Relatório - Terceiro Trabalho INF2545

Aluno André Mazal Krauss

Professora Noemi Rodriguez

INF2545 – Sistemas Distribuídos, PUC-Rio 2019.1

O jogo desenvolvido

O jogo desenvolvido é um lobby virtual simples. Ou seja, um espaço virtual em que jogadores em diferentes máquinas podem entrar, se movimentar, mandar mensagens e interagir com elementos espaciais simples. Espaços como este são comuns em MMOs (*jogos online massivamente multi-jogador*), e a minha implementação se baseia na arena de Hockey do extinto MMO *Club Penguin.* Porém, minha implementação dispensa várias complexidades desse tipo de jogo como customização de personagens, inventários e trocas, e além disso, evidentemente, não é robusto para um número tão grande de jogadores.

Nesse lobby simples implementado, os jogadores podem se movimentar livremente usando o clique do mouse, digitar mensagens usando o teclado e enviá-las publicamente aos demais jogadores, e interagir com um disco de hockey, que se movimenta pela tela quando um jogador passa sobre ele. No espírito do *Club Penguin*, implementei também um mecanismo de censura para as mensagens mandadas, que procura bloquear o envio de palavras ofensivas.

Controles:

- Botão esquerdo do mouse – movimentação do personagem
- Teclado – digitação de mensagens
- Enter – enviar mensagem

Estrutura e Comunicação

Como proposto pelo enunciado, toda a comunicação entre máquinas é realizada através de filas de inscrição, utilizando o protocolo MQTT e o servidor Mosquitto. Toda ação do jogador não tem efeito imediato em sua próprio máquina, mas sim publica uma alteração na fila. Somente quando o estado local for notificado da publicação é que esta terá um impacto visual na tela. A ideia dessa separação era permitir que todos as instâncias de jogo pudessem reagir aos eventos seguindo a mesma ordem, porém o resultado disso é uma jogabilidade não responsiva em vários momentos. Minha conclusão é que apesar de conceitualmente essa separação fazer sentido, na prática, com a jogabilidade que propus, não haveria ônus em fazer com que o estado local respondesse imediatamente às mudanças.

Para a implementação, utilizo, ao todo, três filas. Cada uma carrega informações que são indenpendentes entre si e cuja ordenação não é importante uma à outra. Para todas as filas, utilizo um mecanismo de serialização/deserialização similar ao que utilizamos no último trabalho, utilizando binser e mime. As filas são:

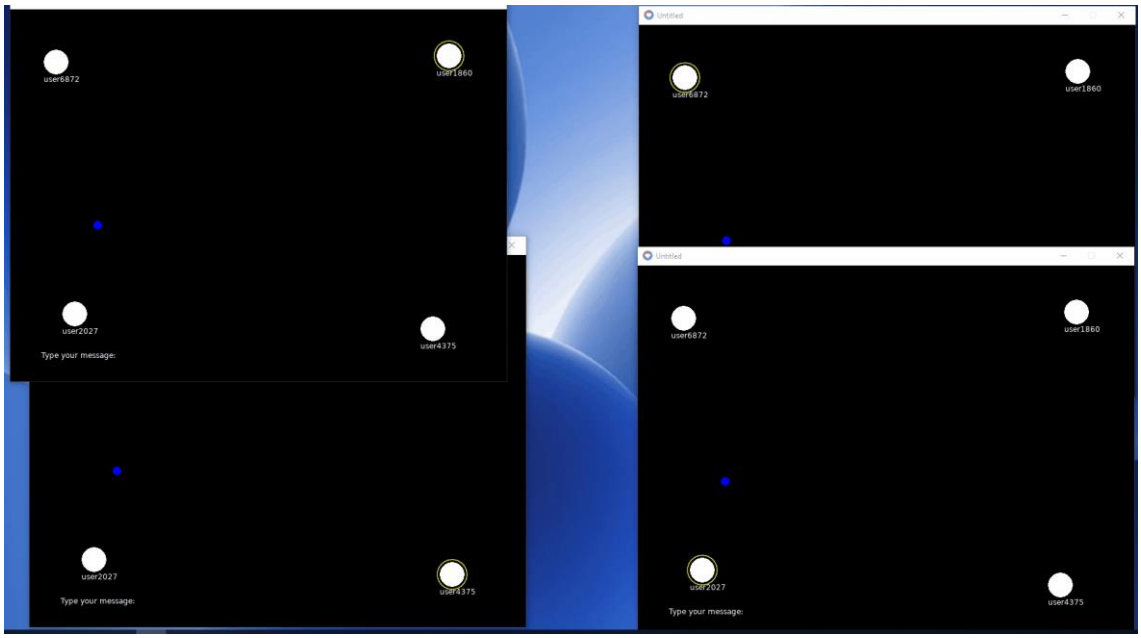
- Players_ping: carrega consigo todas as informações básicas sobre um jogador. Ela é utilizada para registrar o que um jogador se conectou pela primeira vez. Ao mesmo tempo, a inatividade nesta fila por parte de um jogador é utilizada para detectar a

desconexão dele. Vale notar que ela também comunica e implementa a movimentação dos jogadores, mas não através de coordenadas absolutas. É somente comunicada a *coordenada desejada* de um determinado jogador, e cada instância local de jogo fica responsável por gradativamente deslocá-lo até a posição desejada. Assim, todo a movimentação é calculada localmente, o que permite uma movimentação mais fluida do que seria possível se cada *step* dependesse de uma comunicação através do Mosquitto.

- Textbox: usada para a transmissão das mensagens de texto. Cada mensagem é transmitida como uma simples string, que acompanha o número identificador do jogador. Assim, cada mensagem pode ser atribuída visualmente ao jogador que a escreveu. A censura é implementada localmente, de maneira que mensagens com palavras impróprias sequer são adicionadas à fila.
- Puck_info: usada somente para a passagem da informação da localização do disco de hockey. Assim como na movimentação dos players, é informada somente a posição desejada, e a trajetória até lá é computada localmente. Vale também notar que a detecção do contato entre um jogador e o disco é calculada somente para o player local, em cada instância de jogo.

Testes de Escalabilidade

O jogo rodou, com alguma lentidão, em quatro instâncias simultâneas no meu computador:



Para além disso, realizei os seguintes testes:

1. Procurei detectar algum erro no ordenamento das mensagens (script teste1.lua). Porém, para até 10000 publicações em sequência, não detectei erros dessa maneira. Vale notar que não realizei testes a partir de diferentes máquinas em redes diferentes.
2. Em diversos momentos, tentava sem sucesso utilizar o servidor Mosquitto. Por diversas vezes não consegui conectar-me com sucesso, e por isso optei por utilizar outro servidor nos testes e durante o desenvolvimento, o iot.eclipse.org.

3. Escrevi um script (teste2.lua) para tentar medir o *ping médio* do lua_mqtt com o servidor iot.eclipse.org e o Mosquitto. Considero o *ping* como o tempo decorrido entre a publicação de uma mensagem e o recebimento da notificação correspondente, na mesma máquina. Para isso, considerei o envio de uma string de 1K, assim como foi feito no primeiro trabalho. Nessa situação, obtive uma média de 0.275 para o iot.eclipse.org e, não consegui realizar o teste para o mosquitto.

De qualquer maneira, essas medições acabaram por ser muito dependentes da rede local, e foram realizadas em uma mesma máquina em uma mesma rede. Assim, não são testes adequados para avaliar a viabilidade de uso do MQTT para jogos de maior porte.