U.PORTO

FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Laboratório de Computadores
2 de janeiro de 2017

Aturi Brickiet

2016/2017 – 1º Semestre
MIEIC
Trabalho realizado por:

Grupo T1G13

Afonso Jorge Ramos up201506239@fe.up.pt
Miguel Mano Fernandes up201503538@fe.up.pt

Índice

Breve Descrição do Projeto

No âmbito da disciplina de Laboratório de Computadores, foi-nos proposto implementar em C uma aplicação que utilize o sistema operativo Minix como base de execução.

Sendo que o Minix se assemelha a uma consola decidimos optar por desenvolver um jogo que se encaixa nos "clássicos". Baseado no sucesso mundial da Atari, Breakout, criámos o Aturi Brickiet.

Tentámos ao máximo manter o *feel* tradicional do jogo, mantendo o aspeto visual original, mas adicionando algumas funcionalidades e efeitos visuais para deixar um toque genuíno num jogo que já foi replicado inúmeras vezes a nível mundial.

Instruções de Utilização

Menu Principal

Figura 1 - Menu Principal

Este é o menu principal, no qual está presente o logotipo do nosso jogo, logotipo este feito manualmente e, propositadamente, similar ao original da Atari (figura 2), e com o nome do jogo, brickiet, com um *art style* com um tom mais espirituoso, que tenta mostrar o que o nosso jogo tenta ser face ao original e ligeiramente mais aborrecido Breakout.

Neste menu, temos três opções, sendo o clique nos botões para cada opção efetuado através do movimento do rato e do botão esquerdo do rato, sendo que este é mostrado com um cursor. Para além

Figura 2 - Logotipo Original do Jogo Breakout

Figura 3 - Jogo com Opção Selecionada

disso, possuímos um relógio em tempo real no canto superior direito para mostrar ao utilizador a hora atual indicando hora, minutos e segundos.

No caso de o cursor sobrevoar alguma das opções, essa opção é realçada ao utilizador. Começando pela primeira opção, através do botão "PLAY" entramos no jogo em si. E na segunda opção entramos na lista de recordes (ver secção seguinte). Para finalizar podemos sair do menu principal através do clique em "EXIT" ou se pressionarmos a tecla *ESC* saímos do modo gráfico e da aplicação.

Lista de Recordes

Após o clique no botão "HIGHSCORES", entramos na lista de recordes do jogo, na qual qualquer recorde superior ao último por parte do utilizador entra nessa lista. Lista essa que é ordenada por maior pontuação, e que regista data de tal evento e

Figura 4 - Highscores

três letras que representam o utilizador que a obteve, tal como era feito nas antigas máquinas arcade.

Modo de Jogo

Depois de clicar no botão "PLAY" o jogo inicia automaticamente colocando logo em teste as reações do utilizador para apanhar a bola pela primeira vez, na qual a sua direção é aleatória.

Figura 5 - Gameplay

O jogo sendo um tradicional arcade é bastante simples e intuitivo. Sendo que no canto superior esquerdo mostra o *score* atual que vai atualizando à medida que o utilizador vai destruindo blocos, enquanto que no canto superior direito possuímos um contador de tempo desde que o jogo começou.

Figura 6 - Atari Breakout

Posteriormente, temos o aspeto do jogo em si que fizemos com que se assemelhasse ao original, mas dando uma cor mais viva e adicionando funcionalidades, como por exemplo, contador e bola redonda ao invés de uma quadrada.

O objetivo do jogo é destruir todos os blocos obtendo a maior pontuação possível, sendo que a destruição dos blocos ocorre na colisão da bola com qualquer um dos blocos. Sendo que sempre que uma colisão ocorre a bola ressalta. Tal como no jogo original a bola apenas se move em quatro direções, nordeste, noroeste, sudeste e sudoeste. Evidentemente, após qualquer colisão resulta de um ressalto no sentido oposto, seja uma colisão horizontal, ou vertical. Por isso, o utilizador deve controlar a barra para que a bola não passe para debaixo da mesma, pois caso deixe, este perde o jogo.

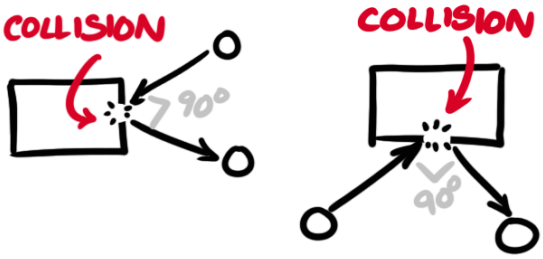


Figura 7 - Funcionamento das Colisões

O movimento da barra é controlado pelas teclas "A" e "D", tal como a maioria dos jogos atuais que usam o conjunto "WASD". Portanto, é usado o "A" para a deslocação para a esquerda e o "D" para a direita. O utilizador pode também usar os botões do rato para o movimento, sendo, claramente, o botão direito para a navegação para a direita e o esquerdo para a esquerda.

Por vezes, com probabilidades definidas por nós, a destruição dos blocos pode causar a queda de um *power-up*, podendo estes multiplicar a pontuação por cada bloco destruído por 4, por 2 ou a criação de uma bola auxiliar, sendo que esta é exatamente igual à bola principal, tirando o facto de que se cair, o utilizador não perde o jogo, e apenas desativa o *power-up* da bola extra. Assim, este último *power-up* testa a capacidade do utilizador de avaliar, em tempo real, o risco e a recompensa, pois terá de controlar duas bolas diferentes e, ainda, ter mais atenção a uma delas, pois apenas uma delas é que pode causa a derrota.

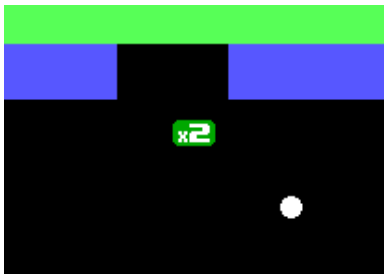


Figura 8 - Queda de um Power-Up

Todas estas atividades são geradas 60 vezes por segundo, no entanto, se algum objeto se mantem estático entre dois *frames*, esse objeto não volta a ser *rendered*. Isto é, apenas o que é atualizado no *frame* é que é redesenhado no próximo *frame*. Ao longo do desenvolvimento do projeto, considerámos implementar *double-buffer* ou até *triple-buffer*, no entanto, devido à simplicidade dos objetos e à presença fixa de muitos dos objetos apresentados em ecrã, decidimos atualizar, estritamente, o que é necessário. Aumentando, assim, o desempenho e atualizando, inteligentemente, apenas o essencial.

Submissão de Recordes

Após a derrota em jogo, se ultrapassarmos algum recorde definido na tabela de recordes (*figura 4*) aparece o ecrã de gravação de um novo recorde, no qual o utilizador escolhe as três letras que o irão representar na tabela de recordes através das teclas "W" e "S", que andam para cima e para baixo no abecedário, respetivamente, novamente trazendo memórias arcade a um jogo dos dias de hoje. Ao dar "SUBMIT" voltamos ao ecrã inicial, para o caso de o utilizador querer voltar a jogar. De referir ainda, todas estas trocas de "ecrãs" (por exemplo, do jogo para os recordes e dos recordes para o menu) são realizadas através de uma máquina de estados por nós criada.

Figura 9 - Submissão de Recorde

Estado do Projeto

Dispositivos Usados

Dispositivo	Utilização	Interrupção
Timer	Atualização do estado do jogo	Sim
Keyboard	Interface com o jogo e atalhos no menu	Sim
Mouse	Navegação nos menus e interface com o jogo	Sim
Video Graphics	Desenho de XPMs com texto, imagens e sprites.	Não
Real Time Clock	Data atual do computador	Não
UART		

Timer

A principal função do timer é a atualização do estado do jogo, sobretudo para desenhar os gráficos do jogo, isto é, atualizar as coordenadas da bola, verificar colisões, e o movimento da barra. Trata-se do dispositivo mais importante do projeto, pois todo o tipo de informação é manuseado pelas interrupções do timer.

Teclado

Este dispositivo é, também, usado na lógica do jogo, para além do rato. No menu inicial é usado para sair do programa (opção Exit) quando carregada a tecla "ESC" e durante o jogo esta tecla é usada para desistir.

Durante o jogo em si, serve para movimentar a barra, nomeadamente, mover a barra para a esquerda, "A", e para a direita, "D", e, depois, na submissão de recordes, para percorrer as letras do abecedário usando "W" e "S", para cima e para baixo, respetivamente.

Rato

O rato é utilizado fora e dentro do jogo, sendo que, fora do jogo serve para navegar o menu, através do movimento e do clique. Enquanto que, dentro do jogo, é usado, tal como o teclado, para mover a barra.

Placa Gráfica

Este dispositivo é outra das bases de todo o jogo, visto que é utilizado para desenhar todas os XPMs usados no programa. O modo vídeo que decidimos usar é 0x105, de resolução 1024 por 768 pixéis, usando a palete das cores fornecida.

De modo a tornar o jogo fluído, decidimos implementar uma técnica de atualização inteligente, já que no nosso programa muitos dos objetos ficam estáticos, que apenas atualiza o que é necessário. Quanto à deteção de colisões, achamos mais adequado desenvolver um algoritmo de deteção de colisões entre a bola e os blocos, tanto para a bola principal, como para a bola extra.

As letras e números que são usadas para apresentar a pontuação, recordes, menus, entre outros são apresentadas através de uma fonte por nós criada.

Real Time Clock (RTC)

O RTC é usado para ler a data e hora atual do computador e ser apresentado como informação adicional ao utilizador no menu inicial. Para além disso, quando um utilizador bate um recorde é registada a data em que tal feito aconteceu.

Organização do Código

Breaker – 2%

Este módulo é o módulo de inicialização na qual inicializa as *sprites* e carregamento de recordes e por fim entrada no ciclo principal do jogo.

I8042 – 2%

Módulo importado do código das aulas práticas dos laboratórios 3/4/5, com algumas modificações para que os botões do rato e teclado funcionassem corretamente.

I8254 – 2%

Importado do código das aulas praticas de laboratório 2.

Kbd – 8%

Este módulo foi utilizado para a função subscribe e unsubscribe do teclado para além da função que extrai a tecla do teclado após a sua notificação. Temos ainda uma função que possui um ciclo que é percorrido enquanto a tecla "ESC" não for premida.

Mouse – 5%

Este módulo foi utilizado para a função subscribe e unsubscribe do rato e para a função que ativa o mouse para além de definir o *stream mode.* Temos ainda a função que processa o *packet* retornado pelo rato.

Read_XPM – 2%

Importado do código das aulas praticas de laboratório 5 e possibilita a leitura de XPMs e a sua manipulação.

RTC – 5%

Este módulo foi criado com o intuito de extrair a data atual do computador tendo primeiro que verificar se o relógio está a ser atualizado, se o valor recebido está em complemento para dois, e realizar a conversão do valor anterior para binário.

Settings – 2%

Este módulo possui algumas constantes que definem o funcionamento do programa.

Sprites – 5%

Neste módulo estão presentes todos os XPMs que vão sendo utilizados ao longo do programa.

StateMachine – 10%

Este módulo foi utilizado a criação de uma máquina de estados sobre a qual o programa vai correr. O próximo estado é definido dependendo do evento que é enviado.

Timer – 20%

Este módulo foi utilizado para a função subscribe e unsubscribe do timer. Temos também uma função que executa um ciclo durante um número especificado de segundos ou interrupções. Sendo que apenas usámos tais funções em fases de testes muito iniciais. Neste módulo é processado o teclado, que conforme o estado do jogo em que o jogador se localiza, executa alguma função, seja para percorrer o alfabeto na submissão de recordes ou no jogo em si para mover a barra. Temos ainda, o processamento do mouse que, novamente, conforme o estado da maquina de estados, realça a seleção de um dos botões presentes nesse ecrã através da análise

do *packet* devolvido pelo módulo Mouse determinando a sua posição e o verificando, caso ocorra, um clique. Está ainda presente o processamento do RTC, que procede à atualização da *struct* do *real time clock* conforme a data extraída do computador.

Temos ainda neste módulo o ciclo principal sobre o qual o jogo corre que trata de todas as interrupções. É nesta função que o *mouse*, o *timer* e o *keyboard* são subscritos e apenas nesta função. Esta é a função que permite que o jogo corra a 60 *frames* por segundo, testando à medida que a bola se move colisões com blocos, barras e paredes. É também verificado quando o utilizador perde o jogo e se é necessário atualizar a lista de recordes devido a esse último jogo.

VBE – 2%

Este módulo foi utilizado apenas para definir macros para o modo gráfico.

Video_gr – 35%

Neste módulo são definidas as seguintes *structs:*

- **Ball –** Esta bola pode colidir com objetos para que contem pontos. Ressalta aquando existe uma colisão com algum objeto. É guardada a sua sprite, a sua posição e velocidade.
- **Brick –** Esta pertence a um bloco sendo que pode ter várias cores. Estes blocos são guardados num array destes mesmos. Guarda a sprite para cada um dos blocos e se está visível ou não.
- **Cursor –** Guarda a posição do cursor, sensibilidade e sprite.
- **Score –** Apresenta o valor do score através de três sprites diferentes, para cada um dos dígitos.

- **Timer –** Guarda os segundo e minutos desde que o utilizador começou a jogar, mostrando 4 sprites dependendo dos valores.

- **Paddle –** Guarda a sua posição, velocidade e sprite para a barra.

- **Powerup –** Esta estrutura guarda a posição do powerup, se está a cair, se está ativo e a sprite correspondente a cada powerup.

- **Highlight –** Esta é uma estrutura simples que indica que botão está realçado.

- **Highscore –** Esta é das maiores estruturas presentes neste jogo, no entanto é bastante simples. Contêm as letras que representam o jogador que obteve esse recorde, o valor do recorde e a data do recorde, sendo a esses valores associadas *sprites* para mostrar ao utilizador no modo gráfico.

- **Fontcycle –** Guarda a posição do alfabeto de cada letra aquando da submissão de recorde.

- **Clock –** Esta estrutura guarda todos os valores para a data (segundos, minutos, horas, dias, meses e anos) e *sprites* associadas.

- **Flow –** Esta estrutura define o multiplicador, a velocidade a que um power-up cai e a probabilidade de queda do mesmo.

- **Render –** Esta estrutura define que objetos têm de voltar a ser desenhados no ecrã conforme a sua necessidade.

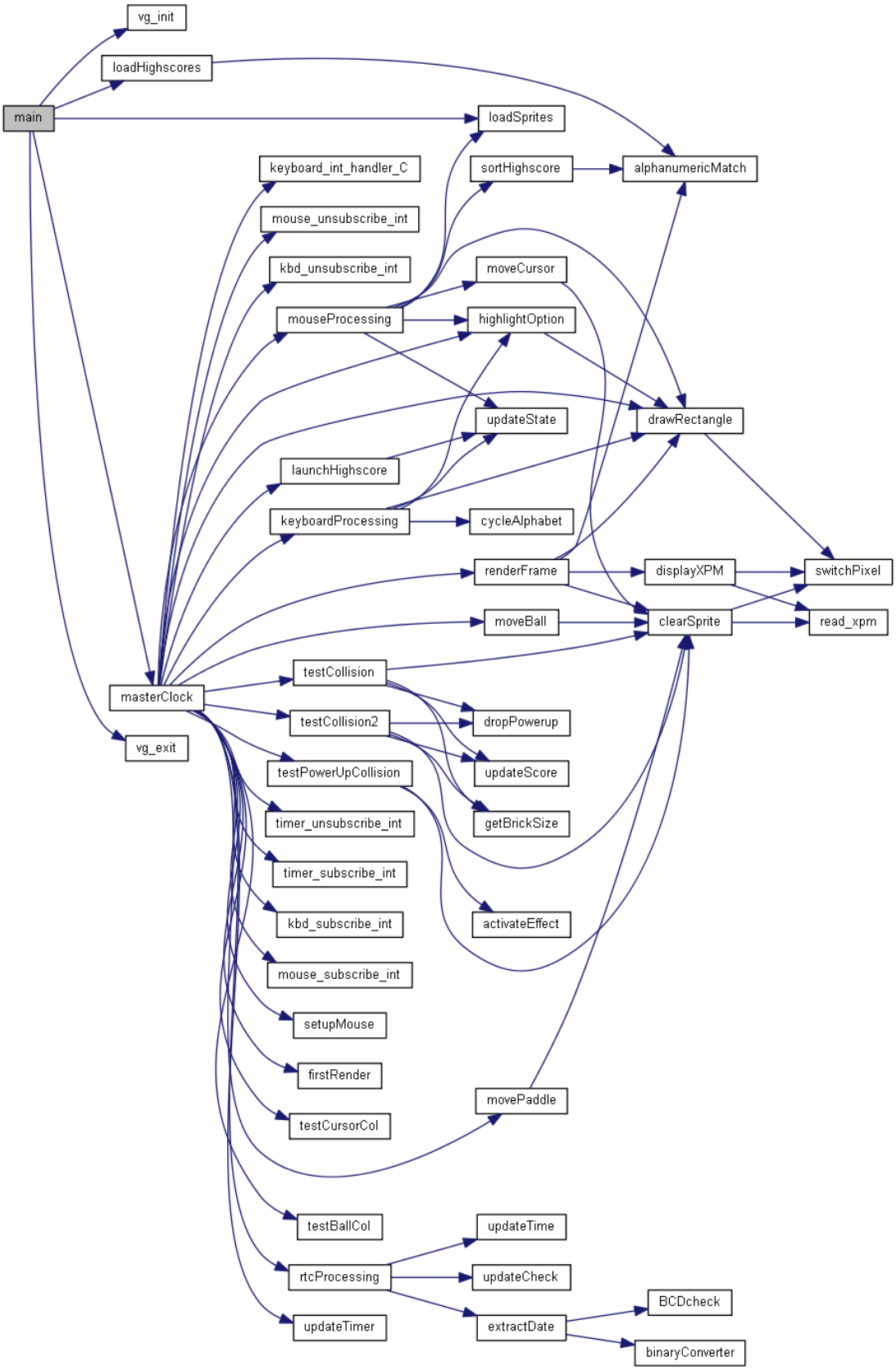
Para além disso, são criadas neste módulo alguns *arrays* necessários para o correr do jogo. Estão ainda presentes muitas funções, começando pelas funções de inicialização e saída do modo gráfico,

de mudança de pixel para uma cor definida, de desenho de retângulos e de desenho de XPMs. Criámos ainda uma função que apaga qualquer *sprite* do ecrã mudando a cor de todos os pixéis dessa mesma *sprite* para preto.

Estão, também presentes as funções de teste de colisão, de queda de power-up, e de associação de um número ou letra ao seu carater alfanumérico. Qualquer função de relativa aos recordes e a percorrer as letras do alfabeto estão, também, aqui definidas. Como o movimento de objetos faz com que sejam desenhadas novas *sprites* as funções de movimento estão, igualmente, neste módulo.

Para além disso, temos ainda uma função que carrega todas as *sprites* para os objetos presentes no jogo, uma que atualiza a pontuação, uma que atualiza o tempo gasto, outra que dá render ao *frame* inicial e, por fim, uma função que determina que objetos do ecrã têm de ser atualizados, sendo, portanto, esta a nossa função de atualização inteligente dos *frames*.

Gráfico de chamada de funções



Detalhes da Implementação

Ao longo do desenvolvimento deste projeto desenvolvemos alguns métodos que achamos interessante mencionar. Começando por todas as estruturas que definimos que nos facilitou à organização mesmo por vezes não sabendo o que o colega tinha implementado, visto que, bastava chamar o objeto e tínhamos acesso imediato às suas definições. Isto é, a programação orientada a objetos à qual nos cingimos foi bastante influente na nossa performance na programação.

Outra implementação importante foram as colisões que ocorrem de todos os sentidos tendo sempre ressalto lógico, entre as bolas principal e secundária com a barra, blocos e paredes, para além dos power-ups com a barra. O facto de após qualquer jogo terminar guardar os recordes num ficheiro facilita também o acesso permanente a informação que deve ir sendo guardada conforme os jogadores vão batendo os recordes anteriores.

Uma das implementações mais importantes foi, sem dúvida, o processamento com máquinas de estados, pois permitiu-nos segmentar o código dependendo do estado desta máquina. Esta última permitiu-nos ainda a uma nova implementação altamente importante, que segmenta o processamento do teclado e rato obedecendo ao estado ativo na máquina de estados. Assim, cada clique e cada teclar só funcionariam se o estado permitisse e dentro dos parâmetros definidos por ele.

Já a nossa implementação que informava o programa de o que tinha de voltar a redesenhar foi também relevante, já que através de uma das structs conseguíamos definir tudo o que tinha de ser desenhado evitando trabalho desnecessário de desenhar elementos estáticos como por exemplo os blocos ou as paredes.

Por ultimo, mas não menos importante, foi a nossa implementação de uma só função para correr todo o jogo. Nessa função, também dependente do estado da máquina de estados anteriormente mencionada, chamaríamos as várias funções que fazem o projeto correr, como, a função mencionada imediatamente atrás de desenhar apenas o necessário, testar as colisões, movimentar a bola, colisão de power-ups, etc. Em suma, esta função permite-nos, organizadamente, gerir tudo o que o programa faz.

Conclusão

Embora o desenvolvimento deste projeto final se tenha revelado uma experiência empolgante durante as últimas semanas, consideramos necessário realçar alguns aspetos que são exteriores ao desenvolvimento do trabalho em si e incidem mais na organização da unidade curricular.

De facto, terá havido algumas instâncias em que sentimos que o acompanhamento das aulas laboratoriais terá sido insuficiente, não pela falta de competência dos docentes/monitores, mas sim pela grande discrepância entre o número de dúvidas dos alunos e a capacidade de atendimento dos professores.

Em relação ao projeto, talvez a maior dificuldade terá incidido na organização estrutural do código. Obter uma resolução simples, prática e eficiente para alguns dos problemas que nos foram surgindo no decorrer do desenvolvimento foi, decididamente, desafiante.

Em conclusão, ambos os elementos do grupo trabalharam pelo menos 40 horas para a realização do projeto, tendo havido bastante empenho de ambas as partes.

Instruções de Instalação

Ao realizar o checkout do Redmine e, para assegurar o correto funcionamento do programa, é necessário o utilizador assegurar-se que extrai o programa para o local adequado. A pasta extraída, para além de possuir todos os laboratórios, terá também a pasta 'brickbreaker', a fonte do jogo.

Portanto, assegurando-se que tem permissão root, navegue para o diretório home/lcom/ através do comando 'cd'. De seguida, e assumindo que tem ligação VPN ativa, proceda ao checkout, utilizando o comando:

- "svn checkout https://192.168.50.135/repos/lcom1617-t1g13".

De seguida, verifique que tem os seus diretórios organizados da seguinte forma: **home/lcom/lcom1617-t1g13/brickbreaker**. Caso este não seja o caso, poderá ser impossível executar o programa com sucesso!

Para instalação do jogo, é necessário mover o ficheiro brickbreaker que se encontra na pasta conf para o diretório etc/system.conf.d.

Para executá-lo, bastará correr, com permissão root, o script 'run.sh' através do comando "sh run.sh" ou, alternativamente, 'service run `pwd`/brickbreaker'.