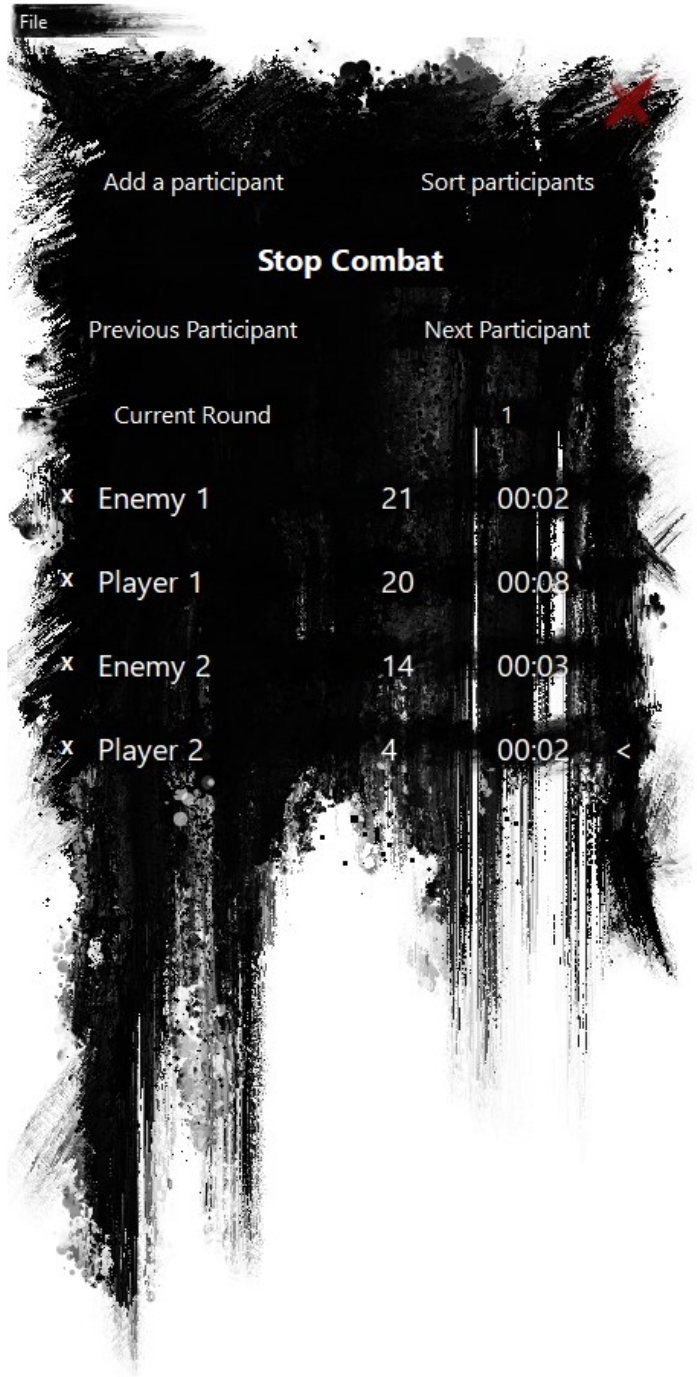
Combat tracker

By Riku Ruusulaakso

What is a combat tracker?

- It is a software used in turn based games, like Baldur's Gate 3 or Dungeon's and Dragons.
- It lists individual characters in combat and tracks whose turn it is currently.
- On top of that my version even features a stopwatch which shows how long each participant has spent time in their turn.



How is this related to secure programming?

- It does not contain cryptography, but it manages memory.
- Each participant is dynamically created and the input fields might be used with malicious intent. Most notably for bufferoverflow.

```
// Connect user input fields to update Participant object
connect(nameEdit, &QLineEdit::textChanged, [participant](const QString& text)
    {
        participant->name = text;
    });

connect(iniEdit, &QLineEdit::textChanged, [participant](const QString& text)
    {
        participant->ini = text.toInt();
    });
```

- I chose to approach the subject in ”what if” –mentality, meaning that even though the data used by the application is not sensitive and doesn’t need securing, it could be, in a different GUI application.

Turns out Qt provides security tools

- I had originally thought this would have been more difficult, but turns out Qt has a library of tools which helps improve security.
- Not only can I use functions like setMaxLength and setValidator, QLineEdit itself uses Qstring which in turn is programmed to protect from bufferoverflows.

```
// Participant's name
QLineEdit* nameEdit = new QLineEdit(rowWidget);
if (participant->name == "Enter name") { nameEdit->setPlaceholderTex
else { nameEdit->setText(participant->name); }
nameEdit->setPlaceholderText(participant->name);
nameEdit->setMaxLength(20);

// Participant's initiative score
QLineEdit* iniEdit = new QLineEdit(rowWidget);
if (participant->ini == 0) { iniEdit->setPlaceholderText(QString::nu
else { iniEdit->setText(QString::number(participant->ini)); }
iniEdit->setValidator(new QIntValidator(0, 100, rowWidget));
```

Some ideas for the future

- As the memory management and user input were my primary goals, having them solved so easily requires some thinking of additional features.
- Encryption could be added to the feature of saving users in the JSON file.
- Checking that the JSON file contains correctly formatted information and is bleached before use.
- The program could be turned in to a webapp, this would open a new set of potential vulnerabilities.

AI usage

- I used ChatGPT 4o for creating the code of the program and to check my writing in the report.
- I estimate that over 70% of the code was made by ChatGPT with minor manual editing.