1. **Game description**

We will be doing a high-paced shooter game for this assignment. It will be a 2d game that takes inspiration from “The Binding of Isaac”, however it will allow 2 players to play with each other in a smaller map.

The goal of the player will be to kill another player. At the start of the game each player has 3 hearts. Every time one of the players is hit by another player or another entity, the player that has been hit loses one of the hearts. When the amount of hearts reaches 0, the player dies and the other player wins the game.

At the start of the game both players can only shoot basic projectiles. However, throughout the game, power ups will appear on the map that players can grab. Powerups can range from speeding up the projectiles to bigger projectiles etc.

We will also try to implement very basic enemies (which are not controlled by AI for simplicity) that will be able to shoot players and be killed by the players.

2. **Game / server design**

For the game’s design we will use python libraries (e.g. PyGame) or even maybe a game engine. The players will be able to move with WASD xor arrow keys and shoot with space. Power ups will be picked automatically upon reach. UI positioning will be based on “The binding of Isaac”, however it will be tailored to our version of the game:

As we mentioned the game will be a high-paced shooter game that uses dead-reckoning and UDP, so both the server and the client (game) will be tailored to meet this short description.

3. **Description of the approach**

Upon starting, the server will store the starting set positions of both players. Once the server sees that 2 players have connected, it will start the game. Since the starting locations are

set, they won't have to be sent by the server - the client will know them already and will only wait for the signal that will be sent after both players are connected from the server to start the game. After the signal, the server will keep track of positions of both players, powerups (will appear randomly), other game entities and send data in a set format to both clients that will receive the data. The received data will be interpreted, shown on screen in game for the player.

We will use classes for players and bullets. The properties of these elements, such as player position and health, or bullet speed and damage will change throughout the game. For instance, if a player obtains a power up that increases bullet damage, the bullets that will be shot by him will cause more damage on the other player.

The client will be sending data also in a set format in fixed time intervals that will contain information about players decisions (moving, shooting, etc). The server will receive that data and interpret it in its own way.

We will use dead-reckoning to get the proper positions of the entities and calculate where they should end up next.

Two types of messages will be sent during the course of the game.

1. messages sent by the client
   in this scenario players send a message in a format:
   " <direction_input> <shooting> <current_position>"
   id of the player is their ip address
   direction input - what button player has clicked, so that server knows where the player wants to go
   shooting - boolean variable - is the player shooting or not
   current_position - what is the current position of the player
2. messages sent by the server
<players positions> <bullet positions> <available power ups> <current power ups> <player_status>
   player positions - where are the other players
   bullets positions - where are the bullets
   available power ups - powerup type and position on the map
   current power ups - power ups currently in use by the other players
   player status - information about player's health and status

Since the format of the messages will not change, the game and the server will be able to communicate without problems. Since the messages between the client and server will be sent in short intervals, if one of the messages contains an error or will be dropped, the game will keep going using dead reckoning to predict the player's moves and position.

The logic of the game will be run on the server. It means that clients will only send their input to the server. Server will then accept the input and based on it send back the response containing all the information to the client. So the simulation will take place on the server and the client will only interpret the data from the server and print it on the correct positions on the players screen.

The consensus between players is established before the game starts - when the server detects 2 players. Then it sends a message to the players <start game> and the clients send back <ok>. The server starts the game when it receives a response <ok> from every current player.
During the course of the game new players can join - if that happens the game still runs but the server sends a message <start game> to the new player and waits for the response. When that happens the new player joins the game and his input is added to appropriate fields in the server messages.

The server will be a UDP server, since in this way the transmission of the data will be faster. The 2nd type of message will be used as the acknowledgement message.
<sequence_num ok>
Then if the server receives the message out of order - it will interpret it as incorrect input and won't use this message to calculate new positions of the players. If the message is received in order the above ack message will be sent to the client, which will interpret it. We used a similar acknowledgement system in A4.

Since the server is responsible for the game logic, and the clients don't have access to it, cheating will not be possible - clients only send their current position (used for tracking and errors, however servers players positions are superior), what their direction is and if they are shooting or not. Since the game is designed to be played on different computers, the ip addresses will also be used as the players' id - in this way players can't change other players positions.