings** in the Max menu bar.

Welcome to Wall Worm

The screenshot shows the 'Wall Worm Global Settings' window with several sections highlighted and annotated:

- Game Paths:** A green box highlights the 'Game and Asset Paths' section, which includes fields for modelsrc, materialsrc, mapsrc, Game Info, Game EXE, and Bin Dir.
- Compiler Paths:** An orange box highlights the 'Compilers' section, which includes fields for studiomdl, vtex.exe, vbsp.exe, vvis.exe, vrad.exe, bspzip.exe, and makesheet.
- FGD:** A cyan box highlights the 'FGD and Entity Settings' section, which includes a checkbox for 'MXS Entity Definitions', a 'Get FGD' button, a 'Repair FGD and Update Entity Definitions' button, and a text field for the FGD path.
- Version:** An arrow points to the 'Wall Worm Pro v3.446' label at the top right.
- Engine:** An arrow points to the 'Engine' dropdown menu.
- Saved Presets:** An arrow points to the 'Setting Presets' list, which includes presets like 'Counter-Strike: Global Offensive', 'Black Mesa XEN', 'BMS SDK**', etc.
- Activate WW Pro:** An arrow points to the 'Activate Wall Worm Pro' button.

Use the presets to easily switch between different games or different projects within one game where the paths or settings might not be the same.

* Marked Options for Wall Worm Pro Only

- Game Info is required for many functions to work. Without it, WW cannot get assets from your game (such as MDL, Materials, textures, etc). Also, assets will not compile.
- Compiler Paths can be left empty. When blank they are all assumed to be in the Bin Dir.
- FGD is required to load entities. If no FGD is defined, then you cannot create and edit entities inside Max.

There are five folder paths and one file path that have to be set.

- Modelsrc Directory
- Materialsrc Directory
- Mapsrc Directory
- Bin Directory

- Game Info Directory
- FGD File

While you can enter these manually, the easiest way to set up Wall Worm is to Import all the settings from your mod automatically, which is explained below. **Important:** Many functions in Wall Worm will simply not work if you do not configure WW for your desired game. The following functions will fail if not properly set up: Importing/Exporting models, materials and textures.

Import Settings

1. Open the Wall Worm Settings (**Wall Worm > Wall Worm Settings**).
2. Click Import button at bottom right.
3. Browse for **GameConfig.txt** for your mod. The **GameConfig.txt** is generally located in the bin folder inside your game.
4. Click on your Mod's name in the preset list.

Now the paths are set and a new game preset is added to the settings. Above is a screen shot of the settings that I use for Wall Worm. Most games follow similar structures.

Tips on Settings

The Wall Worm Settings floater controls global settings that affect all things in Wall Worm. There are some settings that create some drastic changes in how Wall Worm works, and these settings are generally only there to be compatible with some of the legacy exporters.

Here are some specific recommendations:

- The **Bin Dir** is a folder that contains some files necessary for compiling models, textures and levels. There are often multiple folders named “bin” in games. Make sure that you choose the one containing studiomdl.exe.
- **Use Wall Worm Pro if installed, otherwise the Wall Worm SMD Exporter** as the model export plugin (in the Models Tab of the settings). While you can use both the Wunderboy and Cannonfodder plugins with Wall Worm, there are many functions and features that only work with Wall Worm Pro and the Wall Worm SMD Exporter (such as Explicit Normals, Arbitrary Mapping Channels, etc).
- When using the **(recommended)** Wall Worm SMD Exporter, **do not** turn on the Legacy VTF and Legacy Tex Name options unless you have the engine set to Goldsource! Conversely, you need to turn them on when using the legacy SMD exporters (not recommended).
- If there is a syntax error in the FGD file your mod uses, Wall Worm will fail to create entities. At the time of this current publication, CS:GO has such a syntax error and must be edited to work properly in Wall Worm. See the Wall Worm forums for more information.

All recent and future references (in this document, online and videos) use and presume that these settings are followed. While the legacy options are available, they are only provided for legacy purposes. Wall Worm no longer tests how any updates affect and relate-to the legacy tools. There is a good chance that Wall Worm will drop support entirely for the legacy plugins in the future.

This also means that the information included in a large percentage of the online documenta-

tion and tutorials regarding models and their materials is invalid with working with Wall Worm's preferred settings. The majority of the materials on the Internet regarding adding materials and textures to models for exporting to Source refer to the requirements of the legacy SMD Exporters. Wall Worm has a different method that offers more robust control of materials and textures, which are covered in Chapter 9: Materials and Textures.

Understanding and Using Wall Worm

For the most part, Wall Worm is a collection of pipeline tools and utilities. It bridges the gap between 3ds Max and Source. The philosophy behind it is to allow you to think more about using 3ds Max itself as a design tool, and less about technical aspects that are requirements in Source. At this point, an expert user of 3ds Max can do all level design, modeling, texturing, entity scripting, animating, exporting and packing entirely inside one single application (without transferring assets into 3rd-party applications, opening text files, writing batch files and working directly with compilers).

Wall Worm is for both beginners and advanced users. If all you want to do is re-skin a model, or build a basic prop, you can do this simply—without any need to learn the under-the-hood aspects of Source. But if you are already knowledgeable of Source, Wall Worm gives you a GUI inside 3ds Max for the majority of Source Engine properties in models and materials, and all for level design. As much as possible, all technical aspects of exporting models that does not require creativity and can be automated have automation—for example, when setting up a ragdoll model, Wall Worm will simply build the ragdoll settings in the QC from standard bone and hierarchy attributes you've set inside Max—meaning you don't have to edit text files and guess bone angle limits, etc. It just works.

The Wall Worm menu in Max is divided into eight sections that relate to different types of utilities and tools. Those dealing with models and managing model elements are located in Wall Worm Model Tools. Those tools related to level design (Anvil, Sky Writer, etc) are located in

the Wall Worm Level Design section. Some tools that are relevant to different categories are located in multiple menus (for example, Sky Writer is located both under Wall Worm Level Design as well as Wall Worm Materials.)

The available menus as of this publication are:

- Wall Worm Model Tools
- Wall Worm Level Design
- Wall Worm Materials
- Wall Worm Utilities
- Wall Worm Importers
- Wall Worm Exporters
- Wall Worm Extras
- Wall Worm Online

Wall Worm Model Tools

Wall Worm Model Tools (also called **WWMT**) were the first tools released in the Wall Worm tool bag. These tools are for sending models inside 3ds Max into Source without much fuss. This category of tools includes controls for Model Settings, Body Groups, Collision Hulls and more.

Explaining every feature inside Wall Worm Model Tools is beyond the scope of this discussion. See Chapter 8: Source Engine Models for more information.

Wall Worm Level Design Tools & Anvil

The Wall Worm Level Design flyout menu contains the longest list of menu functions in Wall Worm. From here you can launch **Anvil** and other tools. Anvil is the floater that contains the tools necessary for creating displacements, tagging geometry as World Geometry or Skybox objects and managing multiple scene WWMT Helpers and Proxies. These tools will be covered in later chapters, as they require more detail.

From the Wall Worm Level Design menu, you can also find functions for opening the Entity Tools, Worm Face (for placing decals and overlays), a leak file loader, a RES file generator and more. Using these tools will be explained later in context.

Wall Worm Materials

This set of functions allows you to export materials in the scene to VMT/VTF files, import material libraries from Source, add Source VMT shader properties to Max materials, render overview/radar maps and render out 2D sky textures.

Utilities, Importers, Exporters and Extras

The Utilities menu in Wall Worm contain miscellaneous functions (like checking for Wall Worm Updates or Check for Problems in Max or the scene). The importers will allow you to import some Source Engine Assets into 3ds Max (like props, materials, levels). The Exporters are the tools for sending different objects from Max to Source (your models, textures, levels). And the Extra are some general-purpose functions that are relevant to situations even beyond game design and Source.

Wall Worm Conventions

There are a few conventions that are used in many Wall Worm functions. One of the main conventions is that functions that work on a selection of objects will generally work on all relevant scene objects if no objects are selected.

For example, the function to Export Selected Model Textures will export the materials and textures of any currently selected WWMT Helpers, but if nothing is selected, all textures from all models in the scene will export and compile.

Wall Worm's Focus is about Creating Custom Content

As a final note about Wall Worm, it is important to point out one of the main principles behind Wall Worm. **Wall Worm is about allowing you to create all of your own custom content from scratch.** But that doesn't mean you cannot bring your pre-existing content into Max. There are a several importers built into Wall Worm to let you finish projects you've started in Hammer or elsewhere. You can import Source assets like VMT, VMF, MAP, MDL, QC, SMD and VTA into Max. Wall Worm Pro can also import VTF files.

In Max 2015+, Wall Worm can get MDL, VMT, VTF and some other data straight from the game folders and out of VPK files. The [MDL Loader](#) is a special geometry class available in WW that allows you to place Source Engine models in the scene like any other native geometry object.

The exporters in Wall Worm include: QC, SMD, VTA, DMX, VMF, MAP, RES, RAD, VMT, VTF and more. Many of the tools will run compilers automatically to convert these into game files like MDL, BSP, etc.

Wall Worm Team Work

One of the focuses in Wall Worm is making it easy for teams of users to collaborate on projects. While this document is largely focused on getting a single user up-to-speed on using Wall Worm, you may want to fit Wall Worm into a team effort. Wall Worm creates an actual pipeline for Source instead of just a random collection of methods that you traditionally used to put together a game in Source. You can learn more about a team project in Chapter 21 Anatomy of a Design Team.

Extra Wall Worm Functions

Wall Worm has menus and a UI for most functions. However, several of the tools in Wall Worm are not exposed to the UI. For these, you must add them to your UI how you wish. To access these (and all main macroscripts) you can click Customize > Customize User Interface. This menu will allow you to set custom keyboard shortcuts, buttons, quad menus and main menus. To see the Wall Worm functions, you must change the Category drop-down to “wallworm.com”.

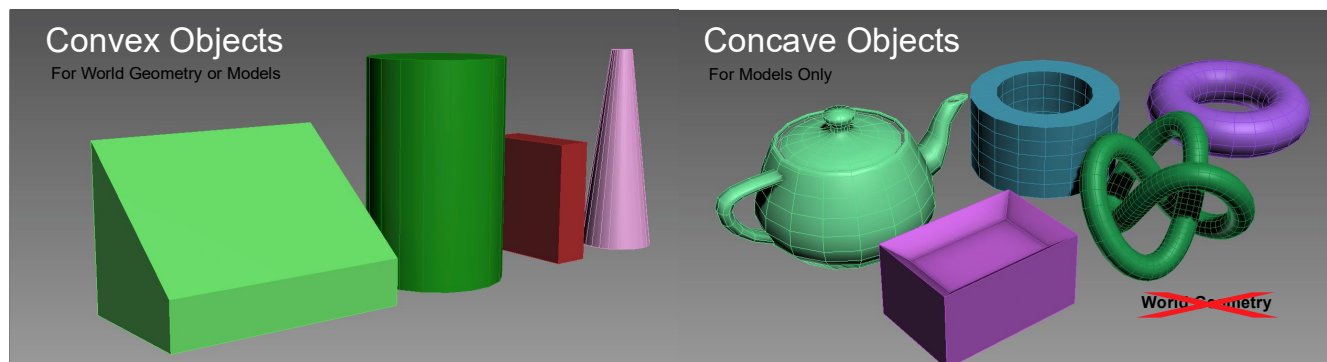
Adding Common Hammer Functions in Max

- **Increase Grid Spacing:** Assign this to your Right Bracket (]) key to make the grid spacing double.
- **Decrease Grid Spacing:** Assign this to your Left Bracket ([) to reduce the current grid spacing in half.
- **Display Selected Object Dimensions:** Assign this to a keyboard shortcut or button to turn on the display of selected object dimensions in the world.

Chapter 7 Source Engine Level Design in Max

As mentioned in Preparing for 3ds Max, there is no real difference to the rules for making your levels between Hammer and 3ds Max. The difference is in the methods you go about to build your level. You have a few extra considerations about your objects when making them in Max that aren't there in Hammer.

Every geometry object you make in Max that you want in the final output must be categorized as either World Geometry or Model. This isn't required in Hammer simply because Hammer cannot make models.



World Geometry

The world geometry is the backbone of your level. It defines the layout and boundaries of the level. It must be composed of convex geometry that does not contain co-planar polygons. Your entire playable area needs to be sealed off from the “outside void” by world geometry. As much as possible, vertices of world geometry should be aligned to the world's 1x1x1 grid (any

vertex at a position like [1.34,23.4,16.0] is not aligned to the grid).

It is always a good idea to have **Snaps** turned on with **Snap To Grid** and/or **Snap to Vertex** enabled when working with World Geometry.

To help with grid snapping, you can install Wall Worm's [SnapVertsToGrid Modifier](#) or [Brushify Modifier](#) (preferred, but only works in Max 2016+). These modifiers will automatically snap your geometry vertices to the grid.

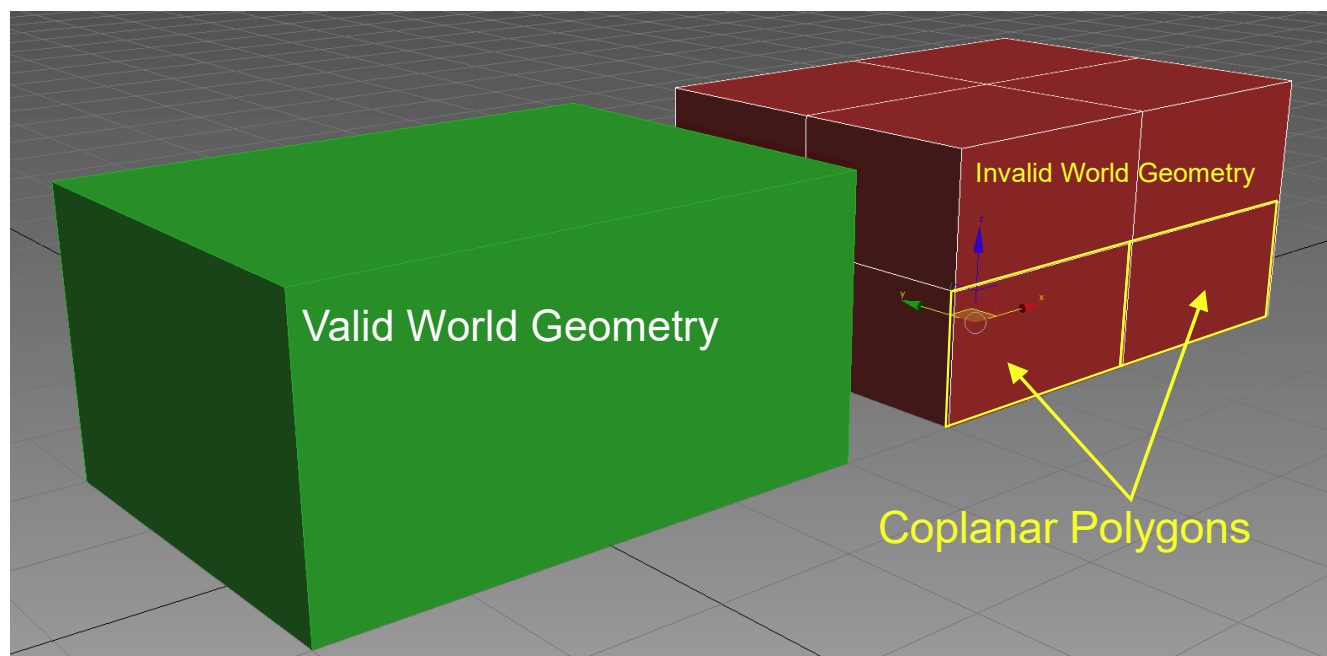
You can use **Standard Primitives** like **Box** for your world geometry, or you can use closed **Splines** that have an **Extrude Modifier** applied (as long as the base spline shape is convex).

Or you can get [CorVex](#), [Arch](#) and [ShellVex](#), which are plugins developed by Wall Worm specifically for the process of building level design in an efficient, parametric manner. These three geometry primitives are designed for making brushes.

The Difference Between Delete and Remove Functions

The following sub-sections indicate using the **Remove** function for cleaning up some kinds of geometry. It is important to realize that this is not identical to using the **Delete** function on the keyboard. The **Delete** command on a vertex or edge is generally not the command you want to run when cleaning up your geometry from non-planar polygons or coplanar polygons because they actually remove polygons from the object; the **Remove** function should generally keep the general convex structure of your geometry unless there are serious flaws with the geometry.

Avoid Coplanar Polygons on World Geometry



When designating an object as world geometry, you must not only remember to use convex objects, but you must also avoid coplanar polygons. For example, the image above shows two Box primitives. The one on the left is valid because it has no coplanar polygons and is composed of length/width/height segments set to 1. The one on the right, however, has four coplanar polygons on each side of the object because its length/width/height segments were set to 2. Failure to follow this rule is one reason you will see a message like the following in your BSP compile log: “Brush 82, Side 4: duplicate plane”.

Fixing Coplanar Polygons Manually

If you have coplanar polygons, you can clean them up manually with these steps:

1. Apply an Edit Poly modifier to your object *or* convert it to an Editable Poly object by right-clicking your object and clicking **Convert To... > Convert to Editable Poly**.

2. Set the **Sub-Object level** to **Edge** in the modify tab (or press the **2** on your keyboard).
3. Select an edge between two coplanar polygons.
4. Click the **Remove** button in the modify tab.
5. Continue this process until there are no more coplanar polygons.
6. Switch to vertex sub-object mode (keyboard shortcut **1**).
7. **Select all vertices** that have only two edges.
8. Click the **Remove** button in the modify tab.

Fixing Coplanar Polygons Automatically

The process outlined above to fix coplanar polygons can be tedious, depending on the number of invalid objects in your scene and their collective complexity. The best way is to use the Fix Coplanar function in the Utilities of [Carver](#). You must purchase Carver to be able to use these steps.

1. **Select the objects** that needs to be fixed.
2. Convert them to **Editable Poly**.
3. **Open Carver** by pressing **Wall Worm > Wall Worm Level Design > Wall Worm Carver**.
4. Press the **Fix Coplanar** button.

Keep Your Polygons Planar (Not all Convex Objects are Valid Because of Non-Planar Polygons)

Valid geometry must not only be just convex with no coplanar polygons, but to properly export as you see in the Max viewport, you must make sure that all polygons are planar. This may seem like an odd statement if you are not familiar with the difference between a face and a polygon. All faces are planar, but not all polygons are planar. Polygons can be composed of any number of faces that do not have to reside on the same plane. As such, it is up to you to understand this.

Wall Worm exports each polygon as a plane of a brush in the VMF exporter because checking for planar polygons at export time is too slow. This can be confusing if you do not understand this concept because an object that has non-planar polygons can still be convex and appear to be valid world geometry. In the following image, you can see that the middle object is convex.



In the above image, the solid white lines are the edges of polygons. The dotted lines repre-

sent the triangulation (of faces). The two left objects will not export as world geometry into your game as they appear in Max. Only the object on the right will correctly export.

Fixing Non-Planar Polygons Manually

You can fix non planar polygons with several methods. Below are two approaches to fixing your polygons via connecting vertices or cutting polygons. The method of using **Connect** between vertices is most accurate but slower. The **Cut** method is faster but sometimes has minor errors than need fixed. Knowing both methods is helpful.

Connecting Vertices

1. Convert your object to an **Editable Poly**.
2. Switch to **Vertex Sub-Object mode** (press **1** on the keyboard).
3. **Select two vertices residing along the same polygon that do not have an edge between them** but should have an edge to divide the polygon into separate polygons.
4. Click the **Connect** button in the modify tab.
5. Continue steps 3-4 until the object no longer has non-planar polygons.

Cutting Polygons

Using Polygon sub-object mode can help you visualize the non-planar polygons a little better sometimes because you can use the Edit Triangulation button to visualize the current triangulation (which will display dotted lines on the triangulation of polygons as in the graphic above).

You cannot actually cut the polygons while in this mode, but you can jump back and forth between the Edit Triangulation and Cut function (which is detailed below).

1. Convert your object to an Editable Poly.
2. Enter Polygon Sub-Object Mode (press **4** on the keyboard).
3. Turn on **Snaps (S)**.
4. Turn on **Snap to Vertex** and turn off all other snapping.
5. Click the **Cut** button in the modify tab (also **ALT+C** *when Max's Keyboard Shortcut Override is turned off*).
6. Now click from one vertex to another vertex sharing the same polygon (where there should be an edge).
7. Once you need to move to different vertex, right-click to cancel cut mode.
8. Repeat steps 5-7 until all polygons are planar.

Fixing Non-Planar Polygons Automatically

There are some other ways to fix non-planar polygons. One way is to turn on Break Non-Planar Polygons in the VMF Exporter. This option is not ideal because it slows down the VMF exporter and invalidates sidelist values which affects entities like decals.

The non-destructive solution is to use a **Turn To Poly** modifier with the *Keep Polygons Convex* option and the *Require Planar Polygons*.

The destructive solution is to use the Break Non-Planar Objects function in Wall Worm. (By destructive, I mean the modifier stack is collapsed to an editable poly.)

1. Select your object(s) that should be fixed.
2. Open the **Problem Checker** by clicking **Wall Worm > Wall Worm Utilities > Problem Checker** .
3. Click the **Break Non Planar Polygons on Selection** button.
4. Verify the results and because this function can sometimes create co-planar polygons which are discussed above.

The Brushify Modifier

Some of the steps above can be most effectively handled with the [Brushify Modifier](#), which has been listed so far as a tool to snap vertices to the grid. Aside from grid-snaps, the Brushify modifier also has settings to Break non-planar polygons non-destructively and to remove coplanar polygons. This modifier (with the Turn To Poly modifier) are the fastest combination of tools to turn existing geometry into cleaner brushes.

Assigning World Geometry

Unlike Hammer, where all generated geometry is World Geometry, Wall Worm requires you to specify what objects in a scene are World Geometry. This means that even if your entire scene is composed of objects that are intended to be World Geometry, those objects won't export unless properly designated.

There are three ways to export objects as World Geometry: Tags, Layers and Native Brushes. The difference between these methods is explained below.

Tags

Tags are how you can designate an object as a certain kind of level geometry. To export any object as World Geometry, select it and click the menu **Wall Worm > Wall Worm Level Design > Set Selection as Brush Geometry**. You may want to assign this as a shortcut inside the 3ds Max customization. To un-tag an object as world geometry, click **Wall Worm > Wall Worm Level Design > Remove Selection from Brush Geometry**.

You can also access these functions in the Tags tab of Anvil.

Layers

Another way to export objects as World Geometry is to put them into one of two layers that are named **Convexity Walls** or **Convexity Floors**. These are not layers that exist by default, but are layers that get created automatically in a third party tool called Convexity. Although Convexity is no longer developed, this layer convention is used to make integration with scenes created with Convexity tools easier to export with Wall Worm. You can simply create one of these layers in the Max layer manager to utilize the layer method.

All objects in these layers will export as World Geometry. So do not place invalid geometry into these layers!

Native Brushes

Many Wall Worm Objects (CorVex, ShellVex, Arch) will export into Source if **Export as World Geometry** is checked (which is on by default when creating them). [CorVex](#) will be discussed later.

Brush Mode

Brush Mode is a utility that will do a few things to make a Hammer user feel more at home. To toggle Brush Mode, click **Wall Worm > Wall Worm Level Design > Brush Mode**. When active, there are two major things that happen:

1. Each New object you create is automatically tagged as world geometry.
2. The currently selected material in the Material Editor is applied to newly created geometry.

Brush mode is very convenient during the layout phase. But you must be careful about creating invalid world geometry while in brush mode! For example, do not create a Teapot primitive while in brush mode—as a teapot is not a valid geometry because it is concave.

Concave Brushes

Although no brush is valid that is exported as a single concave object, Wall Worm does allow you to export concave objects as brushes. You can tag an object as a “Concave Brush” to tell the VMF exporter to explode the object into its elements—and each element sub-object will export as a single convex brush.

This function still requires that *each element* of an object is convex. So even with the Concave Brush tag, a teapot will never export as valid world geometry!

A practical application of this is to combine several convex objects (like Box primitives) into a single editable poly object where each of the original boxes is a convex element in the editable poly. Working with combined objects like this has some benefits when you want to work with some kinds of modifiers or tools of an editable poly.

To set an object as a concave brush, press **Wall Worm > Wall Worm Level Design > Set as Concave Brush**.

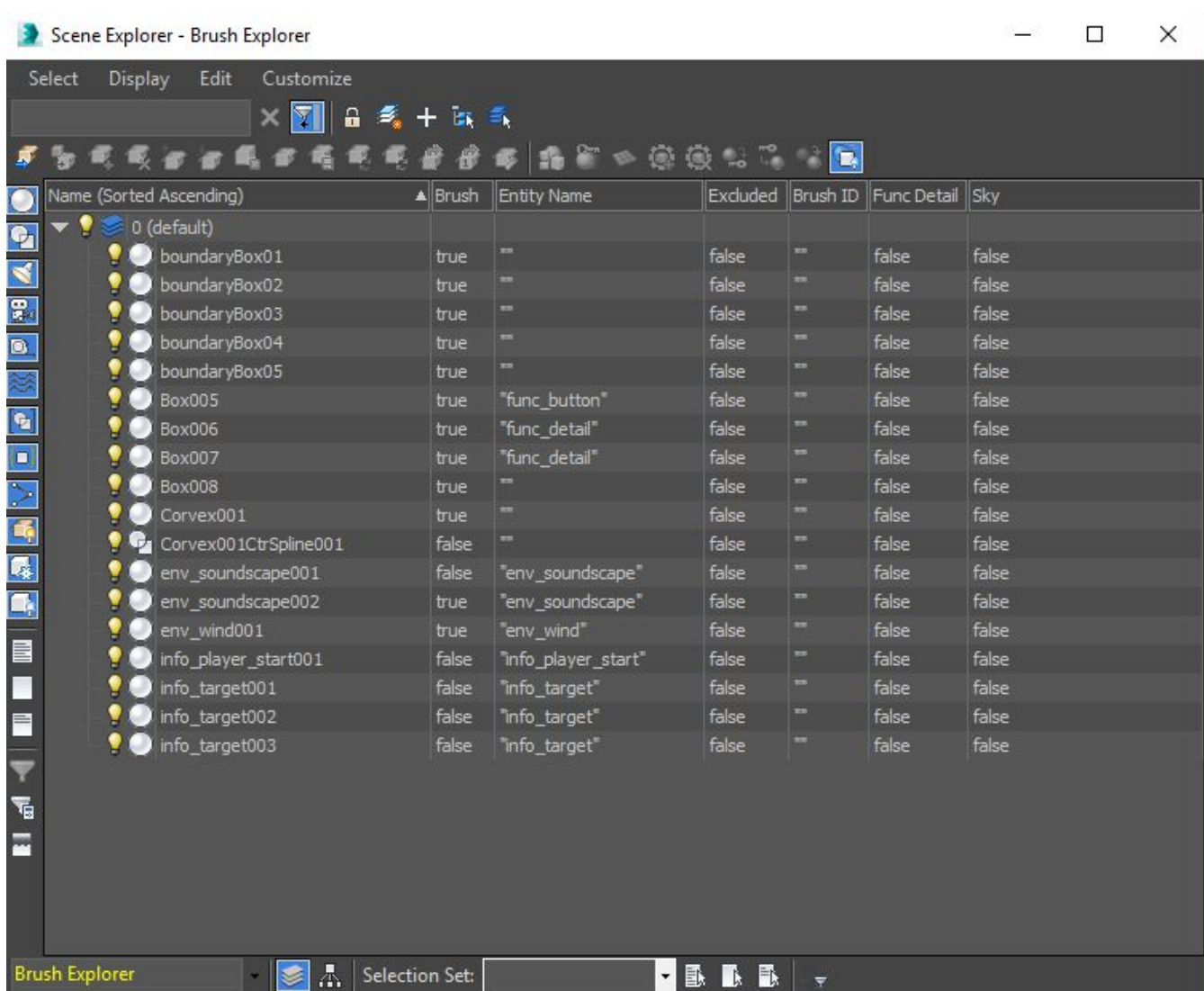
If you want to convert several brushes into a condensed concave brush, you can do this with the these steps:

1. Select the brushes to condense.
2. Click Wall Worm > Wall Worm Level Design > **Condense Brushes**.

Now all the brushes are merged into a single editable poly object. This is an especially helpful tool if you have a collection of brushes that you want to use as the backbone of displacements where the brushes were created originally in Hammer. Once condensed, you can use standard editable poly tools to select specific faces (for example, from all the materials that should be used to make displacements).

Brush Explorer

There is a brush explorer that will let you view whether objects are tagged as brushes or not. To launch the brush explorer, click Wall Worm > Wall Worm Level Design > **Brush Explorer**. See following graphic.



Entities

Entities are the brains of your level. You can place point entities and use brush entities just like in Hammer. Entities are discussed more in Chapter 12: Working With Entities.

One thing that is different between Hammer and Max is that you should not add *prop entities* into a Max scene. Instead, you should use Wall Worm Model Tools (WWMT) and Wall Worm Proxies (discussed in the next chapter). This doesn't mean you cannot add props via a Hammer-like method of adding a prop entity and browsing for a model; but if you have a representation of a prop that has a WWMT Helper assigned to it, you should use the Proxy tools.

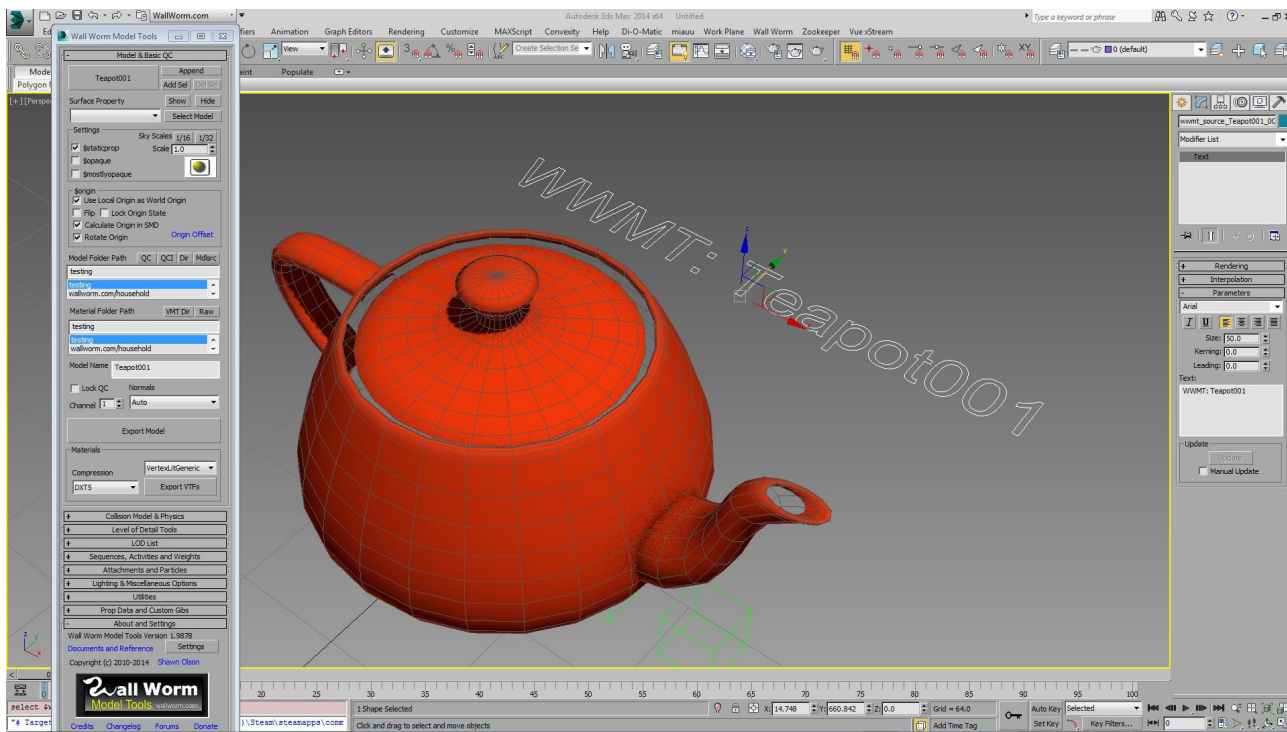
Chapter 8 Source Engine Models

Where world geometry is the layout of your level, models are all the details that bring it to life. A few years ago, it could take minutes to hours to compile a single prop into Source (even after all the actual design is done). That's not the case anymore thanks to Wall Worm. Simple props now take mere moments. Even creating LOD models, collision hulls, prop data and more is all fairly simple. There are even batch tools to export/compile hundreds of models with the click of a button, all at once!

Make Your First Model in Source

1. **Open** a new empty scene in 3ds Max.
2. **Click** the **Create Panel** in the **Command Panel**.
3. **Click** on the **Teapot** object type button.
4. **Create** a Teapot in the viewport.
5. Now open Wall Worm Model Tools (**Wall Worm > Wall Worm Model Tools > Wall Worm Model Tools**).
6. Click the **Pick Model** button in the WWMT UI.
7. Now **Pick** the Teapot in the viewport by clicking on it.
8. Click on the **Export VTFs** button to open the Material Exporter.

9. Click the **Export Selected Textures** button. You'll see a prompt that the Materials Exported.
10. Click **OK** and close the Material Export dialog.
11. Now click the **Export Model + QC** button in Wall Worm Model Tools. You'll see a compile process window.



That is the process of getting a model into Source at the most basic level. Of course we only made a material with no texture, and we didn't do anything fancy. But now this teapot model is inside your game and is accessible to levels in the game.

The WWMT Helper

Whenever you pick a model with Wall Worm Model Tools, a new object gets added to the scene. In the image above, that item is currently selected and has the words “WWMT: Teapot001”. That is called the WWMT Helper. Technically, it is just a Text Shape object. But it also stores all of the data for your model.

When you want to retrieve the settings for your model, you can open Wall Worm Model Tools, click **Pick Model** and pick the **WWMT helper**.

Some things to note about the WWMT Helper:

- It is linked as a child to the model when created. This means that moving or rotating your model will result in the WWMT Helper moving as well. It is safe to unlink the WWMT Helper.
- When you update the Model Name field in the Wall Worm Model Tools UI, the WWMT Helper will update to match the new Model Name which defaults to the node's name when created.
- If the WWMT Helper is selected, you can edit most of the settings in the Modify Tab without opening the WWMT Floater.

Wall Worm Proxies

Wall Worm Proxies are copies of your model that can be scattered around a scene. The purpose of a WW Proxy is identical to placing props in Hammer. These are important objects for both making models and for your level layout.

Inside WWMT, you can create a Proxy by clicking the Create Proxy button in the Utilities roll-out. (You can also find this ability in Anvil and the WWMT Proxy Tools under the level design menus).

As of WW 2.9, newly generated proxies are part of a geometry class called WallWormMDL. They will reference the actual compiled (MDL) models for 3ds Max 2015+.



Illustration 1: Proxies places manually and with Object Paint.

If you do alter your WWMT model after proxies have been made, you can update the proxy meshes by selecting the proxies in the scene and clicking the **Update Selected Proxy Meshes** button in the Proxy Tools floater. The scene above is populated in large part by Proxies (trees, palm fronds, ground vegetation). It is from a level that I've been using to test features as I've developed Wall Worm.

When using Wall Worm for your level editor, it is important to know about Proxies and how they work. If you need multiple copies of a model in the scene, use Proxies instead of copying the model and WWMT helper. I like to use the **Object Paint** tool in Max to paint proxies onto an environment, because you have several scattering and rotations options; this is a good way to paint a landscape with rocks, plants, trees, etc. Another great way to scatter proxies is with Forest Pack from Itoosoft as demonstrated in image below.



Illustration 2: Proxy distribution with Forest Pack from Itoo Software.

Make Skins from Proxies

One thing you can do with proxies is generate skins for your model. For example, all of the

palm fronds above are a single model in game, but there are different textures applied to each (which, in game terms, means they are instances of the same model that are using different skins).

Use the following steps to generate skins for your WWMT model based on its proxies in the scene:

1. Create a Proxy of your model. You can do this in the WWMT UI by opening the Utilities rollout and clicking the **Create Proxy** button.
2. You can copy that proxy as much as you need to (making one proxy per each skin you want to create).
3. For each proxy to be used for a skin, select it, open the modify tab and click the **Collapse for Skinning** button.
4. Apply a new material on each proxy that should represent a new skin. Do whatever you want to the material and bitmaps (making sure to use the same UV channel in your main material).
5. When done, click the Collect Skins from Proxies button in the Utilities rollout. This function automatically creates the skin material from the proxies and applies it to the WWMT helper.
6. Re-export the VTFs and VMTs for this model (which will have more now that there are skins).
7. Re-export the model.

Note that the Skin Material is stored as a Multi/Sub-Object Material on the WWMT Helper itself. Also note that skins will only be collected from collapsed proxies (see #3 above) since the MDL objects are already tied to actual skins.

Tips on WWMT Helpers and Proxies

By default, all WWMT Helpers and all WWMT proxies will export as **props** in a level exported with the VMF Exporter. Sometimes, it is best to only export the proxy models and not the WWMT Helper model. You might want to do this if your WWMT Helper model is in a separate design layer or outside the bounds of your level, which you might do to separate some WWMT Helpers from a cluttered scene to optimize testing different settings in your models.

For this purpose, you can exclude WWMT Helpers from the export by selecting the WWMT Helpers you want to exclude and click **Wall Worm > Wall Worm Level Design > Exclude Export of Model**.

More Tips on Proxies

- Your base WWMT model should have its Xform reset before any proxies are created for it.
- Once you make proxies of a WWMT model, you should no longer change the WWMT model's position, rotation or scale. Do not reset its Xform after creating proxies.
- Do not ever reset the proxy Xform.
- Only change the scale of a proxy if you have also tied it to a point entity like prop_static and your mod has prop scaling. In this case, you may want to use the model scale

property in the entity parameters to control scale and not the Max scaling function (it will visualize in the scene).

- When using the Forest Pack plugin from Itoosoft, you should assign only WWMT Proxies as custom objects and never a WWMT model. (Do not use scaling in the Forest object unless the mod supports prop scaling.)

Controlling Prop Types of WWMT and Proxies

By default, the VMF Exporter will automate the prop type when you export a VMF. These are the rules:

- If the WWMT Helper has Prop Physics settings, it will export as a prop_physics.
- If the WWMT Helper has \$staticprop checked and no Prop Physics settings, it will export as a prop_static.
- If the WWMT Helper does not have \$staticprop checked and no Prop Physics settings, it will export as prop_dynamic.

To override these defaults, select the WWMT Helper and click **Wall Worm > Wall Worm Level Design > Point Entities**, choose the desired prop type, and click **Selection as Point Entity**. *Note that when applied to the WWMT Helper, the entity propagates to all proxies.* If you desire another effect, select a subset of the proxies and choose the prop_type you desire.

Working with Multiple Models and Proxies

For efficiency and rapid development, Wall Worm has tools for working with multiple models

at once. The place for these is in the Models tab of Anvil. These functions allow you to create many WWMT helpers at once, change properties of multiple existing WWMT helpers at once, create collision hulls on many models and compile multiple models and their textures with a single click of a button.

Multiple WWMT Helpers

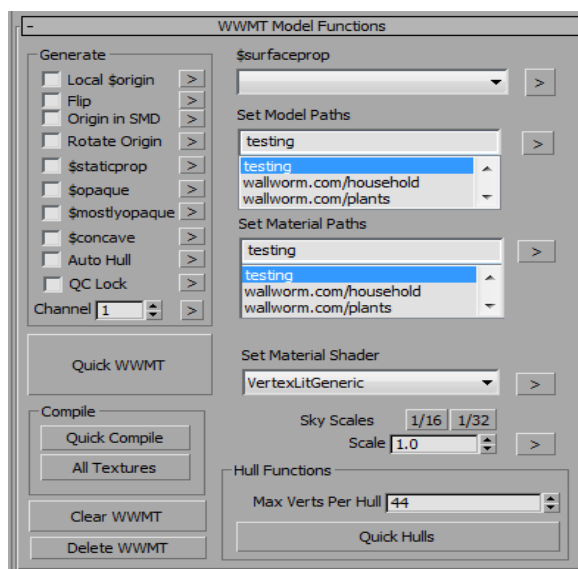
The WWMT Model Functions rollout in the Models tab of Anvil allows you to create multiple WWMT helpers at once as well as update some basic settings across multiple existing WWMT helpers in the scene.

Quick WWMT

The **Quick WWMT** button will make a WWMT Helper for each currently selected object in the scene. For example, if you have 25 different objects selected when you press Quick WWMT, there will be 25 WWMT Helpers created. And if you hit the Quick Compile button, all 25 models will export to Source.

When you press the Quick WWMT button, all of the settings in the check-boxes, com-bo-boxes and select menus inside the Model Functions rollout will get used in the

WWMT helper(s) that are created with the Quick WWMT results. For example, if the Local \$origin and \$staticprop are checked and the \$surfaceprop is set to Wood, then all WWMT



Helpers created will have those settings applied.

You can also update individual properties across all selected WWMT helpers in the scene by clicking the > button next to the property. For example, if you switch the Model Path to “mystuff/newpath3” then all selected WWMT helpers will update to use that path when you press the little arrow button.

Wall Worm Proxy Tools

Also located in the models tab of Anvil are the Wall Worm Proxy Tools. These tools will allow you to create proxies from selected WWMT Helpers, set the start/end fade distances of selected proxies and more. You can also load the proxy tools independently of Anvil by clicking **Wall Worm > Wall Worm Level Design > WW Proxy Tools**.

Many of the functions in this tool should be self-explanatory, but if you have questions about what they do, you can hover over the button or menu for a tool tip.

Changes that Require Re-Compile

An important concept for new Wall Worm users (and anyone new to Source) is that some changes you make in a scene apply to properties exported to the VMF (the file that the compilers convert to a playable level in BSP format), whereas other apply to properties exported into the models or materials.



Almost any change you make to a WWMT Helper's settings require you to recompile the model. Also, changes to the WWMT material path will also require you to re-export the VMT (material). Changes you make to a model's skins also require you to re-export the model and materials. Any change to a scene entity settings (like a fade-distance of a prop) requires you to re-export and recompile the level.

Converting Scenes to Models

One tool that is helpful to create models is the Convert Scene to Model tool. This tool will allow you to convert scenes into models, cluster props into single models and cull model faces by the landscape. You can open this under Wall Worm > Wall Worm Model Tools > **Convert Scene to Model**.

Convert Scene to Model has two basic preset modes: Clustering and Scene to Sky Model. The intent of clustering is to condense multiple props into clusters to help reduce prop count.

The Scene to Sky Model is to convert entire levels or sections of levels into single props as you might need if you wanted a level to be a skybox model for another level.

One of the handier features of this tool is the Union and Culling functions in the Advanced Options group. The Union Parts (which requires the Wall Worm Carver plugin) will cull away all intersecting faces of clustered props. The Use Culling Objects will allow you to set larger surfaces (like the terrain) from which to remove faces that intersect. This way, faces of props that are never seen by the player can be removed entirely from the resulting models.

Clustering Models

Assume you've imported a VMF into 3ds Max and you want to convert a bunch of the props into single models. To do this you'll use the clustering function.

Clustering Setup

First you'll want to assign all culling objects. Normally this will be the displacements, but might also include brushes. It's best to only use one culling object at a time—so it's recommended to create a sculpt mesh of all the displacements in the area you are clustering (see Chapter 10 Displacements).

1. Import your VMF into Max (Wall Worm > Wall Worm Importers > Import VMF or Map).
2. After imported, click Wall Worm > Wall Worm Model Tools > **Convert Scene to Model**.
3. Choose **Clustering** from the Presets.
4. **Select** the Culling Object (the sculpt mesh info mentioned above).

5. Click the Add Scene Selection in the Advanced Options.
6. Set a Model Name. You might set this to MyMapCluster or treecluster or rockcluster.
7. Turn on Move to Layer (so each clustered set will go to a new layer). By default the layer name is Collapsed Props, but you can select or choose any layer.

Clustering Your Props

Now that the clustering settings are ready, you can start the clustering process. You will want to plan out the clustering before you start by thinking about what props should be clustered into their own model. Once you have a plan, take these steps:

1. Select all the props that should be a single cluster
2. Click the button labeled Do It.
3. After the function finishes, verify that the results are accurate.
4. *If the results are accurate*, click the **Accept Last** button. Immediately the prop will be hidden to let you know it's already been collapse.
5. *If the results are inaccurate*, click the **Undo Last Collapse** button. Repeat steps 1-5 a few times to see if the results are good.*

Once you click accept, there will now be a WWMT Helper tied to the new prop with a unique model name based off of the current model name field (myMapCluster01, myMapCluster02, etc). That WWMT Helper can be exported as the new model. Note that it will include a collision hull that was derived from the original props' hulls.

Note that when you accept the results, the resulting prop is automatically hidden if the Hide Accepted button is on (which is on by default). This is a convenient way of not re-collapsing the same props.

* If a prop collapse fails due to a bad boolean, you should continue the undo/do it cycle a few times. This is because the function for the Union is affected by the node selection order, and when you click Undo Last Collapse the tool will randomize the selection to try and get a better result.

Chapter 9 Materials and Textures

When I first delved into 3ds Max, I was a little confused about the difference between Materials and Textures. The two always seemed to be discussed together, and were often used in *seemingly* interchangeable contexts. Part of my early confusion is probably a result of the terminology I learned from Hammer itself. In Hammer, there is the **Texture Application Tool** and the **Texture Browser**, but no *Material Browser*. This is likely because in Goldsource, objects only had access to simple materials with a single texture.

Another confusion for new users is the terminology of map. In some communities that make levels in Hammer, the word “map” is often reserved only for the actual game level (and “mapping” is the process of building a level). But in 3ds Max, the word “map” is applied to textures, because it comes from the term **Texture Mapping**. What Hammer calls the “map” is what 3ds Max calls the Scene. In Max, almost every reference to a map deals with textures or mapping textures to surfaces. If this is confusing to you, just think of another word you use all the time: **Bitmap**. Bitmaps are images where each bit of information is mapped to a location in X/Y.

The words **texture** and **map** can be used interchangeably in 3D. For the rest of this document, the word map will apply to textures.

Even after the Source Engine was introduced, Hammer continued to label all the materials as textures—despite the fact that the engine has many kinds of materials that utilize many kinds of textures (maps). The Hammer label is actually incorrect for Source, as you are not actually browsing for textures at all—you are browsing for materials.

Dropping the Hammer Approach

Dropping the Hammer approach to making materials is easy, because Hammer doesn't allow you to create materials and textures. However, there are long-standing methods used in the Source community that can hinder your understanding of the way you can really unleash your creativity for texturing scenes in 3ds Max. Integrating material/texture design into your level design process is key to understanding an added benefit of Max over Hammer.

First, realize that you can make materials and textures entirely inside 3ds Max. *Entirely.* I'd suggest that the vast majority of your texturing can stay within Max, and only when you need to do final tweaks should you consider moving to other tools (like PHOTO-PAINT, Gimp, Affinity Photo, others) if at all! Tools like Viewport Canvas let you paint layered textures directly on models inside Max, and the Slate material editor has a robust array of tools for mixing textures into material channels. Finally, Wall Worm will export materials and textures for you, alleviating the need to open text files or VTF converters.

Defining Materials and Textures

A material in 3D is similar to a material in the real world. Materials include Steel, Wool, Water, etc. Every material has a set of properties that define how we perceive surfaces composed of such materials. Is it dull? Is it reflective? Is it blue? What does it sound like when we hit it with a crowbar?

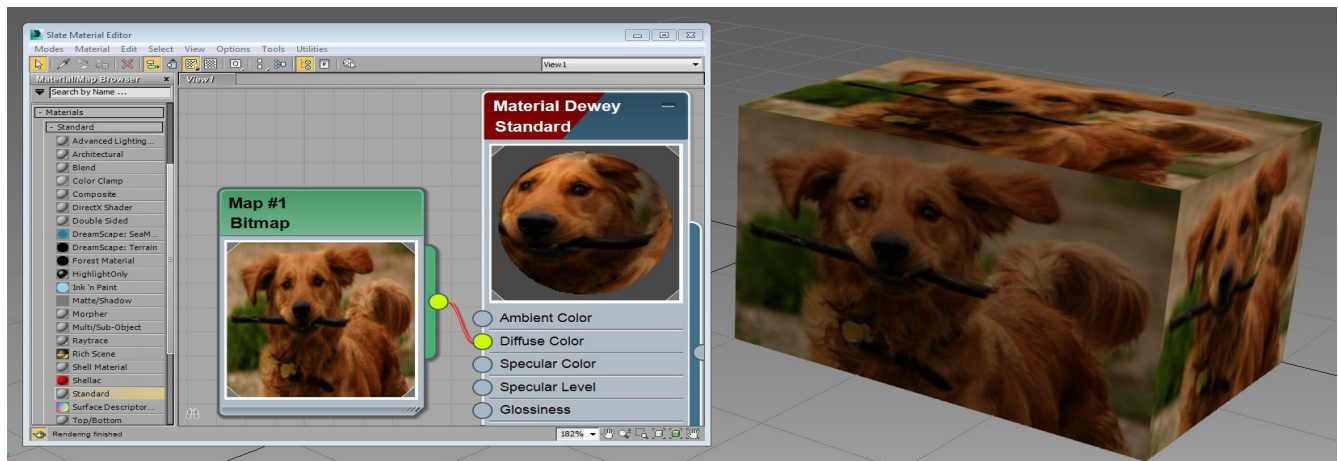
Materials are very complex things, just as complex as models and objects, and they are just as important as the geometry.

Different classes of materials (in both Max and Source) have different sets of properties. Inside 3ds Max, the differences are determined by what material you choose and sometimes by a Shader Type. In Source, the parameters available are determined by the Shader assigned

Materials and Textures

in the material. Materials are often classified by the general types of objects they are applied to. For example, in Source there is a Water Shader specifically designed to handle the special case of large volumes of water.

Although there are many types of properties in a given material, many materials have properties that expect textures as the input. For example, in a Standard Material, there is a property called Diffuse Color. This is the actual color of the material. For this texture, you can apply a single solid color, or you could use a map like a Bitmap (maybe a picture of your dog). Any object referencing that material will have the image you applied to the diffuse map (a solid color or your dog's photo).



In the image above, the floater on the left is the Slate Material Editor. It is a node graph that shows you the relationships of maps and materials. Notice how the Map #1 bitmap node is piped into the Diffuse Color parameter of the Material Dewey.

The material editor is a powerful tool that allows you to create complex node trees of maps to generate the textures that end up in your materials. Mastering the tool will take time and ex-

perience. For now, it is important to realize the relationships of maps, materials and objects.

- Objects can have only one material, but not a texture! Get used to the idea that you only apply materials to objects, not textures.
- Materials can have maps/textures, or be composed of other materials that use maps/textures.
- Maps/Textures can have other Maps/Textures. (This makes no sense for bitmaps, but many maps have map parameters and can be composed of many other maps.)
- When multiple materials are needed on an object, the object should use the Mult/Sub-Object material that can then store many other materials.

New Max users should watch [Autodesk's Creating Materials and Maps videos](#). The link is Part one and is followed by two videos. I suggest you watch all three videos to get a better understanding of the possibilities. Note that the video demonstrates the Slate Material Editor (SME), which was not introduced until 3ds Max 2011. All Wall Worm videos and docs refer to the SME. Most traditional Source materials refer to the older Compact Material Editor (CME). I prefer and use SME.

Setting Up Materials and Maps for Source

3ds Max is often called the Swiss Army Knife of 3D. It can do anything. As such, the material editor, maps and tools are applicable to almost any field. Not all of the materials and maps are directly relevant to the game engine, however.

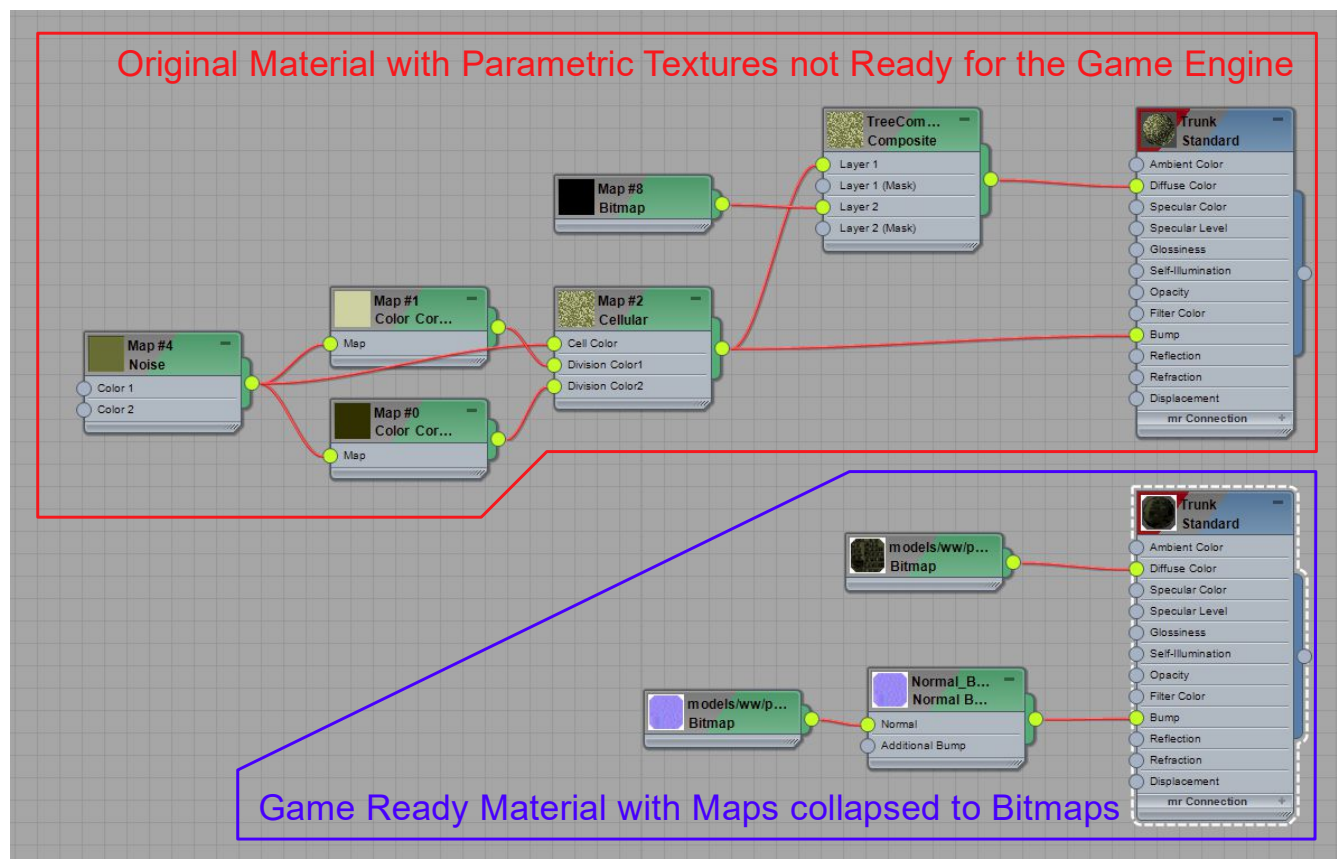
There is currently no one-to-one Max-to-Source material conversion. Instead, there is a system of importers/exporters built into Wall Worm that will convert many materials from one en-

vironment to the other. There are some considerations you must keep in mind when setting up a material to export into Source.

Although you can use any of the tools inside 3ds Max to generate the bitmaps that will export as the textures inside Source, there are only a few specific materials that will export into Source. Also, the free version of Wall Worm only exports Bitmap nodes using TGA and PSD files (where Wall Worm Pro will export almost any texture type).

While making the material and textures, you can make a material and texture system as complex as you want to get the effect that you are looking for. Once you have finished your “raw” textures, you will turn them into bitmaps that the game engine can use by one of two methods: Render to Texture (also known as RTT and/or baking) or Render Map.

The results of Render Map or RTT are bitmaps, as demonstrated in the graphic below.



Both RTT and Render Map have their place in your design process. Choosing which to use is often situational and based on experience, but here are some tips below. Just remember that for Source, you should always render bitmaps into a square dimension with a power of 2 (512x512, 1024x1024) and of the TGA file extension.

Render Map

Render Map is done inside the Material Editor. This function allows you to take any texture node and render it to a bitmap. For example, you could use the Cellular procedural texture map to generate some alien skin... then render this map to a bitmap. You can do this by right-clicking a texture node in a Slate view and choosing the Render Map function.

Often times you will use Render Map to create component bitmaps for larger systems or for making tiling textures.

Render to Texture

Render to Texture (RTT or Baking) is the process of baking complex material systems into a simple material with collapsed bitmaps. It even has the ability to project complex geometry into rendered textures with Projection Mapping, which is a way to take very complex models and bake details into textures. Using RTT is more complex than the current discussion and will be discussed in more detail later.

Available Materials

There are only a few materials that you can use for the Source Exporters, listed below:

- Standard

- Blend
- Multi/Sub-Object* (Sub-Materials)
- Shell Material* (Baked Material)
- DirectX Shader* (Render Material)

*Although there are five allowed materials above, the material that actually gets exported in those designated with asterisks are material properties that are other materials that must be a Standard or Blend. For example, a Multi/Sub-Object material is composed of multiple other Sub-Materials. When being exported, the Multi/Sub-Object material does not actually export, but instead any of its Sub-Materials that are Standard or Blend will export.

It is possible that Wall Worm will support other materials in the future. Since ShaderFX was added to 3ds Max 2015+, it is likely that some of its features will be integrated.

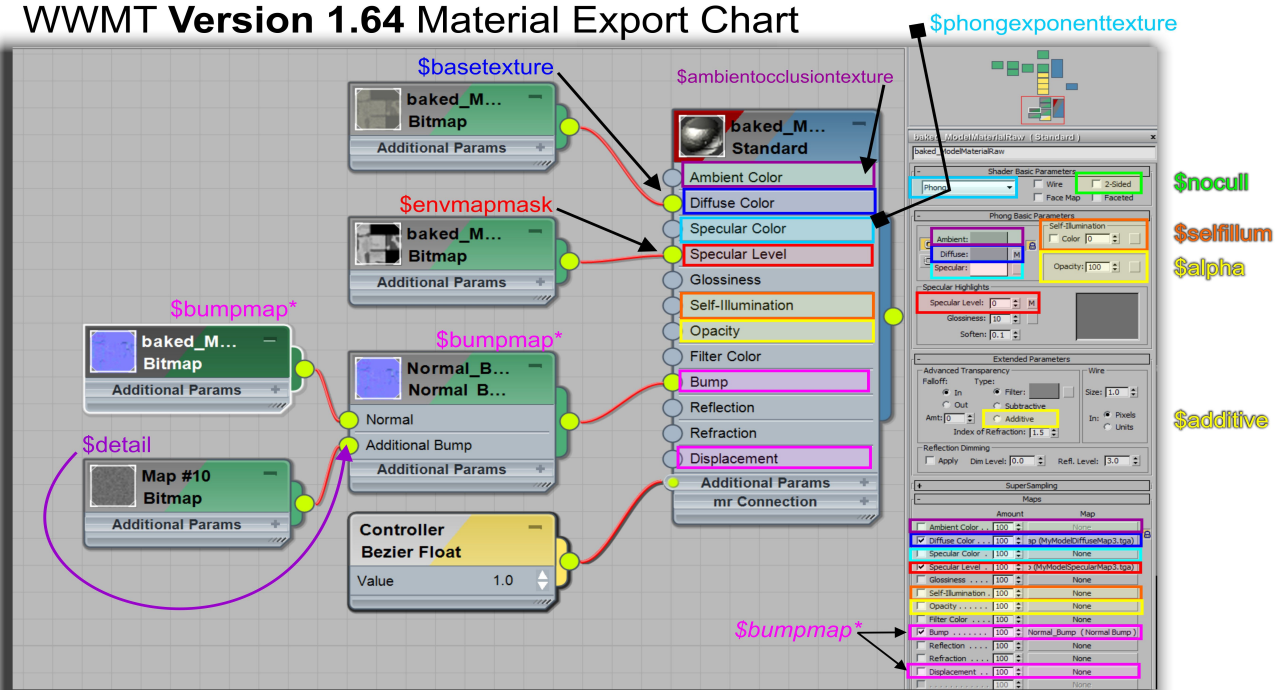
Material and Map Conventions

Wall Worm has a different way of using bitmaps and materials for material names than in legacy systems/tutorials. In WW, you refer to a VMT (material file) by the Name of a material, and not by the file name of the diffuse bitmap. VTF output paths are the names of the bitmap node in Slate (except when a WW Pro texture attribute is applied—in which case the name is the initial path but a VTF Path field controls the final output path).

The Materials and Source Shader Parameters

While many of the maps in a Standard material will export automatically into the Source Engine, the automated correlation of parameters from Max to Source is very basic. For many simple props, it is good enough to apply a diffuse texture or a couple other settings and export. A basic chart of the relationship of Standard Materials and Source Materials is demonstrated in the following graphic:

WWMT Version 1.64 Material Export Chart



Map subject to change. Refer to www.wallworm.com for latest docs.

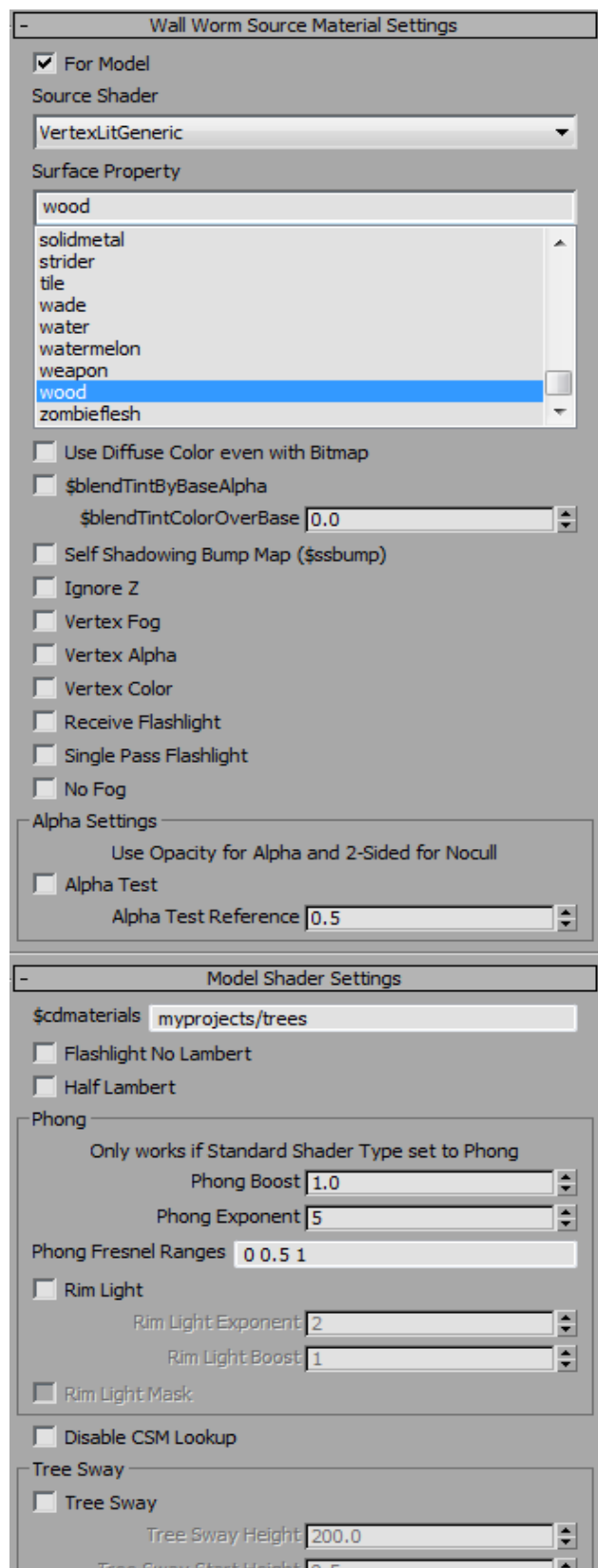
Advanced Source Material Controls

If you need more control over the material in Source, you will need to apply a Wall Worm Material custom attribute to your material. This will give you the ability to control a large percentage of the available properties in Source materials. By doing this you can control surface properties, shader-settings, decal controls, compile parameters, Phong settings, Tree Sway, flashlight settings and more.

You can apply a WW Material to your materials in two ways. You can add the parameters to the currently selected material in the Slate Material Editor, or to the currently selected objects in Max.

Apply to Materials in Material Editor

If you want to add the WW Material to a material in the Material Editor, double-click the material in a Slate View so that it is the currently active material and then click **Wall Worm > Wall Worm Materials > Add WW to Selected Mats in Editor**.



Apply to Materials of Selected Objects

To apply the WW Material to the materials of selected objects in the scene, click **Wall Worm > Wall Worm Materials > Give Obj Mats WW Materials** .

Those materials that get the WW Materials will now have a large collection of options and settings in the Material's Parameter Editor in the Material Editor. To the right is an example of some of the parameters you can set inside Max that will export into the Source Material files (VMT files). Note that the image only shows some parameters.

Note that when you add the WW Material with the methods outlined above, only Standard and Blend Materials will actually get the WW Material. For example, if adding these options to a DirectX Shader, the parameters will be applied to the Render Material of the DirectX Shader and not the DirectX Shader.

Some of the settings in the WW Material UI will enable/disable/change as the result of different settings. For example, turning on the For Model option will remove WorldVertexTransition and LightMappedGeneric from the Valve Shader menu and enable the Model Shader Settings rollout. Unchecking the For Model option disables all items in the Model Shader Settings rollout.

There is no error checking, so if you add an invalid value for an item, you will not know about the problem until you test it in-game.

Blends, WorldVertexTransition and LightMapped_4wayBlend Shader

In the Source Engine there are special materials for displacements that blend materials based on vertex alpha or vertex colors. This feature is available in 3ds Max with Wall Worm and needs a little clarification on implementation.

In Source, materials using the WorldVertexTransition shader use alpha blending; Lightmapped_4wayblend uses vertex colors. The 3ds Max equivalent are Blend and Composite materials. The Blend material uses two sub-material inputs and a mask map to control which material to show at a given location and the Composite material layers materials. Unfortunately, 3ds Max does not, by default, display vertex-alpha masks applied to a Blend material and does not display Composite materials at all. To preview these, you must use a DirectX Shader material.

Different versions of 3ds Max handle DirectX Shaders differently. As of 3ds Max 2014, DirectX 11 is supported. Previously, DirectX 9 was supported. This matters because when creating your own material manually, you need to use the correct Effect File.

Wall Worm uses four files to use for this purpose. Which one you are using is dependent on what version of Max you are using and your specific needs. You can change them in your global settings in the Materials tab.

- Native (uses a DX shader shipping with Max)*
- blend_dx11_nitrous (optional DX11 shader)
- Black Mesa (a blend shader that can show blend modulation)
- LightMapped_4wayBlend (shader to blend 4 textures at once for CS:GO and Black Mesa)

At this time, the recommended blend shader is the Black Mesa shader for WorldVertexTransition and Lightmapped_4WayBlend for multiblending (although this shader is only valid for some games).

Objects with one of these shaders will now display material blends based off of vertex alpha values as the mask or vertex colors. This is used extensively on Displacements, which are discussed in the next chapter.

Note that when you change the global shader setting, it will only affect newly generated blend materials—it won't change the shader on existing materials.

Convert Blend/Composite to DX Shader

You can create a DX Shader for an existing object in the scene:

1. **Select** one or more objects that have a **Blend** material or **Composite** Material
2. Click Wall Worm > Wall Worm Materials > **Convert Blend/Composite to DX Shaders**

Once done, the objects' current materials will be put into the Render Material of the DX shader(s) and the DX shader(s) will be assigned to the object(s). Note that for a Composite Material, only the first four material slots are used.

Model Materials VS World Materials

Generally speaking, Wall Worm references a material based off of the name of the material. So if the **material's name** in the Material Editor is “myprojects/walls/brick1” then the material exporter will create a material file (**brick1.vmt**) inside the **walls** folder of the **myprojects** folder of the **materials** folder.

This is always how it works for materials exporting for world geometry shaders. However, there is some fuzziness with Model Materials. Model materials do not always require you to

write a path to the VMT in the material name because the path is part of the WWMT Helper settings—and when the WWMT Texture export function is run, the path in the WWMT settings is used. The problem is when you export a VMT with the **Write Material File** button in the Miscellaneous rollout of any material having the Wall Worm Source Material Settings: when using this function for model materials, you must use the full output path in the material's name or add the output path to the \$cdmaterials property in the Model Shader Settings in the material.

Export Model Materials

There are a few methods for exporting your model's materials and textures. The primary way is to click the Export VTFs button in WWMT. This will bring up a dialog showing all the model's materials and associated textures. Any material that has not been converted to a VMT, and any TGA/PSD that hasn't been converted to a VTF, will have the checkbox pre-checked to let you know that it must be exported. If you need to update an existing VMT/VTF, simply check its corresponding box.

You can also batch export model materials by selecting a set of WWMT Helpers in the scene and clicking **Wall Worm > Wall Worm Exporters > Export Selected Model Textures**. This function does not bring up any dialog but simply exports all VMT/VTFs for the models selected.

Export Brush and Displacement Materials

Exporting materials for brushes and displacements is as simple as selecting a set of objects in the scene and clicking **Wall Worm > Wall Worm Exporters > Export Brush Textures**. You will get a dialog similar to that used with model materials. (If no objects are selected, then all objects are used for collecting and exporting materials.)

Worm Face (Hammer-like Texture Application)

Worm Face is a tool to drop decals or to lift textures from picked faces in the scene and dropping them onto other faces. You can open it by clicking **Wall Worm > Wall Worm Materials > Worm Face**.

Set Worm Face Material

1. Open **Worm Face**.
2. Set **Mode** to **Texture Drop**.
3. Click **Start**.
4. **Alt+LMB** any face on any object in the scene with a material.

Drop Material on a Face

1. First set the Worm Face material (above).
2. Click on any face in the scene.

The Material Drop only works on Editable Poly and Editable Mesh objects. Any other object must be converted to an Editable Poly. If you want to automate this, press the Convert to Poly button in Worm Face before dropping a material. To end the drop function, right-click the viewport.

Working With Textures

The textures you use in your materials are generally bitmap textures (and for standard Wall Worm, they should always be TGA bitmaps). The tools and methods for working with textures depend on whether you are using Wall Worm or Wall Worm Pro. Wall Worm converts TGA bitmaps into VTF (Valve Texture Files) via an application that comes with your game called vtex.exe. VTEX is limited. When using Wall Worm Pro, the application used is called VTFCMD (developed and provided by Ryan Gregg and Neil Jedrzejewski).

The output path of a VTF is determined by the Bitmap Node's name in Slate. That is always the case for Wall Worm, *but for Wall Worm Pro, it's just the default path*. This means that if your bitmap node's name is “myproject/buildings/” and the bitmap file name is “brick5.tga” then the final output of the VTF will be in game/materials/myproject/buildings/brick5.vtf.

Advanced Settings (Wall Worm Pro)

Wall Worm Pro has tools to control almost all possible settings of your VTF directly inside 3ds Max, including compression levels, texture flags, normal settings, etc. Moreover, with Wall Worm Pro, you can export any arbitrary (and procedural) texture as a VTF, not just bitmap textures. See [Advanced Texture Controls in conjunction with Substance Maps](#).

To assign these parameters to a texture node, select an object in the scene with the materials that include the texture and then click **Wall Worm > Wall Worm Materials > Give Obj Mats + Tex WW Materials**. When you do this, WW Pro adds a custom attribute to each input texture of the material that will now export as a VTF. The advanced settings can now be controlled by double-clicking the texture in the Slate view and editing them in the material editor.

The new settings now have a property called Path. This lets you explicitly set the output path and file name. Its default is derived from the texture's name for procedural textures and the

texture name plus bitmap file name for bitmaps. Once this path is set, neither the texture node's name or the bitmap file (if present) affect the output VTF—only the path setting will affect it.

- You can export these textures inside the material editor with the Export Texture as VTF button.
- For compression, use DXT1 for textures that don't need alpha; DXT3 for textures that have simple alpha; DXT5 for those that need high quality alpha.

Bitmaps to Materials

There is a function in Wall Worm to help you create your own material library from your own photos very quickly. This function requires you to own Allegorithmic's Bitmap2Material and Wall Worm Pro. To see this function in action, [watch the Bitmaps to 3ds Max Material Library video](#).

- While this function can utilize B2M 3, for Source Engine work you should set the tool to use B2M 2.
- The tool automatically generates a Substance tree that outputs Diffuse, Specular and Normal textures in your material.
- Each bitmap found in the generation function will convert into a material.

Tri-Planar Mapping inside Max

There is also a new function inside the Bitmap2Material floater that will allow you to create

seamless materials that use texture coordinates that are not bound to an objects UVs. This is very helpful when creating complex procedural textures for your high-poly meshes. See the [Bitmaps to Materials documentation](#) for more information on this and the above functions.

Note that in 3ds Max 2017.1+ there is a native texture map called BlendedBoxMap.

Chapter 10 Displacements

Displacements in Source are meshes that make up organic terrains. Almost all out-door landscaping and underground tunnels in Source environments are composed of displacements. What displacements are, do, and are exported as, is identical between 3ds Max and Hammer. But the techniques for creating and editing them are not similar between Max and Hammer.



In Hammer, displacements are always created from a brush face that has four sides. That is the only method of creating displacements in Hammer. In 3ds Max, you can make a displacement from 1) any four-sided surface (not just brushes); 2) any Plane primitive with length/width segments of 4, 8 or 16; 3) via the Create Displacement at Helper function.

Illustration 3: Displacement terrain and roads made with Wall Worm Displacements. Provided by Til Sichel.

Understanding Wall Worm Displacements

Wall Worm has three basic objects that control displacements. Understanding these objects is important in effectively using WW to create your landscape. Also please be aware that there was a major update to how displacements work in WW version 2.82+. That change is noted at the end of this chapter.

The three Displacement objects WW creates are:

1. Displacement Meshes
2. Brush Face Meshes
3. Sculpt Meshes

Each time you create a Displacement with Wall Worm, you will always be creating a linked pair that includes a Displacement Mesh and Brush Face Mesh. From here on out, I'll refer to them as the displacement and the brush face. Normally, the displacement is the one that you see immediately and the brush face is hidden.

Deleting either one of these two objects will immediately delete the other!

The displacement is a special Editable Poly object that can be edited with various tools in Max; the best tool set for sculpting displacements is probably the Graphite Modeling Tools in later versions of Max. You can edit the displacement with Push/Pull, Relax and other tools. You can also manipulate at different sub-object levels like Vertex, Face, etc. You can also modifiers that do not modify the vertex and face count/order (such as Relax, Displace, Push, etc).

When created, the displacement's transformations are locked and you cannot by default move the displacement. Generally speaking, you should not move a displacement. Instead, you should move the brush face. You can get the displacement's brush face by clicking the Move Mode in the modify tab of the command panel, or click Move Mode in Anvil.

The displacement is a child of the brush face in Max... so moving the brush will automatically move the displacement. But the reverse is not true.

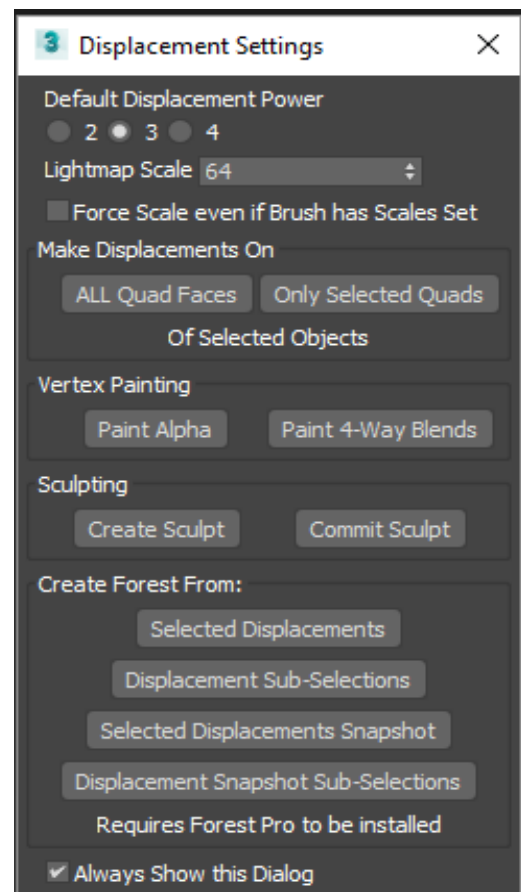
You should never edit the brush face mesh in any way whatsoever except to change its position in the scene. You can also rotate it, but rotation is not recommended. **Do not scale the brush mesh or move any of its vertices!**

Creating Displacements

There are several ways to create displacements in Wall Worm. The best way is to do it in the traditional Hammer method by creating them from brushes, as this will mean that there is an actual brush rather than a virtual one when exported to the VMF. Preferably the brushes should be clean and snapped to the world grid.

1. **Select** brushes in the scene
2. **Add a Poly Select Modifier** to the selection
3. Enter **Polygon Sub-Object mode**
4. **Select** all **sides** that you want to create a displacement from.
5. **Click** Wall Worm > Wall Worm Level Design > **Launch Displacement Floater**
6. **Click** the button **Only Selected Quads**

Now Wall Worm will create displacements from all the faces and hide the brush. Each displacement will derive the power and lightmap scale from the displacement floater. Instead of launching the Displacement



Floater, you could skip this and instead click Wall Worm > Wall Worm Level Design > **Create Displacement from Face Selections**. Note that that function will also launch the Displacement Floater if the global setting for always launching the display floater is turned on (which you can turn off with the Always Show this Dialog check box).

The Sculpt Mesh

Although it might seem natural to start editing displacements as they are in the scene, you should make a habit of not actually editing and working with individual displacements. Instead, you should use a sculpt mesh. Generally speaking, the only edits you should do to individual displacements is to use the Sew function.

If your scene is composed of multiple displacements, you can consider using one or more Sculpt Mesh(es). Sculpt meshes are special Editable Poly objects that are generated from the surface of a collection of multiple displacement meshes where the vertices of the sculpt mesh are mapped to the original displacement vertices.

Create a Sculpt Mesh

To create a sculpt mesh:

1. Select some displacements in the scene. (If you only select one or zero objects, Wall Worm will create a sculpt from all displacements).
2. Click Wall Worm > Wall Worm Level Design > **Create Sculpt Mesh**

WW will create the sculpt mesh as well as hide and lock the underlying displacements and brushes. Depending on how many displacements are used, what power those displacements

are and how fast your computer is, it could take a second to several minutes to create the sculpt mesh.

Once generated, you can now sculpt as you would a single displacement, but you can now treat the collection of displacements as a single object. This has many benefits, especially when using tools like Relax or when using Modifiers.

The underlying displacements do not get updated until you click the Commit button in the modify tab of the command panel when you have the displacement sculpt mesh selected.

You can now have any number of sculpt meshes, and they never need to be destroyed; the only thing you have to remember is that each sculpt mesh needs to be committed for any changes applied to it are extended to the underlying displacements. Deleting the sculpt mesh will unhide the underlying displacements, which will be at the last committed state of the deleted sculpt mesh.

For convenience, you may find it good practice to make sculpt meshes of any groups of displacements that are in separate areas of your scene.

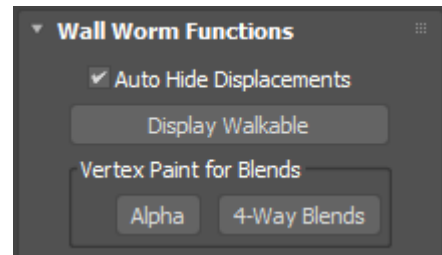
Watch this [Wall Worm Displacement Video](#) for an in-depth discussion and demonstration.

Sculpt Mesh Functions

There are many functions specifically for working with sculpt meshes. You can get to them by selecting a sculpt mesh and opening the Modify Tab. The sculpt mesh functions are in the roll-out labeled “Wall Worm Functions.”

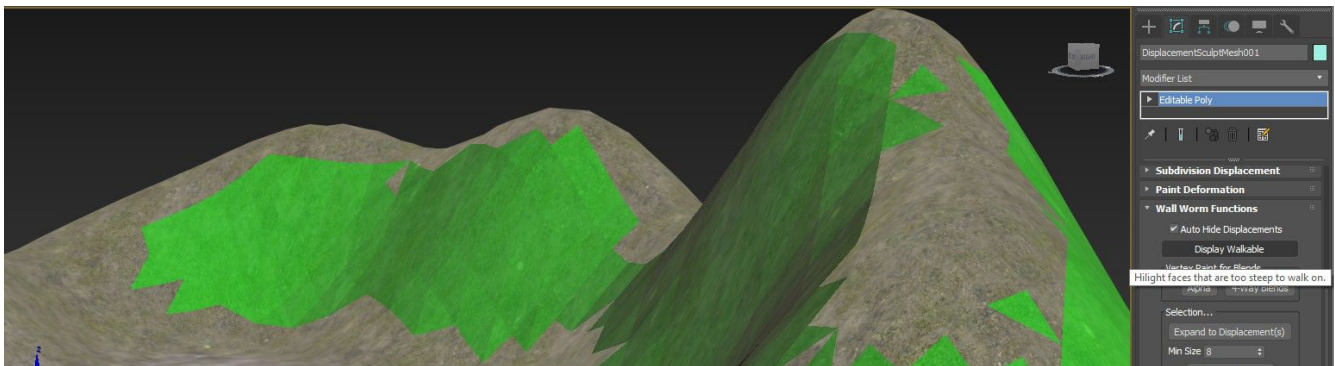
Auto Hide Displacements

This option will automatically hide all displacements that compose the sculpt mesh whenever you select the sculpt mesh and open the modify tab. This option is useful to help you from accidentally editing the displacements instead of the sculpt mesh.



Display Walkable

The display walkable button will run the Display Walkable Xview to high-lite faces that a player cannot walk on due to the face angle being too steep. The image below shows the display walkable Xview highlighting the faces a player cannot walk on in green.



Vertex Paint for Blends

The blending of materials on displacements and sculpt meshes is done via Vertex Painting. In 3ds Max vertex painting is done via the Vertex Paint modifier. Max has many channels that

Displacements

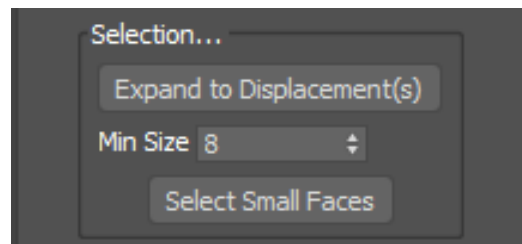
you can paint onto. Wall Worm uses the Vertex Alpha channel for blend materials (WorldVertexTransition) and channel 10 for Lightmapped_4wayblend materials.

Alpha: Press this button to paint on the alpha channel (-2).

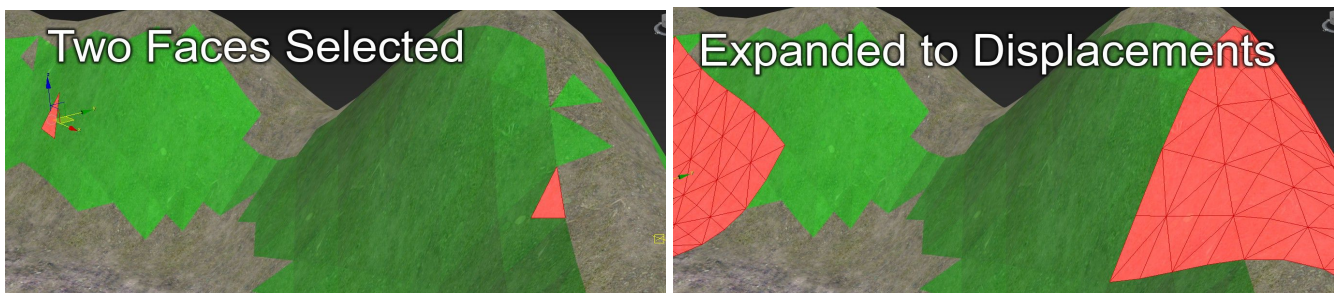
4-Way Blends: Press this button to paint 4-way blends (channel 10).

Selection Functions

The Selection functions allow you to select faces that are relevant to working with a sculpt mesh.



Expand to Displacement(s): This button will take the current face or vertex sub-selection and expand them to include all the faces that are part of the same underlying displacements. See the example images below:

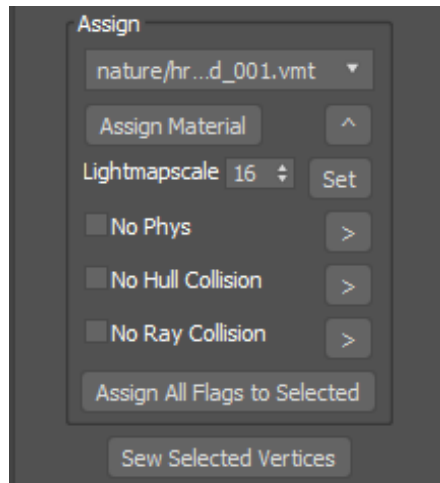


Select Small Faces: This button will select all faces in the sculpt mesh that have a surface area below the minimum size. This helps you isolate areas that can cause problems. (You do not want zero-area surfaces.)

Assign Functions

The assign functions allow you to make changes to the properties of the displacements that belong to the displacements. *All of these functions work on the current sub-object selection to determine what displacements should be updated with the new values.*

Assign Material: This button will set the material of the selected displacements to the one selected in the material list. Note that the material list is populated by the current multi-material assigned to the sculpt mesh, so to add new materials you need to assign them in the material editor. New materials will not show up in the list until you deselect/reselect the sculpt mesh or hit the button labeled “^” to the right of the Assign Material button. *Hint: Since material names may not fit in the drop-down UI's space, the full path of a material is printed in the MAXScript listener when you select a material from the drop-down.*



Lightmap Scale: The button labeled “Set” will assign the lightmap scale from the spinner to all selected displacements.

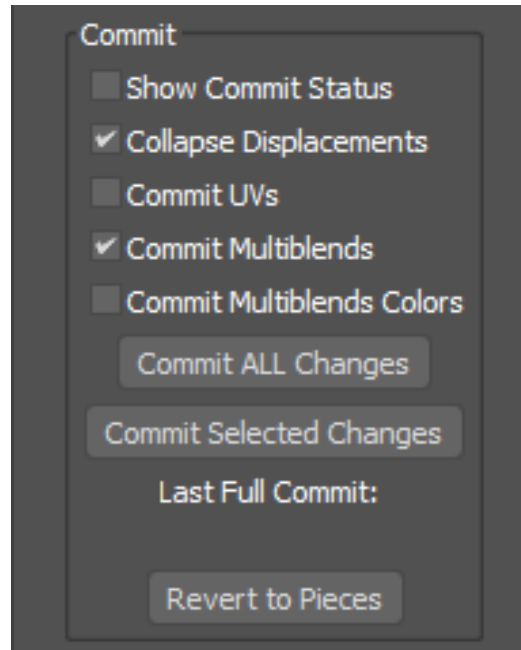
Flags: You can explicitly set the flags for **No Phys**, **No Hull Collision** and **No Ray Collision** to selected displacements. The buttons to the right of each will assign that particular flag setting to the selected displacements. The **Assign All Flags to Selected** button will assign all three flag settings at once to all selected displacements.

Sew Selected Vertices:

This button will move all selected vertices to their average position. This function can be useful if the sculpt mesh was not created from cleanly sewn displacements.

Commit

The commit section deals with updating the actual displacements so that their vertex positions and alpha blending will match the sculpt mesh. Remember that changes made to a sculpt mesh do not propagate to the underlying displacements until the sculpt mesh has been committed. This is important because it is the displacements, not the sculpt mesh, that gets committed to game.



Show Commit Status

This option will force the Max status line (below the viewports) to update during the commit of a sculpt mesh. This information can show you how far along a commit is. This option does have a minor performance hit and also can cause some confusion because it forces the Max UI to stay active during the commit. When using this option, do not edit anything in the scene until the commit is finished.

Collapse Displacements

When on, this option will convert each displacement to an editable poly with no modifiers. If off, the modifiers are not collapsed.

Commit UVs

When on, the commit function will update the underlying displacement's UVs with those of the

sculpt mesh. You should never turn this on if you have used the Quadrify Me function.

Commit Multiblends

This option will bake Channel 10 into the displacements, which is what the Lightmapped_4WayBlend shader uses. Not necessary if the sculpt mesh is using only LightmappedGeneric and WorldVertexTransition.

Commit Multiblends Colors

This will commit channels 12-15. Not used in most games and should probably be turned off as the more channels committed, the slower the commit function is.

Commit All Changes

This button will update all the displacements composing the sculpt mesh. For displacements with hundreds of displacements, this can take a while.

Commit Selected Changes

This button will only commit changes to the displacements belonging to the current sub-object selection. For a large sculpt mesh, using this button in areas you are working on will speed up commits as it won't take time processing areas that haven't been changed or do not need committed at this time.

Revert to Pieces

This button will destroy the sculpt mesh and unhide all of the displacements. When you press this button you will be prompted to simply delete the sculpt mesh or to first commit the sculpt mesh. If you choose not to commit, any changes to the sculpt mesh since the last commit will be lost. (If you have not made any changes to the sculpt mesh since created or since the last commit, nothing will be lost.)

Sculpt Mesh Modify Functions

These functions allow you to add or remove displacements from a sculpt mesh and to split a sculpt mesh into multiple sculpt meshes.

Add Displacement

When you press this button, you can select a displacement in the scene that doesn't already belong to this sculpt mesh to add it into the sculpt mesh.

Remove Selected (Displacement)

This button will remove the displacements belonging to the current face selection from the sculpt mesh.

Remove and Delete

This button will remove the displacements belonging to the selected faces of the sculpt mesh, then delete from the scene.

Remove to New Sculpt

This button will remove the displacements of the selected faces from this sculpt mesh and create an entirely new sculpt mesh from them. You might use this to break up different areas of a level that you intend to work on separately.

Sculpt Mesh Miscellaneous Functions

This group of functions are generally not used much except for the Launch Displacement Floater.

Launch Displacement Floater

This will launch the Displacement floater which can be used to create new displacements or run useful functions that utilize Itoo's Forest plugin.

Bake Info

This will bake all of the information for the sculpt mesh's displacements into each of those displacements. This will speed up VMF export times but it will also increase file size. Because the information is baked, it means that new changes to the displacement(s) will not take effect unless the data is cleared.

Clear Bake

This method clears all the baked information in the sculpt mesh's displacements, allowing them to accept changes to the displacements.

Send Material to Pieces

This function will update the materials used on all the displacements to match those used in the sculpt mesh at the location representing that displacement.

Fix Alphas of Displacements

This function will add map channel -2 (Alpha Channel) to all the displacements. This is useful if the displacements are missing their alpha channel but are using WorldVertexTransition DX shaders.

Collapse on Quadrify

This setting relates to the Quadrify Me function (below). When on, the sculpt mesh is collapsed to an Editable Poly when quadrified.

Quadrify Me

This function will quadrify the sculpt mesh (removing the triangulation). Being quadrified is useful for having access to many sub-object selection shortcuts available to Editable Poly objects. There are some caveats with the quadrify function:

- Never use the Native Quadrify All function in Max. Doing so will invalidate your sculpt mesh and mangle your displacements!
- Only use the Quadrify Me function if your shader is using Seamless Scale. This is because the UVs are not properly kept (see next item). Seamless scale will not care about the UVs.

Displacements

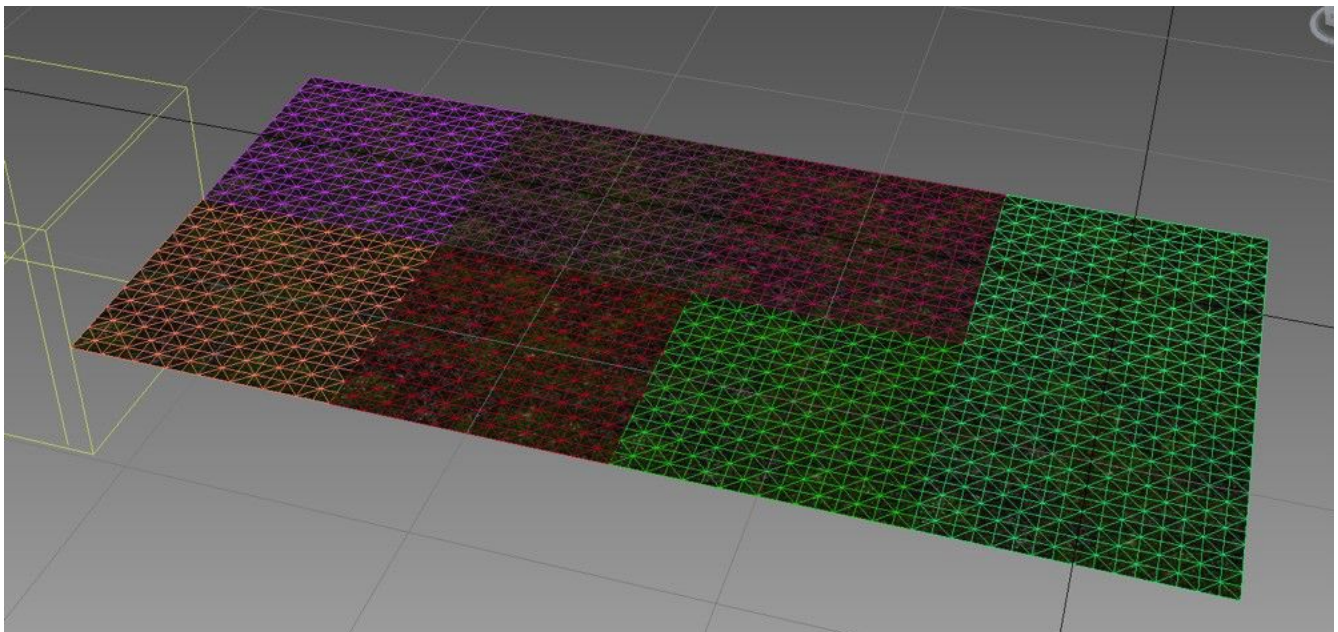
- When using the quadrify function, do not turn on the Community Uvs option (above) as the quadrify function does not properly keep the Uvs.

Triangulate Me

At this time, this function will not work.

Displacement Examples

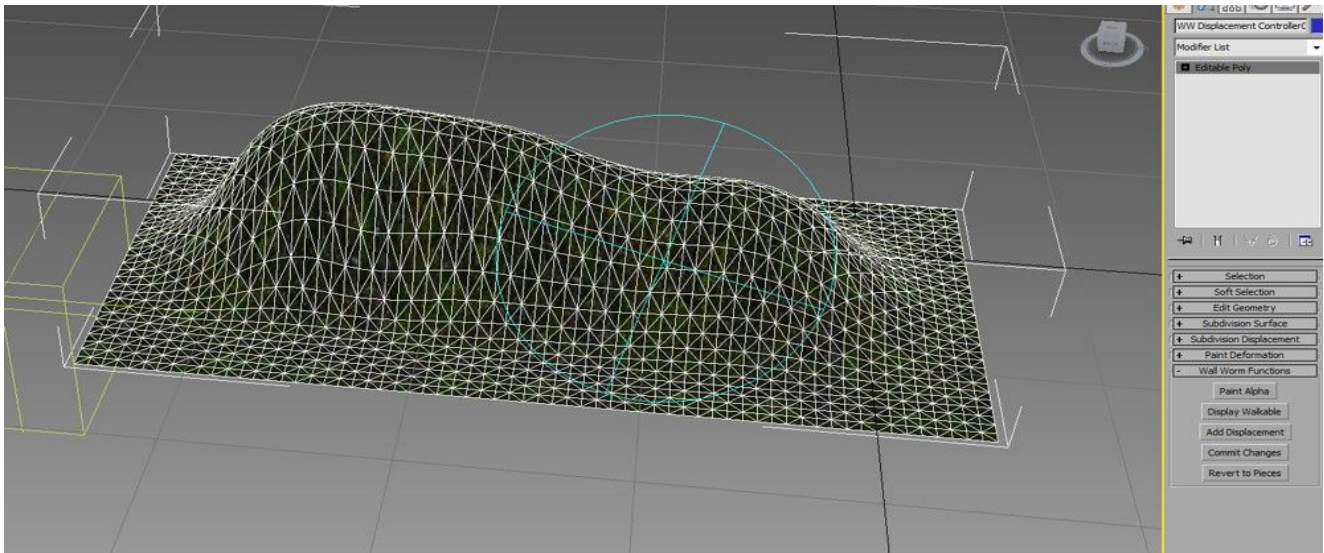
Below, eight (8) displacement pieces are created with Anvil. The displacements are separate objects that can be edited like a standard Editable Poly object. It is best to not move these objects—instead move the underlying (hidden) brush mesh by clicking the Move Mode button in the Modify Tab.



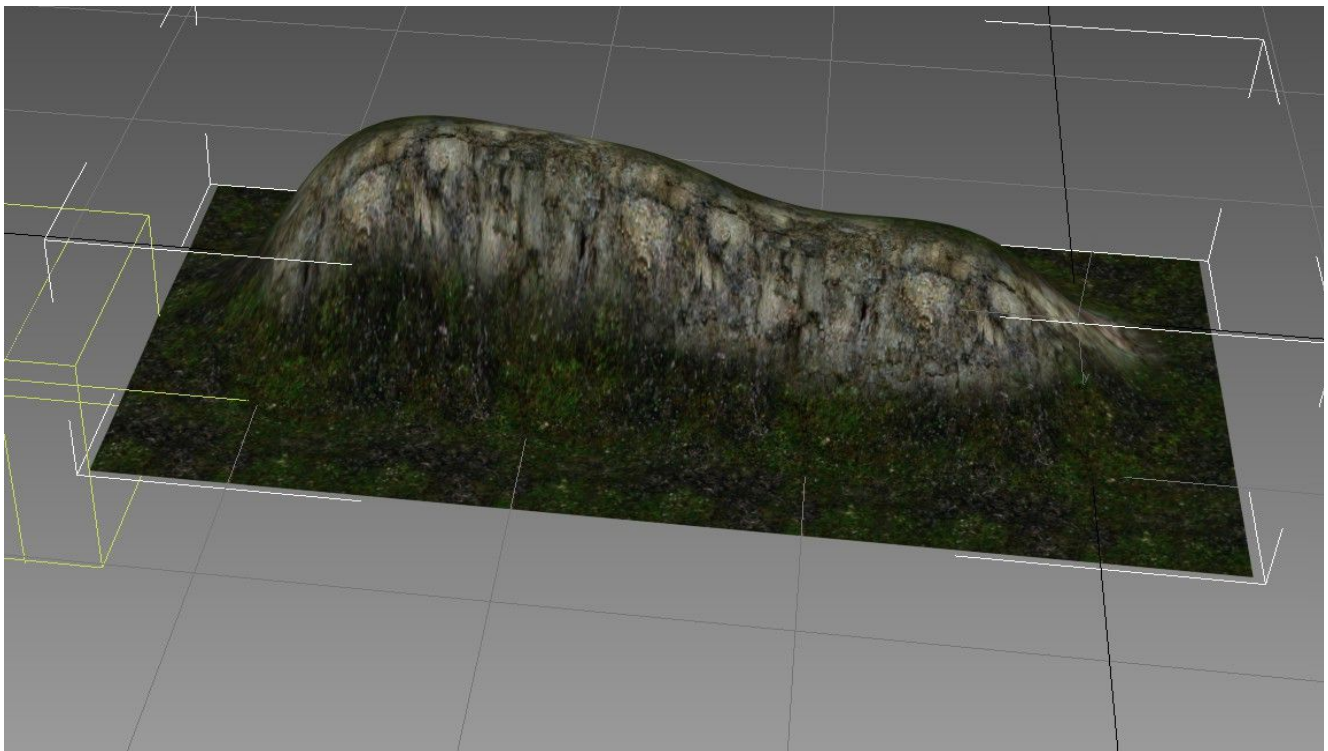
In the next image, the displacements have been converted into a sculpt mesh and sculpted

Displacements

with the Push/Pull Tool.

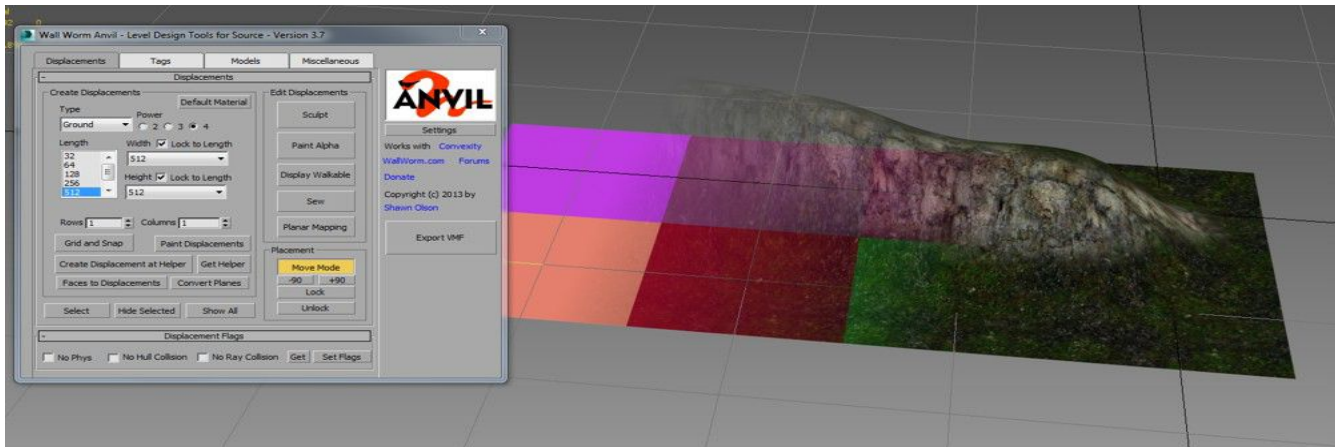


Below is an example of two textures blended by the vertex alpha. This is accomplished with the Vertex Paint modifier or clicking the Paint Alpha button in the Modify Tab.



Displacements

Below, the sculpt mesh has been reverted back into its constituent displacement pieces. The Move Mode in Anvil was pressed to hide this displacements and allow movement. (The image is a composite of Move Mode and non-Move Mode so that you can see the relationship of a displacement and the brush face mesh.)



One final reminder—**do not add/remove any vertices or polygons** on any displacement mesh, brush mesh or sculpt mesh. Feel free to *move* them around as needed. But changing topology (manually or via modifiers) will invalidate your displacements!

Changing Power of Displacements

To change the power of a displacement in the scene (increase/decrease the resolution), you must select the displacements you want to change, open Anvil, choose the power you want to change to and click the **mod** button in the Displacement tab.

- You can only change the power of displacements that are not currently part of a Sculpt Mesh.

Displacement UVs

Displacement UVs should never be modified per vertex. Instead, they should only be edited per displacement. The best way to manipulate a displacement's UVs is with a UVW Xform modifier.

Clipping Displacements

Wall Worm does not currently offer the ability to clip/slice an existing displacement! Avoid scenarios where this is required.

Controlling the Brushes Behind Displacements

In the Source Engine, displacements must be derived from brushes that are removed from the final game. Depending on how the displacements are created in Wall Worm, the displacements may or may not actually have brushes that control them inside the 3ds Max scene. For any displacement not tied to a brush (and a specific brush side), the Wall Worm VMF Exporter automatically generates a virtual brush for that displacement. For many cases, this is adequate. However, there are cases where you may want to control specifically the brush that makes up a displacement or to bind several displacements to a single brush node.

If you use the Polygons to Displacements function, Wall Worm will bind the source object to the displacement as its source. If that object is tagged as a brush (and is a valid brush or valid concave brush) then the displacement will use that brush as its source. Note that if you edit that brush after displacements are created, you can invalidate the displacement, so it's best to not edit a brush after you've generated a displacement for it.

Manually Assign Brush Face to Displacement

You can also manually assign a displacement to a brush side after-the-fact. You need to know both the brush object and the side number ahead of time.

1. Select the displacement.
2. Click Move Mode (which selects the Displacement's Brush Face representation).
3. Type into MAXScript listener: `$.sourceNode = $nodeName` (where \$nodename is name of brush object)
4. Type into MAXScript listener: `$.sourceFace = 1` (where this number is the actual face you want to use).

Advanced Displacement Discussions

Working with Displacements in Max can offer a wider degree of freedom and efficiency than available in Hammer. Understanding the available tools will help you create more interesting environments.

Parametric Displacements

One really cool method of generating landscapes is to utilize textures as the source for blending materials and displacing the landscape. For example, you can use a height map to control both the texture blending and the actual displacements.

Use Height Map to Control Displacements and Blending

A really convenient way to make a landscape always match the texture blending with the contours of the land is to control this with a single texture. The following instructions will work with 3ds Max 2015+, Wall Worm 3.731+ and requires that you have installed the free plugin [MapToVColor from RPManager](#).

1. **Open the Material Editor**
2. **Add a texture** map that will control the displacement and the blending (this can be either a procedural map like Cellular or a Bitmap)
3. **Select** your **Sculpt Mesh** (preferably one that is Planar)
4. If the Sculpt Mesh does not have a DirectX Shader for blending already applied, add a blend material to the displacement and click Wall Worm > Wall Worm Materials > **Convert Blend/Composite to DX Shaders**.
5. **Add a Displace Modifier** to the Sculpt Mesh
6. **Drag** the texture from the material editor to the Map slot of the Displace Modifier
7. Change Amount spinner in the Displace Modifier to create the desired effect
8. Add a MapToVColor modifier to the Sculpt Mesh
9. Drag the texture from the material editor to the Map slot of the MapToVColor Modifier
10. Add a ChannelMod Modifier to the Sculpt Mesh

11. If the texture used as a heightmap is procedural, change values in the material editor to see the landscape and blending update in realtime.*

* If you do not have the ChannelMod (which is only available in Max 2015+) then you can still utilize this technique with some extra steps. In Max 2014 and older, you will have to use Tools > Channel Info and copy/paste from the Vertex Color Channel to the Vertex Alpha channel after each change of the texture.

Important Changes in Displacements (WW 2.82+)

Wall Worm version 2.82 introduced a significant overhaul in how displacements are generated, saved and manipulated. This change vastly increased the speed at which sculpt meshes are created and committed, and it reduced file sizes for scenes using sculpt meshes. But a very significant new limitation was introduced: *Displacements and Sculpt Meshes no longer support quadrified polygons!* This means that you should never remove edges. (Previously, you could remove edges to convert triangulated meshes into Quad polygons. This is no longer supported). This is very important. **Never change the face count or take any action to change face vertex ordering in displacements or sculpt meshes. Doing so will now break the displacements entirely.**

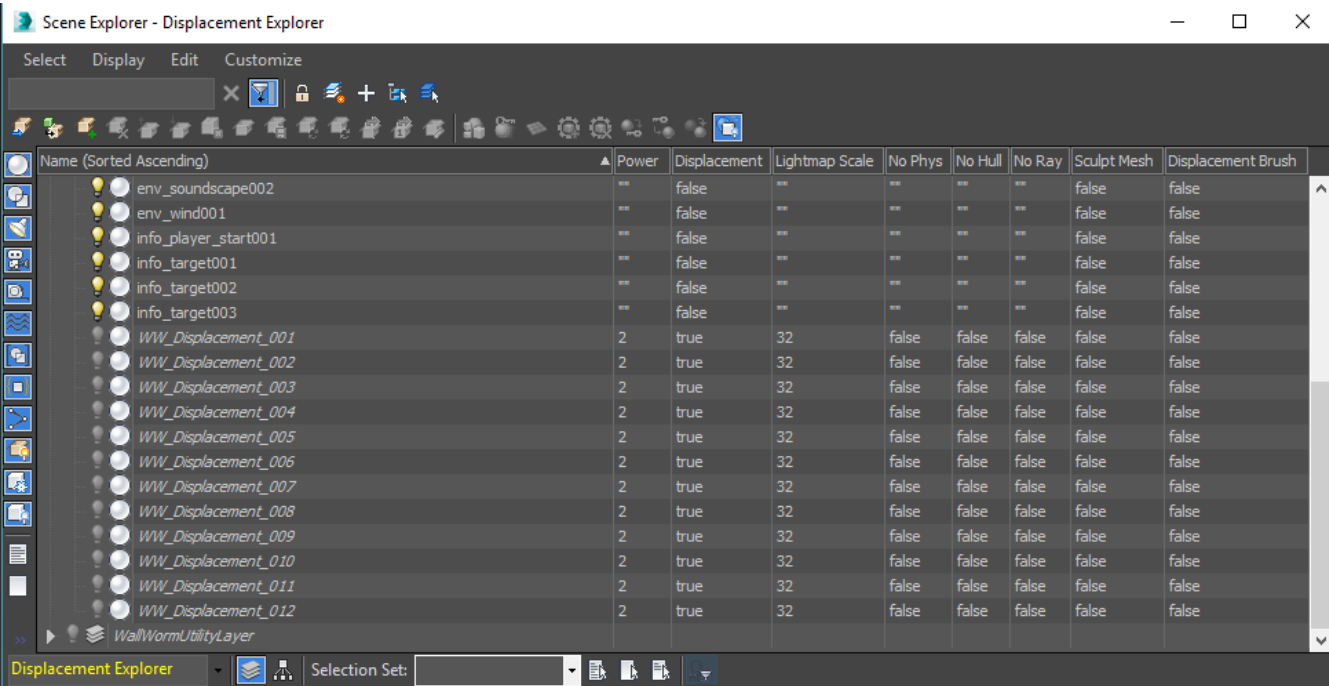
If you want to quadrify a sculpt mesh, use the Quadrify function in the Wall Worm Functions in the modify tab. Using any other method will break the displacements. Warning: A Quadrified sculpt mesh may not show accurate results with the Display Walkable xView and this function cannot be undone. If you have quadrified a sculpt mesh and wish to return it to a triangulated state, you must revert it to pieces and recreate the sculpt mesh from the displacements.

Displacement Explorer

The Displacement Explorer is a special scene explorer that will let you see displacement

Displacements

properties of objects in the scene. To open it click Wall Worm > Wall Worm Level Design > **Displacement Explorer**.



Chapter 11 Skies

Making a custom sky for your level is an important part of creating the perfect mood of your scene. If you are new to Source you may be a little bit confused about what goes into making your sky.

The first thing to understand is that there are two completely different aspects of your sky. One is the 3D Skybox that is the viewable extended world beyond the playable areas and the other is the 2D sky that is a giant cube map image that is the backdrop for the scene (usually encompassing the sky, clouds, stars, sun, etc).

This chapter explains how to build these two parts of your level in a much more convenient and optimized fashion than is done in traditional work flows. Wall Worm allows you to start your skies earlier in the design process (really from the start) and keep them as seamless environments that are easier to adjust and visualize throughout the entire process.

Getting Started with your Sky

When designing a level with Wall Worm, it is important to realize that you can start the 2d/3d sky process from the very beginning. You can (and maybe should) think of your 3D Sky from the very moment you start setting down your actual level layout. In fact, with Wall Worm, you design your 3D Skybox at scale as part of your level as if it were simply part of the playable level. It will remain at scale as a normal aspect of your level (that players can't get to).

This concept is very important, and goes against the traditional way of building 3D Skyboxes in Source inside Hammer. But it is vital to understand this approach.

Initial Setup

To make the process most efficient, you may want to follow these tips.

1. Make three layers in Max: Layout, 3D Sky and Sky Writer.
2. Design the playable area inside the Layout layer.
3. Design the 3D Skybox layout in the 3D Sky layer.
4. Design your renderable sky objects in the Sky Writer layer.

These layers and conventions are not required for making your skies. But they are helpful in organizing your scene and understanding how the system works. If you keep this convention, then you can hide the 3D Sky objects conveniently by hiding that single layer, for example. (You may choose to use even more layers in your scene but it is probably best to at least use one layer for the 3D Sky and one for Sky Writer.)

It is important to note that the objects in the Sky Writer layer are never going to be exported in the game (or shouldn't be, at least), as the purpose of that layer is to simply build terrain, oceans, skies, etc that are even beyond the 3D Skybox. The objects in the Sky Writer layer do not have restrictions on texture sizes, poly count, etc as the net result is that the objects in that layer are going to be rendered as a sky texture.

Hopefully you are starting to get a feeling for how convenient this setup is for you. It is very easy to build a cohesive environment in this fashion because the playable area, 3D skybox and rendered sky are all designed in place, right along each other.

Lighting

As soon as you want to start thinking about lighting your sky, you can add one of the following types of lights to your scene:

- mr Sun
- Daylight System
- Sunlight System
- Free Direct
- Target Direct
- Dreamscape Sun

Any one of these lights will export as a `light_environment` entity driven by the parameters in the light object. This light should be added *to the playable area* in the Layout layer (or other layer for entities, etc), and the earlier you add it to the scene the earlier you can get a sense of the scene lighting/shadows in the Max viewport and the sooner you can render out your sky.

Because your primary light source (Sun, etc) is being used for both the level lighting and the sky writer render, your sky and lighting will always make sense. In other words, you won't have to worry about aligning the angle of your sun to the predefined angle that you generally get when using stock skies or making the sky the old fashioned way.

If working with a team, you may actually have all the Sky Writer level in a separate Max file

where the main level and Sky Writer files are interlinked as XRef Scenes. This allows the landscape artist to design potentially expansive extended environments that do not necessarily balloon the level's file size and can be worked on independently and in tandem.

3D Sky Layout

The same principles of sealing your scene (and 3D Skybox) with brush geometry applies to Wall Worm as it does in Hammer. You may want to seal your playable area first. Then freeze your Layout layer (via the freeze button in the layer manager) to make it obvious what objects are playable and what are in the 3D skybox. Finally, switch the active layer to the 3D Skybox layer and design your layout.

When making your layout, remember to turn on Grid Snapping in Max. You may want to consider purchasing the [CorVex](#) developed by Wall Worm as it makes the blocking and layout stage quicker. The boundary of your 3D Skybox should be composed of brush geometry that has a material named "tools/toolsskybox". Simply making a blue Standard material and naming it this will be good enough. Use this same material on brush geometry in the Layout layer that should see any of the 3D Sky or the Sky Writer layer.

Remember that the level cannot have a leak in the Layout layer or the 3D Skybox layer. Depending on the version of Source and various compile parameters, trying to export and compile a level with a leak will either create bad lighting, bad sky effects and/or compile failure. If you do not fully understand this principle, you should learn about leaks here:

<https://developer.valvesoftware.com/wiki/Leak>

Being meticulous with the layout will help you avoid leaks, but if you do create one, you can load the leak file in Max under **Wall Worm > Wall Worm Level Design > Wall Worm Map Compile Tools > Load Leak File**.

Making your 3D Skybox

As noted earlier, Wall Worm allows you to start designing your **3D Skybox** from the very start of your project. This offers more versatility than traditional Hammer methods because you can get an idea of how an environment will look early on and find potential problems that can be a nightmare to modify if you wait until the end to finish.

As noted in the previous article, you should make a habit of putting all of your 3D Skybox objects into a layer called 3D Skybox.

Your 3D Skybox objects should never be scaled down as they are in the Hammer method. Instead, all you need to do is tag objects in Anvil as Skybox objects. You can do this two ways:

- **Anvil > Tags > Add**
- **Wall Worm > Wall Worm Level Design > Set Selection as Skybox Item**

When you call that function, all objects currently selected will be tagged as skybox objects. You can call this on entities, WWMT helpers and brushes. Remember that brushes need to be tagged as World Geometry which is located in similar menus as the skybox functions. Brush geometry has a special function to tag as both world geometry and a sky object at once: Add as Sky and Brush (in Anvil).

Now, one last thing to do is add your Sky Camera. Do this in Anvil by pressing the Get/Set Sky Cam in the Tags tab.

This will create a sky_camera entity in the scene. You should move this entity off away from your level to wherever you want your 3D skybox to export to in scene. *That is the only entity that you should ever place outside your level.* As long as you have properly sealed your 3D skybox and tagged all 3D skybox brushes correctly, you will now have a fully functioning 3D skybox in your level when you export and compile.

That's it.

Since you never have to shrink and move objects, you can make changes that effect your scene and skybox any time you want. And if you incorporate parametric design principles, the system can be highly efficient. For example, if you are using [CorVex](#), you can control the boundaries between layout and skybox with single splines. But those techniques are beyond the scope of this article.

Your 2D Sky and Sky Writer

Wall Worm

Sky Writer

Turn your 3ds Max scene into a Source Sky Texture



In the last article we made the 3D Skybox. You could have started with the **2D Sky** just as well. But now we are here and it's time to explore some of the cool things you can do in **3ds Max** to generate your **sky texture**.

It is important to understand that when you are making your 2D sky you can take the gloves off and throw every single thing you have at the design. There are no polygon limits to worry about; you can do whatever you want, including using volumetric effects, particles, and any kind of material/map combination.

What you are doing at this stage is modeling a scene to "photograph" as the **backdrop to your level**.

The tools you use for this are going to be dependent on the environment you are creating and your level of expertise. Essentially, you want to use 3ds Max as your canvass for painting a scenic backdrop. You do this by sculpting a landscape, adding cities, forests, oceans, clouds, etc. When you are done, [Wall Worm's Sky Writer](#) will take a **cubic snapshot** of your scene and send it to Source.

Common Tools for Making Skies

Making your sky can be as simple or as complex as you want. To help you achieve your goals, there are many tools that come with Max or that you can purchase.

- [Super Rune's Guide to Making Procedural Skies in 3ds Max \(Free tutorial and Sky Rig\)](#)
- [Dreamscape](#) (Commercial)
- [Vue Xstream](#) (Commercial)

Common Tools for Populating your Landscape

- [Itoo's Forest Pack](#) (Free and Commercial)
- [Vue Xstream](#) (Commercial)
- [GhostTown](#) (Free and Commercial)

You have no rules for making your sky. The main work flow issue to keep in mind is to orga-

nize your scene well, and the most basic way to do that for your scene is to keep all of your 2D sky in its own layer as mentioned in previous articles. You may want to make a convention of calling this layer Sky Writer. For multi-user teams, you may want to also keep all objects in this layer in a self-contained file so that a level designer and environment artist can work in parallel in two separate files but see real-time feedback via XRef Scenes (which can be worked on remotely if your studio or group uses Autodesk Vault).

This article will not explain to you how to actually construct your sky. You really should read and follow [Super Rune's Good Looking Skies from Procedural Textures](#) to learn about some cool techniques that do not require any plugins. If you are using the Mental Ray renderer, you can look into using the **mr Sun** with a **mr Sky** map in the scene's environment map.

Finishing Your Sky

Once you are ready to test your sky in game, you simply need to open [Wall Worm's Sky Writer](#). Follow these steps:

1. Open Sky Writer (Wall Worm > Wall Worm Level Design > Sky Writer)
2. Enter a Name for your sky (like jungle_sunset or arid_noon, etc).
3. Choose the dimensions for your sky textures. A good start is 512.
4. Click Create New Sky
5. The Sky Writer node is now added to the scene at position [0,0,0]. You may need to move it to create the best render angle (for example, move it up if nearby hills are too close to allow the distance to render).

6. Hide all layers and objects that should not be part of your sky (your Layout and 3D Skybox layers and all other non-sky layers).
7. Click Render in Sky Writer. This process will take some time if you are using higher resolution, extensive environments and/or many volumetric effects.
8. When the render is done, click the Compile Sky to VTFs. This will send the sky into your game. (Some mods like CS:GO should have the Sky check button turned off before you compile the sky.)

After the sky is compiled, you can find the sky in your game's material/skybox folder. When exporting the scene to VMF with Wall Worm, you will see the Sky listed in the sky list. The sky will be used when you export the level.

Chapter 12 Working With Entities

Entities are the brains in a level. They control all the interaction and rules. You can create both Point and Brush Entities. To do so, simply load the appropriate tool in the Wall Worm Level Design flyout.

Note that not all entities work in all games. Some entities exist in all games, but many games have their own set of entities.

For convenience, Wall Worm's entity tools have links to the Valve Developer Wiki at the bottom of the modifier rollout for the current entity. For those entities that have been defined on the Wiki, you will get the documentation. For those that are mod/game specific, the amount of documentation available varies.

Access to Entities

Access to entities in Wall Worm is dependent on having your global settings set up correctly. This means that your global settings point to a FGD file (and all of its required parent FGD files are accessible in the same folder as the FGD). For example, for CS:GO, you need to point the FGD to the bin/csgo.fgd file and the base.fgd must also exist in that folder (since the csgo.fgd file refers to the base.fgd file).

When a FGD file is added to the global Wall Worm settings, WW creates a cache of all the entities. Sometimes it is necessary to update this cache if WW adds new features to entities. If a WW update requires this, simply go to the WW settings and hit Re-parse. It is common for the parse process to take a couple minutes to finish—in which time Max appears to be frozen.

Point Entities

Point entities are objects that exist at a point in the scene and have no volume in the scene. These kinds of entities generally are used to indicate locations of events/objects (like where players should spawn) or act as relays and rules for how the level works.

Some objects will always export as entities in the VMF with not need for choosing an entity type explicitly. Some objects that do this include WWMT Helpers, WWMT Proxies and Max light objects. If you want to add a light entity, simply use an Omni light in Max.

Objects that Automatically Export as Point Entities

Object in Max	Output Entity
<ul style="list-style-type: none">• WWMT Helpers• WWMT Proxies• WallWormMDL (Source Model)	<ul style="list-style-type: none">• \$cyclor (when Engine is Goldsource)• prop_static (\$staticprop is on)• prop_physics (prop_data is set)• prop_dynamic (neither of above)
<ul style="list-style-type: none">• Detailer objects (commercial plugin from Wall Worm)	<ul style="list-style-type: none">• prop_detail• prop_static (if tied to a WWMT model and WWMT as Sprite is off)

Object in Max	Output Entity
<ul style="list-style-type: none"> • Omni Light 	<ul style="list-style-type: none"> • light
<ul style="list-style-type: none"> • Free Spot • Target Spot 	<ul style="list-style-type: none"> • light_spot • light_spot + info_target
<ul style="list-style-type: none"> • Free Direct • Target Direct • Sunlight • Skylight • mrSun • Daylight • Dreamscape Sun 	<ul style="list-style-type: none"> • light_environment • light_environment + info_target

Working With Entities

For light entities, many of the native Max properties will translate into the game. The color of the light, multiplier, near and far attenuation are all exported with the light. If you need to add a specific Source entity property, you can add the Source entity properties to the light by selecting the light, opening the Point Entity floater, choosing the appropriate entity and then clicking the **Selection As Point Entity** button. (This will give you access to entity outputs and extra properties like HDR Brightness. You can do the same thing with WWMT Proxies to set more than the default export properties.)

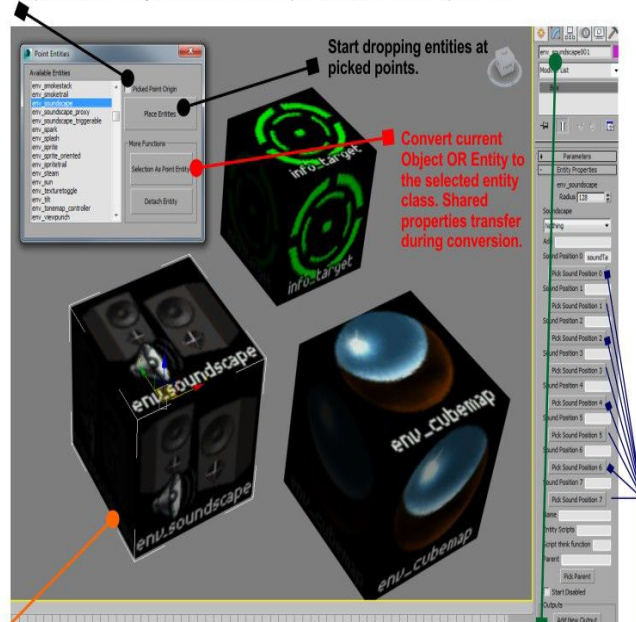
Note that for most parameters that are in both the Max object and entity (color, brightness, etc), the entity property takes precedence in the export—meaning that the color of a

light in Max is not always the color that the exported entity will have. (Lights not tied to an entity will use the light color.) However, the position and angles of an entity will always come from the node's transformation.

Also of importance is realizing how the light multiplier translates into the brightness value of your *non-entity lights*. Since the scale in Max lights does not match that in Source, there is a light multiplier that is used globally when exporting. All spot and omni lights will have a bright-

Point Entity Overview

When checked, the entity origin is placed precisely at the picked point. This may not be appropriate for all entities; by default, this is off and the entity is placed 8 units above the picked point. **HINT:** to drop entities onto object faces, turn on Snaps and choose Face Snaps in Max.



Editing an entity properties is possible when you select the entity and open the Modify tab in the Command Panel.

Materials for point entities are derived from the iconsprite of the entity. If there is a material in the scene with the same name, it will be used; if not, WW will look for a file called Editor.mat in your Max materiallibraries folder and check it for a material name matching the iconsprite. If none, it will attempt to parse it from VMT and TGA in the various directories WW checks for. It is best to create Editor.mat with the Material Library Generator.

Pick Buttons in Entity Properties will only let you select entities that have the appropriate properties. Those that need a target only work on entities that have a designated target Name. For Output picking, you can only pick entities in scene that allow inputs.

ness equal to their multiplier times the **light multiplier** in the VMF exporter UI. All lights that convert to `light_environment` use the **env mult** spinner instead.

Placing other Entities

Generally speaking, point entities are added to the scene as small cubes. Point entity orientation can be visualized by selecting the entity and changing the active Coordinate System to Local (as discussed in Chapter 3). Unlike in Hammer, you should generally not place `prop_static` or other such entities in your scene—as you should be using Wall Worm Model Tools and Wall Worm Proxies instead, which are discussed in the next chapter.

To turn world geometry into a brush entity, simply select the object, open the Brush Entities floater, select the desired entity, and click the Tie Entity button. To remove the entity from the brush, select the object and click the Move to World button.

Appending to Brush Entity

If you already assigned a brush entity to a node, you can assign other objects to the entity by selecting the existing entity object, open the Modify tab of the command panel, and scroll down to the Included Nodes group of settings. Click the Add Brush button and select the object to include. This allows you to use a single entity with multiple nodes.

Entity Outputs

Outputs are the way entities communicate with each other. You can control these parameters in the Outputs section of the Modify tab. The following page diagrams the output controls available in most entities.

Entity Outputs Controls

When you have an entity selected that has the ability to fire Outputs, you will see an Outputs group below all the other entity parameters.

Add a new Output to the list. If no outputs are selected, then an empty output is created. If one is selected, a copy is made.

The current list of outputs. Select one to edit parameters below and flash the recipient entities in the scene. *Right-click the list to select the recipients in Max.*

You can also manually type in the edit field above the list to set the output directly (and add outputs that are not defined in the entity). Separate each part of the entity with a ">". The order of the parameters is

OutPut Name > Target > Target Input > Param Override > Delay > Fire

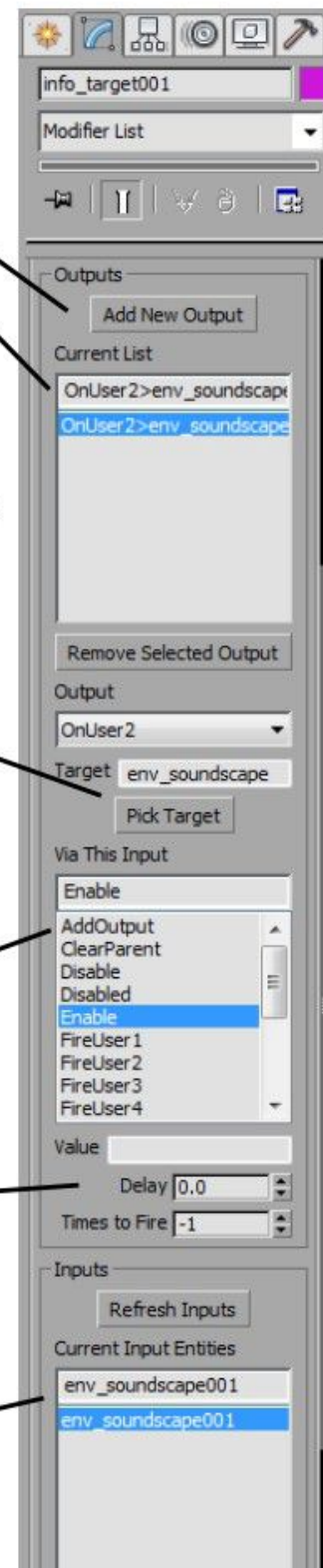
Note that the values won't update if typing manually until you change the focus from the edit field.

The target field is the name of your target. Note this is not the Max object name but the Entity property name. Updating this field will update the available Inputs below. Wildcards (*) are allowed. Or use the Pick Target button to select an entity in the viewport that accepts inputs--it will add the entities target name if present or the entity's class name if no target name is found.

The current list of inputs based on the target (specific entity or classes). If there is an input not in the list that you need to add, type it into the text field above and it will appear when you lose focus on the list.

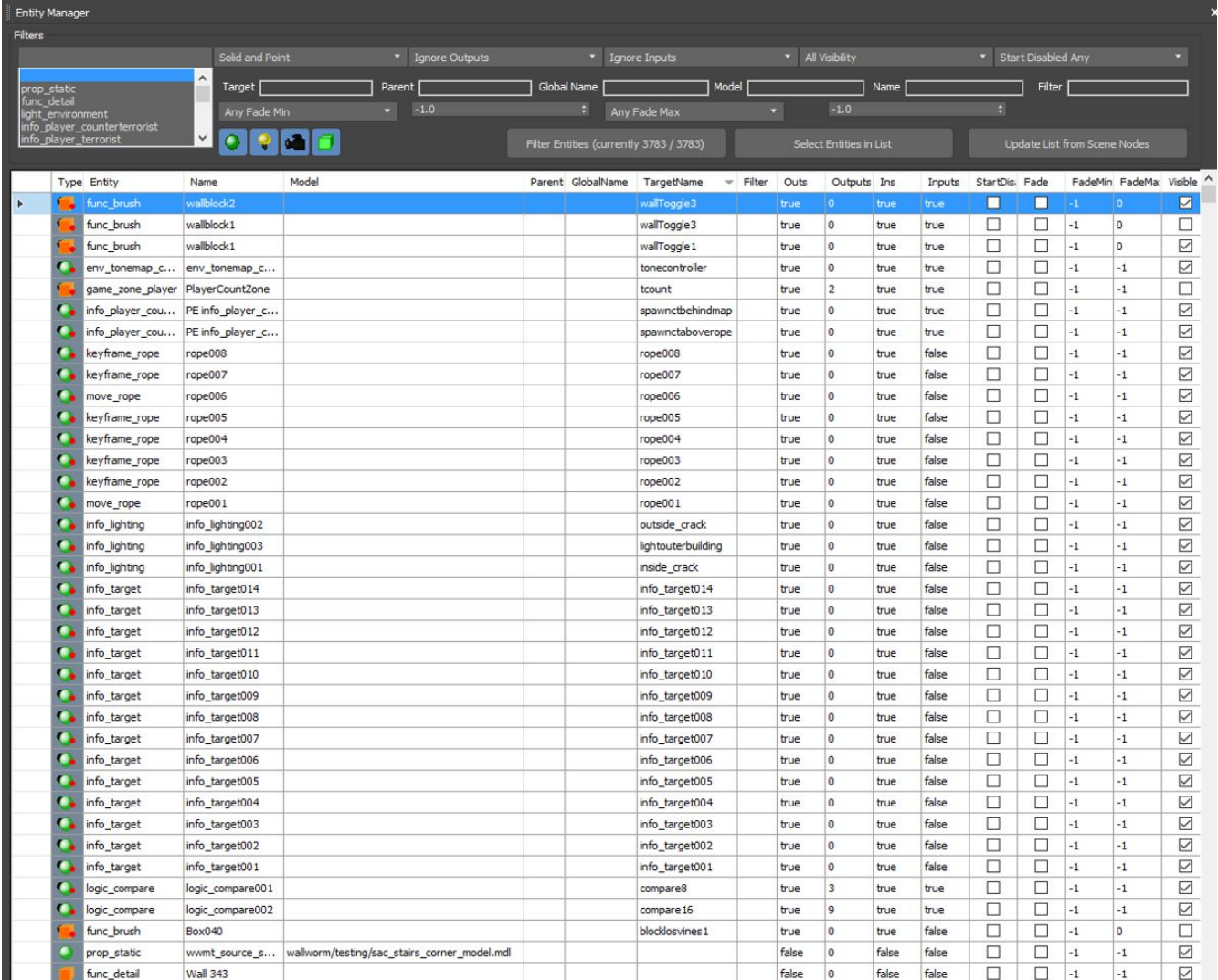
Enter the override value, delay before firing and times to fire. The default times to fire is -1.

Current list of entities in the scene that have outputs affecting this entity. If you click one, the relevant entity will flash in the scene. To go to it, click the entity and then click the Go To Input button below the list.



Entity Manager

There is also a way to see all the entities in the scene and some of their relationships by opening the **Entity Manager** floater. This tool will allow you to search for entities in your scene and edit them. To open the manager, click **Wall Worm > Wall Worm Level Design > Entity Manager**.



The Entity Manager floater displays a list of entities in the scene. The interface includes filters at the top and a table of entity details below.

Type	Entity	Name	Model	Parent	GlobalName	TargetName	Filter	Outs	Outputs	Ins	Inputs	StartDis	Fade	FadeMin	FadeMa	Visible
func_brush	func_brush	wallblock2				wallToggle3	true	0	true	true			-1	0		
func_brush	func_brush	wallblock1				wallToggle3	true	0	true	true			-1	0		
func_brush	func_brush	wallblock1				wallToggle1	true	0	true	true			-1	0		
env_tonemap_c...	env_tonemap_c...	env_tonemap_c...				tonecontroller	true	0	true	true			-1	-1		
game_zone_player	game_zone_player	PlayerCountZone				tcoun	true	2	true	true			-1	-1		
info_player_cou...	info_player_cou...	PE info_player_c...				spawnctbehindmap	true	0	true	true			-1	-1		
info_player_cou...	info_player_cou...	PE info_player_c...				spawnctaboverope	true	0	true	true			-1	-1		
keyframe_rope	keyframe_rope	rope008				rope008	true	0	true	false			-1	-1		
keyframe_rope	keyframe_rope	rope007				rope007	true	0	true	false			-1	-1		
move_rope	move_rope	rope006				rope006	true	0	true	false			-1	-1		
keyframe_rope	keyframe_rope	rope005				rope005	true	0	true	false			-1	-1		
keyframe_rope	keyframe_rope	rope004				rope004	true	0	true	false			-1	-1		
keyframe_rope	keyframe_rope	rope003				rope003	true	0	true	false			-1	-1		
keyframe_rope	keyframe_rope	rope002				rope002	true	0	true	false			-1	-1		
move_rope	move_rope	rope001				rope001	true	0	true	false			-1	-1		
info_lighting	info_lighting	info_lighting002				outside_crack	true	0	true	false			-1	-1		
info_lighting	info_lighting	info_lighting003				lightouterbuilding	true	0	true	false			-1	-1		
info_lighting	info_lighting	info_lighting001				inside_crack	true	0	true	false			-1	-1		
info_target	info_target	info_target014				info_target014	true	0	true	false			-1	-1		
info_target	info_target	info_target013				info_target013	true	0	true	false			-1	-1		
info_target	info_target	info_target012				info_target012	true	0	true	false			-1	-1		
info_target	info_target	info_target011				info_target011	true	0	true	false			-1	-1		
info_target	info_target	info_target010				info_target010	true	0	true	false			-1	-1		
info_target	info_target	info_target009				info_target009	true	0	true	false			-1	-1		
info_target	info_target	info_target008				info_target008	true	0	true	false			-1	-1		
info_target	info_target	info_target007				info_target007	true	0	true	false			-1	-1		
info_target	info_target	info_target006				info_target006	true	0	true	false			-1	-1		
info_target	info_target	info_target005				info_target005	true	0	true	false			-1	-1		
info_target	info_target	info_target004				info_target004	true	0	true	false			-1	-1		
info_target	info_target	info_target003				info_target003	true	0	true	false			-1	-1		
info_target	info_target	info_target002				info_target002	true	0	true	false			-1	-1		
info_target	info_target	info_target001				info_target001	true	0	true	false			-1	-1		
logic_compare	logic_compare	logic_compare001				compare8	true	3	true	true			-1	-1		
logic_compare	logic_compare	logic_compare002				compare16	true	9	true	true			-1	-1		
func_brush	func_brush	Box040				blockdovines1	true	0	true	false			-1	0		
prop_static	prop_static	wvmt_source_s...	wallworm/testing/sac_stairs_corner_model.mdl				false	0	false	false			-1	-1		
func_detail	func_detail	Wall 343					false	0	false	false			-1	-1		

The entity list shows all entity classes currently used in the scene. You can select a class from that list and click the Filter Entities button to list only that class. You can also use any of the other filters to display based on entity type (solid/point), implementation of inputs or outputs, model name, target name, name, fade distances and more.

Double-clicking the icon (type column) will select that item in the viewport. Double-clicking the Outs, Outputs, Inds and Inputs field will open a floater to edit that entity's parameters. If you want to select multiple entities, select them in the manager and then click the **Select Entities in List** button.

You can sort the view by columns by clicking the column headers. Unfortunately, at this time, the column headers will go out of view when you scroll down a long list—meaning you need to scroll to the top if this is the case.

Worldspawn

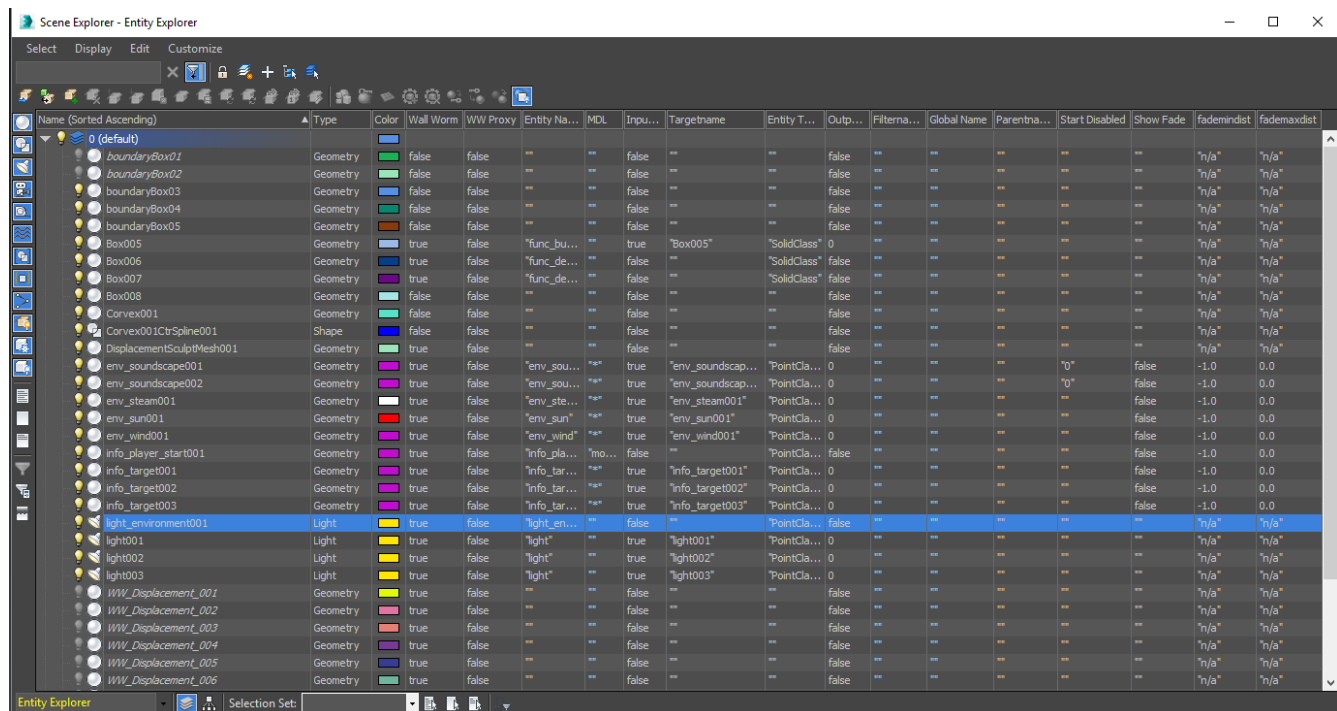
The **Worldspawn** entity is a special entity that you cannot create directly with Wall Worm. This entity is automatically created as a custom attribute attached to the scene's Root Node. It stores settings such as the sky name, occlusion settings, fade settings and more. You can access the Worldspawn entity in the World tab of Anvil.

Entity Explorer

The entity explorer is a special Scene Explorer that will show data relevant to entities in the scene similar to the Entity Manager but using a standard Max explorer view. With this you can quickly see if a node is tied to an entity as well as some of the common entity parameters that you may want to edit at once. Some of the fields in the Entity Explorer allow you to edit such as the fademindist and fademaxdist.

Working With Entities

To open the Entity Explorer, click Wall Worm > Wall Worm Level Design > **Entity Explorer**.



Edit Selected Entity

There is a macroscript in Wall Worm that you should consider utilizing. The macro name Edit Selected Entity will launch a floater containing a UI to edit the parameters of the selected entity in the scene as well as its outputs. You can add this to your quad menus so it appears in your right-click menu by clicking Customize > Customize User Interface > Quads and selecting Edit Selected Entity from the wallworm.com category. Then drag it to your quad menu (a good place is below the Object Properties menu in the quad).

Chapter 13 Importing Your VMF and MDLs into Max

One common task is to open your level inside Max that you had already started in Hammer. Wall Worm will allow you to import both VMF (Source) and MAP (Goldsource) levels. This chapter will only detail importing VMF.

What steps you need to take to successfully import a level with all the assets (brushes, displacements, models, entities and materials) is dependent on a couple factors. Using older versions of Max will require more steps, and is not covered in this chapter. This chapter presumes you are using 3ds Max 2015+. With Max 2015+ Wall Worm can natively pull assets (models, materials, textures) directly from your game files and do not need any special preparation. For the easiest process, you should use Max 2015 or later.

The next thing to understand is that, just like with Hammer, you must configure Wall Worm to know where your game files are and what FGD file to use. If your global settings do not point to your Game Info Directory and the game's FGD file, you will not be able to properly import the level. Common problems with not having the settings correct include all props appearing as cubes and materials/textures missing from models and brushes.

Import a VMF

To import a VMF you should have an empty scene. If not, the importer will warn you that the scene already has objects. Generally you should click File > New before importing a level.

1. **Click Wall Worm > Wall Worm Importers > Import VMF or Map File.**
2. For the most part, the only option you'll want to change for the import is the turn on In-

stances if you want to import Instanced VMF files (if your scene used func_instance entities). By default instances are turned off because they slow down the import.

3. **Click the **Choose VMF or Map to Import** button.**
4. Immediately after the import is finished, you should click the button named Select Imported Geometry that Needs Attention.
5. If step #4 finds anything and selects it, then change the Rounding Threshold Spinner to 0.0 and click the Rebuild Selected button.

Note that steps #4-5 will generally fix any brushes or displacements that did not properly import, which can happen on occasion.

Working with Imported VMF

Now that the level is imported, you must choose your intent. You can continue working on your level inside Max and simply finish it in Max. Or you can simply use the scene as reference for building props.

The imported data will be organized much as it was inside Hammer. Some differences are that your Visgroups are now Layers inside Max. Brushes are already tagged to re-export as brushes.

If you did not have a proper FGD selected in the global settings, all entity data will be found in the generic keyvalues of imported entities (found in the modify tab). But if you had a correct FGD set for your settings, full entity controls should be available in the modify tab.

From here on out you can continue working as if the scene had started inside Max.

Importing Models

So long as you are using 3ds Max 2015+, importing models is simple. There are a few methods of bringing props into 3ds Max. At this point in time the MDL Importer in WW will bring in the mesh, materials, bodygroups, hulls, LODs and skins of props. As of this writing, animations, flex data and gibs are not yet supported.

When WW loads a MDL file, it will be presented with a geometry class called WallWormMDL. Those parameters that can be manipulated in Max are available in the Modify tab when a WallWormMDL node is selected. See the [MDL File Loader documentation](#) for more information on the MDL Loader.

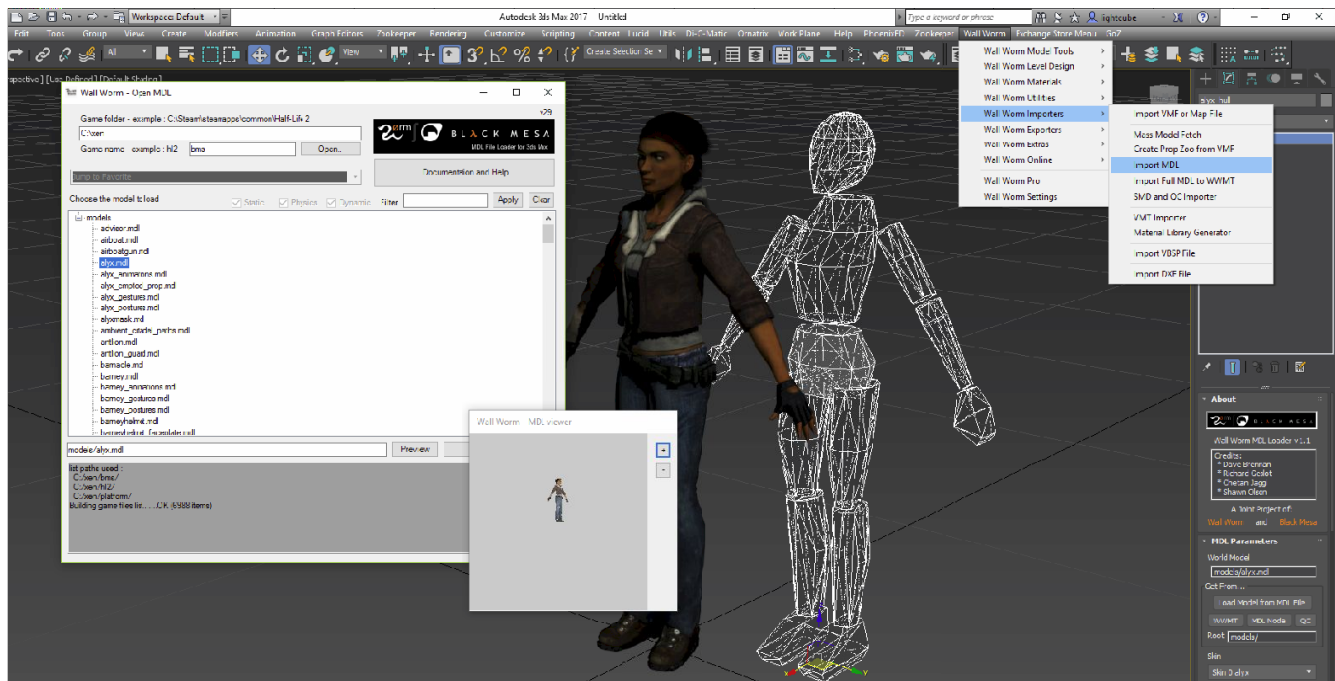
Import Single MDL

To import a single MDL node you can simply click Wall Worm > Wall Worm Importers > **Import MDL**. You'll then see a model browser that you can preview your props with. When you find a prop you want to import, simply double-click the prop's name.

Now Wall Worm will load one or two WallWormMDL nodes into the scene at the world origin. One node will display the prop. If the model has a collision hull, there will be a second copy of this node with the Show Hull option (which displays the model's hull instead of the mesh).

Here is an example of Alyx with the hull offset to show the comparison of the mesh and hull:

Importing Your VMF and MDLs into Max



Import Prop to Modify

If your intent is to re-export the prop (perhaps you are re-skinning or modifying the prop), then instead of using the menu above, you should click Wall Worm > Wall Worm Importers > **Import Full MDL to WWMT**. This function will extract all available model data and set up a full WWMT setup that will include the mesh, bodygroups, LODs, collision hull and various other settings.

Create Prop Zoo From VMF

If you want to see all the props used in a VMF, you can create a prop zoo. This will load one

copy of each MDL referenced in a VMF file and line them all up. To do this, click Wall Worm > Wall Worm Importers > **Create Prop Zoo From VMF**.

Load Multiple Props at Once

If you want to load several props at once into the scene, you can click Wall Worm > Wall Worm Importers > **Mass Model Fetch**. This tool lets you create path and filename filters that will fetch any model that matches. You can double click the pre-populated folder paths to import all props in that folder and sub folders. The more specific you make the filters, the more targeted the props will be when they come in. You can use a comma-separated list of paths for the import paths.

If you find that you re-use the same props a lot, you can save a path filter as a Favorites that you can reload later.

By default, the props will be placed into a layer named Imported Props. You can select another layer if you wish.

Creating WallWormMDL Nodes

The WallWormMDL geometry class can be created at any time like any other object (just like a Box, Sphere, etc).

1. **Open the Create Tab** of the **Command Panel**.
2. **Pick the Geometry Type**.
3. **Select Wall Worm** from the **category drop-down**.

4. **Click** the **Source Model** button.
5. **Pick** points the the scene to add the nodes.
6. When done placing, **select** a WallWormMDL node you placed and **open** the **Modify Tab**.
7. **Click** the **Load Model from MDL File** button.
8. **Browse** for the MDL to use.

MDL Node Functions

There are several useful functions available to WallWormMDL nodes. You can access many of these in the modify tab when one is selected. The Wall Worm Connection rollout will allow you to tie the node to an entity, set flags to exclude the node from the VMF exporter, tag as a sky object (to be offset around the sky camera) and set manual key values.

Right-clicking the Skin drop-down will randomize the skin of the model if the prop has multiple skins.

The MDL Utilities rollout will let you select other MDL nodes in scene that are using the same prop, have the same skin, or same prop but different skin.

If you want to change the model and generate a WWMT Helper for it to quickly export, click the Generate WWMT button.

If the node is considered a WWMT Proxy (because it was generated with the Create Proxy button) then you should click the Collapse for Skinning button to convert it into a WWMT

Proxy Editable Poly that can be properly exported and used for a WWMT function to collect skins. Failure to do this can create issues with the functions for skinning and you may lose future changes to that Proxy's material if you do not collapse it (as the WallWormMDL nodes have functions to reset the skin to those defined in the MDL).

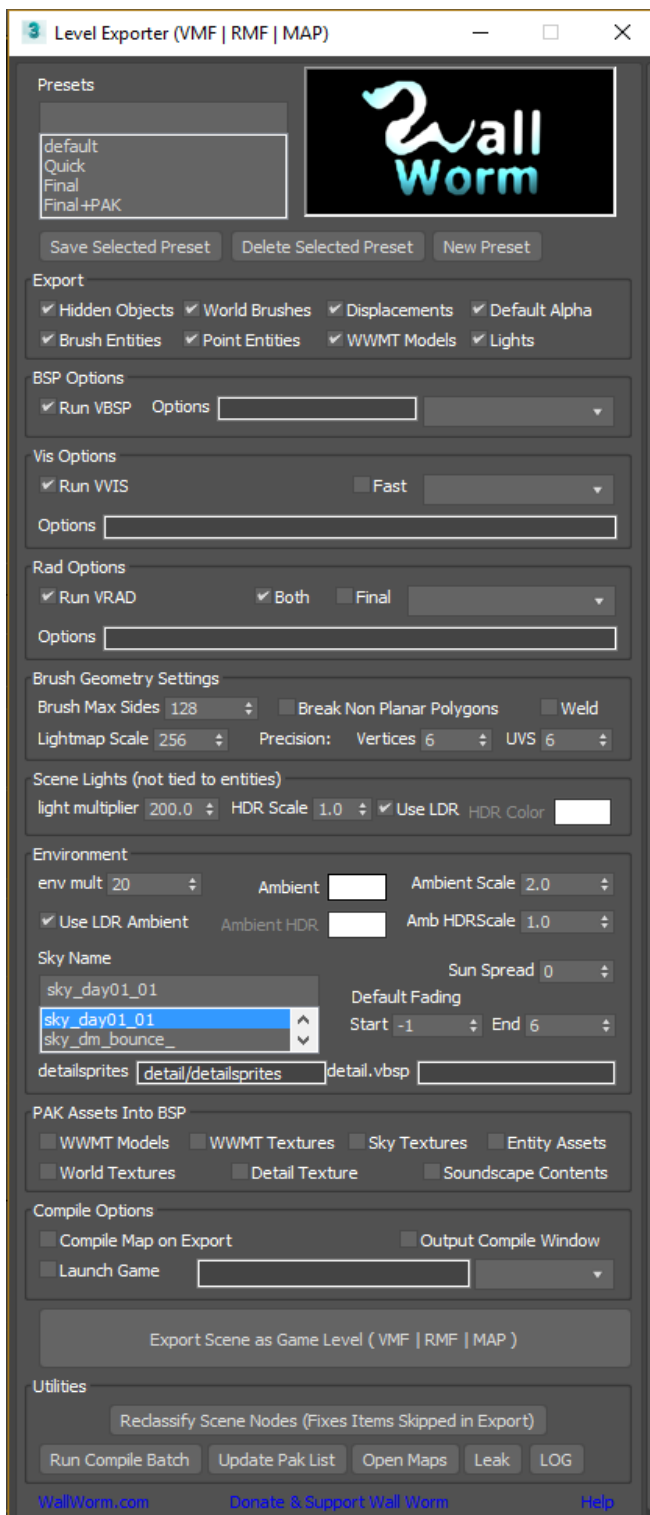
Chapter 14 Exporting Your Level into Source

To get your level into Source, simply open the VMF Exporter. Unlike many export function, this is not located in the normal Export menus in Max. Instead, it is located in the Wall Worm menus. You can export a level by clicking **Wall Worm > Wall Worm Exporters > Export Scene as VMF**.

This menu gives you control of various export options, including compile parameters, global light settings and PAKing options.

Most of these options should be self-explanatory and refer to similar export and compile options available in Hammer. Those that differ are explained below.

One important thing to know is that in later versions of Wall Worm, there is a global manager that tracks the exportable objects in the scene. Sometimes that manager's data can get out-of-sync with the objects in the scene, causing objects to go missing from the exported file. In that case, you can click the Reclassify Scene Nodes button at the bottom of the exporter. See Wall



Worm Scene Manager.

Light Settings

Since light units in Max and Hammer are not identical and some settings such as HDR are not part of light objects in Max, you can give some defaults that effect all lights in the scene that are not explicitly tied to light entities. For example, you can set a default HDR scale for any light that hasn't been specifically given a light scale via entity or light tools in Wall Worm.

The light multiplier will take the current value of non-environment lights (omni and spot lights) and multiply it by the light multiplier across all relevant lights in the scene for the VMF file. For example, any omni lights with multipliers set to 1.4 will output as a brightness of 280 if the export multiplier is set at 200 ($1.4 \times 200 = 280$). *Remember—this calculation applies only to lights that are not tied to Source Entities. If you light is tied to an entity, then the brightness is controlled inside the entity parameters.*

Note that changes to the VMF Exporter light multiplier is global and only affects the exported light values, not those in the Max scene. This means that if you want to globally modify the lights in your exported Source scene, simply change the multiplier.

The Environment light settings is separated from the standard lights. This is because the environment lights (Direct Lights, Sun, etc) are used differently.

RAD Files for Radiant Materials and Models with Transparency

If your scene has textures that should light the environment or if you have models with transparent textures that need to cast accurate shadows, you need to create a RAD file before exporting/compiling the VMF. You can do this easily with Wall Worm by clicking: **Wall Worm >**

Wall Worm Level Design > Rad Worm. If there are any self-illuminated materials in the scene or if there are any WWMT helpers that do not have the \$opaque setting, you will be prompted to save a RAD file in your maps folder. Models without the \$opaque setting will get added as a **forcetextureshadow** entry in the RAD file. Self illuminated materials will get listed as casting light into the level and have their self-illumination color used for this illumination (and if this material has a WW Shader custom attribute applied, the illumination brightness will correspond to the material's Radiosity Amount in the Miscellaneous rollout of the material).

- The RAD file must match the name of your level. For example, if your map is mymap.vmf, then your RAD file should be mymap.rad.
- The RAD file should be saved in the same folder as your VMF file.
- Set your RAD compile parameters in the VMF Exporter to use **-TextureShadows** and **-StaticPropPolys**.

PAK Assets

The PAK functions in the VMF Exporter will force all of the selected asset types to get PAKed into the BSP file. (The BSP file is the final output of a level for the game engine.) This means that all models, materials and relevant files that are in the system will get automatically PAKed without the need to open external applications or write batch files. This includes assets in Xref Scene files.

The assets that get PAKed must reside in the game's folders. This means that assets that only exist in a VPK file will not get PAKed. It also means that any models and textures that have not been exported into MDL and VTF format will also be missing from the PAK.

Generally speaking, you may want to avoid PAKing until you want to distribute the level to

others.

Working with PAK and RES files

Although the PAK function is useful, there are times when you also want to use RES files (files that list assets in a level). For example, some assets won't work as expected when PAKed into a BSP file—such as loading screens in CS:GO.



As such, you can also create a RES file for your level with Wall Worm. It generally uses the same options as the PAK but has some other automatic file collection. Click **Wall Worm > Wall Worm Level Design > RES File** to generate a RES file from the scene.

Using the Res Maker

1. Open Res Maker (above).
2. Select the types of assets to include (WWMT Models, textures, etc).
3. Click the Generate RES File button.
4. Save the RES File into your mod's maps folder alongside the BSP file. Make sure that

the RES File has the same name as your map except use the .RES file extension.

After generated, you can edit the file.

More Info

The PAK/Res Maker will also automatically include the following files if detected. New files are often added to this list as well.

In this list, the “mapname” is used to denote the name of your map. All items are relative to the mod directory.

- maps/mapname.bsp*
- maps/mapname.txt*
- maps/mapname.nav*
- maps/mapname.res*
- maps/mapname.ent
- maps/mapname.kv*
- maps/mapname.jpg*
- maps/mapname_exclude.lst
- maps/mapname_manifest.txt
- maps/mapname_commentary.txt
- maps/soundcache/mapname.cache
- maps/soundcache/mapname.manifest
- maps/cfg/mapname.cfg

- particles/mapname_manifest.txt
- scripts/soundscapes_mapname.txt
- materials/overviews/mapname.vmt
- materials/overviews/mapname.vtf
- materials/overviews/mapname_radar.vmt
- materials/overviews/mapname_radar.vtf
- materials/mapDesc/mapname.jpg
- materials/vgui/maps/menu_thumb_mapname.vmt
- materials/vgui/maps/menu_thumb_mapname.vtf
- resource/overviews/mapname.txt
- resource/overviews/mapname.dds*
- resource/overviews/mapname_radar.dds*
- resource/flash/loading-mapname.swf

* This list of assets is the automatically detected items for both the RES maker and the BSP Pak functions in the VMF Exporter. Those with an asterisk are only added to the RES and not to the PAK.

Include Options

The RES generator lets you choose groups of objects to include. Here is how they work:

WWMT Models

This option will make the RES collector find all WWMT helpers in the scene and collect these files, if found on disk:

- .MDL
- .VVD
- .SW.VTX
- .DX80.VTX
- .DX90.VTX
- .PHY

WWMT Textures

This option will look for any materials and bitmaps used in the WWMT models in the scene. Here are the files it will collect:

- .VMT
- .VTF

Sky Textures

This will find all the skybox textures in the scene that are associated to a Sky Writer helper.

Entities / CVX Props

This function will collect MDL, VMT, WAV and MP3 files that are listed in any Wall Worm entities or Convexity Entities in the scene.

Note that for MDL and VMT files gathered from entities, the collection does not look for associated files—for example, VTF file included in the VMT—unless there is an associated Wall Worm construct in the scene (for example, a WWMT Helper related to the model property of an entity).

World Textures

This will collect VMT and VTF files that belong to brush, decal and displacement objects in the scene.

Soundscapes

This will parse the soundscape that is associated with this level (scripts/soundscapes_map-name.txt) and get all the WAV/MP3 files listed in the soundscape file.

Distributing Your Level

Once you've finished the compile process and are ready to start distribution, you can easily package your game level for distribution. Open the RES Maker UI as listed above and click the ZIP From Res. This will create a ZIP file with your level and all assets that are listed in the RES file. Requires [WinRar](#).

Chapter 15 Troubleshooting Export and Compile Problems

Troubleshooting your level is something you will need to learn. Making levels for Source has various technical requirements that you are forced to learn and understand in order to be successful. For the most part, the troubleshooting between Hammer and Max is very similar. However, because Max has functions and objects that you don't normally create in Hammer, there are some extra things you may have to look at when troubleshooting problems.

Below are a few important tips:

Scene is Blank or Objects Missing in Game or Hammer

There are three possible reasons that your scene is blank when exported:

- Your world geometry was not tagged to export in Max.
- The User visgroup the geometry belongs to in Hammer may be turned off.
- You have hidden geometry and did not check the Hidden Geometry option in the VMF exporter.
- The Scene Manager's data caches is out-of-sync with the scene. See Wall Worm Scene Manager for more information.

Invalid Geometry when Scene Opened in Hammer

If you open your exported scene in Hammer and get a notice about invalid brushes that need to be deleted, you will have to edit the brush geometry in Max or convert the brushes to models. Remember the rules for brushes (convex, planar polygons and no co-planar polygons).

Be aware that there are times that the object are valid (and will compile to a brush straight from Max) but Hammer has rounding problems that will break the brush inside Hammer.

Finding the Invalid Brushes in Max

If you compile a level from Max, you may see an error in the compile log about a problem with a brush. If so, you'll need to fix that brush in Max and re-export. You can Get Brush by ID floater (**Wall Worm > Wall Worm Level Design > Wall Worm Map Compile Tools > Get Brush by ID**). This will select the brushes you need to fix.

If you open the VMF in Hammer and your version of Hammer tells you what brushes are invalid when opening, you can write the brush Ids into Notepad and paste them as a comma-separated list into the Get Brush by ID floater. For those versions of Hammer that don't give you the Ids, you'll need to refer to the compile log.

Finding Leaks

Anyone who has ever made a level for Source has learned about Leaks. A leak means that one or more point entities in your level are exposed to the Void. In other words, the level isn't properly sealed with world geometry brushes.

You can load a leak file in Max to help find the leak. Click **Wall Worm > Wall Worm Level De-**

sign > Wall Worm Map Compile Tools > Load Leak File. Browse for a file in your mapsrc folder that matches the name of your compiled level but with a “.lin” file extension.

Viewing Portal Files

If you need to see how the Visleaves look inside Max, you can load them with the PRT loader. Click **Wall Worm > Wall Worm Level Design > Wall Worm Map Compile Tools > Load PRT File.** Browse for a file in your mapsrc folder that matches the name of your compiled level but with a “.prt” file extension.

Each Portal designates the opening between one visleaf and another. In Max, the portal is a Renderable Spline with a name that contains the VisLeaf Ids of the two visleaves it divides. In the Portal has a Portal Information Custom Attribute in the modify tab with some functions to help work with it, including a button to select all other portals belonging to Leaf 1 or Leaf 2.

When loaded into the scene, the Portal splines are placed into a layer named **PRT Import Layer.**

An Error Freezes the Max UI

To speed up the export process, the VMF Exporter turns off some Max UI elements. If there is an uncaught exception during the export, those features don't get turned back on. To fix this, simply click **Wall Worm > Wall Worm Utilities > Unfreeze UI.** Please report any such cases in the WW forums.

Chapter 16 Wall Worm Scene Manager

Wall Worm has a scene manager that is always tracking your scene in the background. When new objects are added to the scene, Xref scenes are loaded or several Wall Worm functions are run, the scene manager classifies the objects so that internal look-ups can work more efficiently. This allows many functions to limit queries to the collections of objects related to those functions.

Below is a list of some functions that are dependent on the scene manager:

- VMF Exporter
- Entity Manager
- Entity Inputs/Outputs

There are times that the manager can get out-of-sync with the scene. When this happens, objects may not be included in the results of the function. You can force the manager to rebuild the cache with these methods:

- Click Wall Worm > Wall Worm Utilities > Classify Scene Nodes
- In the VMF Exporter, click the Reclassify Scene Nodes button in the Utilities section (bottom of exporter).
- In the VMF Exporter, right-click the Export Scene as Game Level button. (This will only work in WW 3.731+). Doing this will reclassify then export the scene.

- In the Entity Manager, Right-click the Update List from Scene Nodes button.

Advanced Scene Manager Topics

The scene manager is a global object in Max that you can access directly via MAXScript. You can access it via this variable: **::wallwormHelperOps**. Below is a list of some of the parameters you can query with this object. Generally, you should avoid calling the methods of this object unless they are documented as some of the methods are meant to be internal only. Unfortunately, methods and properties could not always be set to private/protected until later versions of Max. To keep WW as compatible as possible, all members are still public. This may change in the future.

Scene Manager Collections

You can query the collections of the scene manager for your own needs if Wall Worm does not already give you access to ways to find them. To do so, you iterate over the collections by referencing them in your script. Here is an example query:

```
validEnts = for ent in ::wallwormHelperOps.entities where isValidNode ent collect ent
```

This code will set the value of variable `validEnts` to an array of all entities in the scene that have not been deleted. (Wall Worm does not remove entities from the lists when deleted from the scene. Generally speaking, the entities are not cleansed until a new scene is opened or created.)

- Brushes : brushes in the scene.
- Displacements : All displacements in the scene.

- Entities : All entities in the scene.
- PointEntities: All point entities in the scene.
- BrushEntities: All brush entities in the scene.
- Sculpt: All sculpt meshes in the scene.

Scene Manager Methods

Here are a few of the methods available in the scene manager.

- SanitizeAll() : Will clean up all the collections by deleting all invalid nodes (those that have been delete).
- SanitizeEntities() : This function will clear just the entity collections by removing those that are not entities any more.
- exportable(sceneNode) : This function will return true if the sceneNode passed to it will export in a VMF based on certain native settings or false if it will be skipped. For example, it will return false if the object is within a layer called "VMF_Exclude" or it has been tagged to not export in the VMF.
- getEntityClassName(sceneNode) : Returns a string of the entity class name of sceneNode. If the object is not a valid entity returns "unknown".

Chapter 17 Introductory Summation

The first section was to give you basic concepts of 3ds Max, Wall Worm, certain broad principles and tools. Some of the core concepts you should understand now are:

- You can use 3ds Max as a replacement to Hammer for Source Engine Level Design.
- Although the exact same principles for BSP level design apply for both Hammer and Max, the methods of creating world geometry in Max is not the same as in Hammer.
- The philosophy of Wall Worm is to optimize the entire process of *creating custom assets* (models, materials, textures and levels).
- You can create all models, materials and textures directly inside 3ds Max without opening third-party software.
- Methods for working with Displacements and 3D Skybox objects is entirely different than in Hammer.
- Learning the Max UI is imperative to becoming proficient in using Wall Worm to its full potential.
- Learning 3ds Max and Wall Worm is more involved than learning Hammer, and will take more time to learn than Hammer—but the trade-off is increased productivity and power to be creative.

Special, Personal Tips

As a final note to the this section, I think it is important to point out some *extra* important principles. Based on the volume of emails and forum posts I've responded to over the years, I know that far too many people fail to cultivate some very essential personal qualities that affect their education and progress. As such, here are some added tips:

- Learn to solve problems.
- Learn to have patience.
- Learn to use Google to look for answers.
- Learn to use the 3ds Max documentation.
- While you should feel free to ask questions, please do not waste the time of others by asking questions in forums that are easily answered via the documentation or Google.
- Experiment with different ways to solve problems.
- Do not believe everything you read in forums, even if you read it 10,000 times.
- Don't answer questions in forums unless you are qualified to answer them—in other words, don't give an answer to something just because you've read it 10,000 times in forums; instead, only answer those questions with which you have deep, personal experience.
- If you encounter a bug in 3ds Max, please report the bug to Autodesk via the Help >

Report a Problem link.

- If you encounter a bug in Wall Worm, please report it in the Wall Worm forums.
- Keep your software up-to-date—both Wall Worm and 3ds Max (using the latest service pack of the latest version of 3ds Max is always preferred).
- Learn something new every day.
- Stay humble, and learn to appreciate the collective creativity of others in your community.

In the next section, we are going to cover various essential concepts via practical examples.

Part II

Digging a Little Deeper

Chapter 18 Uvw and Texture Mapping

One of the most common challenges in 3D is understanding the concept of UVW and texture mapping. It's been my experience that this is a huge hurdle for many people, including those who are very sophisticated with using Hammer.

Texture Mapping is the function of determining what part of a texture (like a bitmap) appears at a specific location of a piece of geometry (like a wall). It is texture mapping that determines if the brick texture is aligned to the bottom of a wall or the top (or both), and whether the texture is stretched.

The controls for UVW in Hammer are very simple, and are determined entirely by coordinates stored in each face of each piece of geometry. Inside Hammer, the user only has the ability to change to rotation of a face's UV coordinates in one axis and the scale of the texture in two axes. Aside from some basic alignment tools, this is usually all the Hammer user must consider when working with UVW.

The world of UVW is far more complex in 3ds Max. The extent of the tools and the difference from which it works in Hammer make UVW a large obstacle in moving to 3ds Max. This chapter is all about getting you to think about working with textures in Max in new and innovative ways. And to start, we will return to advice offered earlier in the book: Do not Approach UVW in Max like you do in Hammer. Instead, start thinking like a Max user—think parametrically. We'll use this in our first case study shortly.

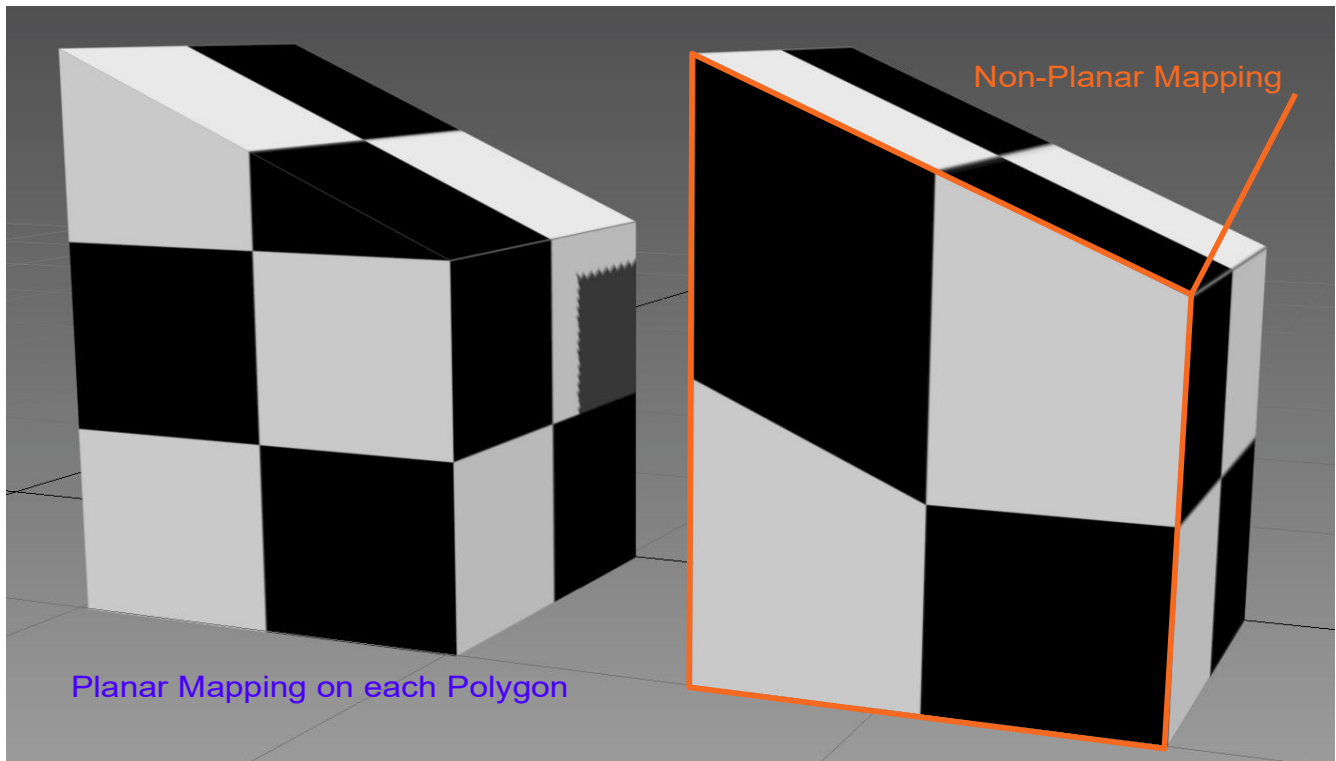
Different UVW Strategies

There are different tools and strategies for using the correct kind of UVW for the specified

type of geometry. For World Geometry, you are generally looking for ways to get your UVW to flow and tile seamlessly from one polygon to the next. For models, you may need to use a technique called Unwrapping, which is akin to unwrapping the skin of an animal to hang on a wall. Unwrapping can work for some cases of world geometry, and tiling UVs can work for models, but generally you will use tiling UV strategies for world geometry and unwrapping for models.

For brush geometry, you should explore how to use these standard modifiers that come with 3ds Max: UVW Map, UVW Xform, MapScaler, Unwrap UVW. When using the Unwrap UVW modifier, research the Flatten Mapping functions in the UVW Editor.

In Hammer, all UVW is tightly controlled planar mapping. This means that all polygons on any geometry made in Hammer will have flat textures. This isn't always the case in 3ds Max, as mapping can be at the tri instead of the polygon. This is because 3ds Max is not only designed for the special needs of world geometry of game engines, as is Hammer, but also for other needs (like special effects and characters). To help understand this, we'll use some graphics.



Notice in the graphic how the textures on the left image are uniform across the front polygon. That polygon (and all other visible polygons except the one highlighted in orange) have a planar mapping. The orange highlighted polygon on the right looks skewed because the UVW mapping is not planar.


Hammer never creates the scenario on the right, but it is easy to produce in Max. Knowing how and why this happens will help you understand how to avoid the dilemma. This happens in Max because mapping is stored, internally, on each triangle that makes up a polygon. It is up to you to keep all tris in polygons planar when working with world geometry.

Forcing a polygon to have planar mapping is always as easy as applying one of the many UVW modifiers available in Max. Keeping the polygon mapping planar while editing objects already mapped is a little more complex, and is dependent on the type of object you are editing. When editing the geometry in an editable poly object, the solution is to **turn on the Pre-**

serve UVs option in the modify tab.

Case Study: Texturing a Simple Scene

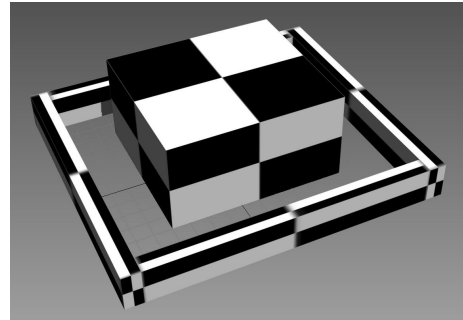
Let's start a new scene and build a simple building with a wall around it. We will start with some of the basic primitives that are inside 3ds Max.

1. Turn on **Snap to Grid**.
2. Set the **Home Grid Spacing** to 64.
3. Select the **Perspective View** if present in your screen and maximize that viewport (**Alt+W**).
4. Open the **Slate Material Editor**.
5. **Right-click** an empty part of the material editor view and choose **Materials > Standard**.
6. **Left click** the circle next to the **Diffuse Color** and drag to an empty area of the view.
7. Let go of the mouse and choose **Checker**.
8. **Double-click** the Material so that it has a little white outline in the view and the material parameters appear at the right side of Slate.
-  9. Click the **Show Shaded Material in Viewport** button (so this material will display in the viewport).

10. Click **Wall Worm > Wall Worm Level Design**

> Brush Mode. This will automatically apply the selected material to new objects and tag them as World Geometry.

11. Open the **Create Panel** and choose **Box** from the **Standard Primitives** geometry class.

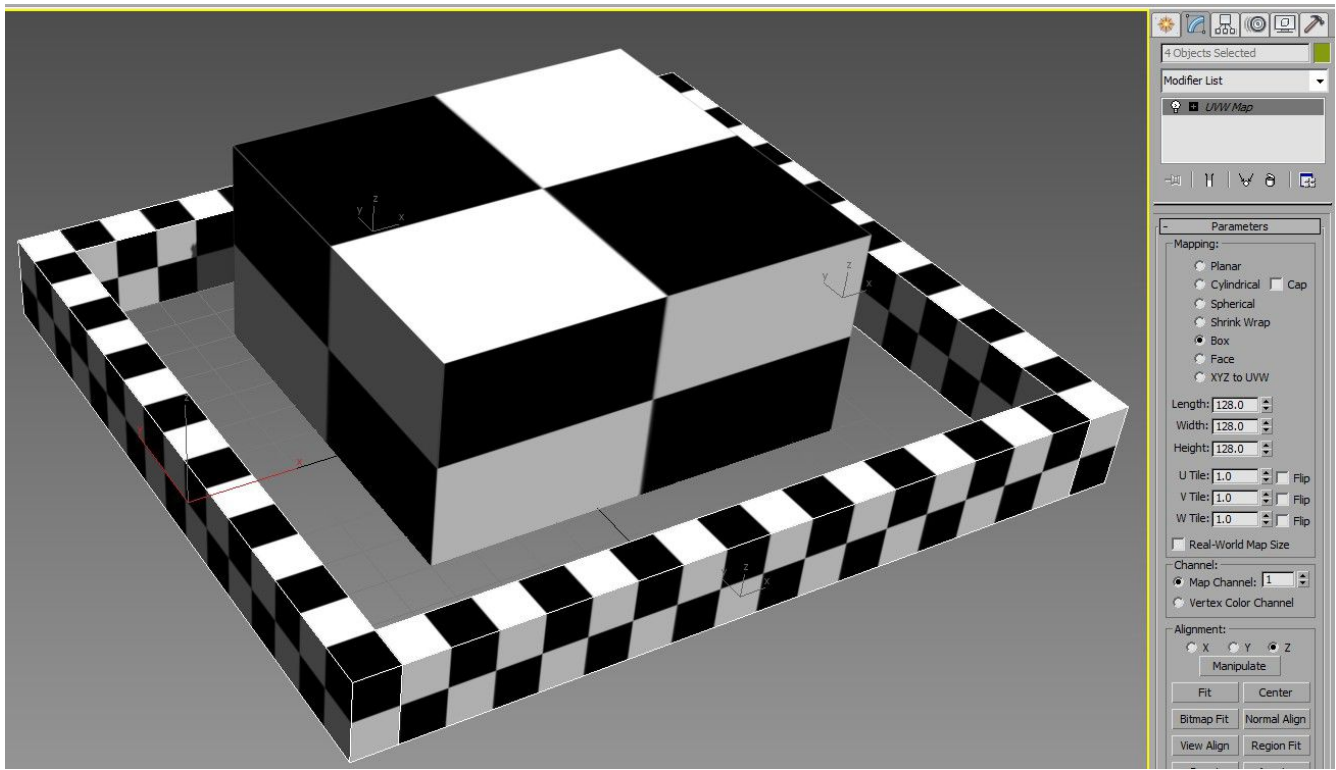


12. Now proceed to recreate the image to the right with four long box primitives composing the outer wall and one large box primitive representing a basic building structure.

Notice how the default mapping is such that the texture tiles exactly one time per polygon. Although the mapping is planar, it doesn't have the same scale from one polygon to the next.

In Hammer you would fix this by using the Texture Application tool to set the UV the faces. In the scenario above, you'd apply a UVW modifier. In this case, you want to apply a single UVW modifier to groups of objects that should share the same UV attributes.

1. Select all of the outer wall boxes.
2. Switch the **Command Panel** to the **modify tab**.
3. Scroll down the modifier list and choose **UVW Map**. Notice that the textures on the wall objects have now become uniform... but the sides are strange.
4. Change the **Mapping** selection from **Planar** to **Box**. Now the sides display better.
5. Set the **Length**, **Width** and **Height** parameters in the **UVW Map** to 128. Notice the textures now have a much more expected output.



As long as this modifier is applied to the selected objects, the modifier itself will control the UVW on them.

Because the UVW Map modifier was applied to multiple objects, the modifier is considered to be *instanced* across them. This means that you can always select any object using the modifier... and changes to the instanced modifier immediately affect all the other objects with that modifier.

To modify the UV offsets, you can select the Gizmo Sub-Object of the UVW Map modifier (by clicking the plus icon in the modifier list) and move, rotate and scale it visually. Notice all objects updating immediately.

Acquiring UVs

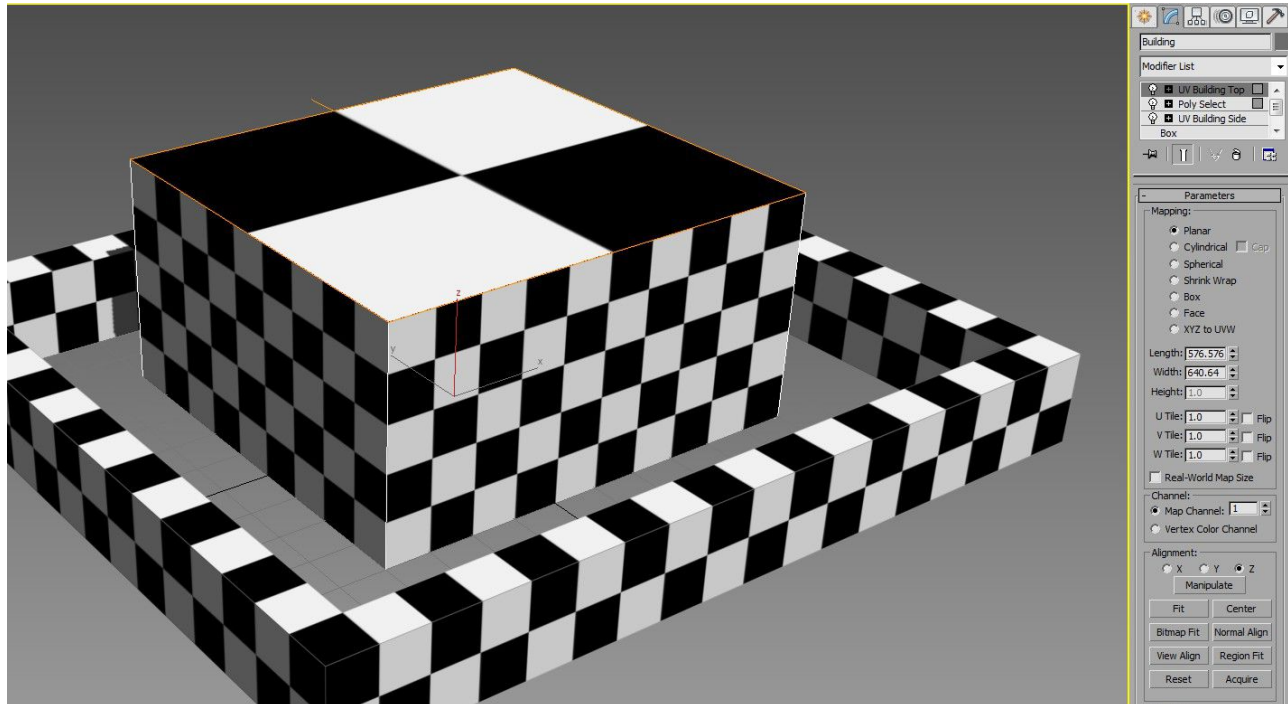
Now lets use the wall's UV as a starting point for the building.

1. **Select** the building block.
2. Switch to the **modify tab**.
3. Choose **UVW Map** from the modifier list.
4. Scroll down to the **Acquire** button and select one of the walls.
5. Choose Absolute. Now the building's UVs match the scale of the walls.

Layering UV Modifiers

Now you are going to use multiple UV modifiers to control different parts of an object.

1. **Select** the Building object.
2. Apply a **Poly Select Modifier** in the **Modifier List**.
3. Switch to **Polygon sub-object mode** (keyboard shortcut **4**).
4. **Select** the top polygon of the building.



5. Apply a **UVW Map modifier** and keep it set to planar. Notice that the mapping is now back to a normalized scale (similar to Fit in Hammer).
6. **Press F2** to toggle the face selection highlight. At this point, changing the parameters of the first UVW map control the sides of the building, but the second modifier controls the top UVs. Because it can get confusing if using multiple UV modifiers, you should make a habit of naming the modifiers in the modifier stack. To do this, right-click the modifier and click Rename. Call one “UV Building Top” and the other “UV Building Side”. Because the modifier stack allows you to edit the base object and the different modifiers up and down the stack, you can change the box height and any of the modifiers in the list at will. Take note of the fact that if you select a level in the list that is below changes made above, you may lose those changes in the viewport. You can continue to see the final output by clicking the Show End Results button right below the modifier list.

Apply Materials to Object

So far we've been using a “procedural” texture for the material (the checker texture displayed on the objects we made). Such a texture is not game ready, as we need to use bitmaps. Furthermore, you probably want to apply a different material to the walls as the building, and you may want more than one material per object.

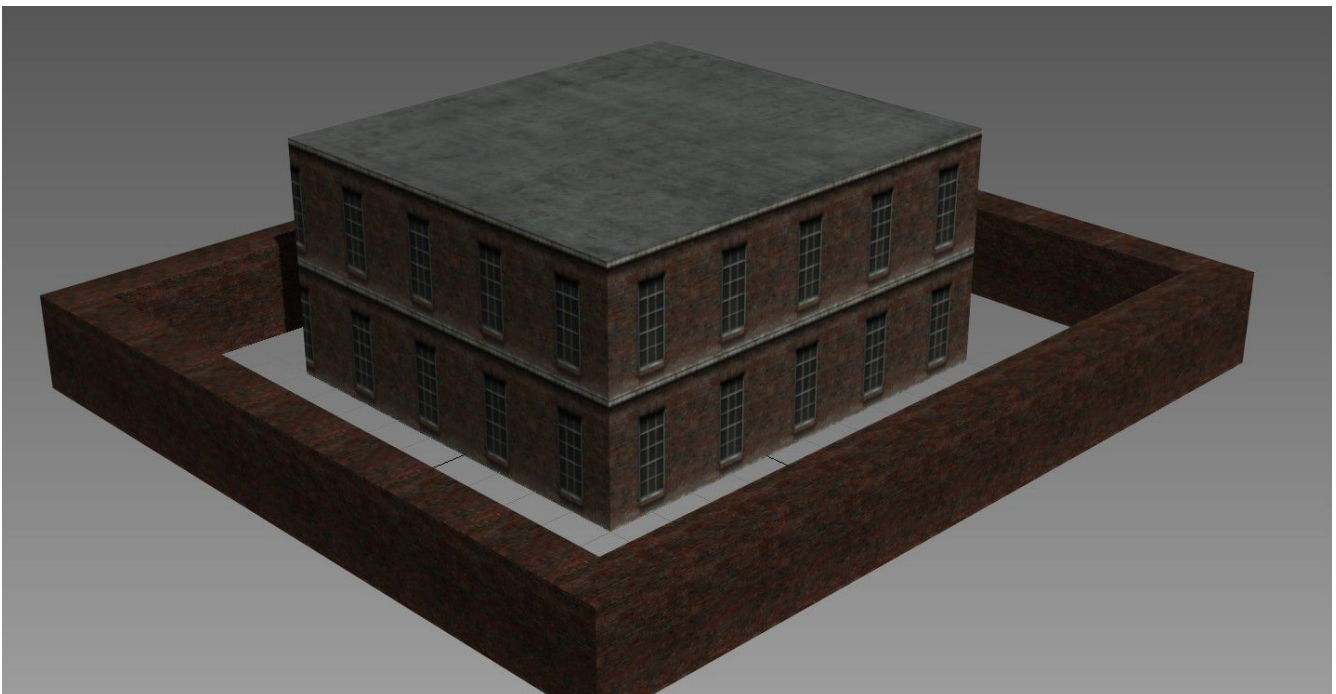
1. **Open** the Slate Material Editor and create three new Standard Materials. Find or create three bitmaps, one of each: brick, brick with windows, concrete. Pipe each bitmap into the diffuse slot of each of the three materials you just made.



2. **Double-click** the brick and enable the Display Shaded Materials in Viewport option. Repeat this for the other two materials.
3. **Select** all the outer walls. Now apply the brick material to those objects by dragging the output circle (on right of material) to one of the selected objects in the viewport.
4. Now **create** a new Multi/Sub-Object Material in Slate.
5. **Pipe** your Brick and Windows material into the first sub-material of the Multi/Sub-Object Material.
6. **Pipe** your concrete material into the second sub-material of the Multi/Sub-Object Material.
7. **Apply** the Multi/Sub-Object material to the building object. At this point, the building may look completely wrong (with un-textured sides and unexpected materials).
8. **Add a Material modifier.** This will tell all face selections at this point in the modifier

stack to use the material in the Multi/Sub-Object material matching the Material ID in the spinner value of this modifier. Since the current face selection from further down on the stack is the roof, we will choose the concrete material—so change the Material ID spinner to 2. Notice the concrete appear on the roof.

9. **Add a Poly Select Modifier**, go to **Polygon sub-element Mode** and select all the side polygons around the building.
10. **Add** another **Material modifier** to the modify list.
11. Notice how the sides now have the brick and windows.
12. To change the UVS of the building side, simply select the UV Building Side modifier and adjust parameters or gizmo.



The methods demonstrated here are useful for controlling generic UVs across multiple objects in a parametric fashion. There are other tools and methods that can be more appropriate depending on the scenario, but the intent here is to get you to start thinking about how some of the different tools can work in Max, how UVs generally work and how to incorporate the Modifier List into your thinking process.

For individual objects that have built-in Material ID controls (such as an Editable Poly) the steps for adding Poly Select modifiers and Material modifiers would not have been needed. Furthermore, we could have added a single Edit Poly modifier instead of the multiple modifiers... but again this is about getting you to think about multiple modifiers and traversing the modifier stack.

When applying some of these principles into your scenes, look for ways to re-use instanced modifiers across objects whenever appropriate, as it makes the process of updating and changing easier to manage. In this example scene, you can re-use the brick wall UVW map modifier on any object in the scene as long as it is aligned to the UVW Map gizmo... and changing this one modifier will affect all relevant objects.

Reusing Your Modifiers

When we added a the outer wall UVW Map earlier, we applied it to a selection of objects all at once. But what if you add a new wall and want to re-use a modifier that already exists? You'll do this by pasting an instance of your modifier.

Let's start by adding a new extension to your wall (perhaps a new Box primitive butted up against the existing wall.

- Select one of the objects from the original wall and open the modify tab.

- Right-click on the UVW Map modifier in the list and choose Copy.
- Select the new wall object and open the modify tab.
- Right-click on the object's entry in the Modifier List and click Paste Instance.

Now your new object will be controlled by the same UVW settings in the original wall. Changes to the modifier on this new object's UVW will be immediately applied to the original walls.

Fine-Tuning Your UVs

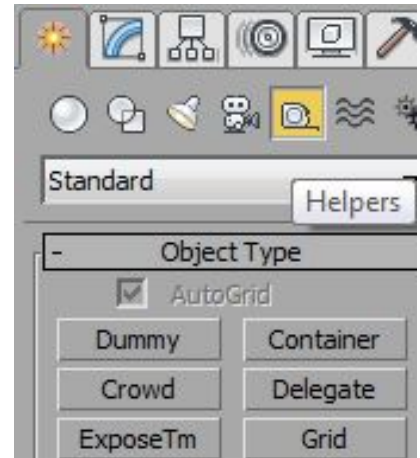
Using the methods demonstrated in the previous sections can take you a long way with aligning your textures across your scene. But there are some limitations with the approach explained above. For example, the UVW Map modifier is best for in the special cases of Planar mapping (a large area like a floor or ground), Box mapping (buildings and walls where the corners are all at 90 degree angles) and Cylindrical mapping.

If your layout is more complex (with walls at many angles) you may need to use another approach: Unwrap UVW.

It's common for Max users to think of unwrapping for models, but unwrapping can also be used for world geometry. Let's demonstrate how we might use UVW Unwrap on our scene.

1. Turn on Snaps.
2. Turn off Snap to Grid.

3. Turn on Snap to Vertex.
4. Open the Create Panel.
5. Choose Helper objects (Tape Icon).
6. Choose the Grid object.
7. Draw a Grid object on the top of the building using the vertex snaps to match the grid to the exact dimensions of the roof.
8. Switch the command panel to the modify tab.
9. Change the Grid spacing to a multiple of 2 (32 should be a good value for now).
10. Right-click the new grid object in the viewport and choose Activate Grid. (To use Home grid *later*, you will right-click the grid and choose Activate Home Grid.)
11. Turn off Snap to Vertex.
12. Turn on Snap to Grid. All grid snaps are on this new grid since it is the new Active Grid.
13. Choose Line from the Splines Categories
14. Draw an octagon on the grid on the roof.
15. Add an Extrude Modifier to the Spline.



16. Change the amount to a value like 128.
17. Turn on Generate Mapping Coords in the modifier.
18. Apply the Brick Material from the outer wall to this new object. You'll notice that the UVW on the default wall object is normalized (and ugly). See image to right.
19. Select your main building and right-click the UV Building Side UVW modifier and click Copy.
20. Select the new wall on the top and paste the modifier onto the modifier list. Now the texturing is a lot better, but there is a noticeable misalignment at some of the corners where the textures are not seamless.
21. Add an Unwrap UVW modifier to the object.
22. Click the Open UV Editor.
23. Display the wall texture but selecting it from the drop-down menu at the top-right of the Edit UVWs window.
24. Click the Polygon Mode.
25. Select a Polygon. You can now move, scale and rotate your textures as you want.



To continue to describe the steps would be too much for this document. At this point, you should take time experimenting with the UV Editor to transform the UVs as desired. You should pause and [watch this set of Videos on UV Unwrapping](#).

Keep in mind that when transforming/editing UVs for world geometry, you must always keep all mapping faces per polygon together and planar. To make this easier, keep all mapping faces welded.

Better Method than Unwrap UVW: PolygonMap

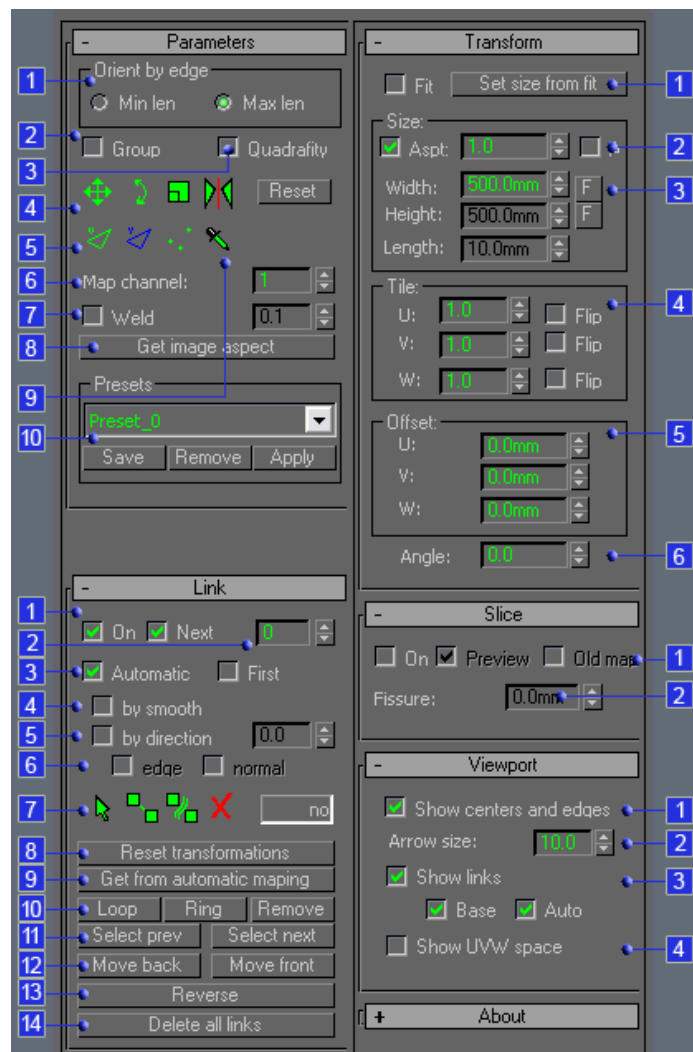
Once you take some time to learn about Unwrapping, you may notice a few things, especially how inconvenient it can be to align textures with the methods above. Using Unwrap UVW, you may take more steps in Max than you would in Hammer for the simple case of the object we made on the roof above.

Now we come to the first must-have third-party plugin on this book: [PolygonMap by VG Plugins](#). This plugin is specifically designed at helping you quickly apply planar UV mapping

to objects in a way that is necessary for Source Engine world geometry.

Unlike with Unwrap UVW where you must open the Edit UVW window to manipulate UVs, PolygonMap allows you to manipulate the texture coordinates directly in the Max viewport. It feels more natural, and it affords a much more Hammer-like experience.

PolygonMap is the level design UVW tool for ninjas! It has all the tools Hammer users are used to (getting texture transforms by selecting a face in the scene) plus a whole lot more. Download the demo ver-



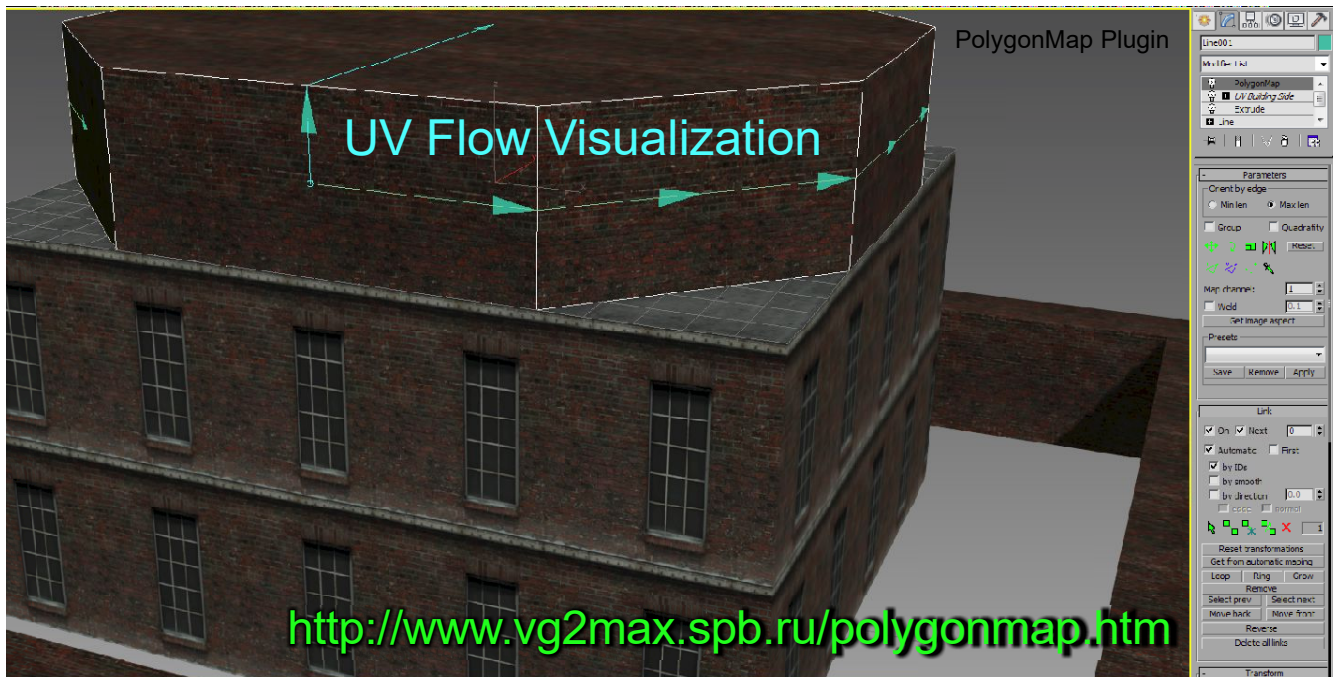
sion now, install and restart 3ds Max. Make sure to read the complete installation instructions that come with the file, as it requires a function library with download instructions in the readme file.

In the case of the object we made in the last scene, the UVs can be transformed in a couple clicks and manipulated visually in the viewport.

1. **Open** the scene we made in the previous section.
2. **Select** the object we added to the Roof.
3. **Open** the **modify tab** and add a **PolygonMap** modifier.
4. **Click** the **Use Old Mapping** check box.



5. **Click** the **Pick UV Transform** button.
6. **Pick** the bottom left corner of a polygon to set that UV point as the transform for the UV flow.
7. **Un-check** the **Use Old Mapping** checkbox.
8. It is very likely the UVs now tile seamlessly across the object. If not, use the link tools to click from polygon to polygon to control the flow.



PolygonMap is an essential tool for your level design projects. For added features, consider buying the commercial version that adds extra features such as the ability to save and reuse presets. PolygonMap has only one limitation that I can think of, which is that it doesn't work as expected when instanced across objects. If enough Wall Worm users who purchase the commercial version request this capability, it might be possible to get the developer to add the functionality.

MapScaler Modifier

Another modifier that works similarly to PolygonMap is the native MapScaler modifier. You can use MapScaler to quickly force UVs to flow around an object. While useful, it does not have as many options and as much control as Unwrap UVW or PolygonMap. To see how to utilize MapScaler, see this video on [Automatic Planar UVs](#).

Case Study: Texturing Models

Texturing a model is usually not the same as texturing world geometry. Although there is nothing stopping you from using the techniques in the previous section, models usually have a different set of needs especially if they are not simple objects with tiling textures. Whereas world geometry must always be composed of tiling planar texture coordinates, models do not have this restriction. In fact, to get the most realistic look, most models will avoid tiling textures.

Unlike world geometry where Unwrap UVW can be a clunky method, with models unwrapping is almost essential. As recommended before, it is highly recommended that you watch this set of [videos from Autodesk on UV Unwrapping](#).

Design Setup

Texturing a model can be the topic of an entire book by itself. There are vast number of tools in 3ds Max to do amazingly advanced and complex texturing. Explaining the entire range of these tools is far beyond the scope of this book. If you are a new Max users, you should take note of the topic list at the end of this section to guide further learning on some of these things.

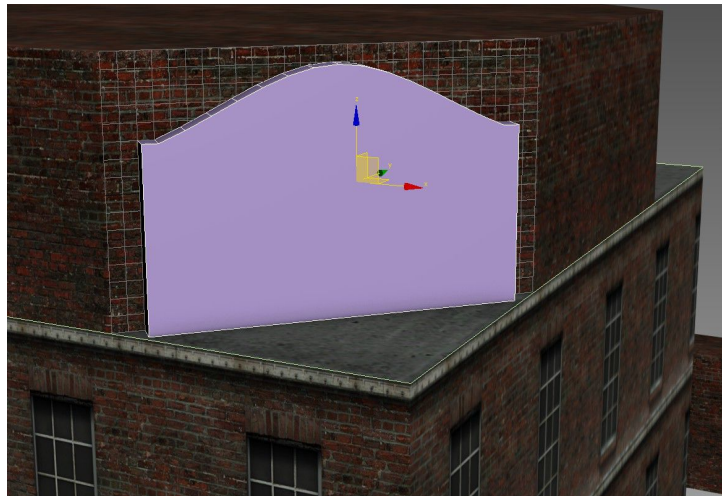
In this example, we are going to unwrap a simple model for texturing. The intent is to demonstrate briefly how the unwrapping and texturing work with the intent that you will delve into the already extensive resources on the subject of unwrapping that are available from Autodesk and the many freely available on-line resources.

Let's build a simple model of a billboard to put on the building we made in the last chapter.

1. Turn on **Snap to Vertex**.

2. Click the **Create Panel** and choose the **Helpers Category**.
3. Choose **Grid**.
4. **Turn on AutoGrid** (checkbox near top of Object Type buttons).

5. **Create** a new **Grid** object on one of the diagonal sides of the top part of our building from the last chapter and activate the Grid. By using Snap to Vertex and AutoGrid, you should be able to make this easily fit the entire Polygon.



6. **Create** a new **Spline**.
7. Draw a billboard outline (in a fashion outlining the geometry demonstrated to the right).
8. Add an **Extrude Modifier** with an amount of 12 and Generate Mapping Coords. on.



9. To make it easier to work with this one object, **click** the **Isolate Selection Toggle** at the bottom of the window.

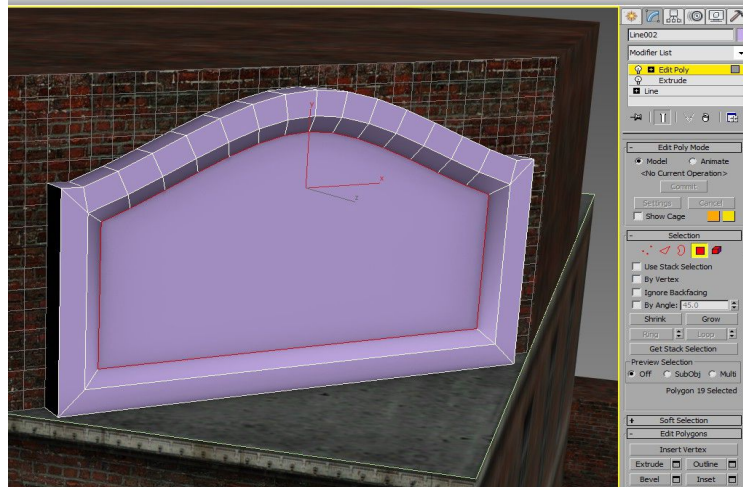
10. Add an **Edit Poly Modifier**.

11. Go to **Polygon Sub-Object mode** and select the back and bottom polygons (that are normally hidden from view).

12. Press the **Delete** key on the keyboard. This will remove those polygons.

13. **Select** the front polygon and click the **Inset** button in the **modify** tab. Now click and drag to create a bordered set of polygons.

14. Switch to the **Bevel** button and bevel the front central polygon to create a depression.



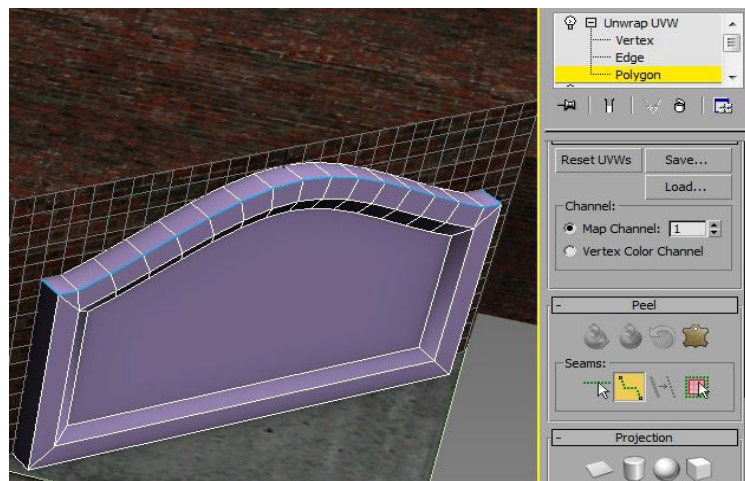
15. **Toggle** the **Isolate Selection** to see entire scene again.

16. **De-select** the front **Polygon**.

17. **Exit** Polygon Sub-Object Mode.

18. Add an **Unwrap UVW Modifier**.

19. In the Unwrap UVW Modifier, click the Polygon mode. (There may be a UI bug where this is highlighted by default, so you may have to click it twice to actually enter the correct mode.) Notice that the existing Map Seams are dis-



played on your model as green lines.

20. Un-check the Map Seams display option.



21. Click the **Point-to-Point Seams** button.

22. Make a peel seam boundary across the top section. It appears as a blue line.

23. **Right-click** when done making the peel seam.

24. **Exit Point-to-Point Seam** mode.

25. **Select** the main, front polygon.

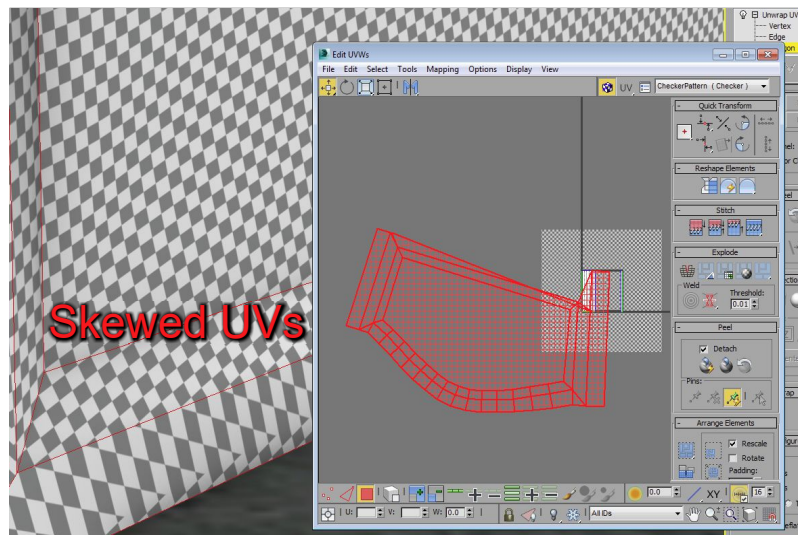


26. Press the **Expand Polygon Selection to Seams** button.



27. Press the **Quick Peel** button. This will open the **Edit UVWs** window.

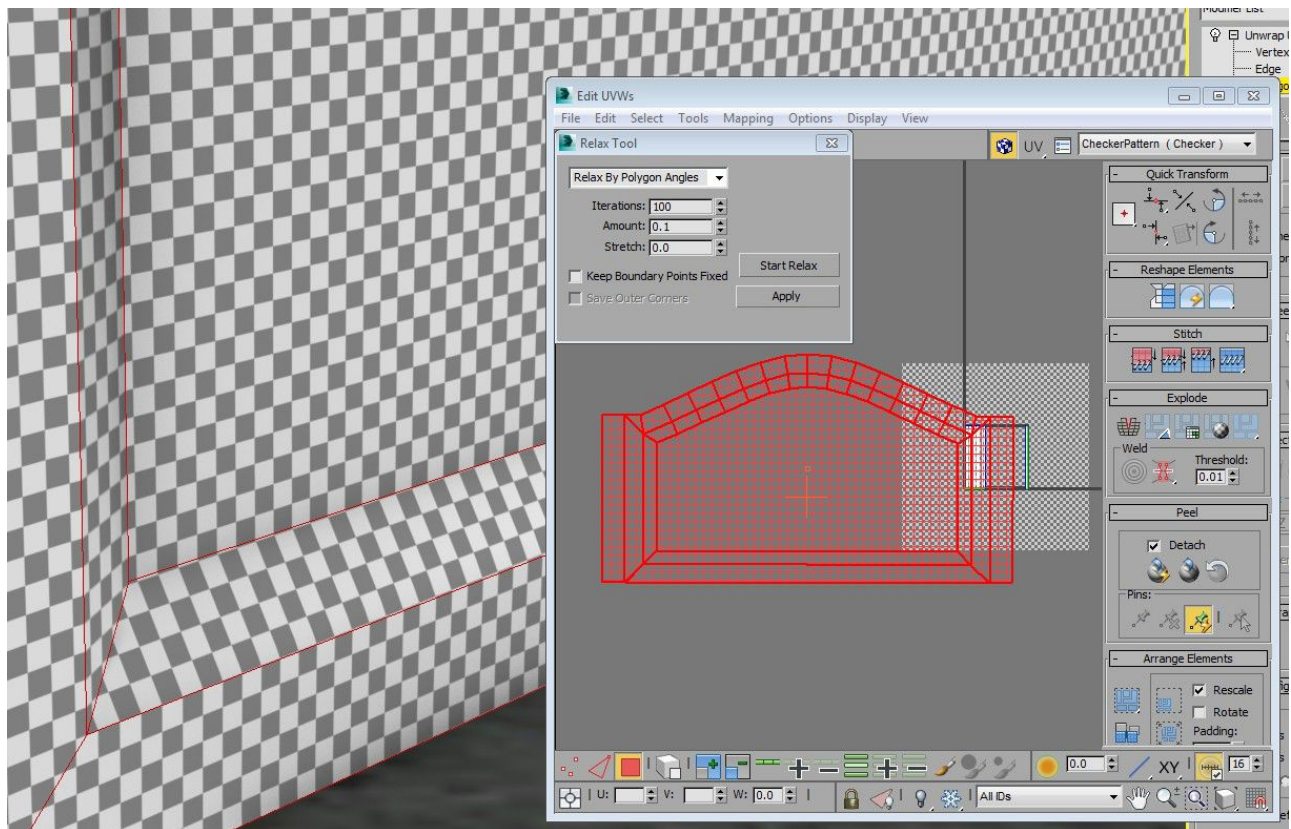
28. Click the drop-down menu at the top-right of the Edit UVWs window and pick the **CheckerPattern** (even though it was already selected). This will preview the mapping with a checker texture. Notice that



some of the checkers are skewed.

29. To fix the bad UV stretching, we must **Relax**. Click **Tools > Relax**.

30. Select the **Relax By Polygon Angles** options.

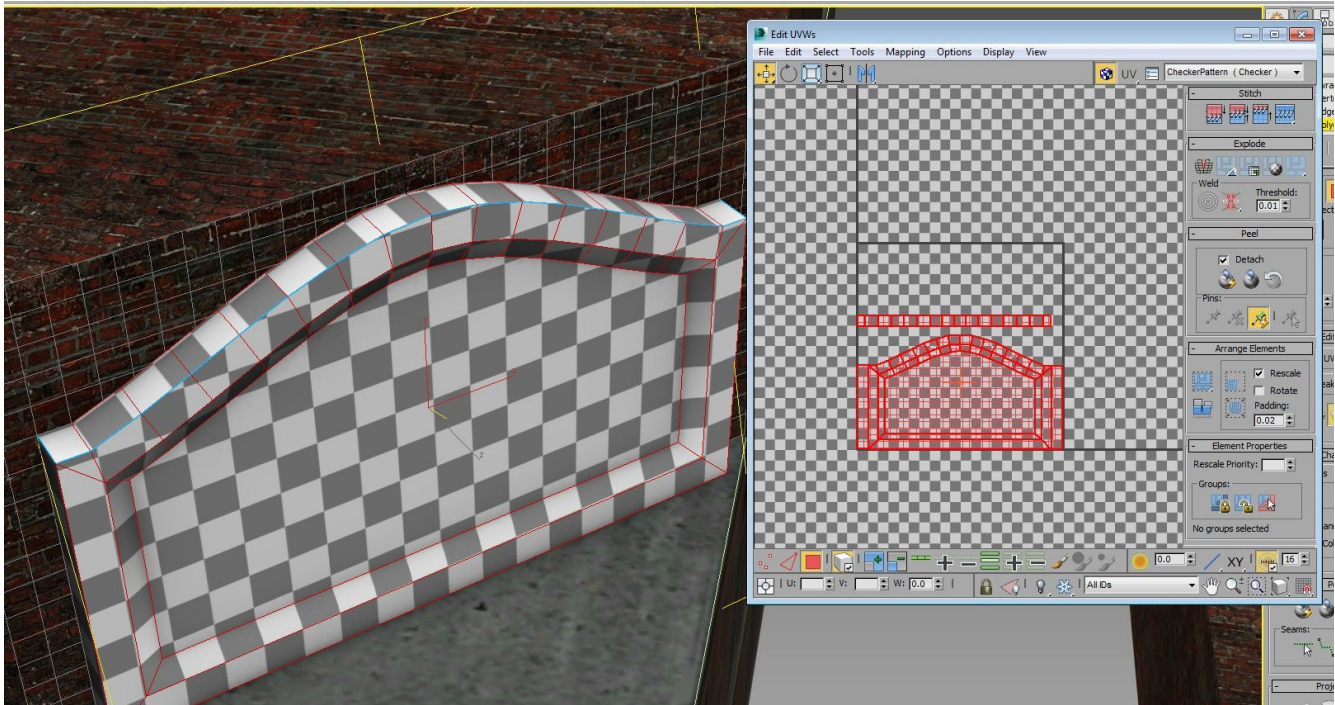


31. Press the **Start Relax** button. Watch the UVs look more uniform in the viewport.
(Compare the Uvs to those in the previous graphic.)

32. Now we need to repeat steps 25-32, except this time start by selecting one of the polygons from the top of the model.



33. Now Select all of the Polygons and click the Pack Normalize button. You should end up with something similar to the following image.



Adding the Materials

Now that we have unwrapped the model, we can add some materials. There are going to be two phases. In the First phase, we are going to create generic materials from procedural maps and some bitmaps.

1. **Right-click** the main model and click **Clone**. At the prompt, choose **Copy**. Select this new clone.

2. **Right-Click** this new clone and click Convert To > **Convert to Editable Poly**.



3. **Open** the **Slate Material Editor** (click M).

4. **Create** a **Standard Material**.

5. **Select** Material in Slate.



6. Turn on **Show Shaded Material in Viewport** in Slate.

7. Pipe a **Noise** map into the **Diffuse Color**. Pipe the same map into the **Bump** slot of the material.

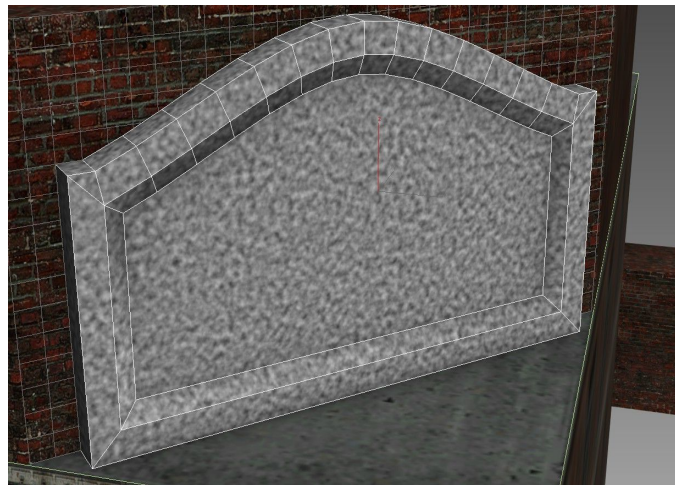
8. Set the following parameters in the Noise map: **Noise Type = Fractal** and **Size = 1.0**.

9. **Apply** this **material** to the model.

10. To add some text, we will start with a new model in the scene.

Open the **Create Panel** and choose the **Shapes** object type.

11. **Click** the **Text** button.



12. **Turn on Snaps** and only use **Snap to Face** and Enable Axis Constraints (turn off all other snap types).

13. **Create** a text shape on the main area of the model's face.
14. **Alter** the text size, font and position to one better for this.
15. **Add** a **Bevel Modifier** to the text object.
16. Now **create** a new **material** with a gold color and apply this to the text.

17. **Create** any other objects that you want to be rendered into the texture of you model.

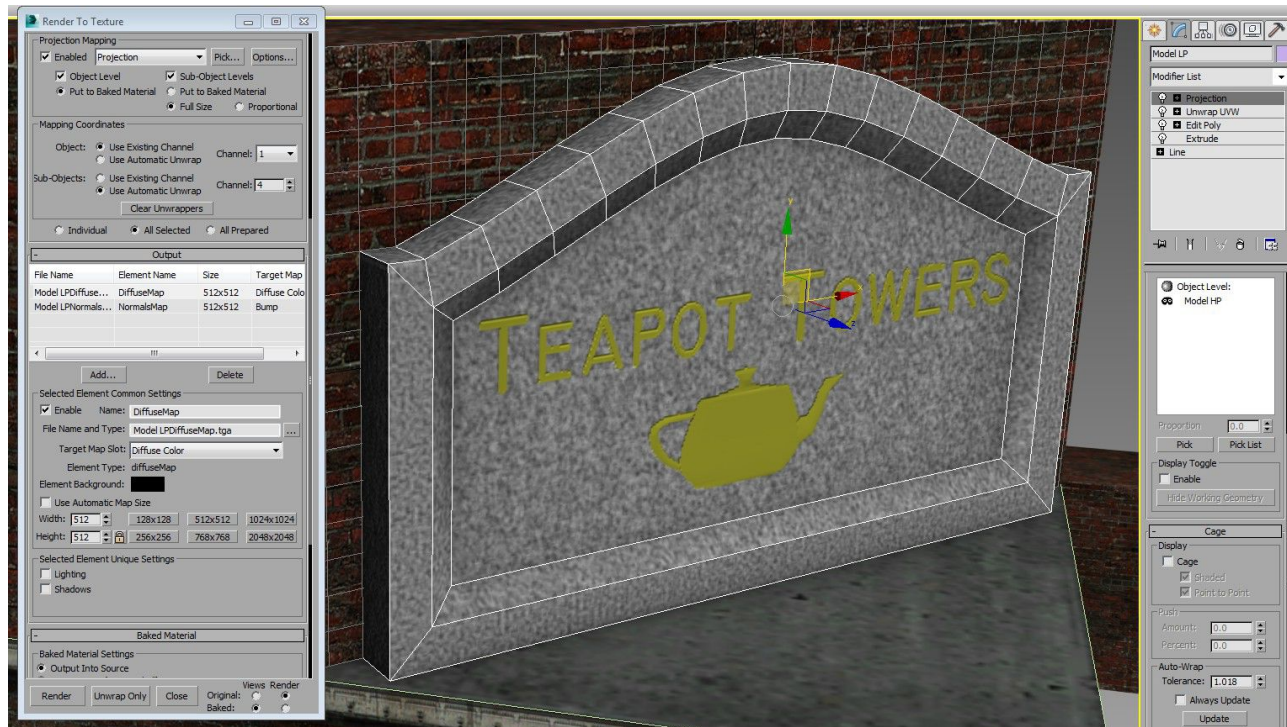
18. **Rename** this object "Model High Poly".

19. **Edit** this object if you'd like (for example, adding extra beveling along corners).



20. Now we have two sets of objects: the original low-poly model and all of the more detailed models (the high poly models). We are going to bake the geometry of the high poly models into the low poly model's material.
21. **Select** the Low Poly model and add a **Projection** modifier.
22. In the **Projection** parameters, **click** the **Pick List** button in the **Reference Geometry Rollout**. **Choose** all of the High Poly Objects. In the case of the above image, it would include the higher-detailed sign model along with the text and the teapot.

23. **Enable** the **Cage** and **Shaded** option in the Projection modifier.
24. **Increase** the **Push** amount. Notice the transparent cage grow slightly. You will need to push it enough so that all objects you want to include in the baking are surrounded. If the cage starts to intersect on itself, you may have to edit the cage manually. See the documentation on editing the Projection Cage. Or watch this [video by Louis Marcoux](#).
25. **Open** the **Render To Texture (RTT)** dialog (press 0).
26. Make sure that it is your low poly object that has the projection modifier and is currently selected. You should see it listed in the Objects to Bake list in the RTT dialog.
27. **Enable** the **Projection Mapping** option in RTT.
28. In the **Mapping Coordinates** section, make sure to change the Object option to **Use Existing Channel** and the channel is set to 1 (as this should be the same as the Unwrap UVW channel we added earlier).
29. In the **Output** menu, click **Add**. In the prompt, click DiffuseMap. Hold down the control key and add the NormalsMap.
30. **Click** the **Add Elements** button. You'll see two bitmaps listed in the output list.
31. **Select** the DiffuseMap in the list so you can edit the output settings for the bitmap. **Change** the dimensions to 512x512. Do the same for the NormalsMap.
32. In the **Baked Materials** section of RTT, choose the option to **Output Into Source**. (There are times to choose other options but for this tutorial it is more convenient.)



33. Now **click** the **Render** button at the bottom.

34. **Disable** the **Cage** view of the Projection modifier. Now you should see that the texture on your low poly model has the high poly detail baked into it. The following image shows this.

35. Now we can paint some extra details with . Click Tools > **Viewport Canvas**.



36. **Click** the **Paint** button in Viewport Canvas and choose to paint on the **Diffuse Color** (not the Bump map).

37. If a Layers floater did not appear, you can open it by clicking the **Layers Dialog** under the color swatches. Now you can add layers as you would in a bitmap editor like PHOTO-PAINT, Photoshop, Gimp, etc.



38. **Add** a new **Layer** for adding some grime.

39. In the **Brush Images** section click the Color box and choose the Rust image and for the Mask choose the Dirt black and white image.

40. Change **Rotation** to **Random**.

41. Start **painting** on your model.

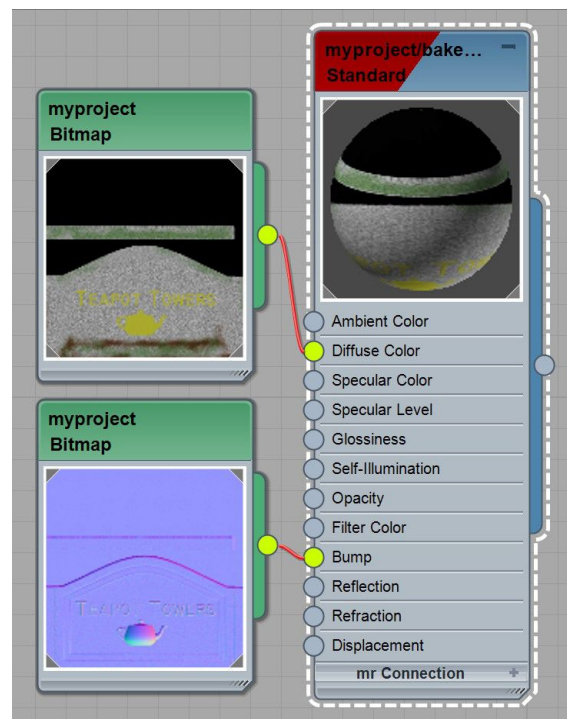


42. **Create** extra layers for different elements and modify layer transparency and blend mode in the layers dialog.

43. Continue painting until you are finished. Then **right-click** the viewport background.

44. At the prompt, you can either collapse all the layers into the original bitmap by clicking the **Flatten layers and save the current texture**. However, if you want to edit this image later in the Viewport Canvas again (or if you want to edit it in another bitmap editor) choose the option labeled **Save as PSD and replace texture in material**.

In this example, we started with a Standard Material that was composed of procedural textures (the noise map). We could have made the original texture tree far more complex with an number of layered texture trees. We then rendered the procedural textures and geometry into flattened bitmaps, which we edited with Viewport Canvas. What we ended up with is a Standard Material with flat bitmaps in the diffuse color and bump channels as in the graphic to the right.



Learn More About Texturing

The preceding section was a quick example of how to unwrap a model and Render to Texture. However, the lesson was very basic. Not demonstrated is how the model's topology limits the way you unwrap your model. In the case of the model above, its topology could be edited to allow UV breaks that could help fill up all the texture space, which is important for keeping your texture quality higher and reducing your project's file-size expense.

One aspect of textures and UVs not discussed here is something called Real World Scale. You can simplify many kinds of texturing tasks by learning about real world scale tools in 3ds Max. Because Real World Scale is limited to certain kinds of geometry and textures, I did not

discuss this. See [Real World Mapping](#).

Here are some good resources on topics that will enhance your capabilities.

- [Autodesk Videos on UV Unwrapping](#)
- [Ted Boardman's Unwrap UVW fundamentals](#) (uses Max 2010 and an older version of the UV editor, but the principles are explained well)
- [Slate Material Editor](#)
- [Using Viewport Canvas](#)
- [Louis Marcoux Tips and Tricks on Viewport Canvas](#)

Chapter 19 CorVex and Parametric Level Design

This chapter will cover using parametric design principles and the [CorVex level design plugin by Wall Worm](#). While the free Wall Worm tools include integration with CorVex, CorVex is a separate inexpensive product. You can purchase it with the link above.

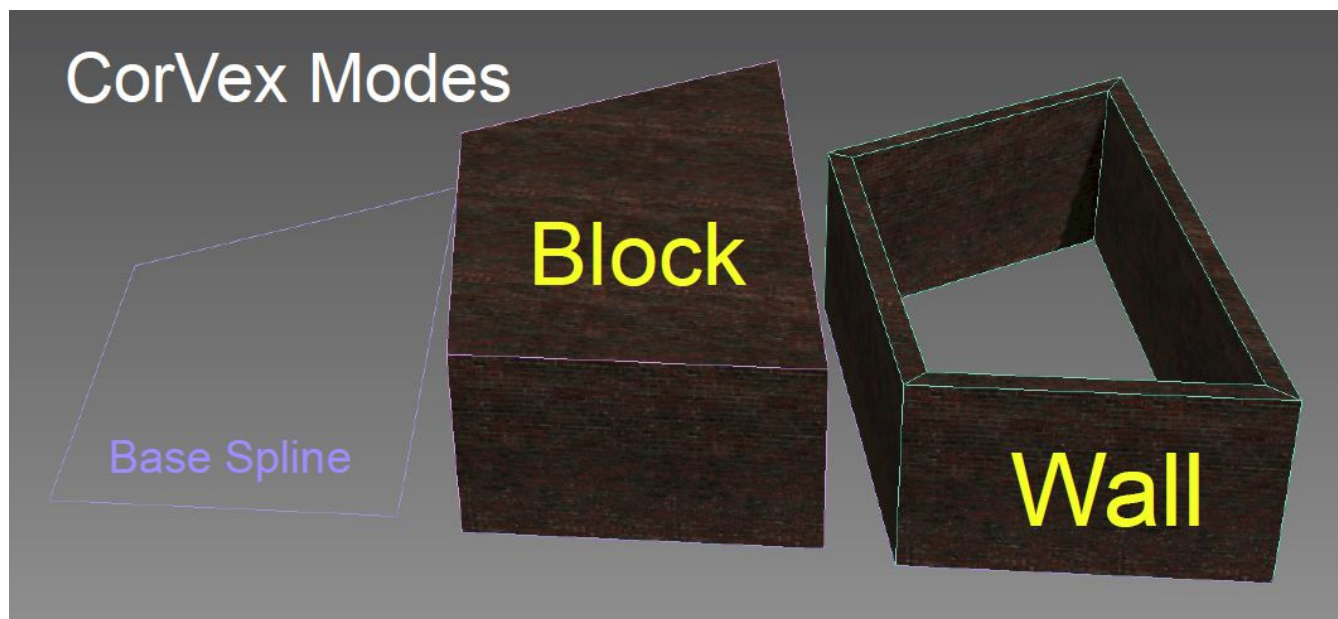
CorVex is, like any other primitive object in 3ds Max, parametric. That means it stores parameters that control aspects of the objects. These parameters can be edited non-destructively.

Parametric design is an alien concept to the Hammer philosophy of level design. But it affords greater flexibility and design efficiency.

Introduction to Using CorVex

CorVex's main purpose is for creating your level's world geometry and layout. It can also be used for making models, but the majority of its tools target BSP level design.

CorVex has two basic modes: Block Mode and Wall Mode. These two modes are for two distinct kinds of tasks in your layout. By default, new CorVex objects are in Block Mode. All



CorVex objects derive the base from Spline shapes in the scene.

In this example, both CorVex objects are using identical splines for generating their geometry, but in the block mode each spline becomes a single block. In Wall Mode, each segment of the spline becomes a separate wall block.

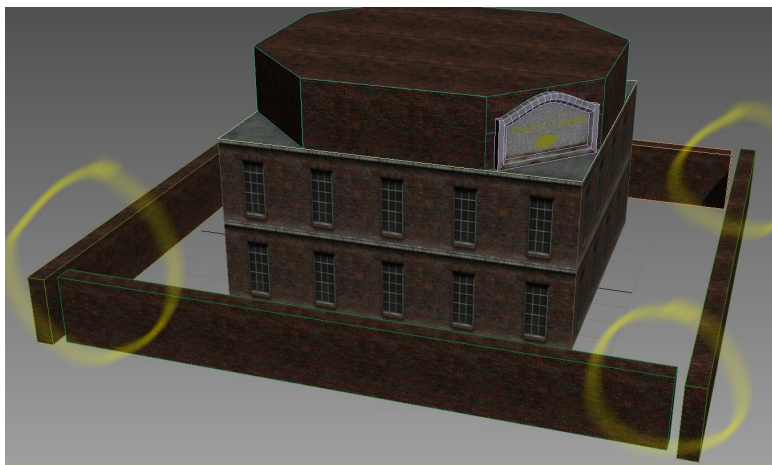
Since the splines that drive the objects are parameters of the CorVex objects, editing the splines changes the CorVex geometry.

When working with CorVex, the philosophy is all about keeping everything simple and efficient. You use one or more CorVex objects to drive block geometry (like buildings that have no interior, ground, Hint brushes, etc). And you use one or more CorVex objects to drive wall systems (buildings you can enter, section divides, map borders, etc). In either case, portions of the level that are going to share similar properties (like height, materials, etc) can often be a single CorVex object. For example, one CorVex object in Wall Mode can often be the entire level border—and the height of all parts of the border are controlled with one single spinner in the UI.

Unlike most standard Max objects, the controls for the UVW coordinates are built right into the object parameters. For the most part, you should not need to use UVW Modifiers when working with CorVex. When you do need to use a UVW Modifier, you can target specific sections (like sides, tops, bottoms, etc) with built-in selection functions.

Your First CorVex-driven Geometry

In this section we are going to return to the scene we've created in the previous chapters. First, let's discuss a new scenario: imagine



you want to change the height of the outer walls and make them less thick. As the objects are currently, this would not be straightforward. Because those walls were made of Box primitives, changing the width and heights is fairly simple by selecting each wall and changing the dimension parameters of each. It will be tedious, but possible. But as soon as you've finished, you'll notice that there are now gaps between the objects. This forces you to move the objects and reset the dimensions.

An option for editing the walls would be to convert them to Editable Poly and move the various sides with the standard Max transform and snap tools. But still, this is fairly tedious.

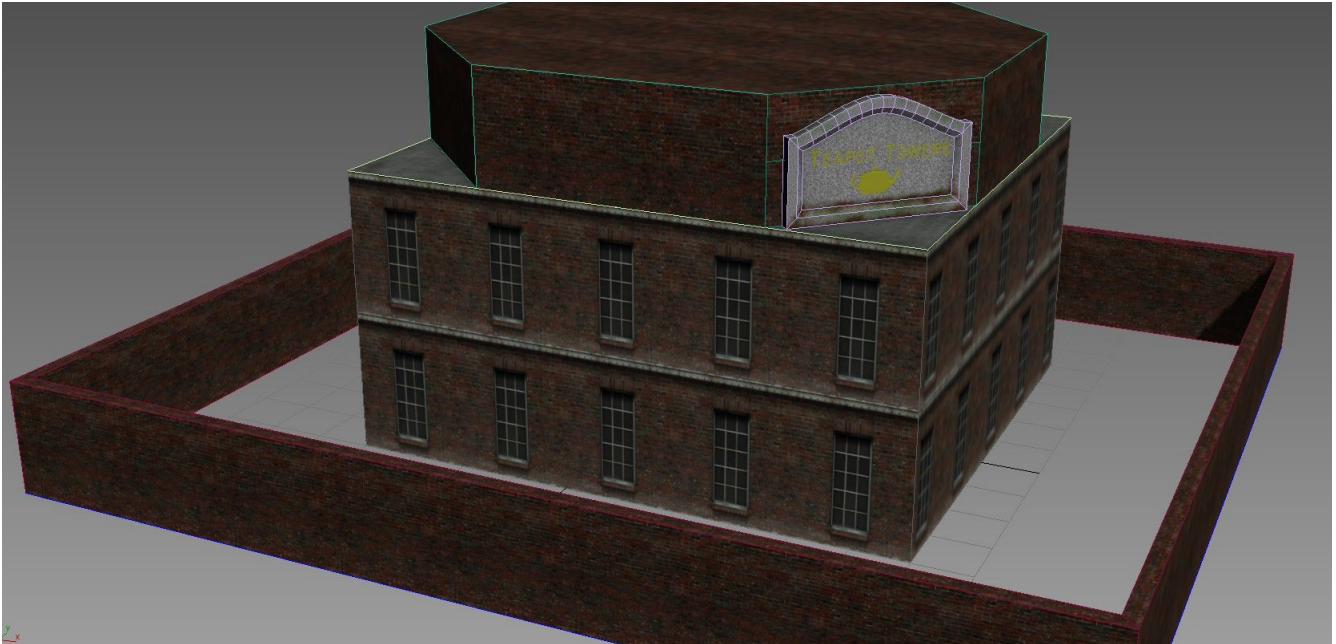
Your answer to this is CorVex.

Rebuild the Wall With CorVex

1. Delete the outer wall created earlier.
2. Open the **Create Panel** and choose **Wall Worm** from the geometry category list.
3. Click the **CorVex** button and add a CorVex node anywhere in the scene.
4. Right-click to exit **Create Mode** and open the **modify tab**.
5. Click the **Wall** checkbox.
6. Turn on **Snaps** and **Snap to Grid**.
7. Click the **Add New Spline** button.
8. **Click the grid points** in the viewport that represent the four corners of the boundary of

the original wall. Notice that the walls are created from the last segment of the wall to each new segment you create.

9. When you **click** on the **same point** the wall started on, you'll be prompted to close the spline. Choose **Yes**.
10. Now **scroll** down in the **Utilities** rollout and click **Create Multi Material**. The object now has a default multi material with a checkered texture. This will help you visualize the Uvs.
11. Now you can change various parameters of all the wall parts at once such as the height or thickness of the walls.
12. For walls, you'll get best results if you choose these parameters: **Side Method = Wall Match Both Sides**. If you want the top to flow along with the wall, choose **Wall Top/Bottom Method = From Side**.
13. Open the **Slate Material Editor**.
14. Click the **Pick Material From Object** button. Pick the new CorVex object as well as the octagonal. Now you'll see the **Multi/Sub-Object Material** on the CorVex object and the **Standard Material** top of the building.
15. Pipe the Brick Material into the first slot of the Multi/Sub-Object Material (replacing the link to the orange checkered material).



Now you can always add walls to this CorVex object by clicking the **Add New Spline** button or by editing the current spline that controls the wall.

Using Instanced Splines & Wire Parameters

Now we will explore methods for optimizing your work flow with instancing objects and wiring parameters together. We are going to create a wall above the existing outer wall that will always match the wall on the bottom. The goal is to create objects that directly influence one another so that a change in one will immediately be reflected in the other.

Instancing Objects

Instanced objects are always identical geometrically. Whatever you do to one instanced object immediately happens to all other instances of it. The instancing applies to all parameters you'd find for the object in the modify tab. However, they do not have to share the same materials or transformations.

1. Select the base spline that controls the outer wall. (You can do this easily by selecting the current CorVex object and double-clicking the spline in the Base Splines list in the modify tab.)
2. Right-click the viewport and choose **Clone**. When the dialog prompts you with clone options, choose these settings: Object = Instance and Name = "BoundarySkyBase".
3. Now BoundarySkyBase is the current selected Spline. Notice that it's Base Object (named Line) in the modify stack is now bold. This indicates that it is an instance.

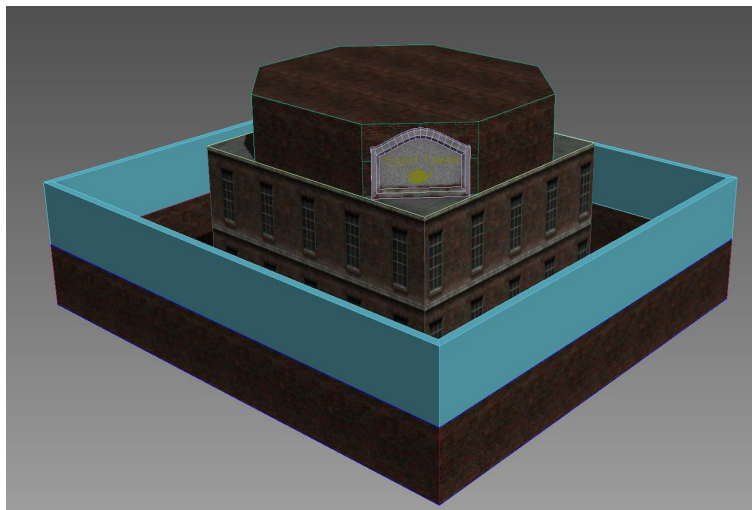
Wiring Parameters

When you wire parameters, you force objects to control one another. This is convenient because it allows you to reduce the number of times needed to make any edits.

1. Right-click the BoundarySkyBase and choose **Wire Parameters**.
2. At the prompt, choose **Transform > Position > Z Position**.
3. Now you need to choose the objects in the scene and it's parameters to wire. As you move the mouse, you'll see a dotted line to indicate your are wiring a parameters. Click

on the outer wall CorVex object.

4. At the prompt, choose **Object (Corvex) > offset**.
5. Now there is a Parameter Wiring dialog. The dialog displays various parameters between the Spline object (on the left) and the CorVex Object on the right. The parameters chosen for wiring are highlighted. We want to use a **Two-Way Connection** for this wiring. Click the button with this symbol: <--->. Then click the **Connect** button. You can now close the dialog.
6. You may notice that your wall is suddenly flat. This is because it's height (in the offset parameter) is now bound to the z-position of your BoundarySkyBase spline.
7. **Select** the Wall CorVex and bring its height back to 128 units. Notice that the BoundarySkyBase spline is now aligned precisely at the top.
8. **Switch** to the **Create Panel** and choose the Geometry object type.
9. **Select** Wall Worm from the Category drop-down.
10. **Click** the **CorVex** button and **add** another CorVex object to the scene.
11. **Switch** to the **modify** tab.
12. **Scroll** down to the **Utilities** rollout and click **Get Settings From CorVex**.
13. **Pick** the original wall ob-



ject. This transferred all settings (except splines) from your original CorVex.

14. Now **scroll** up to the Spline Bases menus in the modify tab and click **Add Existing Spline**.

15. **Pick** the **BoundarySkyBase** spline. Now there a wall above the original wall.

16. **Select** the bottom CorVex object. **Change** its **Height** and notice what happens to the upper wall.

17. **Reset** the brick wall at the base to 128 units.

18. **Select** the octagonal part of the roof top and **rename** it to “BuildingTower”.

19. **Select** the second CorVex object. **Rename** it to “SkyBoundary”.

20. **Right-click** SkyBoundary and choose **Wire Parameters**.

21. **Choose Object (Corvex) > wallWidth**.

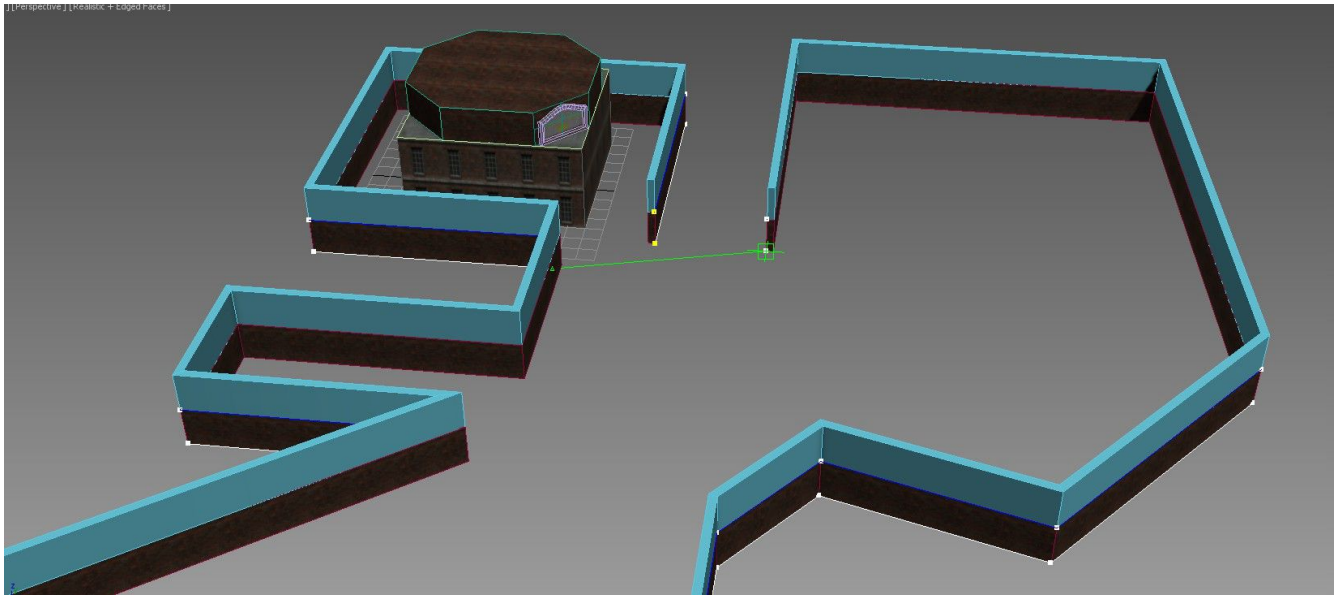
22. **Pick** BuildingTower and pick **Object (Corvex) > wallWidth**. In the Parameter wiring dialog, make the parameters a two-way connection and hit Connect. Close the Parameter Wiring Dialog.

Now we have two CorVex objects that control each other's wall width. We also have instanced splines that the two CorVex objects use for the wall layouts. At this point, all we need to do is edit the bottom spline on the ground and both CorVex objects will immediately update.

Modify an Existing CorVex Spline Base

1. **Select** the bottom CorVex and **rename** this object as “BrickWalls”.
2. In the **modify tab**, double-click the spline name in the Base Splines list. Now the selection changed to that object in the scene. Rename this spline as “BrickWallLayout”.
3. Turn on **Snap to Grid**.
4. Enter **Vertex** Sub-Object mode.
5. Click the **Refine** button.
6. You can now click grid points along your spline to add vertex points. **Click two points** which will represent corners for extending the walls in more complex ways.
7. To open up a section, you'll need to **select a vertex** and click the **Break** button. **Delete or move** an end vertex to make an opening in the wall.
8. To extend a wall, click the **Create Line** button and start from an existing end vertex. Use this method to create an outline boundary for your brick wall. Bring this wall all the way around until it comes back to another open end in this spline and click on the open vertex.
9. When prompted to **close the spline**, click **Yes**.

Notice that both CorVex objects match each other. You can continue to edit this layout by editing a single spline without ever having to manually align blocks and clip brushes were you to do this in Hammer.



With this type of control, all you do is edit the spline.

You do not have to edit the spline at the vertex level. You can also change to Segment sub-object level and move/delete segments.

From now on out, simply edit this spline to control the layout. You can add new sub-splines to this with the Create Line in the modify tab. Just remember to keep all the spline vertices on the grid.

Sealing the Level With CorVex

In this example, we've started with the walls. You could also have started with the floors. In any event, we will now use CorVex objects in Block Mode to make the floors.

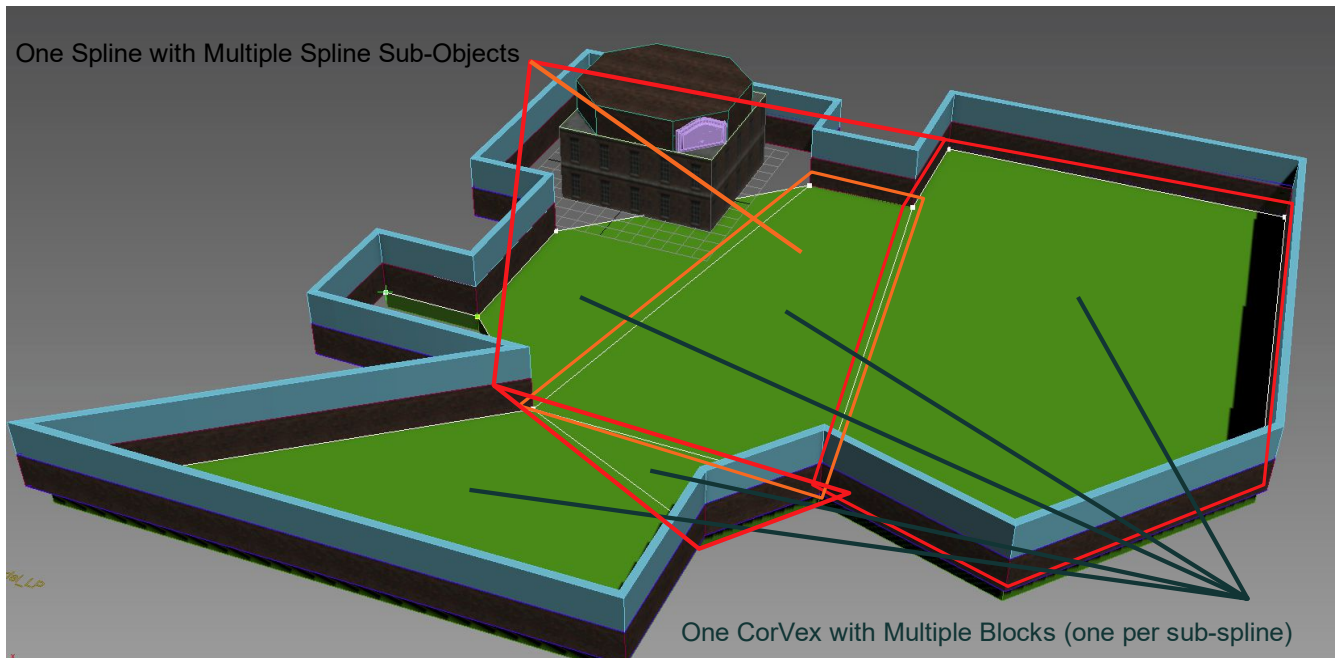
First, notice how the vertices in the control splines of the wall are on the outside of the level (rather than on inside corners). To make your job easier with this layout using the current grid spacing, we want to move the wall to the other side of the control splines. You have two ways of accomplishing this.

- Change the Wall Width to a negative Value.
- Or select the spline's spline sub-object mode and click the Reverse button in the modify tab.

Once you've done that, it will be more convenient to add splines inside the level for other walls and for the floor blocks.

Now let's start with the floor blocks.

1. Add a new **CorVex** object to the scene.
2. For the **Height** of the block, enter a value of -64.
3. Now click **Add New Spline**.
4. We need to start adding multiple blocks that must remain convex. Turn on only snap to Vertex and create convex sections of the floor.



Now you can add a Multi-Material to this object as you did with the wall objects, or you can apply a single ground material.

To seal this level off, simply raise the height of SkyBoundary so that it is above our building. Instance the spline used for the ground, move that new instanced spline to the top of SkyBoundary, and make a new Block CorVex using this copied spline.

Continue adding CorVex objects and props to fill up the scene.

This layout is posing some potential problems for later on in the design process, though. *If we wish to use displacements on the ground*, it will be far easier and cleaner if we make all of the top faces of each block a quad (four-sided polygon). Notice in the preceding graphic that the blocks outlined in red do not have four sides. Also look at the empty area not yet blocked

around the building. For our purposes, we will get better results if we block that section out such that no splines go under the building (especially if the intent is to use displacements or have tiling textures that follow contours in the landscape).

If you don't intend on using Displacements, the design above poses no problems.

Planning Your CorVex Layout with Displacements in Mind

If your level is going to use displacements, its best to plan for them from the start. Planning ahead can save you some headaches as well as make the process work faster. When starting the layout with CorVex objects that serve as the template for displacements, keep in mind the following principles:

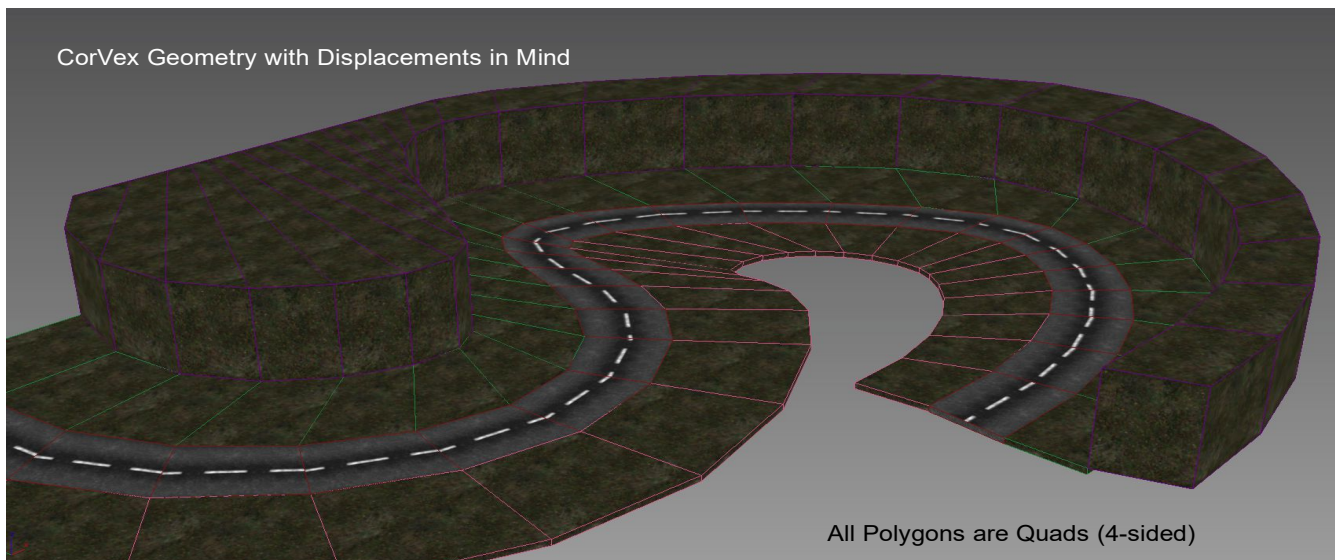
- Hold off creating your displacements until you are confident with the design of the CorVex layout.
- For all parts intended to be converted to displacements, keep all sub-splines in your Block CorVex splines four-sided. (This is not necessary for Wall CorVex.)
- To ensure that all parts of your displacement can be easily sewn, keep all touching sides of displacements aligned to each other and of equal length. Because Wall Worm does not support slicing (clipping) displacements, this is best considered at creation-time.
- Get a good idea of the texture scale you want to use for the displacements before created by applying the texture you want to use on the displacements onto the CorVex object. Adjust the CorVex UV settings before creating the displacements.

Make Displacements from CorVex

For a concise demonstration on how to use CorVex as the scaffold for your landscape you should watch this video on [Roads and Terrain in 3ds Max with CorVex](#) or [Wall Worm Level Design Episode 1](#).

The main principle to keep in mind when building a basic landscape scaffold for displacements is that you must keep all polygons 4-sided and planar (both geometrically and in the UVs). This applies not only to CorVex base geometry, but even geometry bases such as planes, etc.

To more easily create complex terrains, you can use the CorVex Utility Floater to mass generate CorVex splines and CorVex objects from a few splines that represent your terrain's flow. This floater is demonstrated in the video above and is how the following image was created by simply drawing five (5) splines.



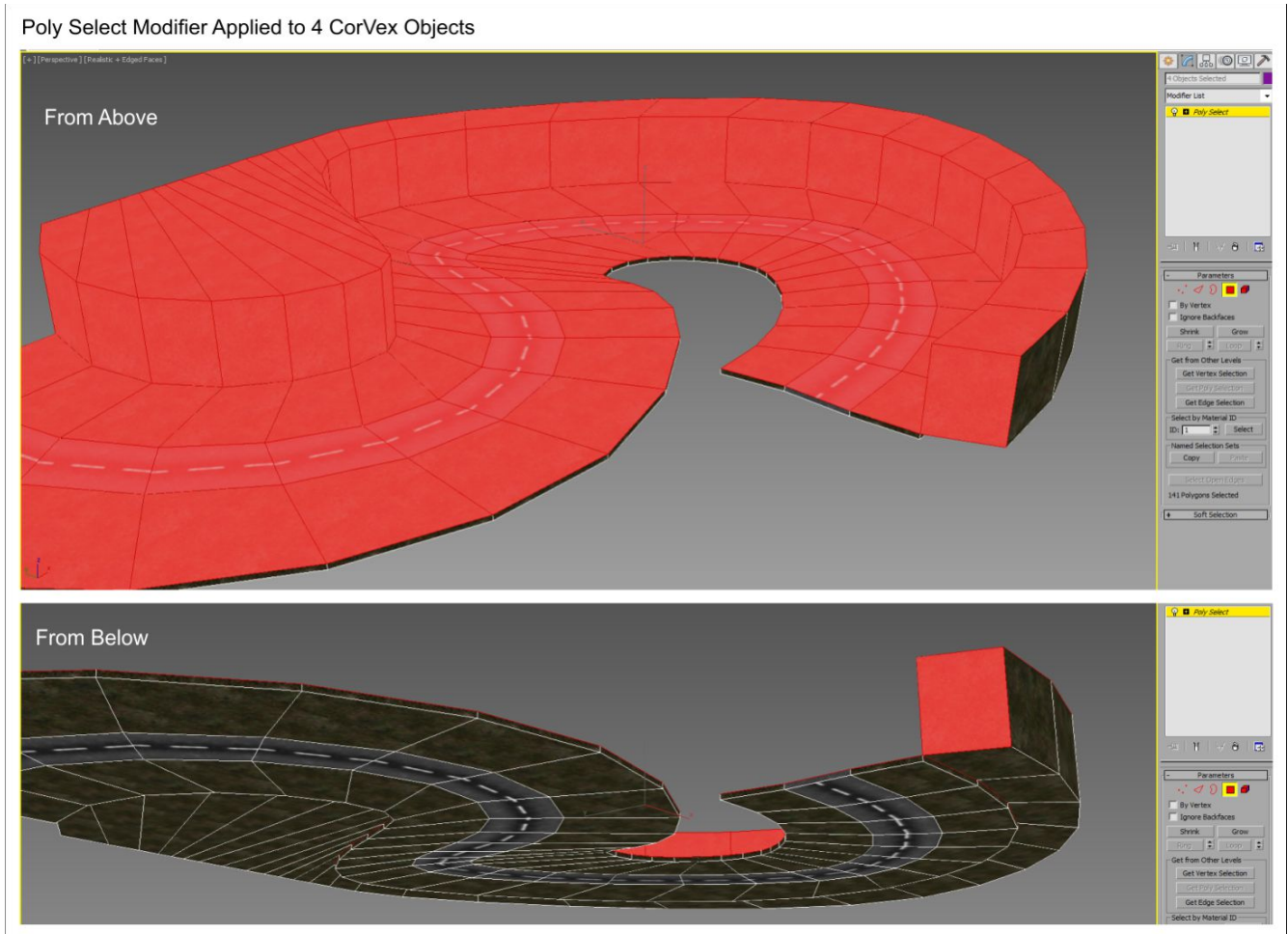
This CorVex object was designed with a displacement terrain in mind. Notice that all the polygons are quads (four-sided).

Before converting the terrain to displacements, you should manipulate the UVW such that it matches the scale and orientation you want to use on your displacements. You can do the same with the Vertex Alpha. The Material, UVW (channel 1) and the Vertex Alpha will transfer from the base object to the displacements.

Create Displacements from Selected Faces

1. Once you have the UVs, materials and vertex alpha to your liking, **select all of these objects** in the scene.
2. **Apply a Poly Select Modifier** to the objects.
3. Go to the **Polygon Sub-Object level**.
4. **Select all of the faces** you want to convert to displacements.

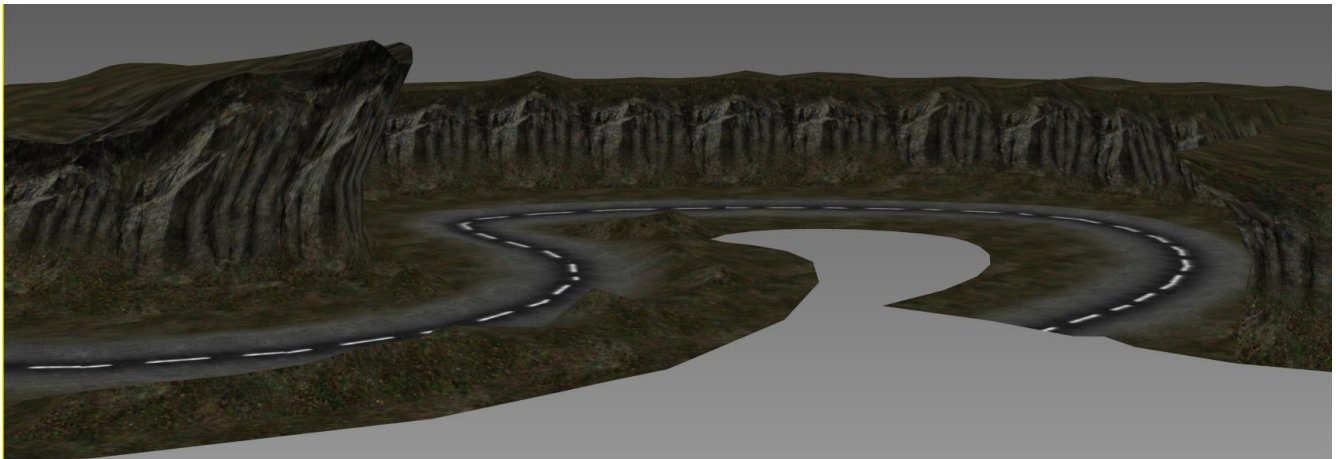
CorVex and Parametric Level Design



5. **Click** Wall Worm > Wall Worm Level Design > **Launch Displacement Floater**.
6. **Choose** the appropriate **Power** for your displacements.
7. **Click** the **Only Selected Quads** button.
8. **Wait** for the process to finish (this could take a few seconds to several minutes).
9. Once finished, **click** the **Sculpt** button in Anvil to create a Sculpt Mesh of the displacements.

In the image above, four CorVex objects have an instanced Poly Select modifier. Only the selected faces will be used to generate displacements. Notice that none of the faces facing down are selected (highlighted in red).

The following image shows the scene as it was after the displacements were created and a Sculpt mesh applied for easier sculpting. The entire scene, from start to finish, took only a few minutes.



There are many tools for working with your terrain. Editing the terrain is easiest when you convert sections of displacements to sculpt meshes (because the sculpt mesh can be edited as a single object that is easier to manage than multiple objects).

Some Tools helpful for Sculpt Mesh

- **Freeform Tools** in the Graphite Modeling Tools floater. These include brush-based tools for Push/Pull, Relax, Flatten, Noise, etc.

- **Displace Modifier.** Experiment with driving both the terrain and textures at the same time. See the chapter on Displacements for tips.

Using CorVex with PropLine

One convenient method for building your scene is to re-use splines in both CorVex objects and PropLine object. PropLine is a parametric prop distribution tool that, like CorVex, uses splines. By using the same spline for a CorVex and PropLine, you can easily and quickly lay-out your walls and related props. [Watch this video on using CorVex, PropLine and WWMT Proxies.](#)

Chapter 20 Scattering Props and Using Forest

Now we will discuss populating your scene with Forest. Forest is a commercial plugin from Itoosoft. Although there is a free version of Forest for evaluation, you will need the commercial version to follow this tutorial.

Forest allows you to populate the scene with models in a parametric and efficient manner. The concept of what Forest does is similar to how most Source games will scatter detail props on displacements using rules controlled by a detail.vbsp file. However, using Forest offers far more control.

Scattering Props with VBSP vs Using Forest

In Hammer the way scenes are populated with detail props (grass, small rocks, etc) is traditionally done using detail.vbsp. Although Wall Worm does have tools to both import VBSP data and to create actual VBSP files, the best method for scattering is to use Forest.

Flaws of using Detail.VBSP Files

Before going into the details of how to use Forest, let's first outline the drawbacks of detail props and VBSP.

- For sprites/cards, all billboards generated by detail.vbsp are limited to a single texture sheet defined in the VMF.
- All props scattered with detail.vbsp (both billboards and MDL files) must use Unlit-

Generic which produces poor lighting.

- Only scatters on displacements which have materials that specify specific detail types.
- No in-editor controls for scattering rules (either in Max or in Hammer).
- No longer works in some games (like Counter-Strike: Global Offensive).

Benefits of Using Forest

By comparison, Forest does not suffer from any of the limitations mentioned above. Above this, here are added benefits:

- Can use any entity type (prop_static, prop_dynamic, light, anything). This means you can create rules to scatter any type of entity.
- Does not limit to models using UnlitGeneric.
- Can scatter on any surface, not just displacements. This means you can scatter on brushes or other large model nodes.
- Reduces node-count in Max so improves viewport performance.
- Has extensive in-editor controls that allow you to specify scattering and distribution rules that you can preview in real-time. Controls include altitude limits, surface slope limits, density rules, clustering, randomization, transformation and many more.

Wall Worm and Forest

Wall Worm will export into a VMF any entity or MDL node that is used as a custom object in any active Forest node in the scene. Furthermore, there are some functions in WW to quickly set up a Forest using selected MDL nodes and selected displacements/sculpt meshes.

Prepare to Scatter Props

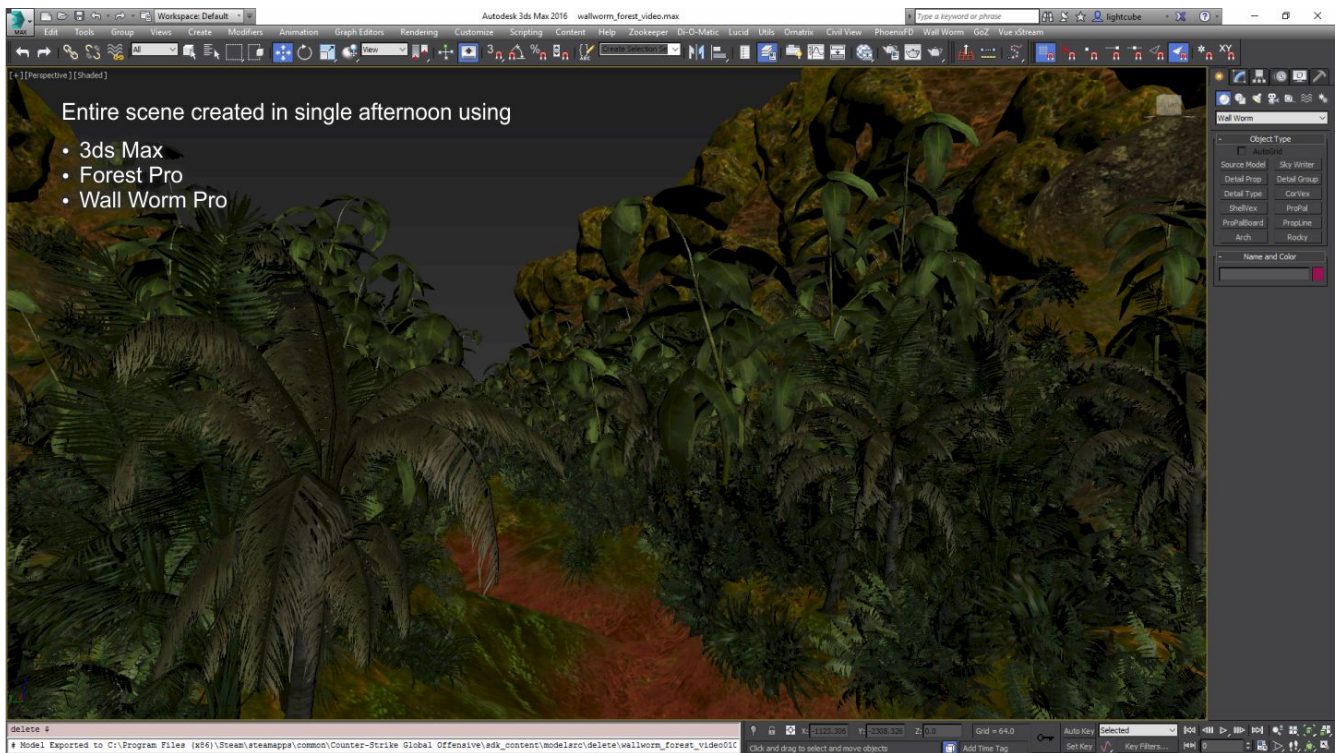
To start scattering props on your scene, you'll want to first bring in the MDL Nodes. If the scene already has props you want to use that are WallWormMDL nodes, then you are set. Otherwise bring in all the props with the MDL Loader that you might need. See the earlier chapter on Importing MDL nodes.

Once you have the props ready, you should select the sculpt mesh or displacements upon which you want to scatter the props. Then click Wall Worm > Wall Worm Level Design > **Launch Displacement Floater**. When this is open, select the displacements and the props and click the **Selected Displacements** button in the **Create Forest From** group.

This will create a Forest node that uses all the selected props and use the selected displacements as the surfaces for the forest.

Now simply utilize all the tools in Forest to properly scatter your props as you desire. The following image is a scene that was made from scratch (all rock models, displacements, displacement materials created in Max). The trees are WallWormMDL nodes loaded from Counter-Strike: Global Offensive and scattered with Forest. The entire scene (including custom models, landscape sculpting and alpha blending, took only a few hours from a blank slate). See this [video on the usage of Forest and clustering many of the props](#).

Scattering Props and Using Forest



Chapter 21 Anatomy of a Design Team

One of the benefits of using 3ds Max is that it has several ways to make teamwork and collaboration simple. This chapter will detail on getting your team on the same page for working together on a Source Engine project.

While there are no hard rules for how your team will work, this chapter will outline a setup that makes teamwork most seamless in Wall Worm. To utilize these steps, you and all team members will need to install [Wall Worm](#) 3.731+ and use 3ds Max 2015+ (where the preferred version is 3ds Max 2018.4+).

This document is broken into three sections: **Project Setup**, **Team Member Setup** and **Pipeline Overview**. Project Setup is for the project administrator. Although the information is useful for all members, only the Project administrator needs to take the steps in that section.

Project Setup

In this setup, we are going to use the standard 3ds Max Project Folder. By default, Max has a basic file structure for a project that separates and organizes the types of files used in a project. To get started with this, the administrator of the project should create a new Project Folder. To set this up in 3ds Max 2018, click File > Set Project Folder and browse to a new folder on your system. You should choose a folder path that will exist on all member's computers. In this example, I'm creating a folder at C:\3dsmax\vault\myproject. (You may want to use a folder name that describes your project like "de_mapcore" or "bm_xenland". For some a project might represent a single level where for others a project might represent an entire game.)

Once you let Max create this folder, Max will auto-generate a set of subfolders that are com-

mon to all Max projects. Among the folders will be these key folders:

- downloads
- export
- import
- sceneassets/images
- scenes

There are other folders Max will create, but above are those we will use in this tutorial. There are also some extra Source-specific folders we want, but they will be generated for us later with a tool in Wall Worm.

Setting Up Wall Worm with this Project

Now we need to set up Wall Worm to use this new project. Click Wall Worm > Wall Worm Settings and click the Import GameConfig File button at the far right side of the settings window. Browse to your game's gameconfig.txt file (for example: "C:\Program Files (x86)\Steam\steamapps\common\Counter-Strike Global Offensive\bin\Gameconfig.txt"). You will see that a new game preset appears in the preset list. Select that preset from the list and click "Load Selected Preset".

To keep this preset recognizable as for your specific team, you may want to change the preset name now by typing in a new preset label and then clicking the Save button below the preset list. For example, change the label from "Counter-Strike Global Offensive" to "CSGO Myteam1" or something else representative of your team or project. This way, if you ever re-import that CSGO gameconfig, you won't have two identically named presets. You should do this for any new project so that you can easily set Max/WW up by choosing the preset.

Paths

Now many of the paths will be set up to match the new project folders. However, we will still need to edit a few from their defaults. There is a utility in the next section on saving the preset to automate setting the paths to local as well as creating some extra paths.

Models

Open the Models tab in the settings and set these settings:

- **SMD Exporter:** Wall Worm Pro*
- **Normals:** Explicit*
- **Default Model Path:** *myproject*
- **Default Material Path:** *myproject*

Note that for the default model and material paths, you should choose a name that will be unique to your project/name. I generally use “wallworm.com”. You should choose names that are not likely to be the same among other users. Note also that you can set up multiple saved default paths. Whichever is selected will be the default output path for all new materials and models.

* Wall Worm Pro should be the SMD Exporter of choice if you have Wall Worm Pro installed. Otherwise set it to Wall Worm SMD Exporter. Also, if you have Wall Worm Pro or you are working with assets created in other DCC apps like Maya, set the Normals method to Explicit Normals. Note that the Explicit Normal option is much slower in the free version of Wall Worm compared to Wall Worm Pro.

Materials

In the Materials tab, set these settings:

- **Do Extra Lookups to Find Materials:** OFF
- **Do Extra Lookups to find Textures:** OFF
- **Assign Relative Path for imported textures:** ON

Miscellaneous

In the Miscellaneous tab, use these settings:

- **Debug Mode:** OFF
- **Displacement Topo Listen:** ON
- **Check for Home Project:** ON

Save and Store Preset

Now we must set some of the paths to local and save our settings/presets for other members to reuse. Fortunately, there is a utility to do this in the global settings.

Click the **Create Local Game** Info button at the right side of the Wall Worm settings. When you do this, you will be asked if you want to set the asset paths to local. Click Yes.

At this point, your game's gameinfo.txt is copied into a subfolder that matches your game's name in the project root. Also, a folder named sdk_content is added to the project root with a maps and modelsrc subfolder.

Alongside the new gameinfo.txt is a file called SourcePresets.ms. This file contains the all the settings for your project that you can share with the other members--assuring everyone is us-

ing the same settings.

Once you have finished setting up the settings to your needs, click the Save button in the global settings. This will save your preset.

Edit GameInfo.txt

Now also edit the *original* GameInfo.txt file for your game to include a line to our new extra game path. This file will open automatically at the end of the last step--and the path you need to add to it is also copied to your system clipboard. So all you need to do is paste the new line to the end of the SearchPaths block. In this example, the Search Paths are edited to look like this for the actual CS:GO gameinfo.txt file:

```
SearchPaths
{
    Game      |gameinfo_path|.
    Game      csgo
    Game      "C:/3dsmax/vault/myproject/csgo"
}
```

By adding the extra search path to our original gameinfo.txt file, it means that we can keep all assets from our project completely independent of the actual game files and your game can still find the assets. All of the models, materials, textures and levels we create will be stored in the project folders and not contaminate the game folders. **Important:** Make sure your extra path is at the bottom, below the official paths.

Create a MAXSTART file

3ds Max will auto-load any file called maxstart.max in the current project folder. We will create

one because we can enforce all users in the project to use some of the same settings.

1. In 3ds Max click File > New
2. Set the system to use Generic Units where 1 Unit = 1 Inch (Customize > Units Setup)
3. Click Wall Worm > Point Entities and browse to info_player_start in the entity list
4. Click Place Entity and pick a point in the viewport
5. Right-click to stop adding entities
6. Select the entity
7. Click W (Move)
8. Right-click the X/Y/Z spinners to move this object to world origin
9. So this entity does not always export, find the Wall Worm Connection rollout in the modify tab and check the Exclude From VMF checkbox.
10. Save this file to scenes/maxstart.max

Vault Setup

Autodesk Vault is a data management software that helps organize and manage projects. While you do not need to use Vault to work on a team, it has several advantages over other options (such as SVN, Git, DropBox, etc). Among the advantages is the fact that 3ds Max itself has tight integration with it. If you have access to 3ds Max, you also have access to Vault Basic (which is what is used in this document).

The project administrator will need to install the Vault Server on a computer. Team members do not need to install the Vault Server. Team members should, however, install the Vault Client.

Once installed, you will need to create a user account for each member of your team. You will also need to create a set of permission groups and a Vault Project for your team. Installing

Vault itself is not covered in this document.

Create Vault

A Vault can be thought of as a virtual repository of all the assets of a project. In a way it's just like a file system folder. However, it contains version control and asset linking. It also allows users to check in/out files from remote locations.

To get started, we need to create a Vault. This must be done on the server from which Vault is located.

1. Open the Vault Server
2. Launch the Autodesk Data Management Server Console
3. Expand the Server name to show a Vaults folder
4. Right-Click the Vaults folder and click Create
5. Name your Vault (preferably to something relating to your project)

Now that the Vault exists on the Vault Server, you can finish all remaining tasks remotely using the Vault Basic Client.

Create Users

For your team to have access to the Vault, each member needs their own user account in the Vault.

1. Open the Vault Basic Client
2. Click Tools > Administration > Global Settings
3. Click the Users Button
4. Add one user for each member of your team
5. Assign the user to have access to the Vault

6. Alternatively Create User Groups and assign them to your team appropriately

Set Vault Settings

Each projects you work on should have its own Vault. Although not required, you should consider forcing all users to keep the Vault in the same location on their computers. Doing so will help avoid problems that can happen if the system is not set up this way. In this case, we will force each user to keep the vault at: C:\3dsmax\vault\myproject

1. Open the Vault Basic Client
2. Click Tools > Administration > Vault Settings
3. Click the Define button in the Working Folder Options
4. Turn on the Enforce consistent working folder for all clients and set it to
C:\3dsmax\vault\myproject
5. Click OK

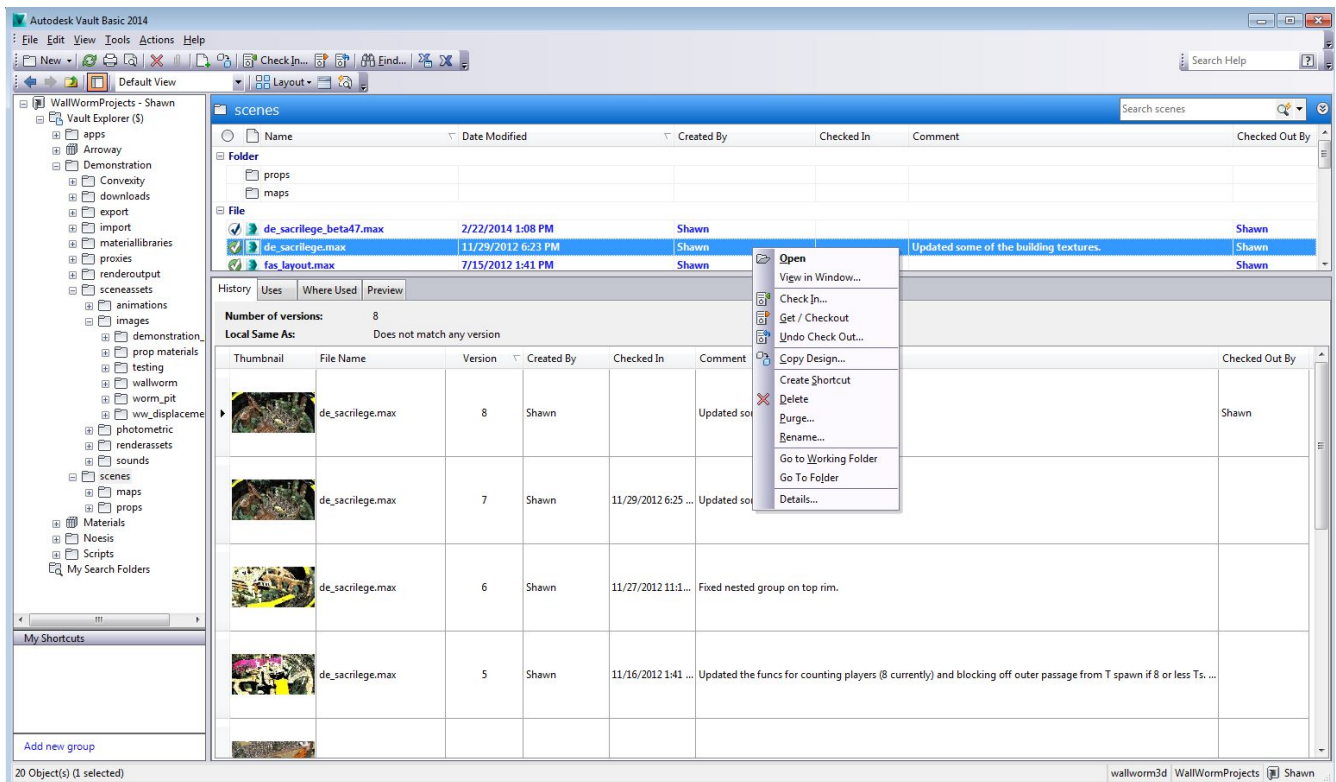
Add Folders and Files to the Vault

Now we need to tell Vault what files and folders belong to the Vault. Vault will only track those folders and files you added to it explicitly.

The easiest way to add multiple files to your Vault is to browse to the Vault folder in Windows Explorer, select all the folders, and drag them onto the Vault Explorer icon in Vault Basic. You can also right-click the Vault Explorer icon and choose Add Files.

When you add files to Vault it will prompt you to type a message as well as give you the option to keep the file checked out. You should always make a habit of typing a message for your checkin/checkout actions. As for checking out the file--you should refrain from checking out files that you are not editing.

Anatomy of a Design Team



At this point, add all of the folders and files from the project folder into the Vault. Once you've done this, the Vault is now set up for others to use. Just remember that in order for team members to have access to each other's files, the members must always remember to check in their files to the Vault.

Other considerations are that your Vault Server must be running from a computer that has Internet access if collaborating remotely. You may also need to setup port-forwarding on your router to point ports 80 or 443 to your Vault server. See the Vault documentation or your router documentation for more information. Preferably your Vault server is located at an IP address that does not change to the outside world--or else you will need to keep your team apprised of IP address changes.

Team Member Setup

Now that the Vault Server is ready, we need to get each team member on the Vault. There are two ways to do this. One is simply through 3ds Max. The other is through the Vault Client. It is probably a good idea to get each member set up with both.

Vault Client Login

The first you open the Vault Client, you will need to log in with your username and password.

1. Launch Vault Client
2. Click File > Log In
3. Choose Authentication Vault Client
4. Enter your username
5. Enter your Password
6. In the Server, enter the IP Address or computer name
7. For Vault, click the button at right to browse for Vaults on your server and choose the appropriate one
8. Turn on the Automatically log in next session option.
9. Click OK

Now you are able to browse the files in your Vault. From here you can download file (Get), Checkout Files (Get and lock others from editing), Add files. You can also get previous versions of files.

Download (Get) Files

At this point you should select folders and files you want to bring to your computer. It is not

necessary for you to get all files unless you need them. For example, you may not need to open a Substance Paint file that another user loaded unless you actually need to open/edit that file.

For now, we are going to download the entire project because it is very small still.

1. Open the Vault Client
2. Click the Vault Explorer folder in the left pane
3. In the Vault Explorer window in the right, Left Click the top folder
4. Now Hold Down the SHIFT button and left click the bottom file (myproject.mxp)
5. Now that all items are selected, right click and choose Get...
6. Since you are not currently planning on editing these files, do not click the Checkout icon (Checkbox).
7. Click OK and wait for the files to download

Vault in 3ds Max

You can also log in to Vault directly in 3ds Max. To do this you should launch the Asset Tracking Dialog (SHIFT+T) or click File > Reference > Asset Tracking Toggle.

To integrate with Vault, click the Options menu and uncheck the Disable Asset Tracking and turn on the Auto Login option.

Now click the Server menu and choose Login and log in with your credentials.

Important Note: There are two important bugs with the Vault login in Max. First, the option to remember your password doesn't work in some versions of Max. The second is that the password field will become plain-text when you click the button to login. Because of this you should not login to Vault via 3ds Max during a screen sharing session.

Asset Tracking

At this point, you should familiarize yourself with how the Asset Tracking tools in Max work. The best resource is to read about these functions in the [3ds Max documentation on the Asset Tracking Dialog](#).

Installing the Project Presets

Now that the files are all downloaded, we need to set Wall Worm up for this Project.

1. Click Wall Worm > Wall Worm Settings
2. Click the Import Presets File button at the right side
3. Browse for C:/3dsmax/vault/myproject/csgo/SourcePresets.ms
4. You will see one or more new presets in the setting preset list
5. Choose the proper preset (which will have an asterisk to denote it was imported)
6. Once selected, click the Load Selected Preset Button

Edit GameInfo.txt

Now you need to browse to the actual game you are working on and edit the GameInfo.txt file. This will be in the game folder of your actual game. In this example, it is the gameinfo.txt file for Counter-Strike Global Offensive.

Edit the file's search path block to include the bold section below:

Now also edit the *original* GameInfo.txt file that we originally copied to include a line to our new extra game path. In this example, the Search Paths are edited to look like this for the ac-

tual CS:GO gameinfo.txt file:

```
SearchPaths
{
    Game      |gameinfo_path|.
    Game      csgo
    Game      "C:/3dsmax/vault/myproject/csgo"
}
```

This will allow you to access any assets that are compiled into the Vault project files without contaminating the actual game files.

Now you should be ready to use the new Vault project folder with your team. Before getting started with work, it's a good idea to double-check that the paths are all as expected. In your Game paths, the modelsrc, materialsrc, mapsrc and Game Info paths should go to your project folder. The Game EXE and Bin Dir should still go to the actual game paths.

Pipeline Overview

Now that your Vault Server is set up and the team is set up, there are a few things to understand.

File and Project Structure

Every asset for your project should be contained within the project folder. This includes all raw assets (PSDs, TGAs, MAX Files, etc) as well as exported and compiled assets (VMF, BSP, QC, SMD, MDL, VTF, VMT, etc) . Nothing should be outside this project.

Compiled assets should end up in their respective folders inside the game folder. In our example with a CSGO project the finished game files will all go into:

C:/3dsmax/vault/myproject/csgo/

- maps (BSP)
- materials (VMT, VTF)
- models (MDL, VTX, PHY)

Raw Images like PSD and TGA should generally reside in the C:/3dsmax/vault/myproject/sceneassets/images folder (and subfolders).

Note that keeping any reusable asset in the sceneassets folder is always useful to help find the images especially if they are used in multiple scenes. However, you may consider making a policy that images that are specific to individual models (baked to a unique UV, for example) might reside within the same folder as the Max scene. This is a scenario your team should consider. Different teams will likely choose their own policy. Because Max defaults to using the sceneassets/images folder for the Render to Texture dialog, choosing a scene folder can sometimes require extra diligence.

All Max scenes should be contained in the Scenes folder. You should keep levels inside the scenes/maps subfolder and all props in the scenes/props subfolder(s).

For any/all assets that might be used outside the current project (reusable material libraries

Check In/Out Files

In general, you should make a habit of using the Vault Checkin/Checkout functions. However, there are some things to keep in mind that should influence your checkin/checkout procedure.

Checkout Vs Get

There are two methods of fetching assets from the Vault Server: Get and Checkout. Get will download the file but will not let you edit it. This means that you can open it and reference it from another file. But if you are not the person who checked it out, you cannot edit it. Only the person who has the file checked out can edit. If the file is currently checked out, you can only Get it.

You may think it's always best to Check Out a file. This is not the case. Generally you should only check out files if/when you need to edit it. This ensures that the file is available to those who need to edit it. The fact that a file is locked when someone else checks it out ensures that two (or more) people do not do incompatible edits to the same file at the same time. If you absolutely need to edit a file while it's checked out, you can get it and simply rename the file and edit the new file.

Checking In Files

The check-in process is what sends a snapshot of your file to the server. That creates a version of the file that can always be reloaded at that version state. This means you can retrieve older versions of the file if a drastic change needs to be undone.

You may think it's a good idea to check in a file every time you actually save your scene. However, that may not be the best strategy. Each time you check in a file, there is a complete version of that file saved on the Vault server. For a file that is several hundred megabytes, this could quickly fill up your server's hard disk space.

Each team will need to make its own policy, but here are some ideas for appropriate checkin strategies:

- After any major change to a file that represents several hours of work

- At regular intervals (every Friday evening, for example)
- When the artist working on a file is switching tasks*
- When the artist working on a file will be unavailable for a period of time*

* In cases where the current artist will be unavailable for a time, the artist should also make sure that the file is not immediately checked back out during the commit so that others can edit as needed.

Of course there are times that a remote teammate may become inexplicably unreachable. In that case, the administrator can forcibly unlock files. Documentation is available in the Vault docs.

Model and Scene Referencing

There are many ways to reference model assets in 3ds Max from one scene to the next. Effectively collaborating in Max entails that the team understands what the options are and which is used most effectively in what scenario. Below are some of these along with use-cases.

The types of ways models and scenes can be referenced:

- XRef Scene
- XRef Object
- Container
- MDL (Wall Worm Source Model)
- Linked FBX Files
- Imported Asset (OBJ, FBX, SMD, etc)
- Merged Asset (from another Max file)

Which of these you use is situational. As much as possible you should use the first three options, especially if you are creating entirely new assets. The MDL and Linked FBX are the next preferred ways to reference models. That last two options should be avoided when possible--and when used they should be one-off functions that turn into native assets to be referenced with earlier options.

Referencing Understood

So what is this referencing anyway? The best way to understand it from a traditional Hammer-user's perspective is the fact that your props are all MDL files that Hammer is just using over and over but never edits. Some of the referencing methods are very similar, including XRef Object, Linked FBX and MDL. In fact, the Wall Worm MDL loader works with MDL files very similar to how they work in Hammer.

The bottom line is that the props are encapsulated and separate from the main scene(s). This means that artists can work on props independently of the level designer(s); each time a modeler updates a prop scene, the level designers will see the updates in their level; each time a texture artist updates their Substance, all scenes using the substance will update accordingly.

Which model reference to use depends on the use case.

XRef Scene

XRef Scenes can be thought of different views of the same scene. XRef Scenes should be used when you need to work on the same environment but are partitioning different sections or tasks to different users. While useful for the artists, they play the most important role for level designers, environment artists and animators.

Files that reference each other as XRef Scenes can view each other but do not interact in the Max viewport. A level can be broken into tasks. For example, one Max scene accounts for all

the brush work, another accounts for displacements and artwork, and a third contains just entity I/O. With this setup, the scene can be worked on in tandem between three (or more) artists and level designers and never has any bottleneck of one member's work being held off until another member's task is complete.

Animators may also take advantage of XRef scenes because they can load the XRef scene to get the environment around which their characters/animated models need to move. Being an XRef scene means that the objects of the external scene aren't actually cluttering the current file's data.

To load another Max scene as an XRef Scene into the current file, Click File > Reference > XRef Scene and browse for the scene.

XRef Scene Hints:

- Always turn on the Overlay Option in the XRef Scene load dialog. Failure to do this will cause problems if the loaded file also references the current file.

XRef Object

The XRef Object is a way to bring in a reference of a specific object from another Max scene. For example, I may have created a set of trees in a Max scene under scenes/props/trees/trees1.max . The level designer or environment artist can now create XRef Object nodes that reference each of those trees and scatter them about in the current scene. Updates to the models in the trees1.max file will cascade to all XRef Object nodes in all scenes.

Choosing XRef Object is an excellent candidate for scenes where you intend to cluster the props (see the [Convert Scene to Model documentation](#)). If you do not intend to cluster the

props, then you must take measures to tie the props to entities manually.

You can create an XRef Object by clicking File > Reference > XRef Object.

Container

A container is similar to both the XRef Scene and an XRef Object. Like an XRef Scene it can bring in many objects at once; unlike the XRef Scene but like an XRef Object, it can be moved and copied around a scene.

Containers should be used as a replacement for a func_instance. In fact, when you create a func_instance entity with the Point Entity floater in Wall Worm, it creates an empty Container. You should generally use Wall Worm Point Entity func_instance to load containers.

Unlike XRefs, Containers can be edited locally even though their data is stored in a separate file. Normally you should create the Container and its content in an external file before loading it into the current master scene.

Source Model (WallWormMDL)

Max can load MDL nodes into the scene just like Hammer can (in 3ds Max 2015+) when Wall Worm is installed. It can find and load MDL data and their materials/models provided that the MDL can be found in the paths designated by the gameinfo.txt file that is contained in the Game Info directory. (This is why we created a gameinfo.txt with a search path pointing to this project's game folder as well as the actual game folder--so Wall Worm can find the native game assets as well as all of those we compile to this project.)

This type of model reference is the best solution for placing props that have already been compiled and are not going to be clustered. You can still cluster these kinds of props, but it's

best to use native Max geometry when possible for clustering.

These types of objects will automatically export as single props.

There are several methods of creating Source Model nodes:

- Command Panel > Create Tab > Geometry [category: Wall Worm] > [Source Model](#)
- Wall Worm > Wall Worm Importers > [Mass Model Fetch](#)
- Wall Worm > Wall Worm Importers > Create Prop Zoo from VMF
- Wall Worm > Wall Worm Importers > Import MDL

Linked FBX Files

The FBX format is a standard model format owned by Autodesk that many applications can open. This file format is most useful in the Wall Worm project pipeline if some of the artists are working in other DCC apps other than 3ds Max (such as Maya). In that case, the Maya artist can export updates to an FBX file that will natively update inside 3ds Max.

See the [documentation on the File Link Manager](#) for more information.

A Linked FBX file is most appropriate for cases where a modeler on your team is not familiar with 3ds Max and the model is going to be used in a WWMT Helper (See Exporting Assets below) or in XRef Objects.

Importing and Merging

The import/merge options are generally starting points. Importing a non-native format is usually destructive. As such, it should always be viewed as a one-way-street just to get assets into Max. Once in Max, you should re-use that asset with strategies outlined above.

Exporting Assets

Almost all assets must be converted from their original format into a game-friendly asset. For the most part, you should always use the native exporters in Wall Worm to generate your assets. Furthermore, you should utilize the Wall Worm Model Tools method of exporting models rather than a traditional model export method (which entails the inefficient method of exporting SMDs manually, writing QCs from scratch and compiling manually). The primary reasons you should do this are explained below.

- **Consistency.** Wall Worm's exporters have a very consistent format for how modelsrc and final assets are saved. This means that if you can find a MDL file, you can immediately find it's QC/SMD files; if you can find the QC, you can find the Max scene that created it.
- **Multi-User Overlap.** If an asset is exported outside of a Wall Worm flow, other Wall Worm functions may not know about that change. This means that if another user edits a file and re-exports, the changes from the first user are lost because the data wasn't kept in the Wall Worm data structures. For example, if a Normal map in 3ds Max needs the VTF compression level changed, it should be done inside the 3ds Max material editor so that the Max scene will know the correct compression. Another example is that if you manually add a new skin to a QC instead of in the Max scene, then another user edits the model and re-exports, the skin edits are lost--whereas if you add the skins to the WWMT helper, the skins will propagate with all future exports.
- **Expediency.** Once you understand how the Wall Worm system works, you will find that all export functions work much faster from inside 3ds Max rather than outside.

This does not mean that 100% of the work can be done directly inside Max. You will still find occasion to need to edit a QC or VMT. However, for QC files, you should do most edits in the QCI file that WWMT automatically generates for each prop--as this QCI file is never overwrit-

ten when Wall Worm exports a model. For VMT files, you may find some settings that must be written manually. The number of times you should need to edit those files manually should be very minimal.

Collaboration

There is no absolute set of rules for how team members should collaborate. This section is a guideline based on years of experience working with 3ds Max, Source and Wall Worm. There is a good chance that these guidelines do not align with traditional Source Engine pipelines.

User Roles

Depending on the size of your team, you may fit into different categories. However, there are a few key roles that are important and detailed below. In some teams, members may wear multiple hats, and in some the members might be specialized.

Project Manager

The project manager is the one who sets the goals, deadlines and structure of the project. The project manager should have administrative rights to the Vault used in this project--being able to add/edit files and enforce file policies (such as the policy to force all members to use the same file paths for the project on their systems).

Team Leaders

Team leaders are members who oversee major realms of the project. If your team is large enough to require team leaders, they might oversee different departments such as Level Design, Art, Animation, Programming, etc. Not all teams will need team leaders.

Level Designers

The level designer creates the backbone of all levels. Traditionally, the level designer was the “Hammer User.” In a fully Wall Worm pipeline, level designers do not use Hammer--but should be either experienced in Hammer or fully understand the principles of Source Engine BSP requirements. The level designer is responsible for creating Levels, Brushes, Entities and Entity interactions.

Artists

Artists are those who create art assets. This can include models, textures, materials and other assets for populating your levels. While the traditional pipeline might put this entirely on the level designers, in Wall Worm the artists often also work on the displacements (landscaping). As such, the artist will often need to understand how Source Engine displacements work, how Wall Worm’s Sculpt Meshes work, and how the Vertex Paint modifier is used for texture blending on displacements.

Animators

In some teams, the animator is a separate class of artist that animates the props.

Audio Technicians

Audio technicians create the sounds. Audio work is an area of Source Engine development that sits outside much of the Wall Worm pipeline. There are some audio-specific tools in Wall Worm related to generating Soundscape files, but actually creating the audio is outside the scope of 3ds Max and Wall Worm.

Programmers

Programmers may be the most overworked and underappreciated class of user in a Source Engine project. For most projects, there won't be a programmer (as most projects will be collaborative mapping projects where programming isn't part of the scope of the project). However, for mod teams and full game studios, the programmer is an absolutely essential role in a project.

Like audio technicians, programmers are not much considered in the Wall Worm pipeline. However, programmers can get involved in the art pipeline or help build custom tools for solving design tasks inside 3ds Max.

Role Assignments

How you assign the roles is a team-by-team or project-by-project consideration. In all likelihood, you will have members that use multiple roles. The main essential considerations that are necessary for every project are technical considerations:

- Level Designers must understand basic BSP principles and how to handle brushes inside 3ds Max.
- Level Designers must understand how to compile levels from 3ds Max.
- Modelers must understand how the Wall Worm Model Tools (WWMT) helpers work.
- Modelers must understand the differences of Smoothing Groups and Explicit Normals--and how to address each in the export process.
- Environment artists must understand how to work with displacements in Source and how the Wall Worm Sculpt meshes work.
- Environment artists must understand the difference between a prop that will export individually into the game and how to cluster props.
- Animators must understand the difference between skeletal animations and vertex animations--and how to handle them with WWMT.

- Animators must learn about linking objects, hierarchies and skinning.
- Texture artists need to know how to bring in image assets into 3ds Max (TGAs, Substance files, etc).

Level Setups

Setting up a level should start with the project manager or level designer. A level can be broken into different areas/departments. For expediency, you should make one file for each separate department that might work on a level at the same time, as well as each separate section/user that might work on that department. These sections should be individual 3ds Max files where each file has an XRef Scene record of all the others in the level.

Example Level

In this example, we are making a level called `de_awesomemap` for the game Counter-Strike Global Offensive. Assuming we have two level designers, two environment artists and two prop artists working on the level, we can create these Max scenes:

- `scenes/maps/de_awesomemap.max*`
- `scenes/maps/de_awesomemap/brushes.max*`
- `scenes/maps/de_awesomemap/entities.max*`
- `scenes/maps/de_awesomemap/environment_plants.max*`
- `scenes/maps/de_awesomemap/environment_displacements.max*`
- `scenes/models/plants1.max`
- `scenes/models/plants2.max`
- `scenes/models/rocks1.max`
- `scenes/models/maps/de_awesomemap/map_models1.max`

In this example, all items listed with an asterisk should reference all of the other asterisked

scenes as XRef Scene Records. Those items in the scenes/models subfolders should only be referenced in the main scene files as XRef Objects. The main model Max files contain generic objects that might be used in multiple levels, whereas the files in scenes/models/maps/ should contain props that are only relevant to specific levels.

In this example (which is just one of a myriad of structures you could so), we could have users working on all of these scenes in tandem and independently:

- Level Designer 1 working on scenes/maps/de_awesome/brushes.max to create the brush work
- Level Designer 2 working on scenes/maps/de_awesome/entities.max to create entities and their Inputs/Outputs
- Environment Artist 1 working on scenes/maps/de_awesome/environment_plants.max to scatter the plant props referenced in the scenes/models/plants files. This user might be using tools like Object Paint, GrowFX or Itoo's Forest to distribute plants.
- Environment Artist 2 working on scenes/maps/de_awesome/environment_displacements.max to sculpt displacements and alpha blending or 4way blending.
- Prop Artist 1 working on plants1.max and plants2.max.
- Prop Artist 2 working on rocks1.max.

In this case, the level designers and environment artists can see all the work others are doing inside their viewports but are not interfering with each others' work. Because we are working with native Max objects, there is no need to compile the models or textures during this design stage.

This is just one example setup. You may also break up the sections by level designer instead of type. For example, instead of having brushes.max and entities.max you may have Area1.max and Area2.max where one level designer is assigned to built brushes and entities (and perhaps displacements and environment art) in each specific area. Another setup would



include less core scenes because there are less artists working on the team. For a single-user project, you might even do everything in a single scene--though it's still probably a good idea to keep model scenes separate from the main level especially for props that are reusable.

Conclusion

The main principle of working on a team in 3ds Max with Wall Worm is to understand that your project should be “multi-threaded”. In other words, you should break up files and assignments in such a way that no file or artist is a bottleneck for others. No one should have to wait for files to be handed off for the next person to do their job. Although it takes a little bit of initial planning to create the most efficient setup, the reward of a team working in 3ds Max and Wall Worm is a level of efficiency that is not possible in legacy Source Engine pipelines.

Chapter 22 Useful Tools for the Source Engine

The great thing about 3ds Max is that aside from being so robust by itself, it is extensible. Developers can build scripts and plugins that add to the base capabilities. Below are some of the tools that I've found to be of great value.

Tool	Description	General Use	Cost
 CorVex	Block and Wall Primitive with native snap verts-to-grid, UV controls.	Level Design	Inexpensive
 ShellVex	Blocks and Brushes from Source Polygons	Level Design	Inexpensive
 PropLine	Distribute Prop entities parametrically with Splines. Similar to RailClone and the Max Spacing Tool but specifically designed with Source in mind.	Level Design	Inexpensive

Useful Tools for the Source Engine

Arch Primitive	Robust, brush-friendly arch primitive in Max.	Level Design	Inexpensive
 Carver	Carve, Make Hollow and Cleanup Functions.	Level Design	Inexpensive
Brushify Modifier	Modifier that moves object vertices to the grid and breaks non-planar polygons.	Level Design	Inexpensive (requires 3ds Max 2016 SP2+)
SnapVertsToGrid	Modifier to move object vertices to the Grid. <i>Use Brushify instead if you have Max 2016+.</i>	Level Design	Inexpensive
Miauu's Work Plane	Interactive grid tools.	Level Design	Inexpensive
PolygonMap	Planar UVW Mapping Tools	Level Design/Modeling	Inexpensive
Bitmap2Material	Easy PBR Texturing	Level Design/Modeling	Mild Expense
Zookeeper	Node-editor and nest-	Scene Management	Mild Expense

Useful Tools for the Source Engine

	ed layer tools.		
Rayfire	Interactively destruct geometry	Modeling/Environment Art	Mild Expense
VFB+	Improved Virtual Frame Buffer	Rendering	Free
Power Nurbs	Nurbs modeling tools	Modeling	Expensive
GrowFX	Plant modeling tools	Modeling/Environment Art	Mild Expense
Forest Pack	Plant scattering tools	Environment Art	Free and Commercial
Ghost Town	Parametric Buildings	Environment Art	Commercial

General Source Engine Tools

There are also a few non-3ds Max tools you'll want to consider adding to your tool kit when working with Source. They include:

[VTF Edit](#): Tool for converting VTF files to TGA files, or vice versa. Use this tool to convert your VTF library into TGA files so that you can create a Material Library inside 3ds Max (necessary unless you have Wall Worm Pro).

[GCFscape](#): Tool to browse and unpack VPK and GCF files (which are like zip files but used by the Source Engine). Use this to get VTF and MDL files from your mods to convert with VTF Edit or Crowbar. Generally not needed with Max 2015+.

[Crowbar](#): Model Decompiler and Compiler tool. Useful if you want to convert existing model assets into SMD files that Wall Worm can import.

More 3ds Max Resources

There are many valuable resources for learning to use 3ds Max. Google and Youtube are full of answers. But there are also some other places you may want to go for tools, advice and tips.

3ds Max Tools and Developers

- [ScriptSpot](#)
- [Max Plugins](#)
- [Boomer Labs](#)
- [JokerMartini](#)
- [Marius Silaghi](#)
- [Grant Adams](#)

Source Engine Resources

By now, it's very unlikely that you have not visited Valve's resources available for developing the Source Engine. But just in case:

- [Valve Developer Community](#)
- [Source SDK Forums](#)

More Game-related forums

And finally, here are a few of the game developer forums I've used over the years with many knowledgeable people:

- [Polycount](#)
- [Mapcore](#)
- [Interlopers](#)

Also, you will find many levels for source on [GameBanana](#), where several Wall Worm users are actively sharing their models or levels.

Chapter 23 Conclusion

If you've gotten this far you are serious about expanding your design arsenal. It probably means you already have what it takes to learn 3ds Max and Wall Worm—as the majority of what you need is bound up in creativity, patience, willingness to learn and practice. Now it's all up to you to get familiar with 3ds Max (if that is still new to you) and Wall Worm.

If you have learned anything useful, please share your knowledge with others. Also share the link to this book with anyone who wants to learn how to use 3ds Max and Wall Worm.

When you've gotten something you're proud of, please share screen shots in the Wall Worm forums, as it is great to share with other creative people the fruits of your efforts.

Try not to fall into the trap I personally fell into during the years developing Wall Worm—fear of inadequacy. Before I started Wall Worm, I did not spend much time in forums and rarely noticed the criticisms and talk that goes into the design process. But after I started Wall Worm and spent more time in game development forums, I realized that there are many great level designers out there and talented modelers creating artwork that is simply amazing.

On one hand these amazing artists inspired me. But on the other hand, they often intimidated me from sharing projects publicly as freely as I did a decade ago. Back then, a level didn't need as much detail to fit in because the gaming world was used to levels with less detail. But now players and designers have set the bar pretty high for environments and texturing.

While writing this book I realized that something was missing from my experience designing levels while working on Wall Worm. That missing element was simply the carefree feeling of having fun making a level. I think that this may be one of the most important things to remember when exploring your new worlds in Max. As my time building the core features for Wall

Conclusion

Worm comes to an end, I'm being reminded of the reason I built Wall Worm: having a great time building levels for the games I love.

Learn, Play and Explore. The rest is really a distraction.

Please support Wall Worm by getting your own copy of [Wall Worm Pro](#), [CorVex](#), [ShellVex](#), [PropLine](#), [Brushify](#), [Carver](#) and other Wall Worm plugins!

And share off your levels with a chance for great prizes in the [Angry Teapot Awards](#)!