X-Ray extensions

Материал из xrWiki

X-Ray extensions

Тип	Расширения движка
Авторы	malandrinus, kolmogor, K.D., RayTwitty (aka Shadows), Real Wolf, kstn, _призрак_, SkyLoader
Оф. репозиторий	GitHub
Оф. тема	AMK форум

Этот проект представляет собой набор добавлений, расширяющих возможности движка для скриптеров, а также, в ряде случаев, меняющих поведение игры. Изменения реализованы в виде бинарных патчей библиотек и исполняемых файлов.

Информация данной статьи актуальна для **187** ревизии официального репозитория.

Содержание

ТЧ

Скриптовый API

Класс game_object

Основные методы

Регистрация функций в движке

```
// CActor
set_actor_visual(string)
float get_sprint_factor()
set_sprint_factor(float)
float get_actor_take_dist()
set_actor_take_dist(float)
float get_actor_max_weight()
set_actor_max_weight(float)
float get_actor_max_walk_weight()
set_actor_max_walk_weight(float)
open_inventory_box(game_object*)
set_camera_direction(vector)
update_camera_direction(game_object*)
enable_car_panel(bool)
attach_vehicle(game_object*)
detach_vehicle(vector)
set_goodwill_ex(int, int, int)
change_goodwill_ex(int, int, int)
float get_camera_fov()
set_camera_fov(float)
restore_camera_fov()
float get_hud_fov()
set_hud_fov(float)
set_hud_animation_channel(int)
set_hud_animation_callback_param(int)
set_use_hud_animation_callback(bool)
```

```
clear_personal_record(int, int)
uint actor_body_state()
bool is_actor_normal()
bool is_actor_crouch()
bool is_actor_creep()
bool is_actor_climb()
bool is_actor_walking()
bool is_actor_running()
bool is_actor_sprinting()
bool is_actor_crouching()
bool is_actor_creeping()
bool is_actor_climbing()

// CCar
float get_fuel()
set_fuel(float)
float get_fuel_consumption()
set_fuel_consumption(float)
float get_fuel_tank()

// CEntityAlive
heal_wounds(float)
update_condition()

// CGameObject
string specific_character()
int get_bone_visible(string, int)
int set_bone_visible(string, int)
int get_bone_id(string, int)
string get_bone_name()
bool has_visual()
ini_file* get_visual_ini()
bool nonscript_usable()

// CHangingLamp
set_lsf_speed(float)
set_lsf_amount(float)
set_lsf_smap_jitter(float)
float get_lsf_speed()

// CHolderCustom
game_object* get_holder_owner()

// CHudItem
play_hud_animation(string, bool)
uint get_hud_animation_remaining_time()
int get_hud_animation_length(string, int)
bool is_cyclic_hud_animation()
bool has_hud_animation(string)
set_hud_animation_end_time(int)
stop_hud_sounds()
uint get_hud_item_state()
int get_hud_bone_id(string, int)
vector get_hud_bone_pos(string)
int get_hud_bone_visible(string, int)
int set_hud_bone_visible(string, int)

// CInventoryBox
game_object* object_from_inv_box(int)
uint inv_box_count()

// CInventoryItem
float get_weight()

// CInventoryOwner
bool is_on_belt(game_object*)
bool is_in_ruck(game_object*)
bool is_in_slot(game_object*)
bool move_to_ruck(game_object*)
bool move_to_belt(game_object*)
```

```
bool move_to_slot(game_object*)
bool move_to_slot_and_activate(game_object*)
game_object* item_on_belt(int)
game_object* item_in_ruck(int)
uint belt_count()
uint ruck_count()
uint slot_number()
float get_inventory_weight()
invalidate_inventory()

// CObject
float radius()
string get_visual_name()
set_visual_name(string)

// CProjector
projector_on()
projector_off()
bool projector_is_on()
switch_projector(bool)

// CSpaceRestrictor
float get_shape_radius()

// CTorch
switch_torch(bool)
bool is_torch_enabled()
set_torch_range(float)
set_torch_color(vector)
set_torch_omni_range(float)
set_torch_omni_color(vector)
set_torch_glow_radius(float)
set_torch_spot_angle(float)
set_torch_color_animator(string)
switch_night_vision(bool)

// CWeapon
int get_wpn_bone_visible(string, int)
int set_wpn_bone_visible(string, int)
```

CActor

`set_actor_visual(string<path>)` - устанавливает путь до визуала.
`float<speed> get_sprint_factor()` - возвращает скорость бега.
`set_sprint_factor(float<speed>)` - устанавливает скорость бега.
`float<dist> get_actor_take_dist()` - возвращает дистанцию поднятия предметов.
`set_actor_take_dist(float<dist>)` - устанавливает дистанцию поднятия предметов.
`float<max_weight> get_actor_max_weight()` - возвращает максимальный вес рюкзака.
`set_actor_max_weight(float<max_weight>)` - устанавливает максимальный вес рюкзака.
`float<max_walk_weight> get_actor_max_walk_weight()` - возвращает максимально-переносимый вес рюкзака.
`set_actor_max_walk_weight(float<max_walk_weight>)` - устанавливает максимально-переносимый вес рюкзака.
`open_inventory_box(game_object*<box_obj>)` - открывает окно обыска инвентарного ящика.
`set_camera_direction(vector<yaw, pitch, roll>)` - устанавливает повороты камеры.
`update_camera_direction(game_object*<target_obj>)` - направляет камеру в позицию объекта.
`enable_car_panel(bool<is_visible>)` - устанавливает видимость панели машины (прогресс-бар состояния).
`attach_vehicle(game_object*<holder_obj>)` - садит ГГ в машину или станковое оружие.
`detach_vehicle(vector<exit_pos>)` - высаживает ГГ из машины в заданную позицию на уровне.

set_goodwill_ex(int<who_id>, int<to_whom_id>, int<goodwill>) - устанавливает отношения между персонажами.
change_goodwill_ex(int<who_id>, int<to_whom_id>, int<goodwill_change>) - изменяет отношения между персонажами на заданное значение.
float<fov> get_camera_fov() - возвращает текущий fov камеры.
set_camera_fov(float<fov>) - устанавливает fov камеры.
restore_camera_fov() - восстанавливает предыдущий fov камеры (заданный методом set_camera_fov).
float<dist> get_hud_fov() - возвращает расстояние от камеры до худа оружия.
set_hud_fov(float<dist>) - устанавливает расстояние от камеры до худа оружия.
set_hud_animation_channel(int<blend_channel>) - устанавливает номер канала блендинга анимации, который будет передан в колбек на завершение анимации (только при проигрывании анимации через метод play_hud_animation).
set_hud_animation_callback_param(int<param>) - устанавливает произвольный параметр, который будет передан в колбек на завершение анимации (только при проигрывании анимации через метод play_hud_animation).
set_use_hud_animation_callback(bool<use_callback>) - включает использование колбека на завершение анимации (только при проигрывании анимации через метод play_hud_animation).
clear_personal_record(int<id_from>, int<id_to>) - очищает отношения между двумя мобами по их id.
uint<state> actor_body_state() - возвращает номер текущего стейта.
bool is_actor_normal() - возвратит true, если ГГ просто стоит.
bool is_actor_crouch() - возвратит true, если ГГ присел.
bool is_actor_creep() - возвратит true, если ГГ глубоко присел.
bool is_actor_climb() - возвратит true, если ГГ находится на лестнице.
bool is_actor_walking() - возвратит true, если ГГ идет медленным шагом.
bool is_actor_running() - возвратит true, если ГГ идет быстрым шагом.
bool is_actor_sprinting() - возвратит true, если ГГ бежит.
bool is_actor_crouching() - возвратит true, если ГГ двигается в присяде.
bool is_actor_creeping() - возвратит true, если ГГ двигается в глубоком присяде.
bool is_actor_climbing() - возвратит true, если ГГ двигается на лестнице.

CCar

float<fuel> get_fuel() - возвращает текущее количество топлива.
set_fuel(float<fuel>) - устанавливает текущее количество топлива.
float<consumption> get_fuel_consumption() - возвращает расход топлива.
set_fuel_consumption(float<consumption>) - устанавливает расход топлива.
float<tank_size> get_fuel_tank() - возвращает размер топливного бака.

CEntityAlive

heal_wounds(float<factor>) - уменьшает силу кровотечения.
update_condition() - принудительно обновляет внутренние параметры состояния персонажа (жизнь, выносливость, кровотечение и т.д.).

CGameObject

string<profile_name> specific_character() - возвращает название профиля персонажа.
int<is_visible>[0 or 1] get_bone_visible(string<bone_name>) - возвращает видимость кости по имени.
set_bone_visible(string<bone_name>, int<visible>[0 or 1]) - устанавливает видимость кости по имени.
int<bone_id> get_bone_id(string<bone_name>) - возвращает id кости по имени.
string<bone_name> get_bone_name() - возвращает имя кости по id (для правильной работы

необходимо использовать скриптовую обвязку get_bone_name_by_id #Функции).
bool<visual> has_visual() - возвратит true, если объект имеет визуал.
ini_file* get_visual_ini() - возвращает ссылку на внутренний конфиг модели.
bool<is_usable> nonscript_usable() - возвратит true, если над объектом разрешены нескриптовые действия.

CHangingLamp

set_lsf_speed(float<speed>) - устанавливает скорость затухания тени (чем больше - тем быстрее).
set_lsf_amount(float<amount>) - устанавливает максимальное ослабление тени (1.0 - пропадает до нуля, 0.0 - не ослабляется вовсе).
set_lsf_smap_jitter(float<jitter>) - устанавливает коэффициент размытия теней (чем больше число, тем больше размытие с отдалением от источника).
float<speed> get_lsf_speed() - возвращает скорость затухания тени.
Для работы данных методов необходимы дополнительные правки движка и шейдеров. Этот функционал широко используются в моде OGSE/OGSR.

CHolderCustom

game_object*<owner_obj> get_holder_owner() - возвращает владельца машины или станкового оружия.

CHudItem

play_hud_animation(string<anim_name>, bool<mix_in>) - проигрывает анимацию с заданным именем.
uint<remaining_time> get_hud_animation_remaining_time() - возвращает время, оставшееся до окончания текущей анимации.
int<length> get_hud_animation_length(string<anim_name>) - возвращает длительность анимации по её имени.
bool<is_cyclic> is_cyclic_hud_animation() - возвратит true, если текущая анимация зациклена.
bool<has_anim> has_hud_animation(string<anim_name>) - возвратит true, если анимация есть в модели.
set_hud_animation_end_time(int<end_time>) - устанавливает время конца текущей анимации.
stop_hud_sounds() - останавливает все звуки худа.
uint<state> get_hud_item_state() - возвращает номер текущего стейта.
int<bone_id> get_hud_bone_id(string<bone_name>) - возвращает id кости по имени.
vector<pos> get_hud_bone_pos(string<bone_name>) - возвращает позицию кости по имени.
int<is_visible>[0 or 1] get_hud_bone_visible(string<bone_name>) - возвращает видимость кости по имени.
set_hud_bone_visible(string<bone_name>, int<visible>[0 or 1]) - устанавливает видимость кости по имени.

CInventoryBox

game_object*<item_obj> object_from_inv_box(int<object_number>) - возвращает предмет в ящике по номеру.
uint<items_count> inv_box_count() - возвращает количество предметов в ящике.

CInventoryItem

float<weight> get_weight() - возвращает вес предмета.

CInventoryOwner

`bool<is_on_belt> is_on_belt(game_object*<item_obj>)` - возвратит true, если предмет на поясе.
`bool<is_in_ruck> is_in_ruck(game_object*<item_obj>)` - возвратит true, если предмет в рюкзаке.
`bool<is_in_slot> is_in_slot(game_object*<item_obj>)` - возвратит true, если предмет в слоте.
`move_to_ruck(game_object*<item_obj>)` - перемещает предмет в рюкзак.
`move_to_belt(game_object*<item_obj>)` - перемещает предмет на пояс.
`move_to_slot(game_object*<item_obj>)` - перемещает предмет в слот.
`move_to_slot_and_activate(game_object*<item_obj>)` - перемещает предмет в слот и активирует его.
`game_object*<item_obj> item_on_belt(int<object_number>)` - возвращает предмет на поясе по номеру.
`game_object*<item_obj> item_in_ruck(int<object_number>)` - возвращает предмет в рюкзаке по номеру.
`uint<count> belt_count()` - возвращает количество предметов на поясе.
`uint<count> ruck_count()` - возвращает количество предметов в рюкзаке.
`uint<number> slot_number()` - возвращает количество слотов.
`float<total_weight> get_inventory_weight()` - возвращает суммарный вес инвентаря.
`invalidate_inventory()` - принудительно обновляет все данные в инвентаре.

CObject

`float<radius> radius()` - возвращает виртуальный радиус объекта.
`string<path> get_visual_name()` - возвращает путь до визуала объекта.
`set_visual_name(string<path>)` - устанавливает путь до визуала объекта.

CProjector

`projector_on()` - включает прожектор.
`projector_off()` - выключает прожектор.
`bool<is_on> projector_is_on()` - возвратит true, если прожектор включён.
`switch_projector(bool<switch_on>)` - переключает прожектор.

CSpaceRestrictor

`float<radius> get_shape_radius()` - возвращает радиус рестриктора.

CTorch

`switch_torch(bool<switch_on>)` - переключает фонарь.
`bool<is_on> is_torch_enabled()` - возвратит true, если фонарь включён.
`set_torch_range(float<range>)` - устанавливает дальность основного света фонаря.
`set_torch_color(vector<R, G, B>)` - устанавливает цвет основного света от фонаря.
`set_torch_omni_range(float<range>)` - устанавливает дальность амбиент-света фонаря.
`set_torch_omni_color(vector<R, G, B>)` - устанавливает цвет амбиент-света от фонаря.
`set_torch_glow_radius(float<radius>)` - устанавливает радиус глоу-эффекта от фонарика.
`set_torch_spot_angle(float<angle>)` - устанавливает радиус светового пятна от фонарика.
`set_torch_color_animator(string<path>)` - устанавливает путь до аниматора цвета.
`switch_night_vision(bool<switch_on>)` - переключает состояние ПНВ.

CWeapon

`int<is_visible>[0 or 1] get_wpn_bone_visible(string<bone_name>)` - возвращает видимость кости оружия по имени.
`set_wpn_bone_visible(string<bone_name>, int<visible>[0 or 1])` - устанавливает видимость кости оружия по имени.

Хак для изменения значений в памяти объектов

Регистрация функций в движке

```
// CActor
float get_actor_float(int)
set_actor_float(vector, float, int)
int get_actor_int(string, int)
set_actor_int(int, int)
int get_actor_int16(string, int)
string get_actor_shared_str()
int set_actor_shared_str(string, int)
float get_actor_condition_float(int)
set_actor_condition_float(vector, float, int)
int get_actor_condition_int(string, int)

// CCar
float get_car_float(int)
set_car_float(vector, float, int)
int get_car_int(string, int)
set_car_int(int, int)
int get_car_int16(string, int)
set_car_int16(int, int)

// CCustomMonster
float get_custom_monster_float(int)
int get_custom_monster_int(string, int)

// CGameObject
float get_go_float(int)
set_go_float(vector, float, int)
int get_go_int(string, int)
set_go_int(int, int)
int get_go_int16(string, int)
set_go_int16(int, int)
string get_go_shared_str()
int set_go_shared_str(string, int)

// CHolderCustom
int get_holder_int(string, int)

// CHudItem
float get_hud_float(int)
set_hud_float(vector, float, int)
string get_hud_shared_str()
int set_hud_shared_str(string, int)
int save_hud_bone_float(string, int)

// CInventoryItem
float get_inventory_item_float(int)
set_inventory_item_float(vector, float, int)
int get_inventory_item_int(string, int)
set_inventory_item_int(int, int)
int get_inventory_item_int8(string, int)
set_inventory_item_int8(int, int)
int get_inventory_item_int16(string, int)
set_inventory_item_int16(int, int)
string get_inventory_item_shared_str()
int set_inventory_item_shared_str(string, int)

// CWeapon
float get_wpn_float(int)
```

```
set_wpn_float(vector, float, int)
int get_wpn_int(string, int)
set_wpn_int(int, int)
int get_wpn_int8(string, int)
set_wpn_int8(int, int)
int get_wpn_int16(string, int)
set_wpn_int16(int, int)
string get_wpn_shared_str()
int set_wpn_shared_str(string, int)

// CWeaponMagazinedWGrenade
int get_wpn_gl_int(string, int)

// Memory access functions
float get_memory_float(int)
set_memory_float(vector, float, int)
int get_memory_int(string, int)
set_memory_int(int, int)
int get_memory_int8(string, int)
set_memory_int8(int, int)
int get_memory_int16(string, int)
set_memory_int16(int, int)
```

Методы предназначены для изменения тех значений в памяти игровых объектов, которые недоступны через стандартные средства.
Применение всех функций практически одинаково, поэтому далее будет рассматриваться только пример на классе *CGameObject*.

Смещение - это позиция нужной переменной в памяти относительно начала класса. Узнать смещение можно несколькими способами:

1. посмотреть в папке help официального репозитория;
2. при помощи движка с отладочной информацией через дизассемблер (например, в *IDA Pro* выбрав нужную структуру во вкладке *Local Types*);
3. сканируя все смещения и выводя их значения в лог (однако, чтобы выяснить какое значение соответствует какому свойству класса, необходимо проводить дополнительные проверки).

Функция для сканирования значений класса CActor
mode - режим поиска (`"int"` - целочисленный тип, `"float"` - число с плавающей запятой)
range - диапазон сканирования (например, 2000)

```
function scan_actor_values(mode, range)
	get_console():execute("clear_log")
	log1("~ scan "..mode.." values")
	for i = 1, range do
		if mode == "int" then
			log1(i.." = "..db.actor:get_actor_int(nil, i))
		elseif mode == "float" then
			log1(i.." = "..db.actor:get_actor_float(i))
		end
	end
	flush_log()
end
```

Результат работы функции записывается в лог игры.

`float<value> get_go_float(int<offset>)` - возвращает значение с плавающей точкой по заданному смещению.
`set_go_float(nil, float<value>, int<offset>)` - устанавливает значение с плавающей точкой по заданному смещению (первый аргумент - заглушка).

`int<value> get_go_int(nil, int<offset>)` - возвращает целочисленное значение по заданному смещению (первый аргумент - заглушка).
`set_go_int(int<offset>, int<value>)` - устанавливает целочисленное значение по заданному смещению.
`string<value> get_go_shared_str()` - возвращает строковое значение по заданному смещению (смещение задается через глобальную функцию `set_int_arg0(int<offset>)`).
`set_go_shared_str(string<value>, int<offset>)` - устанавливает строковое значение по заданному смещению.

Важно!

При поиске значений для ГГ необходимо учитывать тот факт, что часть из них находится в отдельном классе - *CActorCondition*. Сила прыжка, сытость, алкоголизм и другие свойства расположены именно тут.

Методы для получения адреса базового объекта

Регистрация функций в движке

```
uint cast_car()
uint cast_game_object()
uint cast_hud_item()
uint cast_inventory_box()
uint cast_inventory_item()
uint cast_weapon()
```

Методы позволяют получить адрес в памяти, откуда начинается базовый объект (адрес в большинстве случаев не совпадает с началом объекта в целом). Результат можно использовать в разных хаках, когда известно смещение относительно конкретного класса.

Методы для проверки типа объекта

Регистрация функций в движке

```
bool is_actor()
bool is_ammo()
bool is_anomaly()
bool is_antirad()
bool is_artefact()
bool is_binoculars()
bool is_bottle_item()
bool is_car()
bool is_custom_monster()
bool is_eatable_item()
bool is_entity_alive()
bool is_explosive()
bool is_food_item()
bool is_game_object()
bool is_grenade()
bool is_grenade_launcher()
bool is_hanging_lamp()
bool is_helicopter()
bool is_holder()
bool is_hud_item()
bool is_inventory_box()
bool is_inventory_item()
bool is_inventory_owner()
bool is_knife()
```

```
bool is_medkit()
bool is_missile()
bool is_monster()
bool is_outfit()
bool is_physics_shell_holder()
bool is_projector()
bool is_scope()
bool is_script_zone()
bool is_silencer()
bool is_space_restrictor()
bool is_stalker()
bool is_torch()
bool is_trader()
bool is_weapon()
bool is_weapon_gl()
bool is_weapon_magazined()
bool is_weapon_pistol()
bool is_weapon_shotgun()
```

Методы предназначены для проверки типа объекта (к какому классу принадлежит). Применение достаточно простое:

```
if obj:is_<class_name>() then
...
end
```

Методы не имеющие привязки к определенному объекту

Регистрация функций в движке

```
game_object* get_object_arg_1(int)
set_object_arg_1(game_object*)
set_calibrating_vector(vector)
set_vector_global_arg_1(vector)
set_vector_global_arg_2(vector)
set_vector_global_arg_3(vector)
set_vector_global_arg_4(vector)
```

Эти методы предназначены для передачи или получения значений в(из) другие(их) методы(ов), в которых, по причине отсутствия необходимого прототипа, штатно это сделать нельзя. Могут вызываться для любого типа объекта в игре.

Класс matrix

Регистрация функций в движке

```
mul_43(matrix, matrix)
transform_tiny(vector, vector, vector, vector)
transform_tiny1(vector, vector, vector, vector)
transpose(float)
```

`mul_43(matrix<m1>, matrix<m2>)` - умножает две матрицы без последнего столбца и записывает результат в матрицу для которой вызывался метод.
`transform_tiny(vector<res>, vector<v>)` - умножает матрицу на вектор (перевод из одной системы координат в другую), с записыванием результата в вектор res.

`transform_tiny1(vector<v>)` - умножает матрицу на вектор (перевод из одной системы координат в другую), с записыванием результата в сам вектор.
`transpose()` - транспонирует матрицу (зеркалирование относительно главной диагонали).

Класс CTime

Регистрация функций в движке

```
set_value(int, int, int)
int, int, int, int, int, int, int get_value()
```

`set_value(int<lv>, int<gv>)` - устанавливает значение объекта из компонент.
`int<lv>, int<gv> get_value()` - возвращает младшую и старшую часть счетчика.

Класс CALifeSimulator

Регистрация функций в движке

```
CSE_Abstract* teleport_object(string, vector, uint, uint, uint)
CSE_Abstract* assign_story_id(string, vector, uint, uint, uint)
```

`teleport_object(nil, vector<position>, uint<level_vertex_id>, uint<game_vertex_id>, uint<id>)` - телепортирует объект по его id в заданную позицию на уровне.
`assign_story_id(nil, nil, uint<id>, uint<story_id>)` - устанавливает story_id объекту по его id.

Класс CSE_Abstract

Свойства

`angle<vector>` - задает направление поворота объекта.

Класс CSE_ALifeObject

Регистрация функций в движке

```
use_ai_locations(bool)
```

`use_ai_locations(bool<use>)` - задает использование AI-локаций (сетки) для объекта.

Пространство имен level

Регистрация функций в движке

```
game_object* get_target_obj(int)
float get_target_dist()
int get_target_element(int)

int has_cam_effector(int)
int has_pp_effector(int)
bool has_indicators()
```

```
int advance_game_time(int)
game_object* get_second_talker(int)
int vertex_id(int)

float get_memory_float(string, int, bool, string)
int get_memory_int(int)

bool perform_ray_pick_query()
float get_ray_pick_dist()
game_object* get_ray_pick_obj(int)
int get_ray_pick_element(int)

CUIDialogWnd* get_inventory_wnd()
CUIDialogWnd* get_pda_wnd()
CUIDialogWnd* get_talk_wnd()
CUIDialogWnd* get_car_body_wnd()
CUIDialogWnd* get_trade_wnd()

int send_event_mouse_wheel(int)
int send_event_key_hold(int)
int send_event_key_release(int)
int send_event_key_press(int)

set_ce_time(float)
set_ce_amplitude(float)
set_ce_period_number(float)
set_ce_power(float)
bool add_ce()

vector get_tri_vertex1(int)
vector get_tri_vertex2(int)
vector get_tri_vertex3(int)

float get_tri_shootfactor(string, int, bool, string)
int get_tri_flags(int)
```

`game_object*<obj> get_target_obj()` - возвращает игровой объект (если он имеет шейпы), на который в данный момент направлена камера.
`float<dist> get_target_dist()` - возвращает дистанцию до точки, на которую в данный момент направлена камера.
`int<element> get_target_element()` - если камера направлена на игровой объект (который имеет шейпы), то возвращает номер кости, иначе возвращает номер полигона статической геометрии.

`int<is_cam_eff>[0 or +] has_cam_effector(int<effect_id>)` - вернет true, если заданный эффектор камеры в данный момент активен (0 - эффектор не активен, иначе - активен).
`int<is_pp_eff>[0 or +] has_pp_effector(int<effect_id>)` - вернет true, если заданный эффектор постпроцесса в данный момент активен (0 - эффектор не активен, иначе - активен).
`bool has_indicators()` - вернет true, если интерфейс худа сейчас показан.

`advance_game_time(int<time_ms>)` - прокручивает игровое время вперед на заданное количество миллисекунд.
`game_object*<obj> get_second_talker()` - возвращает игровой объект, с которым в данный момент открыто окно разговора.
`int<lvid> vertex_id()` - возвращает level_vertex_id по позиции на уровне (для правильной работы необходимо использовать скриптовую обвязку `level.vertex_id_by_pos` #Функции).

`float<value> get_memory_float(nil, int<addr>)` - возвращает число с плавающей точкой по абсолютному адресу.
`int<value> get_memory_int(int<addr>)` - возвращает целое число по абсолютному адресу.

`bool<check> perform_ray_pick_query()` - выполнит запрос на трассировку и вернет true, если она была успешна.
`float<dist> get_ray_pick_dist()` - если трассировка была успешна, вернет дистанцию до точки, иначе вернет ранее заданный диапазон трассировки.
`game_object*<obj> get_ray_pick_obj()` - если трассировка была успешна, вернет игровой объект.
`int<element> get_ray_pick_element()` - если луч пересек игровой объект, то вернет номер кости, иначе вернет номер полигона статической геометрии (при неудачной трассировке вернет -1).

`CUIDialogWnd*<inv_wnd> get_inventory_wnd()` - возвращает объект окна инвентаря.
`CUIDialogWnd*<pda_wnd> get_pda_wnd()` - возвращает объект окна КПК.
`CUIDialogWnd*<talk_wnd> get_talk_wnd()` - возвращает объект окна разговора.
`CUIDialogWnd*<carbody_wnd> get_car_body_wnd()` - возвращает объект окна обыска.
`CUIDialogWnd*<trade_wnd> get_trade_wnd()` - возвращает объект окна торговли.

`send_event_mouse_wheel(int<vol>)` - имитация события вращения колеса мыши.
`send_event_key_hold(int<DIK_keys>)` - имитация события удержания кнопки.
`send_event_key_release(int<DIK_keys>)` - имитация события отпускания кнопки.
`send_event_key_press(int<DIK_keys>)` - имитация события нажатия кнопки.

`set_ce_time(float<total_time>)` - устанавливает продолжительность эффектора шатания камеры.
`set_ce_amplitude(float<amp>)` - устанавливает максимальную амплитуду эффектора шатания камеры.
`set_ce_period_number(float<periods>)` - устанавливает количество циклов эффектора шатания камеры.
`set_ce_power(float<power>)` - устанавливает интенсивность эффектора шатания камеры.
`add_ce()` - запускает эффектор шатания камеры с ранее установленными параметрами.

`vector<pos> get_tri_vertex1(int<element>)` - возвращает первую вершину статического треугольника.
`vector<pos> get_tri_vertex2(int<element>)` - возвращает вторую вершину статического треугольника.
`vector<pos> get_tri_vertex3(int<element>)` - возвращает третью вершину статического треугольника.

`float<factor> get_tri_shootfactor(nil, int<element>)` - возвращает фактор пробиваемости материала геометрии (0.0 - непробиваемый, 1.0 - полностью пробиваемый).
`int<flags> get_tri_flags(int<element>)` - возвращает флаги материала геометрии (#mtlFlags).

Глобальное пространство имен

Регистрация функций в движке

```
log1(string)
fail(string)
int flush_log(int, int)
int bind_to_dik(int, int)

int get_extensions_flags(int, int)
int set_extensions_flags(int, int)
int get_actor_flags(int, int)
int set_actor_flags(int, int)
int get_input_language(int, int)
int set_input_language(int, int)

int set_trade_filtration_on(int, int)
```

```
int set_trade_filtration_off(int, int)
int set_manual_grouping_on(int, int)
int set_manual_grouping_off(int, int)
int set_manual_highlight_on(int, int)
int set_manual_highlight_off(int, int)
int get_manual_highlight(int, int)
int set_highlight_color(int, int)

int sum_args(int, int)
int sub_args(int, int)

int GetGoodwill(int, int)
int update_inventory_window(int, int)
init_external_libs(string)

set_game_time(float, float)
int print_level_time(int, int)
int print_alife_time(int, int)
int set_ignore_game_state_update(int, int)

screenshot0(string)
screenshot1(string)
screenshot2(string)
screenshot3(string)

set_hud_inertia(float, float)
set_hud_inertia_param2(float, float)

float get_static_rescale_factor()
set_static_rescale_factor(float, float)

int set_int_arg0(int, int)
int set_int_arg1(int, int)
int set_int_arg2(int, int)
int set_int_arg3(int, int)
int set_int_arg4(int, int)
int set_int_arg5(int, int)
int set_int_arg6(int, int)
set_float_args_12(float, float)
set_float_args_34(float, float)
```

`log1(string<message>)` - выводит строку-сообщение в лог.
`fail(string<message>)` - принудительно останавливает игру и выводит сообщение в лог.
`flush_log()` - записывает содержимое лога в лог-файл (аналогично вызову консольной команды flush).
`int<DIK_keys> bind_to_dik(int<key_bindings>)` - переводит код команды-экшена в код клавиши, забинденной на этот экшен.

`int<ex_flags> get_extensions_flags()` - возвращает состояние флагов работы колбеков (1 - key_press, 2 - key_release, 4 - key_hold, 8 - mouse_wheel, 16 - mouse_move).
`set_extensions_flags(int<ex_flags>)` - устанавливает состояние флагов работы колбеков (1 - key_press, 2 - key_release, 4 - key_hold, 8 - mouse_wheel, 16 - mouse_move).
`int<flags> get_actor_flags()` - возвращает состояние флагов из перечисления psActorFlags.
`set_actor_flags(int<flags>)` - устанавливает состояние флагов из перечисления psActorFlags.
`int<lang>[0 or 1] get_input_language()` - возвращает номер языка для ввода в полях редактирования (0 - eng, 1 - rus).
`set_input_language(int<lang>[0 or 1])` - устанавливает номер языка для ввода в полях редактирования (0 - eng, 1 - rus).

`set_trade_filtration_on()` - включает режим скриптовой фильтрации предметов в окне торговли.

`set_trade_filtration_off()` - выключает режим скриптовой фильтрации предметов в окне торговли.
`set_manual_grouping_on()` - включает режим скриптовой группировки предметов в окне торговли и обыска.
`set_manual_grouping_off()` - выключает режим скриптовой группировки предметов в окне торговли и обыска.
`set_manual_highlight_on()` - включает режим скриптовой подсветки предметов в окне торговли и обыска.
`set_manual_highlight_off()` - выключает режим скриптовой подсветки предметов в окне торговли и обыска.
`int<is_highlight>[0 or 1] get_manual_highlight()` - возвращает включенность режима скриптовой подсветки предметов.
`set_highlight_color(int<color_id>, int<GetARGB(A, R, G, B)>)` - устанавливает соответствие между индексом цвета и кодом цвета.

`int<result> sum_args(int<val1>, int<val2>)` - производит сложение двух чисел и возвращает целочисленный результат (при ариф. операциях, Lua возвращает число с плавающей точкой).
`int<result> sub_args(int<val1>, int<val2>)` - производит вычитание двух чисел и возвращает целочисленный результат (при ариф. операциях, Lua возвращает число с плавающей точкой).

`int<goodwill> GetGoodwill(int<who_id>, int<to_whom_id>)` - возвращает текущее отношение между персонажами.
`update_inventory_window()` - обновляет все графические элементы окна инвентаря.
`init_external_libs()` - вызывает подключение дополнительных библиотек, прописанных в движке, внутри этой функции (используется в моде OGSE/OGSR).

`print_level_time()` - выводит отладочную информацию в лог о времени уровня.
`print_alife_time()` - выводит отладочную информацию в лог о времени алайфа.
`set_ignore_game_state_update()` - устанавливает запрет на выполнение одного цикла обновления внутренних параметров времени (используется в моде OGSE/OGSR).

`screenshot0(string<path>)` - создает скриншот в обычном режиме и сохраняет файл по указанному пути.
`screenshot1(string<path>)` - создает скриншот в режиме cubemap и сохраняет файл по указанному пути.
`screenshot2(string<path>)` - создает скриншот в режиме save и сохраняет файл по указанному пути.
`screenshot3(string<path>)` - создает скриншот в режиме levelmap и сохраняет файл по указанному пути.

`set_hud_inertia(float<speed>)` - устанавливает скорость инерции худа.
`set_hud_inertia_param2(float<shift>)` - устанавливает силу инерции худа.

`float<rescale> get_static_rescale_factor()` - возвращает коэффициент сжатия элементов окна на выбранном разрешении.
`set_static_rescale_factor(float<rescale>)` - устанавливает коэффициент сжатия элементов окна на выбранном разрешении.

Остальные методы предназначены для передачи или получения значений в(из) другие(их) методы(ов), в которых, по причине отсутствия необходимого прототипа, штатно это сделать нельзя.

Оконные классы

Класс CUIWindow

Регистрация функций в движке

```
DetachFromParent()
BringToTop()
Update()
float GetVPos()
float GetHPos()
float GetCursorX()
float GetCursorY()
float GetAbsolutePosX()
float GetAbsolutePosY()
```

`DetachFromParent()` - отсоединяет окно от родительского с последующем удалением.
`BringToTop()` - перемещает окно на вершину в иерархии дочерних окон.
`Update()` - дополнительно вызывает виртуальный метод обновления окна.
`float<x> GetVPos()` - возвращает положение окна по оси X.
`float<y> GetHPos()` - возвращает положение окна по оси Y.
`float<x> GetCursorX()` - возвращает положение мыши в окне по оси X.
`float<y> GetCursorY()` - возвращает положение мыши в окне по оси Y.
`float<x> GetAbsolutePosX()` - возвращает абсолютное положение окна по оси X.
`float<y> GetAbsolutePosY()` - возвращает абсолютное положение окна по оси Y.

Класс CUIStatic

Регистрация функций в движке

```
SetTextComplexMode(bool)
AdjustWidthToText()
AdjustHeightToText()
SetVTextAlign(uint)
SetTextPos(float, float)
CanRotate(bool)
```

`SetTextComplexMode(bool<mode>)` - включает режим комплексной работы с текстом.
`AdjustWidthToText()` - растягивает ширину окна под размер текста.
`AdjustHeightToText()` - растягивает высоту окна под размер текста.
`SetVTextAlign(uint<align>)` - задает выравнивание текста по вертикали (#EVTextAlignment).
`SetTextPos(float<x>, float<y>)` - задает положение текста внутри окна.
`CanRotate(bool)` - устанавливает возможность вращения текстуры окна.

Класс CUIListWnd

Регистрация функций в движке

```
SetSelectedItem(int)
```

`SetSelectedItem(int<idx>)` - устанавливает выбранный элемент списка по его индексу.

Класс CUIComboBox

Регистрация функций в движке

```
AddItem(string)
string GetText()
```

`AddItem(string<text>)` - добавляет новый элемент с заданным текстом.
`string<text> GetText()` - возвращает текущую надпись в окне выпадающего списка.

Класс CUITrackBar

Регистрация функций в движке

```
float GetFValue()
bool IsChanged()
```

`float<value> GetFValue()` - возвращает текущее значение ползунка.
`bool<changed> IsChanged()` - возвратит true, если значение было изменено с момента последнего сохранения окна опций.

Скриптовые колбеки

Рекомендуется использовать таблицу с идентификаторами колбеков #callback, чтобы не запутаться в их номерах.

Название	Номер	Вызывается для	Аргументы	Описание
on_key_press	123	CActor	int<dik>[DIK_keys]	нажатие клавиши (#Активация колбеков ввода)
on_key_release	124	CActor	int<dik>[DIK_keys]	отпускание клавиши (#Активация колбеков ввода)
on_key_hold	125	CActor	int<dik>[DIK_keys]	удерживание клавиши (#Активация колбеков ввода)
on_mouse_wheel	126	CActor	int<vol>	движение мыши (#Активация колбеков ввода)
on_mouse_move	127	CActor	int<x>, int<y>	вращение колеса мыши (#Активация колбеков ввода)
anomaly_hit	128	CCustomZone	game_object*<obj>	хит объекта аномалией
drop_item_from_inventory	129	CActor	game_object*<obj>	выкидывание предмета через интерфейс инвентаря (контекстное меню и клавиша G)
on_item_belt	130	CActor	game_object*<obj>	перемещение предмета на пояс
on_item_ruck	131	CActor	game_object*<obj>	перемещение предмета в рюкзак
on_item_slot	132	CActor	game_object*<obj>	перемещение предмета в слот
select_inventory_item	133	CActor	game_object*<obj>	выделение предмета в инвентаре
switch_torch	134	CTorch	int<light_on>[0 or 1]	переключение активности фонарика
set_dest_lvid	135	CAI_Stalker	int<level_vertex_id>	вызов после вызова метода `set_dest_level_vertex_id`
create_cell_item	136	CActor	game_object*<obj>	создание объекта *CUICellItem* в инвентаре (получить статик в этот момент можно при помощи вызова `CUIFrameWindow():GetTitleStatic()`)
attach_vehicle	137	CActor	game_object*<car>	посадка в машину
use_vehicle	138	CActor	game_object*<car>	использование машины (проверку на дистанцию необходимо организовать вручную)
detach_vehicle	139	CActor	game_object*<car>	выход из машины
after_save	140	CActor		вызов после сохранения игры
cell_item_focus_start	141	CActor	game_object*<obj>	получение фокуса объектом *CUICellItem* в инвентаре (получить статик в этот момент можно при помощи вызова `CUIFrameWindow():GetTitleStatic()`)
cell_item_focus_end	142	CActor	game_object*<obj>	потеря фокуса объектом *CUICellItem* в инвентаре (получить статик в этот момент можно при помощи вызова `CUIFrameWindow():GetTitleStatic()`)
group_cell_item	143	CActor	game_object*<obj1>	сравнение объектов для группировки в инвентаре (получить второй объект можно при помощи вызова `db.actor:get_object_arg_1()`, запретить группировку для текущего сравнения при помощи вызова `db.actor:set_object_arg_1(nil)`)
hit_effector	144	CActor	int<mob_type>[0 or 1], int<side>	появление хит-эффектора на экране (mob_type: 0 - stalker, 1 - monster)
set_goodwill	145	CActor	int<id_from>, int<id_to>	изменение отношений
before_update	150	CActor		вызов перед первым апдейтом
drop_item_in_box	151	CInventoryBox	game_object*<obj>	перемещение предмета в инвентарный ящик
before_hit	152	CEntityAlive	int<hit_data>, int<ignore_flags>	вызов перед получением хита (позволяет модифицировать параметры хита перед тем, как он произошел, однако требует дополнительных функций для работы с памятью)
npc_hit	153	CEntityAlive	game_object*<who>	вызов перед получением хита (позволяет установить игнорирование хита при помощи вызова `set_int_arg0(1)`, по смыслу частично повторяет функционал колбека *before_hit*, однако не требует дополнительных функций)
update_addons_visibility	154	CWeapon		обновление видимости костей аддонов оружия (для активации колбека необходимо использовать скриптовую обвязку `activate_visibility_updates` #Функции)
update_hud_addons_visibility	155	CWeapon		обновление видимости костей аддонов худа оружия (для активации колбека необходимо использовать скриптовую обвязку `activate_visibility_updates` #Функции)
before_use_item	156	CActor	game_object*<obj>	вызов перед использованием предмета
hud_animation_end	157	CActor	int<param>, int<blend_channel>	завершение проигрывания анимации худа через метод `play_hud_animation` (для активации колбека, а также для передачи параметров используются методы `set_use_hud_animation_callback`, `set_hud_animation_callback_param`, `set_hud_animation_channel`)
init_addons	158	CWeapon	int<wpn_type>[1 or 2]	смена аддонов оружия (1 - CWeaponMagazined, 2 - CWeaponMagazinedWGrenade)
select_pda_contact	180	CActor	int<id>	выбор контакта во вкладке КПК "Контакты"

Активация колбеков ввода

Для активации колбеков клавиатуры и мыши необходимо вызвать глобальную функцию `set_extensions_flags` и передать в нее флаги колбеков (1 - key_press, 2 - key_release, 4 - key_hold, 8 - mouse_wheel, 16 - mouse_move). Это нужно сделать на событии **net_spawn**, а в **net_destroy** обнулить ранее установленные флаги.

```
function actor_binder:net_spawn(data)
        ...
        set_extensions_flags(31) -- активирует все пять колбеков
        return true
end

function actor_binder:net_destroy()
        ...
        set_extensions_flags(0)
        object_binder.net_destroy(self)
```

```
end
```

Дополнительная информация

Подсветка текста в консоли

В связи с добавлением функции вывода произвольного текста в лог, стала доступна возможность выделять этот текст различными цветами.

Цвет	Спец-символ цвета	Код цвета в формате RGBA
Зеленый	"-"	(0, 255, 0, 255)
Красный	"!"	(255, 0, 0, 255)
Желтый	"~"	(255, 255, 0, 255)
Серый	"*"	(128, 128, 128, 255)
Бирюзовый	"#"	(0, 222, 205, 145)

Пример использования

```
log1("# Hello world!") -- текст будет отображаться бирюзовым цветом
```

После спец-символа цвета обязательно должен быть хотя бы один пробел, иначе строка будет выведена обрезанной.

Общие рекомендации

При использовании правок движка из этого проекта, рекомендуется расширить некоторые таблицы новыми значениями, например таблицы с идентификаторами колбеков, идентификаторами клавиш и т.п. Кроме того, ниже приведены некоторые функции, которые представляют из себя скриптовую обвязку для разрозненных движковых методов.

Перечисления

Код следует поместить в главный модуль **_g.script**.

callback

```
callback["on_key_press"] = 123
callback["on_key_release"] = 124
callback["on_key_hold"] = 125
callback["on_mouse_wheel"] = 126
callback["on_mouse_move"] = 127
callback["anomaly_hit"] = 128
callback["drop_item_from_inventory"] = 129
callback["on_item_belt"] = 130
callback["on_item_ruck"] = 131
callback["on_item_slot"] = 132
callback["select_inventory_item"] = 133
callback["switch_torch"] = 134
callback["set_dest_lvid"] = 135
callback["create_cell_item"] = 136
callback["attach_vehicle"] = 137
callback["use_vehicle"] = 138
callback["detach_vehicle"] = 139
callback["after_save"] = 140
callback["cell_item_focus_start"] = 141
callback["cell_item_focus_end"] = 142
```

```
callback["group_cell_item"] = 143
callback["hit_effector"] = 144
callback["set_goodwill"] = 145
callback["before_update"] = 150
callback["drop_item_in_box"] = 151
callback["before_hit"] = 152
callback["npc_hit"] = 153
callback["update_addons_visibility"] = 154
callback["update_hud_addons_visibility"] = 155
callback["before_use_item"] = 156
callback["hud_animation_end"] = 157
callback["init_addons"] = 158
callback["select_pda_contact"] = 180
```

DIK_keys

```
DIK_keys["MOUSE_4"] = 340
DIK_keys["MOUSE_5"] = 341
DIK_keys["MOUSE_6"] = 342
DIK_keys["MOUSE_7"] = 343
DIK_keys["MOUSE_8"] = 344
```

key_bindings

```
key_bindings["kSPRINT_TOGGLE"] = 8
key_bindings["kENGINE"] = 15
key_bindings["kARTEFACT"] = 30
key_bindings["kWPN_FIREMODE_PREV"] = 38
key_bindings["kWPN_FIREMODE_NEXT"] = 39
key_bindings["kPAUSE"] = 40
key_bindings["kCHAT_TEAM"] = 45
key_bindings["kACTIVE_JOBS"] = 53
key_bindings["kMAP"] = 54
key_bindings["kCONTACTS"] = 55
key_bindings["kVOTE_BEGIN"] = 57
key_bindings["kVOTE"] = 58
key_bindings["kVOTEYES"] = 59
key_bindings["kVOTENO"] = 60
key_bindings["kSPEECH_MENU_0"] = 63
key_bindings["kSPEECH_MENU_1"] = 64
key_bindings["kUSE_BANDAGE"] = 73
key_bindings["kUSE_MEDKIT"] = 74
key_bindings["kQUICK_SAVE"] = 75
key_bindings["kQUICK_LOAD"] = 76
```

EVTextAlignment

```
-- вертикальное центрирование текста
EVTextAlignment = {
        valTop                          = 0,
        valCenter               = 1,
        valBottom               = 2
}
```

mtlFlags

```
-- флаги материала геометрии
```

```
mtlFlags = {
        flBreakable                             = 1,                        -- 0    разрушаемый
        flBounceable                    = 4,                        -- 2    возможность рикошета (0 -
есть, 1 - нет)
        flSkidmark                              = 8,                        -- 3    оставляет тормозной
след
        flBloodmark                             = 16,                        -- 4    оставляет кровь
        flClimable                              = 32,                       -- 5    невидимая лестница
        flPassable                              = 128,                       -- 7    проходимый для
физических объектов
        flDynamic                               = 256,                      -- 8    динамический
объект
        flLiquid                                = 512,                      -- 9    жидкость (вода)
        flSuppressShadows               = 1024,                     -- 10   заглушает тени
        flSuppressWallmarks             = 2048,                       -- 11   заглушает отметины от
пуль
        flActorObstacle                 = 4096,                     -- 12   препятствие
(силовое поле) для ГГ
        flInjurious                             = 268435456,        -- 28   отбирает здоровье
        flShootable                             = 536870912,        -- 29   непростреливаемый
        flTransparent                   = 1073741824,       -- 30   непрозрачный
        flSlowDown                              = 2147483648        -- 31   замедляет движение
}
```

rq_target

```
-- флаги трассировки
rq_target = {
        rqtNone,
        rqtObject               = 1,
        rqtStatic               = 2,
        rqtShape                = 4,
        rqtObstacle             = 8,
        rqtBoth                 = 3, -- rqtObject + rqtStatic
        rqtDyn                  = 13, -- rqtObject + rqtShape + rqtObstacle
}
```

Функции

Глобальное пространство имен

```
-- возвращает имя кости по ее индексу
function _G.get_bone_name_by_id(obj, bone_id)
        set_int_arg0(bone_id)
        return obj:get_bone_name()
end

-- возвращает true, если колбеки видимости аддонов оружия активированы
function _G.get_visibility_updates_activated(wpn)
        return bit_and(wpn:get_wpn_int(nil, 936), 64) ~= 0
end

-- активирует колбеки видимости аддонов оружия
function _G.activate_visibility_updates(wpn, activate)
        local flags = wpn:get_wpn_int8(nil, 936)
        local new_flags = activate and bit_or(flags, 64) or bit_and(flags, 191)
        wpn:set_wpn_int8(936, new_flags)
end
```

level

```
-- прокручивает игровое время вперед на заданное количество минут, часов и дней
function level.change_game_time(m, h, d)
	level.advance_game_time((m or 0) * 60000 + (h or 0) * 3600000 + (d or 0) * 86400000)
	set_ignore_game_state_update()
end

-- возвращает level_vertex_id по позиции на уровне
function level.vertex_id_by_pos(position)
	db.actor:set_vector_global_arg_1(position)
	return level.vertex_id()
end

-- запускает эффектор шатания камеры с заданными параметрами
function level.add_cam_effector3(total_time, amplitude, period_number, power)
	level.set_ce_time(total_time)
	level.set_ce_amplitude(amplitude)
	level.set_ce_period_number(period_number)
	level.set_ce_power(power)
	level.add_ce()
end
```

relation_registry

```
function relation_registry.get_goodwill(who_id, to_whom_id)
	return GetGoodwill(who_id, to_whom_id)
end

function relation_registry.set_goodwill(who_id, to_whom_id, goodwill)
	return db.actor:set_goodwill_ex(who_id, to_whom_id, goodwill)
end

function relation_registry.change_goodwill(who_id, to_whom_id, goodwill_change)
	return db.actor:change_goodwill_ex(who_id, to_whom_id, goodwill_change)
end
```

ray_pick

Флаги трассировки #rq_target

```
-- устанавливает базовые параметры трассировки
function ray_pick.init(pos, dir, range, flags, obj)
	ray_pick.set_position(pos)
	ray_pick.set_direction(dir)
	ray_pick.set_range(range)
	ray_pick.set_flags(flags)
	ray_pick.set_ignore_object(obj)
end

-- устанавливает стартовую точку трассировки
function ray_pick.set_position(pos)
	db.actor:set_vector_global_arg_2(pos)
end

-- устанавливает направление трассировки
function ray_pick.set_direction(dir)
	db.actor:set_vector_global_arg_1(dir)
end

-- устанавливает диапазон трассировки
function ray_pick.set_range(range)
	set_float_args_12(range, 0)
end

-- устанавливает флаги трассировки (rq_target)
```

```
function ray_pick.set_flags(flags)
	set_int_arg1(flags)
end

-- устанавливает игнорируемый игровой объект для трассировки
function ray_pick.set_ignore_object(obj)
	db.actor:set_object_arg_1(obj)
end

-- выполняет запрос на трассировку и возвращает true, если она была успешна
function ray_pick.check()
	return level.perform_ray_pick_query()
end

-- если трассировка была успешна, возвращает дистанцию до точки, иначе возвращает ранее заданный диапазон трассировки
function ray_pick.get_distance()
	return level.get_ray_pick_dist()
end

-- если трассировка была успешна, возвращает игровой объект
function ray_pick.get_object()
	return level.get_ray_pick_obj()
end

-- если луч пересек игровой объект, то возвращает номер кости, иначе возвращает номер полигона статической геометрии (при неудачной трассировке возвращает -1)
function ray_pick.get_element()
	return level.get_ray_pick_element()
end
```

Автор: **RayTwitty (aka Shadows)**

Источник — «https://xray-engine.org/index.php?title=X-Ray_extensions&oldid=1185»

Категория:

Движок
