ÚVOD DO FALLOUT 2 EDITORU

Dostává se vám do rukou Fallout 2 editor, skripty a kompilátor skriptů. Našli jsme si něco času a tak jsme dali dohromady trochu dokumentace a instalátor. Přesto bude jeho použití vyžadovat mnoho zkoušení metodou pokus-omyl.

Věci které by jste si měli uvědomit:

Tento editor není svatým grálem. Původně se vůbec neuvažovalo o tom, že by byl uvolněn pro volné použití. Proto se může stát že editor spustíte a nebude splňovat vaše očekávání. Možná utrpíte poškození sítnice. Možná vás bude všude svědit. Možná dokonce dostanete zácpu. Proto si nestěžujte, že jsme vás neupozornili – hodláte pracovat s editorem, který byl původně určen pouze pro profesionální herní vývojáře.

Editor používáte na vlastní nebezpečí. Před nějakým časem proběhl mezi fanoušky průzkum (http://feedback.blackisle.com/forums/showthread.php?s=&threadid=51095), jestli by radši editor dříve, ale bez dokumentace, a většina souhlasila, takže zde ho máte. Nakonec jsme přibalili i nějakou tu dokumentaci, která by vám měla alespoň mírně ulehčit strasti pronikání do editoru.

Aby editor fungoval, musíte mít nainstalovaný Fallout 2 (pokud možno plnou instalaci, ale není to podmínkou).

Nezapomeňte si pročíst soubor readme.txt v adresáři Scripts! Vysvětluje jak kompilovat skripty a jak používat různé kompilátory C pro jejich přípravu preprocesorem. Dále jsou zde informace a odkaz na Open Watcom, což je volně dostupný kompilátor Céčka.

Toto není zdrojový kód Falloutu 2 a ani neplánujeme v brzké době tento kód uvolnit.

Pozor – některé problémy s editorem jsou uvedeny v sekci „Známé Problémy“. Před použitím editoru by jste se s nimi měli seznámit.

Fallout 2 editor není společností Interplay podporován a proto pokud s ním máte nějaký problém, neobracejte se na technickou podporu, jelikož by nevěděli, o čem to vlastně mluvíte.

Pokud máte nějaké dotazy ohledně editoru, pište je ve Fallout 1 a 2 fóru (www.interplay.com), a my se je pokusíme zodpovědět. Nezapomeňte, že je to už docela dávno, co jsme editor naposledy používali, a proto je dost našich znalostí editoru v troskách nebo docela zapomenuto. No ale uvidíme co se dá dělat.

Je tu mnoho lidí, kterým by jste měli děkovat za vypuštění editoru.

Joshi Sawyerovi. Kdyby nebylo Joshe, nebylo by ani editoru. To on při jednom z našich setkání nadnesl nápad zveřejnit editor, a my jsme všichni souhlasně přikyvovali, ale jen do té doby, než jsme zjistili, jakou třísku do zadnice jsme si to zadřeli.

Darrenu Monahanovi a Feargusi Urquhartovi. Kdyby nebylo Darrena a Ferga, nebylo by ani editoru.

Chrisi Heidarimu. Chris Heidari strávil docela dost času při testování editoru a jako bonus připravil instalátor.

Scotty Evertsovi, který odvedl lví podíl při přípravě dokumentace.

...a nejvíc ze všech Chrisi Jonesovi. Chris dával ve svém volném čase editor dohromady. Kdyby nebylo Chrise, nebylo by ani editoru.

Díky a užijte si to

Chris Avellone

Překlad Jay Kowalski, Vault Šílené Brahmíny (madbrahmin.bonusweb.cz)

POUŽITÍ EDITORU MAP

Úvod

Fallout editor byl napsán před více než šesti lety. I když byl v průběhu vývoje Falloutu 1 a 2 vylepšován a upravován, je na dnešní standardy zastaralý, a proto budete při tvorbě vlastních map potřebovat notnou dávku trpělivosti (prvních pár map bude porod, ale později, až si to procvičíte, to zas tak těžké nebude). Hlavním problémem při vytváření map bude nalezení všech potřebných součástí, které na mapě chcete.

Vše je rozděleno do šesti skupin, bohužel zde nejsou žádné podskupiny, které by je rozdělovaly na menší části – engine je prostě nepodporuje. Když byly postupně vytvářeny další objekty, jednoduše byly přidány na konec skupiny. Takže je docela možné, že najdete části jedné budovy roztroušené po celé sekci. Naštěstí jsou zde záložky, které to aspoň trochu usnadňují (bude vysvětleno později).

Tato verze obsahuje všechnu grafiku a objekty z F2 a znovupoužitou grafiku a objekty z F1. Grafiku z F1, pro kterou nebylo žádné využití, jsme smazali abychom ušetřili místo, ale když si s tím trochu pohrajete, klidně ji tam zase dostanete.

Základní koncepty

Každá mapa je tvořena šesti základními typy objektů.

- **Dlaždice (Tiles)** – Ty se dělí na podlahové a střešní dlaždice – základ každé mapy. Pokud je volba „Roof“ (střecha) vypnuta, pak se dlaždice pokládají jako podlaha. Pokud volbu „Roof“ zapnete, budou se dlaždice pokládat jako střešní. To znamená, že všechny dlaždice mohou být použity i jako podlaha i jako střecha.

- **Zdi (Walls)** – Části zdí budov. Existují ve dvou směrech – severně-jižní (N/S) a východně-západní(E/W). Všechny budovy, ať už vnější nebo vnitřní, jsou tvořeny právě zdmi.

- **Dekorace (Scenery)** – Doplňkové věci jako jsou židle, stoly, sračky, vraky atd., které dokreslují prostředí. Dveře jsou také mezi dekoracemi.

- **Předměty (Items)** – Předměty jako jsou zbraně, stimpaky atd., které se umisťují na příšery, kontejnery nebo rovnou na zem. Kontejnery (containers), např. bedny, skříňky atp., jsou též v této kategorii.

- **Příšery (Critters)** – Souhrnné označení všech lidí a monster. Aby reagovaly na podněty z prostředí, musíte k nim připojit správný skript.

- **Ostatní (Misc)** – Všechny zbylé věci jako grafika střel a podobně. Krom jiného sem též patří části odchodové mřížky (exit grid).

Objekt, který chcete umisťovat vyberete pravým kliknutím na jeho znázornění v seznamu objektů (viz. obrázek níže), poté ho levým kliknutím umístíte na mapu.

Abyste vybrali umístěný objekt, klikněte na něj levým tlačítkem. Nijak se sice nezvýrazní, ale že je vybraný poznáte tak, že se jeho název objeví vpravo dole v informačním okně. Občas budete muset kliknout vícekrát, než se objekt vybere. Objekt také můžete přesouvat a to tak, že na něm stisknete levé tlačítko myši a přesunete jej do požadovaného umístění.

Menu – Menu je ukryto, dokud nepřesunete kurzor myši k hornímu okraji obrazovky.

Záložky (bookmarks) – V každé kategorii si můžete nastavit až 9 záložek, ke kterým se budete později vracet pomocí kláves 1-9. Abyste si záložku nastavili, vyberte si pozici v seznamu objektů. Poté v menu vyberte příkaz „Tools/Set Bookmark“ (nástroje/nastav záložku). Následně stiskněte číslici, na kterou si chcete záložku nastavit.

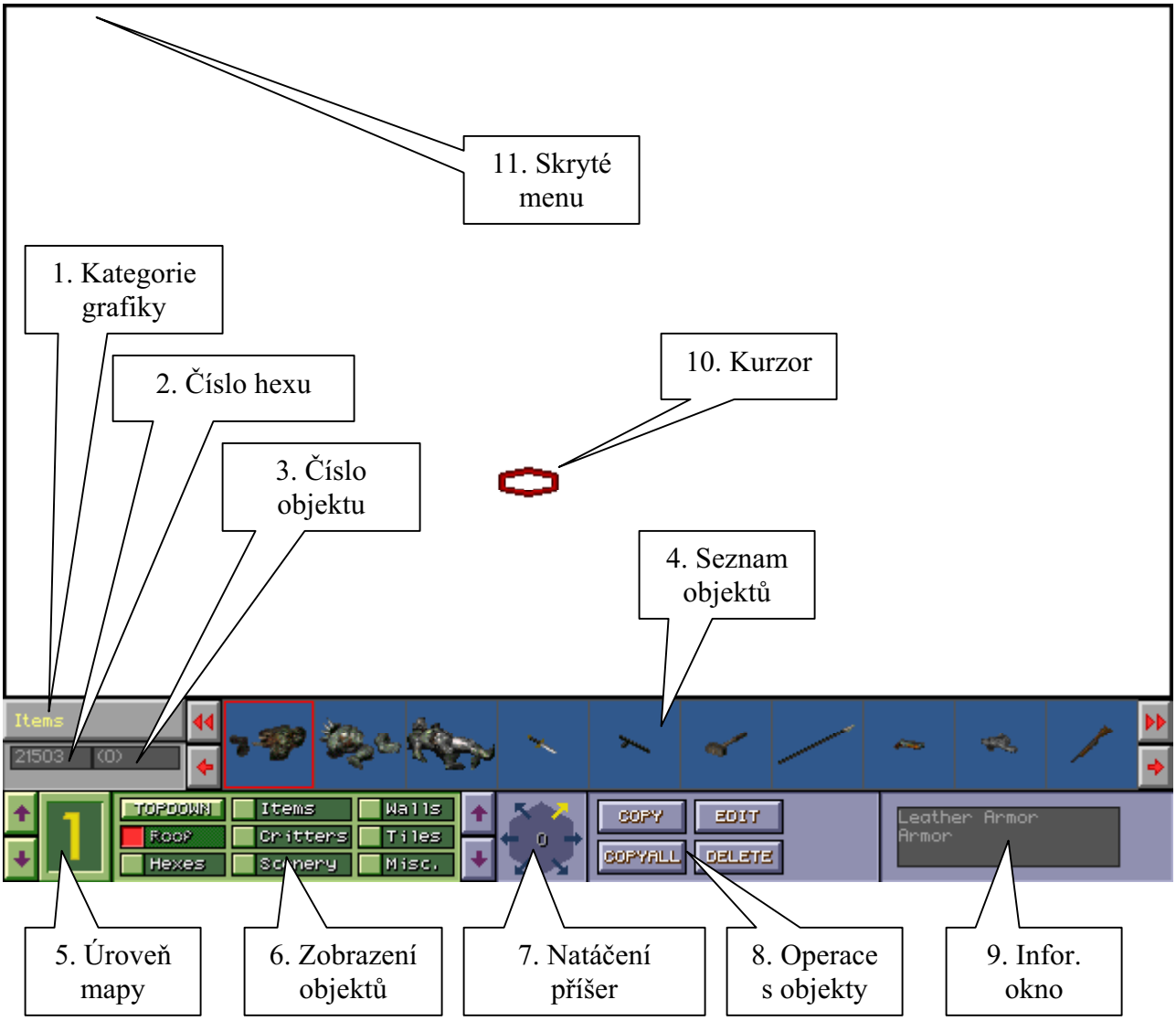
Každá mapa má tři úrovně. Pro přepínání mezi jednotlivými úrovněmi slouží dvě šipky v levém dolním rohu obrazovky. Tyto úrovně umožňují zkrácení načítací doby mezi přechodem mezi mapami.

Klávesa “**H**“ vypíná a zapíná zobrazení hexové sítě. Hex vyplněný červenou barvou je *blokovaný* a ani hráč (PC) ani nehráčské postavy (NPC) na něj nemohou vstoupit. Většina zdí a dekorací má nastaveno blokování. Občas, při umisťování zdí nebo dekorací, vznikají mezi jednotlivými částmi díry, které jsou průchozí a průstřelné. Proto jsou zde i speciální blokující hexy (vysvětleno dále).

Klávesa “**F8**“ přepne editor do „herního“ módu. Můžete se po mapě pohybovat se svou postavou a zkoušet, zda nejsou ve zdech nějaká prázdná místa nebo jestli funguje blokování posunu atd. Tohle je to nejvíc nejlepší tlačítko v celém vesmíru.

Klávesou “**F12**“ si uložíte screenshot. Nemusíte být v herním módu – screenshoty můžete pořizovat kdekoli se v editoru nacházíte.

Základní funkce editoru



1. **Kategorie grafiky** – (art categories) Místo výběru pomocí této nabídky můžete použít klávesy F1-F6.

2. **Číslo hexu –** (hex number) Číslo hexu na kterém umísťujete objekt. Užitečné pokud chcete říci skriptérovi které ze 34 a víc šlapek na mapě máte připojit speciální skript, stejně tak jako při vytyčování hranic pro působení některých skriptů.

3. **Číslo objektu** – (art object number) Číslo objektu, který je na prvním místě v seznamu objektů.

4. **Seznam objektů –** (art object bar) Zde se zobrazují objekty vybrané kategorie. Dvojité šipky posouvají výběr o stránku, jednoduché o jednu pozici. Pravým tlačítkem vybíráte objekt, levým ho umisťujete na mapu. Klávesy "+" a "- " posouvají seznamem, pokud držíte shift, posouvá se o stránku.

5. **Úroveň mapy** – (map level, elevation) V jednom souboru mohou být až tři úrovně map. Úspora času při nahrávání map typu jeskyní nebo vnitřků budov.

6. **Zobrazení objektů** – (art object toggle) Vypíná nebo zapíná zobrazování objektů různých typů. „Roofs“ je standardně vypnuto, pokud chcete umisťovat střešní dlaždice, musíte si „roofs“ zapnout. Nezapomeňte to potom zase vypnout, abyste mohli znova pokládat podlahové dlaždice. K vypínání / zapínání „roofs“ můžete použít klávesu “R“. Tlačítko „Top Down“ nefunguje – je to funkce, která nebyla nikdy implementována.

7. **Natočení příšer –** (critter object rotation) Slouží k určení natočení příšery při jejím umístění na mapu. Když na vás všichni supermutanti, které umístíte na mapu, vystrkují zadek, měli byste zkontrolovat natočení příšer a změnit ho na to, které potřebujete.

8. **Operace s objekty –** (Object Functions)

   - “**Copy**” zkopíruje vybrané prvky aktuální kategorie. To znamená, že pokud máte zrovna vybránu kategorii „dlaždice“, zkopírují se jen dlaždice atd. Táhněte myší pro výběr prvků které chcete zkopírovat. Zkopírovanou skupinu můžete umístit kdekoli na mapě. Výběr zrušíte pravým tlačítkem.

   - “**Copy All**” podobně jako Copy, ale s tím rozdílem že bude zkopírováno vše vybrané, dlaždice, příšery, dekorace, prostě vše. Bezva zábava na mejdanech.

   - “**Edit**” je pro změny vlastností vybraného objektu.

   - “**Delete**” natrvalo odstraní vybraný objekt.

9. **Informační okno –** (info screen) Zobrazuje názvy objektů a ostatní doplňující informace.

10. **Kurzor** – Mění se podle módu editoru nebo prováděné operace.

11. **Skryté menu –** Zobrazí se když přesunete kurzor do horní části obrazovky.

Menu

File (soubor) -

Standardní příkazy pro ukládání, načítání a ukončování. Však to znáte.

Tools (nástroje) -

- “**Create Pattern**” / “**Use Pattern**” – (Vytvořit vzorek / Použít vzorek) Umožňuje vám vybrat si jeden z předdefinovaných stylů dlaždice a potom je umisťovat. Vytváření nových stylů nefunguje. „Use Pattern“ obsahuje seznam předdefinovaných stylů, které zjednodušují pokládání střech a podlah. Vyberte jeden ze stylů, levým kliknutím ho nyní můžete položit na mapu. Klávesy “+“ a “- “ mění velikost pokládané plochy. Pravým kliknutím ukončíte režim pokládání vzorků.

- “**Move Map**” – (Přesunout mapu) Umožňuje posunutí celé mapy. V případě že při vytváření mapy dojdete až na okraj, můžete celou mapu o kus posunout. Mapa se přesouvá pomocí šipek, ale měli jsme s tím trošku problémy.

- “**Move Map Elevation**” – Přesune celou mapu do jiné úrovně.

- "**Copy Map Elevation**" – Zkopíruje celou mapu do jiné úrovně. Pokud vytváříte dvě velmi podobné mapy, tak vám toto ukrátí čas.

> **Poznámka:** Vypadá to, že s funkcí „Copy Map Elevation" je trošku problém. Když se pokoušíte zkopírovat mapu do jiné úrovně, editor spadne a zahlásí, že „provedl neplatnou operaci". Sakra, může vám to fungovat, ale nám se to rozchodit nepodařilo.

- "**Toggle Block Object View**" – (Zobrazení blokovacích objektů) Máme zde několik neviditelných blokovacích objektů, které jsou zde proto, aby zabraňovaly pohybu nebo posouvání mapy. Tento příkaz tyto objekty zviditelní, takže s nimi můžete všemožně manipulovat – ve hře nejsou viditelné nikdy, pokud ovšem nemáte nějakou mírně jetou verzi Falloutu. Mnohdy se vám stane, že vám mezi umístěnými zdmi a dekoracemi zbudou prázdná místa. Od toho jsou zde blokovací objekty, které tyto díry vyplňují, takže jimi ani hráč ani nehráčské postavy nemůžou projít (nebo střílet). Pokud se chcete podívat na nějaký příklad, prohlédněte si jakoukoli z původních F2 map.

  - **Blokovací zeď –** (Wall Blocker) Tmavě zelené "W". Zabraňuje pohybu, na automapě vypadá jako zeď (světle zeleně). Je zde ve dvou verzích, ale zdá se, že používána byla jen ta s označením „Stěna s.t." („Wall S.T."). Kategorie zdi, #621.

  - **Blokovací dekorace –** (Secret Blocking Hex) Světle zelené "S". Brání pohybu, na automapě se zobrazuje jako dekorace (tmavě zeleně). Kategorie dekorace, #66.

  - **Neviditelné blokování –** (Block Hex Auto Inviso) Žluté "SAI". Zabraňuje pohybu, nezobrazuje se na automapě. Používá se k zablokování částí mapy, které nechcete ukázat na automapě. Kategorie dekorace, #343.

  - **Zdroj světla –** (Light Source) Zelený "symbol slunce". Nebrání pohybu, vytváří světelný zdroj. Světlo může být upraveno pomocí tlačítka „Edit". Kategorie dekorace, #140.

  - **Značka odchodové mřížky –** (Exit Grid Map Marker) Modré "EG". Speciální značka, na automapě zobrazená v okrové barvě, kterou je označeno, kde se nachází východ z mapy. Kategorie dekorace, #48.

  - **Blokování posunu –** (Scroll Blocker) Bílé "S". Speciální značka, která brání v posunu obrazovky. Když střed obrazovky přejede přes tento hex, posun se zastaví a nejde se posunout dál. Používá se k definici okrajů mapy. Uvědomte si, že blokování posunu funguje oboustranně. To znamená, že když se hráč ocitne nějakou náhodou mimo okraje, nebude schopen přesunout pohled zpět. Kategorie ostatní, #11.

  Několik těchto blokovacích objektů bylo navrženo aby spolupracovalo s herní automapou. Automapa zobrazuje objekty z kategorie zdí v světle zelené barvě, dekorace v tmavě zelené. Proto, aby byla automapa použitelná, byly použity tyto dva odstíny. Takže když umisťujete blokovací hexy, pro vyplňování děr ve zdích používejte blokovací zeď a pro vyplňování děr v dekoracích používejte blokovací dekoraci. Neviditelné blokování se používá k zablokování větších oblastí, jako jsou jezera, radioaktivní kaluže atd., a které nechcete ukazovat na automapě.

- **Nastavení odchodové mřížky –** (Exit Grid) Odchodové mřížky jsou hnědě nebo zeleně vyšrafované oblasti, které umožňují přechod hráče mezi mapami. Grafika těchto vyšrafovaných oblastí je v kategorii ostatní. Zelené oblasti jsou určeny pro přechody mezi mapami nebo úrovněmi map, hnědé pak pro přístup do mapy světa (je zde ještě jedna černě zbarvená mřížka, ale nebyla k ničemu využita). Všimněte si, když pokládáte odchodovou mřížku, že každá má jeden z hexů v modré barvě (při zapnutém zobrazování hexů). Tyto modré hexy jsou důležité – aby odchodová mřížka fungovala, musí přes tento hex přejít hráč. Takže když pokládat odchodovou mřížku, zkontrolujte, zda není někde přerušená a hráč vždy musí přejít přes jeden z modrých hexů, aby se dostal k východu

    - "**Set Exit Grid Data**" – (nastav data odchodové mřížky) Nastaví, jak se bude chovat odchodová mřížka. Poté, co zde vše nastavíte, dalšími dvěma položkami menu označíte potřebné mřížky. Pokud možnost „Exit Grid Dest Map" (cílová mapa) nastavíte na –1, pak tato mřížka odešle hráče do mapy světa.

    - "**Mark Exit Grids**" – (označ odchodovou mřížku) Tento příkaz přepne editor do režimu, ve kterém můžete označit jednotlivé části odchodové mřížky tím, co jste si nastavili v předchozí nabídce. Klávesou "ESC" ukončíte tento režim. To, že jste v režimu označování odchodových mřížek není v editoru nijak indikováno. Ostatní funkce nemusí pracovat, dokud jste v tomto režimu.

    - "**Mark All Exit Grids**" – (označ všechny odchodové mřížky) Tento příkaz označí VŠECHNY odchodové mřížky, které jsou na mapě, tím, co jste nastavili příkazem „Set Exit Grid Data". Toto přepíše vše, co jste dosud označili.

- "**Clear Map Level**" – Vymaže všechny objekty na mapě.

Zbývající příkazy se týkají skriptů a nejsou v tomto dokumentu popsány.

Známé problémy s editorem

Zde jsou problémy, kterými editor trpí:

- Editor spadne, pokud se pokusíte zobrazit skript mapy poté, co jste zrušili předchozí výběr skriptu mapy. Vyzkoušejte si to sami: v menu Scripts -> Set Map Script, dvakrát stiskněte ESC pro zrušení nabídky. Nyní zkuste zobrazit skript mapy příkazem Scripts -> Show Map Script. V tomto okamžiku editor spadne.

- Ojediněle se vám může stát, když načítáte mapy, že načtete mapu, ale obrazovka zůstane úplně černá. To by mělo zmizet, pokud posunete obraz nebo pohnete kurzorem.

- Občas, když alt-tabnete ven z editoru, může editor zmizet z dolní lišty vedle nabídky Start. Stále se však do editoru můžete alt-tabem vrátit. Editor se prostě s alt-tabem nemá rád.

- Další problém je s funkcí „Copy Map Elevation". Když se pokoušíte zkopírovat mapu do jiné úrovně, editor spadne s obvyklou hláškou „Program provedl neplatnou operaci.".

Skriptovací příkazy editoru

Následuje seznam možných příkazů použitelných ve skriptech, který může být užitečný pro programátory a skriptéry z Falloutovské komunity, kteří se snaží rozluštit, co co znamená. Nevím, co znamenají šedě označené příkazy, ale může to být významné. Možná jsou tyto příkazy špatné nebo nepracují (některé určitě fungují – pozn. překl.). Bádejte a přijďte na to, co to má znamenat, nebo počkejte až to udělá Red! nebo jeden z těch ruských šílenců z Team X.

action_being_used *int Script*	Vrací aktuální dovednost, která byla použita na objekt ke kterému je připojen tento skript.
add_obj_to_inven *void Inven* who (ObjectPtr) item (ObjectPtr)	Přidá objekt (*item*) do inventáře jiného objektu (*who*). Toto pracuje pouze s objekty typu předmět (item).
add_mult_objs_to_inven *void Inven* who (ObjectPtr) item (ObjectPtr) count (int)	Přidá několik (*count*) instancí objektu (*item*) do inventáře jiného objektu (who). Toto pracuje pouze s objekty typu předmět (item).
add_timer_event *void Meta(Time)* obj (ObjectPtr) time (int) info (int)	Přidá do fronty načasovanou událost (timed event) s daným časovým zpožděním (*time*), který zavolá skript objektu (*obj*). *Info* je použito k odlišení jednotlivých událostí. Skript si *info* načte pomocí funkce *fixed_param*. Čas je udáván ve kmitech (ticks - můžete použít funkci *game_ticks(seconds_num)*, abyste získali čas ve vteřinách).
anim *void Anim* who (ObjectPtr) anim (int) direction (int)	Nastaví animaci (*anim*) objektu (*who*), která proběhne v daném směru (*direction*).
anim_action_frame *int Anim* who (ObjectPtr) frame (int)	Navrací akční snímek (action frame) z daného snímku (*frame*) daného objektu (*who*) Toto může být použito jako zpoždění v sekvenci registrace animací.
anim_busy *int (boolean) Anim* who (ObjectPtr)	Navrací True, jestliže objekt (*who*) právě provádí animaci, jinak navrací False. Toto může být použito k určení, zda už daný objekt dokončil animaci.
animate_move_obj_to_tile *void Anim* who (ObjectPtr) tile (int) speed (int)	Nastaví animaci příšery (*who*), kterou dojde na danou dlaždici (*hex*) danou rychlostí (*speed*). Rychlost (chůze / běh) může mít navíc nastavený příznak pro přerušení aktuální animace (viz define.h) objektu (*who*).
animate_rotation *void Anim* direction (0-5)	Změní orientaci objektu *self_obj* (objekt ke kterému je připojen prováděný skript) na daný směr (*direction*).
animate_run_to_tile *void Anim* tile (int)	Nastaví animaci objektu *self_obj* pro doběhnutí na danou dlaždici (*hex*).

animate_set_frame *void Anim* newFrame (int)	Změní aktuální snímek animace objektu *self_obj* na daný snímek (*newFrame*). Toto může být použito např. pro změnu lampy z normální na rozbitou nebo blikající atd. Dále může nahradit funkci *animate_stand* pro 2-snímkové animace.
animate_stand *void Anim*	Nastaví animaci aktuálního objektu (*self_obj*) na animaci stání. Toto může být použito k otevření dveří, otevření kontejnerových objektů, nebo ošívání se přišer.
animate_stand_obj *void Anim* obj (ObjectPtr)	Nastaví animaci objektu (*obj*) na animaci stání. Toto může být použito k otevření dveří, otevření kontejnerových objektů, nebo ošívání se přišer.
animate_stand_reverse *void Anim*	Nastaví animaci aktuálního objektu (*self_obj*) na obrácenou animaci stání. Toto se používá pouze pro kontejnery nebo dveře a slouží k jejich zavření.
animate_stand_reverse_obj *void Anim* obj (ObjectPtr)	Nastaví animaci objektu (*obj*) na obrácenou animaci stání. Toto se používá pouze pro kontejnery nebo dveře a slouží k jejich zavření.
art_anim *void Anim* fid (int)	Navrací animaci, kterou reprezentuje daný *fid* (ANIM_stand, ANIM_pickup, atd.).
attack *void Combat* who (ObjectPtr)	Způsobí, že se aktuální objekt (*self_obj*) pokusí zaútočit na objekt (*who*). Toto je makro zkracující použití funkce *attack_complex*.
attack_complex *void Combat* who (ObjectPtr) called_shot (int) num_attacks (int) bonus (int) min_damage (int) max_damage (int) attacker_results (int) target_results (int)	Způsobí, že se aktuální objekt (*self_obj*) – musí to být příšera – pokusí zaútočit na objekt (*who*). Útok samotný je ovlivněn několika parametry: *called_shot* – cílené střely – 0/1/specifické znamená ne/náhodně/specificky (hlava, trup atd.). *num_attacks* – počet extra útoků, které získává *self_obj* před cílem. *bonus* – bonus k pravděpodobnosti zásahu v prvním kole. *min_damage* – minimální poškození, které způsobí první útok. *max_damage* – maximální poškozená, které způsobí první útok. *attacker_results* – v jakém stavu bude útočník po prvním útoku. *target_results* – v jakém stavu bude cíl po prvním útoku.
attack_setup *void Combat* who (ObjectPtr) victim (ObjectPtr)	Spustí útok objektu (*who*) na cíl (*victim*), bez toho, aby do toho byl zatažen tento skript. Může být použito k spuštění útoků skriptem mapy.
car_current_town *int Map*	Navrací aktuální oblast, ve které se nachází hráčovo auto. Čísla oblastí se dají nalézt v souboru maps.h.
car_give_to_party *int Map*	Přidá auto do družiny a přesune ji do mapy světa.
car_give_gas *int Map* amount (int)	Doplní do auta dané množství (*amount*) paliva.
combat_difficulty *int*	Navrací aktuální nastavení obtížnosti boje (určuje se v obrazovce nastavení).
combat_is_initialized *int*	Navrací True, pokud se systém nachází v bojovém režimu, jinak navrací False.

create_object *int (?) Object* pid (int) tile_num (int) elev (0-2)	Vytvoří objekt z daného prototypu (*pid*) a umístí ho na daný hex (*tile_num*) v dané úrovni (*elev*). Pokud prototyp uvádí, že k němu bude připojen skript, tak se tak stane.
create_object_sid *int (?) Object* pid (int) tile_num (int) elev (0-2) sid (int)	Vytvoří nový objekt z prototypu (*pid*) a umístí ho na daný hex (*tile_num*) v dané úrovni (*elev*). Pokud není *sid* (script id) –1, pak bude místo implicitního použit skript daný touto hodnotou.
critter_add_trait *int Critter* who (ObjectPtr) trait_type (int) trait (int) amount (int)	Příšeře (*who*) přidá rys (*trait*) daného typu (*trait_type*). Možné typy rysů dané systémem SPECIAL jsou omezeny na: Perky (Perks) Rysy (Traits) Informace o objektu, jako je tým, ai-balík (ai-packet) apod.
critter_attempt_placement *int Map* who (ObjectPtr) hex (int) elev (0-2)	Pokusí se umístit příšeru (*who*) na dané místo (*hex*) v dané úrovni (*elev*). Pokud neuspěje, pokusí se umístit příšeru na další hex co nejblíže původnímu umístění. Tato funkce nekontroluje, zda je cílový hex viditelný na obrazovce.
critter_damage *void Critter* who (ObjectPtr) dmg_amount (int)	Udělí příšeře (*who*) dané poškození (*dmg_amount*), které ji eventuálně i zabije.
critter_heal *void Critter* who (ObjectPtr) amount (int)	Vyléčí příšeru (*who*) o dané množství hp (*amount*) až do jejího maxima.
critter_injure *int Critter* who (ObjectPtr) how (int)	Zraní danou příšeru (*who*) tak, že jí ochromí končetiny, oslepí atd. (definováno konstantami DAM_CRIP_ARM_LEFT, DAM_BLIND, atd. v souboru define.h).
critter_inven_obj *(ObjectPtr) Critter/Inven* who (ObjectPtr) where (int)	Navrací ukazatel na objekt (předmět), který je v daném umístění (*where*) objektu (*who*). Pokud zde žádný předmět není, funkce vrací NULL. Možná umístění jsou INVEN_TYPE_WORN (brnění), INVEN_TYPE_RIGHT_HAND (pravá ruka) a INVEN_TYPE_LEFT_HAND (levá ruka).
critter_is_fleeing *int Critter* who (ObjectPtr)	Navrací True pokud je nastaven příznak FLEE (zda utíká) příšery (*who*).
critter_mod_skill *int Critter* who (ObjectPtr) skill (int) amount (int)	Upravuje danou dovednost (*skill*) příšery (*who*) o dané množstvím (*amount*). Poznámka: tato funkce funguje pouze na objektu hráčské postavy (obj_dude).

critter_rm_trait *int Critter* who (ObjectPtr) trait_type (int) trait (int) amount (int)	Odstraní určitý rys (*trait*) daného typu (*trait_type*) z příšery (*who*). Viz. funkce critter_add_trait.
critter_set_flee_state *int Critter* who (ObjectPtr) flee_on (Boolean)	Nastaví příznak FLEE příšery (*who*) na zapnuto / vypnuto (*flee_on*). Toto ovládá, jestli příšera zdrhne z boje.
critter_skill_level *int Critter* who (ObjectPtr) skillNum (int)	Navrací aktuální velikost dovednosti (*skill_num*) objektu (*who*).
critter_state *int Critter* who (ObjectPtr)	Navrací stav dané příšery (*who*), který určuje, zda je příšera mrtvá, v bezvědomí atd.
critter_stop_attacking *int Critter* who (ObjectPtr)	Nastaví objektu (*who*) příznak toho, že nechce dále pokračovat v boji.
cur_map_index *int Map*	Navrací číslo aktuální mapy. Konstanty pro jednotlivé mapy jsou definovány v souboru define.h.
cur_town *int Map*	Navrací číslo aktuálního města. Konstanty pro jednotlivá města jsou definovány v souboru define.h.
days_since_visited *int Map*	Navrací počet dní od poslední návštěvy této mapy. Pokud mapa ještě nebyla navštívena, vrací –1.
debug_msg *void Debug* text (string)	Vypíše řetězec (*text*) do ladícího monitoru (debug monitor). K výpisu ladících informací by měla být místo funkce *display_msg()* použita tato funkce.
destroy_object *int Object* obj (ObjectPtr)	Zničí objekt (*obj*), což spustí proceduru *destroy_proc* daného objektu (pouze pokud tento objekt není zároveň volajícím objektem této funkce).
destroy_mult_objs *int Object* item (ObjectPtr) count (int)	Zničí více (*count*) instancí daného objektu (*obj*). Tato funkce si zjistí, ve kterém inventáři se objekt nachází (pokud není na zemi). Pokud se objekt nachází na zemi, tak je zde samozřejmě jen jedna jeho instance, takže se zničí pouze tato jedna.
dialogue_reaction *void Dialog* mood (int)	Nastaví animaci reakce v dialogovém systému.
dialogue_system_enter *void Dialog*	Sdělí dialogovému systému, že tento objekt požaduje spuštění rozhovoru. Tato funkce se používá, když chce skript spustit rozhovor sám, aniž by čekal na hráče až rozhovor spustí. Zavolá proceduru *talk_proc* tohoto objektu.
difficulty_level *int*	Navrací aktuální nastavení obtížnosti hry (určené v obrazovce nastavení).
display_msg *void* message (string)	Zobrazí řetězec (*message*) v herním okně pro zprávy (levý dolní roh).

do_check *int (roll_result) Skill* who (ObjectPtr) check (int) modifier (int)	Provede ověřovací hod proti jedné ze základních statistik (síla, vnímání atd.).
drop_obj *void Inven* obj (ObjectPtr)	Donutí objekt *self_obj* k odstranění daného předmětu (*obj*) z jeho inventáře na zem. Toto animuje *self_obj*.
drug_influence *int Critter* who (ObjectPtr)	Navrací True, pokud je daná příšera (*who*) pod vlivem nějaké drogy. Pokud ne, navrací False.
dude_obj *(ObjectPtr)*	Navrací ukazatel na objekt hráčské postavy.
elevation *int Map* obj (ObjectPtr)	Navrací aktuální úroveň mapy, která je zobrazována.
end_dialogue *void Dialog*	Ukončí dialogový systém.
endgame_movie *void Meta*	Spustí závěrečné video.
endgame_slideshow *void Meta*	Spustí závěrečné titulky. Grafika titulků má svou vlastní paletu, takže je vhodné předtím zavolat funkci *gfade_out(1)* a potom nechat tento příkaz zpravit paletu jak je potřeba.
explosion *int Anim* where (int) elevation (0-2) damage (int)	Způsobí výbuch na daném hexu (*where*) v dané úrovni (*elevation*), který způsobí poškození (*damage*) všemu co se nachází v jeho dosahu.
fixed_param *int*	Vrací hodnotu fixovaného parametru skriptu. Toto se používá například při použití *add_timer_event* k předání infa zpět ke skriptu.
float_msg *void* who (ObjectPtr) msg (str) type (int)	Pokusí se o vytvoření plovoucí textové zprávy (*msg*) nad objektem (*who*) za použití dané barvy (*type*). Jsou zde ještě dva speciální typy plovoucích zpráv a to WARNING a SEQUENTIAL. WARNING se používá k vypsání zprávy do středu obrazovky (například jako upozornění o dokončení úkolu). SEQUENTIAL cykluje mezi barvami, takže dává příšerám možnost odlišit zprávy jedné od druhé.
game_ticks *int Time* seconds (int)	Navrací počet herních kmitů rovných danému počtu vteřin (*seconds*).
game_time *int Time*	Navrací aktuální herní čas ve kmitech.
game_time_advance *void Time* amount (int)	Postoupí v herním čase o daný počet kmitů (*amount*).
game_time_hour *int Time*	Navrací aktuální hodinu herního času v normálním formátu, ale bez oddělovače. Např. 7:21 am se bude jevit jako 721.
game_ui_disable *void Meta*	Vyřadí vstupy hráče do uživatelského prostředí (pro odstavení hráčových podnětů při sekvenci atd.) Někdy později *MUSÍTE* hráčovy vstupy znovu odblokovat, jinak nebude moci ovládat hru.

game_ui_enable *void Meta*	Obnoví hráčovy vstupy do uživatelského prostředí. Toto *MUSÍTE* provést relativně brzo po vyřazení vstupů, jinak hráč nebude moci cokoli dělat.
game_ui_is_disabled *int Meta*	Navrací True, jsou-li vyřazeny hráčovy vstupy, jinak navrací False.
gdialog_barter *int Dialog*	Sdělí dialogovému systému, aby se přepnul do obrazovky pro obchodování (nastaví modifikátor obchodu na 0)
get_critter_stat *int Critter* who (ObjectPtr) stat (int)	Navrací hodnotu daného atributu/statistiky (*stat*) daného objektu (*who*)
get_day *int Time*	Navrací aktuální den v měsíci herního času.
gdialog_mod_barter *int Dialog* modifier (+/- percent)	Sdělí dialogovému systému, aby se přepnul do obrazovky pro obchodování za použití daného modifikátoru obchodu (*modifier*).
get_month *int Time*	Navrací aktuální měsíc z roku herního času.
get_pc_stat *int Critter* pcStat (int)	Navrací hodnotu statistiky (*pcStat*), kterou má pouze hráčská postava (*obj_dude*). Tyto jsou k nalezení v souboru define.h a začínají předponou "PCSTAT_".
get_poison *Critter* who (ObjectPtr)	Navrací aktuální úroveň otravy dané příšery (*who*).
gdialog_set_barter_mod *void Dialog* mod (int)	Nastavuje aktuální modifikátor obchodu na dané procento (*mod*). Toto se používá ke zvýhodnění/znevýhodnění obchodu, i když byl vyvolán hráčem).
gfade_in *void Meta* time (int)	Provede ztmavení palety do černé barvy. Parametr *time* není použit.
gfade_out *void Meta* time (int)	Provede zesvětlení palety z černé barvy. Parametr *time* není použit.
giQ_Option *void Dialog* iq_test (int) msg_list (int) msg_num (int) target (procedure) reaction (int)	Umožní volbu v rozhovoru, pokud je inteligence hráčské postavy větší nebo rovna dané velikosti (*iq_test*). Text volby se načítá z daného seznamu (*msg_list*) na daném řádku (*msg_num*). Způsobí danou reakci (*reaction*) a pokud je tato volba vybrána, přesune se na danou proceduru (*target*).
give_exp_points *void* points (int)	Přidá hráči daný počet zkušenostních bodů (*points*).
global_var *int Map* var_index (unsigned int)	Navrací hodnotu globální proměnné (*GVAR_*). dané svým indexem (*var_index*).
goto_xy *Map*	

gSay_End *void Dialog* var_index (unsigned int)	Ukončí dialogovou sekvenci.
gSay_Message *void Dialog* msg_list (int) msg_num (int) reaction (int)	Nastaví odpověď rozhovoru s jedinou volbou [hotovo]. Parametr *msg_list* určuje, který seznam textů se použije, *msg_num* který se použije řádek.
gSay_Option *void Dialog* msg_list (int) msg_num (int) target (procedure) reaction (int)	Umožní volbu v rozhovoru. Text volby se bere z daného seznamu (*msg_list*) na daném řádku (*msg_num*). Způsobí danou reakci (*reaction*) a pokud je vybrána, přesune se na danou proceduru (*target*).
gSay_Reply *void Dialog* msg_list (int) msg_num (int)	Nastaví odpověď dialogu (to co říká ten, s kým mluví hráč).
gSay_Start *void Dialog*	Spustí novou dialogovou sekvenci.
has_skill *int Skill* who (ObjectPtr) skill (int)	Navrací aktuální hodnotu dovednosti i se základem (*skill*) objektu (*who*).
has_trait *int Critter* trait_type (int) who (ObjectPtr) trait (int)	Navrací hodnotu rysu (*trait*) daného typu (*trait_type* - viz. define.h) daného objektu (*who*). Toto se používá např. k zjištění, zda má postava určitý perk, rys, ai balík, je členem určitého týmu nebo na kterou stranu je natočená.
how_much *int Skill* val (int)	Navrací výsledek provedeného hodu dovednost vs. dovednost (o kolik se hody lišily). Toto vyžaduje, abyste před použitím této funkce zavolali jednu z funkcí hodu kostkou jako *roll_vs_skill, skill_contest* atd.
inven_count *int Critter* what (ObjectPtr)	Navrací počet zaplněných slotů inventáře objektu (*what*).
inven_ptr *(ObjectPtr) Critter* what (ObjectPtr) slotNum (int)	Navrací ukazatel na objekt v daném slotu (*slotNum*) inventáře daného objektu (*what*).
inven_unwield *void Critter*	Pokusí se přinutit self_obj k odložení jakékoli třímané zbraně / vybavení. Pokud jsou vypnuty animace, pak jen bezprostředně změní grafiku.
is_critical *int Skill* val (int)	Navrací True, je-li výsledek daného dovednostního hodu kritický (ať už úspěch nebo neúspěch), jinak vrací False.
is_loading_game *boolean Map*	Navrací True, pokud hra zrovna nahrává, jinak False. Toto se používá k tomu, aby se nestávaly špatnosti při nahrávání map kvůli tomu, že skript zrovna obsluhuje proceduru *map_enter_proc*.

is_skill_tagged *int Skill* skillNum (int)	Navrací True, je-li daná dovednost (*skillNum*) talent. Jinak navrací False.
is_success *int Skill* val (int)	Navrací True, pokud je výsledek daného dovednostního bodu úspěch. Při neúspěchu vrací False.
item_caps_adjust *int Inven* obj (ObjectPtr) amount (int)	Upravuje množství zátek v objektu (*obj*) o dané množství (*amount*).
item_caps_total *int Inven* obj (ObjectPtr)	Navrací aktuální počet zátek, které jsou v inventáři daného objektu (*obj*).
jam_lock *int Object* lockableObj (ObjectPtr)	Zasekne daný zámek (*lockableObj*), takže s ním nebude příštích přibližně 24 hodin možno manipulovat. Používá se při kritickém neúspěchu páčení.
kill_critter *void* obj (ObjectPtr) death_frame (int)	Zabije příšeru (*obj*) a rovnou ji umístí do vybraného snímku smrti (*death_frame*). Poznámka: toto neprovede animaci smrti příšery, ani neaktualizuje obrazovku! To znamená, že se toto používá hlavně ve skriptech které se spouštějí při vstupu do / odchodu z mapy (*map_init/map_exit*).
kill_critter_type *void Map* pid (int)	Zabije všechny příšery daného prototypu (*pid*). Viz. *kill_critter* výše.
language_filter_is_on *int (boolean) Meta*	Navrací True, je-li zapnuto filtrování vulgárních slov, jinak vrací False.
load_map *void Map* map_name (string) start_location (int)	Načte novou mapu (*map_name*) a odstraní všechny probíhající skripty. Počáteční pozice na nové mapě (*start_location*) se předává části map_init skriptu této mapy.
local_var *int Map* var_index (unsigned int)	Navrací hodnotu lokální proměnné (*LVAR_*) daného indexu (*var_index*).
map_first_run *int Map*	Navrací True, běží-li aktuální mapa poprvé (jinými slovy, nebyla načtena z uložené pozice).
map_is_known *int Meta* mapNum (int)	Navrací True, pokud hráč zná index dané mapy (*mapNum*), False pokud ne.
map_known *int Meta* mapNum (int)	Navrací True, pokud hráč zná danou mapu (*mapNum*), False pokud ne.
map_var *int Map* var_index (unsigned int)	Navrací hodnotu globální mapové proměnné (*MVAR_*) daného indexu (*var_index*).

message_str *char ** list (int) msg_num (int)	Navrací řetězec z daného seznamu (*list*), který se nachází na daném řádku (*msg_num*).
move_to *int Map* obj (ObjectPtr) tile_num (int) elev (0-2)	Okamžitě přesune objekt (*obj*) na určenou pozici (*tile_num*) v dané úrovni (*elev*).
move_obj_inven_to_obj *int Inven* srcObj (ObjectPtr) destObj (ObjectPtr)	Přesune obsah inventáře jednoho objektu (*srcObj*) do inventáře jiného objektu (*destObj*).
obj_art_fid *(ObjectPtr) Object* obj (ObjectPtr)	Navrací *fid* (frame id) daného objektu (*obj*).
obj_being_used_with *(ObjectPtr) Object*	Navrací ukazatel na objekt, který je použit na jiný objekt.
obj_can_hear_obj *boolean Map* src_obj (ObjectPtr) dst_obj (ObjectPtr)	Navrací True, pokud je zdrojový objekt (*src_obj*) schopen slyšet cílový objekt (*dest_obj*). Jsou v tom započítány modifikace za vzdálenost, činnost (stání/chůze/běh) a použití dovedností (plížení atd.).
obj_can_see_obj *boolean Map* src_obj (ObjectPtr) dst_obj (ObjectPtr)	Navrací True, má-li zdrojový objekt (*src_obj*) volný výhled (line-of-sight – LOS) na cílový objekt (*dest_obj*). Pokud jsou to objekty příšer, započítávají se sem výsledky hodů na vnímání, plížení atd.
obj_carrying_pid_obj *(ObjectPtr) Object* who (ObjectPtr) pid (int)	Navrací ukazatel na instanci objektu daného prototypu (*pid*), pokud je v inventáři objektu *who*.
obj_close *void Object* what (ObjectPtr)	Pokusí se zavřít daný objekt (*what*), pokud je to objekt zavíratelného typu (dveře, truhla...).
obj_drop_everything *void Inven* who (ObjectPtr)	Způsobí, že příšera (*who*) odhodí všechny předměty z inventáře na zem.
obj_is_carrying_obj_pid *boolean Object* obj (ObjectPtr) pid (int)	Navrací počet objektů daného prototypu (*pid*), které jsou v inventáři objektu (*obj*).
obj_is_locked *int Object* what (ObjectPtr)	Navrací True, je-li objekt (*what*) zamčený. Pokud objekt není zamčený nebo ne5lze zamykat, vrací False.
obj_is_visible_flag *int Object* who (ObjectPtr)	Navrací True, je-li daný objekt (*obj*) viditelný, pokud ne, vrací False.

obj_is_open *int Object* what (ObjectPtr)	Navrací True, je-li daný objekt (*what*) otevřený. Je-li zavřený, nebo není-li otevíratelný, vrací False.
obj_item_subtype *int Object* obj (ObjectPtr)	Navrací podtyp objektu (*obj*) typu předmět (item) Např. jídlo, zbroj, zbraň atd.
obj_lock *void Object* what (ObjectPtr)	Pokusí se zamknout daný objekt (*what*), je-li zamykatelného typu.
obj_name *void Object* what (ObjectPtr)	Navrací řetězec, který obsahuje jméno daného objektu (*what*).
obj_on_screen *int Object* what (ObjectPtr)	Navrací True, pokud je daný objekt (*what*) zrovna vykreslený na obrazovce.
obj_open *void Object* what (ObjectPtr)	Pokusí se otevřít objekt (*what*), je-li to objekt otevíratelného typu. .
obj_pid *int Object* obj (ObjectPtr)	Navrací číslo prototypu (pid) daného objektu (*obj*).
obj_set_light_level *void Object* obj (ObjectPtr) intensity (1-100) distance (0 - 8)	Nastaví úroveň osvětlení daného objektu (*obj*) na danou intenzitu (*intensity* – procento maximální intenzity) s daným dosvitem (*distance*).
obj_type *int Object* obj (ObjectPtr)	Navrací typ objektu (*obj*). Tím bude předmět, zeď, dekorace atd.
obj_unlock *void Object* what (ObjectPtr)	Pokusí se odemčít daný objekt (*what*), je-li to objekt zamykatelného typu.
override_map_start *void Map* x (int) y (int) elev (0-2) rot (0-5)	Tato funkce se používá při načítání nové mapy a donutí hráče (*dude_obj*) aby začínal na určeném místě (*x, y*) v dané úrovni (*elev*) a s daným natočením (*rot*).
party_add *void Party* who (ObjectPtr)	Přidá danou příšeru (*who*) do seznamu členů družiny. Zároveň to nastaví, že se tato příšera nebude ukládat v mapách a další věci.
party_member_obj *ObjectPtr Party* pid (int)	Navrací ukazatel na člena družiny daného prototypu (*pid*). Pokud není tato příšera členem party, funkce navrací NULL.
party_member_count *ObjectPtr Party* countHidden (int)	Navrací aktuální počet příšer, které jsou členem družiny. Parametr *countHidden* určuje, zda se budou započítávat i skrytí členové.
party_remove *void Party* who (ObjectPtr)	Odstraní danou příšeru (*who*) ze seznamu členů družiny. Zároveň nastaví, že se skripty a mapy budou k této příšeře chovat odlišně.

pickup_obj *void Inven* obj (ObjectPtr)	Způsobí, že příšera *self_obj* spustí animaci a pokusí se sebrat daný objekt (*obj*).
play_gmovie *Meta*	Zahraje jedno z Fallout videí (celoobrazovkově, komprimovaně atd.)
play_sfx *Sound*	Zařadí do fronty na přehrání další zvukový efekt.
poison *Critter* who (ObjectPtr) amount (int)	Zvýší úroveň otravy dané příšery (*who*) o dané množství (*amount*).
proto_data *int OR string Object* pid (int) data_member (int)	Navrací hodnotu datového člena (*data_member*) daného prototypu (*pid*).
radiation_dec *Critter* who (ObjectPtr) amount (int)	Sníží hladinu ozáření dané příšery (*who*) o dané množství (*amount*). Poznámka: toto by mělo být, kvůli limitacím enginu, používáno pouze na hráče (*dude_obj*).
radiation_inc *Critter* who (ObjectPtr) amount (int)	Zvýší hladinu ozáření dané příšery (*who*) o dané množství (*amount*). Poznámka: toto by mělo být, kvůli limitacím enginu, používáno pouze na hráče (*dude_obj*).
random *int Script* min (int) max (int)	Navrací náhodnou hodnotu v rozmezí *min – max.*
reg_anim_animate *void Anim* what (ObjectPtr) anim (int) delay (int)	Přidá do sekvence animací daného objektu (*what*) jednu vhodnou animaci (*anim*) s daným zpožděním od předchozí animace (toto by mělo být vždy –1).
reg_anim_animate_forever *void Anim* what (ObjectPtr) anim (int) delay (int)	Přidá do sekvence animací daného objektu (*what*) jednu vhodnou animaci (*anim*) s daným zpožděním od předchozí animace (toto by mělo být vždy –1). Tato animace se bude stále opakovat, do té doby než ji něco v systému nepřeruší. Používejte velmi šetrně.
reg_anim_animate_reverse *void Anim* what (ObjectPtr) anim (int) delay (int)	Přidá do sekvence animací daného objektu (*what*) jednu vhodnou převrácenou animaci (*anim*) s daným zpožděním od předchozí animace (toto by mělo být vždy –1).
reg_anim_begin *void Anim*	Sdělí systému, aby spustil přehrávání sekvence animací.
reg_anim_clear *void Anim* object (ObjectPtr)	Ukončí všechny animace, které jsou registrovány na daném objektu (*object*).

reg_anim_end *void Anim*	Aktivuje sekvenci animací. Bez tohoto volání se žádná z animací nespustí. Poznámka: Všechny sekvence animací musí být zaregistrovány najednou! Jinak řečeno, nemůžete nechat skončit skript a dokončit registraci animací později.
reg_anim_obj_move_to_obj *void Anim* who (ObjectPtr) dest_obj (ObjectPtr) delay (int)	Přidá animaci, která způsobí chůzi příšery (*who*) k cílovému objektu (*dest_obj*) s daným zpožděním od předchozí animace (toto by mělo být vždy –1).
reg_anim_obj_run_to_obj *void Anim* who (ObjectPtr) dest_obj (ObjectPtr) delay (int)	Přidá animaci, která způsobí běh příšery (*who*) k cílovému objektu (*dest_obj*) s daným zpožděním od předchozí animace (toto by mělo být vždy –1).
reg_anim_obj_move_to_tile *void Anim* who (ObjectPtr) dest_tile (int) delay (int)	Přidá animaci, která způsobí chůzi příšery (*who*) k cílové dlaždici (*dest_tile*) s daným zpožděním od předchozí animace (toto by mělo být vždy –1).
reg_anim_obj_run_to_tile *void Anim* who (ObjectPtr) dest_tile (int) delay (int)	Přidá animaci, která způsobí běh příšery (*who*) k cílové dlaždici (*dest_tile*) s daným zpožděním od předchozí animace (toto by mělo být vždy –1).
reg_anim_play_sfx *void Anim* who (ObjectPtr) sfx_name (string) delay (int)	Přidá animaci, která způsobí, že objekt (*who*) přehraje daný zvukový efekt (*sfx_name*) v daném zpoždění od předchozí animace.
rm_fixed_timer_event *void Meta(Time)* who (ObjectPtr) fixed_val (int)	Odstraní všechny časované události připojené ke skriptu daného objektu (*obj*), které mají daný fixovaný parametr (*fixed_val*).
rm_obj_from_inven *void Inven* who (ObjectPtr) obj (ObjectPtr)	Odstraní objekt (*obj*) z inventáře jiného objektu (*who*). Poznámka: tato funkce nechá odstraněný předmět ležet na mapě na pozici (0,1). Pro přesun objektu zpět na mapu musíte zavolat funkci *move_to().*
rm_mult_objs_from_inven *int Inven* who (ObjectPtr) obj (ObjectPtr) count (int)	Odstraní počet (*count*) instancí objektu (*obj*) z inventáře jiného objektu (*who*). Poznámka: tato funkce nechá odstraněný předmět ležet na mapě na pozici (0,1). Pro přesun objektu zpět na mapu musíte zavolat funkci *move_to().* Tato funkce navrací aktuální počet instancí objektu, které byly odstraněny. Tuto hodnotu *MUSÍTE* uložit do nějaké proměnné (i když pro ni nemáte žádné využití).

rm_timer_event *void Meta(Time)* obj (ObjectPtr)	Odstraní (vymaže) všechny časované události připojené ke skriptu daného objektu (*obj*).
roll_dice *Skill*	Navrací výsledek provedeného hodu kostkou. NEFUNKČNÍ (tady si trošku nejsem jistý významem zkratky unimped – nejspíš se jedná o unimplemented – neimplementováno – pozn. překl.).
roll_vs_skill *int (roll_result) Skill* who (ObjectPtr) skill (int) modifier (int)	Navrací výsledek provedeného hodu proti dovednosti (*skill*) daného objektu (*who*), upraveného o daný modifikátor (*modifier* – může být i nula). Tato hodnota může být následně předána funkci *is_success* a *is_critical* k určení výsledku a funkci *how_much* k určení rozdílu hodu o který objekt uspěl / neuspěl.
rotation_to_tile *int (1…5) Map* srcTile (int) destTile (int)	Navrací natočení (0 – 5) potřebné k otočení se čelem na danou dlaždici (*destTile*) z dané dlaždice (*srcTile*).
running_burning_guy *int*	Navrací nastavení pro running-burning-guy (huh? – pozn. překl), které se definuje v obrazovce nastavení.
scr_return *void Script*	Nastaví návratovou hodnotu pro skript uzlu C-enginu, která bude použita v C kódu.
script_action *int Script*	Navrací akci, která aktivovala tento skript. Příklady obsahují požadavky na popis objektu (*description_proc*), oznámení prostorovému skriptu, že byl aktivován když byla něčím překročena jeho hranice (*spatial_proc*), nebo tepy příšer (*critter_proc)*, jinými slovy jí bylo řečeno aby se pohnula).
script_overrides *void Script*	Sděluje C-enginu, že tento skript potlačí standardní chování objektu. To znamená, že se C-engine nebude pokoušet dělat věci, které by normálně dělal, protože skript si to zpracuje sám. Toto je **DŮLEŽITÝ** příkaz! Obvykle bývá používán pro obecné akce hráče, kterými nějak pracuje s objektem, jako např. prohlížení si objektu (požadavek o popis), používání (třeba otevření dveří) nebo používání jiných objektů na objekt (použití paklíče nebo klíče na zamčené dveře).
self_obj *(ObjectPtr) Script*	Navrací ukazatel na objekt připojený k tomuto skriptu.
set_critter_stat *int Critter* who (ObjectPtr) stat (int) val (int)	Nastaví hodnotu daného atributu/statistiky (*stat*) daného objektu (*who*) na danou hodnotu (*val*).
set_exit_grids *void Map* markElev (elevation) mapID (int) elevation (int) tileNum (int) rotation (int)	Nastaví všechny odchodové mřížky na dané úrovni mapy (*markElev*), aby odkazovaly na danou mapu (*mapID* – může být –1, což znamená zůstaň na této mapě), úroveň (*elevation*), pozici (*tileNum*) a natočení (*rotation*).
set_global_var *void Map* var_index (unsigned int) value (int)	Nastaví globální proměnnou (*var_index*) na danou hodnotu (*value*).

set_light_level *void Map* level (int: 1-100)	Nastaví úroveň okolního osvětlení (*level*). Hodnoty se mohou pohybovat od úplné temnoty až po plné světlo.
set_local_var *void Map* var_index (unsigned int) value (int)	Nastaví lokální proměnnou (*var_index*) na danou hodnotu (*value*).
set_map_var *void Map* var_index (unsigned int) value (int)	Nastaví globální mapovou proměnnou (*var_index*) na danou hodnotu (*value*).
set_map_start *void Map* x (int) y (int) elev (0-2) rot (0-5)	Nastaví startovní pozici a natočení této mapy, při její příští návštěvě.
set_obj_visibility *void Object* obj (ObjectPtr) visibility (boolean)	Nastaví příznak OBJ_OFF objektu (*obj*) – způsobí jeho nevykreslování.
signal_end_game *void*	Sdělí systému, že skript indikuje, že by se hra měla ukončit. Navrátí hráče do hlavního menu.
skill_contest *Skill*	Navrací výsledek provedeného hodu dovednost vs. dovednost pro účely funkcí *is_succes* a *is_critical*.
source_obj *(ObjectPtr) Script*	Navrací ukazatel na zdrojový objekt (aktivátor) této akce. Např. zdrojový objekt pro proceduru *pickup_proc* (zvednutí předmětu) bude příšera, která objekt zdvíhá.
start_dialogue *void Dialog* who (ObjectPtr) mood (int)	Spustí dialogový systém, vystředěný na příšeru (*who*) a začínající v dané náladě (*mood*). Toto volání nastaví všechny potřebná dialogová okna, grafiku hlavy atd. Pokud se tato funkce nevolá před klasickými dialogovými funkcemi (sayReply, sayMessage, sayOption atd.), pak se nezobrazí dialogové okno a bude zobrazen pouze text v šedém ohraničení.
start_gdialog *void Dialog* msgFileNum (int) who (ObjectPtr) mood (int) headNum (int) backgroundIdx (int)	Spustí dialogový systém, vystředěný na příšeru (*who*) a začínající v dané náladě (*mood*). Toto volání nastaví všechny potřebná dialogová okna, grafiku hlavy atd. Pokud se tato funkce nevolá před klasickými dialogovými funkcemi (sayReply, sayMessage, sayOption atd.), pak se nezobrazí dialogové okno a bude zobrazen pouze text v šedém ohraničení.
target_obj *(ObjectPtr) Script*	Navrací ukazatel na cílový objekt této akce.
terminate_combat *void Combat*	Sdělí bojovému systému, aby se předčasně ukončil. POUŽÍVEJTE OPATRNĚ. Toto nezabrání jinému (nebo možná tomu stejnému) skriptu, aby znova spustil souboj, takže se ujistěte, že jste přepnuli všechny příznaky nepřátelství atd.

tile_contains_obj_pid *boolean Map* tile (int) elev (0-2) pid (int)	Navrací True, pokud daný hex (*tile*) na dané úrovni mapy (*elev*) obsahuje objekt daného prototypu (*pid*).
tile_contains_pid_obj *boolean Map* tile (int) elev (0-2) pid (int)	Navrací ukazatel na první objekt daného prototypu (*pid*), který se nachází na daném hexu (*tile*) na dané úrovní mapy (*elev*).
tile_distance *int Map* tile1 (int) tile2 (int)	Navrací vzdálenost mezi dvěma hexy.
tile_distance_objs *int Map* obj1 (ObjectPtr) obj2 (ObjectPtr)	Navrací vzdálenost mezi dvěma objekty (mezi hexy, na kterých stojí).
tile_is_visible *boolean Map* tile (int)	Navrací True, je-li daný hex (*tile*) viditelný, tj. že objekt na něm umístěný bude vykreslen na obrazovku. Toto zahrnuje i hexy, které technicky mohou mít být svůj základ mimo obrazovku, ale objekty na nich položené zasahují do viditelné oblasti.
tile_num *int Map* obj (ObjectPtr)	Navrací číslo hexu daného objektu (*obj*).
tile_num_in_direction *int Map* start_tile (int) dir (0-5) distance (int)	Navrací číslo hexu v daném směru (*dir*) a vzdálenosti (*distance*) ze startovního hexu (*start_tile*).
town_known *int Meta* townArea (int)	Navrací True, pokud hráč zná danou část města (*townArea*). Jinak navrací False.
town_map *void*	Posílá požadavek hernímu enginu, aby vyvolal městskou mapu, která umožňuje přechod mezi jednotlivými částmi města.
use_obj *(ObjectPtr) Script* obj (ObjectPtr)	
use_obj_on_obj *(ObjectPtr) Script* item (ObjectPtr) targetObj (ObjectPtr)	Pokusí se použít objekt typu předmět (*item*) na jiný objekt (*targetObj*). Toto se dá použít např. pro příšeru, která vyléčí hráče stimpakem nebo použije klíč na dveře atd.
using_skill *boolean Skill* who (ObjectPtr) skill (int)	Navrací True, pokud je používána daná aktivní dovednost (*skill*), pokud ne vrací False. Příklad aktivní dovednosti je plížení nebo první pomoc..
violence_level_setting *int (boolean) Meta*	Navrací aktuální nastavení úrovně násilí. Hodnoty naleznete v souboru define.h.

wield_obj *void Inven* obj (ObjectPtr)	Nastaví animaci, která způsobí, že si příšera (*self_obj*) vezme do rukou objekt (*obj*) z jejího inventáře.
wield_obj_critter *void Inven* who (ObjectPtr) obj (ObjectPtr)	Nastaví animaci, která způsobí, že si příšera (*who*) vezme do rukou objekt (*obj*) z jejího inventáře.
wm_area_set_pos *void* areaIdx (int) xPos (int) yPos (int)	Nastaví koordináty na mapě světa pro dané město (*areaIdx*) na danou xovou (*xPos*) a yovou (*yPos*) pozici.
world_map *void*	Posílá požadavek hernímu enginu, aby vyvolal mapu světa, která umožňuje přechod mezi lokacemi.
world_map_x_pos *int*	Navrací aktuální X-ovou souřadnici družiny na mapě světa.
world_map_y_pos *int*	Navrací aktuální Y-ovou souřadnici družiny na mapě světa.

Souhrn akcí skriptů

Description	Objekt	Požadavek na podrobné prozkoumání objektu.
Combat	Příšera	Probíhá bojová akce.
Create	Skript	Skript se právě vytváří (NEFUNKČNÍ).
Critter	Příšera	Probíhá tep skriptu příšery.
Damage	Objekt	Objekt obdržel poškození.
Destroy	Skript	Skript se právě ničí.
Drop	Předmět	Objekt je odhazován z inventáře jiného objektu na zem. (NEFUNKČNÍ).
Look At	Objekt	Požadavek na stručné prozkoumání objektu.
Map Enter	Mapa	Vstup do mapy (mapa byla právě načtena).
Map Exit	Mapa	Opuštění mapy (mapa bude uložena jako uložená hra).
Map Update	Mapa	Aktualizace mapy (po změně úrovně, osvětlení atd.).
None	Skript	Prázdná akce
Pickup	Předmět	Pokus o zvednutí předmětu a umístění do inventáře, případně kradení, obírání mrtvol.
Spatial	Skript	Byla překročena hranice tohoto skriptu.
Start	Skript	Skript se poprvé spouští.
Talk	Příšera	Požadavek na spuštění dialogu.
Timed Event	Skript	Aktivovala se časovaná událost.
Use	Objekt	Pokus o použití objektu.
Use Object On	Objekt	Použití objektu na jiný objekt.
Use Skill On	Objekt	Použití dovednosti na objekt.

Skupiny akcí skriptů

Objekt:

Obecné akce, které mohou být provedeny s kterýmkoli objektem. Například, když něco požaduje k stručnému prozkoumání objektu, pak to bude platné pro jakýkoli normální objekt (kromě objektů uživatelského prostředí). Jednotlivé prototypy objektů určují, která akce bude povolená. Takže například, pokud nebude v prototypu objektu nastavena volba USE, nebude možno tento objekt použít.

Předmět:

Akce s předměty jsou akce určené pro ty objekty, které může hráč zvednout, položit a nosit. Prostě pro předměty.

Příšera:

Akce s příšerami jsou akce volané na objekty, které představují aktivní bytosti. Jinými slovy, tyto akce se spustí jen s 'inteligentními' objekty, které jsou téměř ve všech případech schopné pohybu, zahájení boje, rozhovoru atd.

Mapa:

Akce s mapami jsou určené pro speciální události spojené s mapou, jako třeba když se mapa načetla, uložila nebo byla aktualizována. Původně byly akce s mapami přístupné pouze pro skripty map, ale zjistilo se, že by mohly být potřebné i v jiných skriptech. Tyto akce jsou velmi užitečné při nastavování proměnných při startu mapy. Když poprvé vstoupíte do mapy, první ze všeho se spustí skript této mapy a potom všechny další skripty, které mají nastaven příznak pro volání po vstupu. Toto může být použito k ukládání proměnných mapy a pro jejich následný 'export'. Potom si je může jakýkoli skript, který některou z těchto proměnných vyžaduje, 'importovat', protože je garantováno že existují.

Skript:

Akce se skripty se týkají jakýchkoli volání specifických pro skripty, jako při zničení skriptu, aktivaci prostorového skriptu nebo při spuštění načasované události. Tyto akce nepřichází přímo od hráče (jako u akcí s objekty), ale od skriptovacího systému.

Popis akcí skriptů

Description:

Hráč se pokouší prozkoumat objekt tohoto skriptu. Standardní chování je vytisknutí obsáhlého popisu z prototypu objektu do okna pro zprávy v levém dolním rohu obrazovky.

Combat:

Právě probíhá boj a nastala jedna z následujících událostí. Tyto události dovolují skriptu reagovat na průběh boje.

- HIT_SUCCEEDED – Objekt tohoto skriptu úspěšně zasáhl svůj cíl. Toto může být použito k udělení dodatečného poškození (např. radiace), nebo k počítání úspěšných útoků.
- SEQUENCING – Testuje se sekvence boje (aby se zjistilo, zda příšera nechce vstoupit/vystoupit z boje).
- TURN – Začíná další bojové kolo. Můžete potlačit standardní chování kola, abyste zabránili příšeře reagovat na boj (nebo ji můžete přinutit udělat něco speciálního)
- NONCOM_TURN – (NEFUNKČNÍ).

Create:

(NEFUNKČNÍ).

Critter:

Probíhá tep příšery. Toto v podstatě probíhá tak často, jak to jen jde (klidně i několikrát za vteřinu), takže to umožňuje skriptu reagovat na podněty z prostředí nebo na změny příznaků atd. Zde může být umístěn kód, který umožní příšeře volnou chůzi po mapě.

Damage:

Něco způsobilo poškození objektu tohoto skriptu. Ve většině případů to znamená, že příšeru někdo zasáhl v boji. Zde se může nastavit příznak nepřátelství (který může později ovlivnit reakce v rozhovoru nebo v úkolech), nebo se zde může příšera vyléčit aby nezemřela (toto není zrovna nejlepší postup, ale v několika málo případech může být užitečný, třeba když chcete mít jistotu, že příšera předtím než zemře něco řekne nebo udělá).

Destroy:

Tento skript bude zničen a spolu s všemi připojenými událostmi (např. načasovanými) bude odstraněn ze systému. To většinou znamená, že byla zabita příšera nebo byl použit předmět na jedno použití. Věci, které se mohu udělat v tomto místě jsou např. přidělení zkušenosti hráči (za zabití příšery nebo za splnění nějakého úkolu), aktualizace počítadel, jako je počet zářezů (počet příšer daného typu, které hráč zabil) nebo místní počítadlo (kolik zbývá v jeskyni radškorpiónů, kolik členů gangu ještě sužuje město atd.).

Drop:

Předmět bude odložen na zem z inventáře příšery.

Look At:

Hráč se pokouší prohlédnout si objekt tohoto skriptu. Standardní chování je vytištění krátkého popisu, jinými slovy zobrazení jména prototypu v okně pro zprávy v levém dolním rohu.

Map Enter:

Hráč vstoupil na mapu a tento skript je spuštěn před tím, než se dál pokračuje ve hře. Zde se můžou nastavit proměnné této mapy.

Map Exit:

Mapa bude opuštěna. Toto nastává, když hráč vstoupí na mapu města / světa.

Map Update:

Mapa bude aktualizována. Zde se může měnit úroveň okolního světla, jako třeba když se setmí, nebo hráč přejde do podzemní / nadzemní úrovně mapy.

None:

Nic se neděje. Toto by nemělo být nikdy voláno, ale je to zde jako implicitní akce.

Pickup:

Tato procedura znamená jiné věci pro různé druhy skriptů:

Skripty předmětů – Příšera se pokouší zvednout předmět. Pro kontejnerové objekty to znamená, že se je příšera pokouší otevřít / vybrat.

Skripty příšer – Příšera se pokouší krást z objektu tohoto skriptu.

Spatial:

Objekt se přesunul do prostoru působnosti tohoto skriptu. Tento skript možná bude vyžadovat kontrolu typu objektu (příšera, hráč atd.).

Start:

Skript je spuštěn úplně poprvé.

Talk:

Příšera se pokouší započít rozhovor, ať už to vyžadoval hráč nebo skript. Zde je potřeba nastartovat dialogový systém a umístit zde volání dialogů skriptovacího jazyka.

Timed Event:

Byla aktivována časovaná událost tohoto skriptu. Obvykle se tato událost dříve nastaví tímto skriptem. Dovolí skriptu provedení akcí se zpožděním.

Use:

Něco (obvykle příšera) se pokouší použít tento objekt. Funkce *source_obj* navrátí, kdo to byl. Téměř vždy to znamená, že se příšera (nejčastěji hráč) pokouší použít tento objekt, ale občas může toto volání provést jiný skript, což může být použito pro odlišení chování tohoto objektu. Například vstup do vaultu – pokud použijete ovládací počítač, pak se otevřou, ale pokud je použijete přímo, nic se nestane.

Use Object On:

Příšera se pokouší použít předmět na objekt tohoto skriptu. Toto může být třeba použití páčidla na dveře, stimpaku na příšeru, leguána-na-špejli na psa atd.

Use Skill On:

Příšera se pokouší použít dovednost na objekt tohoto skriptu. Standardní chování záleží na použité dovednosti.