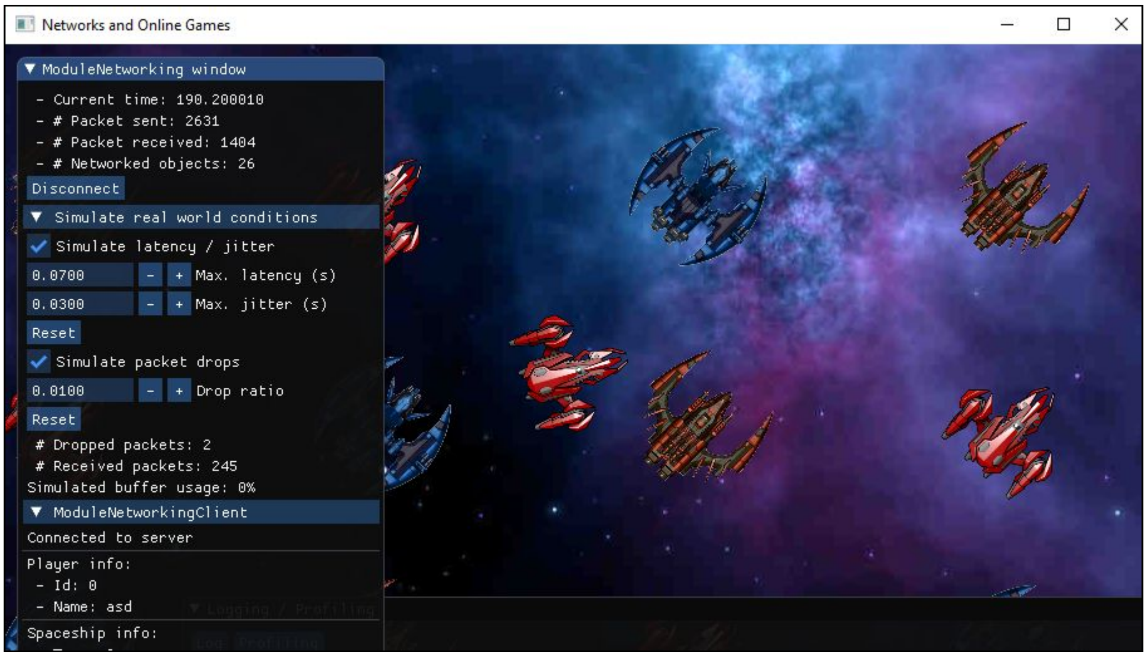
- The submission deadline is **December 31st at 23:59:59**.
- To deliver the exercise, **create a release in GitHub** and upload the **link** to the **release** into the CITM virtual campus: **Multiplayer game C++**.
- In the **ZIP file**, **include a folder with the solution** and the source code.
- **If the release cannot be properly executed, the exercise won't be accepted.**
- **Do not disable the UI to simulate real world conditions.** Will be key for the teacher to test the effectiveness of the implemented techniques.



```
Networks and Online Games
▼ ModuleNetworking window
 - Current time: 190.200010
 - # Packet sent: 2631
 - # Packet received: 1404
 - # Networked objects: 26
Disconnect
▼ Simulate real world conditions
✓ Simulate latency / jitter
0.0700     - + Max. latency (s)
0.0300     - + Max. jitter (s)
Reset
✓ Simulate packet drops
0.0100     - + Drop ratio
Reset
 # Dropped packets: 2
 # Received packets: 245
Simulated buffer usage: 0%
▼ ModuleNetworkingClient
Connected to server
Player info:
 - Id: 0
 - Name: asd
Spaceship info:
```

Do not remove the options to *"Simulate latency / jitter"* and *"Simulate packet drops"* from your code. The teacher will use these options to test the level of reliability accomplished by your implemented techniques.

Statement

Implement a simple 2D game prototype using the skeleton of the engine provided at class. Something similar to the initial spaceship wars seen in the example will do, but a bit extended (i.e. introduce some online game mechanics). Other examples if you decide to change the genre could be games such as Pong, collaborative space invaders, 2D racing games, 2D RPG-like with overhead camera, etc.

Even if a polished result and a good idea for a multiplayer game will be taken into account, there is more **interest in developing the techniques explained during the lab sessions** in order to mitigate the impact of the network issues (to **handle latency, jitter, and packet loss**).

Application requirements

At minimum, your multiplayer game should:

- Accept a certain number of players.
- Handle players join/leave events.
    - Depending on the type of game, this needs to be handled carefully (what happens if a player disconnects in a two-player tennis match?).
- Implement the world state replication so that clients receive the state of the world.

To opt for a higher mark, you will also have to implement the techniques explained during the lab sessions:

- Redundant sending of input packets to improve on packet loss ratios.
- Delivery manager that notifies about successful deliveries, and about delivery failures on timeout.
    - Integrate it with the delivery of replication packets. The server needs to be sure about clients receiving the current world state. If a delivery fails, the server will resend **the current state** of the replication commands in the failed delivery.
- Client side prediction with server reconciliation, to avoid laggy user input.
- Entity interpolation, to make network objects update smoothly on screen.

Documentation requirements

The documentation will be presented in a markdown or text file (**README.md** or **README.txt**) provided in the root folder of the release. It will contain the following information:

- Names of the group members
- Game name and description
- Gameplay tutorial / instructions
- List of implemented features/techniques and authors of each one
    - It is expected that everybody implements some of the techniques explained in the lab sessions (i.e. everybody has to code).
    - For each feature in the list, describe the degree of completeness reached: tried but not achieved / achieved with some known bugs / completely achieved.
    - For each feature in the list explain also the result accomplished and the known bugs or issues found. Flawless implementations that execute with no issues and solve the problem addressed perfectly will have short explanations. Otherwise, you are meant to be verbose on the identified bugs and the reason why you think they happen.
    - **Be honest!** Your code and released binaries are there at GitHub.