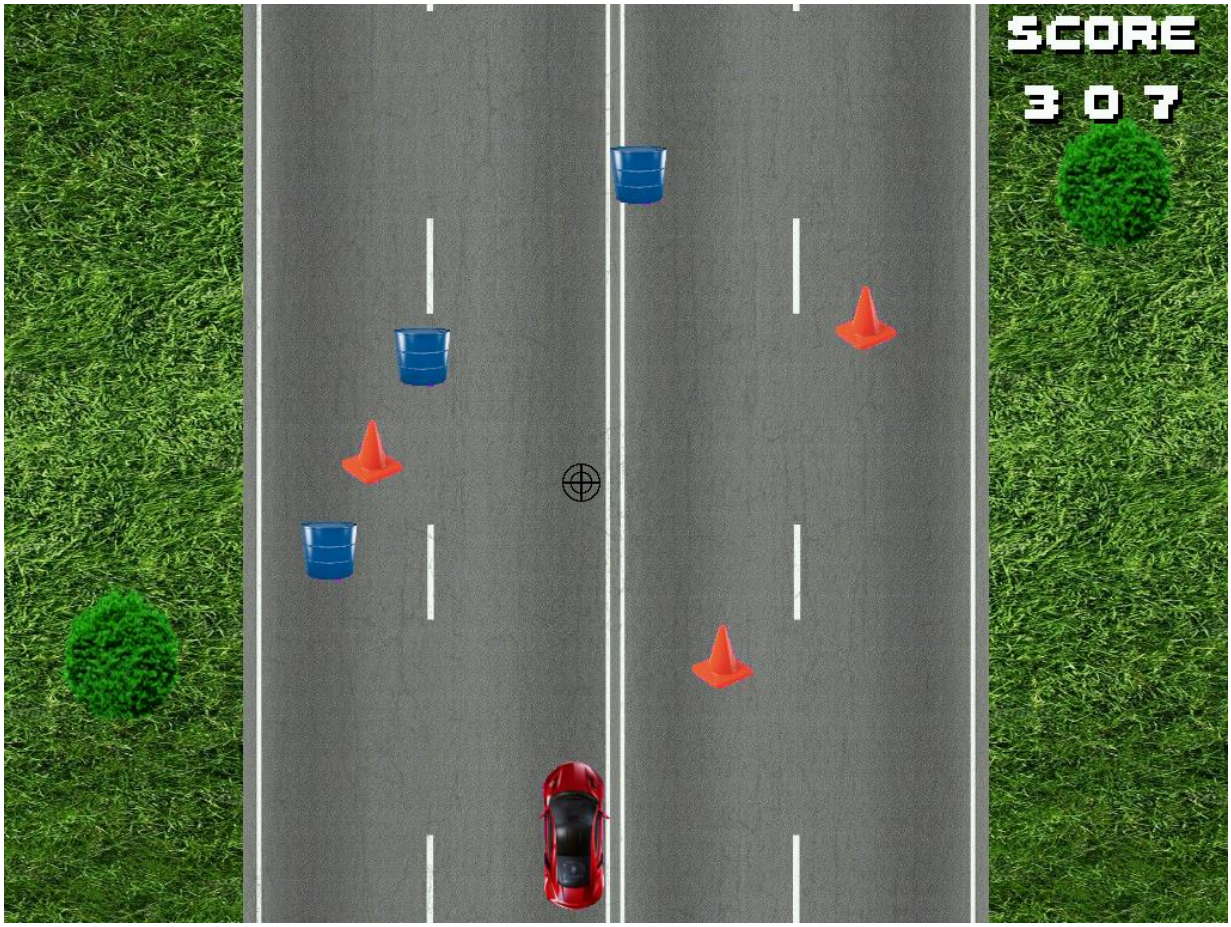
Laboratório de Computadores
EIC0020

Ano Letivo 2017/2018

Mestrado Integrado em Engenharia Informática e Computação
2º Ano

"Minix Vice"



Turma 6 – Grupo 1

João Conde – up201503256@fe.up.pt

Tiago Carvalho – up201504461@fe.up.pt

Índice

Tabela de Figuras

1 Instruções de Utilização do programa

1.1 Menu Inicial

Figura 1 - menu inicial

O menu inicial surge quando se inicia a aplicação, permitindo o uso do rato (através do movimento e dos botões) para selecionar a opção de 'Play' ou 'Quit'. Também é possível usar o teclado. Tal como indicado, caso seja primida a tecla 'H' é apresentado um ecrã de instruções sobre o jogo.

1.1.1 Botão "Play"

Este botão permite iniciar uma nova sessão de jogo.

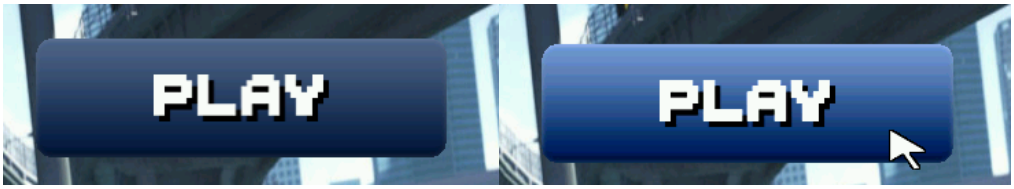


Figura 2 - botão "Play" normal / selecionado

1.1.2 Botão “Quit”

Este botão permite termina a sessão de jogo e sair da aplicação.

Figura 3 - botão "Quit" normal / selecionado

1.1.3 Atalho do teclado ‘H’

Primir esta tecla muda de menu, permitindo ao utilizador visualizar um painel de instruções visualmente simples e agradável ao olhar, interiorizando rapidamente os conceitos do jogo.

Através da tecla ESC, como mencionado, volta-se para o ecrã inicial.

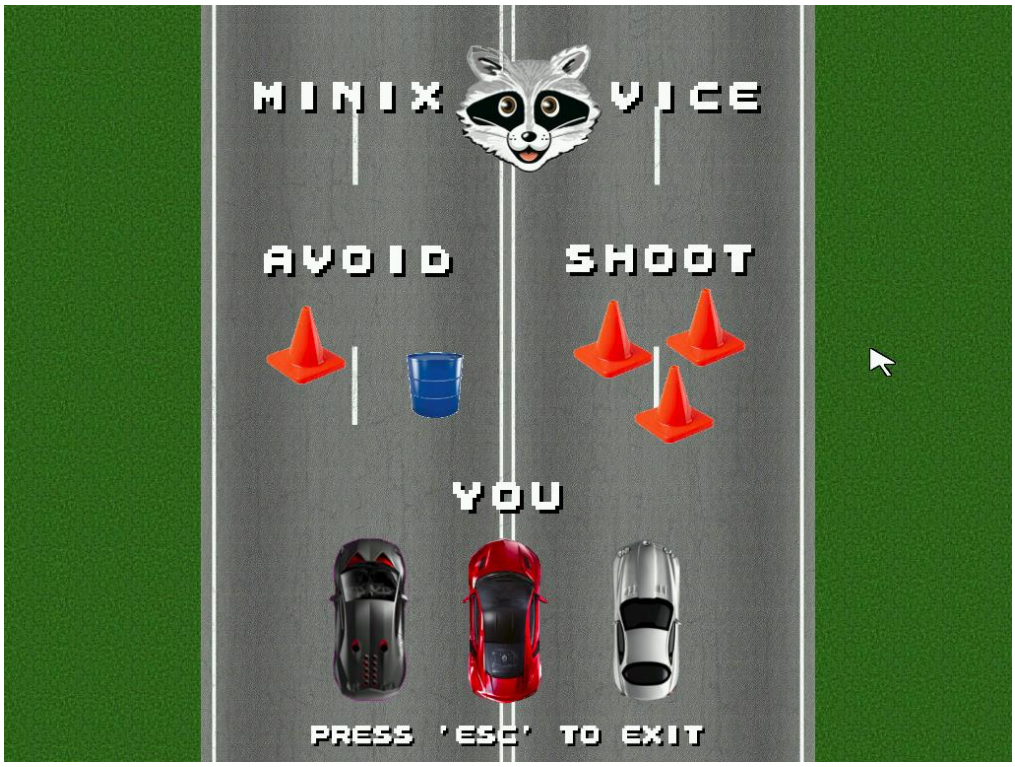


Figura 4 – Ecrã de Ajuda

1.2 Menu de Seleção de Carro

Neste menu devemos escolher o carro com que vamos jogar. Apesar de os carros terem as mesmas propriedades, o visual muda e torna a experiência visualmente mais agradável para o jogador. Assim, de entre os três carros devemos escolher um, sendo que estes três são um carro vermelho, um lamborghini preto e um mercedes cinza.

Figura 5 – Menu de Seleção

Cada um destes carros possui a propriedade de, quando o rato se encontra por cima do mesmo (hovering) se destacar dos outros. Em baixo visualiza-se esta propriedade.

Figura 6 – Rato em cima do Lamborghini

Figura 7 – Rato em cima do Mercedes

Figura 8 – Rato em cima do carro vermelho

1.3 Game

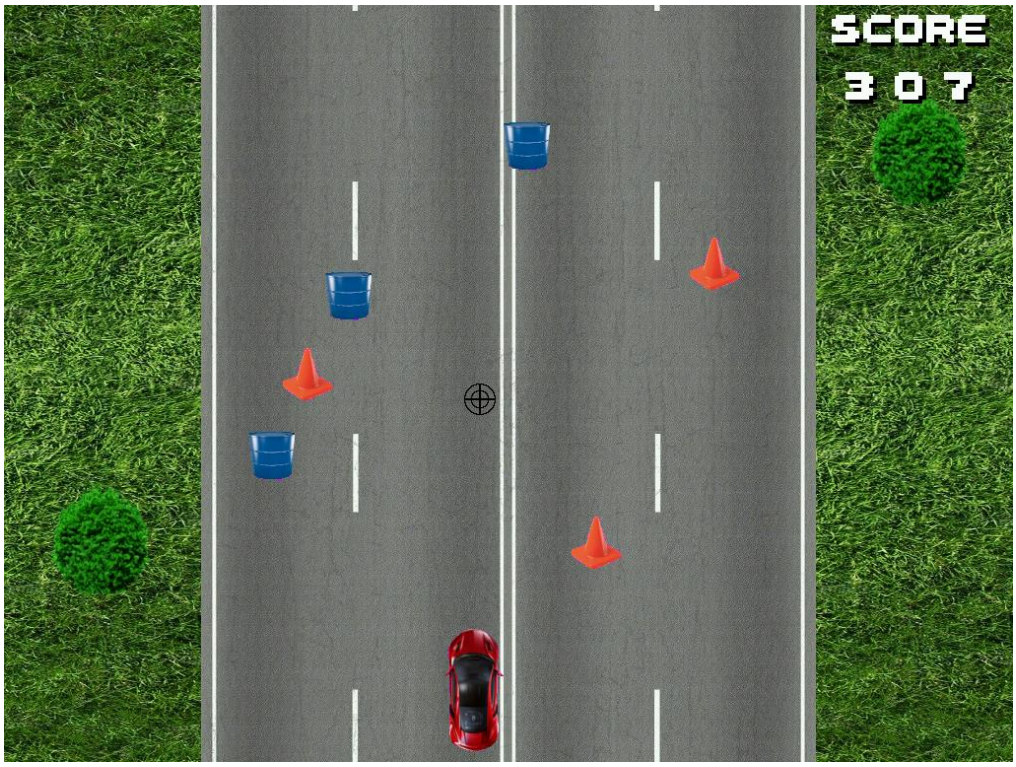


Figura 9 – Ecrã de jogo

No topo é visualizada a pontuação atual do jogador.

O jogador está limitado à estrada e os obstáculos são apenas aí gerados.

O jogador controla o carro e deve, através do uso das teclas A/D, virar à esquerda ou à direita, respetivamente, evitando assim os obstáculos no seu caminho. O toque com qualquer um destes sinaliza o término da partida. A pontuação contudo é calculada com base na velocidade do jogador. Assim, se pretender ganhar mais pontos deve acelerar o carro. Note-se que o incremento de pontuação por unidade de tempo não é linear relativamente ao aumento de velocidade. Assim, o jogador que se desloque mais rapidamente é beneficiado. Para acelerar deve ser usada a tecla W e para travar o S.

Figura 10 – Animação de um disparo bem sucedido

Durante a partida existem dos tipos de obstáculos: barris e cones de sinalização. A colisão com qualquer um destes é fatal. Contudo, a principal diferença entre os dois é a característica que os cones possuem de poder ser destruídos. Para este efeito, o jogador controla com o rato uma mira. Premindo com o botão esquerdo do mesmo sobre um cone, um tiro letal é disparado do carro para o cone atingido, sendo que o jogador será recompensado com um caminho mais livre, um incremento na pontuação e um espetáculo visual resultante da destruição do cone.

1.4 Game Over Menu

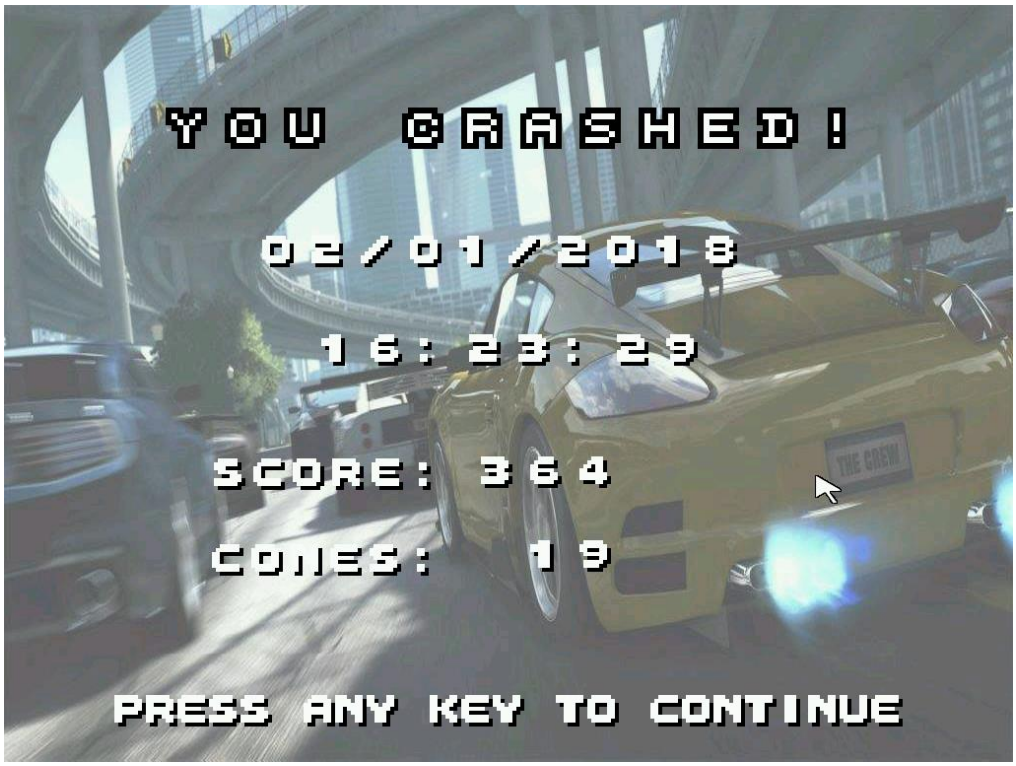


Figura 11 – Ecrã de game over

No ecrã final de jogo ou ecrã de game over, é apresentada a informação ao jogador que perdeu assim como estatísticas da sua sessão de jogo entre as quais a pontuação obtida e o número de cones destruídos. Também é visualizada a data e hora atuais, atualizadas no próprio ecrã, significando que enquanto aqui o jogador permanecer é feita a sua atualização, permitindo ao jogador ver a data e hora atual. Não foi visto grande interesse do RTC para o projeto, contudo, para sua utilização, aqui foi colocado, sendo considerado como o melhor sítio para tal.

Para retornar ao menu inicial e ser possível iniciar uma nova sessão de jogo qualquer tecla do teclado deve ser primida, após um intervalo de tempo definido, o que obriga o utilizador a visualizar o ecrã por algum tempo, impedindo que mal acabe o jogo evite este escrã, sendo confrontado com o seu falhanço... mas presenteado com estatísticas.

Aquando do disparo sobre um cone é originada uma explosão e um pop-up da pontuação ganha aparece em cima da mesma (o mesmo bmp '+10' é usado, so que deslocado veritcalmente para dar o efeito de "sair" da explosão).

Após o jogador perder a totalidade das suas vidas é automaticamente retirado do modo *singleplayer* e passa para a animação de final de jogo, vista em detalhe mais à frente. O jogador pode também regressar ao menu inicial a qualquer momento premindo a tecla *ESC*.

2 Estado do Projeto

2.1 Dispositivos Utilizados

Dispositivo	Utilização	Interrupção?
Timer	Atualizar o ecrã e todas as entidades de jogo.	Sim
Rato	Navegação e interação com os menus e controlo da mira de jogo (movimento e disparo).	Sim
Teclado	Controlo do jogador e interação nos menus.	Sim
Placa Gráfica	Desenho no ecrã dos menus e do jogo.	Não
RTC	Obtenção da data e hora atual no ecrã final do jogo.	Não

2.2 Timer

O *timer*, mais precisamente o *timer 0*, é utilizado para atualizar o estado do jogo e, respetivamente, todas as entidades nele presente, como os barris, cones e suas animações e claro o jogador. Para este efeito está configurado para gerar interrupções 60 vezes por segundo, por defeito.

2.3 Rato

O rato serve de interface de comunicação entre o utilizador e a aplicação. Para este efeito permite o clique e seleção de botões nos menus, bem como o disparo sobre cones no jogo.

Para facilitar a sua utilização, o funcionamento deste periférico (movimento e botões) foi abstraído através da *struct Mouse*, contendo a informação relevante sobre o mesmo.

2.4 Teclado

O uso deste periférico é evidente no controlo do carro, através do W/S/A/D permite acelerar o carro, travar, virar à esquerda ou virar à direita, respetivamente. Também nos menus é utilizado: no inicial, caso seja primido, dará display ao menu de ajuda onde com o ESC será possível regressar ao inicial. No ecrá final é possível sair primindo qualquer tecla.

2.5 Placa Gráfica

A placa gráfica é utilizada no modo RGB 5:6:5 com resolução 1024 pixéis de largura por 768 pixéis de altura (modo de código 0x114), sendo usada para expor imagens no ecrã (imagens essas criadas ou editadas por nós). No entanto, o código necessário para desenhar, carregar e apagar *Bitmaps* é da autoria de Henrique Ferrolho, estando disponível no seu blog. Porém, para cobrir as nossas necessidades, foi necessário alterar o algoritmo de desenho dos bitmaps. De facto, a transparência de imagens é conseguida através de uma cor de transparência (rosa-choque). Esta cor simula a transparência e, no código por nós alterado, quando se verifica que a cor de um determinado pixel é a predefinida como de transparência, não é pintada. Sendo assim, a cor de fundo da maioria das nossas imagnes é a de transparência. Contudo a continua verificação de cor de cada pixel a pintar no ecrã pode ser desnecessária, nomeadamente quando pretendemos desenhar um fundo (por exemplo fundo de jogo ou fundo de um menu). Nestes casos, possuimos uma função alternative que desenha o bitmap sem testar ou ter em conta a cor de transparência. É, como já mencionado, usado no caso do fundo de jogo ou dos menus.

Fazemos também uso da técnica *double buffering. Assim, o "flickering" da imagem é evitado.* Este fenómeno, por nós experienciado inicialmente, designa-se por "screen tearing". Ocorre quando um artefacto visual a ser desenhado no ecrã está atualmente a mostrar informação relativa a múltiplos frames, num único desenho no ecrã. Este problema ocorre quando a informação desenhada no ecrã não ocorre ao mesmo ritmo que a informação lhe é passada.

A deteção de colisões é feita com recurso a retângulos (não visíveis) que acompanham cada entidade de jogo. Assim, foram desenvolvidos algoritmos que verificam colisões entre as necessárias formas geométricas.

Implementamos também a animação de explosões, através de 16 *sprites* em rápida sucessão e outra que serve de pop-up da pontuação ganha aquando o abate de um cone.

2.6 RTC (*Real Time Clock*)

O RTC (relógio em tempo real) é utilizado para efeitos de display da hora e data atual. Pode ser visualizada aquando o fim do jogo, no ecrã de estatísticas.

3 Organização e estrutura do código

Todo o código respeitante aos seguintes módulos (exceto o creditado ao aluno Henrique Ferrolho) foi desenvolvido pelo estudante João Conde. Contudo, o contributo do estudante Tiago Carvalho é significativo na edição bitmaps e decisão gráfica.

3.1 Bitmap

Funções básicas para fazer loading, desenhar e apagar imagens no ecrã. Os ficheiros são .bmp. Devem ser exportados em modo 16-bit (rgb 5:6:5). A grande maioria do código é da autoria de Henrique Ferrolho (no fim do relatório creditado novamente com link para o seu blog). Contudo, para suprimir as necessidades do grupo, foi alterado de modo a permitir imagens com transparência ou, como originalmente, sem transparência. Com este propósito foi reservada a cor rosa-choque para cor de transparência. Para além disso, foram realizadas mudanças para permitir desenhar o nosso fundo que se mexe com o tempo, proporcionando a ideia de movimento.

As imagens usadas no nosso projeto são todas editadas ou criadas por ambos os elementos do grupo, usando os programas de edição de imagens "Photoshop", "GIMP" e "Paint", ainda que o crédito pela maioria das imagens devido ao elemento Tiago Carvalho.

3.2 bmpHandling

Responsável pelo loading dos bitmaps necessários ao funcionamento correto do programa, assim como a sua eliminação.

Esta *struct* implementa comportamento semelhante a uma classe *singleton*. Para este efeito tem o seu construtor definido apenas no ficheiro *.c* e precedido pela *keyword* "static", de modo a limitar o número de instâncias a uma. Para aceder ao único objeto da *struct* foi definido um método que, caso esse objeto esteja instanciado, devolve-o, caso não esteja instanciado cria-o e devolve este novo.

3.3 IO

Este módulo refere-se a todos os dispositivos de Input/Output do programa. Subscreve e desubscreve as interrupções de dispositivos como o timer, o rato e o teclado e gere com recurso a polling o RTC.

3.4 MinixVice

Este módulo abstrai uma "classe" típica de jogo no paradigma de programação orientada a objetos. Está implementada para que exista apenas uma instância da mesma, seguindo o padrão de desenho singleton. Possui as funções e variavéis de controlo do jogo.

3.5 ColliderBox

Abstrai uma “caixa” que envolve uma entidade para testar colisões ou, por exemplo, cliques de rato ou mesmo se o rato simplesmente passa por cima do retângulo definido pela “caixa”.

3.6 i8042

Mac*ros* cruciais para o funcionamento do teclado e rato.

3.7 i8254

Mac*ros* cruciais para o funcionamento do *timer*..

3.8 entities

Este módulo contêm todas as structs que abstraem as diferentes “entidades” de jogo como botões, menus, obstáculos, animações e o jogador.

3.9 kbd

Contém o conjunto de funções essenciais para o funcionamento do teclado.

3.10 graphics

Módulo gráfico responsável pelo desenho de todas as entidades do jogo (excetuando o rato que possui o seu método de desenho).

3.11 game

Módulo responsãvel pela execução do jogo. Inicializa-o, gere o seu ciclo e termina-o. Diretamente invocado por main para começo da sessão de jogo.

3.12 mouse

Contém o conjunto de funções essenciais para o funcionamento do rato. Código desenvolvido aquando do laboratório 4.

3.13 logic

Contém toda a lógica de jogo. Atualiza as entidades de jogo, gere os diferentes estados de jogo e verifica a ocorrência de colisões, assim como cria, atualiza e termina animações.

3.14 RTC

Contém as funções que permitem a operação e configuração do RTC.

3.15 timer

Contém o conjunto de funções essenciais para o uso e configuração do *timer*.

3.16 state_machines

Contem a definição dos varios estados e eventos de jogo. Funções para a atualização do estado do rato, jogador e jogo, mediante determinados eventos.

3.17 VBE

Contém as funções que permitem a obtenção de informações sobre o modo de vídeo utilizado, bem como sobre a placa gráfica..

3.18 video_gr

Contém o conjunto de funções que permitem a operação e configuração da placa gráfica.

3.19 Function Call Graph

Achamos mais útil incluir o *function call graph* para a função *play()*, que é responsável por iniciar a sessão de jogo, geri-la e no fim termina-la. Esta é diretamente invocada pela função main() do projeto, de tal forma que a função main tem como unico proposito invocar esta.

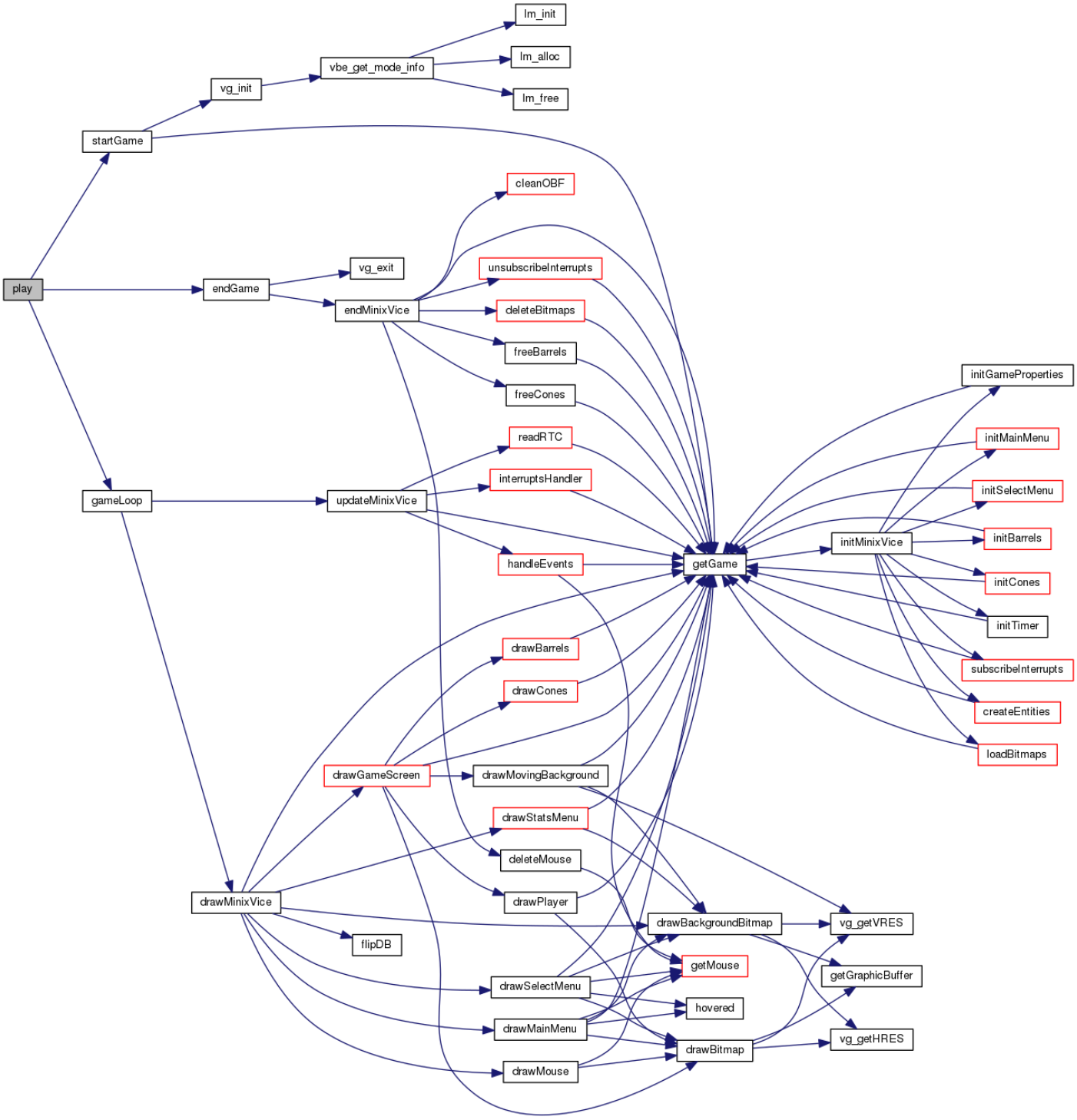


Figura 12 – Call graph of function play()

4 Detalhes de Implementação

O desenvolvimento da aplicação teve sempre em mente uma estruturação *event driven*, conseguida pelo uso de várias máquina de estados, bem como o tratamento de interrupções em si. As máquina de estados implementadas tem como objetivo facilitar a programação e compreensão das mudanças entre os diferentes estados do jogo, jogadores e do rato. Todo o funcionamento do programa se baseia no tratamento de interrupções (eventos gerados pelos periféricos) no ciclo *driver_receive* único, executando adequadamente as tarefas a realizar como resposta a determinada interrupção. Todas estas características, para além de tornarem o código mais legível, contribuem para a uma fácil escalabilidade das funcionalidades implementadas.

Mantivemos também uma consistente modularidade ao longo do desenvolvimento, separando claramente funcionalidades, que nos permite uma maior abstração na implementação do jogo em si. Para este efeito procuramos também seguir uma orientação a objetos onde possível (object oriented C), apesar das limitações da linguagem neste campo (inexistência de classes propriamente ditas, sendo abstraidas por structs).

É de realçar que no jogo o que na verdade se "move" são todas as entidades de jogo e o fundo, dado a ideia de movimento ao carro, quando na verdade este se encontra parado na posição inicial, apenas se movimentando na horizontal.

Creditando novamente o aluno Henrique Ferrolho, é de realçar que foi mudado parte do código por ele fornecido, não só para conseguir desenhar imagens com transparência mas também para desenhar um fundo em movimento.

Efetivamente, ao longo do projeto são frequentemente usadas variáveis do tipo "ColliderBox" por nós definido, sendo que esta abstração nos ajudou bastante, uma vez que, através da mesma, realizamos todo o teste de colisões ao longo do jogo assim como deteção de cliques ou de passagem do rato por cima de botões nos menus.

No que diz respeito a animações, as structs criadas "Shot" e "Bonus" fornecem uma base bastante sólida para implementação das mesmas, sendo que foram, devido a boa organização do código, rapidamente implementadas e dão um visual muito bom ao jogo elevando a experiência do jogador que é, em qualquer jogo, o objetivo final.

5 Conclusões

5.1 Avaliação da Unidade Curricular

Podemos afirmar que a carga horária exigida ultrapassa de longe qualquer outra unidade curricular. Porém, o trabalho está uniformemente distribuído ao longo do semestre, o que facilita consideravelmente a realização do mesmo e do projeto final.

Não obstante, a tolerância com as datas de entrega para entrega dos trabalhos laboratoriais e mesmo componentes do projeto, deve ser aumentada consideravelmente. Isto pois, no meio de tanto detalhe e pequenas metas e entregas que a cadeira impõe e, não sendo a única que os estudantes frequentam, o stress aumenta consideravelmente. Mesmo aquando a escrita deste relatório, para a entrega final, somos deparados com informação contraditória sobre como proceder à mesma, uma vez mais contribuindo para o receio de , por distração ou simplesmente má compreensão, e não falta de conhecimentos leccionados em LCOM, de ser penalizados injustamente por questões de logística.

Provavelmente a melhor ideia que temos para o melhoramento da unidade curricular é recorrer ao que nos foi dito como sendo algo do passado do MIEIC: a troca entre a UC de SOPE (Sistemas Operativos) e LCOM, em termos de ocorrência temporal. SOPE proprociona aos estudantes muito à vontade com conhecimentos que não são o objetivo de LCOM mas que facilitam a compreensão dos mesmos.

Com tudo o que foi dito acima, sentimos que a UC de LCOM é uma unidade desafiante mas recompensadora e com conteúdo interessante. Porém, devido a más decisões de logística e de métodos de avaliação perde um pouco o interesse para a maioria dos estudantes.

A nosso ver, a UC de LCOM foi uma das mais recompensadoras e, honestamente, das que mais nos contentou com o resultado final. É lamentável que por questões não relacionadas com a matéria leccionada pela cadeira em si, os estudantes do MIEIC se sintam desecorajados na realização da mesma e esta seja injustificadamente infame entre os mesmos.

5.2 Apêndice

Para facilitar a instalação dos recursos necessários ao funcionamento do programa, implementamos alguns *shell scripts* simples. Assim, as instruções de instalação, compilação e execução são as seguintes:

- Entrar no diretório do projeto *minix-vice*;
- Trocar de utilizador para superuser;
- Correr o *script install.sh*;
- Correr o script *compile.sh*;
- Correr o *script run.sh*.