Praktikumsaufgaben für VW8

Programmierung interaktiver Systeme im Wintersemester 2023/24
Prof. Dr. Martin Weigel

In dieser Aufgabe erweitern wir das aus P03 bekannte PokePong Spiel um eine Netzwerkmodus. Nutzen Sie für diese Aufgabe ihre MVC-Version des PokePong Spiels. Falls Sie keine funktionierende MVC-Version besitzen können Sie diese Aufgabe aber auch mit der Nicht-MVC Version aus dem Moodle umsetzen.

1 `TCPokePong`

Jeder Spieler soll eine eigene Anwendung starten: ein Spieler als Server und einer als Client. Der Spieler auf der linken Seite (*Pikachu*) spielt in der Server-Anwendung und der Spieler auf der rechten Seite (*Evoli*) in der Client Anwendung. Der Spieler darf nur sein eigenes Pokemon, bzw. seinen Schläger, bewegen dürfen (z.B. mit den Pfeiltasten).

Das Verwalten von mehreren Spielezuständen kann schnell unübersichtlich werden und zu Synchronisationsfehler führen. Daher wird folgende Implementierung bei Client-Server-Anwendungen empfohlen: Der Server besitzt und verwaltet den kompletten Zustand des Spieles (`PongData`) und sendet diesen bei Änderungen an den Client. Der Client zeichnet immer die als letztes empfangenen Spieledaten. Die Tastatureingaben vom Client werden direkt an den Server weitergesendet und im Server verarbeitet. Dadurch vergrößern sich zwar die Latenzzeit im Client, aber es wird immer der gleiche Spielezustand im Client und Server verwendet.

Sie können zum Testen beide Anwendungen auf dem gleichen Rechner starten und für die Eingabe zwischen den Fenstern wechseln.

1.1 Flexible Client-Server-Anwendung

Backup Empfehlung: Machen Sie sich eine Kopie des Quellcodes, bevor Sie ihn für diese Aufgabe verändern.

Schreiben Sie das `TCPong` so um, dass die Reihenfolge des Startens keine Rolle mehr spielt.

Schreiben Sie dazu zwei Main-Methoden, welche sich aber nur in der übergebenen IP des Partners und Port unterscheiden. Zum Testen auf Ihrem System können Sie beiden Konstruktoren `localhost` als IP und den gleichen Port (z.B. 8080) übergeben. Das Programm soll automatisch erkennen, ob bereits ein Server unter der gegebenen IP und dem Port gestartet wurde. Falls ja, soll sich das Programm als Client mit dem Server verbinden. Falls der Server noch nicht gestartet wurde soll das Programm selbst einen Socket auf dem gegebenen Port öffnen.

Das Bedeutet die erste Anwendung wird automatisch ein Server. Die als zweites gestartete Anwendung automatisch ein Client. Sie können `socket.connect(address, timeout);` verwenden, um zu prüfen ob eine Verbindung zu einem Server möglich ist. Falls eine Verbindung nicht möglich ist, wirft diese Methode eine Exception auf die Sie reagieren müssen.

1.2 Fehlertolerante Sockets

Versuchen Sie das Programm möglichst fehlertolerant zu schreiben: Wenn eine der beiden Anwendungen beendet wird soll das Partnerprogramm automatisch versuchen den Verbindungsprozess erneut zu starten. Dabei kann es auch vorkommen, dass ein existierender Client zu einem Server wird.