Documentation: Gestionnaire UDP pour le projet R-Type

Le gestionnaire UDP est une composante clé du projet R-Type, chargé de faciliter une communication rapide et efficace entre le serveur et les clients. Ce document fournit des détails sur sa conception, y compris la structure des messages, le processus de sérialisation/désérialisation et les considérations liées à la bande passante.

1. Structure des messages UDP

Le gestionnaire UDP utilise un format de message bien défini pour la communication. Chaque message est représenté par la structure suivante :

```
struct Message {
    unsigned int id : 10;    // 10 bits pour 'id' (jusqu'à 1024 IDs uniques)
    unsigned int action : 6; // 6 bits pour 'action' (jusqu'à 64 actions)
    std::string params;      // Paramètres optionnels pour le message
};
```

- `id` :Correspond à l'index de entitée dans l'ecs server. Prend en charge jusqu'à 1024 IDs uniques grâce à ses 10 bits.
- `action` : Spécifie l'action à effectuer, limitée à 64 types grâce à ses 6 bits. (Liée à l'Enum d'évents)
- `params` : Une chaîne de caractères de longueur variable contenant des paramètres supplémentaires pour le message.

2. Sérialisation et Désérialisation

Pour envoyer et recevoir des messages efficacement via UDP, le gestionnaire convertit les messages en un format binaire.

Processus de sérialisation :

- 1. Combine `id` et `action` en un en-tête unique de 16 bits.
- 2. Ajoute la longueur de la chaîne `params` sous forme de valeur 4 octets.
- 3. Ajoute le contenu de `params`.

```cpp
void MessageCompressor::serialize(const Message &msg, std::vector<char> &buffer)
{
    buffer.clear();

    unsigned short header = (msg.id & 0x3FF) | ((msg.action & 0x3F) << 10);
    buffer.push_back(header & 0xFF);
    buffer.push_back((header >> 8) & 0xFF);

    uint32_t paramsSize = msg.params.size();
    buffer.insert(buffer.end(), reinterpret_cast<const char *>(&paramsSize),
                  reinterpret_cast<const char *>(&paramsSize) + sizeof(paramsSize));
    buffer.insert(buffer.end(), msg.params.begin(), msg.params.end());
}
```

Processus de désérialisation :

- 1. Extrait l'en-tête de 16 bits et décode `id` et `action`.
- 2. Extrait la longueur de `params` et récupère son contenu.

```cpp
void MessageCompressor::deserialize(const std::vector<char> &buffer, Message &msg)
{
    size_t offset = 0;

    unsigned short header = buffer[offset] | (buffer[offset + 1] << 8);
    msg.id = header & 0x3FF;
    msg.action = (header >> 10) & 0x3F;
    offset += 2;

    uint32_t paramsSize;
    std::memcpy(&paramsSize, &buffer[offset], sizeof(paramsSize));
    offset += sizeof(paramsSize);
    msg.params = std::string(buffer.begin() + offset, buffer.begin() + offset + paramsSize);
}
```

3. Calcul de la bande passante

La bande passante nécessaire pour la communication UDP dépend du nombre de messages envoyés par seconde et de la taille de chaque message.

Répartition de la taille des messages :

- En-tête : 2 octets (16 bits pour `id` et `action`)
- Longueur de `params` : 4 octets
- Contenu de `params` : Variable (dépend du message)

Exemple :

Pour un message contenant 100 octets dans `params` :
- Taille totale = 2 octets (en-tête) + 4 octets (longueur) + 100 octets (params) = 106 octets

Formule de bande passante :

Bande passante (octets/s) = Taille du message × Messages par seconde

- Taille du message : 106 octets
- Taux de ticks du serveur : 40 TPS (Ticks Par Seconde)
- Messages par tick : 100

Bande passante = 106 × 40 × 100 = 424 000 octets/s = 424 Ko/s