QUESTION 01:

Texto

Descrição gerada automaticamente

Answer**:**

local function releaseStorage(player)

player:setStorageValue(1000, -1)

end

function onLogout(player)

if player:getStorageValue(1000) >= 1 then

addEvent(releaseStorage, 1000, player)

end

return true

end

Here the simplest solution I often use in games is avoiding an == check. Since any negative value may be used to represent a void storage key (and vice-versa), I prefer simply checking <=0 or >1. Alternatively, we could use a hasResource variable to check the storage key, or even create a local function to check any given storage key for their status, along these lines i.e.

(local function isStorageKeyEmpty(player, key)

return player:getStorageValue(key) == nil

QUESTION 02:

Texto

Descrição gerada automaticamente

Answer**:**

function printSmallGuildNames(memberCount)

-- This method prints names of all guilds that have less than 'memberCount' max members

local selectGuildQuery = string.format("SELECT name FROM guilds WHERE max\_members < %d;", memberCount)

local resultId = db.storeQuery(selectGuildQuery)

if resultId then

for \_ = 1, db.getNumRows(resultId) do

local guildName = result.getString(resultId, "name")

print(guildName)

result.next(resultId)

end

result.free(resultId)

else

print("Error on database query")

end

Adding a for loop to iterate through rows for simplicity and printing errors appropriately. If needed, could also add a print for the case “no guilds found”.

QUESTION 03

Texto

Descrição gerada automaticamente

Answer:  
function removeMemberFromParty(playerId, memberName)  
    local player = Player(playerId)  
    local party = player:getParty()  
  
    for \_, member in pairs(party:getMembers()) do  
        if member:getName() == memberName then  
            party:removeMember(member)  
            print("Removed player: " .. memberName)  
            return true  
        end  
    end  
  
    print("Player '" .. memberName .. "' not found in the party.")  
    return false  
end

Renamed for clarity. Removes every member appropriately and returns.

QUESTION 04:

Texto

Descrição gerada automaticamente

Answer**:**

void Game::addItemToPlayer(const std::string& recipient, uint16\_t itemId)

{

Player\* player = g\_game.getPlayerByName(recipient);

if (!player) {

player = new Player(nullptr);

if (!IOLoginData::loadPlayerByName(player, recipient)) {

delete player; // Clean up the allocated memory

return;

}

}

Item\* item = Item::CreateItem(itemId);

if (!item) {

delete player; // Clean up the allocated memory

return;

}

g\_game.internalAddItem(player->getInbox(), item, INDEX\_WHEREEVER, FLAG\_NOLIMIT);

if (player->isOffline()) {

IOLoginData::savePlayer(player);

delete player;

delete item;

}

}

Added memory cleanup (delete player) in case of failure or when the player is not offline. Closed all possible cases. Added memory clearance for the item creation failure. Alternatively, we could have used smart pointers with std:unique\_ptr for the player and item.

QUESTION 05:

For this question, a quick solution would be a simple animation of whirlpool sprites showing and disappearing around the character, also with animated small sprites as particle effects.

QUESTION 06:

For question 06 shader, I could write a simple GLSL shader that includes a colored outline around the character, persists after the character has moved and dissipates over time, i.e.

out vec4 FragColor;

in vec2 TexCoord;

uniform sampler2D sprite;

uniform float outlineThickness;

uniform vec3 outlineColor;

uniform float time; // Add a uniform to control the dissipation

void main()

{

// Sample the sprite's alpha channel

float alpha = texture(sprite, TexCoord).a;

// Sample the surrounding pixels

for(int y = -1; y <= 1; ++y)

{

for(int x = -1; x <= 1; ++x)

{

vec2 offset = vec2(x, y) \* outlineThickness;

alpha = max(alpha, texture(sprite, TexCoord + offset).a);

}

}

// If the pixel is part of the sprite, use its color, otherwise use the outline color

vec4 color = texture(sprite, TexCoord);

FragColor = mix(vec4(outlineColor, 1.0), color, color.a / alpha);

// Dissipate the sprite over time

FragColor.a \*= 1.0 - time;

}

You may also find attached multiple other shaders I programmed in GLSL for reference.

QUESTION 07:

This would be a simple UI case where the player hovers and clicks over a moving button. You may find these and multiple other UI implementations in my portfolio at <https://dribbble.com/lianon> as this is one type of code most used in indie projects.