Imagem e Ação

Introdução

Para o projeto do jogo Imagem e Ação, foi escolhida a linguagem C++ com a biblioteca Boost.

O projeto foi inteiramente desenvolvido com o compilador Clang, utilizando o padrão C++98. Uma tentativa de utilizar C++11 foi feita, mas devido a bugs no compilador, a mesma foi abandonada.

Para a comunicação cliente-servidor, foi utilizada a ASIO da Boost, e para threads, a biblioteca de Threads da Boost.

Cliente-Servidor

Os clientes se comunicam com o servidor (e vice-versa) através de sockets criados pela boost::asio. É uma biblioteca de comunicação 100% assíncrona, utilizando buffers e até mesmo threads implícitas para gerenciar as conexões.

É possível rodar múltiplos servidores com o mesmo executável, bastando passar uma porta diferente para cada servidor. Cada servidor roda em uma thread separada.

Cada servidor possui vários times. O limite de times é definido pelo tamanho de um long int na máquina do servidor, o que o torna virtualmente ilimitado.

Por questões de performance, ao iniciar-se um servidor, o mesmo cria 10 times para si. Os jogadores sempre entram no time 0 ao conectar-se.

Cada time pode ter, também, um número virtualmente ilimitado de jogadores.

Jogo

A lógica de jogo está na classe server, que também é responsável por iniciar um servidor e aceitar conexões.

A classe deveria criar uma thread para cada time, e através de um mutex compartilhado com uma sessão (que representa um jogador conectado ao servidor) controlar o jogo.

Dificuldades

Inicialmente, seria utilizada a biblioteca de threads do C++11, a nova versão da linguagem. Infelizmente havia um bug no clang que deixava de sintetizar o construtor de cópia para uma classe que fosse passada por referência para uma thread. Caso o construtor de cópia fosse manualmente definido, o clang era incapaz de escolher entre ele e outra alternativa sobrecarregada do mesmo em algumas instâncias, e falhava novamente a compilação.

Foi optado, então, por utilizar a biblioteca de threads da boost. Infelizmente, um bug parecido está presente nela (já que ela foi a base para a biblioteca de threads do C++11 e o compilador é o mesmo). Nesta versão do problema, o programa compila, mas falha em passar um objeto como referência para a thread, tentando passar uma cópia do mesmo para ela, fazendo com que a thread acesse lixo. Não foi encontrado um workaround para este bug, portanto o trabalho ficou incompleto.

O que funciona

Do jeito que o trabalho se encontra, temos um sistema de chat com múltiplas salas em múltiplos servidores. Cada cliente pode conectar-se a apenas uma sala, mas é possível rodar mais de uma instância do cliente no mesmo computador, podendo este conectar-se a mais de um servidor ou a mais de uma sala no mesmo servidor. Toda a comunicação entre eles é assíncrona, e toda a espera de mensagem dispara uma thread nova, feita implicitamente pela boost.

Como testar

Para compilar o programa, basta executar `make`. Caso o compilador clang não esteja disponível no sistema, deve-se editar o makefile e trocar o valor da variável $(CC) por outro compilador (como o g++). O nome das bibliotecas da boost pode mudar ligeiramente. Em algumas distribuições Linux elas podem se chamar libboost_system_mt_s, oposto a libboost_system_mt.

Após compilado o projeto, no diretório bin haverá 3 executáveis: client, server e managequestions.

É importante executar os binários a partir do diretório bin.

O executável managequestions adiciona perguntas (e suas respectivas respostas) ao banco de dados de perguntas e respostas do servidor. Um banco exemplo é copiado para o diretório ao compilar o programa.

O executável client espera dois parâmetros de linha de comando: o IP do servidor e a porta do mesmo.

Uma vez que o cliente esteja conectado ao servidor, qualquer string digitada nele será entregue a todos os demais clientes conectados no mesmo servidor e no mesmo time, incluindo o cliente que enviou a mensagem.

Para trocar de time, deve-se enviar uma mensagem no formato "/t <número>", onde número é a ID do time. Caso não haja um time com esta ID no servidor, um time será criado.

O executável server espera pelo menos um parâmetro: a porta em que ele ouvirá. Para iniciar mais de um servidor com o mesmo executável, basta passar portas adicionais como argumento.