1 Introduction

The goal of the R-Type project is to create an online multiplayer copy of the classic R-Type game (1987).

The key words:

- MUST
- MUST NOT
- REQUIRED
- SHALL
- SHALL NOT
- SHOULD
- RECOMMENDED
- MAY
- OPTIONAL

in this document are to be interpreted as described in RFC 2119.

1. R-Type architecture

The R-Type architecture is a classic client-server game architecture. All the game engine is in the server. A client connects to the server by using the R-Type TCP protocol. When connected, the client has the choice between creating a lobby or joining an existing lobby. Multiple clients can connect to the server at the same time. Then, a lobby's creator client can launch a new game. The server can run several games at the same time.

R-Type UDP Protocol

```
Once the game is launched, the client-server communications are done
by using this R-Type UDP Protocol.

Frames transmission
```

The first purpose of the R-Type UDP Protocol is to send all the frames to display (in the client) from the server to the client.

The server send the sprites informations to the client, then the client manage the srites on it's own.

A SPRITE_LIST is a succession of SPRITE.

A SPRITE is a succession of 7 numbers representing a sprite to display and its position. Each one of these numbers MUST be coded on 4 bytes. Thus, a SPRITE MUST be 28 bytes long, and a SPRITE_LIST of (n) SPRITE MUST be (n * 28) bytes long.

The SPRITE composition MUST be the following:

- 1st number = the index of the sprite's spritesheet
- 2nd number = X coordinate (in pixels) of the sprite's top left corner in its spritesheet
- 3rd number = Y coordinate (in pixels) of the sprite's top left corner in its spritesheet
- 4th number = the sprite's width (in pixels)
- 5th number = the sprite's height (in pixels)
- 6th number = X coordinate (in pixels) of the sprite's top left corner in the game window
- 7th number = Y coordinate (in pixels) of the sprite's top left corner in the game window

The order of the SPRITE elements in SPRITE_LIST MUST be the final order of displaying (in the game, every SPRITE will be displayed over the previous ones).

The client MUST NOT respond to the server when a payload has been

received.

Player events

The player, on the client-side, can execute several actions. On each player action, the client MUST send to the server a specific event payload containing informations about the action. This payload MUST begins with an element (on 4 bytes) from the EVENT enumeration, described below:

```
enum EVENT {
    MOVE,
    SHOOT,
    QUIT
};
```

The MOVE event:

The player wants to move (to the left, right, up or down). The client MUST add to the payload an element (on 4 bytes) of the DIRECTION enumeration (described below) right after the EVENT element. Thus, if the player wants to move, the payload MUST be 8 bytes long.

```
enum DIRECTION {
   LEFT,
   RIGHT,
   UP,
   DOWN
};
```

The SHOOT event:

The player wants to shoot. In this case, the payload MUST be only 4 bytes long.

The QUIT event:

> The player wants to quit the game. The client MUST warn the server with this 4 bytes long payload before quitting. The client MAY exit without any response from the server.