Subject: final scripts.dll 3.0 changelog and big secret feature announcement
Posted by jonwil on Thu, 16 Nov 2006 10:00:17 GMT
View Forum Message <> Reply to Message

Complete changelog for scripts.dll 3.0:
Migrated everything up to Visual C++ .NET 2005
Fixed the messagebox displayed if scripts.dll cant find a function in bhs.dll to display the correct message.
Fixed the "bhs.dll didnt load" and "scripts2.dll didnt load" messageboxs to display the correct message.
Fixed the "bhs.dll version mismatch" messagebox to display the correct message.
Increased the version to 3.0
new hook to detect players leaving the game
new engine call to display the "you dont have the required security to access this terminal" dialog for a given player.
new engine call to send a number to shaders.dll (e.g. to trigger or turn off a given post process shader effect) for a given player
JFW_Cinematic_Attack_Command (clone of M00_Cinematic_Attack_Command_DLS)
JFW_Cinematic (clone of Test_Cinematic)
JFW_Cinematic_Kill_Object (clone of M00_Cinematic_Kill_Object_DAY)
JFW_Reflect_Custom_Delay (this script will send any message it recieves back to whatever object sent it with a delay)
JFW_Radar_Jammer (script to disable the radar when a "jammer" unit is close to the radar dome)
JFW_Sonar_Pulse (script for a sonar pulse)
JFW_Global_Stealth_Controller (scripts based off the gap generator scripts intended for use as a stealth generator by e.g. reborn)
JFW_Stealth_Generator_Building (scripts based off the gap generator scripts intended for use as a stealth generator by e.g. reborn)
JFW_Stealth_Generator_Vehicle (scripts based off the gap generator scripts intended for use as a stealth generator by e.g. reborn)
JFW_Stealth_Generator_Ignored (scripts based off the gap generator scripts intended for use as a stealth generator by e.g. reborn)
There is one script for a global controler, one for a stealth generator building, one for a mobile stealth generator vehicle and one that goes on stuff
that should not be made stealth (such as stealth tanks)
JFW_Sidebar_Key_2 (script to display the sidebar when a key is pressed)
JFW_Sidebar_PT (triggers the sidebar when poked)
JFW_2D_Sound_Startup (script to play a 2D sound on startup)
JFW_Time_Remaining_Sounds (script to play 2D sounds to indicate time remaining in the game)
JFW_Vehicle_Lock (script to manage vehicle locking including theft by spies)
A small bug fix to JFW_Nod_Turret
A small bug fix to JFW_Nod_Obelisk
A small bug fix to JFW_Advanced_Guard_Tower_Missile
A small bug fix to JFW_Advanced_Guard_Tower_Gun
JFW_Disable_Loiter (clone of M00_Disable_Loiter_DAY)
JFW_InnateIsStationary (clone of M00_InnateIsStationary)
JFW_Generic_Conv (clone of M00_Generic_Conv_DME)
JFW_Disable_Hibernation (clone of M07_Disable_Hibernation)

JFW_Radar_Spy_Zone (script to cover putting a spy in the enemy radar dome/com center and having the radar come back on if yours is gone)
JFW_2D_Sound_Zone_Team (plays a 2D sound for a team after a timer has expired)
JFW_Repair_Zone_2 (script for a repair zone that costs money)
JFW_Infantry_Force_Composition_Zone (script to display enemy infantry composition for the spy)
JFW_Vehicle_Force_Composition_Zone (script to display enemy vehicle composition for the spy)
Kamuix_Death_Team_Win (new script by Kamuix)
Kamuix_Kill_Change (new script by Kamuix)
MDB_Set_Ammo_Granted_Weapon_On_Pickup (new script by WhiteDragon to set ammo on pickup)
MDB_Set_Ammo_Current_Weapon_On_Pickup (new script by WhiteDragon to set ammo on pickup)
MDB_Set_Ammo_On_Pickup (new script by WhiteDragon to set ammo on pickup)
RA_Vehicle_Regen (script to handle vehicle regeneration)
RA_Thief (script for a thief)
RA_Credit_Theft_Zone (script for a thief)
RA_DestroyNearest_OnDeath (script to destroy the nearest instance of a preset when the object with it attached is killed)
RA_Ore_Truck (script for a player controled ore truck)
RA_Ore_Field (script for a player controled ore truck)
RA_Ore_Delivery_Zone (script for a player controled ore truck)
RA_Vehicle_Team_Set (script to make a vehicle remain owned by the team for a little while when the driver gets out)
RA_Vehicle_Team_Timer (script to make a vehicle remain owned by the team for a little while when the driver gets out)
RA_Visible_Driver (makes a person model show up in a vehicle when someone is inside it)
RA_Vision_Control (controls vision such as Set_Screen_Fade_Opacity and
Set_Screen_Fade_Color and fog)
RA_Fog_Level_Settings (controls fog settings for the level)
SH_Spawn_Difficulty (script to spawn a different object depending on the current difficulty level)
RA_Global_Gap_Controller (scripts for a gap generator)
RA_Gap_Generator_Building (scripts for a gap generator)
RA_Gap_Generator_Vehicle (scripts for a gap generator)
RA_Gap_Generator_Ignored (scripts for a gap generator)
There is one script for a global controler, one for a gap generator building, one for a mobile gap generator vehicle and one that goes on stuff
that should not be cloaked (such as phase tanks if they get implemented)
Changes to the ExpVehFac scripts to send a custom to the vehicle letting the vehicle know who its owner is (only if its flying or naval, ground is handled elsewhere)
A small change to ActionParamsStruct
A fix to the definition of Get_Sync_Time
void Destroy_Connection(int PlayerID); //Drop a player from the game by cutting off their network link
A clone of RawFileCass (only usable on win32, intended so that the clones of ChunkLoadClass and ChunkSaveClass can be used outside of renegade)
Fixes to a few engine calls (e.g. memory leak fixes)
Removal of the GetMaxPlayerID engine call (it was broken and didnt work)
Removal of the Change_String engine call (now that we have a proper implementation of

StringClass, you can use StringClass::Operator= instead
FileClass *Get_Data_File(const char *file); //Open a file using the mix file opening logic
void Close_Data_File(FileClass *file); //Close a file that was opened with Get_Data_File
A complete clone of ChunkLoadClass
A complete clone of ChunkSaveClass
bool Is_Unit_In_Range(const char *preset,float range,Vector3 location,int team); //Is the given unit type in range of a location
void Set_Ladder_Points(int PlayerID,int amount); //Set the ladder points of a player
void Set_Rung(int PlayerID,int amount); //Set the rung of a player
int Get_Current_Weapon_Style(GameObject *obj); //Get weapon style for an objects current gun
int Get_Position_Weapon_Style(GameObject *obj,int position); //Get weapon style for an objects gun at a specific position
int Get_Weapon_Style(GameObject *obj,const char *weapon); //Get weapon style for a specific gun (if the object doesnt have the gun, return is zero)
A complete clone of StringClass
const char *Get_Powerup_Weapon_By_Obj(GameObject *Powerup); //Get the name of a powerup weapon given a PowerupGameObj
void Disable_Preset_By_Name(unsigned int Team,const char *Name); //Disable a preset by name
void Disable_Enlisted_By_Name(unsigned int Team,const char *Name); //Disable an enlisted unit by name
void Create_Effect_All_Of_Preset(const char *object,const char *preset,float ZAdjust,bool ZSet); //Create an object above all objects of a given preset, also set the facing to match the object its being created over.
float Steal_Team_Credits(float percentage, int team); //Steal credits from a team
GameObject *Find_Nearest_Preset(Vector3 position, const char* preset); //find the nearest preset
GameObject *Find_Random_Preset(const char* preset, int min, int max); //find a random preset
void Ranged_Stealth_On_Team(Gap_ListNode* FirstNode); //Apply stealth to all units in the range of this on the relavent team
void Send_Custom_All_Players(int message,GameObject *sender,int team); //send a custom to all players
void Ranged_Gap_Effect(Gap_ListNode* FirstNode); //apply gap effect to all units of a team within a range
AmmoDefinitionClass *Get_Weapon_Ammo_Definition(const char *weapon,bool PrimaryFire); //Get the AmmoDefinitionClass of a weapon given its preset name
AmmoDefinitionClass *Get_Current_Weapon_Ammo_Definition(GameObject *obj,bool PrimaryFire); //Get the AmmoDefinitionClass of an objects current weapon
AmmoDefinitionClass *Get_Position_Weapon_Ammo_Definition(GameObject *obj,int position,bool PrimaryFire); //Get the AmmoDefinitionClass of an objects weapon at tbe specified position
WeaponDefinitionClass *Get_Weapon_Definition(const char *weapon); //Get the WeaponDefinitionClass of a weapon given its preset name
WeaponDefinitionClass *Get_Current_Weapon_Definition(GameObject *obj); //Get the WeaponDefinitionClass of an objects current weapon
WeaponDefinitionClass *Get_Position_Weapon_Definition(GameObject *obj,int position); //Get the WeaponDefinitionClass of an objects weapon at the specified position
ExplosionDefinitionClass *Get_Explosion(const char *explosion); //Get the ExplosionDefinitionClass of an explosion given its preset name
const wchar_t *CharToWideChar(const char *str); //convert a char to a wide char

Definitions of the remaining unknowns in cGameData
A complete clone of WideStringClass
const char *Get_Translated_Definition_Name(const char *preset); //Get the translated name string for a preset name
A clone of FileClass
A clone of FileFactoryClass
A clone of ReferenceableClass
A clone of ReferencerClass
A clone of PlayerDataClass
Updates to the clone of NetworkObjectClass
A clone of Matrix4
A clone of ChunkHeader
A clone of MicroChunkHeader
A clone of IOVector2Struct
A clone of IOVector3Struct
A clone of IOVector4Struct
A clone of IOQuaternionStruct
A clone of RefCountClass
A clone of BaseGameObjDef
A clone of BaseGameObj
A clone of ScriptableGameObjDef
A clone of AudioCallbackClass
A clone of ScriptableGameObj
A clone of TransitionGameObjDef
A clone of TransitionGameObj
A clone of DamageZoneGameObjDef
A clone of DamageZoneGameObj
A clone of ScriptZoneGameObjDef
A clone of ScriptZoneGameObj
A clone of DefenceObjectDefClass
A clone of DamageableGameObjDef
A clone of DefenceObjectClass
A clone of DamageableGameObj
A clone of cPlayer
A clone of Vector2
A clone of RectClass
A clone of AmmoDefinitionClass
A clone of WeaponDefinitionClass
A clone of MuzzleFlashClass
A clone of WeaponClass
A clone of ExplosionDefinitionClass
Changes to the following engine calls so that they go through the new class clones:
As_ScriptableGameObj
As_PhysicalGameObj
As_VehicleGameObj
As_SmartGameObj
As_DamageableGameObj
As_SoldierGameObj

As_ScriptZoneGameObj
As_BuildingGameObj
Get_Object_Type
Set_Object_Type
Get_Definition
Post_Re_Init
Get_Player_Name
Get_Player_Name_By_ID
Change_Team_By_ID
Change_Team
Get_Player_ID
Get_GameObj
Set_Max_Health
Set_Max_Shield_Strength
Get_Shield_Type
Get_Skin
Set_Skin
Remove_Script
Remove_Duplicate_Script
Remove_All_Scripts
Get_Current_Weapon
Get_Current_Bullets
Get_Current_Clip_Bullets
Get_Current_Total_Bullets
Get_Position_Bullets
Get_Position_Clip_Bullets
Get_Position_Total_Bullets
Get_Bullets
Get_Clip_Bullets
Get_Total_Bullets
Get_Current_Max_Bullets
Get_Current_Clip_Max_Bullets
Get_Current_Total_Max_Bullets
Get_Position_Max_Bullets
Get_Position_Clip_Max_Bullets
Get_Position_Total_Max_Bullets
Get_Max_Bullets
Get_Max_Clip_Bullets
Get_Max_Total_Bullets
Is_Script_Attached
Is_A_Building
Get_Team
Get_Rank
Get_Kills
Get_Deaths
Get_Score
Get_Money
Get_Damage_Points

Get_Death_Points
Set_Death_Points
Set_Damage_Points
Get_Translated_String
Get_Wide_Translated_String
Get_Translated_Preset_Name
Get_Current_Wide_Translated_Weapon
Get_Translated_Weapon
Get_C4_Planter
Get_C4_Attached
Get_Player_Color
Set_Current_Bullets
Set_Current_Clip_Bullets
Set_Position_Bullets
Set_Position_Clip_Bullets
Set_Bullets
Set_Clip_Bullets
Get_Vehicle_Owner
Set_Enlisted
Set_Beacon
Set_Refill
Set_Preset
Set_Alternate
Disable_Preset
Disable_Enlisted
Purchase_Item
IsInsideZone
Create_Building
Get_Zone_Type
Get_Zone_Box
Set_Zone_Box
Create_Zone
Set_Money
Set_Score
A new dll, shaders.dll that provides hooks into the rendering engine to allow for custom shaders (all the stuff below applies to shaders.dll):
Definition of ProgrammableShaderClass, the base class for all custom shaders
Definition of the shader factory system used to register custom shaders
Definition of ShaderManagerClass, the class that manages all the loaded custom shaders (loads shader database files, unloads shader database files etc)
Definition of the base exported shaders.dll functions:
Release_Resources (called when the device is released or reset)
Reload_Resources (called after the device is reset)
Render (called to do actual rendering)
MapLoaded (called on map load)
MapUnloaded (called on map unload)
FrameStart (called on frame start)
FrameEnd (called on frame end before UI is drawn)

ScopeTrigger (called when scope is triggered or closed)
ScopeChange (called when scope is changed)
ShaderSet (called when set through engine call in bhs.dll, used to allow scripts to trigger specific stuff in shaders.dll)
New funcionality (in bhs.dll and shaders.dll) that overrides the drawing of most in-game meshes to add Tangent and Binormal data to them.
A series of engine calls (some copied from engine.h/engine.cpp) for shaders to use.
A clone of ShaderClass
A clone of several enums that apply to ShaderClass
A clone of the WW3DFormat enum
A clone of enums that apply to TextureClass and VertexMaterialClass
A clone of TextureMapperClass
A clone of FVFInfoClass
A clone of VertexBufferClass
A clone of IndexBufferClass
A clone of Vector4
A clone of VertexMaterialClass
A clone of TextureClass
A clone of RenderStateStruct
A clone of DX8Caps
A clone of VertexBufferLockClass
A clone of VertexBufferClass::WriteLockClass
A clone of VertexBufferClass::AppendLockClass
A clone of DX8VertexBufferClass
A clone of VertexFormatXYZNDUV2
A clone of SortingVertexBufferClass
extern unsigned int *render_state_changed; //Which render states are to be updated, uses ChangedStates enum
extern RenderStateStruct *render_state; //Current render state
extern myIDirect3DDevice8 **Direct3DDevice; //Current Direct3D8 Device
extern IDirect3DDevice9 *Direct3DDevice9; //Current Direct3D9 Device
extern unsigned int *MinTextureFilters;
extern myIDirect3D8 **Direct3D; //Current Direct3D Interface
unsigned long CRC_Memory(const unsigned char *data,unsigned long length,unsigned long crc); //Calculate the CRC of a block of memory
void TextureInitialize(unsigned int stage); //initialize and load a current state texture if it hasnt already been loaded
void TextureInitialize2(TextureClass *t); //initialize and load a texture if it hasnt already been loaded
void Draw(unsigned int primitive_type, unsigned short start_index, unsigned short polygon_count, unsigned short min_vertex_index, unsigned short vertex_count); //Call this to do the normal rendering pipe
void Buffers_Apply(); //Apply the vertex and index buffers
bool Texture_Exists(char *texturename); //Does a texture exist
TextureClass *_stdcall Load_Texture(const char *path,MipCountType mip,WW3DFormat format, bool IsCompressionAllowed); //Load a texture
extern myIDirect3DTexture8 **Textures; //Pointers for current IDirect3DTexture interfaces
extern unsigned int *SyncTime; //Current sync time

extern bool *TexturingEnabled; //Is texturing enabled
extern DX8Caps *CurrentCaps; //Current caps bits
extern D3DCOLOR *AmbientColor; //current ambient color
void _stdcall Free_Texture(TextureClass *texture); //Free a texture
ShaderCaps, a class to retrieve capabilities relavent to shaders.
extern unsigned int *MinTextureFilters; //texture filter values
extern unsigned int *MagTextureFilters; //texture filter values
extern unsigned int *MipMapFilters; //texture filter values
extern D3DMATERIAL9 *DefaultMaterial; //default material structure
extern bool *CurrentDX8LightEnables; //current light enables
Vector4 *GetColorVector4(D3DCOLOR *color); //Converts a D3DCOLOR into a Vector4
extern unsigned int *ActiveDialog; //If this is non zero, there is a dialog on the screen otherwise there is no dialog on the screen
Matrix4* Get_Projection_Matrix(); //Get the projection matrix
void Set_Texture_Stage_State(DWORD Stage,D3DTEXTURESTAGESTATETYPE Type,DWORD Value); //Set a Texture Stage State Type
void Set_Render_State(D3DRENDERSTATETYPE State,DWORD Value); //Set a Render State Type
void Set_Light(int pos,D3DLIGHT8 *light); //Set a Light
void Draw_Skin(char *fvfcc); //draw skin models
void Draw_Rigid(char *fvfcc,char *MeshClass); //draw rigid models
extern unsigned int *_PolygonCullMode; //current polygon culling mode
extern ShaderCaps* TheShaderCaps; //ShaderCaps pointer
A new tool, the shader database editor. This edits shader databases.
Definition of engine calls required for the shader database editor.
Definition of EditorShaderClass, the base classe for the editor components of a shader
Definition of the shader factory system used to register custom shaders in the editor
Definition of EditorShaderManagerClass which manages shaders (loads and saves databases etc) in the editor
Classes and implementation for a Glow Shader including the shader database editor implementation so you can create this shader in a shader database plus shader code for the shader
Classes and implementation for a Glass Shader including the shader database editor implementation so you can create this shader in a shader database plus shader code for the shader and some sample textures you can use
Classes and implementation for an Offset Map Shader including the shader database editor implementation so you can create this shader in a shader database plus shader code for the shader
Classes and implementation for a do nothing shader that is applied to objects that dont have any other shader applied and that passes through to the fixed function pipeline
Classes and implementation for post process shaders that let you apply effects to the screen before the UI is drawn. (still WIP)
shader database editor code for post process shaders (still WIP)
A complete example shader database
Base implementation of the shader database editor, including resources
Changes to the way renegade detects video cards so that it will detect more video cards (especially newer ones that arent detected by the old code).
Also, it will correctly detect more video card driver versions. Plus, if you have a newer NVIDIA

card, it will enable and use DXT1 (whereas the old renegade code would disable it).
There is a new feature, d3d9. This consists of a dll called d3d9.dll that changes renegade so it talks to Direct3D9 instead of Direct3D8.
This replaces scorpio9a's old RenD3D9 (without the fancy options rend3d9 has and without the bugs rend3d9 has). This is a required part of scripts.dll, not using d3d8.dll (or using any other d3d8.dll such as scorpio9a's dll) will probobly lead to crashes.
This includes a feature whereby the size of the shadow texture is increased which leads to shadows that look better in game.
new console command VIEW that displays a dialog box containing a w3d file with an animation (similar to what the encyclopedia displays in SP).
new console command HUD that disables or enables the HUD.
A change to the way the player list is iterated (should fix the issues with some players not being displayed by the ID and PINFO console commands)
Cleanups to the way I patch memory that make applying patches and such easier and cleaner
Cleanups to bhs.dll as a result of the new class clones
Changes to how VehicleOwnershipDisable works so that when its enabled, it sends a custom to the vehicle letting the vehicle know who its owner is
New hud.ini keywords MenuHiliteColor to change the color you see when you mouse over a menu control.
MerchandiseTextColor for the color on a purchase button
ListColumnColor for changing the color for list controls
Removal of the StealthRenderState hud.ini keywords (which were broken and are made obsolete by shaders.dll anyway)
New keywords to give the sidebar seperate purchase sounds for refill, infantry and vehicles
New hud.ini keyword to change what registry key the renegade update/patching code reads the version number from
New hud.ini keyword WeaponImageVisibleNonVehicle that disables the display of the weapon icon when you are not in a vehicle (i.e. you see the steering wheel, gun and seat icons but not the weapon icon)
New feature to make the radar map rotate when you have a texture as the background (i.e. an overhead view texture of the map you are on) (I may have to drop this if it doesnt work the way I expect)
A fix from Black-Intel for the "vehicles getting stuck near ladders" problem
The Black-Intel wall lag fix
A change so that ::Created is called for C4 objects
A change so that the windows FDS doesnt try to write into the registry "RunOnce" key anymore.
A change so that the HUD is not affected by Set_Screen_Fade_Color or Set_Screen_Fade_Opacity anymore.
The black-intel turret lag fix.
A change to the edit box so that ctrl-x, ctrl-c and ctrl-v work for cut, copy and paste.
Cleanups/new info for the following classes:
DialogControlClass
Render2DClass
Render2DSentenceClass
ButtonCtrlClass
MerchandiseCtrlClass
ImageCtrlClass
ListIconMgrClas

ScrollBarCtrlClass
ListCtrlClass
DialogBaseClass
PopupDlgClass
Definitions of:
ListEntryClass
ListColumnClass
ListRowClass
AABoxClass
ViewerCtrlClass
InputCtrlClass
IMECandidateCtrlClass
EditCtrlClass
MenuEntryCtrlClass
CheckBoxCtrlClass
HealthBarCtrlClass
DropDownEntry
DropDownCtrlClass
ComboBoxCtrlClass
DialogTextClass
ChildDialogClass
A new dialog to configure bhs.dll features, it configures the following:
Client chat log enabled
Screenshot format
High quality shadows enabled
Also, it configures the extended keys.cfg keys in the way that keycfg.exe does.
However, you cant add keys to the list, only change the keys already in keys.cfg. If you wish to add
keys to the list, edit keys.cfg manually or use keycfg.exe
Hooks in bhs.dll to call shaders.dll
Changes to crashdump.txt to dump:
If appropriate, the current map, mod package, player count and time remaining
The CRC32 of all modules (not just a few)

The "big secret feature" I have talked about for a while is shaders.dll. What this basicly means is that there are now
hooks into the rendering engine to allow you to override the drawing of objects in the scene and draw them your own way (more speficially by applying Direct3D9 shaders to those objects).
You need C++ skills and knowledge of Direct3D to write custom shaders.
3.0 will include a Glow shader, a Glass/Environment Map shader plus an Offset Normal Mapping shader as examples.
Some pictures of what these shaders we include can do:
http://users.tpg.com.au/jfwfreo/glass2.png
http://users.tpg.com.au/jfwfreo/offset2.png

With scripts.dll 3.0, you MUST put the d3d8.dll in your renegade folder along with bhs.dll, scripts.dll and shaders.dll.
Not doing so (either on the client or the FDS) WILL cause problems. The same applies to using

any other d3d8.dll (such as rend3d9)

The fancy options that rend3d9 has (such as FSAA) are to be investigated for a future version of the scripts.dll.
