Game Object - Code Examples

Overview

In this section, we are going to showcase some usefull code examples that you could use for your own scripts. We advise you to understand the code before copying and pasting it. Before beginning, have a look at Game Object - Functions since you will be able to find all available functions for game objects there.

Starting The Journey

Before continuing, note that you should have some idea about callbacks, what they are and how they work, and the way functions work in LUA. Check Core - Callbacks

---

Example 1 - Retrieving the Tank From Your Party

In this case, we already prepared a function that does this functionality for you. It's located in the unit_helper module.

```lua
---@type unit_helper
local unit_helper = require("common/utility/unit_helper")

---@param local_player game_object
---@returns game_object | nil
local get_tank_from_party(local_player)
    local allies_from_party = unit_helper:get_ally_list_around(local_player:get_position(), 40.0, true, true)

    for k, ally in ipairs(allies_from_party) do
        local is_current_ally_tank = unit_helper:is_tank(ally)

        if is_current_ally_tank then
            return ally
        end
    end

    return nil
end
```

NOTE

To retrieve the healer, we can do likewise and use the unit_helper. In this case, just use the function is_healer instead of is_tank

---

Example 2 - Retrieving the Skull-Marked Unit

```lua
---@type enums
local enums = require("common/enums")

---@return game_object | nil
local function get_skull_marked_unit()
    -- first, we check if local player exists
    local local_player = core.object_manager.get_local_player()
    if not local_player then
        return
    end

    -- then, we check all possible units in 40yds radius (for example)
    local search_radius = 40.0

    -- we will use squared distance, since this is much more efficient than the regular distance function,
    -- as it prevents a square root operation from being performed. (The final result is exactly the same one)
    local squared_search_radius = 40.0 * 40.0

    local all_units = core.object_manager.get_all_objects()

    -- important: do not try to get the local player position inside the loop, since this
    -- will drop fps as your CPU would be performing useless extra work.
    local local_player_position = local_player:get_position()
    for k, unit in ipairs(all_units) do
        local unit_position = unit:get_position()
        local squared_distance = unit_position:squared_dist_to_ignore_z(local_player_position)

        if squared_distance <= squared_search_radius then
            local unit_marker_index = unit:get_target_marker_index()
            local is_skull = unit_marker_index == enums.mark_index.SKULL

            if is_skull then
                return unit
            end
        end
    end

    return nil
end
```

TIP

You can check that the code works by using the following lines:

```lua
core.register_on_update_callback(function()
    local skull_marked_npc = get_skull_marked_npc()
    if skull_marked_npc then
        core.log("The Skull-Marked NPC's name is: " .. skull_marked_npc:get_name())
    else
        core.log("No Skull-Marked NPC was found!")
    end
end)
```

You can also retrieve the units marked with other marks by re-using the same code, you would just need to change the enums.mark_index. index. (Basically, you would just need to change line 30)

Example 3 - Retrieving the Future Position of a Unit

```lua
---@param unit game_object
---@param time number
---@return vec3
local function get_future_position(unit, time)
    local unit_current_position = unit:get_position() -- vec3
    local unit_direction = unit:get_direction()       -- vec3
    local unit_speed = unit:get_movement_speed()      -- number

    -- first, we normalize the direction vector to ensure it has a length of 1
    local unit_direction_normalized = unit_direction:normalize()

    -- then, we calculate the displacement: distance = speed * time
    local displacement = unit_direction_normalized * unit_speed * time

    -- finally, we just calculate the future position by adding the displacement to the current position
    local future_position = unit_current_position + displacement

    return future_position
end
```

TIP

To test the previous code, you could use the following lines:

```lua
core.register_on_render_callback(function()
    local local_player = core.object_manager.get_local_player()
    if not local_player then
        return
    end

    -- lets draw a line between current position and our calculated future position
    local local_player_position = local_player:get_position()
    local local_player_future_pos_in_1_sec = get_future_position(local_player, 0.50)

    core.graphics.circle_3d(local_player_future_pos_in_1_sec, 2.5, color.cyan(200), 25.0, 1.5)
    core.graphics.line_3d(local_player_position, local_player_future_pos_in_1_sec, color.cyan(255), 6.0)
end)
```

Example 4 - Set Glowing To Enemies That Are Not On Line Of Sight

```lua
---@type unit_helper
local unit_helper = require("common/utility/unit_helper")

---@type enums
local enums = require("common/enums")

core.register_on_update_callback(function()
    local local_player = core.object_manager.get_local_player()
    if not local_player then
        return
    end

    local local_player_position = local_player:get_position()
    -- we get the enemies around 40 yards from player
    local enemies = unit_helper:get_enemy_list_around(local_player_position, 40.0, true)

    for k, enemy in ipairs(enemies) do
        local enemy_pos = enemy:get_position()
        -- trace line will return true if the enemy is in line of sight, as long as we pass the enums.collision
        local is_in_los = core.graphics.trace_line(local_player_position, enemy_pos, enums.collision_flags.Line
        -- we have to check if the unit is glowing already
        local is_glowing_already = enemy:is_glow()

        if not is_in_los then
            if not is_glowing_already then
                -- make it glow if not glowing yet
                enemy:set_glow(enemy, true)
            end
        else
            if is_glowing_already then
                -- make it not glow if it's in line of sight, if it was glowing before
                enemy:set_glow(enemy, false)
            end
        end
    end
end)
```

Previous
« Game Object - Functions

Next
Buffs »