

**Manual Técnico**

**Projecto Final**

****

Programação Concorrente e Distribuída

**2011/2012**

Trabalho realizado por:

Miguel Oliveira Nº 38427

João Barreto Nº 38443

**Introdução**

***“The function of good software is to make the complex appear to be simple.”***

Com base neste comentário de Grady Booch e numa metodologia *TOP-DOWN*, iniciámos o desenvolvimento deste projecto, tentando efectuar uma separação de código de forma a que este fosse legível, funcional e de fácil compreensão.

Neste projecto pretendia-se desenvolver um jogo de tipo **MOBA**[[1]](#footnote-1) onde vários jogadores em computadores distintos podiam competir. O objectivo do jogo consiste em eliminar o rei de cada um dos adversários. O jogador vencedor é aquele que conseguir manter o seu rei vivo quando todos os outros jogadores tiverem perdido o seu rei (*King*).

**Desenvolvimento**

**Cliente**

O cliente contém três camadas distintas:

* **Camada lógica:** contém todo o código de envio de mensagens para o servidor. Este envio de mensagens é extremamente importante na relação cliente-servidor. A comunicação é alcançada através de *Sockets* e de *Object Streams* do *Java*.

Em relação à comunicação **entre** jogadores, foi criado um sistema de *chat* onde foi necessário a criação de uma *thread* dedicada ao envio e recebimento de mensagens de texto.

Foi utilizado um recurso partilhado na forma de caixa de mensagens (*MessageBox.java),* que gere todas as comunicações entre cliente-servidor com o auxílio de duas *threads*. um *sender* e um *receiver*.

Sempre que é colocada uma mensagem na lista (*MessageBox)*, é feito um *notify* para que o *sender* a possa recolher e enviá-la ao servidor.

Quando é inicializado um jogo, o cliente envia uma mensagem para o servidor solicitando toda a informação necessária para a construção do ambiente gráfico.

* **Camada gráfica:** contém todo o código referente à interface gráfica (não contém qualquer tipo de concorrência).

Nesta camada são geridos os *inputs* do jogo. Estes *inputs* podem ser do tipo gráfico: selecção e movimento de *characters*.

Esta camada é construída quando o servidor envia ao jogador os dados do respectivo jogo criado ou seleccionado. Foi usada uma *JLayeredPane* para gerir as diferentes camadas gráficas, *world*, *characters* e *HUD*[[2]](#footnote-2).

* **Camada sonora**:contém todo o código necessário para o processamento de áudio na forma de músicas de fundo, voz de *characters* e *chat.*

Esta camada é construída com recurso a *threads* que permitem que vários sons sejam processados pela aplicação e ouvidos pelo utilizador.

**Servidor**

No servidor existe um conjunto de situações onde nos deparamos facilmente com problemas de sincronização e concorrência.

|  |  |
| --- | --- |
| ***World & Characters*** |  |
| * A camada *World* contém toda a informação referente às posições que os *characters* podem ou não sobrepor (*Tiles* normais e *Tiles* bloqueadoras). * A camada *Characters* contém todas as *threads* relacionadas com movimento e com os combates. |

Numa primeira fase são identificados dois recursos partilhados.

Para que um *character* se mova, necessita consultar o *World* de forma a analisar o espaço a percorrer. Essas consultas são efectuadas através de métodos sincronizados uma vez que podem existir diversas *threads* a concorrer directamente ao mesmo espaço. Foi utilizado também um semáforo nas *threads* que representa os *characters,* com intuito de experimentar uma solução alternativa na metodologia concorrente.

No que toca ao motor de combate, este é apenas chamado quando um *character* encontra no seu *range* um inimigo.

A análise do *range* e eliminação de *characters* são métodos sincronizados uma vez que é necessário alertar as várias *threads* em *wait* para ataques iminentes ou eliminações.

A comunicação cliente-servidor é feita através de um recurso partilhado sincronizado denominado de *Broadcaster* uma vez que estecontém todos os métodos necessários à comunicação do jogo.

1. MOBA - Multiplayer Online Battle Arena [↑](#footnote-ref-1)
2. **HUD** (**heads-up display**) método pela qual a informação é visualmente passada ao jogador. [↑](#footnote-ref-2)