**AAS Documentation for Return to Castle Wolfenstein**

By Justin Marshall

PROPERTY OF JMV SOFTWARE – Work in progress

The AAS system is used in idTech3 and idTech4 games to provide a powerful pathfinding engine for ID games as well as licensee’s that license there engines. When Quake 3 was released under GPL the AAS code was included as well but didn’t really take off as much as it should off. This was mostly due to the complexity of the AAS system, the fact it was more geared toward bot play than using it for singleplayer monsters. Well five years later the source code for Return to Castle Wolfenstein was released and this document covers the specifics of the AAS system that’s used in RTCW.

The AAS system is very complex not only because the code is very robust but because of its readability. This usually scares a lot of people aware from even looking at the code base.

AAS File Format:

Unencrypted:

AAS File ID(4 bytes) -> “EAAS”

AAS Version(4 byte int) -> 8

Encrypted:

BSP Checksum(4 bytes)

Aas\_lump lumps[MAX\_LUMPS]

The encryption on the AAS header is a simple XOR’ed encryption scheme. It is import that each byte is xor’ed by a signed byte in C# or a unsigned char in c++.

//

// AAS\_DecryptHeader

//

private void AAS\_DecryptHeader( ref idFile file, int size ) {

int i;

idMsgReader buffer;

byte[] data = new byte[size];

for (i = 0; i < size; i++)

{

sbyte b = file.ReadSignedByte();

b ^= (sbyte)(i \* 119);

data[i] = (byte)b;

} //end for

buffer = new idMsgReader(data);

bspchecksum = buffer.ReadInt();

for (i = 0; i < aas\_lumptype.AAS\_LUMPS; i++)

{

lumps[i].fileofs = buffer.ReadInt();

lumps[i].filelen = buffer.ReadInt();

}

buffer.Dispose();

}