#include "sys\_chargen\_h"

#include "wrappers\_h"

#include "utility\_h"

#include "sys\_rewards\_h"

#include "approval\_h"

#include "sys\_autolevelup\_h"

/\* Jye Nicolson 5-Jan-2010

This function set duplicates the full functionality chain of UT\_HireFollower, with the following exceptions:

- Followers can gain XP

- Autolevel status can be set (default off)

- Followers can be set to any starting state (default Available) and will still be properly initalised and added to the party pool

- Autolevel tables for non-core followers can be explicitly set.

- Class and Specialisation can be chosen via script

- Followers without specialisations are granted a spec point by default.

It should only ever be called once each for characters you intend to be full followers.

Much of the protective code handling summoned creatures etc. in player\_core is not present here.

Calling the function:

Simple:

hireCustomFollower(oFollower, CLASS\_WARRIOR);

Change the class to CLASS\_WIZARD or CLASS\_ROGUE as appropriate.

This will hire your follower and make them available.

They will auto level up with a default package, and receive a free spec point.

Best Practice:

hireCustomFollower(oFollower, CLASS\_WARRIOR, PLT\_YOUR\_PARTY\_PLOT, YOUR\_FOLLOWER\_JOINED\_FLAG, ABILITY\_TALENT\_HIDDEN\_CHAMPION);

Where the plot and flag are those for your module (remember to create the plot and include it on the calling script), and ABILITY\_TALENT\_HIDDEN etc is the desired spec.

You should also have a custom ALTable set up.

See wiki for details, and remember to edit it in to GetCustomFollowerALTable below or pass it directly as an argument to hireCustomFollower.

Full argument list:

void hireCustomFollower (

object oFollower, //Pass your follower object, mandatory

int nForceClass, //Pass a Class constant here, usually CLASS\_ROGUE, CLASS\_WARRIOR, CLASS\_WIZARD. Mandatory due to a bug.

string sPlot = "", //It's recommended you have a plot flag to be set when the follower joins. Pass the plot constant here. Remember to #include in calling script

int nPlotFlag = "", //And then pass the flag constant. Will be set to TRUE if available.

int nForceSpec = 0, //This is the ID of the Specialisation you want. Note they are NOT classes, but abilities. The full list is:

//ABILITY\_SPELL\_HIDDEN\_ARCANE\_WARRIOR, ABILITY\_SPELL\_HIDDEN\_BLOODMAGE, ABILITY\_SPELL\_HIDDEN\_SHAPESHIFTER, ABILITY\_SPELL\_HIDDEN\_SPIRIT\_HEALER

//ABILITY\_SPELL\_HIDDEN\_BARD, ABILITY\_TALENT\_HIDDEN\_ASSASSIN, ABILITY\_TALENT\_HIDDEN\_DUELIST, ABILITY\_TALENT\_HIDDEN\_RANGER

//ABILITY\_TALENT\_HIDDEN\_BERSERKER, ABILITY\_TALENT\_HIDDEN\_CHAMPION, ABILITY\_TALENT\_HIDDEN\_REAVER, ABILITY\_TALENT\_HIDDEN\_TEMPLAR

//I recommended forcing a spec, particularly if your ALTable includes abilities from one.

int nALTable = 0, //This is the ID of an ALTable from 2DA\_base.GDA or your module's m2DA\_base\_\*.GDA I recommended the latter, but you can edit that into GetCustomFollowerALTable below rather than passing it.

int bInvokePicker = FALSE, //Sets whether the party picker should be opened on hiring. I think it's cleaner to call the picker outside this script, particularly if you have multiple hires at once.

int nInitialState = FOLLOWER\_STATE\_AVAILABLE, //This sets whether the follower joins the active party or not. Options are:

//FOLLOWER\_STATE\_ACTIVE (put them in the active party)

//FOLLOWER\_STATE\_LOCKEDACTIVE (force them into the active party and keep them there, remember to change this later.

//FOLLOWER\_STATE\_AVAILABLE (make them available on the party picker (if you've set it up for them), but not in the active party)

//Plus some others you're unlikely to need at this time. Defaults to AVAILABLE because having 4+ active followers is screwy.

string sCurrPlot = "", //If you set FOLLOWER\_STATE\_ACTIVE or FOLLOWER\_STATE\_LOCKEDACTIVE, the script will check to see if you passed this.

//It is recommended that you have a plot flag set for a given follower being in the active party, this makes conversation interjection etc. much easier.

int nCurrPlotFlag = 0, //This flag will be set if FOLLOWER\_STATE\_ACTIVE or FOLLOWER\_STATE\_LOCKEDACTIVE are true

//AND sCurrPlot has a value AND nCurrPlotFlag is > 0.

//ie if you added someone to the active party and have a plot flag to cope with it.

int nAutolevel = 0, //Sets the Autolevel flag on the character sheet. 0 is off, 1 is on, 2 forces it on and removes it so the player can't turn it off.

bFreeSpecPoint = TRUE, //This grants a specialisation point to the follower if they do not have a specialisation.

//It's important to set this false for classes that do not have specs, such as CLASS\_DOG.

int nTargetLevel = 0, //If you want a specific level, set this. Generally not worthwhile unless you set it higher than the player, since they'll just get XP from the party picker anyway.

int nMinLevel = 0 //Set this if there's a specific level you don't want the follower to go below. Probably only useful if the PC might be very low level but not necessarily so.

)

\*/

/\* GetCustomFollowerALTable()

This function is where you put your custom table assignments.

You should explicitly test for the tag of your follower (not mine!) and assign a value to nTable from your m2DA extension to M2DA\_base

See wiki for details on how to do this, or ignore it to get the default Warrior/Rogue/Wizard AL tables.

NOTE: you MUST explicitly set a table for non-Warrior/Rogue/Wizards, eg dogs. Use TABLE\_AL\_DOG for a default Mabari.

\*/

int GetCustomFollowerALTable(object oFollower) {

int nTable = \_GetTableToUseForAL(oFollower);

if (GetTag(oFollower) == "bc\_party\_miera") {

nTable = 50143;

}

if (GetTag(oFollower) == "bc\_party\_jysavin") {

nTable = 50144;

}

if (GetTag(oFollower) == "bc\_party\_geldual") {

nTable = 50145;

}

if (GetTag(oFollower) == "bc\_party\_braghon") {

nTable = 50146;

}

return nTable;

}

// This just cleans up the main function a little

int GetCustomFollowerTargetLevel(object oFollower, object oHero, int nPackage, int nMinLevel = 0) {

int nPlayerLevel = GetLevel(oHero);

int nTargetLevel = 0;

if((nPlayerLevel >= 13) || (nPlayerLevel == 1) || (!\_UT\_GetIsPlotFollower(oFollower))) {

nTargetLevel = nPlayerLevel;

} else {

nTargetLevel = nPlayerLevel + 1;

}

if (nMinLevel == 0) { //If nMinLevel is not specified, checks package 2DA for a value

nMinLevel = GetM2DAInt(TABLE\_PACKAGES, "MinLevel", nPackage);

}

if(nMinLevel > 0 && nMinLevel > nTargetLevel) {

nTargetLevel = nMinLevel;

}

return nTargetLevel;

}

// Moving this black box out :) I don't really understand it, but it should function if you have tactics set up in a package.

void InitCustomFollowerTactics(object oFollower, int nPackage) {

int nTableID = GetM2DAInt(TABLE\_PACKAGES, "FollowerTacticsTable", nPackage);

if (nTableID != -1)

{

int nRows = GetM2DARows(nTableID);

int nMaxTactics = GetNumTactics(oFollower);

int nTacticsEntry = 1;

int i;

for (i = 1; i <= nRows && nTacticsEntry <= nMaxTactics; ++i)

{

int bAddEntry = FALSE;

int nTargetType = GetM2DAInt(nTableID, "TargetType", i);

int nCondition = GetM2DAInt(nTableID, "Condition", i);

int nCommandType = GetM2DAInt(nTableID, "Command", i);

int nCommandParam = GetM2DAInt(nTableID, "SubCommand", i);

int nUseType = GetM2DAInt(TABLE\_COMMAND\_TYPES, "UseType", nCommandType);

if (nUseType == 0)

{

bAddEntry = TRUE;

}

else

{

bAddEntry = HasAbility(oFollower, nCommandParam);

}

if (bAddEntry)

{

SetTacticEntry(oFollower, nTacticsEntry, TRUE, nTargetType, nCondition, nCommandType, nCommandParam);

++nTacticsEntry;

}

}

}

}

/\* InitCustomFollowerSpec:

This function tries to set the forced Specialisation. If there is none, it checks the package for one.

If there isn't either of those, it grants a free spec point if bFreeSpecPoint is true.

\*/

void InitCustomFollowerSpec(object oFollower, int nPackage, int nForceSpec, int bFreeSpecPoint) {

// Find specialization, and optionally add a spec point if none is found.

if (nForceSpec == 0) {

int nSpecAbility = GetM2DAInt(TABLE\_PACKAGES, "switch1\_class", nPackage); // followers can have only 1 advanced class

if(nSpecAbility > 0)

{

AddAbility(oFollower, nSpecAbility);

} else {

if (bFreeSpecPoint) {

SetCreatureProperty(oFollower, 38, 1.00);

}

}

} else {

AddAbility(oFollower, nForceSpec);

}

}

/\* hireCustomFollower() (See doco at top of page)

I strongly suggest you reorder the parameters if you're adding many followers with advanced options.

Feel free to leave them alone if you only want to set class, plot, spec or don't mind long declarations.

Note nForceClass is currently compulsory due to flakiness with GetCreatureCoreClass()

\*/

void hireCustomFollower(object oFollower, int nForceClass, string sPlot = "", int nPlotFlag = 0, int nForceSpec = 0,

int nALTable = 0, int bInvokePicker = FALSE, int nInitialState = FOLLOWER\_STATE\_AVAILABLE, string sCurrPlot = "",

int nCurrPlotFlag = 0, int nAutolevel = 0, int bFreeSpecPoint = TRUE, int nTargetLevel = 0, int nMinLevel = 0)

{

object oHero = GetHero();

/\* ################# BEGIN BASIC FOLLOWER JOIN BLOCK ###################

This loosely replicates WR\_SetFollowerState.

\*/

if (nForceClass == 0) {

nForceClass = GetCreatureCoreClass(oFollower); //This is not working. Hence nForceClass mandatory.

}

SetGroupId(oFollower, GetGroupId(oHero)); //Puts the follower in the pc's Group.

SetEventScript(oFollower, RESOURCE\_SCRIPT\_PLAYER\_CORE); //This makes them act like a player.

SetFollowerState(oFollower, nInitialState); //This sets whether they are available, in the active party etc.

/\* ################# END BASIC FOLLOWER JOIN BLOCK ##################### \*/

/\* ################# BEGIN PLAYER\_CORE EVENT\_TYPE\_PARTY\_MEMBER\_HIRED EMULATION #################

This replicates the EVENT\_TYPE\_PARTY\_MEMBER\_HIRED handler from player\_core, stripped down for simplicity and allowing our custom options.

\*/

Chargen\_EnableTacticsPresets(oFollower); //I assume this is important.

SetLocalInt(oFollower, FOLLOWER\_SCALED, 1); //This should prevent the follower being rescaled by player\_core or what have you

int nPackage = GetPackage(oFollower); //Gets the package, which will be used to find a number of 2DA IDs.

int nPackageClass = GetM2DAInt(TABLE\_PACKAGES, "StartingClass", nPackage); //I don't think this is used, even by player\_core

// set behavior according to package

int nBehavior = GetM2DAInt(TABLE\_PACKAGES, "FollowerBehavior", nPackage);

if(nBehavior >= 0) {

SetAIBehavior(oFollower, nBehavior);

}

Chargen\_InitializeCharacter(oFollower); //We initialise the follower and choose race/class.

Chargen\_SelectRace(oFollower,GetCreatureRacialType(oFollower));

Chargen\_SelectCoreClass(oFollower,nForceClass);

if (nTargetLevel == 0) { //This block picks a target level if not specified

nTargetLevel = GetCustomFollowerTargetLevel(oFollower, oHero, nPackage, nMinLevel);

}

int nXp = RW\_GetXPNeededForLevel(Max(nTargetLevel, 1)); //Here is where the XP is calculated and rewarded

RewardXP(oFollower, nXp, FALSE, FALSE);

// -------------------------------------------------------------

// add hidden approval talents - (JN: I don't know how to set these yet, but when I figure it out this should make it work)

// -------------------------------------------------------------

int nIndex = Approval\_GetFollowerIndex(oFollower);

Approval\_AddFollowerBonusAbility(nIndex, 0);

//Handle Specialisation

InitCustomFollowerSpec(oFollower, nPackage, nForceSpec, bFreeSpecPoint);

// -------------------------------------------------------------

// This spends all available attribute and stat points on the

// creature according to the levelup table. (JN: this replicates AL\_DoAutoLevelUp but with our choice of table)

// -------------------------------------------------------------

if (nALTable == 0) {

nALTable = GetCustomFollowerALTable(oFollower);

}

AL\_SpendAttributePoints(oFollower, nALTable, FALSE);

AL\_SpendSkillPoints(oFollower, nALTable, TRUE);

AL\_SpendSpecializationPoints(oFollower, nALTable);

AL\_SpendTalentSpellPoints(oFollower, nALTable, TRUE);

// -------------------------------------------------------------------------

// Update various UIs

// -------------------------------------------------------------------------

Chargen\_SetNumTactics(oFollower);

SetCanLevelUp(oFollower,Chargen\_HasPointsToSpend(oFollower));

// load tactics

InitCustomFollowerTactics(oFollower, nPackage);

/\* ################# END PLAYER\_CORE EVENT\_TYPE\_PARTY\_MEMBER\_HIRED EMULATION ################# \*/

SetAutoLevelUp(oFollower, nAutolevel); //This is the autolevel flag on the character sheet.

//Set plot flags

if (!((sPlot == "") || (nPlotFlag == 0))) { //Joined Party

WR\_SetPlotFlag(sPlot, nPlotFlag, TRUE);

}

if ((nInitialState == FOLLOWER\_STATE\_ACTIVE) || (nInitialState == FOLLOWER\_STATE\_LOCKEDACTIVE)) {

if (!((sCurrPlot == "") || (nCurrPlotFlag == 0))) {

WR\_SetPlotFlag(sCurrPlot, nCurrPlotFlag, TRUE); //Currently in Party

}

}

// Invoke picker if requested.

if (bInvokePicker) {

SetPartyPickerGUIStatus(2);

ShowPartyPickerGUI();

}

}