«Робимо випадкову ймовірність знаходження броні в Сталкерів з випадковою ступеню поломки»

Автор уроку: **Infernis**

Слава Україні! Раніше, коли я тільки починав працювати над модами, великою проблемою для мене раніше було зробити нормальну робочу систему зняття броні з трупів. Хоча, дуже сильно хотілося))0))0)))0)))0)) Тому що я ніколи не розумів цього ідіотизму, коли перед тобою лежить дохлий мужик в екзоскелеті, а ти не можеш стягнути з нього цю екзу, і ходиш у якій занюханій «Зорі» або куртці. Особливо це бісило в ТЧ та деяких модах на ЗП, але хрін з ним.

Але коли збираєшся сідати та «позичати» це з інших модів, стикаєшся з другою проблемою – моди, які дозволяють стягувати броню з трупів, як правило, або притягують разом із собою купу іншого непотребу по типу нової броні з відверто страшними іконками, яку, до того ж, зможу зробити і я сам, при чому набагато краще – або зроблено настільки складно та незрозуміло, що витягувати це все діло собі дорожче, у плані стабільності моду в тому ж числі – або банально працює не так, як повинно, і ти розумієш, що краще вже без цього.

Гадаю, всі намагалися це все прописати через **death_items_бла_бла_бла.ltx**, і успішно стикнулися з проблемою, коли в персонажа, який вдягнений в Зорю, з невеликою імовірністю можуть заспавнитися Зоря, Сєва, протигаз, ще якийсь шолом, шкіряний плащ-пальто, і екзоскелет ОДНОЧАСНО, бо дільожка іде не по візуалам у які вони вдягнені, а по угрупованням. Або прописати це все діло в **character_desc_якийсь_там.xml**, щоби в конкретних НПС спавнилася конкретна броня, але стикнулися вже з іншою проблемою – мало того, що броня завжди спавниться з імовірністю 100%, так ще й в ідеальному стані, і це вже суттєво б’є по балансу гри! Такий ось пат виходить, або використовувати скріпти з якихось сайтів і молитися щоби не вилетіло, або використовувати неідеальні способи, які і так відомі всім.

Але цього разу я зроблю все набагато краще, бо буквально на момент написання цієї статті (5 серпня 2024 року) я винайшов ідеальний метод для написання конфігів і скриптів у моді, щоби з персонажів можна було стягувати броню, із наступними особливостями:

1) Ми можемо повністю регулювати, в яких саме персонажів буде броня, а в яких – ні. Ми можемо зробити це як для випадкових НПС, так і для квестових окремо;

2) Ми можемо повністю регулювати, яка саме броня буде в персонажів;

3) Броня з’являтиметься з регульованою нами ймовірністю, а не 100%;

4) При цьому, броня матиме різний рівень поломки під час знаходження в НПС, як, власне, і зброя.

І робитимемо ми це все протягом десяти хвилин максимум. Ну що ж, почнемо!

Після розпакування геймдати в власній папці з модом створюємо дві нові папки – **«configs»** та **«scripts»** відповідно під конфіги та скріпти. У конфігах робимо одну єдину папку – **«gameplay»**. Хто вже здогадався, молодці) Ми реалізовуватимемо перший спосіб через характер деск як більш стабільний, але додамо до нього все, чого не вистачало. Тому в папку **«gameplay»** з розпакованої геймдати перекидуємо файл **character_desc_general.xml**. У цьому файлі зберігаються всі базові профілі НПС та те, яка зброя/броня/предмети/ще якась фігня в них спавниться. І всюди ми бачимо приблизно ось таку картину:

```
  <specific_character id="sim_default_stalker_2_default_1" team_default = "1">
    <name>GENERATE_NAME_stalker</name>
    <icon>ui_inGame2_neutral_2_mask</icon>
    <map_icon x="1" y="0"></map_icon>
    <bio>Опытный сталкер. Детальная информация отсутствует.</bio>

    <class>sim_default_stalker_2</class>
    <community>stalker</community> <terrain_sect>stalker_terrain</terrain_sect>
    <snd_config>characters_voice\human_02\stalker\</snd_config>

    <rank>40</rank>
  <money min="1500" max="3500" infinitive="0"/>
    <reputation>0</reputation>

    <visual>actors\stalker_neutral\stalker_neutral_2_mask</visual>
    <supplies>
      [spawn] \n

      wpn_l85 \n
      ammo_5.56x45_ap = 1 \n
      wpn_colt1911 \n
      ammo_11.43x23_fmj = 1 \n
      grenade_rgd5 = 2 \n

#include "gameplay\character_items_2.xml"
#include "gameplay\character_food.xml"
#include "gameplay\character_drugs_2.xml"

    </supplies>
#include "gameplay\character_criticals_4.xml"
#include "gameplay\character_dialogs.xml"
  </specific_character>
```

Те що в нас прикрито трикутними дужками з `<supplies>`, є речами, які спавнитимуться в персонажа. І там же ми можемо прописати броню:

```
      stalker_outfit = 1 \n
```

У результаті в нас має з’явитися броня, але, як я вже сказав, ми маємо як мінімум дві проблеми! Нас зараз цікавить саме перша – ймовірність появлення броні в будь-якому разі буде дорівнювати 100%, оскільки поправити ймовірність ми не можемо! Чи можемо?

А ось у чому справа – я забув параметр **«probability»**, який ПРИСУТНІЙ у xml-файлах та може використовуватися в імовірності спавну НПС! Зараз поясню по-людськи:

Якщо пошаритись по конфігах xml-файлів, то можна помітити в файлах накшталт **character_items.xml** вже готові набори предметів, які використовуються в інших профілях НПС. Скажімо так, щоби не прописувати без кінця одні й ті самі предмети в інвентареві НПС, краще заінклудити файл, у якому вони вже прописані. Виглядає це приблизно якось так:

```
guitar_a = 1, prob=1 \n
harmonica_a = 1, prob=1 \n
wpn_binoc = 1, prob=1 \n
detector_simple = 1, prob=0.35 \n
device_torch = 1, prob=1 \n
```

І саме тут ми знаходимо найцікавіше – параметр **prob**! Ось він – цей самий недостающий фрагмент для нормального спавну броні, бо саме **prob** відповідає за те, яка ймовірність того, що цей предмет з’явиться в НПС! Тобто, якщо вказати **prob = 0.9**, то ймовірність появи комбінезону «Зоря» буде 90%, **prob = 0.13** значить що ймовірність 13%, і так далі! Мало того, що пробабіліті – гнучкий параметр, в якому нам необов’язково вказувати або 0% або 10, а ми можемо обрати, наприклад, 9% і воно його не округлить – так ще й налаштовувати ми це можемо не під окремі візуали, а під персонажів! Тепер нам залишається лише прописати цей параметр після самої броні та її кількості.

До речі, не перевіряв, але ризикну припустити, що коли ми спавнимо більше від одного предмету з конкретною ймовірністю, то сама ймовірність появи множиться на кількість предметів, збільшуючи при цьому шанс появи одного предмета. Але я не уявляю, кому прийде в голову спавнити декілька бронежилетів в одного персонажа.

Прописуємо параметр та отримуємо ось це:

```
    stalker_outfit = 1, prob = 0.55 \n
```

Таким чином ми зробили, щоби в персонажа спавнилася броня сталкера з імовірністю 55%. Тепер дублюємо цю строчку кожному з «досвідчених сталкерів», роблячи всюди різні ймовірності, щоби було цікавіше)))) Або як я – просто вказали всюди одну й ту саму, і хай буде.

Ну і давайте зробимо так, щоби в сталкерів у протигазі також була окрема ймовірність на спавн протигазу. Виглядатиме це чудо-юдо приблизно так:

```
  <specific_character id="sim_default_stalker_1_default_5" team_default = "1">
    <name>GENERATE_NAME_stalker</name>
    <icon>ui_inGame2_neutral_2</icon>
    <map_icon x="1" y="0"></map_icon>
    <bio>Опытный сталкер. Детальная информация отсутствует.</bio>

    <class>sim_default_stalker_1</class>
    <community>stalker</community> <terrain_sect>stalker_terrain</terrain_sect>
    <snd_config>characters_voice\human_03\stalker\</snd_config>

    <rank>30</rank>
  <money min="1000" max="2000" infinitive="0"/>
    <reputation>0</reputation>

    <visual>actors\stalker_neutral\stalker_neutral_2</visual>
    <supplies>
      [spawn] \n

      wpn_ak74u = 1 \n
      ammo_5.45x39_fmj = 1 \n
      wpn_hpsa = 1 \n
      ammo_9x19_fmj = 1 \n
      grenade_rgd5 = 1 \n
      stalker_outfit = 1, prob = 0.5 \n
      helm_respirator = 1, prob = 0.75 \n

#include "gameplay\character_items.xml"
#include "gameplay\character_food.xml"
#include "gameplay\character_drugs_2.xml"
    </supplies>
#include "gameplay\character_criticals_4.xml"
#include "gameplay\character_dialogs.xml"
  </specific_character>
```

Таким чином, ми зробили так, щоби в цього сталкера (у якого, зазначу, візуал саме «Зорі» з протигазом!) спавнилася «Зоря» з імовірністю 50%, та протигаз з імовірністю 75%. Але в нас лишилася ще одна проблемка. Справа в тому, що кондішн все ще лишається 100%, що, переводячи на людську мову, значить що кожна з цих бронь спавнитиметься в ідеальному стані. Щоби це виправити, нам потрібно буде поритися в скріптах, і зробити наступну річ: У папку «scripts» з розпакованої геймдати в нашу переміщуємо два файли: **_g.script** та **death_manager.script**. Почнемо, мабуть, із другого.

У файлі **death_manager.script** нас цікавить функція **keep_item**, а саме ось ці строчки:

```
    if isWeapon(item) and not(get_clsid(item)==clsid.wpn_grenade_rgd5_s or
get_clsid(item)==clsid.wpn_grenade_f1_s) then
        set_weapon_drop_condition(item)
        return
    end
```

Ось цей параметр відповідає за те, що під час того, коли НПС помирає, зброя яка знаходиться в його інвентареві починає псуватися. Сама функція виконується одноразово для кожного дохлого НПС, але ступіть поломки визначається в момент його смерті, тому саму функцію можна непогано абузити якщо хочеться взяти шмотку в більш хорошому стані. Також для функції створене подвійне виключення в вигляді гранат, у яких ступіню поломки немає, що в принципі логічно. Зручна штука для СГМу, але хрін з ним.

Скажу прямо – зараз ми можемо продублювати цю функцію, замінивши все, що пов'язане зі зброєю – на те, що пов'язане з бронею – і в результаті отримаємо аналог функції псування зброї, який працюватиме й на броню. І таким чином ми зробимо так, щоби броня після смерті НПС з'являлася не в ідеальному стані, але заждіть. Що це за **isWeapon** такий? Де знаходиться ця функція? Прогортав усі скріпти та витративши немалу кількість часу на пошуки (десь приблизно 4-5 секунд), я дізнався, що функція **isWeapon** знаходиться в файлі **_g.script**, та виглядає наступним чином:

```
function isWeapon(object, class_id)
    local id = class_id or get_clsid(object)
    return weapon_classes[id] == true
end
```

Дана функція **«перевірки на те, що предмет – зброя»** виконується, якщо предмет є елементом таблиці **weapon_classes**, значить потрібно дізнатися, де знаходиться ця таблиця, як виглядає, та створити додаткову таблицю для класів броні. Спойлер – таблиця знаходиться в цьому ж файлі:

```
    weapon_classes = {
        [clsid.wpn_vintorez_s]          = true,
        [clsid.wpn_ak74_s]              = true,
        [clsid.wpn_lr300_s]             = true,
        [clsid.wpn_hpsa_s]              = true,
        [clsid.wpn_pm_s]                = true,
        [clsid.wpn_shotgun_s]           = true,
        [clsid.wpn_auto_shotgun_s]      = true,
        [clsid.wpn_bm16_s]              = true,
        [clsid.wpn_svd_s]               = true,
```

```
    [clsid.wpn_svu_s]               = true,
    [clsid.wpn_rg6_s]               = true,
    [clsid.wpn_rpg7_s]              = true,
    [clsid.wpn_val_s]               = true,
    [clsid.wpn_walther_s]           = true,
    [clsid.wpn_usp45_s]             = true,
    [clsid.wpn_groza_s]             = true,
    [clsid.wpn_knife_s]             = true,
    [clsid.wpn_grenade_f1_s]        = true,
    [clsid.wpn_grenade_rgd5_s]      = true,
    [clsid.wpn_grenade_launcher]    = true,
    [clsid.wpn_grenade_fake]        = true}
```

Тепер створимо таку саму таблицю, тільки замість класів зброї вставимо вже класи броні – і діло в шляпі! Тільки варто зазначити, що на різних рушіях класи броні можуть відрізнятися. В оригіналі таблиця з умовною назвою **armor_classes** виглядатиме ось так:

```
armor_classes = {
    [clsid.equ_stalker_s]           = true,
    [clsid.equ_helmet_s]            = true}
```

У Ганслінгері же зовсім інші класи броні, тому ця ж таблиця матиме вже інший вигляд:

```
armor_classes = {
    [clsid.equ_stalker_s]       = true,
    [clsid.equ_stalker]         = true,
    [clsid.equ_scientific]      = true,
    [clsid.equ_military]        = true,
    [clsid.equ_exo]             = true}
```

Просто майте це на увазі. Бо я коли робив спочатку для гансу, а потім для ванільки, теж зніяковів, думав «дідько, щось не те..», а виявляється – все те! Ну і про всяк випадок скажу, що ви можете цю таблицю називати як завгодно, хоч **armor_classes**, хоч **i_love_my_lord_infernis** – без різниці.

Що ж, таблицю ми створили. Тепер потрібно прописати її в переліку, знайти його можна по назвам предметів у цьому ж файлі трішечки вище від функції **isWeapon**, яку ми знайшли на самому початку:

```
local armor_classes = {}
```

Прописали. Тепер створюємо саму функцію прямо біля **isWeapon**, я назову її **isArmor**, щоби не плутатись зайвий раз:

```
function isArmor(object, class_id)
    local id = class_id or get_clsid(object)
    return armor_classes[id] == true
end
```

У результаті повністю вся лінія перевірок має виглядати приблизно якось так:

```
--------------------------------------------------------------------------------
local monster_classes = {}
local stalker_classes = {}
local weapon_classes = {}
local armor_classes = {}
local artefact_classes = {}

function IsMonster (object, class_id)
        local id = class_id or get_clsid(object)
        return monster_classes[id] == true
end
function IsStalker (object, class_id)
        local id = class_id or get_clsid(object)
        return stalker_classes[id] == true
end

function isWeapon(object, class_id)
    local id = class_id or get_clsid(object)
    return weapon_classes[id] == true
end

function isArmor(object, class_id)
    local id = class_id or get_clsid(object)
    return armor_classes[id] == true
end

function isArtefact(object, class_id)
        local id = class_id or get_clsid(object)
    return artefact_classes[id] == true
end
```

Все, перевірку зареєстровано! Тепер залишається лише повернутися до файлу **death_manager.script** та прописати нову перевірку біля перевірки на зброю, замінивши всі функції для зброї – на нові функції. Значить, знаходимо ось цю функцію:

```
    if isWeapon(item) and not(get_clsid(item)==clsid.wpn_grenade_rgd5_s or
get_clsid(item)==clsid.wpn_grenade_f1_s) then
        set_weapon_drop_condition(item)
        return
    end
```

Робимо її дублікат, видаляємо старі перевірки та прописуємо нову:

```
    if isArmor(item) then
        set_weapon_drop_condition(item)
        return
    end
```

В принципі, вже на цьому моменті все працюватиме, але зараз для поламки броні ми використовуємо функцію **set_weapon_drop_condition**, яка знаходиться в цьому ж файлі:

```
function set_weapon_drop_condition(item)
    local condition  = (math.random(40)+40)/100
    --printf("condition [%s]", tostring(condition))
    item:set_condition(condition)
end
```

Ця функція визначає для кожного предмету, для якого вона використовується, ступіть поломки за формулою:

$$\left.\begin{array}{c} \mathrm{random}_1 \in [0{,}40] \\ \mathrm{condition}_1 = \dfrac{\mathrm{random}_1 + 40}{100} = (\mathrm{random}_1 + 40)\% \end{array}\right\} \Rightarrow \mathrm{condition}_1 \in [40\%, 80\%]$$

Звідси ми дізнаємося, що мінімальна ступінь поломки зброї без взаємодії зовнішніх подразників (актору, гранати, вибухів каністри з бензином, ножа, кулі тощо) – 40%, максимальна – 80%. Але це чисто в теорії, бо на практиці я подекуди зустрічав зброю, яка була значно краща за 80%, але хрін з ним. Нас зараз цікавить те, що ми можемо створити по суті власну функцію з власною формулою, яка відрізнятиметься від цієї, і наприклад, зробити мінімальну ступінь поломки гіршою від 40%, залишивши при цьому 80% максимумом. Виглядатиме ця формула приблизно наступним чином:

$$\left.\begin{array}{c}\text{random}_2 \in [0{,}40] \\ \text{condition}_2 = \dfrac{\text{random}_1 + 40}{100} = (\text{random}_1 + 40)\%\end{array}\right\} \Rightarrow \text{condition}_1 \in [40\%, 80\%]$$

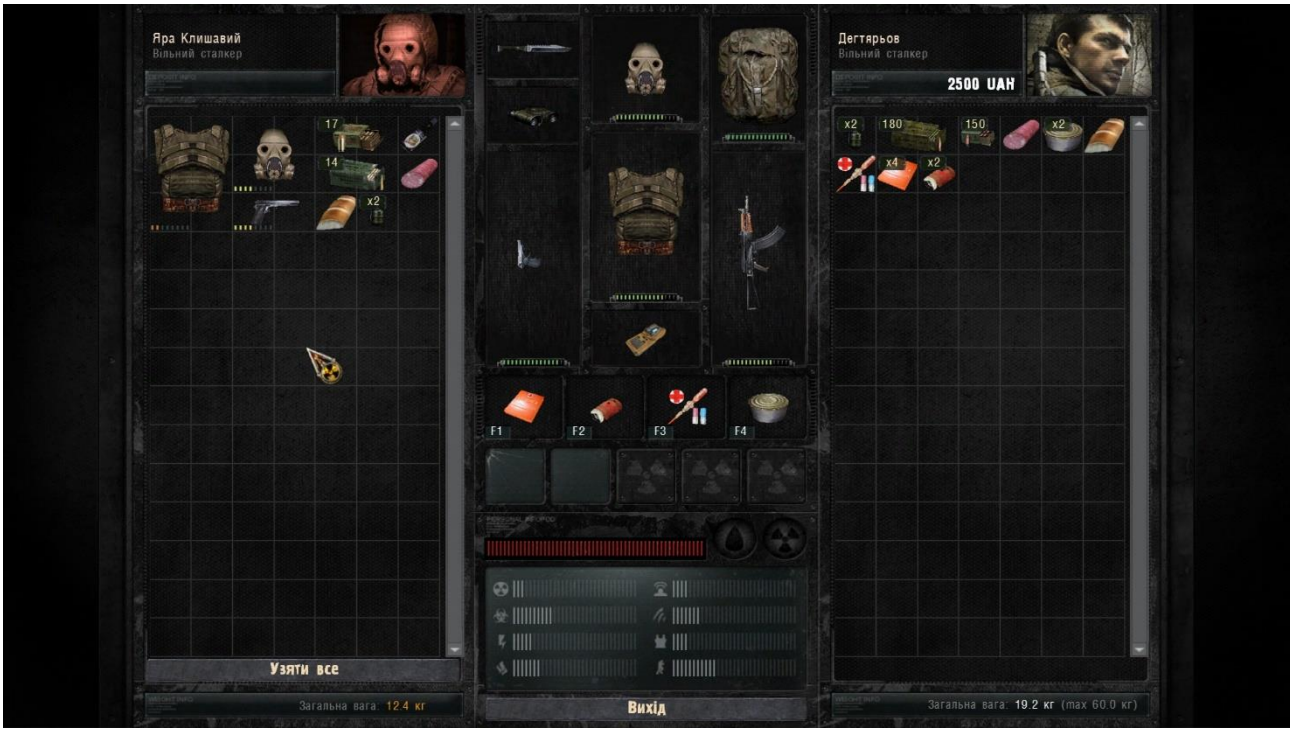
Для тих, хто не зрозумів: залишивши найкращим варіантом рандому 80, ми віддали 20 з базового доданку в рандом, і тепер у випадку, коли рандом дасть нам 0 з 60-ти, ступінь поломки броні складатиме 20%, а не 40, як було раніше. Тобто, тепер найгірша ступінь поломки без подразників складає 0.2, а не 0.4. При цьому, найкращою залишається 0.8. Тепер реалізуємо цю формулу в новій функції **set_outfit_drop_condition**, і допишемо її поряд із функцією **set_weapon_drop_condition**:

```
function set_weapon_drop_condition(item)
    local condition  = (math.random(40)+40)/100
    --printf("condition [%s]", tostring(condition))
    item:set_condition(condition)
end

function set_outfit_drop_condition(item)
    local condition  = (math.random(60)+20)/100
    --printf("condition [%s]", tostring(condition))
    item:set_condition(condition)
end
```

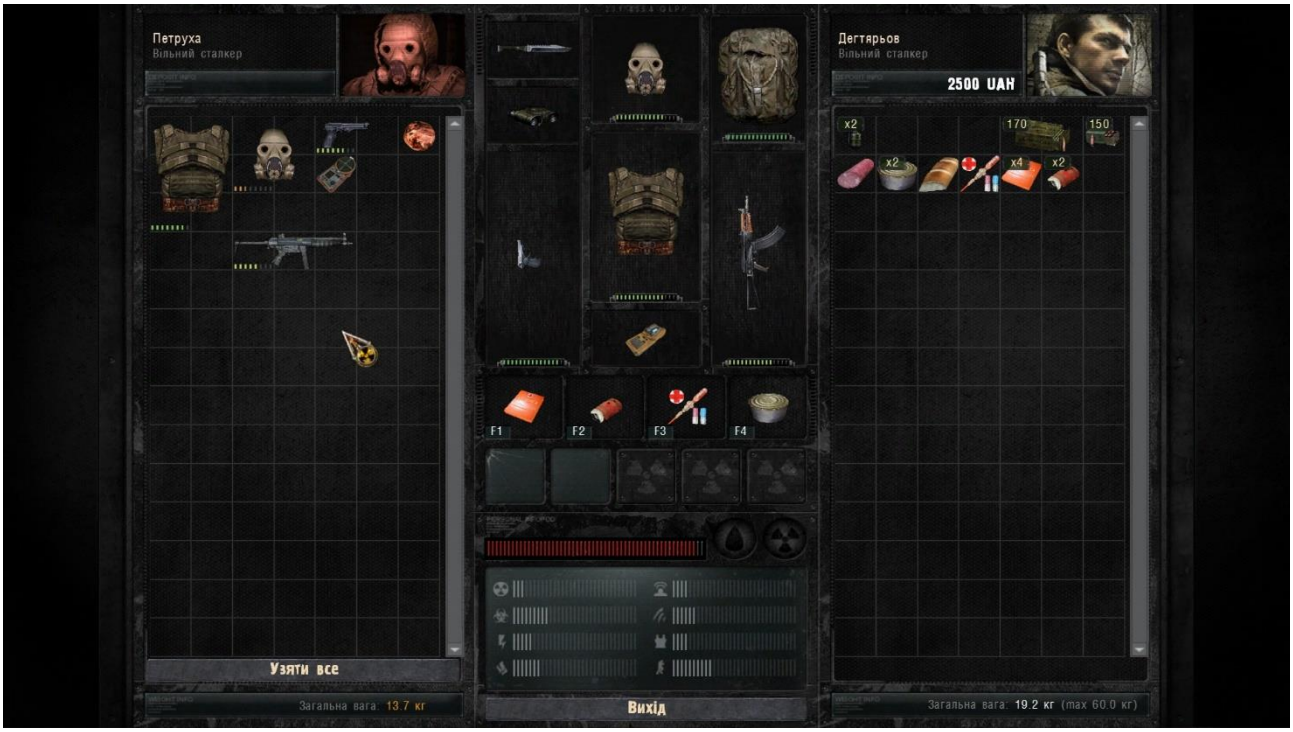
Тепер залишається переробити нашу перевірку на броню, яку ми правили нижче, вставивши **outfit_drop_condition** замість **set_weapon_drop_condition**:

```
    if isWeapon(item) and not(get_clsid(item)==clsid.wpn_grenade_rgd5_s or
get_clsid(item)==clsid.wpn_grenade_f1_s) then
        set_weapon_drop_condition(item)
        return
    end

    if isArmor(item) then
        set_outfit_drop_condition(item)
        return
    end
```

Ну що ж, панове, настав момент істини! Зберігаємо всі файли та запускаємо гру!



Якщо Ви все зробили правильно, то тепер з імовірністю 50% у вбитих досвідчених сталкерів з’являється комбінезон «Зоря», а в тих хто в протигазі – додатково може з’явитися й протигаз з імовірністю 75%. Кожен із них може бути зламаний на 20-80%. А ось що буде, якщо прописати це саме вже квестовому персонажеві в умовному **character_desc_zaton.xml**:



Все так само працює, як треба)

Можливо, комусь не сподобається те, що по суті однакові строчки треба прописувати в УСІХ випадкових персонажів, яких при використанні збройових паків (того ж OWR Infernal Edition) може перевалити за декілька сотен, але як на мене це навпаки плюс, оскільки можна додатково налаштувати, в кого ти б хотів бачити цю броню, а в кого – ні. До того ж, якщо ти лінуєшся працювати над своїм же модом – не дивуйся, що потім він вилітатиме через те, що ти вирішив зайвий раз не паритись. І не забувайте, що цей спосіб працює на всіх НПС – у тому ж числі й квестових, та нових! Те ж саме стосується й нової броні, якщо захотів додати шкіряну куртку – можеш сміло прописувати цим методом її для сталкерів-новачків – вона заспавниться та зіпсується. Тому, користуйтесь цим методом! Я старався)

p.s. Мабуть, варто було сказати це на самому початку, але можна все набагато спростити якщо в падлу копирсатися зі скріптами та взагалі уникнути правок файлу **_g.script**, поправивши початкову перевірку, додавши до неї недостаючі класи броні.

```
    if isWeapon(item) and not(get_clsid(item)==clsid.wpn_grenade_rgd5_s or
get_clsid(item)==clsid.wpn_grenade_f1_s) or (get_clsid(item)==clsid.equ_stalker_s
or get_clsid(item)==clsid.equ_helmet_s) then
        set_weapon_drop_condition(item)
        return
    end
```

Тоді в нас так само псуватиметься броня, просто з формулою, яка вказана в функції **set_weapon_drop_condition**, але як на мене цей спосіб максимально костильний та нечемний для синтаксису програмування, бо чисто технічно ми використовуємо для броні функцію, яка призначалася для зброї. Але якщо комусь так буде легше зробити те, що він хотів – я буду тільки радий від того, що у Вас це вийшло)

До того ж, конкретно такий код я не тестував, тому за те, що гра не вилетить через 5 хвилин гри, не ручаюсь. Розумієте, для мене раніше важливо було зробити хоч якось, аби працювало. Але з часом я зрозумів, що ти це робиш не для інших – ти це робиш в першу чергу для себе. От ти би сам хотів, щоби в твоєму моді все було побудовано на костилях і багах? Та для себе ти би відшліфував усе до дірок, і це правильно. Тому краще вчитися робити одразу якісно, аніж потім переучуватися.

Сподіваюся, ця стаття змогла комусь допомогти! Якщо ще комусь буде треба – **прикріплюю відос**, де я все пояснюю на пальцях та показую, куди що треба вставити та прописати. Дякую за прочитання! Слава нації!


character_desc_zaton.xml


character_desc_general.xml


death_manager.script


_g.script