Introduction

This is a working draft of the specifications for a new map format for Wolfenstein 3D as well as other games that run on the Wolfenstein 3D engine. It is loosely based off of the UDMF specification standardized by the Doom community.[1] The syntax for the TEXTMAP portion of this document should be assumed to be the same as that defined by the UDMF standard.

In order to simply this document here are some definitions:

- **Tile** - Maps textures to a numerical index.
- **Sector** - A Sector defines the floor and ceiling texture to use.
- **Zone** - Indicates areas where sound passes. Zones are never connected (even if they touch) unless a door is opened between them.
- **Angle** - An angle is indicated in degrees where 0 is east rotating counter clockwise.

Packaging

A map must be packaged within a WAD file. An empty marker lump is used to indicate the name of the map and will be followed by a TEXTMAP lump. Any additional data pertaining to the map, such as scripts, shall be placed following the required data, but before the ENDMAP marker. Lumps outside of these markers is not considered a part of the map.

TEXTMAP Lump

Global Scope

A TEXTMAP lump must begin with a name space declaration. The “Wolf3D” name space is reserved for the base spec described here. Other name spaces may be defined.

```
namespace = “Wolf3D”;
tilesize = <integer>; // Usually 64
name = <string>;
width = <integer>;    // Map size, usually 64
height = <integer>;   // Usually 64
```

Tiles

In order to simpify map design, a palette of tiles must be defined. Tiles will automatically

[1] http://www.doomworld.com/eternity/engine/stuff/udmf11.txt

be assigned an index based in the order in which they are defined (starting at 0). Textures can either be the name of a texture or a hexadecimal color prefixed with a hash character (#). Using the texture "-" indicates that there is no texture for that side.

```
tile
{
	texturenorth = <string>;
	texturesouth = <string>;
	texturewest = <string>;
	textureeast = <string>;

	offsetvertical = <bool>; // For doors offsets ½ unit.
	offsethorizontal = <bool>; // Default: false (for both)

	blockingnorth = <bool>; // Default: true (for all)
	blockingsouth = <bool>;
	blockingwest = <bool>;
	blockingeast = <bool>;
}
```

Sectors

A palette of sectors may also be defined.

```
sector
{
	texturefloor = <string>;
	textureceiling = <string>;
}
```

Zones

Place holder block to assign properties to sound zones.

```
zone
{
}
```

Planes

Unlike the original format, a plane is regulated to a collection of cubes and sectors. Planes are stacked starting with the the bottom-most level.

```
plane
{
      depth = <integer>;    // The height of this plane,
                            // usually 64.
}
```

Things

```
thing
{
      x = <float>;
      y = <float>;
      z = <float>;    // 0 is the bottom of the first plane.

      angle = <integer>;    // Angle of thing in degrees
                            // Default: 0 (East)
      type = <integer>;     // Editor number for the thing
                            // Defined elsewhere.
      ambush = <bool>;      // Previously indicated by sector
                            // Default: false
      patrol = <bool>;      // Default: false
      skill1 = <bool>;
      skill2 = <bool>;
      skill3 = <bool>;
      skill4 = <bool>;
}
```

Thing Types

Types only apply to Wolf3D/Spear of Destiny.

1. Player 1
2. *Reserved* *// Future use for multiplayer starts?*
3. *Reserved*
4. *Reserved*
5. *Reserved*
6. *Reserved*
7. *Reserved*
8. *Reserved*
9. *Reserved* *// DM Start?*
10. Patrol Point
11. Guard

12. Officer
13. SS Guard
14. Dog
15. Mutant
16. Hans
17. Schabbs
18. Fake Hitler
19. Hitler
20. Gretel
21. Gift
22. Fatface
23. Trans
24. Uber Mutant
25. Willhelm
26. Deathknight
27. Angel of Death
28. Spectre
29. Blinky
30. Clyde
31. Pinky
32. Inky
33. Puddle
34. Green Barrel
35. Table With Chairs
36. Floor Lamp
37. Chandelier
38. Hanged Man
39. Dog Food
40. White Pillar
41. Green Plant
42. Skeleton Flat
43. Sink
44. Brown Plant
45. Vase
46. Table
47. Ceiling Light
48. Kitchen Stuff
49. Suit of Armor
50. Empty Cage
51. Cage with Skeleton
52. Bones 1
53. Gold Key
54. Silver Key

55. Bed
56. Basket
57. Food
58. First Aid Kit
59. Clip
60. Machine Gun
61. Gatling Gun
62. Cross
63. Chalice
64. Chest of Jewels
65. Crown
66. Extra Life
67. Blood
68. Barrel
69. Well
70. Empty Well
71. Gibs
72. Flag
73. Call Apogee
74. Bones 2
75. Bones 3
76. Bones 4
77. Pots
78. Stove
79. Spears
80. Vines
81. Dead Guard
82. Skulls on a Stick
83. Cage with Blood
84. Cage with Skulls
85. Red Ceiling Light
86. Cow Skull
87. Well with Blood
88. Angel of Death Statue
89. Brown Column
90. Big Ammo Pack
91. Truck Rear
92. Spear of Destiny
93. Green Guard
94. Lost Episodes SS Nazi
95. Doberman
96. Bat
97. Lost Episodes Officer

98. Dead Green Guard
99. Submarine Willy
100. Professor Quarkblitz
101. Major Hans “The Axe” Schlieffen
102. Robot
103. Devil
104. Ghoulish Ghost
105. Puddle 2
106. Horizontal Green Barrel
107. Table With Chairs 2
108. Tall Floor Lamp
109. Chandelier 2
110. Hanged Man 2
111. Dog Food 2
112. Cracked White Pillar
113. Green Plant 2
114. Skeleton Flat 2
115. Brown Plant 2
116. Vase
117. Metal Table
118. Ceiling Light 2
119. Cage with Blood 2
120. Suit of Armor 2
121. Hanging Cage 2
122. Broken Cage
123. Bones 5
124. Cage with Rat
125. Dead Rat
126. Gibs 2
127. Tesla Coil
128. Red Ceiling Light 2
129. Bones 6
130. Bare Light Bulb
131. Green Slime
132. Chemical Table
133. Nuclear Barrel
134. Blue Column
135. Bubbles
136. Devil Statue
137. BJ Was Here
138. Spear of Destiny 2
139. Blue Clip
140. Ammo Box 2

141. Blue AK47
142. Food 2
143. Medikit 2
144. One Up Pill
145. Blood 2
146. Radio
147. Plutonium
148. Control Panel
149. Bomb
150. Skull Pile
151. Horizontal Barrel
152. Well 2
153. Empty Well 2
154. Yellow Key
155. Cyan Key
156. Blue Gatling Gun
157. Radioactive Mist

Triggers

A trigger is a switch action. For example, a trigger can cause a tile to exit the map, act as a push wall, or even to handle normal doors when the tile is activated.

```
trigger
{
      x = <integer>; // Coordinates based on grid.
      y = <integer>;
      z = <integer>; // Plane number
      activatenorth = <bool>;    // Default: true
      activatesouth = <bool>;    // Default: true
      activatewest = <bool>;     // Default: true
      activateeast = <bool>;     // Default: true
      action = <integer>;  // Action to perform.
      arg0 = <integer>;    // Default for args: 0
      arg1 = <integer>;
      arg2 = <integer>;
      arg3 = <integer>;
      arg4 = <integer>;

      playeruse = <bool>;  // Default: false
      monsteruse = <bool>; // Default: false
      playercross = <bool>;// Default: false
      repeatable = <bool>; // Default: false
```

```
        secret = <bool>;      // Default: false
    }
```

Tigger Actions

For any direction argument the direction is relative to the triggering side or absolute if 8 is added. (0 = right, 1 = back right, 2 = back, 3 = back left, 4 = left, 5 = towards left, 6 = towards, 7 = towards right, 8 = east, 9 = northeast, 10 = north, 11 = northwest, 12 = west, 13 = southwest, 14 = south, 15 = southwest). A tag is an arbitrary number used to link switches. Unused arguments should be 0 for forwards compatibility.

Speed is represented in the number of pixels per tic/16. For a standard pushwall/door this would be 8 and 16 respectively.

1. Door_Open(tag, speed, delay, lock, type) - Type is for how the door should animate and which orientation, 0 = Wolf3D sliding horizontal, 1 = Wolf3D sliding veritcal (further values are reserved). Each key is represented by a lock number defined in LOCKDEFS. 0 = None 1 = Gold 2 = Silver 101 = Gold+Silver. The delay is in tics until the door closes (300 for typical Wolf3D door, -1 means hold open until manually closed).
2. Pushwall_Move(tag, speed, direction, distance) - Distance is in tiles
3. Exit_Normal(pos) - Pos is reserved for later use, but will allow multiple starts for a level
4. Exit_Secret(pos)
5. Teleport_NewMap(map, pos, flags) - Flags: 1 = Keep direction, 2 = Keep position
6. Exit_VictorySpin()
7. Exit_Victory(pos)
8. Trigger_Execute(x, y, z) - Activates all triggers at the given map spot (even those without any other activation means.)

Planemaps

A tile tells the engine what to do for each tile on the grid. There must be Width*Height tile definitions. Tiles start at (0,0) and are assigned in row-major order. The index -1 can be used where a property is not needed. Planemaps assign data to planes in the order they are defined.

Tags are an arbitrary number for use with trigger actions. It is optional and will default to 0 if not specified for the spot.

```
planemap
{
      // {tile,sector,zone}
      {<integer>,<integer>,<integer>},
      // {tile,sector,zone,tag}
      {<integer>,<integer>,<integer>,<integer>},
      // ...
```

```
}
```