Faculdade de Engenharia da Universidade do Porto

2012/2013 – 2º Semestre

Bombermen

LPOO – 2º Trabalho Prático (Projeto Integrado) – Entrega Final

Duarte Duarte – ei11101@fe.up.pt

João Almeida – ei10099@fe.up.pt

Miguel Marques – ei11099@fe.up.pt

Introdução

Para o Projeto Integrado da unidade curricular Laboratório de Programação Orientada por Objetos, decidimos fazer uma adaptação do jogo clássico Bomberman, lançado originalmente em 1983. Hoje em dia existem dezenas de sequelas, "spinoffs" e adaptações.

Super Bomberman – SNES (1993)

Neste jogo, o jogador controla um Bomberman, que se pode deslocar por um mapa gerado de forma semi-aleatória: os mapas consistem numa grelha de blocos quadrados, que podem ser paredes destrutíveis, indestrutíveis ou simplesmente espaços vazios. As paredes destrutíveis têm uma posição fixa, e as restantes são colocadas aleatoriamente.

O jogador pode colocar bombas, que explodem após um determinado tempo. As explosões ocorrem numa cruz com o centro no local onde a bomba foi colocada, e têm um raio predefinido. As explosões atingem: paredes, destruindo-nas se não forem paredes não destrutíveis; outras bombas, fazendo com que estas expludam (reacção em cadeia); jogadores e powerups. Quando uma parede é destruída, há uma certa probabilidade de aparecer um powerup (bonus) no sítio onde estava a parede.

Existe um servidor com uma interface simples, que permite criar um jogo novo especificando o porto (network port) e o mapa pretendido.

O jogo é jogado por jogadores humanos, em rede. O objectivo é derrotar os restantes jogadores, fazendo com que a explosão da bomba chegue até eles, e ganhar o maior número de pontos. O jogador ganha pontos por destruir paredes, apanhar powerups e fazer explodir outros jogadores; e perde pontos ao ser apanhado em explosões (provocadas por outros jogadores ou até ele próprio).

Manual de Utilização

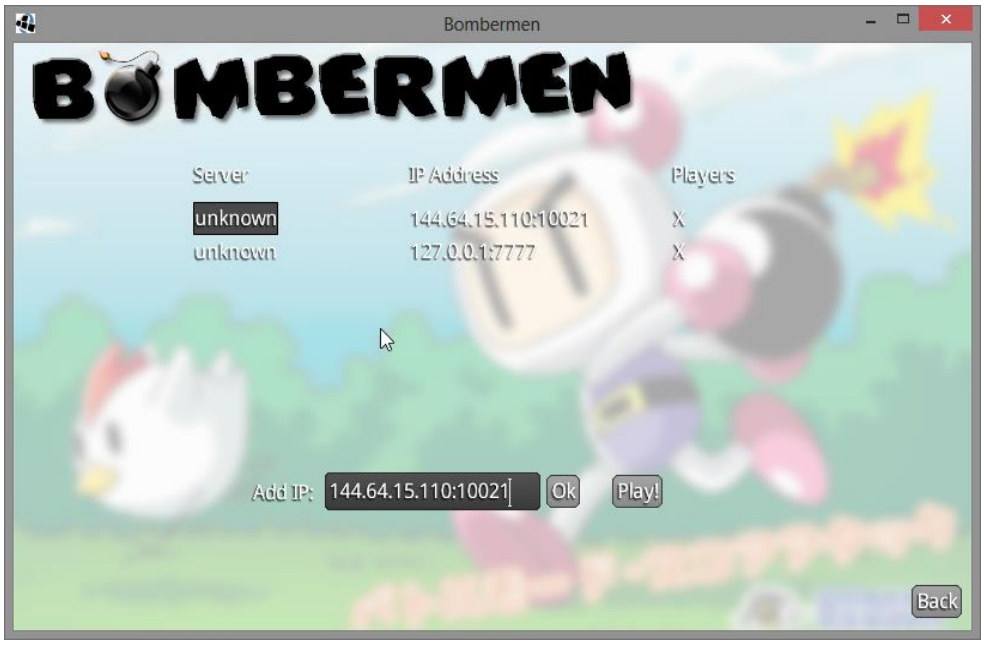
Cliente - Menu Principal

- Botão “Play” - aceder ao menu de seleção de servidor
- Botão “Create” Server - abrir uma nova janela, que permite criar um servidor de jogo
- Botão “Settings” - aceder ao menu de configuração
- Botão “Exit” - sair do jogo

Cliente - Menu de Configuração

- Slider “Sound Volume” - alterar volume dos efeitos sonoros: explosões, apanhar powerups, morrer, etc.
- Slider “Music Volume” - alterar volume da música de fundo
- Checkbox “Fullscreen” - colocar a janela em tamanho máximo
- ComboBox “Keys” - configurar as teclas de movimento
- Botão “Back” - voltar ao menu principal

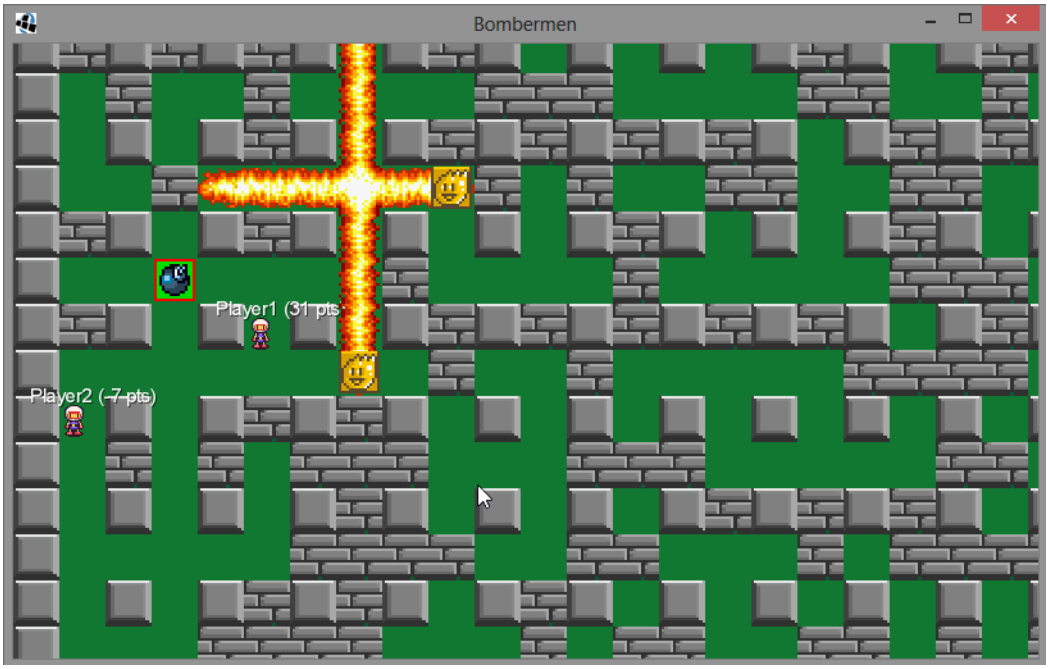
Cliente - Menu de Seleção de Servidor



- Listbox “Server / IP Address / Players” - Lista dos últimos IPs introduzidos, permite selecionar o servidor onde se pretende jogar.
- TextField “Add IP” - Introduzir um novo endereço de IP à lista.
- Botão “Play!” - Entrar no servidor selecionado
- Botão “Back” - Voltar ao menu principal

Depois de se clicar em “Play!”, aparece uma janela de diálogo para introduzir o nome do jogador. Mostram-se mensagens de erro apropriadas no caso de o IP ou o nome introduzidos não serem válidos, ou do servidor estar desligado.

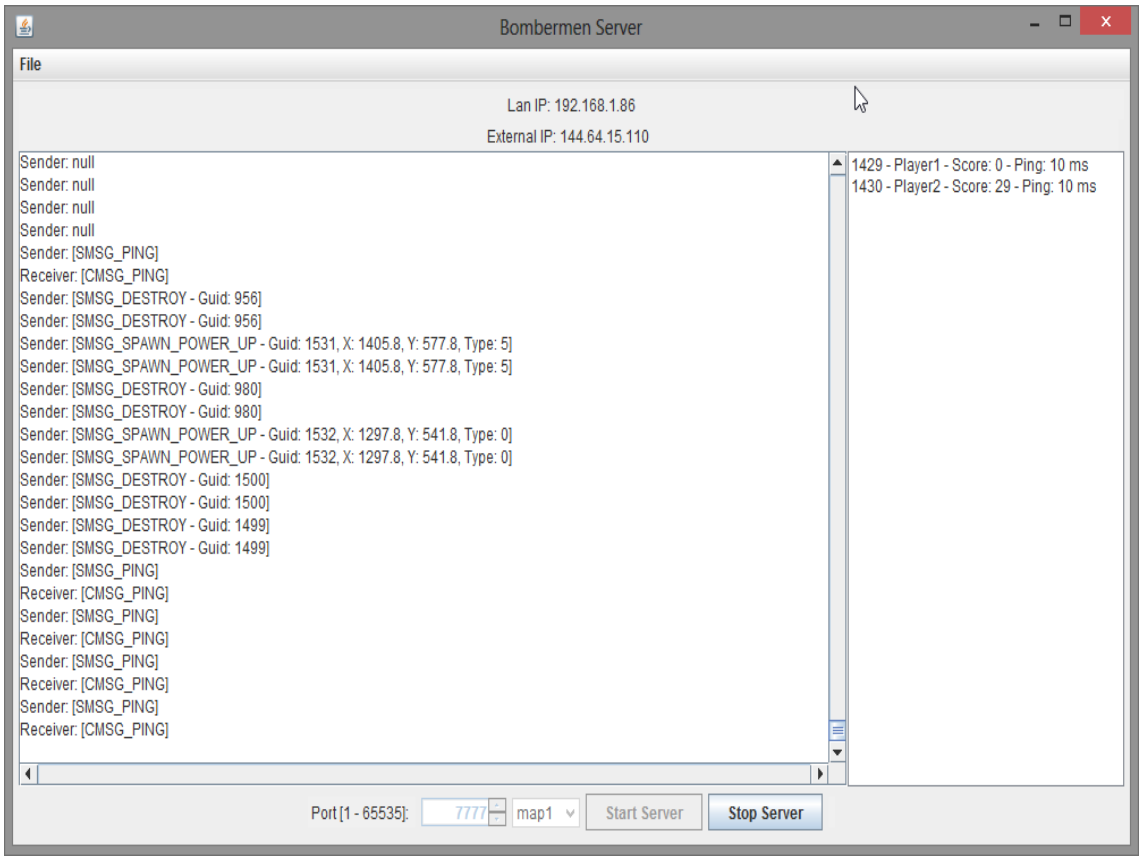
Cliente – Ecrã de jogo



É onde toda acção acontece. O jogador pode mover o Bomberman usando as teclas especificadas nas opções (W - cima, A - esquerda, S - baixo e D - direita, por defeito) e barra de espaços para colocar uma bomba.

A imagem mostra dois jogadores com diferentes nomes e pontuações, vários powerups e uma bomba a explodir.

Servidor

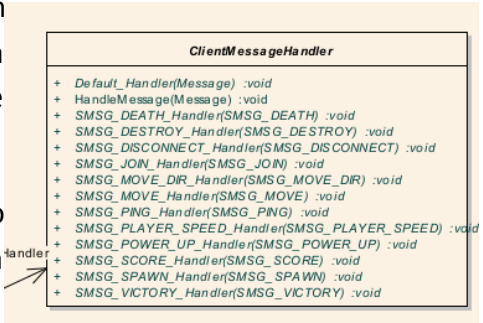


Em cima, é mostrado tanto o endereço de IP interno (intranet) como o externo (internet). No lado esquerdo da janela, é apresentado um log de todas as mensagens e acções transmitidas entre o servidor e os clientes. No lado direito, é apresentado uma lista dos jogadores, contendo o respetivo GUID (globally unique identifier) interno, nome, pontuação e latência (Round trip delay time) em milisegundos.

Conceção do Projeto

O projeto está organizado em dois programas: um servidor, responsável por toda a simulação lógica do jogo, e um cliente, reponsável por desenhar as entidades geradas pelo servidor e por transmitir ao servidor o *input* do jogador. Desta forma, o jogo torna-se mais seguro, no que toca a eventuais tentativas de *hacks* de jogadores.

A comunicação cliente-servidor e servidor-cliente foi implementada recorrendo a um sistema de mensagens enviadas através de sockets. Essas mensagens são todas subclasses de uma classe abstrata *Message* que define um mecanismo de RTTI (Run-time type information) para podermos, de forma rápida, apurar o tipo de mensagem que foi recebida e processá-la recorrendo a um *MessageHandler*.



O *MessageHandler* foi implementado recorrendo a um padrão *strategy* que tem um método que, com base no tipo da mensagem recebida, chama o método respetivo.

À semelhança do *MessageHandler*, o *CollisionHandler* também foi definido com base no padrão *Strategy*.

Numa versão inicial do projeto, concebeu-se uma solução que usava RMI (Remote Method Invocation) para a comunicação entre o servidor e os clientes, porém, esta solução foi abandonada devido a problemas de configuração do próprio RMI e ao delay que o este introduzia na comunicação, que não era adequado a um jogo *multiplayer* em tempo real.A solução final do projeto usa java.net.Socket, abstraída através do sistema de mensagens que implementámos.

GUI do servidor

Para a implementação da GUI do servidor foi usado a interface ClientListener, que utiliza o padrão Observer. Este listener é passado à classe BombermenServer, que é responsável pela lógica do servidor. Sempre que existe uma alteração nos clientes ligados ao servidor, nomeadamente quando são adicionados, removidos ou atualizados, é invocado um dos métodos da interface que é utilizada para atualizar a lista de clientes na GUI do servidor.

Input do jogador

Para o input do jogador, tambem se utilizou o padrão Observer. Este foi usado recorrendo à interface Command definida dentro da classe Input que é utilizada para que esta possa informar o cliente de que o jogador carregou numa tecla mapeada a uma determinada ação, nomeadamente mover o jogador ou colocar uma bomba. A classe de jogo, uma vez informada de uma ação transmite essa informação ao servidor.

Desenho do jogo

No cliente são usadas as classes Actor e Group que utilizam o padrão Composite com o intuito de armezenar e organizar as entidades a desenhar.

Potenciais melhorias ao projeto

- Atenuar a latência que existe entre o servidor e os clientes em relação ao movimento do jogador (foi feito bastante trabalho nesta área, apesar o resultado final ainda não ser o ideal);
- Implementar todos os powerups usados;
- Adicionar inimigos controlados pelo computador ao mapa de jogo;
- Adicionar outros modos de jogo (o modo de jogo atual não tem fim e é do estilo *deathmatch*, assim que um jogador morre, aparece noutro local do mapa);
- Usar sprites adicionais para ser possível distinguir os jogadores visualmente;
- Adicionar uma consola de comandos ao servidor que permita expulsar um jogador, mudar o nome e efetuar outras acções de administração.

Referências

- Artigo da Wikipedia sobre o jogo - http://en.wikipedia.org/wiki/Bomberman
- Case Study: Bomberman Mechanics in an Entity-Component-System - https://bitly.com/151nfuT (referencia inicial, não implementado; optou-se por uma organização mais clássica, com hierarquia de entidades)
- Sprites - http://www.spriters-resource.com/snes/sbomber5
- LibGDX - biblioteca usada no cliente - http://libgdx.badlogicgames.com