

Гайд по GMST от W_wex

The Elder Scrolls III
MORROWIND

Нет знаний, не дающих силу.
2023

Оглавление

1. GMST	8
fRepairMult.....	9
fRepairAmountMult	10
fSpellValueMult.....	11
fSpellMakingValueMult.....	12
fEnchantmentValueMult	13
fTravelMult.....	14
fTravelTimeMult.....	15
fMagesGuildTravel	16
fRestMagicMult	17
fWortChanceValue	18
fMinWalkSpeed	19
fMaxWalkSpeed	20
fMinWalkSpeedCreature	21
fMaxWalkSpeedCreature	22
fEncumberedMoveEffect.....	23
fBaseRunMultiplier	24
fAthleticsRunBonus.....	25
fJumpAcrobaticsBase	26
fJumpAcroMultiplier.....	27
fJumpEncumbranceBase	28
fJumpEncumbranceMultiplier.....	29
fJumpRunMultiplier	30
fJumpMoveBase	31
fJumpMoveMult.....	32
fSwimWalkBase	33
fSwinRunBase.....	34
fSwimWalkAthleticsMult	35
fSwimRunAthleticsMult	36
fSwimHeightScale	37
fHoldBreathTime	38
fHoldBreathEndMult.....	39
fSuffocationDamage.....	40
fMinFlySpeed	41
fMaxFlySpeed	42
fStromWindSpeed	43
fStromWalkMult	44
fFallDamageDistanceMin	45
fFallDistanceBase	46
fFallDistanceMult.....	47
fFallAcroBase	48
fFallAcroMult	49
iMaxActivateDist.....	50
iMaxInfoDist	51
fVanityDelay	52
fMaxHeadTrackDistance	53
fInteriorHeadTrackMult.....	54
iHelmWeight.....	55
iPauldronWeight.....	56
iCuirassWeight	57
iGauntletWeight.....	58

iGreavesWeight.....	59
iBootsWeight	60
iShieldWeight	61
fLightMaxMod.....	62
fMedMaxMod	63
fUnarmoredBase1	64
fUnarmoredBase2.....	65
iBaseArmorSkill.....	66
fBlockSkillBonus.....	67
fDamageStrenghtBase.....	68
fDamageStrenghtMult	69
fSwingBlockBase	70
fSwingBlockMult.....	71
fFatigueBase	72
fFatigueMult	73
fFatigueReturnBase	74
fFatigueReturnMult.....	75
fEndFatigueMult	76
fFatigueAttackBase	77
fFatigueAttackMult	78
fWeaponFatigueMult	79
fFatigueBlockBase	80
fFatigueBlockMult.....	81
fWeaponFatigueBlockMult	82
fFatigueRunBase	83
fFatigueRunMult.....	84
fFatigueJumpBase	85
fFatigueJumpMult.....	86
fFatigueSwimWalkBase	87
fFatigueSwimRunBase.....	88
fFatigueSwimWalkMult.....	89
fFatigueSwimRunMult	90
fFatigueSneakBase	91
fFatigueSneakMult	92
fMinHandToHandMult.....	93
fMaxHandToHandMult.....	94
fHandToHandHealthPer	95
fCombatInvisoMult	96
fCombatKODamageMult	97
fCombatCriticalStrikeMult.....	98
iBlockMinChance	99
iBlockMaxChance	100
fLevelUpHealthEndMult	101
fSoulGemMult	102
fEffectCostMult.....	103
fSpellPriceMult	104
fFatigueSpellBase	105
fFatigueSpellMult	106
fFatigueSpellCostMult.....	107
fPotionStrengthMult.....	108
fPotionT1MagMult	109
fPotionT1DurMult	110

fPotionMinUsefulDuration	111
fPotionT4BaseStrengthMult	112
fPotionT4EquipStrengthMult	113
fIngredientMult	114
fMagicItemCostMult	115
fMagicItemPriceMult	116
fMagicItemOnceMult	117
fMagicItemUsedMult	118
fMagicItemStrikeMult	119
fMagicItemConstantMult	120
fEnchantmentMult	121
fEnchantmentChanceMult	122
fPCbaseMagickaMult	123
fNPCbaseMagickaMult	124
fAutoSpellChance	125
fAutoPCSpellChance	127
iAutoSpellTimesCanCast	128
iAutoSpellAttSkillMin	130
iAutoSpellAlterationMax	132
iAutoSpellConjurationMax	134
iAutoSpellDestructionMax	136
iAutoSpellIllusionMax	138
iAutoSpellMysticismMax	140
iAutoSpellRestorationMax	142
iAutoPCSpellMax	144
iAutoRepFacMod	145
iAutoRepLevMod	146
iMagicItemChargeOnce	147
iMagicItemChargeConst	148
iMagicItemChargeUse	149
iMagicItemChargeStrike	150
iMonthsToRespawn	151
fCorpseClearDelay	152
fCorpseRespawnDelay	153
fBarterGoldResetDelay	154
fEncumbranceStrMult	155
fPickLockMult	156
fTrapCostMult	157
fMessageTimePerChar	158
fMagicItemRechargePerSecond	159
i1stPersonSneakData	160
iBarterSuccessDisposition	161
iBarterFailDisposition	162
iLevelupTotal	163
iLevelupMajorMult	164
iLevelupMinorMult	165
iLevelupMajorMultAttribute	166
iLevelupMinorMultAttribute	167
iLevelupMiscMultAttribute	168
iLevelupSpecialization	169
iLevelUp01Mult	170
iLevelUp02Mult	171

iLevelUp03Mult	172
iLevelUp04Mult	173
iLevelUp05Mult	174
iLevelUp06Mult	175
iLevelUp07Mult	176
iLevelUp08Mult	177
iLevelUp09Mult	178
iLevelUp10Mult	179
iSoulAmountForConstantEffect	180
fConstantEffectMult	181
fEnchantmentConstantDurationMult	182
fEnchantmentConstantChanceMult	183
fWeaponDamageMult	184
fSeriousWoundMult	185
fKnockDownMult	186
iKnockDownOddsBase	187
iKnockDownOddsMult	188
fCombatArmorMinMult	189
fHandToHandReach	190
fVoiceIdleOdds	191
iVoiceAttackOdds	192
iVoiceHitOdds	193
fProjectileMinSpeed	194
fProjectileMaxSpeed	195
fThrownWeaponMinSpeed	196
fThrownWeaponMaxSpeed	197
fTargetSpellMaxSpeed	198
fProjectileThrownStoreChance	199
iPickMinChance	200
iPickMaxChance	201
fDispRaceMod	202
fDispPersonnalityMult	203
fDispPersonnalityBase	204
fDispFactionMod	205
fDispFactionRankBase	206
fDispFactionRankMult	207
fDispCrimeMod	208
fDispDiseaseMod	209
iDispAttackMod	210
fDispWeaponDrawn	211
fDispBargainSuccessMod	212
fDispBargainFailMod	213
fDispPickPocketMod	214
iDaysinPrisonMod	215
fDispAttacking	216
fDispStealing	217
iDispTresspass	218
iDispKilling	219
iTrainingMod	220
iAlchemyMod	221
fBargainOfferBase	222
fBargainOfferMulti	223

fDispositionMod.....	224
fPersonalityMod	225
fLuckMod.....	226
fReputationMod.....	227
fLevelMod.....	228
fBribe10Mod.....	229
fBribe100Mod.....	230
fBribe1000Mod.....	231
fPerDieRollMult	232
fPerTempMult	234
iPerMinChance.....	235
iPerMinChange	236
fSpecialSkillBonus.....	237
fMajorSkillBonus	238
fMinorSkillBonus	239
fMiscSkillBonus	240
iAlarmKilling	241
iAlarmAttack	242
iAlarmStealing	243
iAlarmPickPocket.....	244
iAlarmTrespass	245
fAlarmRadius.....	246
iCrimeKilling	247
iCrimeAttack.....	248
fCrimeStealing.....	249
iCrimePickPocket.....	250
iCrimeTrespass	251
iCrimeTreshold.....	252
iCrimeTresholdMultiplier.....	253
fCrimeGoldDiscountMult	254
fCrimeGoldTurnInMult.....	255
iFightAttack	256
iFightAttacking.....	257
iFightDistanceBase	258
iFightDistanceMultiplier	259
iFightAlarmMult	260
iFightDispMult	261
fFightStealing	262
iFightPickpocket.....	263
iFightTrespass.....	264
iFightKilling.....	265
iFlee	266
iGreetDistanceMultiplier	267
iGreetDuration	268
fGreetDistanceReset.....	269
fIdleChanceMultiplier.....	270
fSneakUseDist	271
fSneakUseDelay	272
fSneakDistanceBase.....	273
fSneakDistanceMultiplier.....	274
fSneakSpeedMultiplier	275
fSneakViewMult	276

fSneakNoViewMult	277
fSneakSkillMult	278
fSneakBootsMult	279
fCombatDistance	280
fCombatAngleXY	281
fCombatAngleZ	282
fCombatForceSideAngle	283
fCombatTorsoSideAngle	284
fCombatTorsoStartPercent	285
fCombatTorsoStopPercent	286
fCombatBlockLeftAngle	287
fCombatBlockRightAngle	288
fCombatDelayCreature	289
fCombatDelayNPC	290
fAIMeleeWeaponMult	291
fAIRangeMeleeWeaponMult	292
fAIMagicSpellMult	293
fAIRangeMagicSpellMult	294
fAIMeleeArmorMult	295
fAIMeleeSumWeaponMult	296
fAIFleeHealthMult	297
fAIFleeFleeMult	298
fPickPocketMod	299
fSleepRandMod	300
fSleepRestMod	301
iNumberCreatures	302
fAudioDefaultMinDistance	303
fAudioDefaultMaxDistance	304
fAudioVoiceDefaultMinDistance	305
fAudioVoiceDefaultMaxDistance	306
fAudioMinDistanceMult	307
fAudioMaxDistanceMult	308
fNPCHealthBarTime	309
fNPCHealthBarFade	310
fDifficultyMult	311
fMagicDetectRefreshRate	312
fMagicStartIconBlink	313
fMagicCreatureCastDelay	314
fDiseaseXferChance	315
fElementalShieldMult	316
fMagicSunBlockedMult	317
Конец	318

1. GMST

GMST - это константы игрового мира, в них указывается максимальная и минимальная скорость передвижения NPC и существ, значения влияющие на цены перемещения, скорость и высоту прижка, значения влияющие на получение стартовых заклинаний, значений влияющих на прокачку навыков, проще говоря описывает фундаментальные значения и множители в мире Морровинда.

Информация проверена мной на сколько это было возможно, местами требует дополнения и дальнейшего изучения.

***Все фрагменты кода из OpenMW взяты с сайта:
[https://wiki.openmw.org/index.php?title=GMSTs_\(status\)](https://wiki.openmw.org/index.php?title=GMSTs_(status))
От 13.03.23***

Если у вас замечания, предложения или дополнения по тексту, обязательно напишите мне в дискорд W_wex#9962

fRepairMult

fRepairMult - множитель который используется для определения цены ремонта вещей у NPC.
Значение по умолчанию: 1.0

Известная формула:

*Цена ремонта = (2 * Damage Points * **fRepairMult**).*

Часть кода из OpenMW:

$p = \max(1, \text{basePrice})$

$r = \max(1, \text{int}(\text{maxDurability} / p))$

$x = \text{int}((\text{maxDurability} - \text{durability}) / r)$

$x = \text{int}(\text{fRepairMult} * x)$

$\text{cost} = \text{barterOffer}(\text{npc}, x, \text{buying})$

basePrice - цена вещи

maxDurability - максимальная прочность

durability - текущая прочность

fRepairAmountMult

fRepairAmountMult - определяет какой процент от максимального значения предмета будет восстановлен при использовании инструментов для ремонта.

Значение **по умолчанию: 3.0**

Часть кода из OpenMW:

"Ошибка в оригинальной игре: из-за большей усталости легче починить предмет. Игра должна была умножать на fatigueTerm, а не делить на него." (Перевод translate.google)

fatigueTerm = fFatigueBase - fFatigueMult(1 - normalisedFatigue)*

where normalisedFatigue is a function of fatigue. empty fatigue bar -> 0.0, full fatigue bar -> 1.0

*x = (0.1 * pcStrength + 0.1 * pcLuck + armorerSkill) / fatigueTerm*

roll 100, if roll <= x then repair continues

*y = int(fRepairAmountMult * hammerQuality * roll)*

y = max(1, y)

repair item by y points

fSpellValueMult

fSpellValueMult - отвечает за увеличение или уменьшение стоимости заклинаний при изучении у NPC.

Значение **по умолчанию: 10**

Часть кода из OpenMW:

```
cost spell = barterOffer(npc, spell.magickaCost * fSpellValueMult, buying)
```

fSpellMakingValueMult

fSpellMakingValueMult - множитель используемый при расчете стоимости создания заклинаний.

Значение по умолчанию: 7

Часть кода из OpenMW:

```
y = 0
for each effect in spell:
    x = 0.5 * (max(1, effect.magnitudeMin) + max(1, effect.magnitudeMax))
    x *= 0.1 * effect.magicEffect.baseMagickaCost
    x *= 1 + effect.duration
    x += 0.05 * max(1, effect.area) * effect.magicEffect.baseMagickaCost

    y += x * fEffectCostMult
    y = max(1, y)
    if effect.rangeType & CAST_TARGET: y *= 1.5

magickaCost = int(y)
cost of spellmaking = barterOffer(npc, magickaCost * fSpellMakingValueMult, buying)
```

fEnchantmentValueMult

fEnchantmentValueMult - множитель стоимости для создаваемых игроком магических предметов у NPC.

Значение по умолчанию: 1000

Часть кода из OpenMW:

```
y = 0
z = []
enchantPoints = 0

for each effect in enchantment:
    if enchantment is constant effect:
        if effect.magnitudeMin > 1 or effect.magnitudeMax > 1 or effect.radius > 1:
            effect.duration = int(fEnchantmentConstantDurationMult)
        else:
            effect.duration = 100

    x = 0.5 * (max(1, effect.magnitudeMin) + max(1, effect.magnitudeMax))
    x *= 0.1 * effect.magicEffect.baseMagickaCost
    x *= effect.duration # note difference from spellmaking
    x += 0.05 * max(1, spell.radius) * effect.magicEffect.baseMagickaCost

    y += x * fEffectCostMult
    y = max(1, y)
    if effect.rangeType & CAST_TARGET: y *= 1.5

    enchantPoints += int(y)
    z[effect.order] = int(pcEnchantSkill - y * fEnchantmentChanceMult)

# note enchantPoints not used for cost
cost of enchanting service = barterOffer(npc, y * fEnchantmentValueMult, buying)
```

fTravelMult

fTravelMult - определяет, какое количество золота требуется для оплаты услуг путешествия.

Значение **по умолчанию: 4000**

Часть кода из OpenMW:

```
dist = distance from player to destination  
cost = barterOffer(npc, int(dist / fTravelMult), buying)  
time = int(dist / fTravelTimeMult)
```

fTravelTimeMult

fTravelTimeMult - эта константа определяет, сколько времени пройдет в мире пока игрок путешествует.

Значение по умолчанию: **16000**

Известная формула:

$$x = (\text{Travel Distance} / \text{fTravelTimeMult})$$

fMagesGuildTravel

fMagesGuildTravel - определяет стоимость использования услуг телепортации в Магической гильдии.

Значение **по умолчанию: 10**

Часть кода из OpenMW:

```
cost = barterOffer(npc, fMagesGuildTravel, buying)
```


fRestMagicMult

fRestMagicMult - модификатор отвечающий за восстановление *запаса маны* при отдыхе.

Значение **по умолчанию: 0.15**

Код из OpenMW:

```
magicka += fRestMagicMult * intelligence
```

fWortChanceValue

fWortChanceValue - влияет на то когда игрок увидит скрытые знаком вопроса магические свойства ингредиентов, так же отвечает за вероятность того, что игрок получит первый эффект из съеденного ингредиента . Чем выше значение этой переменной, тем больше шансов на получение эффекта.

Значение **по умолчанию: 15**

При значении 15 игрок увидит первый эффект когда его навык алхимии будет равен или больше 15.

Часть кода из OpenMW:

Первый эффект: `pcAlchemy >= fWortChanceValue`
Второй эффект: `pcAlchemy >= 2 * fWortChanceValue`
Третий эффект: `pcAlchemy >= 3 * fWortChanceValue`
Четвертый эффект: `pcAlchemy >= 4 * fWortChanceValue`

fMinWalkSpeed

fMinWalkSpeed - это одна из переменных игры Morrowind, которая отвечает за минимальную скорость движения игрока и NPC пешком.

Значение **по умолчанию: 100**

Значение этой переменной указывает на минимальную скорость, которую может иметь игрок, когда он движется пешком. Если игрок пытается двигаться медленнее, чем это значение, его скорость будет автоматически увеличиваться до этого значения.

Например, если значение **fMinWalkSpeed** равно 100, то игрок не сможет двигаться медленнее, чем со скоростью 100 единиц. Если игрок попытается двигаться со скоростью например 90 единиц, его скорость автоматически увеличится до 100 единиц.

Известная формула:

$$x = (\text{fMinWalkSpeed} + \text{Speed} * (\text{fMaxWalkSpeed} - \text{fMinWalkSpeed}))$$

Speed - скорость игрока, NPC

fMaxWalkSpeed

fMaxWalkSpeed - это одна из переменных игры Morrowind, которая отвечает за максимальную скорость движения игрока и NPC пешком.

Значение **по умолчанию: 200**

Значение этой переменной указывает на максимальную скорость, которую может иметь игрок, когда он движется пешком. Если игрок пытается двигаться быстрее, чем это значение, его скорость будет автоматически уменьшена до этого значения.

Например, если значение **fMaxWalkSpeed** равно 200, то игрок не сможет двигаться быстрее, чем со скоростью 200 единиц. Если игрок попытается двигаться со скоростью например 210 единиц, его скорость автоматически уменьшится до 200 единиц.

Известная формула:

$$x = (fMinWalkSpeed + \textit{Speed} * (fMaxWalkSpeed - fMinWalkSpeed))$$

Speed - скорость игрока, NPC

fMinWalkSpeedCreature

fMinWalkSpeedCreature - это один из GMST, управляющий минимальной скоростью перемещения существ (Creature).

Значение по умолчанию: 5

Известная формула:

$$x = (\text{fMinWalkSpeedCreature} + \text{Speed} * (\text{fMaxWalkSpeedCreature} - \text{fMinWalkSpeedCreature}))$$

Speed - скорость существа

fMaxWalkSpeedCreature

fMaxWalkSpeedCreature - это один из GMST, управляющий максимальной скоростью перемещения существ (Creature).

Значение **по умолчанию**: 300

Известная формула:

$$x = (\text{fMinWalkSpeedCreature} + \text{Speed} * (\text{fMaxWalkSpeedCreature} - \text{fMinWalkSpeedCreature}))$$

Speed - скорость существа

fEncumberedMoveEffect

fEncumberedMoveEffect - отвечает за уменьшение скорости передвижения персонажа в зависимости от его перегрузки.

Значение по умолчанию: 0.3

Например, если **fEncumberedMoveEffect** установлено на 0, то персонаж, не будет терять скорость при загрузке инвентаря.

Creature не имеют этого параметра в расчете скорости движения, но они скорее всего могут быть перегружены и полностью обездвижены.

$Speed = (Basic\ Speed * (1 - normalizedEncumbrance * fEncumberedMoveEffect))$.

normalizedEncumbrance - отношение текущей загруженности к максимальной

fBaseRunMultiplier

fBaseRunMultiplier - константа которая определяет насколько быстро персонаж может бегать. Она устанавливает базовый коэффициент ускорения для бега, который затем используется для вычисления фактической скорости бега персонажа.

Значение по умолчанию: **1.75**

Этот параметр указывает множитель для скорости перемещения пешком, результатом будет скорость бега.

Известная формула:

Basic Run Speed = (*Basic Walk Speed* * **fBaseRunMultiplier** + *Athletic Skill* * *fAthleticsRunBonus*)

Athletic Skill - Атлетика

fAthleticsRunBonus

fAthleticsRunBonus - отвечает за модификатор скорости передвижения персонажа в зависимости от навыка *Атлетика*.

Значение по умолчанию: 1

Известная формула:

Basic Run Speed = (*Basic Walk Speed* * *fBaseRunMultiplier* + *Athletic Skill* * **fAthleticsRunBonus**)

Athletic Skill - Атлетика

fJumpAcrobaticsBase

fJumpAcrobaticsBase - это параметр который определяет некоторое базовое значение для высоты прыжка в зависимости от навыка [акробатика](#).

Значение **по умолчанию: 128**

Известная формула, скорее всего не точная.

Код из OpenMW:

```
if acrobaticsSkill <= 50:
    a = acrobaticsSkill, b = 0
else:
    a = 50, b = acrobaticsSkill - 50

x = fJumpAcrobaticsBase + pow(a / 15.0, fJumpAcroMultiplier)
x += 3 * b * fJumpAcroMultiplier
x += jumpSpellBonus * 64
x *= encumbranceTerm
if actor is running: x *= fJumpRunMultiplier
x *= fatigueTerm
x -= gravityAcceleration [constant; -627.2 exactly]
x /= 3
```

fJumpAcroMultiplier

fJumpAcroMultiplier - модификатор высоты прыжка зависящий от навыка [акробатики](#).

Значение **по умолчанию**: 4

Чем выше значение **fJumpAcroMultiplier**, тем дальше сможет прыгнуть персонаж при развитых значениях навыка Акробатика.

Известная формула, скорее всего не точная.

Код из OpenMW:

```
if acrobaticsSkill <= 50:  
    a = acrobaticsSkill, b = 0  
else:  
    a = 50, b = acrobaticsSkill - 50  
  
x = fJumpAcrobaticsBase + pow(a / 15.0, fJumpAcroMultiplier)  
x += 3 * b * fJumpAcroMultiplier  
x += jumpSpellBonus * 64  
x *= encumbranceTerm  
if actor is running: x *= fJumpRunMultiplier  
x *= fatigueTerm  
x -= gravityAcceleration [constant; -627.2 exactly]  
x /= 3
```

fJumpEncumbranceBase

fJumpEncumbranceBase - это один из параметров который отвечает за максимальный вес, который персонаж может нести и при этом осуществлять прыжки. Если персонаж переносит больший вес, чем указанный в fJumpEncumbranceBase, то высота и дальность прыжков уменьшается.

Значение по умолчанию: **0.5**

Код из OpenMW:

$encumbranceTerm = fJumpEncumbranceBase + fJumpEncumbranceMultiplier * (1 - normalizedEncumbrance)$

normalizedEncumbrance - отношение текущей загрузки к максимальной

fJumpEncumbranceMultiplier

fJumpEncumbranceMultiplier - является одной из констант, определяющих, как далеко может прыгнуть игрок в игре Morrowind при заданной нагрузке.

Значение по умолчанию: 1

Код из OpenMW:

$encumbranceTerm = fJumpEncumbranceBase + fJumpEncumbranceMultiplier * (1 - normalizedEncumbrance)$

normalizedEncumbrance - отношение текущей загрузки к максимальной

fJumpRunMultiplier

fJumpRunMultiplier - отвечает за то, насколько сильно бег влияет на дальность и высоту прыжка персонажа.

Значение **по умолчанию: 1**

Известная формула:

$x = (\text{Basic Jump Height} * \text{fJumpRunMultiplier})$.

fJumpMoveBase

fJumpMoveBase - определяет базовое расстояние, которое игрок может преодолеть с помощью одного прыжка. Эта константа влияет на максимальное расстояние, которое игрок может прыгнуть.

Значение по умолчанию: 0.5

fJumpMoveMult

fJumpMoveMult - определяет, как изменяется скорость передвижения персонажа при выполнении прыжка. Задаёт множитель для базовой скорости в прыжке персонажа, которая устанавливается с помощью константы fJumpMoveBase.

Значение по умолчанию: **0.5**

fSwimWalkBase

НЕ ИСПОЛЬЗУЕТСЯ В ОРИГИНАЛЬНОМ ДВИЖКЕ MORROWIND

fSwimWalkBase - множитель скорости передвижения персонажа при плавании в "режиме ходьбы".

Значение по умолчанию: **0.5**

fSwimRunBase

fSwimRunBase - множитель скорости персонажа при плавании.

Значение по умолчанию: 0.5

fSwimWalkAthleticsMult

НЕ ИСПОЛЬЗУЕТСЯ В ОРИГИНАЛЬНОМ ДВИЖКЕ MORROWIND

fSwimWalkAthleticsMult - определяет как способность Атлетика (Athletics) персонажа влияет на его скорость плавания в "режиме ходьбы".

Значение по умолчанию: **0.02**

fSwimRunAthleticsMult

fSwimRunAthleticsMult - контролирует как развитость навыка Атлетика будет влиять на скорость плавания.

Значение по умолчанию: 0.1

fSwimHeightScale

ТРЕБУЕТ ПРОВЕРКИ

fSwimHeightScale - множитель скорости определяющий насколько быстро персонаж всплывает и опускается находясь в воде.

Значение по умолчанию: **0.9**

fHoldBreathTime

fHoldBreathTime - константа которая контролирует, как долго персонаж может задерживать дыхание, погружаясь под воду. Измеряется в секундах.

Значение по умолчанию: 20

fHoldBreathEndMult

НЕ ИСПОЛЬЗУЕТСЯ В ОРИГИНАЛЬНОМ ДВИЖКЕ MORROWIND

fHoldBreathEndMult - множитель, который используется для вычисления времени, которое игрок может задерживать дыхание при погружении под воду в зависимости от развитости *выносливости*.

Значение по умолчанию: **0.5**

fSuffocationDamage

fSuffocationDamage - это значение которое определяет количество урона, который игрок получает, когда его персонаж задыхается под водой.

Значение **по умолчанию: 3**

fMinFlySpeed

fMinFlySpeed определяет минимальную скорость полета для существ и NPC.

Значение по умолчанию: 5

fMaxFlySpeed

fMaxFlySpeed - это значение максимальной скорости полета для существ и NPC.

Значение **по умолчанию: 300**

fStromWindSpeed

ТРЕБУЕТ ПРОВЕРКИ

fStromWindSpeed - модификатор скорости ветра.

Значение по умолчанию: **0.7**

fStromWalkMult

ТРЕБУЕТ ПРОВЕРКИ

fStromWalkMult - множитель для определения скорости перемещения персонажа во время бури.

Значение по умолчанию: **0.25**

fFallDamageDistanceMin

fFallDamageDistanceMin - константа которая определяет минимальное расстояние, с которого персонаж может упасть без получения урона.

Значение **по умолчанию: 400**

Код из OpenMW:

fallingDist = distance from peak height

if fallingDist <= **fFallDamageDistanceMin**: soft landing; skip the rest of the function

x = fallingDist - **fFallDamageDistanceMin**

x -= 1.5 * acrobaticsSkill + jumpSpellBonus

x = max(0, x)

a = fFallAcroBase + fFallAcroMult * (100 - acrobaticsSkill)

x = fFallDistanceBase + fFallDistanceMult * x

x *= a

if x > 0: damage health by x * (1 - 0.25 * fatigueTerm)

if acrobaticsSkill * fatigueTerm < x: actor falls over

if actor is not incapacitated: acrobatics skill exercised (skill gain from fall damage)

fFallDistanceBase

fFallDistanceBase - константа используется в формуле для получения урона зависящего от высоты падения и развитости навыка *акробатика*.

Значение **по умолчанию: 0**

Код из OpenMW:

fallingDist = distance from peak height

if fallingDist <= fFallDamageDistanceMin: soft landing; skip the rest of the function

x = fallingDist - fFallDamageDistanceMin

x -= 1.5 * acrobaticsSkill + jumpSpellBonus

x = max(0, x)

a = fFallAcroBase + fFallAcroMult * (100 - acrobaticsSkill)

x = **fFallDistanceBase** + fFallDistanceMult * x

x *= a

if x > 0: damage health by x * (1 - 0.25 * fatigueTerm)

if acrobaticsSkill * fatigueTerm < x: actor falls over

if actor is not incapacitated: acrobatics skill exercised (skill gain from fall damage)

fFallDistanceMult

fFallDamageMult - отвечает за множитель урона при падении на жесткую поверхность.

При падении персонаж теряет здоровье, пропорционально расстоянию падения, умноженному на fFallDamageMult. То есть, чем больше значение fFallDamageMult, тем больше урон персонаж получит при падении.

Значение по умолчанию: **0.07**

Код из OpenMW:

fallingDist = distance from peak height

if fallingDist <= fFallDamageDistanceMin: soft landing; skip the rest of the function

x = fallingDist - fFallDamageDistanceMin

x -= 1.5 * acrobaticsSkill + jumpSpellBonus

x = max(0, x)

a = fFallAcroBase + fFallAcroMult * (100 - acrobaticsSkill)

x = fFallDistanceBase + **fFallDistanceMult** * x

x *= a

if x > 0: damage health by x * (1 - 0.25 * fatigueTerm)

if acrobaticsSkill * fatigueTerm < x: actor falls over

if actor is not incapacitated: acrobatics skill exercised (skill gain from fall damage)

fFallAcroBase

fFallAcroBase - это параметр определяет на сколько меньше повреждения получит игрок при развитии навыке *акробатика*.

Значение **по умолчанию: 0.25**

Код из OpenMW:

fallingDist = distance from peak height

if fallingDist <= fFallDamageDistanceMin: soft landing; skip the rest of the function

x = fallingDist - fFallDamageDistanceMin

x -= 1.5 * acrobaticsSkill + jumpSpellBonus

x = max(0, x)

a = **fFallAcroBase** + fFallAcroMult * (100 - acrobaticsSkill)

x = fFallDistanceBase + fFallDistanceMult * x

x *= a

if x > 0: damage health by x * (1 - 0.25 * fatigueTerm)

if acrobaticsSkill * fatigueTerm < x: actor falls over

if actor is not incapacitated: acrobatics skill exercised (skill gain from fall damage)

fFallAcroMult

fFallAcroMult - это модификатор параметра определяющего на сколько меньше повреждения получит игрок при развитии навыке *акробатика*.

Значение **по умолчанию: 0.01**

Код из OpenMW:

fallingDist = distance from peak height

if fallingDist <= fFallDamageDistanceMin: soft landing; skip the rest of the function

x = fallingDist - fFallDamageDistanceMin

x -= 1.5 * acrobaticsSkill + jumpSpellBonus

x = max(0, x)

a = fFallAcroBase + **fFallAcroMult** * (100 - acrobaticsSkill)

x = fFallDistanceBase + fFallDistanceMult * x

x *= a

if x > 0: damage health by x * (1 - 0.25 * fatigueTerm)

if acrobaticsSkill * fatigueTerm < x: actor falls over

if actor is not incapacitated: acrobatics skill exercised (skill gain from fall dama

iMaxActivateDist

iMaxActivateDist - отвечает за максимальную дистанцию, на которой можно взаимодействовать с объектами в мире игры, такими как двери, контейнеры и переключатели.

Значение по умолчанию: **192**

iMaxInfoDist

iMaxInfoDist - параметр который определяет максимальную дистанцию, с которой игрок может получить информацию о предмете или персонаже.

Значение по умолчанию: **192**

fVanityDelay

fVanityDelay - отвечает за задержку в секундах между сменой точки обзора персонажа через указанное колличество секунд камера переключится на вид от третьего лица и начнет облетать вокруг персонажа.

Значение **по умолчанию: 30**

fMaxHeadTrackDistance

ТРЕБУЕТ ПРОВЕРКИ

fMaxHeadTrackDistance - значение которое определяет максимальную дистанцию, на которой персонажи будут следить поворотом головы за другими персонажами.

Значение по умолчанию: 400

fInteriorHeadTrackMult

fInteriorHeadTrackMult - модификатор для изменение дистанции на которой персонажи будут следить за игроком или NPC поворотом головы.

Значение по умолчанию: **0.5**

iHelmWeight

iHelmWeight - используется для определения категорий весов легкой, средней и тяжелой брони.

Значение **по умолчанию: 5**

iPauldronWeight

iPauldronWeight - используется для определения категорий весов легкой, средней и тяжелой брони.

Значение по умолчанию: 10

iCuirassWeight

iCuirassWeight - используется для определения категорий весов легкой, средней и тяжелой брони.

Значение по умолчанию: 30

iGauntletWeight

iGauntletWeight - используется для определения категорий весов легкой, средней и тяжелой брони.

Значение по умолчанию: 5

iGreavesWeight

iGreavesWeight - используется для определения категорий весов легкой, средней и тяжелой брони.

Значение по умолчанию: **15**

iBootsWeight

iBootsWeight - используется для определения категорий весов легкой, средней и тяжелой брони.

Значение по умолчанию: 20

iShieldWeight

iShieldWeight - используется для определения категорий весов легкой, средней и тяжелой брони.

Значение по умолчанию: **15**

fLightMaxMod

fLightMaxMod - модификатор веса который определяет легкую броню.

Расчитывается умножением **fLightMaxMod** например на *iHelmWeight* результат будет максимальным значением веса для легкой брони конкретно выбранной части доспеха.

Значение по умолчанию: **0.6**

Так же известна формула которая используется в OpenMW и судя по всему соответствует оригиналу. Из которой видно, все, что не относится к легкой (**fLightMaxMod**) или средней (**fMedMaxMod**) броне является тяжелой.

$\epsilon = 5e-4$

$referenceWeight = iHelmWeight, iPauldronWeight, iCuirassWeight, iGauntletWeight, iGreavesWeight, iBootsWeight, iShieldWeight$

if $itemWeight == 0$: return NONE

if $itemWeight \leq referenceWeight * fLightMaxMod + \epsilon$: return LIGHT

if $itemWeight \leq referenceWeight * fMedMaxMod + \epsilon$: return MEDIUM

else: return HEAVY

fMedMaxMod

fMedMaxMod - модификатор веса который определяет среднюю броню.

Расчитывается умножением **fMedMaxMod** например на *iHelmWeight* результат будет максимальным значением веса для средней брони конкретно выбранной части доспеха.

Значение по умолчанию: **0.9**

Так же известна формула которая используется в OpenMW и судя по всему соответствует оригиналу. Из которой видно, все, что не относится к легкой (*fLightMaxMod*) или средней (**fMedMaxMod**) броне является тяжелой.

epsilon = 5e-4

referenceWeight = *iHelmWeight*, *iPauldronWeight*, *iCuirassWeight*, *iGauntletWeight*, *iGreavesWeight*, *iBootsWeight*, *iShieldWeight*

if itemWeight == 0: *return NONE*

if itemWeight <= *referenceWeight* * *fLightMaxMod* + *epsilon*: *return LIGHT*

if itemWeight <= *referenceWeight* * **fMedMaxMod** + *epsilon*: *return MEDIUM*

else: *return HEAVY*

fUnarmoredBase1

fUnarmoredBase1 - это одна из констант которая отвечает за базовое значение защиты для умения *Бездоспешный бой* (Unarmored) для персонажа без доспехов.

Значение по умолчанию: 0.1

Так же известна формула которая используется в OpenMW и судя по всему соответствует оригиналу.

$$rating = (fUnarmoredBase1 * unarmoredSkill) * (fUnarmoredBase2 * unarmoredSkill)$$

fUnarmoredBase2

fUnarmoredBase2 - это одна из констант которая отвечает за базовое значение защиты для умения *Бездоспешный бой* (Unarmored) для персонажа без доспехов.

Значение по умолчанию: **0.065**

Так же известна формула которая используется в OpenMW и судя по всему соответствует оригиналу.

$$rating = (fUnarmoredBase1 * unarmoredSkill) * (fUnarmoredBase2 * unarmoredSkill)$$

iBaseArmorSkill

iBaseArmorSkill - при достижении какого либо из навыков владения броней значения указанного в **iBaseArmorSkill** "Уровень защиты" брони в инвентаре персонажа будет отображаться как тот, что записан в TES CS.

Значение по умолчанию: 30

fBlockSkillBonus

**ЖЕСТКО ЗАПРОГРАММИРОВАН В ОРИГИНАЛЬНОМ ДВИЖКЕ
MORROWIND, ЗНАЧЕНИЕ ИЗ GMST НЕ УЧИТЫВАЕТСЯ**

МСР ИМЕЕТ ПАТЧ ИСПРАВЛЕНИЕ

Работает в OpenMW

ТРЕБУЕТ ПРОВЕРКИ

fBlockSkillBonus - этот параметр дает бонус к шансу защититься если игрок движется в сторону или назад, перестает работать если игрок движется вперед.

Значение **по умолчанию: 1.25**

fDamageStrenghtBase

**ЖЕСТКО ЗАПРОГРАММИРОВАН В ОРИГИНАЛЬНОМ ДВИЖКЕ
MORROWIND, ЗНАЧЕНИЕ ИЗ GMST НЕ УЧИТЫВАЕТСЯ
МСП ИМЕЕТ ПАТЧ ИСПРАВЛЕНИЕ
Работает в OpenMW**

fDamageStrengthBase - отвечает за то, как сила игрока влияет на урон, наносимый оружием в том числе дальнего боя. Чем выше этот параметр, тем больший урон будет наносить игрок своим оружием при ударе или выстреле.

Значение **по умолчанию: 0.5**

Формула выглядит следующим образом

if attack is from a melee or ranged weapon:

*rawDamage *= 0.5 + 0.01 * actor.strength*

*rawDamage *= weapon.condition / weapon.maxCondition*

МСП использует следующую формулу

fDamageStrengthBase + 0.1 * fDamageStrengthMult * actor.strength

fDamageStrenghtMult

**ЖЕСТКО ЗАПРОГРАММИРОВАН В ОРИГИНАЛЬНОМ ДВИЖКЕ
MORROWIND, ЗНАЧЕНИЕ ИЗ GMST НЕ УЧИТЫВАЕТСЯ
МСП ИМЕЕТ ПАТЧ ИСПРАВЛЕНИЕ
Работает в OpenMW**

fDamageStrenghtMult - этот модификатор влияет на наносимый персонажами урон на основе их силы.

Значение **по умолчанию: 0.1**

Формула выглядит следующим образом

Если атака из оружия ближнего или дальнего боя:

*rawDamage *= 0.5 + 0.01 * actor.strength*

*rawDamage *= weapon.condition / weapon.maxCondition*

МСП использует следующую формулу

fDamageStrengthBase + 0.1 * fDamageStrengthMult * actor.strength

fSwingBlockBase

fSwingBlockBase - это параметр, отвечающий за базовую возможность блокирования щитом.

Значение **по умолчанию: 1**

Так же известна формула которая используется в OpenMW, судя по всему она близка к оригиналу.

Если игрок сбит с ног, нокаутирован, парализован, оглушен, находится в стойке заклинания или колдует: нет блока

theta = угол от игрока к врагу, отрицательный — противник слева от центральной линии, положительный — противник справа от центральной линии

*if theta < fCombatBlockLeftAngle: нет блока
if theta > fCombatBlockRightAngle: нет блока*

#Обратите внимание, что приведенные выше тесты неточны, поскольку каждое сравнение рассчитывается с использованием скалярного произведения. Который неправильно преобразуется в угол и немного отличается от указанного.

*blockTerm = pcBlockSkill + 0.2 * pcAgility + 0.1 * pcLuck
swingTerm = attackSwing * fSwingBlockMult + **fSwingBlockBase***

fatigueTerm = fFatigueBase - fFatigueMult(1 - normalisedFatigue)
where normalisedFatigue is a function of fatigue. empty fatigue bar -> 0.0, full fatigue bar -> 1.0
Note fatigueTerm is normally 1.25 at full fatigue.*

*playerTerm = blockTerm * swingTerm
if player is not moving forward at any speed: playerTerm = playerTerm * 1.25 (note this is not fBlockStillBonus)
playerTerm = playerTerm * fatigueTerm*

*Если npc существо: npcSkill = характеристики существ
В противном случае: npcSkill = навык NPC с оружием в руках*

*npcTerm = npcSkill + 0.2 * npcAgility + 0.1 * npcLuck
npcFatigueTerm = fFatigueBase - fFatigueMult*(1 - normalisedFatigue)
using NPC normalisedFatigue
npcTerm = npcTerm * npcFatigueTerm*

*x = int(playerTerm - npcTerm)
if x < iBlockMinChance: x = iBlockMinChance
if x > iBlockMaxChance: x = iBlockMaxChance
roll 100, block if roll < x*

если удар заблокирован, прочность щита уменьшается от входящего урона, смягчение урона не применяется.

fSwingBlockMult

fSwingBlockMult - модификатор для успешного парирования щитом.

Значение по умолчанию: 1

Так же известна формула которая используется в OpenMW, судя по всему она близка к оригиналу.

Если игрок сбит с ног, нокаутирован, парализован, оглушен, находится в стойке заклинания или колдует: нет блока

theta = угол от игрока к врагу, отрицательный — противник слева от центральной линии, положительный — противник справа от центральной линии

if theta < fCombatBlockLeftAngle: нет блока

if theta > fCombatBlockRightAngle: нет блока

#Обратите внимание, что приведенные выше тесты неточны, поскольку каждое сравнение рассчитывается с использованием скалярного произведения. Который неправильно преобразуется в угол и немного отличается от указанного.

*blockTerm = pcBlockSkill + 0.2 * pcAgility + 0.1 * pcLuck*

*swingTerm = attackSwing * **fSwingBlockMult** + fSwingBlockBase*

fatigueTerm = fFatigueBase - fFatigueMult(1 - normalisedFatigue)*

where normalisedFatigue is a function of fatigue. empty fatigue bar -> 0.0, full fatigue bar -> 1.0

Note fatigueTerm is normally 1.25 at full fatigue.

*playerTerm = blockTerm * swingTerm*

*if player is not moving forward at any speed: playerTerm = playerTerm * 1.25 (note this is not fBlockStillBonus)*

*playerTerm = playerTerm * fatigueTerm*

Если npc существо: npcSkill = характеристики существ

В противном случае: npcSkill = навык NPC с оружием в руках

*npcTerm = npcSkill + 0.2 * npcAgility + 0.1 * npcLuck*

npcFatigueTerm = fFatigueBase - fFatigueMult(1 - normalisedFatigue)*

using NPC normalisedFatigue

*npcTerm = npcTerm * npcFatigueTerm*

x = int(playerTerm - npcTerm)

if x < iBlockMinChance: x = iBlockMinChance

if x > iBlockMaxChance: x = iBlockMaxChance

roll 100, block if roll < x

если удар заблокирован, прочность щита уменьшается от входящего урона, смягчение урона не применяется.

fFatigueBase

fFatigueBase - константа которая определяет базовый уровень усталости, используемый для всех персонажей в игре.

Значение по умолчанию: **1.2500**

Усталость является одним из важнейших параметров, который влияет на множество атрибутов и навыков персонажа, включая скорость передвижения, вероятность успешной атаки, способность использовать магию, и многое другое.

Так же известна формула которая используется в OpenMW, судя по всему она близка или идентична оригиналу.

fatigueTerm = **fFatigueBase** - fFatigueMult*(1 - *normalisedFatigue*)

fatigueTerm используется во многих формулах, например при убеждении, ремонте вещей, воровстве и т.д.

normalisedFatigue является функцией утомления. пустая шкала усталости -> 0,0, полная полоса усталости -> 1,0

fFatigueMult

fFatigueMult - множитель константы которая определяет базовый уровень усталости, используемый для всех персонажей в игре.

Значение **по умолчанию: 0.5**

Усталость является одним из важнейших параметров, который влияет на множество атрибутов и навыков персонажа, включая скорость передвижения, вероятность успешной атаки, способность использовать магию, и многое другое.

Так же известна формула которая используется в OpenMW, судя по всему она близка или идентична оригиналу.

fatigueTerm = *fFatigueBase* - **fFatigueMult***(1 - *normalisedFatigue*)

fatigueTerm используется во многих формулах, например при убеждении, ремонте вещей, воровстве и т.д.

normalisedFatigue является функцией утомления. пустая шкала усталости -> 0,0, полная полоса усталости -> 1,0

fFatigueReturnBase

fFatigueReturnBase - определяет базовое значение восстановления выносливости (fatigue), который персонажи получают в покое.

Значение **по умолчанию: 2.5**

Значение fFatigueReturnBase указывает, сколько запаса сил восстанавливается каждый кадр, когда игрок находится в покое (не выполняет никаких действий).

Так же известна формула которая используется в OpenMW, судя по всему она близка или идентична оригиналу.

#Восстанавливается каждый кадр

if currentFatigue < maxFatigue:

$x = \mathbf{fFatigueReturnBase} + \mathbf{fFatigueReturnMult} * \mathbf{endurance}$

*recover (frameTime * x) fatigue*

fFatigueReturnMult

fFatigueReturnMult - является модификатором который влияет на восстановление запаса сил с учетом параметра **выносливости** (*endurance*).

Значение **по умолчанию: 0.02**

Так же известна формула которая используется в OpenMW, судя по всему она близка или идентична оригиналу.

#Восстанавливается каждый кадр

if currentFatigue < maxFatigue:

$x = fFatigueReturnBase + fFatigueReturnMult * endurance$

*recover (frameTime * x) fatigue*

fEndFatigueMult

fEndFatigueMult - множитель который используется для восстановления запаса сил в режиме отжидания и сна.

Значение по умолчанию: **0.04**

Так же известна формула которая используется в OpenMW, судя по всему она близка или идентична оригиналу.

за каждый час для каждого актера во всем мире:

$$x *= \mathbf{fEndFatigueMult} * endurance$$

fFatigueAttackBase

fFatigueAttackBase - это параметр определяет, насколько гарантировано уменьшится Запас сил персонажа при каждой атаке оружием.

Значение **по умолчанию: 2**

Известная формула расчета для запаса сил:

$$\text{Attack Fatigue Cost} = (\text{fFatigueAttackBase} + (\text{normalizedEncumbrance} * \text{fFatigueAttackMult}) + \text{weapon weight} * (\text{fWeaponFatigueMult} * \text{weapon windup})).$$

weapon weight - вес оружия

weapon windup - степень замаха от 0.0 до 1.0

normalizedEncumbrance - отношение текущей загруженности к максимальной

fFatigueAttackMult

fFatigueAttackMult - этот множитель отвечает за дополнительный расход Запаса сил при атаке в зависимости от загруженности инвентаря.

Значение **по умолчанию: 0**

Известная формула расчета для запаса сил:

$$\text{Attack Fatigue Cost} = (\text{fFatigueAttackBase} + (\text{normalizedEncumbrance} * \text{fFatigueAttackMult}) + \text{weapon weight} * (\text{fWeaponFatigueMult} * \text{weapon windup})).$$

weapon weight - вес оружия

weapon windup - степень замаха от 0.0 до 1.0

normalizedEncumbrance - отношение текущей загруженности к максимальной

fWeaponFatigueMult

fWeaponFatigueMult - этот множитель отвечает за дополнительный расход Запаса сил в зависимости от веса используемого оружия, а так же от силы замаха оружием.

Значение по умолчанию: **0.25**

Известная формула расчета для запаса сил:

$$\text{Attack Fatigue Cost} = (\text{fFatigueAttackBase} + (\text{normalizedEncumbrance} * \text{fFatigueAttackMult}) + \text{weapon weight} * (\text{fWeaponFatigueMult} * \text{weapon windup})).$$

weapon weight - вес оружия

weapon windup - степень замаха от 0.0 до 1.0

normalizedEncumbrance - отношение текущей загрузки к максимальной

fFatigueBlockBase

fFatigueBlockBase - это параметр определяет, насколько гарантировано уменьшится Запас сил персонажа при блокировании удара щитом.

Значение **по умолчанию: 4**

Известная формула расчета для запаса сил:

$$\text{Block Fatigue Cost} = (\text{fFatigueBlockBase} + (\text{normalizedEncumbrance} * \text{fFatigueBlockMult}) + \text{Weapon Weight} * (\text{fWeaponFatigueBlockMult} * \text{weapon windup})).$$

weapon weight - вес оружия

weapon windup - степень замаха от 0.0 до 1.0

normalizedEncumbrance - отношение текущей загруженности к максимальной

fFatigueBlockMult

fFatigueBlockMult - этот множитель отвечает за дополнительный расход Запаса сил при блокировании щитом в зависимости от загруженности инвентаря.

Значение **по умолчанию: 0**

Известная формула расчета для запаса сил:

$$\text{Block Fatigue Cost} = (\text{fFatigueBlockBase} + (\text{normalizedEncumbrance} * \text{fFatigueBlockMult}) + \text{Weapon Weight} * (\text{fWeaponFatigueBlockMult} * \text{weapon windup})).$$

weapon weight - вес оружия

weapon windup - степень замаха от 0.0 до 1.0

normalizedEncumbrance - отношение текущей загруженности к максимальной

fWeaponFatigueBlockMult

fWeaponFatigueBlockMult - этот множитель отвечает за дополнительный расход Запаса сил в зависимости от веса используемого оружия, а так же степени замаха.

Значение по умолчанию: 1

Известная формула расчета для запаса сил:

$$\text{Block Fatigue Cost} = (\text{fFatigueBlockBase} + (\text{normalizedEncumbrance} * \text{fFatigueBlockMult}) + \text{Weapon Weight} * (\text{fWeaponFatigueBlockMult} * \text{weapon windup})).$$

weapon weight - вес оружия

weapon windup - степень замаха от 0.0 до 1.0

normalizedEncumbrance - отношение текущей загрузки к максимальной

fFatigueRunBase

fFatigueRunBase - это параметр определяет базовое значение усталости, которое тратится при беге.

Значение **по умолчанию: 5**

Известная формула расчета для запаса сил:

Action Fatigue Cost = (**fFatigueRunBase** + *normalizedEncumbrance* * *fFatigueRunMult*).

normalizedEncumbrance - отношение текущей загрузки к максимальной

fFatigueRunMult

fFatigueRunMult - модификатор определяющий расход запаса сил при беге в зависимости от загруженности инвентаря.

Значение **по умолчанию**: 2

Известная формула расчета для запаса сил:

Action Fatigue Cost = (*fFatigueRunBase* + *normalizedEncumbrance* * *fFatigueRunMult*).

normalizedEncumbrance - отношение текущей загруженности к максимальной

fFatigueJumpBase

fFatigueJumpBase - это параметр определяет базовое значение усталости, которое тратится при прыжке.

Значение по умолчанию: 5

Известная формула расчета для запаса сил:

Action Fatigue Cost = (**fFatigueJumpBase** + *normalizedEncumbrance* * *fFatigueJumpMult*).

normalizedEncumbrance - отношение текущей загрузки к максимальной

fFatigueJumpMult

fFatigueJumpMult - модификатор определяющий расход запаса сил при прыжке в зависимости от загруженности инвентаря.

Значение по умолчанию: 0

Известная формула расчета для запаса сил:

Action Fatigue Cost = (*fFatigueJumpBase* + *normalizedEncumbrance* * *fFatigueJumpMult*).

normalizedEncumbrance - отношение текущей загруженности к максимальной

fFatigueSwimWalkBase

fFatigueSwimWalkBase - это параметр определяет базовое значение усталости, которое тратится при плавании в режиме ходьбы.

Значение по умолчанию: **2.5**

Известная формула расчета для запаса сил:

Action Fatigue Cost = (**fFatigueSwimWalkBase** + *normalizedEncumbrance* * *fFatigueSwimWalkMult*).

normalizedEncumbrance - отношение текущей загрузки к максимальной

fFatigueSwimRunBase

fFatigueSwimRunBase - это параметр определяет базовое значение усталости, которое тратится при плавании в режиме бега.

Значение **по умолчанию**: 7

Известная формула расчета для запаса сил:

Action Fatigue Cost = (**fFatigueSwimRunBase** + *normalizedEncumbrance* * *fFatigueSwimRunMult*).

normalizedEncumbrance - отношение текущей загрузки к максимальной

fFatigueSwimWalkMult

fFatigueSwimWalkMult - модификатор определяющий расход запаса сил при прыжке в зависимости от загрузки инвентаря.

Значение по умолчанию: 0

Известная формула расчета для запаса сил:

Action Fatigue Cost = (*fFatigueSwimWalkBase* + *normalizedEncumbrance* * *fFatigueSwimWalkMult*).

normalizedEncumbrance - отношение текущей загрузки к максимальной

fFatigueSwimRunMult

fFatigueSwimRunMult - модификатор определяющий расход запаса сил при прыжке в зависимости от загрузки инвентаря.

Значение по умолчанию: 0

Известная формула расчета для запаса сил:

Action Fatigue Cost = (*fFatigueSwimRunBase* + *normalizedEncumbrance* * *fFatigueSwimRunMult*).

normalizedEncumbrance - отношение текущей загрузки к максимальной

fFatigueSneakBase

fFatigueSneakBase - это параметр определяет базовое значение усталости, которое тратится в режиме скрытности.

Значение по умолчанию: 1.5

Известная формула расчета для запаса сил:

Action Fatigue Cost = (**fFatigueSneakBase** + *normalizedEncumbrance* * *fFatigueSneakMult*).

normalizedEncumbrance - отношение текущей загрузки к максимальной

fFatigueSneakMult

fFatigueSneakMult - модификатор определяющий расход запаса сил в режиме скрытности в зависимости от загруженности инвентаря.

Значение по умолчанию: **1.5**

Известная формула расчета для запаса сил:

Action Fatigue Cost = (fFatigueSneakBase + *normalizedEncumbrance* * **fFatigueSneakMult**).

normalizedEncumbrance - отношение текущей загруженности к максимальной

fMinHandToHandMult

fMinHandToHandMult - устанавливает минимальное значение множителя урона для рукопашного боя. Когда игрок использует атаку без оружия "Hand-to-Hand", его урон определяется на основе значения базового урона и множителя урона, установленного этой константой.

Значение по умолчанию: 0.1

Известная формула расчета :

Basic Damage = (**fMinHandToHandMult** + *weapon windup* * (*fMaxHandToHandMult* - **fMinHandToHandMult**)).

Basic Fatigue Damage = (*Basic Damage* * *HandToHand Skill*).

Basic Health Damage = (*Basic Damage* * *HandToHand Skill* * *fHandtoHandHealthPer*).

weapon windup - степень замаха от 0.0 до 1.0

HandToHand Skill - уровень навыка Рукопашный бой

fMaxHandToHandMult

fMaxHandToHandMult - определяет максимальный множитель для урона при ближнем бою без оружия "Hand-to-Hand".

Значение по умолчанию: 0.5

Известная формула расчета :

Basic Damage = (*fMinHandToHandMult* + *weapon windup* * (*fMaxHandToHandMult* - *fMinHandToHandMult*)).

Basic Fatigue Damage = (*Basic Damage* * *HandToHand Skill*).

Basic Health Damage = (*Basic Damage* * *HandToHand Skill* * *fHandtoHandHealthPer*).

weapon windup - степень замаха от 0.0 до 1.0

HandToHand Skill - уровень навыка Рукопашный бой

fHandToHandHealthPer

fHandToHandHealthPer - множитель который определяет урон наносимый по здоровью противника для урона в ближнем бою без оружия "Hand-to-Hand".

Значение по умолчанию: 0.1

Известная формула расчета :

Basic Damage = (*fMinHandToHandMult* + *weapon windup* * (*fMaxHandToHandMult* - *fMinHandToHandMult*)).

Basic Fatigue Damage = (*Basic Damage* * *HandToHand Skill*).

Basic Health Damage = (*Basic Damage* * *HandToHand Skill* * **fHandtoHandHealthPer**).

weapon windup - степень замаха от 0.0 до 1.0

HandToHand Skill - уровень навыка Рукопашный бой

fCombatInvisoMult

fCombatInvisoMult - модификатор использующийся в формуле шанса успешного попадания в противника под действием хамелиона или невидимости

Значение по умолчанию: **0.2**

Известная формула расчета :

```
defenseTerm += min(100, fCombatInvisoMult * defender.chameleon)  
defenseTerm += min(100, fCombatInvisoMult * defender.invisibility)
```


fCombatKODamageMult

fCombatKODamageMult - множитель для урона по цели когда она накаутирована.

Значение по умолчанию: **1.5**

fCombatCriticalStrikeMult

fCombatCriticalStrikeMult - константа которая отвечает за множитель критического удара в бою. Когда игрок атакует противника который его не видит, у него есть шанс нанести критический удар, который увеличит урон, наносимый врагу.

Значение по умолчанию: **4**

iBlockMinChance

iBlockMinChance - это параметр который определяет минимальный шанс на успешный блок атаки противника при использовании щита.

Значение по умолчанию: 10

iBlockMaxChance

iBlockMaxChance - это параметр который определяет максимальный шанс на успешный блок атаки противника при использовании щита.

Значение по умолчанию: 50

fLevelUpHealthEndMult

fLevelUpHealthEndMult - отвечает за то, как много здоровья игрок получит при повышении уровня в зависимости от его характеристики Выносливость (Endurance).

Значение по умолчанию: 0.1

$$\text{bonusHP} = \text{fLevelUpHealthEndMult} * \text{baseEndurance}$$

fSoulGemMult

fSoulGemMult - это множитель, который определяет, сколько единиц души может быть сохранено в камне душ, в зависимости от его стоимости.

Значение **по умолчанию**: 3

Soulgem Capacity = (**Price** * **fSoulGemMult**), с округлением в меньшую сторону.

Price - стоимость камня душ

fEffectCostMult

fEffectCostMult - множитель, используемый для расчета стоимости магических эффектов при создании заклинаний и зачарований.

Значение по умолчанию: 0.5

fSpellPriceMult

НЕ ИСПОЛЬЗУЕТСЯ В ОРИГИНАЛЬНОМ ДВИЖКЕ MORROWIND
ТРЕБУЕТ ПРОВЕРКИ

fSpellPriceMult - параметр который контролирует стоимость заклинаний в магазинах. Определяет, насколько дорогим будет изучение заклинания.

Значение по умолчанию: 2

fFatigueSpellBase

fFatigueSpellBase - определяет базовое количество усталости которое отнимается при произнесении заклинания.

Значение **по умолчанию: 0**

Известная формула из OpenMW.

Spell Fatigue Cost = magickaCost * (fFatigueSpellBase * **normalizedEncumbrance** * fFatigueSpellMult)

normalizedEncumbrance - отношение текущей загрузки к максимальной

fFatigueSpellMult

fFatigueSpellMult - множитель для усталости которая отнимается при произнесении заклинания с зависимостью от загруженности.

Значение **по умолчанию: 0**

Известная формула из OpenMW.

Spell Fatigue Cost = magickaCost * (fFatigueSpellBase * **normalizedEncumbrance** * **fFatigueSpellMult**)

normalizedEncumbrance - отношение текущей загруженности к максимальной

fFatigueSpellCostMult

НЕ ИСПОЛЬЗУЕТСЯ В ОРИГИНАЛЬНОМ ДВИЖКЕ MORROWIND И OPENMW

fFatigueSpellCostMult - множитель для затрат запаса сил при произнесении заклинания с зависимостью от стоимости заклинания.

Значение по умолчанию: 0

fPotionStrengthMult

fPotionStrengthMult - множитель для изготавливаемых игроком зелий. Отвечает за параметры силы, продолжительности и стоимости.

Значение **по умолчанию: 0.5**

Известная части кода в OpenMW.

$x = pcAlchemy + 0.1 * pcIntelligence + 0.1 * pcLuck$
roll 100 vs x, succeed if roll <= x

$x *= mortarQuality * \mathbf{fPotionStrengthMult}$

$magnitude = (x / fPotionT1MagMult) / magicEffect.baseMagickaCost$

$duration = (x / fPotionT1DurMult) / magicEffect.baseMagickaCost$

$price = int(iAlchemyMod * x)$

$weight = \text{sum of ingredient weights} / \text{total ingredients added}$

$visual = \text{roll } \{ 'Bargain', 'Cheap', 'Exclusive', 'Fresh', 'Quality', 'Standard' \}$

fPotionT1MagMult

fPotionT1MagMult - отвечает за множитель силы эликсиров которые изготавливает игрок.

Значение по умолчанию: 1.5

Известная части кода в OpenMW.

```
x = pcAlchemy + 0.1 * pcIntelligence + 0.1 * pcLuck  
roll 100 vs x, succeed if roll <= x
```

```
x *= mortarQuality * fPotionStrengthMult  
magnitude = (x / fPotionT1MagMult) / magicEffect.baseMagickaCost  
duration = (x / fPotionT1DurMult) / magicEffect.baseMagickaCost
```

```
price = int(iAlchemyMod * x)  
weight = sum of ingredient weights / total ingredients added  
visual = roll { 'Bargain', 'Cheap', 'Exclusive', 'Fresh', 'Quality', 'Standard' }
```

fPotionT1DurMult

fPotionT1DurMult - это параметр определяет множитель продолжительности эффектов зелий.

Значение **по умолчанию: 0.5**

Известная части кода в OpenMW.

```
x = pcAlchemy + 0.1 * pcIntelligence + 0.1 * pcLuck  
roll 100 vs x, succeed if roll <= x
```

```
x *= mortarQuality * fPotionStrengthMult  
magnitude = (x / fPotionT1MagMult) / magicEffect.baseMagickaCost  
duration = (x / fPotionT1DurMult) / magicEffect.baseMagickaCost
```

```
price = int(iAlchemyMod * x)  
weight = sum of ingredient weights / total ingredients added  
visual = roll { 'Bargain', 'Cheap', 'Exclusive', 'Fresh', 'Quality', 'Standard' }
```

fPotionMinUsefulDuration

НЕ ИСПОЛЬЗУЕТСЯ В ОРИГИНАЛЬНОМ ДВИЖКЕ MORROWIND И OPENMW

fPotionMinUsefulDuration -

Значение по умолчанию: 20

fPotionT4BaseStrengthMult

НЕ ИСПОЛЬЗУЕТСЯ В ОРИГИНАЛЬНОМ ДВИЖКЕ MORROWIND И OPENMW

fPotionT4BaseStrengthMult -

Значение по умолчанию: 20

fPotionT4EquipStrengthMult

НЕ ИСПОЛЬЗУЕТСЯ В ОРИГИНАЛЬНОМ ДВИЖКЕ MORROWIND И OPENMW

fPotionT4EquipStrengthMult -

Значение по умолчанию: 12

fIngredientMult

**НЕ ИСПОЛЬЗУЕТСЯ В ОРИГИНАЛЬНОМ ДВИЖКЕ MORROWIND И OPENMW
ТРЕБУЕТ ПРОВЕРКИ**

fIngredientMult -

Значение по умолчанию: 1

fMagicItemCostMult

НЕ ИСПОЛЬЗУЕТСЯ В ОРИГИНАЛЬНОМ ДВИЖКЕ MORROWIND И OPENMW
ТРЕБУЕТ ПРОВЕРКИ

fMagicItemCostMult -

Значение по умолчанию: 1

fMagicItemPriceMult

**НЕ ИСПОЛЬЗУЕТСЯ В ОРИГИНАЛЬНОМ ДВИЖКЕ MORROWIND И OPENMW
ТРЕБУЕТ ПРОВЕРКИ**

fMagicItemPriceMult -

Значение по умолчанию: 1

fMagicItemOnceMult

**НЕ ИСПОЛЬЗУЕТСЯ В ОРИГИНАЛЬНОМ ДВИЖКЕ MORROWIND И OPENMW
ТРЕБУЕТ ПРОВЕРКИ**

fMagicItemOnceMult -

Значение по умолчанию: 1

fMagicItemUsedMult

**НЕ ИСПОЛЬЗУЕТСЯ В ОРИГИНАЛЬНОМ ДВИЖКЕ MORROWIND И OPENMW
ТРЕБУЕТ ПРОВЕРКИ**

fMagicItemUsedMult -

Значение по умолчанию: 1

fMagicItemStrikeMult

**НЕ ИСПОЛЬЗУЕТСЯ В ОРИГИНАЛЬНОМ ДВИЖКЕ MORROWIND И OPENMW
ТРЕБУЕТ ПРОВЕРКИ**

fMagicItemStrikeMult -

Значение по умолчанию: 1

fMagicItemConstantMult

**НЕ ИСПОЛЬЗУЕТСЯ В ОРИГИНАЛЬНОМ ДВИЖКЕ MORROWIND И OPENMW
ТРЕБУЕТ ПРОВЕРКИ**

fMagicItemConstantMult -

Значение по умолчанию: 1

fEnchantmentMult

fEnchantmentMult - множитель для расчета магической емкости для вещей.

Значение по умолчанию: 0.1

Actual item Enchantment Capacity = (Enchantment Capacity * fEnchantmentMult).

Enchantment Capacity - магическая емкость вещи выставленная в TES CS

fEnchantmentChanceMult

ТРЕБУЕТ ПРОВЕРКИ

fEnchantmentChanceMult - определяет множитель шанса успеха при создании предметов с эффектами.

Значение по умолчанию: 3

Известная формула:

$$\text{Enchant Chance} = ((\text{pcEnchantSkill} + \text{INT} * 0.2 + \text{LUC} * 0.1) - (\text{fEffectCostMult} * \text{fEnchantmentChanceMult})).$$

pcEnchantSkill - навык зачарования

INT - интеллект

LUC - удача

fEffectCostMult - итоговая стоимость

Так же есть фрагмент кода из OpenMW

y = 0

z = []

enchantPoints = 0

for each effect in enchantment:

if enchantment is constant effect:

if effect.magnitudeMin > 1 or effect.magnitudeMax > 1 or effect.radius > 1:

effect.duration = int(fEnchantmentConstantDurationMult)

else:

effect.duration = 100

x = 0.5 * (max(1, effect.magnitudeMin) + max(1, effect.magnitudeMax))

x *= 0.1 * effect.magicEffect.baseMagickaCost

x *= effect.duration # note difference from spellmaking

x += 0.05 * max(1, spell.radius) * effect.magicEffect.baseMagickaCost

y += *x* * fEffectCostMult

y = max(1, *y*)

if effect.rangeType & CAST_TARGET: *y* *= 1.5

enchantPoints += int(*y*)

z[effect.order] = int(pcEnchantSkill - *y* * fEnchantmentChanceMult)

note enchantPoints not used for cost

cost of enchanting service = barterOffer(npc, *y* * fEnchantmentValueMult, buying)

fPCbaseMagickaMult

fPCbaseMagickaMult - модификатор базового запаса маны для игрока.

Значение по умолчанию: 1

Известная формула:

baseMagicka for PC = (*baseActor.intelligence* * *fPCbaseMagickaMult*).

baseActor.intelligence - интеллект

fNPCbaseMagickaMult

fNPCbaseMagickaMult - модификатор базового запаса маны для NPC.

Значение по умолчанию: **2**

Известная формула:

baseMagicka for NPC = (*baseActor.intelligence* * **fNPCbaseMagickaMult**).

baseActor.intelligence - интеллект

fAutoSpellChance

fAutoSpellChance - минимальный шанс удачного каста заклинания, который необходим для того чтобы это заклинание получил NPC когда для него включен Auto-Calc. Таким образом, если установлено значение 80, NPC будет иметь минимум 80% шанс на успешное произнесение тех заклинаний, что они получают.

Значение **по умолчанию: 80**

Часть кода из OpenMW:

```
baseActor.spells = vector()
baseMagicka = fNPCbaseMagickaMult * baseActor.intelligence

spellSchools = { Alteration, Conjuration, Destruction, Illusion, Mysticism, Restoration }
schoolCaps = {} # could be an array indexed by school enum

for each school in spellSchools:
    schoolCaps[school] = { count : 0,
                          limit : iAutoSpell{school}Max,
                          reachedLimit : iAutoSpell{school}Max <= 0,
                          minCost : INT_MAX,
                          weakestSpell : none }

for each spell in the game: # note: iteration order is important, see comments
    if spell.isMarkedDeleted: continue
    if spell.castingType != spell: continue
    if not spell.isAutoCalculate: continue

    if baseMagicka < iAutoSpellTimesCanCast * spell.cost: continue
    if spell is in baseActor.race.racialSpells: continue

    failedAttrSkillCheck = false
    for each effect in spell:
        if (effect.baseEffect.flags & TARGET_SKILL) and baseActor.skills[effect.targetSkill] < iAutoSpellAttSkillMin:
            failedAttrSkillCheck = true
            break
        if (effect.baseEffect.flags & TARGET_ATTR) and baseActor.attribute[effect.targetAttr] <
iAutoSpellAttSkillMin:
            failedAttrSkillCheck = true
            break
    if failedAttrSkillCheck: continue

    school, _ = calcWeakestSchool(spell, actor)
    cap = schoolCaps[school]

    if cap.reachedLimit and spell.cost <= cap.minCost: continue
    if calcBaseCastChance(baseActor, spell, school) < fAutoSpellChance: continue

    baseActor.spells.add(spell)

    if cap.reachedLimit:
        baseActor.spells.remove(cap.weakestSpell)
        cap.weakestSpell = baseActor.spells.findMinCostSpell() # note: not school specific
        cap.minCost = cap.weakestSpell.cost
    else:
        cap.count += 1
```

```
if cap.count == cap.limit:  
    cap.reachedLimit = true  
  
if spell.cost < cap.minCost:  
    cap.weakestSpell = spell  
    cap.minCost = spell.cost
```

baseMagicka - запас маны для NPC

iAutoSpell{school}Max - лимит по максимальному кол-ву заклинаний для каждой из школ.

iAutoSpellTimesCanCast - на какое минимальное количество кастов заклинания должно хватить маны у игрока или NPC для того чтобы получить его как стартовое или при расчете Auto-Calc

fAutoPCSpellChance

fAutoPCSpellChance - минимальный шанс удачного каста заклинания, который необходим для того чтобы это заклинание получил Игрок как стартовое. Таким образом, если установлено значение 50, PC будет иметь минимум 50% шанс на успешное произнесение тех заклинаний, что он получит.

Значение **по умолчанию: 50**

Часть кода из OpenMW:

```
baseMagicka = fPCbaseMagickaMult * baseActor.intelligence
reachedLimit = false
weakestSpell = none
minCost = INT_MAX

baseActor.spells = vector()

for each spell in the game: # note: iteration order is important, see comments
    if spell.isMarkedDeleted: continue
    if spell.castingType != spell: continue
    if not spell.isPCStartSpell: continue

    if reachedLimit and spell.cost <= minCost: continue
    if spell is in baseActor.spells: continue
    if spell is in baseActor.race.racialSpells: continue
    if baseMagicka < spell.cost: continue
    if calcAutoCastChance(spell, baseActor, none) < fAutoPCSpellChance: continue

    failedAttrSkillCheck = false
    for each effect in spell:
        if (effect.baseEffect.flags & TARGET_SKILL) and baseActor.skills[effect.targetSkill] < iAutoSpellAttSkillMin:
            failedAttrSkillCheck = true
            break
        if (effect.baseEffect.flags & TARGET_ATTR) and baseActor.attribute[effect.targetAttr] <
iAutoSpellAttSkillMin:
            failedAttrSkillCheck = true
            break
    if failedAttrSkillCheck: continue

    baseActor.spells.add(spell)

    if reachedLimit:
        baseActor.spells.remove(weakestSpell)
        weakestSpell = baseActor.spells.findMinCostSpell()
        minCost = weakestSpell.cost
    else:
        if spell.cost < minCost:
            weakestSpell = spell
            minCost = spell.cost
        if baseActor.spells.size() == iAutoPCSpellMax:
            reachedLimit = true
```

baseMagicka - запас маны для PC

iAutoPCSpellMax - Максимальное количество заклинаний которые игрок может получить как стартовые.

iAutoSpellTimesCanCast

iAutoSpellTimesCanCast - на какое минимальное количество кастов заклинания должно хватить маны у NPC для того чтобы получить его как стартовое или при расчете Auto-Calc

Значение **по умолчанию**: 3

Часть кода из OpenMW:

iAutoSpell{school}Max - лимит по максимальному кол-ву заклинаний для каждой из школ.

iAutoSpellTimesCanCast - на какое минимальное количество кастов заклинания должно хватить маны у NPC для того чтобы получить его как стартовое или при расчете Auto-Calc

```
baseActor.spells = vector()
```

```
baseMagicka = fNPCbaseMagickaMult * baseActor.intelligence
```

```
spellSchools = { Alteration, Conjuraton, Destruction, Illusion, Mysticism, Restoration }
```

```
schoolCaps = {} # could be an array indexed by school enum
```

```
for each school in spellSchools:
```

```
    schoolCaps[school] = { count : 0,  
                          limit : iAutoSpell{school}Max,  
                          reachedLimit : iAutoSpell{school}Max <= 0,  
                          minCost : INT_MAX,  
                          weakestSpell : none }
```

```
for each spell in the game: # note: iteration order is important, see comments
```

```
    if spell.isMarkedDeleted: continue
```

```
    if spell.castingType != spell: continue
```

```
    if not spell.isAutoCalculate: continue
```

```
    if baseMagicka < iAutoSpellTimesCanCast * spell.cost: continue
```

```
    if spell is in baseActor.race.racialSpells: continue
```

```
    failedAttrSkillCheck = false
```

```
    for each effect in spell:
```

```
        if (effect.baseEffect.flags & TARGET_SKILL) and baseActor.skills[effect.targetSkill] < iAutoSpellAttSkillMin:
```

```
            failedAttrSkillCheck = true
```

```
            break
```

```
        if (effect.baseEffect.flags & TARGET_ATTR) and baseActor.attribute[effect.targetAttr] <
```

```
        iAutoSpellAttSkillMin:
```

```
            failedAttrSkillCheck = true
```

```
            break
```

```
    if failedAttrSkillCheck: continue
```

```
    school, _ = calcWeakestSchool(spell, actor)
```

```
    cap = schoolCaps[school]
```

```
    if cap.reachedLimit and spell.cost <= cap.minCost: continue
```

```
    if calcBaseCastChance(baseActor, spell, school) < fAutoSpellChance: continue
```

```
    baseActor.spells.add(spell)
```

```
    if cap.reachedLimit:
```

```
        baseActor.spells.remove(cap.weakestSpell)
```

```
        cap.weakestSpell = baseActor.spells.findMinCostSpell() # note: not school specific
```



```
    cap.minCost = cap.weakestSpell.cost
else:
    cap.count += 1
    if cap.count == cap.limit:
        cap.reachedLimit = true

    if spell.cost < cap.minCost:
        cap.weakestSpell = spell
        cap.minCost = spell.cost
```

baseMagicka - запас маны для NPC

iAutoSpellAttSkillMin

iAutoSpellAttSkillMin - этот параметр немного сложно понять, он работает так: если одно из заклинаний (соответствует вышеуказанным требованиям) влияет на характеристики или навыки (поглощение, истощение, урон, укрепление и т.д.), но не на здоровье, магии или выносливости, то iAutoSpellAttSkillMin поможет выбрать конкретное заклинание для определенного навыка или характеристики, так как многие из них имеют одинаковую стоимость кастования.

По всей видимости, выбирается то, которое соответствует следующему после высшего навыку или характеристике, которая равна или меньше этого установленного значения.

Таким образом, если ваш *Мистицизм* = 64, а *Длинные клинки* = 97, при **iAutoSpellAttSkillMin** = 64 заклинание, влияющее на навык, будет истощать, поглощать и т.д. *Мистицизм*.

Есть некоторые странные расхождения в том, какие именно заклинания для навыков/характеристик выбираются, но это, кажется, общая форма.

Значение по умолчанию: 70

Часть кода из OpenMW:

iAutoSpell{school}Max - лимит по максимальному кол-ву заклинаний для каждой из школ.

iAutoSpellTimesCanCast - на какое минимальное количество кастов заклинания должно хватить маны у игрока или NPC для того чтобы получить его как стартовое или при расчете Auto-Calc

```
baseActor.spells = vector()
```

```
baseMagicka = fNPCbaseMagickaMult * baseActor.intelligence
```

```
spellSchools = { Alteration, Conjuration, Destruction, Illusion, Mysticism, Restoration }
```

```
schoolCaps = {} # could be an array indexed by school enum
```

```
for each school in spellSchools:
```

```
    schoolCaps[school] = { count : 0,  
                          limit : iAutoSpell{school}Max,  
                          reachedLimit : iAutoSpell{school}Max <= 0,  
                          minCost : INT_MAX,  
                          weakestSpell : none }
```

```
for each spell in the game: # note: iteration order is important, see comments
```

```
    if spell.isMarkedDeleted: continue
```

```
    if spell.castingType != spell: continue
```

```
    if not spell.isAutoCalculate: continue
```

```
    if baseMagicka < iAutoSpellTimesCanCast * spell.cost: continue
```

```
    if spell is in baseActor.race.racialSpells: continue
```

```
    failedAttrSkillCheck = false
```

```
    for each effect in spell:
```

```
        if (effect.baseEffect.flags & TARGET_SKILL) and baseActor.skills[effect.targetSkill] <
```

```
        iAutoSpellAttSkillMin:
```

```
            failedAttrSkillCheck = true
```

```
            break
```

```
        if (effect.baseEffect.flags & TARGET_ATTR) and baseActor.attribute[effect.targetAttr] <
```

```
        iAutoSpellAttSkillMin:
```

```
            failedAttrSkillCheck = true
```

```
            break
```

```
    if failedAttrSkillCheck: continue
```

```
    school, _ = calcWeakestSchool(spell, actor)
```

```
cap = schoolCaps[school]
```

```
if cap.reachedLimit and spell.cost <= cap.minCost: continue
```

```
if calcBaseCastChance(baseActor, spell, school) < fAutoSpellChance: continue
```

```
baseActor.spells.add(spell)
```

```
if cap.reachedLimit:
```

```
    baseActor.spells.remove(cap.weakestSpell)
```

```
    cap.weakestSpell = baseActor.spells.findMinCostSpell() # note: not school specific
```

```
    cap.minCost = cap.weakestSpell.cost
```

```
else:
```

```
    cap.count += 1
```

```
    if cap.count == cap.limit:
```

```
        cap.reachedLimit = true
```

```
    if spell.cost < cap.minCost:
```

```
        cap.weakestSpell = spell
```

```
        cap.minCost = spell.cost
```

baseMagicka - запас маны для NPC

iAutoSpellAlterationMax

iAutoSpellAlterationMax - максимальное количество заклинаний школы Изменения которые может получить NPC при расчете его в Auto-Calc

Значение по умолчанию: 5

Часть кода из OpenMW:

iAutoSpell{school}Max - лимит по максимальному кол-ву заклинаний для каждой из школ.

iAutoSpellTimesCanCast - на какое минимальное количество кастов заклинания должно хватить маны у игрока или NPC для того чтобы получить его как стартовое или при расчете Auto-Calc

```
baseActor.spells = vector()
```

```
baseMagicka = fNPCbaseMagickaMult * baseActor.intelligence
```

```
spellSchools = { Alteration, Conjuration, Destruction, Illusion, Mysticism, Restoration }
```

```
schoolCaps = {} # could be an array indexed by school enum
```

```
for each school in spellSchools:
```

```
    schoolCaps[school] = { count : 0,  
                          limit : iAutoSpell{school}Max,  
                          reachedLimit : iAutoSpell{school}Max <= 0,  
                          minCost : INT_MAX,  
                          weakestSpell : none }
```

```
for each spell in the game: # note: iteration order is important, see comments
```

```
    if spell.isMarkedDeleted: continue
```

```
    if spell.castingType != spell: continue
```

```
    if not spell.isAutoCalculate: continue
```

```
    if baseMagicka < iAutoSpellTimesCanCast * spell.cost: continue
```

```
    if spell is in baseActor.race.racialSpells: continue
```

```
    failedAttrSkillCheck = false
```

```
    for each effect in spell:
```

```
        if (effect.baseEffect.flags & TARGET_SKILL) and baseActor.skills[effect.targetSkill] < iAutoSpellAttSkillMin:
```

```
            failedAttrSkillCheck = true
```

```
            break
```

```
        if (effect.baseEffect.flags & TARGET_ATTR) and baseActor.attribute[effect.targetAttr] <
```

```
            iAutoSpellAttSkillMin:
```

```
                failedAttrSkillCheck = true
```

```
                break
```

```
    if failedAttrSkillCheck: continue
```

```
    school, _ = calcWeakestSchool(spell, actor)
```

```
    cap = schoolCaps[school]
```

```
    if cap.reachedLimit and spell.cost <= cap.minCost: continue
```

```
    if calcBaseCastChance(baseActor, spell, school) < fAutoSpellChance: continue
```

```
    baseActor.spells.add(spell)
```

```
    if cap.reachedLimit:
```

```
        baseActor.spells.remove(cap.weakestSpell)
```

```
        cap.weakestSpell = baseActor.spells.findMinCostSpell() # note: not school specific
```

```
    cap.minCost = cap.weakestSpell.cost
else:
    cap.count += 1
    if cap.count == cap.limit:
        cap.reachedLimit = true

    if spell.cost < cap.minCost:
        cap.weakestSpell = spell
        cap.minCost = spell.cost
```

baseMagicka - запас маны для NPC

iAutoSpellConjurationMax

iAutoSpellConjurationMax - максимальное количество заклинаний школы Изменения которые может получить NPC при расчете его в Auto-Calc

Значение по умолчанию: 2

Часть кода из OpenMW:

iAutoSpell{school}Max - лимит по максимальному кол-ву заклинаний для каждой из школ.

iAutoSpellTimesCanCast - на какое минимальное количество кастов заклинания должно хватить маны у игрока или NPC для того чтобы получить его как стартовое или при расчете Auto-Calc

```
baseActor.spells = vector()
```

```
baseMagicka = fNPCbaseMagickaMult * baseActor.intelligence
```

```
spellSchools = { Alteration, Conjuration, Destruction, Illusion, Mysticism, Restoration }
```

```
schoolCaps = {} # could be an array indexed by school enum
```

```
for each school in spellSchools:
```

```
    schoolCaps[school] = { count : 0,  
                          limit : iAutoSpell{school}Max,  
                          reachedLimit : iAutoSpell{school}Max <= 0,  
                          minCost : INT_MAX,  
                          weakestSpell : none }
```

```
for each spell in the game: # note: iteration order is important, see comments
```

```
    if spell.isMarkedDeleted: continue
```

```
    if spell.castingType != spell: continue
```

```
    if not spell.isAutoCalculate: continue
```

```
    if baseMagicka < iAutoSpellTimesCanCast * spell.cost: continue
```

```
    if spell is in baseActor.race.racialSpells: continue
```

```
    failedAttrSkillCheck = false
```

```
    for each effect in spell:
```

```
        if (effect.baseEffect.flags & TARGET_SKILL) and baseActor.skills[effect.targetSkill] < iAutoSpellAttSkillMin:
```

```
            failedAttrSkillCheck = true
```

```
            break
```

```
        if (effect.baseEffect.flags & TARGET_ATTR) and baseActor.attribute[effect.targetAttr] <
```

```
            iAutoSpellAttSkillMin:
```

```
                failedAttrSkillCheck = true
```

```
                break
```

```
    if failedAttrSkillCheck: continue
```

```
    school, _ = calcWeakestSchool(spell, actor)
```

```
    cap = schoolCaps[school]
```

```
    if cap.reachedLimit and spell.cost <= cap.minCost: continue
```

```
    if calcBaseCastChance(baseActor, spell, school) < fAutoSpellChance: continue
```

```
    baseActor.spells.add(spell)
```

```
    if cap.reachedLimit:
```

```
        baseActor.spells.remove(cap.weakestSpell)
```

```
        cap.weakestSpell = baseActor.spells.findMinCostSpell() # note: not school specific
```

```
    cap.minCost = cap.weakestSpell.cost
else:
    cap.count += 1
    if cap.count == cap.limit:
        cap.reachedLimit = true

    if spell.cost < cap.minCost:
        cap.weakestSpell = spell
        cap.minCost = spell.cost
```

baseMagicka - запас маны для NPC

iAutoSpellDestructionMax

iAutoSpellDestructionMax - максимальное количество заклинаний школы Разрушения которые может получить NPC при расчете его в Auto-Calc

Значение по умолчанию: 5

Часть кода из OpenMW:

iAutoSpell{school}Max - лимит по максимальному кол-ву заклинаний для каждой из школ.

iAutoSpellTimesCanCast - на какое минимальное количество кастов заклинания должно хватить маны у игрока или NPC для того чтобы получить его как стартовое или при расчете Auto-Calc

```
baseActor.spells = vector()
```

```
baseMagicka = fNPCbaseMagickaMult * baseActor.intelligence
```

```
spellSchools = { Alteration, Conjuraton, Destruction, Illusion, Mysticism, Restoration }
```

```
schoolCaps = {} # could be an array indexed by school enum
```

```
for each school in spellSchools:
```

```
    schoolCaps[school] = { count : 0,  
                          limit : iAutoSpell{school}Max,  
                          reachedLimit : iAutoSpell{school}Max <= 0,  
                          minCost : INT_MAX,  
                          weakestSpell : none }
```

```
for each spell in the game: # note: iteration order is important, see comments
```

```
    if spell.isMarkedDeleted: continue
```

```
    if spell.castingType != spell: continue
```

```
    if not spell.isAutoCalculate: continue
```

```
    if baseMagicka < iAutoSpellTimesCanCast * spell.cost: continue
```

```
    if spell is in baseActor.race.racialSpells: continue
```

```
    failedAttrSkillCheck = false
```

```
    for each effect in spell:
```

```
        if (effect.baseEffect.flags & TARGET_SKILL) and baseActor.skills[effect.targetSkill] < iAutoSpellAttSkillMin:
```

```
            failedAttrSkillCheck = true
```

```
            break
```

```
        if (effect.baseEffect.flags & TARGET_ATTR) and baseActor.attribute[effect.targetAttr] <
```

```
            iAutoSpellAttSkillMin:
```

```
                failedAttrSkillCheck = true
```

```
                break
```

```
    if failedAttrSkillCheck: continue
```

```
    school, _ = calcWeakestSchool(spell, actor)
```

```
    cap = schoolCaps[school]
```

```
    if cap.reachedLimit and spell.cost <= cap.minCost: continue
```

```
    if calcBaseCastChance(baseActor, spell, school) < fAutoSpellChance: continue
```

```
    baseActor.spells.add(spell)
```

```
    if cap.reachedLimit:
```

```
        baseActor.spells.remove(cap.weakestSpell)
```

```
        cap.weakestSpell = baseActor.spells.findMinCostSpell() # note: not school specific
```



```
    cap.minCost = cap.weakestSpell.cost
else:
    cap.count += 1
    if cap.count == cap.limit:
        cap.reachedLimit = true

    if spell.cost < cap.minCost:
        cap.weakestSpell = spell
        cap.minCost = spell.cost
```

baseMagicka - запас маны для NPC

iAutoSpellIllusionMax

iAutoSpellIllusionMax - максимальное количество заклинаний школы Иллюзий которые может получить NPC при расчете его в Auto-Calc

Значение по умолчанию: 5

Часть кода из OpenMW:

iAutoSpell{school}Max - лимит по максимальному кол-ву заклинаний для каждой из школ.

iAutoSpellTimesCanCast - на какое минимальное количество кастов заклинания должно хватить маны у игрока или NPC для того чтобы получить его как стартовое или при расчете Auto-Calc

```
baseActor.spells = vector()
```

```
baseMagicka = fNPCbaseMagickaMult * baseActor.intelligence
```

```
spellSchools = { Alteration, Conjuraton, Destruction, Illusion, Mysticism, Restoration }
```

```
schoolCaps = {} # could be an array indexed by school enum
```

```
for each school in spellSchools:
```

```
    schoolCaps[school] = { count : 0,  
                          limit : iAutoSpell{school}Max,  
                          reachedLimit : iAutoSpell{school}Max <= 0,  
                          minCost : INT_MAX,  
                          weakestSpell : none }
```

```
for each spell in the game: # note: iteration order is important, see comments
```

```
    if spell.isMarkedDeleted: continue
```

```
    if spell.castingType != spell: continue
```

```
    if not spell.isAutoCalculate: continue
```

```
    if baseMagicka < iAutoSpellTimesCanCast * spell.cost: continue
```

```
    if spell is in baseActor.race.racialSpells: continue
```

```
    failedAttrSkillCheck = false
```

```
    for each effect in spell:
```

```
        if (effect.baseEffect.flags & TARGET_SKILL) and baseActor.skills[effect.targetSkill] < iAutoSpellAttSkillMin:
```

```
            failedAttrSkillCheck = true
```

```
            break
```

```
        if (effect.baseEffect.flags & TARGET_ATTR) and baseActor.attribute[effect.targetAttr] <
```

```
            iAutoSpellAttSkillMin:
```

```
                failedAttrSkillCheck = true
```

```
                break
```

```
    if failedAttrSkillCheck: continue
```

```
    school, _ = calcWeakestSchool(spell, actor)
```

```
    cap = schoolCaps[school]
```

```
    if cap.reachedLimit and spell.cost <= cap.minCost: continue
```

```
    if calcBaseCastChance(baseActor, spell, school) < fAutoSpellChance: continue
```

```
    baseActor.spells.add(spell)
```

```
    if cap.reachedLimit:
```

```
        baseActor.spells.remove(cap.weakestSpell)
```

```
        cap.weakestSpell = baseActor.spells.findMinCostSpell() # note: not school specific
```

```
    cap.minCost = cap.weakestSpell.cost
else:
    cap.count += 1
    if cap.count == cap.limit:
        cap.reachedLimit = true

    if spell.cost < cap.minCost:
        cap.weakestSpell = spell
        cap.minCost = spell.cost
```

baseMagicka - запас маны для NPC

iAutoSpellMysticismMax

iAutoSpellMysticismMax - максимальное количество заклинаний школы Мистицизма которые может получить NPC при расчете его в Auto-Calc

Значение по умолчанию: 5

Часть кода из OpenMW:

iAutoSpell{school}Max - лимит по максимальному кол-ву заклинаний для каждой из школ.

iAutoSpellTimesCanCast - на какое минимальное количество кастов заклинания должно хватить маны у игрока или NPC для того чтобы получить его как стартовое или при расчете Auto-Calc

```
baseActor.spells = vector()
```

```
baseMagicka = fNPCbaseMagickaMult * baseActor.intelligence
```

```
spellSchools = { Alteration, Conjuraton, Destruction, Illusion, Mysticism, Restoration }
```

```
schoolCaps = {} # could be an array indexed by school enum
```

```
for each school in spellSchools:
```

```
    schoolCaps[school] = { count : 0,  
                          limit : iAutoSpell{school}Max,  
                          reachedLimit : iAutoSpell{school}Max <= 0,  
                          minCost : INT_MAX,  
                          weakestSpell : none }
```

```
for each spell in the game: # note: iteration order is important, see comments
```

```
    if spell.isMarkedDeleted: continue
```

```
    if spell.castingType != spell: continue
```

```
    if not spell.isAutoCalculate: continue
```

```
    if baseMagicka < iAutoSpellTimesCanCast * spell.cost: continue
```

```
    if spell is in baseActor.race.racialSpells: continue
```

```
    failedAttrSkillCheck = false
```

```
    for each effect in spell:
```

```
        if (effect.baseEffect.flags & TARGET_SKILL) and baseActor.skills[effect.targetSkill] < iAutoSpellAttSkillMin:
```

```
            failedAttrSkillCheck = true
```

```
            break
```

```
        if (effect.baseEffect.flags & TARGET_ATTR) and baseActor.attribute[effect.targetAttr] <
```

```
            iAutoSpellAttSkillMin:
```

```
                failedAttrSkillCheck = true
```

```
                break
```

```
    if failedAttrSkillCheck: continue
```

```
    school, _ = calcWeakestSchool(spell, actor)
```

```
    cap = schoolCaps[school]
```

```
    if cap.reachedLimit and spell.cost <= cap.minCost: continue
```

```
    if calcBaseCastChance(baseActor, spell, school) < fAutoSpellChance: continue
```

```
    baseActor.spells.add(spell)
```

```
    if cap.reachedLimit:
```

```
        baseActor.spells.remove(cap.weakestSpell)
```

```
        cap.weakestSpell = baseActor.spells.findMinCostSpell() # note: not school specific
```

```
    cap.minCost = cap.weakestSpell.cost
else:
    cap.count += 1
    if cap.count == cap.limit:
        cap.reachedLimit = true

    if spell.cost < cap.minCost:
        cap.weakestSpell = spell
        cap.minCost = spell.cost
```

baseMagicka - запас маны для NPC

iAutoSpellRestorationMax

iAutoSpellRestorationMax - максимальное количество заклинаний школы Востановления которые может получить NPC при расчете его в Auto-Calc

Значение по умолчанию: 5

Часть кода из OpenMW:

iAutoSpell{school}Max - лимит по максимальному кол-ву заклинаний для каждой из школ.

iAutoSpellTimesCanCast - на какое минимальное количество кастов заклинания должно хватить маны у игрока или NPC для того чтобы получить его как стартовое или при расчете Auto-Calc

```
baseActor.spells = vector()
```

```
baseMagicka = fNPCbaseMagickaMult * baseActor.intelligence
```

```
spellSchools = { Alteration, Conjuraton, Destruction, Illusion, Mysticism, Restoration }
```

```
schoolCaps = {} # could be an array indexed by school enum
```

```
for each school in spellSchools:
```

```
    schoolCaps[school] = { count : 0,  
                          limit : iAutoSpell{school}Max,  
                          reachedLimit : iAutoSpell{school}Max <= 0,  
                          minCost : INT_MAX,  
                          weakestSpell : none }
```

```
for each spell in the game: # note: iteration order is important, see comments
```

```
    if spell.isMarkedDeleted: continue
```

```
    if spell.castingType != spell: continue
```

```
    if not spell.isAutoCalculate: continue
```

```
    if baseMagicka < iAutoSpellTimesCanCast * spell.cost: continue
```

```
    if spell is in baseActor.race.racialSpells: continue
```

```
    failedAttrSkillCheck = false
```

```
    for each effect in spell:
```

```
        if (effect.baseEffect.flags & TARGET_SKILL) and baseActor.skills[effect.targetSkill] < iAutoSpellAttSkillMin:
```

```
            failedAttrSkillCheck = true
```

```
            break
```

```
        if (effect.baseEffect.flags & TARGET_ATTR) and baseActor.attribute[effect.targetAttr] <
```

```
            iAutoSpellAttSkillMin:
```

```
                failedAttrSkillCheck = true
```

```
                break
```

```
    if failedAttrSkillCheck: continue
```

```
    school, _ = calcWeakestSchool(spell, actor)
```

```
    cap = schoolCaps[school]
```

```
    if cap.reachedLimit and spell.cost <= cap.minCost: continue
```

```
    if calcBaseCastChance(baseActor, spell, school) < fAutoSpellChance: continue
```

```
    baseActor.spells.add(spell)
```

```
    if cap.reachedLimit:
```

```
        baseActor.spells.remove(cap.weakestSpell)
```

```
        cap.weakestSpell = baseActor.spells.findMinCostSpell() # note: not school specific
```

```
    cap.minCost = cap.weakestSpell.cost
else:
    cap.count += 1
    if cap.count == cap.limit:
        cap.reachedLimit = true

    if spell.cost < cap.minCost:
        cap.weakestSpell = spell
        cap.minCost = spell.cost
```

baseMagicka - запас маны для NPC

iAutoPCSpellMax

iAutoPCSpellMax - максимальное количество заклинаний которые игрок может получить как стартовые.

Значение **по умолчанию**: 100

Часть кода из OpenMW:

```
baseMagicka = fPCbaseMagickaMult * baseActor.intelligence
reachedLimit = false
weakestSpell = none
minCost = INT_MAX

baseActor.spells = vector()

for each spell in the game:  # note: iteration order is important, see comments
    if spell.isMarkedDeleted: continue
    if spell.castingType != spell: continue
    if not spell.isPCStartSpell: continue

    if reachedLimit and spell.cost <= minCost: continue
    if spell is in baseActor.spells: continue
    if spell is in baseActor.race.racialSpells: continue
    if baseMagicka < spell.cost: continue
    if calcAutoCastChance(spell, baseActor, none) < fAutoPCSpellChance: continue

    failedAttrSkillCheck = false
    for each effect in spell:
        if (effect.baseEffect.flags & TARGET_SKILL) and baseActor.skills[effect.targetSkill] < iAutoSpellAttSkillMin:
            failedAttrSkillCheck = true
            break
        if (effect.baseEffect.flags & TARGET_ATTR) and baseActor.attribute[effect.targetAttr] <
iAutoSpellAttSkillMin:
            failedAttrSkillCheck = true
            break
    if failedAttrSkillCheck: continue

    baseActor.spells.add(spell)

    if reachedLimit:
        baseActor.spells.remove(weakestSpell)
        weakestSpell = baseActor.spells.findMinCostSpell()
        minCost = weakestSpell.cost
    else:
        if spell.cost < minCost:
            weakestSpell = spell
            minCost = spell.cost
        if baseActor.spells.size() == iAutoPCSpellMax:
            reachedLimit = true
```

baseMagicka - запас маны для PC

iAutoRepFacMod

iAutoRepFacMod - модификатор отношения между игроком и NPC в одной фракции.

Значение **по умолчанию**: 2

Часть кода из OpenMW:

if not in a faction:

reputation = 0

else:

*reputation = **iAutoRepFacMod** * rank + iAutoRepLevMod * (level - 1)*

where the entry level rank in the faction means rank = 1

iAutoRepLevMod

iAutoRepLevMod - модификатор отношения между игроком и NPC в одной фракции с зависимостью от уровня.

Значение **по умолчанию**: 0

Часть кода из OpenMW:

if not in a faction:

reputation = 0

else:

*reputation = iAutoRepFacMod * rank + **iAutoRepLevMod** * (level - 1)*

where the entry level rank in the faction means rank = 1

iMagicItemChargeOnce

ТРЕБУЕТ ПРОВЕРКИ

iMagicItemChargeOnce - значение для расчета стоимости каста в зарядах у зачарованной вещи или же для расчета количества зарядов.

Значение по умолчанию: 1

iMagicItemChargeConst

ТРЕБУЕТ ПРОВЕРКИ

iMagicItemChargeConst - значение для расчета стоимости каста в зарядах у зачарованной вещи или же для расчета количества зарядов.

Значение по умолчанию: 10

iMagicItemChargeUse

ТРЕБУЕТ ПРОВЕРКИ

iMagicItemChargeUse - значение для расчета стоимости каста в зарядах у зачарованной вещи или же для расчета количества зарядов.

Значение по умолчанию: 5

iMagicItemChargeStrike

ТРЕБУЕТ ПРОВЕРКИ

iMagicItemChargeStrike - значение для расчета стоимости каста в зарядах у зачарованной вещи или же для расчета количества зарядов.

Значение по умолчанию: 10

iMonthsToRespawn

iMonthsToRespawn - значение отвечающее за количество месяцев для обновления контейнеров с пометкой *Органический* в TES CS.

Значение **по умолчанию**: 4

fCorpseClearDelay

fCorpseClearDelay - константа определяющая количество часов до исчезновения мертвых тел.

Значение по умолчанию: 72

fCorpseRespawnDelay

fCorpseRespawnDelay - константа определяющая колличество часов до возрождения респавнящихся существ и NPC.

Значение **по умолчанию**: 72

fBarterGoldResetDelay

fBarterGoldResetDelay - отвечает через сколько часов запас золота торговца восстанавливается до его исходного значения.

Значение **по умолчанию**: 24

fEncumbranceStrMult

fEncumbranceStrMult - это параметр управляет тем, на сколько увеличение грузоподъемности персонажа зависит от его силы.

Значение **по умолчанию**: 5

Известная формула:

Max Encumbrance = (*pcStrength* * **fEncumbranceStrMult**).

pcStrength - сила

fPickLockMult

ФОРМУЛА ТРЕБУЕТ УТОЧНЕНИЯ

fPickLockMult - отвечает за множитель сложности замков. Он определяет, как быстро персонаж может взламывать замки с помощью инструмента для взлома (сложность взлома).

Значение по умолчанию: -1

Известная формула:

*Picklock chance = (securitySkill + pcIntelligence * 0.2 + pcLuck * 0.1) * Pick Quality * FatigueFunction + (Lock Difficulty * fPickLockMult).*

Код из OpenMW

```
x = 0.2 * pcAgility + 0.1 * pcLuck + securitySkill
x *= Pick Quality * FatigueFunction
x += fPickLockMult * Lock Difficulty

if x <= 0: fail and report impossible
roll 100, if roll <= x then open lock else report failure
```

securitySkill - Навык безопасность

pcIntelligence - Интеллект

pcLuck - Удача

Pick Quality - Качество отмычки

FatigueFunction - Уровень запаса сил

Lock Difficulty - Сложность замка

pcAgility - Ловкость

fTrapCostMult

ФОРМУЛА ТРЕБУЕТ УТОЧНЕНИЯ

fTrapCostMult - отвечает за множитель сложности для ловушки в зависимости от цены (в единицах маны) самого заклинания ловушки.

Значение по умолчанию: 0

Известная формула:

Trap disarming chance = (*securitySkill* + *pcIntelligence**0.2 + *pcLuck**0.1) * *Probe Quality* * *FatigueFunction* + (*trapSpellPoints* * **fTrapCostMult**).

Код из OpenMW

```
x = 0.2 * pcAgility + 0.1 * pcLuck + securitySkill
x += fTrapCostMult * trapSpellPoints
x *= Probe Quality * FatigueFunction
```

if x <= 0: fail and report impossible
roll 100, if roll <= x then untrap else report failure

securitySkill - Навык безопасность

pcIntelligence - Интеллект

pcLuck - Удача

Probe Quality - Качество щупа

FatigueFunction - Уровень запаса сил

trapSpellPoints - Цена заклинания ловушки

pcAgility - Ловкость

fMessageTimePerChar

fMessageTimePerChar - влияет на время отображения подсказок (messagebox)

Значение по умолчанию: 0.1

fMagicItemRechargePerSecond

fMagicItemRechargePerSecond - значение отвечающее за количество восстанавливающихся со временем зарядов в магическом предмете.

Значение **по умолчанию**: 0.05

Код из OpenMW:

$x = tickTime * \text{fMagicItemRechargePerSecond}$ (каждый кадр)

i1stPersonSneakData

i1stPersonSneakData - используется для выставления камеры от первого лица в режиме скрытности.

Значение по умолчанию: 10

iBarterSuccessDisposition

iBarterSuccessDisposition - временный модификатор для успешных сделок бартера.

Значение по умолчанию: 1

Код торговли из OpenMW:

all prices are negative when player is buying, positive when player is selling

accept if playerOffer <= merchantOffer (same for buy and sell)

if npc is a creature: reject (no haggle)

a = abs(merchantOffer)

b = abs(playerOffer)

*if buying: d = int(100 * (a - b) / a)*

*if selling: d = int(100 * (b - a) / b)*

clampedDisposition = clamp int(npcDisposition) to [0..100]

*dispositionTerm = fDispositionMod * (clampedDisposition - 50)*

*pcTerm = (dispositionTerm + pcMercantile + 0.1 * pcLuck + 0.2 * pcPersonality) * pcFatigueTerm*

*npcTerm = (npcMercantile + 0.1 * npcLuck + 0.2 * npcPersonality) * npcFatigueTerm*

*x = fBargainOfferMulti * d + fBargainOfferBase*

if buying: x += abs(int(pcTerm - npcTerm))

if selling: x += abs(int(npcTerm - pcTerm))

roll 100, if roll <= x then trade is accepted

adjust npc temporary disposition by iBarterSuccessDisposition or iBarterFailDisposition

iBarterFailDisposition

iBarterFailDisposition - временный модификатор для неудачных попыток бартера.

Значение **по умолчанию**: -1

Код торговли из OpenMW:

all prices are negative when player is buying, positive when player is selling

accept if playerOffer <= merchantOffer (same for buy and sell)

if npc is a creature: reject (no haggle)

a = abs(merchantOffer)

b = abs(playerOffer)

*if buying: d = int(100 * (a - b) / a)*

*if selling: d = int(100 * (b - a) / b)*

clampedDisposition = clamp int(npcDisposition) to [0..100]

*dispositionTerm = fDispositionMod * (clampedDisposition - 50)*

*pcTerm = (dispositionTerm + pcMercantile + 0.1 * pcLuck + 0.2 * pcPersonality) * pcFatigueTerm*

*npcTerm = (npcMercantile + 0.1 * npcLuck + 0.2 * npcPersonality) * npcFatigueTerm*

*x = fBargainOfferMulti * d + fBargainOfferBase*

if buying: x += abs(int(pcTerm - npcTerm))

if selling: x += abs(int(npcTerm - pcTerm))

roll 100, if roll <= x then trade is accepted

adjust npc temporary disposition by iBarterSuccessDisposition or iBarterFailDisposition

iLevelupTotal

iLevelupTotal - сколько повышений навыков всего требуется для того чтобы перейти на новый уровень

Значение **по умолчанию**: 10

Код из OpenMW:

```
if skill in player.majorSkills:
    totalIncreases += iLevelUpMajorMult
    attribCounter[skill->basicAttribute] += iLevelUpMajorMultAttribute
elif skill in player.minorSkills:
    totalIncreases += iLevelUpMinorMult
    attribCounter[skill->basicAttribute] += iLevelUpMinorMultAttribute
elif skill in player.miscSkills:
    attribCounter[skill->basicAttribute] += iLevelupMiscMultAttriubte # note game setting name has a typo

if totalIncreases >= iLevelUpTotal:
    level up becomes available
    totalIncreases -= iLevelUpTotal # note rollover mechanic
    attribCounter[8] = [0,0,0,0,0,0,0,0]
```

iLevelupMajorMult

iLevelupMajorMult - модификатор того сколько очков повышения уровня дает одно повышение *Главного навыка*.

Значение **по умолчанию**: 1

Код из OpenMW:

```
if skill in player.majorSkills:
    totalIncreases += iLevelUpMajorMult
    attribCounter[skill->basicAttribute] += iLevelUpMajorMultAttribute
elif skill in player.minorSkills:
    totalIncreases += iLevelUpMinorMult
    attribCounter[skill->basicAttribute] += iLevelUpMinorMultAttribute
elif skill in player.miscSkills:
    attribCounter[skill->basicAttribute] += iLevelupMiscMultAttriubte # note game setting name has a typo

if totalIncreases >= iLevelUpTotal:
    level up becomes available
    totalIncreases -= iLevelUpTotal # note rollover mechanic
    attribCounter[8] = [0,0,0,0,0,0,0,0]
```

iLevelupMinorMult

iLevelupMinorMult - модификатор того сколько очков повышения уровня дает одно повышение *Важного навыка*.

Значение **по умолчанию**: 1

Код из OpenMW:

```
if skill in player.majorSkills:
    totalIncreases += iLevelUpMajorMult
    attribCounter[skill->basicAttribute] += iLevelUpMajorMultAttribute
elif skill in player.minorSkills:
    totalIncreases += iLevelUpMinorMult
    attribCounter[skill->basicAttribute] += iLevelUpMinorMultAttribute
elif skill in player.miscSkills:
    attribCounter[skill->basicAttribute] += iLevelupMiscMultAttriubte # note game setting name has a typo

if totalIncreases >= iLevelUpTotal:
    level up becomes available
    totalIncreases -= iLevelUpTotal # note rollover mechanic
    attribCounter[8] = [0,0,0,0,0,0,0,0]
```

iLevelUpMajorMultAttribute

iLevelUpMajorMultAttribute - множитель влияющий на то сколько дополнительных характеристик сможет получить игрок при поднятии уровня (x1 x2 x3 x4 x5).

Значение **по умолчанию**: 1

Код из OpenMW:

```
if skill in player.majorSkills:
    totalIncreases += iLevelUpMajorMult
    attribCounter[skill->basicAttribute] += iLevelUpMajorMultAttribute
elif skill in player.minorSkills:
    totalIncreases += iLevelUpMinorMult
    attribCounter[skill->basicAttribute] += iLevelUpMinorMultAttribute
elif skill in player.miscSkills:
    attribCounter[skill->basicAttribute] += iLevelupMiscMultAttriubte # note game setting name has a typo

if totalIncreases >= iLevelUpTotal:
    level up becomes available
    totalIncreases -= iLevelUpTotal # note rollover mechanic
    attribCounter[8] = [0,0,0,0,0,0,0,0]
```

iLevelupMinorMultAttribute

iLevelupMinorMultAttribute - множитель влияющий на то сколько дополнительных характеристик сможет получить игрок при поднятии уровня (x1 x2 x3 x4 x5).

Значение **по умолчанию**: 1

Код из OpenMW:

```
if skill in player.majorSkills:
    totalIncreases += iLevelUpMajorMult
    attribCounter[skill->basicAttribute] += iLevelUpMajorMultAttribute
elif skill in player.minorSkills:
    totalIncreases += iLevelUpMinorMult
    attribCounter[skill->basicAttribute] += iLevelUpMinorMultAttribute
elif skill in player.miscSkills:
    attribCounter[skill->basicAttribute] += iLevelupMiscMultAttriubte # note game setting name has a typo

if totalIncreases >= iLevelUpTotal:
    level up becomes available
    totalIncreases -= iLevelUpTotal # note rollover mechanic
    attribCounter[8] = [0,0,0,0,0,0,0,0]
```

iLevelUpMiscMultAttribute

iLevelUpMiscMultAttribute - множитель определяющий на сколько повысится *маловажный* навык при его прокачке

Значение **по умолчанию**: 1

Код из OpenMW:

```
if skill in player.majorSkills:
    totalIncreases += iLevelUpMajorMult
    attribCounter[skill->basicAttribute] += iLevelUpMajorMultAttribute
elif skill in player.minorSkills:
    totalIncreases += iLevelUpMinorMult
    attribCounter[skill->basicAttribute] += iLevelUpMinorMultAttribute
elif skill in player.miscSkills:
    attribCounter[skill->basicAttribute] += iLevelupMiscMultAttriubte # note game setting name has a typo

if totalIncreases >= iLevelUpTotal:
    level up becomes available
    totalIncreases -= iLevelUpTotal # note rollover mechanic
    attribCounter[8] = [0,0,0,0,0,0,0,0]
```


iLevelUpSpecialization

ТРЕБУЕТ ПРОВЕРКИ

iLevelUpSpecialization - неизвестно, не ясно где используется.

Значение по умолчанию: 1

Сайт OpenMW сообщает следующее: *used; may not have any gameplay effect.*

iLevelUp01Mult

iLevelUp01Mult - это параметр определяет, сколько очков характеристик вы получаете при повышении уровня.

Значение **по умолчанию: 2**

Код из OpenMW:

```
for each attribute:
    count = attribCounter[attribute]
    if count == 0:
        bonus = 1
    elif count <= 9:
        bonus = {iLevelUp01Mult, .. iLevelUp09Mult} selected by count
    else: # count >= 10
        bonus = iLevelUp10Mult

bonus = min(bonus, 100 - player.attribute[selectedAttribute])
```

iLevelUp02Mult

iLevelUp02Mult - это параметр определяет, сколько очков характеристик вы получаете при повышении уровня.

Значение **по умолчанию: 1**

Код из OpenMW:

```
for each attribute:
    count = attribCounter[attribute]
    if count == 0:
        bonus = 1
    elif count <= 9:
        bonus = {iLevelUp01Mult, .. iLevelUp09Mult} selected by count
    else: # count >= 10
        bonus = iLevelUp10Mult

bonus = min(bonus, 100 - player.attribute[selectedAttribute])
```

iLevelUp03Mult

iLevelUp03Mult - это параметр определяет, сколько очков характеристик вы получаете при повышении уровня.

Значение **по умолчанию: 2**

Код из OpenMW:

```
for each attribute:
    count = attribCounter[attribute]
    if count == 0:
        bonus = 1
    elif count <= 9:
        bonus = {iLevelUp01Mult, .. iLevelUp09Mult} selected by count
    else: # count >= 10
        bonus = iLevelUp10Mult

bonus = min(bonus, 100 - player.attribute[selectedAttribute])
```

iLevelUp04Mult

iLevelUp04Mult - это параметр определяет, сколько очков характеристик вы получаете при повышении уровня.

Значение **по умолчанию: 2**

Код из OpenMW:

```
for each attribute:
    count = attribCounter[attribute]
    if count == 0:
        bonus = 1
    elif count <= 9:
        bonus = {iLevelUp01Mult, .. iLevelUp09Mult} selected by count
    else: # count >= 10
        bonus = iLevelUp10Mult

bonus = min(bonus, 100 - player.attribute[selectedAttribute])
```

iLevelUp05Mult

iLevelUp05Mult - это параметр определяет, сколько очков характеристик вы получаете при повышении уровня.

Значение **по умолчанию: 3**

Код из OpenMW:

```
for each attribute:
    count = attribCounter[attribute]
    if count == 0:
        bonus = 1
    elif count <= 9:
        bonus = {iLevelUp01Mult, .. iLevelUp09Mult} selected by count
    else: # count >= 10
        bonus = iLevelUp10Mult

bonus = min(bonus, 100 - player.attribute[selectedAttribute])
```

iLevelUp06Mult

iLevelUp06Mult - это параметр определяет, сколько очков характеристик вы получаете при повышении уровня.

Значение **по умолчанию: 3**

Код из OpenMW:

```
for each attribute:
    count = attribCounter[attribute]
    if count == 0:
        bonus = 1
    elif count <= 9:
        bonus = {iLevelUp01Mult, .. iLevelUp09Mult} selected by count
    else: # count >= 10
        bonus = iLevelUp10Mult

bonus = min(bonus, 100 - player.attribute[selectedAttribute])
```

iLevelUp07Mult

iLevelUp07Mult - это параметр определяет, сколько очков характеристик вы получаете при повышении уровня.

Значение **по умолчанию: 3**

Код из OpenMW:

```
for each attribute:
    count = attribCounter[attribute]
    if count == 0:
        bonus = 1
    elif count <= 9:
        bonus = {iLevelUp01Mult, .. iLevelUp09Mult} selected by count
    else: # count >= 10
        bonus = iLevelUp10Mult

bonus = min(bonus, 100 - player.attribute[selectedAttribute])
```


iLevelUp08Mult

iLevelUp08Mult - это параметр определяет, сколько очков характеристик вы получаете при повышении уровня.

Значение **по умолчанию: 4**

Код из OpenMW:

```
for each attribute:
    count = attribCounter[attribute]
    if count == 0:
        bonus = 1
    elif count <= 9:
        bonus = {iLevelUp01Mult, .. iLevelUp09Mult} selected by count
    else: # count >= 10
        bonus = iLevelUp10Mult

bonus = min(bonus, 100 - player.attribute[selectedAttribute])
```

iLevelUp09Mult

iLevelUp09Mult - это параметр определяет, сколько очков характеристик вы получаете при повышении уровня.

Значение **по умолчанию: 4**

Код из OpenMW:

```
for each attribute:
    count = attribCounter[attribute]
    if count == 0:
        bonus = 1
    elif count <= 9:
        bonus = {iLevelUp01Mult, .. iLevelUp09Mult} selected by count
    else: # count >= 10
        bonus = iLevelUp10Mult

bonus = min(bonus, 100 - player.attribute[selectedAttribute])
```

iLevelUp10Mult

iLevelUp10Mult - это параметр определяет, сколько очков характеристик вы получаете при повышении уровня.

Значение **по умолчанию: 5**

Код из OpenMW:

```
for each attribute:
    count = attribCounter[attribute]
    if count == 0:
        bonus = 1
    elif count <= 9:
        bonus = {iLevelUp01Mult, .. iLevelUp09Mult} selected by count
    else: # count >= 10
        bonus = iLevelUp10Mult

bonus = min(bonus, 100 - player.attribute[selectedAttribute])
```

iSoulAmountForConstantEffect

iSoulAmountForConstantEffect - минимальный размер души, необходимое для создания чар с постоянным эффектом.

Значение по умолчанию: 400

fConstantEffectMult

НЕ ИСПОЛЬЗУЕТСЯ В ОРИГИНАЛЬНОМ ДВИЖКЕ MORROWIND И OPENMW

fConstantEffectMult -

Значение по умолчанию: 15

fEnchantmentConstantDurationMult

fEnchantmentConstantDurationMult - модификатор стоимости для зачарований на постоянный эффект.

Значение по умолчанию: 0.5

Код из OpenMW:

```
y = 0
z = []
enchantPoints = 0

for each effect in enchantment:
    if enchantment is constant effect:
        if effect.magnitudeMin > 1 or effect.magnitudeMax > 1 or effect.radius > 1:
            effect.duration = int(fEnchantmentConstantDurationMult)
        else:
            effect.duration = 100

    x = 0.5 * (max(1, effect.magnitudeMin) + max(1, effect.magnitudeMax))
    x *= 0.1 * effect.magicEffect.baseMagickaCost
    x *= effect.duration # note difference from spellmaking
    x += 0.05 * max(1, spell.radius) * effect.magicEffect.baseMagickaCost

    y += x * fEffectCostMult
    y = max(1, y)
    if effect.rangeType & CAST_TARGET: y *= 1.5

    enchantPoints += int(y)
    z[effect.order] = int(pcEnchantSkill - y * fEnchantmentChanceMult)

# note enchantPoints not used for cost
cost of enchanting service = barterOffer(npc, y * fEnchantmentValueMult, buying)
```

fEnchantmentConstantChanceMult

fEnchantmentConstantChanceMult - корректировка шанса для постоянных эффектов. По умолчанию сделать постоянный эффект в два раза сложнее (=0.5).

Значение **по умолчанию: 0.1**

Код из OpenMW:

```
for each effect in the enchantment:  
   $x = z[effect.order] - \text{int}(-0.2 * pcIntelligence) + \text{int}(0.1 * pcLuck)$   
   $x = \text{int}(x * fatigueTerm)$   
  if enchantment is constant effect:  
     $x = \text{int}(x * \mathbf{fEnchantmentConstantChanceMult})$   
  roll 100, success if roll < x, else fail and destroy gem
```

Существует серьезная ошибка, из-за которой неудача проверяется для каждого эффекта, а не один раз для всей попытки, при этом по-прежнему используется совокупная стоимость. Это делает самозачарование с несколькими эффектами почти невозможным.

fWeaponDamageMult

fWeaponDamageMult - модификатор износа оружия при ударах.

Значение по умолчанию: 0

*Damage to weapon = (Damage dealt * **fWeaponDamageMult**), (с округлением в меньшую сторону, но не меньше 1)*

if attack missed: rawDamage = 0

*$x = \max(1, \mathbf{fWeaponDamageMult} * \text{rawDamage})$*

weapon loses x condition

fSeriousWoundMult

НЕ ИСПОЛЬЗУЕТСЯ В ОРИГИНАЛЬНОМ ДВИЖКЕ MORROWIND И OPENMW

fSeriousWoundMult -

Значение по умолчанию: 0

fKnockDownMult

fKnockDownMult - модификатор для расчета ловкости в формуле расчета возможности накаутирования.

Значение по умолчанию: 0.5

Код из OpenMW:

damage = incoming damage before armour reduction (unmitigatedDamage)

*agilityTerm = agility * fKnockDownMult*

*knockdownTerm = agility * iKnockDownOddsMult * 0.01 + iKnockDownOddsBase*

roll 100, knockdown occurs if agilityTerm <= damage and knockdownTerm <= roll

iKnockDownOddsBase

iKnockDownOddsBase - базовое значение в формуле вероятности устоять на ногах при входящем ударе.

Значение по умолчанию: 50

Код из OpenMW:

damage = incoming damage before armour reduction (unmitigatedDamage)

*agilityTerm = agility * fKnockDownMult*

*knockdownTerm = agility * iKnockDownOddsMult * 0.01 + iKnockDownOddsBase*

roll 100, knockdown occurs if agilityTerm <= damage and knockdownTerm <= roll

iKnockDownOddsMult

iKnockDownOddsMult - модификатор для ловкости в формуле вероятности устоять на ногах при входящем ударе.

Значение по умолчанию: 50

Код из OpenMW:

damage = incoming damage before armour reduction (unmitigatedDamage)

*agilityTerm = agility * fKnockDownMult*

*knockdownTerm = agility * iKnockDownOddsMult * 0.01 + iKnockDownOddsBase*

roll 100, нокдаун происходит, если agilityTerm <= damage and knockdownTerm <= roll

fCombatArmorMinMult

fCombatArmorMinMult - модификатор минимального процента урона которой пройдет сквозь броню в любом случае.

Значение по умолчанию: **0.25**

Код OpenMW:

Physical damage applied to a target actor

if damage < 0.001: skip rest of mitigation, set damage to 0

unmitigatedDamage = damage

x = damage / (damage + actor.effectiveArmorRating)

*damage *= max(fCombatArmorMinMult, x)*

z = int(unmitigatedDamage - damage)

if damage < 1: damage = 1

armour = weighted selection from actor armour slots

if armour is equipped there, armour loses z condition

if actor is player: player exercises relevant armour skill for armour

fHandToHandReach

fHandToHandReach - отвечает за дистанцию, на которой игрок может наносить удары *рукопашным* боем (Hand-to-Hand). Эта переменная определяет максимальное расстояние, на котором игрок может достигнуть противника в бою, используя только свои руки.

Значение **по умолчанию: 1**

fVoiceldleOdds

fVoiceldleOdds - это параметр который определяет вероятность того, что персонаж произнесет случайную реплику (так называемый "войс-айдл").

Значение по умолчанию: 10

iVoiceAttackOdds

iVoiceAttackOdds - это параметр который определяет вероятность того, что персонаж произнесет случайную реплику в бою.

Значение по умолчанию: 10

iVoiceHitOdds

iVoiceHitOdds - это параметр который определяет вероятность того, что персонаж произнесет случайную реплику при получении урона.

Значение **по умолчанию: 30**

fProjectileMinSpeed

fProjectileMinSpeed - это переменная которая определяет минимальную скорость полета стрел и болтов.

Значение по умолчанию: **400**

fProjectileMaxSpeed

fProjectileMaxSpeed - это переменная которая определяет максимальную скорость полета стрел и болтов.

Значение по умолчанию: 3000

fThrownWeaponMinSpeed

fThrownWeaponMinSpeed - это переменная которая определяет минимальную скорость полета метательного оружия.

Значение по умолчанию: 300

fThrownWeaponMaxSpeed

fThrownWeaponMaxSpeed - это переменная которая определяет максимальную скорость полета метательного оружия.

Значение по умолчанию: 1000

fTargetSpellMaxSpeed

fTargetSpellMaxSpeed - это переменная которая определяет максимальную скорость полета заклинаний.

Значение по умолчанию: **1000**

fProjectileThrownStoreChance

fProjectileThrownStoreChance - определяет шанс вернуть с тела метательное оружие или стрелы.

Значение по умолчанию: 25

iPickMinChance

iPickMinChance - это значение, которое определяет минимальную вероятность, с которой игрок сможет открыть замки и сундуки.

Значение **по умолчанию: 5**

Код из OpenMW:

```
fatigueTerm = fFatigueBase - fFatigueMult*(1 - normalisedFatigue)
#where normalisedFatigue is a function of fatigue. empty fatigue bar -> 0.0, full fatigue bar -> 1.0
#note fatigueTerm is normally 1.25 at full fatigue.
```

```
#checks the whole stack no matter how many you try to take
#note: filled soulgems have the value of an empty soulgem due to a missing calculation
stackValue = itemValue * itemsInStack
valueTerm = 10 * fPickPocketMod * stackValue
```

```
x = (0.2 * pcAgility + 0.1 * pcLuck + pcSneak) * fatigueTerm
y = (valueTerm + npcSneak + 0.2 * npcAgilityTerm + 0.1 * npcLuckTerm) * npcFatigueTerm
t = x - y + x # yes, that is what it does
```

```
if t < pcSneak / iPickMinChance:
    roll 100, win if roll <= int(pcSneak / iPickMinChance)
else:
    t = min(iPickMaxChance, t)
    roll 100, win if roll <= int(t)
```


iPickMaxChance

iPickMaxChance - это значение, которое определяет максимальную вероятность, с которой игрок сможет открыть замки и сундуки.

Значение **по умолчанию: 75**

Код из OpenMW:

```
fatigueTerm = fFatigueBase - fFatigueMult*(1 - normalisedFatigue)
#where normalisedFatigue is a function of fatigue. empty fatigue bar -> 0.0, full fatigue bar -> 1.0
#note fatigueTerm is normally 1.25 at full fatigue.
```

```
#checks the whole stack no matter how many you try to take
#note: filled soulgems have the value of an empty soulgem due to a missing calculation
stackValue = itemValue * itemsInStack
valueTerm = 10 * fPickPocketMod * stackValue
```

```
x = (0.2 * pcAgility + 0.1 * pcLuck + pcSneak) * fatigueTerm
y = (valueTerm + npcSneak + 0.2 * npcAgilityTerm + 0.1 * npcLuckTerm) * npcFatigueTerm
t = x - y + x # yes, that is what it does
```

```
if t < pcSneak / iPickMinChance:
    roll 100, win if roll <= int(pcSneak / iPickMinChance)
else:
    t = min(iPickMaxChance, t)
    roll 100, win if roll <= int(t)
```

fDispRaceMod

fDispRaceMod - отвечает за изменение отношения неписей к игроку в зависимости от расы персонажа.

Значение по умолчанию: 5

fDispPersonnalityMult

fDispPersonnalityMult - модификатор для расчета отношения NPC к игроку.

Значение по умолчанию: **0.5**

Известная формула:

Base Disposition = $(50 + (pcPersonality - fDispPersonalityBase) * fDispPersonalityMult)$.

fDispPersonalityBase

fDispPersonalityBase - базовый параметр для расчета отношения между NPC и игроком.

Значение по умолчанию: **50**

Известная формула:

Base Disposition = $(50 + (pcPersonality - fDispPersonalityBase) * fDispPersonalityMult)$.

fDispFactionMod

fDispFactionMod - Параметр отвечающий за расчет отношения NPC к игроку внутри фракции.

Значение **по умолчанию**: 3

Известная формула:

*Add (Faction Relation * **fDispFactionMod** * fDispFactionRankBase * (fDispFactionRankMult * (Faction Rank - 1)))*.

fDispFactionRankBase

fDispFactionRankBase - Параметр отвечающий за расчет отношения NPC к игроку внутри фракции.

Значение по умолчанию: **1**

Известная формула:

*Add (Faction Relation * fDispFactionMod * **fDispFactionRankBase** * (fDispFactionRankMult * (Faction Rank - 1)))*.

fDispFactionRankMult

fDispFactionRankMult - Параметр отвечающий за расчет отношения NPC к игроку внутри фракции.

Значение **по умолчанию: 0.5**

Известная формула:

Add (Faction Relation * fDispFactionMod * fDispFactionRankBase * (**fDispFactionRankMult** * (Faction Rank - 1))).

fDispCrimeMod

fDispCrimeMod - модификатор отношения с игроком когда его разыскивают.

Значение **по умолчанию: 0**

Известная формула:

Subtract (Crime Level * **fDispCrimeMod**).

fDispDiseaseMod

fDispDiseaseMod - модификтор для отношения с зараженным игроком.

Значение по умолчанию: **-10**

iDispAttackMod

iDispAttackMod - модификатор отношения NPC к игроку когда тот совершает преступление атакуя других NPC.

Значение **по умолчанию**: **-50**

Код из OpenMW:

```
if crime is killing:
    fightTerm, dispTerm = iFightKilling, iDispKilling
if crime is assault, observed:
    fightTerm, dispTerm = iFightAttacking, iDispAttackMod
if crime is assault, victim:
    fightTerm, dispTerm = iFightAttack, fDispAttacking
if crime is theft:
    fightTerm, dispTerm = fFightStealing * stack value, fDispStealing * stack value
if crime is pickpocket, observed:
    fightTerm, dispTerm = iFightPickpocket, fDispPickpocketMod
if crime is pickpocket, victim:
    fightTerm, dispTerm = 4 * iFightPickpocket, fDispPickpocketMod
    if alarmTerm <= 0: alarmTerm = 1.0
if crime is trespass:
    fightTerm, dispTerm = iFightTrespass, iDispTresspass
```

fDispWeaponDrawn

fDispWeaponDrawn - модификатор отношения NPC к игроку когда игрок обножил оружие.

Значение по умолчанию: **-5**

fDispBargainSuccessMod

НЕ ИСПОЛЬЗУЕТСЯ В ОРИГИНАЛЬНОМ ДВИЖКЕ MORROWIND И OPENMW

fDispBargainSuccessMod -

Значение по умолчанию: 1

fDispBargainFailMod

НЕ ИСПОЛЬЗУЕТСЯ В ОРИГИНАЛЬНОМ ДВИЖКЕ MORROWIND И OPENMW

fDispBargainFailMod -

Значение по умолчанию: -1

fDispPickPocketMod

fDispPickPocketMod - модификатор отношения NPC к игроку когда тот совершает преступление карманной кражи.

Значение по умолчанию: **-25**

iDaysinPrisonMod

iDaysinPrisonMod - отвечает за модификатор количества дней, которое игрок проводит в тюрьме при совершении преступления.

Значение **по умолчанию: 100**

fDispAttacking

iDispAttackMod - модификатор отношения NPC к игроку когда тот совершает нападение.

Значение **по умолчанию: -10**

Код из OpenMW:

```
if crime is killing:
    fightTerm, dispTerm = iFightKilling, iDispKilling
if crime is assault, observed:
    fightTerm, dispTerm = iFightAttacking, iDispAttackMod
if crime is assault, victim:
    fightTerm, dispTerm = iFightAttack, fDispAttacking
if crime is theft:
    fightTerm, dispTerm = fFightStealing * stack value, fDispStealing * stack value
if crime is pickpocket, observed:
    fightTerm, dispTerm = iFightPickpocket, fDispPickpocketMod
if crime is pickpocket, victim:
    fightTerm, dispTerm = 4 * iFightPickpocket, fDispPickpocketMod
    if alarmTerm <= 0: alarmTerm = 1.0
if crime is trespass:
    fightTerm, dispTerm = iFightTrespass, iDispTresspass
```


fDispStealing

fDispStealing - модификтор на которой множится цена украденного предмета для расчета цены штрафа для игрока если он попадаете на краже.

Значение **по умолчанию: -0.5**

Код из OpenMW:

```
for each observer in the cell:    # NPCs only, range unknown
    alarmTerm = 0.01 * observer.alarm

    if crime is killing:
        fightTerm, dispTerm = iFightKilling, iDispKilling
    if crime is assault, observed:
        fightTerm, dispTerm = iFightAttacking, iDispAttackMod
    if crime is assault, victim:
        fightTerm, dispTerm = iFightAttack, fDispAttacking
    if crime is theft:
        fightTerm, dispTerm = fFightStealing * stack value, fDispStealing * stack value
    if crime is pickpocket, observed:
        fightTerm, dispTerm = iFightPickpocket, fDispPickpocketMod
    if crime is pickpocket, victim:
        fightTerm, dispTerm = 4 * iFightPickpocket, fDispPickpocketMod
        if alarmTerm <= 0: alarmTerm = 1.0
    if crime is trespass:
        fightTerm, dispTerm = iFightTrespass, iDispTresspass

    if observer is not victim or observer.isGuard:
        dispTerm *= alarmTerm

    fightTerm += fFightDispMult * (50 - dispTerm)
    x = iFightDistanceBase - fFightDistanceMultiplier * distance(player, observer)
    fightTerm = alarmTerm * (fightTerm + x)

    if observer.fight + fightTerm > 100:
        fightTerm = 100 - observer.fight
    fightTerm = max(0, fightTerm)

    observer.fight += int(fightTerm)
    observer.disposition += int(dispTerm)

- lots more crime system stuff occurs here -

if (observer.alarm < 100 or not observer.isGuard) and observer.fight < 100:
    observer.fight is reset to the original value
    if observer is not victim: observer.disposition is reset to the original value
```

iDispTresspass

iDispTresspass - модификатор падения отношения к игроку, когда NPC стало известно о совершенном преступлении игроком.

Значение **по умолчанию**: **-20**

Код из OpenMW:

```
for each observer in the cell:    # NPCs only, range unknown
    alarmTerm = 0.01 * observer.alarm

    if crime is killing:
        fightTerm, dispTerm = iFightKilling, iDispKilling
    if crime is assault, observed:
        fightTerm, dispTerm = iFightAttacking, iDispAttackMod
    if crime is assault, victim:
        fightTerm, dispTerm = iFightAttack, fDispAttacking
    if crime is theft:
        fightTerm, dispTerm = fFightStealing * stack value, fDispStealing * stack value
    if crime is pickpocket, observed:
        fightTerm, dispTerm = iFightPickpocket, fDispPickpocketMod
    if crime is pickpocket, victim:
        fightTerm, dispTerm = 4 * iFightPickpocket, fDispPickpocketMod
        if alarmTerm <= 0: alarmTerm = 1.0
    if crime is trespass:
        fightTerm, dispTerm = iFightTresspass, iDispTresspass

    if observer is not victim or observer.isGuard:
        dispTerm *= alarmTerm

    fightTerm += fFightDispMult * (50 - dispTerm)
    x = iFightDistanceBase - fFightDistanceMultiplier * distance(player, observer)
    fightTerm = alarmTerm * (fightTerm + x)

    if observer.fight + fightTerm > 100:
        fightTerm = 100 - observer.fight
    fightTerm = max(0, fightTerm)

    observer.fight += int(fightTerm)
    observer.disposition += int(dispTerm)

- lots more crime system stuff occurs here -

if (observer.alarm < 100 or not observer.isGuard) and observer.fight < 100:
    observer.fight is reset to the original value
    if observer is not victim: observer.disposition is reset to the original value
```

iDispKilling

iDispKilling - модификатор падения отношения к игроку, когда NPC стало известно о соответствующем преступлении PC.

Значение по умолчанию: **-50**

iTrainingMod

iTrainingMod - модификатор стоимости для обучения.

Значение **по умолчанию: 10**

Код из OpenMW:

Base training price = (Current Skill Level * **iTrainingMod**).

iAlchemyMod

iAlchemyMod - модификатор для стоимости зелий которые изготавливает игрок.

Значение **по умолчанию: 2**

fBargainOfferBase

fBargainOfferBase - отвечает за минимально возможную цену покупки и максимально возможную цену продажи.

Значение по умолчанию: 50

Код из OpenMW:

all prices are negative when player is buying, positive when player is selling

accept if playerOffer <= merchantOffer (same for buy and sell)

if npc is a creature: reject (no haggle)

a = abs(merchantOffer)

b = abs(playerOffer)

if buying: d = int(100 * (a - b) / a)

if selling: d = int(100 * (b - a) / b)

clampedDisposition = clamp int(npcDisposition) to [0..100]

dispositionTerm = fDispositionMod * (clampedDisposition - 50)

pcTerm = (dispositionTerm + pcMercantile + 0.1 * pcLuck + 0.2 * pcPersonality) * pcFatigueTerm

npcTerm = (npcMercantile + 0.1 * npcLuck + 0.2 * npcPersonality) * npcFatigueTerm

x = fBargainOfferMulti * d + **fBargainOfferBase**

if buying: x += abs(int(pcTerm - npcTerm))

if selling: x += abs(int(npcTerm - pcTerm))

roll 100, if roll <= x then trade is accepted

adjust npc temporary disposition by iBarterSuccessDisposition or iBarterFailDisposition

fBargainOfferMulti

fBargainOfferMulti - используется для определения множителя, который учитывается при расчете стоимости предметов, которые игрок может продавать или покупать у торговцев в игре.

Значение **по умолчанию**: -4

Код из OpenMW:

all prices are negative when player is buying, positive when player is selling

accept if playerOffer <= merchantOffer (same for buy and sell)

if npc is a creature: reject (no haggle)

a = abs(merchantOffer)

b = abs(playerOffer)

if buying: d = int(100 * (a - b) / a)

if selling: d = int(100 * (b - a) / b)

clampedDisposition = clamp int(npcDisposition) to [0..100]

dispositionTerm = fDispositionMod * (clampedDisposition - 50)

pcTerm = (dispositionTerm + pcMercantile + 0.1 * pcLuck + 0.2 * pcPersonality) * pcFatigueTerm

npcTerm = (npcMercantile + 0.1 * npcLuck + 0.2 * npcPersonality) * npcFatigueTerm

x = **fBargainOfferMulti** * d + fBargainOfferBase

if buying: x += abs(int(pcTerm - npcTerm))

if selling: x += abs(int(npcTerm - pcTerm))

roll 100, if roll <= x then trade is accepted

adjust npc temporary disposition by iBarterSuccessDisposition or iBarterFailDisposition

fDispositionMod

fDispositionMod - модификатор который определяет сумму возможного торга.

Значение **по умолчанию**: 1

Код из OpenMW:

all prices are negative when player is buying, positive when player is selling

accept if playerOffer <= merchantOffer (same for buy and sell)

if npc is a creature: reject (no haggle)

a = abs(merchantOffer)

b = abs(playerOffer)

if buying: d = int(100 * (a - b) / a)

if selling: d = int(100 * (b - a) / b)

clampedDisposition = clamp int(npcDisposition) to [0..100]

dispositionTerm = **fDispositionMod** * (clampedDisposition - 50)

pcTerm = (dispositionTerm + pcMercantile + 0.1 * pcLuck + 0.2 * pcPersonality) * pcFatigueTerm

npcTerm = (npcMercantile + 0.1 * npcLuck + 0.2 * npcPersonality) * npcFatigueTerm

x = fBargainOfferMulti * d + fBargainOfferBase

if buying: x += abs(int(pcTerm - npcTerm))

if selling: x += abs(int(npcTerm - pcTerm))

roll 100, if roll <= x then trade is accepted

adjust npc temporary disposition by iBarterSuccessDisposition or iBarterFailDisposition

fPersonalityMod

fPersonalityMod - это параметр который определяет эффективность *Привлекательности*.

Значение по умолчанию: 5

Код из OpenMW:

```
persTerm = personality / fPersonalityMod
luckTerm = luck / fLuckMod
repTerm = reputation * fReputationMod
levelTerm = level * fLevelMod
fatigueTerm = fFatigueBase - fFatigueMult*(1 - normalisedFatigue)
# where normalisedFatigue is a function of fatigue. empty fatigue bar -> 0.0, full fatigue bar -> 1.0
# note fatigueTerm is normally 1.25 at full fatigue.
```

fLuckMod

fLuckMod - модификатор удачи, используется во многих формулах.

Значение **по умолчанию: 10**

Код из OpenMW:

```
persTerm = personality / fPersonalityMod
luckTerm = luck / fLuckMod
repTerm = reputation * fReputationMod
levelTerm = level * fLevelMod
fatigueTerm = fFatigueBase - fFatigueMult*(1 - normalisedFatigue)
# where normalisedFatigue is a function of fatigue. empty fatigue bar -> 0.0, full fatigue bar -> 1.0
# note fatigueTerm is normally 1.25 at full fatigue.
```

fReputationMod

fReputationMod - модификатор глобальной репутации.

Значение по умолчанию: 1

Код из OpenMW:

```
persTerm = personality / fPersonalityMod
luckTerm = luck / fLuckMod
repTerm = reputation * fReputationMod
levelTerm = level * fLevelMod
fatigueTerm = fFatigueBase - fFatigueMult*(1 - normalisedFatigue)
# where normalisedFatigue is a function of fatigue. empty fatigue bar -> 0.0, full fatigue bar -> 1.0
# note fatigueTerm is normally 1.25 at full fatigue.
```

fLevelMod

fLevelMod - модификатор уровня, используется в некоторых формулах

Значение **по умолчанию: 5**

Код из OpenMW:

```
persTerm = personality / fPersonalityMod  
luckTerm = luck / fLuckMod  
repTerm = reputation * fReputationMod  
levelTerm = level * fLevelMod  
fatigueTerm = fFatigueBase - fFatigueMult*(1 - normalisedFatigue)  
# where normalisedFatigue is a function of fatigue. empty fatigue bar -> 0.0, full fatigue bar -> 1.0  
# note fatigueTerm is normally 1.25 at full fatigue.
```

fBribe10Mod

fBribe10Mod - определяет сколько отношения базово повысится за взятку в 10 золотых.

Значение по умолчанию: **35**

fBribe100Mod

fBribe100Mod - определяет сколько отношения базово повысится за взятку в 100 золотых

Значение по умолчанию: **75**

fBribe1000Mod

fBribe1000Mod - определяет сколько отношения базово повысится за взятку в 1000 золотых

Значение по умолчанию: 150

fPerDieRollMult

fPerDieRollMult - модификатор изменения отношения при использовании Убеждения.

Значение по умолчанию: 0.3

Код из OpenMW:

Admire

```
target1 = max(iPerMinChance, target1)
roll 100, win if roll <= target1
c = int(fPerDieRollMult * (target1 - roll))
x = max(iPerMinChance, c) on success, c on fail
```

Intimidate

```
target2 = max(iPerMinChance, target2)
roll 100, win if roll <= target2
```

```
if roll != target2:
    r = int(target2 - roll)
else:
    r = 1
```

```
if roll <= target2:
    s = int(r * fPerDieRollMult * fPerTempMult)
    flee = max(iPerMinChance, s)
    fight = min(-iPerMinChance, -s)
```

```
c = -abs(int(r * fPerDieRollMult))
if success:
    if abs(c) < iPerMinChance:
        x = 0, y = -iPerMinChance # bug, see comments
    else:
        x = -int(c * fPerTempMult), y = c
else fail:
    x = int(c * fPerTempMult), y = c
```

Taunt

```
target1 = max(iPerMinChance, target1)
roll 100, win if roll <= target1
```

```
c = abs(int(target1 - roll))
```

```
if roll <= target1:
    s = c * fPerDieRollMult * fPerTempMult
    flee = min(-iPerMinChance, int(-s))
    fight = max(iPerMinChance, int(s))
```

```
x = int(-c * fPerDieRollMult)
```

```
if success and abs(x) < iPerMinChange:
    x = -iPerMinChange
```


Bribe

target3 = max(iPerMinChance, target3)

roll 100, win if roll <= target3

*c = int((target3 - roll) * **fPerDieRollMult**)*

x = max(iPerMinChange, c) on success, c on fail

fPerTempMult

fPerTempMult - модификатор влияющий на временное изменение отношений с NPC, так же влияет на изменение flee и fight. Используется в нескольких формулах.

Значение по умолчанию: 1

iPerMinChance

iPerMinChance - минимальный шанс на успех Убеждения

Значение **по умолчанию: 5**

iPerMinChange

iPerMinChange - минимальное значение на которое изменится отношение при убеждении.

Значение по умолчанию: 10

fSpecialSkillBonus

fSpecialSkillBonus - это параметр, который определяет, насколько быстро повышаются *Классовые* навыки персонажа. (Меньше значит быстрее)

Значение **по умолчанию: 0.8**

Код из OpenMW:

```
progress[skill] += skillGain
progressRequirement = 1 + playerSkills[skill]

if skill in player.majorSkills:
    progressRequirement *= fMajorSkillBonus
elif skill in player.minorSkills:
    progressRequirement *= fMinorSkillBonus
elif skill in player.miscSkills:
    progressRequirement *= fMiscSkillBonus

if skill in player.class.specialization.skills:
    progressRequirement *= fSpecialSkillBonus

if int(progress[skill]) >= int(progressRequirement):
    progress[skill] = 0
    playerSkills[skill] increased by 1, triggering further functions
```

fMajorSkillBonus

fMajorSkillBonus - это параметр, который определяет, насколько быстро повышаются *Главные навыки* персонажа. (Меньше значит быстрее)

Значение **по умолчанию: 0.75**

Код из OpenMW:

```
progress[skill] += skillGain
progressRequirement = 1 + playerSkills[skill]

if skill in player.majorSkills:
    progressRequirement *= fMajorSkillBonus
elif skill in player.minorSkills:
    progressRequirement *= fMinorSkillBonus
elif skill in player.miscSkills:
    progressRequirement *= fMiscSkillBonus

if skill in player.class.specialization.skills:
    progressRequirement *= fSpecialSkillBonus

if int(progress[skill]) >= int(progressRequirement):
    progress[skill] = 0
    playerSkills[skill] increased by 1, triggering further functions
```

fMinorSkillBonus

fMinorSkillBonus - это параметр, который определяет, насколько быстро повышаются *Важные навыки* персонажа. (Меньше значит быстрее)

Значение **по умолчанию: 1**

Код из OpenMW:

```
progress[skill] += skillGain
progressRequirement = 1 + playerSkills[skill]

if skill in player.majorSkills:
    progressRequirement *= fMajorSkillBonus
elif skill in player.minorSkills:
    progressRequirement *= fMinorSkillBonus
elif skill in player.miscSkills:
    progressRequirement *= fMiscSkillBonus

if skill in player.class.specialization.skills:
    progressRequirement *= fSpecialSkillBonus

if int(progress[skill]) >= int(progressRequirement):
    progress[skill] = 0
    playerSkills[skill] increased by 1, triggering further functions
```

fMiscSkillBonus

fMiscSkillBonus - это параметр, который определяет, насколько быстро повышаются *Маловажные навыки* персонажа. (Меньше значит быстрее)

Значение **по умолчанию: 1.25**

Код из OpenMW:

```
progress[skill] += skillGain
progressRequirement = 1 + playerSkills[skill]

if skill in player.majorSkills:
    progressRequirement *= fMajorSkillBonus
elif skill in player.minorSkills:
    progressRequirement *= fMinorSkillBonus
elif skill in player.miscSkills:
    progressRequirement *= fMiscSkillBonus

if skill in player.class.specialization.skills:
    progressRequirement *= fSpecialSkillBonus

if int(progress[skill]) >= int(progressRequirement):
    progress[skill] = 0
    playerSkills[skill] increased by 1, triggering further functions
```


iAlarmKilling

НЕ ИСПОЛЬЗУЕТСЯ В ОРИГИНАЛЬНОМ ДВИЖКЕ MORROWIND И OPENMW

iAlarmKilling -

Значение по умолчанию: 90

iAlarmAttack

НЕ ИСПОЛЬЗУЕТСЯ В ОРИГИНАЛЬНОМ ДВИЖКЕ MORROWIND И OPENMW

iAlarmAttack -

Значение по умолчанию: 50

iAlarmStealing

НЕ ИСПОЛЬЗУЕТСЯ В ОРИГИНАЛЬНОМ ДВИЖКЕ MORROWIND И OPENMW

iAlarmStealing -

Значение по умолчанию: 1

iAlarmPickPocket

НЕ ИСПОЛЬЗУЕТСЯ В ОРИГИНАЛЬНОМ ДВИЖКЕ MORROWIND И OPENMW

iAlarmPickPocket -

Значение по умолчанию: 20

iAlarmTrespass

НЕ ИСПОЛЬЗУЕТСЯ В ОРИГИНАЛЬНОМ ДВИЖКЕ MORROWIND И OPENMW

iAlarmTrespass -

Значение по умолчанию: 5

fAlarmRadius

fAlarmRadius - определяет радиус внутри которого один враждебный NPC увидевший игрока может оповестить своих союзников о игроке.

Значение по умолчанию: **2000**

iCrimeKilling

iCrimeKilling - определяет размер штрафа за убийство.

Значение **по умолчанию: 1000**

iCrimeAttack

iCrimeAttack - определяет размер штрафа за нападение

Значение по умолчанию: **40**

fCrimeStealing

fCrimeStealing - определяет модификатор для штрафа за воровство.

Значение **по умолчанию: 1**

Код из OpenMW:

killling: iCrimeKilling
assault: iCrimeAttack
pickpocket: iCrimePickpocket
trespass: iCrimeTrespass
theft: **fCrimeStealing** * stack value
werewolf transformation: iWerewolfBounty

iCrimePickPocket

iCrimePickPocket - определяет размер штрафа за карманную кражу

Значение по умолчанию: **25**

Код из OpenMW:

killling: iCrimeKilling
assault: iCrimeAttack
pickpocket: **iCrimePickpocket**
trespass: iCrimeTrespass
theft: fCrimeStealing * stack value
werewolf transformation: iWerewolfBounty

iCrimeTrespass

iCrimeTrespass - определяет модификатор для штрафа за попытку спать на чужой кровати.

Значение **по умолчанию: 5**

Код из OpenMW:

killing: iCrimeKilling
assault: iCrimeAttack
pickpocket: iCrimePickpocket
trespass: **iCrimeTrespass**
theft: fCrimeStealing * stack value
werewolf transformation: iWerewolfBounty

iCrimeTreshold

iCrimeThreshold - определяет базовый параметр для суммы штрафа после которой стражники будут убивать игрока без диалога.

Значение **по умолчанию: 1000**

Код из OpenMW:

```
if bounty >= iCrimeThreshold:
    if bounty >= iCrimeThreshold * iCrimeThresholdMultiplier:
        guards will immediately initiate combat and cannot be engaged in dialogue
    else:
        guards in range will run to the player and initiate dialogue
```

iCrimeTresholdMultiplier

iCrimeTresholdMultiplier - определяет модификатор параметра для суммы штрафа после которой стражники будут убивать игрока без диалога.

Значение **по умолчанию: 10**

Код из OpenMW:

```
if bounty >= iCrimeThreshold:
    if bounty >= iCrimeThreshold * iCrimeTresholdMultiplier:
        guards will immediately initiate combat and cannot be engaged in dialogue
    else:
        guards in range will run to the player and initiate dialogue
```

fCrimeGoldDiscountMult

fCrimeGoldDiscountMult - модификатор для скидки при оплате штрафа в гильдии воров.

Значение по умолчанию: **0.5**

fCrimeGoldTurnInMult

fCrimeGoldTurnInMult - Множитель для уровня преступности, когда персонаж сдается.

Значение по умолчанию: 0.9

iFightAttack

iFightAttack - модификатор для настройки боя.

Значение **по умолчанию: 100**

iFightAttacking

iFightAttacking - Модификатор настройки боя для других NPC, которым стало известно об атаке игрока на NPC.

Значение по умолчанию: 50

iFightDistanceBase

iFightDistanceBase - параметр отвечающий за расстояние на котром NPC и существа будут начинать наносить удары.

Значение **по умолчанию: 20**

Код из OpenMW:

$\text{Fight Setting} = (\text{Basic Fight Setting} + \text{iFightDistanceBase} - \text{Distance} * \text{fFightDistanceMultiplier}).$

iFightDistanceMultiplier

iFightDistanceMultiplier - параметр отвечающий за множитель для расстояние на котром NPC и существа будут начинать наносить удары.

Значение **по умолчанию: 0.005**

Код из OpenMW:

$\text{Fight Setting} = (\text{Basic Fight Setting} + \text{iFightDistanceBase} - \text{Distance} * \text{fFightDistanceMultiplier}).$

iFightAlarmMult
НЕ ИСПОЛЬЗУЕТСЯ

Значение по умолчанию: 1

iFightDispMult

iFightDispMult - параметр для расчета отношения при котором NPC нападет на игрока

Значение **по умолчанию: 0.2**

Код из OpenMW:

$\text{Fight Setting} = (\text{Basic Fight Setting} + (50 - \text{Disposition}) * \mathbf{fFightDispMult})$.

fFightStealing

iFightStealing - Модификатор настройки боя для NPC, которым стало известно о соответствующем преступлении игрока.

Значение **по умолчанию: 50**

iFightPickpocket

iFightPickpocket - Модификатор настройки боя для NPC, которым стало известно о соответствующем преступлении игрока.

Значение по умолчанию: **25**

iFightTrespass

iFightPickpocket - Модификатор настройки боя для NPC, которым стало известно о соответствующем преступлении игрока.

Значение по умолчанию: **25**

iFightKilling

FightKilling - Модификатор настройки боя для NPC, которым стало известно о соответствующем преступлении игрока.

Значение по умолчанию: 50

iFlee

НЕ ИСПОЛЬЗУЕТСЯ В ОРИГИНАЛЬНОМ ДВИЖКЕ MORROWIND И OPENMW

iFightFlee -

Значение по умолчанию: 0

iGreetDistanceMultiplier

iGreetDistanceMultiplier - множитель дистанции на которой NPC приветствуют вас фразой

Значение по умолчанию: 6

iGreetDuration

ТРЕБУЕТ ПРОВЕРКИ

iGreetDuration - минимальное количество секунд в течении которых NPC занят приветствием игрока.

Значение по умолчанию: 4

fGreetDistanceReset

fGreetDistanceReset - На какое расстояние должен отойти игрок от NPC для того чтобы NPC снова поприветствовал игрока.

Значение по умолчанию: **512**

fIdleChanceMultiplier

fIdleChanceMultiplier - модификатор шанса для пороизнесения Idle фразы.

Значение по умолчанию: **0.75**

fSneakUseDist

ТРЕБУЕТ ПРОВЕРКИ

fSneakUseDist - Помогает определить может ли игрок спрятаться.

Значение по умолчанию: **500**

fSneakUseDelay

fSneakUseDelay - Помогает определить насколько быстро появляется иконка скрытности.

Значение по умолчанию: 1

fSneakDistanceBase

fSneakDistanceBase - базовый шанс для возможности скрыться.

Значение по умолчанию: 0.5

известная формула

PC sneak chance = (**fSneakDistanceBase** + Distance * fSneakDistanceMultiplier).

Код из OpenMW:

if sneaking:

sneakTerm = fSneakSkillMult * sneak + 0.2 * agility + 0.1 * luck + bootWeight * fSneakBootMult

else:

sneakTerm = 0

fatigueTerm = fFatigueBase - fFatigueMult*(1 - normalisedFatigue)

where normalisedFatigue is a function of fatigue. empty fatigue bar -> 0.0, full fatigue bar -> 1.0

distTerm = **fSneakDistanceBase** + fSneakDistanceMultiplier * Distance

x = sneakTerm * distTerm * fatigueTerm + chameleon (+ 100 if invisible)

fSneakDistanceMultiplier

fSneakDistanceMultiplier - Модификатор для возможности скрыться.

Значение по умолчанию: **0.002**

известная формула

PC sneak chance = (fSneakDistanceBase + Distance * **fSneakDistanceMultiplier**).

Код из OpenMW:

if sneaking:

sneakTerm = fSneakSkillMult * sneak + 0.2 * agility + 0.1 * luck + bootWeight * fSneakBootMult

else:

sneakTerm = 0

fatigueTerm = fFatigueBase - fFatigueMult*(1 - normalisedFatigue)

where normalisedFatigue is a function of fatigue. empty fatigue bar -> 0.0, full fatigue bar -> 1.0

distTerm = fSneakDistanceBase + **fSneakDistanceMultiplier** * Distance

x = sneakTerm * distTerm * fatigueTerm + chameleon (+ 100 if invisible)

fSneakSpeedMultiplier

fSneakSpeedMultiplier - модификатор скорости в режиме скрытности.

Значение по умолчанию: **0.75**

fSneakViewMult

fSneakViewMult - модификатор для NPC на вероятность обнаружить игрока.

Значение по умолчанию: **1.5**

Код из OpenMW:

```
npcTerm = npcSneak + 0.2 * npcAgility + 0.1 * npcLuck - npcBlind  
npcFatigueTerm = fFatigueBase - fFatigueMult * (1 - normalisedFatigue)
```

```
using NPC normalisedFatigue
```

```
if PC is behind NPC (180 degrees):
```

```
    y = npcTerm * npcFatigueTerm * fSneakNoViewMult
```

```
else:
```

```
    y = npcTerm * npcFatigueTerm * fSneakViewMult
```

fSneakNoViewMult

fSneakNoViewMult - модификатор для NPC на вероятность обнаружить игрока.

Значение по умолчанию: **0.5**

Код из OpenMW:

```
npcTerm = npcSneak + 0.2 * npcAgility + 0.1 * npcLuck - npcBlind  
npcFatigueTerm = fFatigueBase - fFatigueMult * (1 - normalisedFatigue)
```

```
using NPC normalisedFatigue
```

```
if PC is behind NPC (180 degrees):
```

```
    y = npcTerm * npcFatigueTerm * fSneakNoViewMult
```

```
else:
```

```
    y = npcTerm * npcFatigueTerm * fSneakViewMult
```

fSneakSkillMult

fSneakSkillMult - модификатор для навыка скрытности.

Значение по умолчанию: 1

Код из OpenMW:

```
if sneaking:  
    sneakTerm = fSneakSkillMult * sneak + 0.2 * agility + 0.1 * luck + bootWeight * fSneakBootMult  
else:  
    sneakTerm = 0
```

```
fatigueTerm = fFatigueBase - fFatigueMult*(1 - normalisedFatigue)  
where normalisedFatigue is a function of fatigue. empty fatigue bar -> 0.0, full fatigue bar -> 1.0
```

```
distTerm = fSneakDistanceBase + fSneakDistanceMultiplier*dist
```

fSneakBootsMult

fSneakBootsMult - множитель, используемый для вычисления способности скрытно перемещаться в зависимости от веса обуви, надетой на персонажа.

Значение **по умолчанию**: -1

Известная формула:

PC sneak chance = (Basic Chance + Boot Weight * **fSneakBootMult**).

fCombatDistance

fCombatDistance - параметр отвечающий за дистанцию на которой противники с оружием начинает вас атаковать

Значение **по умолчанию: 128**

Код из OpenMW:

$\text{Dist} = (\text{Weapon Reach} * \text{fCombatDistance}).$

fCombatAngleXY

fCombatAngleXY - Игрок нанесет удар NPC, если будете атаковать место, которое находится в $(fCombatAngleXY / 2)$ градусах слева/справа от него.

Значение по умолчанию: 60

fCombatAngleZ

fCombatAngleZ - Игрок нанесет удар NPC, если будете атаковать место, которое находится на fCombatAngleZ градусов вверх/вниз от него.

Значение по умолчанию: 60

fCombatForceSideAngle

НЕ ИСПОЛЬЗУЕТСЯ В ОРИГИНАЛЬНОМ ДВИЖКЕ MORROWIND И OPENMW

fCombatForceSideAngle

Значение по умолчанию: 30

fCombatTorsoSideAngle

НЕ ИСПОЛЬЗУЕТСЯ В ОРИГИНАЛЬНОМ ДВИЖКЕ MORROWIND И OPENMW

fCombatTorsoSideAngle -

Значение по умолчанию: **0.3**

fCombatTorsoStartPercent

НЕ ИСПОЛЬЗУЕТСЯ В ОРИГИНАЛЬНОМ ДВИЖКЕ MORROWIND И OPENMW

fCombatTorsoStartPercent

Значение по умолчанию: 0.8

fCombatTorsoStopPercent

НЕ ИСПОЛЬЗУЕТСЯ В ОРИГИНАЛЬНОМ ДВИЖКЕ MORROWIND И OPENMW

fCombatTorsoStopPercent

Значение по умолчанию: -90

fCombatBlockLeftAngle

fCombatBlockLeftAngle - определяет угол с левой стороны от центра экрана на которой игрок будет блокировать удары щитом.

Значение по умолчанию: **-90**

fCombatBlockRightAngle

fCombatBlockRightAngle - определяет угол с правой стороны от центра экрана на которой игрок будет блокировать удары щитом.

Значение по умолчанию: 30

fCombatDelayCreature

fCombatDelayCreature - Базовое значение задержки между ударами существ.

Значение **по умолчанию: 0.1**

Код из OpenMW:

if actor is an npc:

 baseDelay = fCombatDelayNPC

else:

 baseDelay = **fCombatDelayCreature**

delay = min(baseDelay + 0.01 * rand 100, baseDelay + 0.9)

if actor is in range to attack with current weapon:

 actor will initiate a swing/fire a missile if time since end of last attack \geq delay

fCombatDelayNPC

fCombatDelayNPC - Базовое значение задержки между ударами у NPC.

Значение **по умолчанию: 0.1**

Код из OpenMW:

if actor is an npc:

 baseDelay = **fCombatDelayNPC**

else:

 baseDelay = fCombatDelayCreature

delay = min(baseDelay + 0.01 * rand 100, baseDelay + 0.9)

if actor is in range to attack with current weapon:

 actor will initiate a swing/fire a missile if time since end of last attack >= delay

fAlMeleeWeaponMult

ТРЕБУЕТ ПРОВЕРКИ

fAlMeleeWeaponMult - скорее всего определяет предпочтение у NPC для оружия ближнего боя.

Значение по умолчанию: **2**

fAlRangeMeleeWeaponMult

ТРЕБУЕТ ПРОВЕРКИ

fAlRangeMeleeWeaponMult - скорее всего определяет предпочтение у NPC для оружия дальнего боя.

Значение по умолчанию: **5**

fAlMagicSpellMult

ТРЕБУЕТ УТОЧНЕНИЯ

fAlMagicSpellMult - скорее всего определяет предпочтение у NPC для заклинаний на себя или при касании.

Значение по умолчанию: 3

fAIRangeMagicSpellMult

ТРЕБУЕТ УТОЧНЕНИЯ

fAIRangeMagicSpellMult - скорее всего определяет предпочтение у NPC для заклинаний на удаленную цель.

Значение по умолчанию: 5

fAlMeleeArmorMult

fAlMeleeArmorMult - Информация не найдена. Требуется изучения.

Значение по умолчанию: 1

fAlMeleeSumWeaponMult

НЕ ИСПОЛЬЗУЕТСЯ В ОРИГИНАЛЬНОМ ДВИЖКЕ MORROWIND И OPENMW

fAlMeleeSumWeaponMult

Значение по умолчанию: 1

fAI_FleeHealthMult

fAI_FleeHealthMult - модификатор определяющий тенденцию существ и NPC к побегу при уменьшении здоровья.

Значение по умолчанию: 7

fAIFFleeFleeMult

fAIFFleeFleeMult - модификатор базового значения для побега в бою.

Значение по умолчанию: **0.3**

fPickPocketMod

fPickPocketMod - модификатор для определения того насколько будет сложно украсть вещь в зависимости от ее цены.

Значение **по умолчанию: 0.3**

Код из OpenMW:

```
fatigueTerm = fFatigueBase - fFatigueMult*(1 - normalisedFatigue)
#where normalisedFatigue is a function of fatigue. empty fatigue bar -> 0.0, full fatigue bar -> 1.0
#note fatigueTerm is normally 1.25 at full fatigue.
```

```
#checks the whole stack no matter how many you try to take
#note: filled soulgems have the value of an empty soulgem due to a missing calculation
stackValue = itemValue * itemsInStack
valueTerm = 10 * fPickPocketMod * stackValue
```

```
x = (0.2 * pcAgility + 0.1 * pcLuck + pcSneak) * fatigueTerm
y = (valueTerm + npcSneak + 0.2 * npcAgilityTerm + 0.1 * npcLuckTerm) * npcFatigueTerm
t = x - y + x # yes, that is what it does
```

```
if t < pcSneak / iPickMinChance:
    roll 100, win if roll <= int(pcSneak / iPickMinChance)
else:
    t = min(iPickMaxChance, t)
    roll 100, win if roll <= int(t)
```

fSleepRandMod

fSleepRandMod - Модификатор который изменяет вероятность того, что отдых игрока будет прерван существами.

Значение **по умолчанию: 0.25**

Код из OpenMW в котором исправлена ошибка с interruptingCreatures.

if resting in an exterior cell and the region has a sleep creature leveled list:

 x = roll hoursRested

 y = **fSleepRandMod** * hoursRested

 if x < y:

 interruptAtHoursRemaining = int(fSleepRestMod * hoursRested)

 interruptingCreatures = max(1, roll iNumberCreatures)

 sleep will only last (hoursRested - interruptAtHoursRemaining) hours

 unless interruptAtHoursRemaining == 0, then no interruption occurs # contains bug

 sleep will be interrupted with 1 creature from the region leveled list # contains bug

fSleepRestMod

fSleepRestMod - модификатор вероятности быть разбуженным в зависимости от количества часов отдыха

Значение **по умолчанию: 0.3**

Код из OpenMW в котором исправлена ошибка с interruptingCreatures.

if resting in an exterior cell and the region has a sleep creature leveled list:

 x = roll hoursRested

 y = fSleepRandMod * hoursRested

 if x < y:

 interruptAtHoursRemaining = int(f**SleepRestMod** * hoursRested)

 interruptingCreatures = max(1, roll iNumberCreatures)

 sleep will only last (hoursRested - interruptAtHoursRemaining) hours

 unless interruptAtHoursRemaining == 0, then no interruption occurs # contains bug

 sleep will be interrupted with 1 creature from the region leveled list # contains bug

iNumberCreatures

ТРЕБУЕТ ПРОВЕРКИ

iNumberCreatures - количество существ которые могут появиться когда отдых прерван.

Значение **по умолчанию: 1**

Код из OpenMW:

if resting in an exterior cell and the region has a sleep creature leveled list:

 x = roll hoursRested

 y = fSleepRandMod * hoursRested

 if x < y:

 interruptAtHoursRemaining = int(fSleepRestMod * hoursRested)

 interruptingCreatures = max(1, roll **iNumberCreatures**)

 sleep will only last (hoursRested - interruptAtHoursRemaining) hours

 unless interruptAtHoursRemaining == 0, then no interruption occurs # contains bug

 sleep will be interrupted with 1 creature from the region leveled list # contains bug

fAudioDefaultMinDistance

fAudioDefaultMinDistance - требует изучения

Значение по умолчанию: 5

fAudioDefaultMaxDistance

fAudioDefaultMaxDistance - требует изучения

Значение по умолчанию: 40

fAudioVoiceDefaultMinDistance

fAudioVoiceDefaultMinDistance - требует изучения

Значение по умолчанию: 10

fAudioVoiceDefaultMaxDistance

fAudioVoiceDefaultMaxDistance - требует изучения

Значение по умолчанию: 60

fAudioMinDistanceMult

fAudioMinDistanceMult - требует изучения

Значение по умолчанию: 20

fAudioMaxDistanceMult

fAudioMaxDistanceMult - требует изучения

Значение по умолчанию: 50

fNPCHealthBarTime

fNPCHealthBarTime - минимальное время на которое появится полоска здоровья противника.

Значение по умолчанию: 3

fNPCHealthBarFade

fNPCHealthBarFade - определяет время в секундах через которое исчезает полоска здоровья протиника после окончания боя.

Значение по умолчанию: **0.5**

fDifficultyMult

fDifficultyMult - модификатор сложности, влияет на наносимый урон игроку

Значение **по умолчанию: 5**

Код из OpenMW

```
difficulty : int [-100..100]
difficultyTerm = 0.01 * difficulty

if defender is player:
    if difficultyTerm > 0:
        x = fDifficultyMult * difficultyTerm
    else:
        x = difficultyTerm / fDifficultyMult
elif attacker is player:
    if difficultyTerm > 0:
        x = -difficultyTerm / fDifficultyMult
    else:
        x = fDifficultyMult * -difficultyTerm
else:
    x = 0

damage *= 1 + x
```

fMagicDetectRefreshRate

fMagicDetectRefreshRate - частота обновления меток для заклинания *Обнаружить чары*.

Значение по умолчанию: **0.016667**

fMagicStartIconBlink

fMagicDetectRefreshRate - отвечает за сколько секунд до окончания эффекта, иконка на панели эффектов начнет становиться прозрачной

Значение по умолчанию: 3

fMagicCreatureCastDelay

ТРЕБУЕТ ПРОВЕРКИ, ВОЗМОЖНО НЕ ИСПОЛЬЗУЕТСЯ

fMagicCreatureCastDelay - это параметр, управляющий задержкой между использованием способностей магии существами.

Значение по умолчанию: 1.5

fDiseaseXferChance

fDiseaseXferChance - отвечает за шанс передачи болезни при контакте с зараженным существом

Значение по умолчанию: 2.5

Часть кода из OpenMW

```
if defender is not player: return

for each disease in attacker.activeSpells:
    if any of the disease.effects is corpus:
        resist = 1 - 0.01 * defender.resistCorpus
    elif spell.castType == disease:
        resist = 1 - 0.01 * defender.resistDisease
    elif spell.castType == blight:
        resist = 1 - 0.01 * defender.resistBlight
    else:
        continue

if player already has the disease:
    continue

x = int(fDiseaseXferChance * 100 * resist)
if roll 10000 < x:
    defender acquires disease
    display message sMagicContractDisease with disease name
```

fElementalShieldMult

fElementalShieldMult - константа которая контролирует, как много урона возвращают элементальные щиты.

Значение **по умолчанию: 0.1**

Код из OpenMW:

```
saveTerm = attacker.destruction + 0.2 * attacker.willpower + 0.1 * attacker.luck  
saveTerm *= 1.25 * attacker.normalisedFatigue
```

```
for each elementalShield in defender.activeEffects:
```

```
    x = max(0, saveTerm - roll float 100)
```

```
    x = min(100, x + attacker.elementalResist[elementalShield.element])
```

```
    x = fElementalShieldMult * elementalShield.magnitude * (1 - 0.01 * x)
```

```
    attacker takes x damage
```

fMagicSunBlockedMult

fMagicSunBlockedMult - отвечает модификатор урона для заклинания *Удар Солнца*.

Значение по умолчанию: **0.5**

Код из OpenMW:

```
if current cell is interior only: return 0

if gamehour <= Weather.SunriseTime or gamehour >= Weather.SunsetTime + Weather.SunsetDuration:
    sunRisen = 0
elif gamehour <= Weather.SunriseTime + Weather.SunriseDuration:
    sunRisen = (gamehour - Weather.SunriseTime) / Weather.SunriseDuration
elif gamehour > Weather.SunsetTime:
    sunRisen = 1 - ((gamehour - Weather.SunsetTime) / Weather.SunsetDuration)
else:
    sunRisen = 1

# transition is 0 at the start of a weather change and 1.0 at the end
# note that CloudsMaximumPercent is not actually a percentage
if weather is changing and transition < nextWeather.CloudsMaximumPercent:
    t = transition / nextWeather.CloudsMaximumPercent
    sunVisibility = (1-t) * currentWeather.GlareView + t * nextWeather.GlareView
else:
    sunVisibility = currentWeather.GlareView

damageScale = max(sunVisibility * sunRisen, fMagicSunBlockedMult * sunRisen)
damageScale = max(0, min(damage, 1))

return damageScale * spelleffect.magnitude
```

Конец