Faculdade de Engenharia da Universidade do Porto

Sky attack

Relatório do projeto

Projeto Laboratório de Computadores 2022/2023 - 2ºano em Licenciatura em Engenharia Informática e Computação

Docente regente: Pedro Alexandre Souto

Docente das aulas Teórico-Práticas: Mário Cordeiro

Grupo 1 : Turma 15

Estudantes & Autores:

up202108769 Andreia SIlva up202108769@up.pt

up201202666 Bruno Drumond up201202666@up.pt

up202108687 Tomás Ramos up202108687@up.pt

Índice

Descrição do jogo

Sky Attack é um jogo inspirado em Heli Attack 2, no qual o jogador embarca numa batalha contra uma série de helicópteros, enfrentando um de cada vez. A sua missão é destruir o maior número possível dessas aeronaves enquanto se mantém vivo.

No jogo, o soldado utiliza o rato para direcionar seus disparos contra os helicópteros e as teclas A, W e D para se movimentar e escapar dos disparos inimigos. Conforme se progride no jogo, a dificuldade aumenta gradualmente, os helicópteros tornam-se mais ágeis e seus disparos mais intensos.

Telas de jogo

Ecrã inicial

No início do jogo, durante o dia, o jogador é apresentado com a seguinte imagem:

A partir das 18h e até às 6h, o jogador é apresentado com um fundo de noite:

A opção “Start” inicia o jogo e “Instructions” mostra ao jogador as instruções do jogo, no fundo do ecrã é mostrada a hora e dia atual. As opções são selecionadas através do rato e quando este se encontra em cima duma destas, o tipo de letra é alterado:

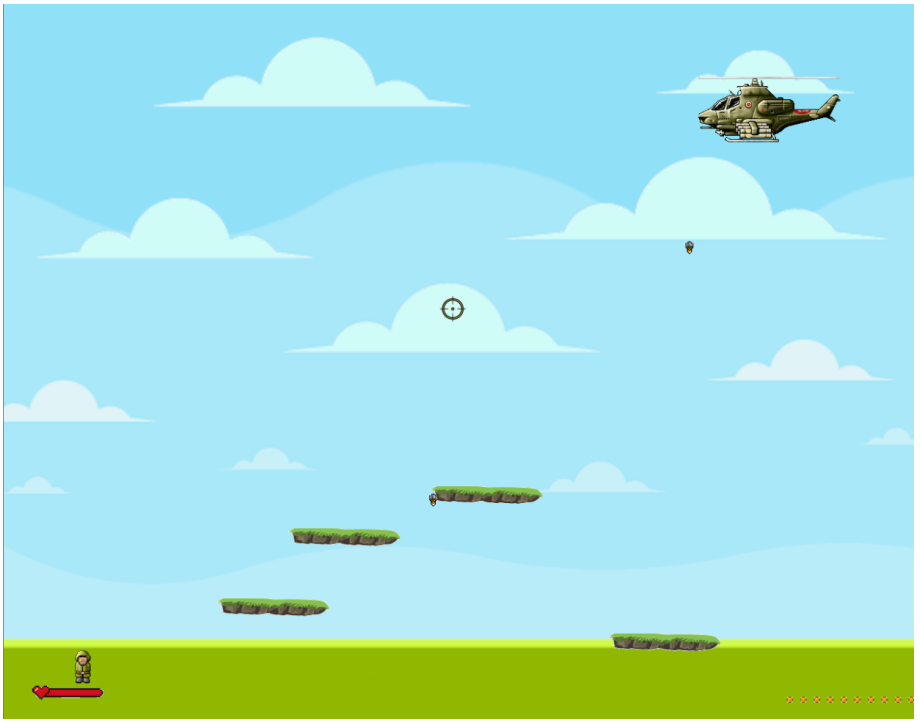
Ecrã de instruções

No ecrã de instruções o jogador é apresentado com uma curta explicação de como jogar:

Ao pressionar “Back”, o jogador retorna ao ecrã principal. O fundo do ecrã também é alterado consoante a hora do dia, para a imagem mostrada na figura 2.

Ecrã de jogo

Durante o jogo, o jogador é apresentado com a seguinte imagem:



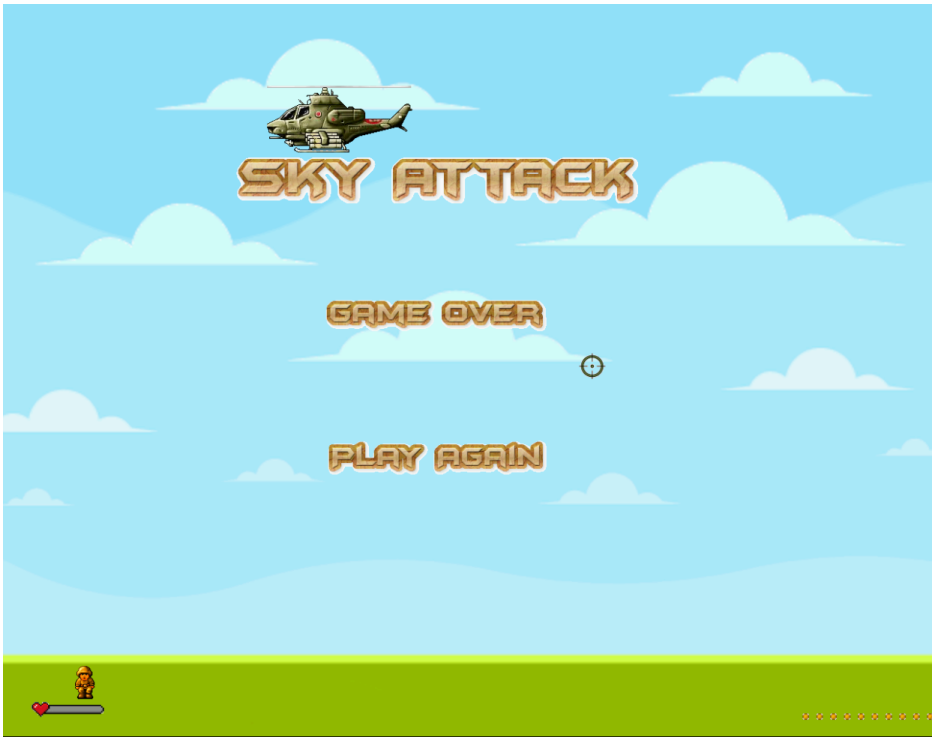
No fundo do ecrã podemos ver as balas disponíveis (canto inferior direito) e a barra de vida do soldado (canto inferior esquerdo).

O jogo termina quando a barra de vida do jogador estiver vazia.

Este ecrã também é alterado consoante a hora em que o jogador se encontra a jogar.

Ecrã final

Após o soldado perder a sua vida, o seguinte ecrã é mostrado:



O fundo do ecrã depende a hora atual. No ecrã encontra-se o estado em que o jogo termina, com o helicóptero e jogador nas últimas posições, a barra de vida do soldado vazia e o soldado morto. Com o rato é possível regressar ao ecrã principal, premindo “Play Again”.

Dispositivos

Os dispositivos utilizados no projeto foram o *timer*, o teclado, o rato e o *real time clock.*

Dispositivos	**Utilização**	**Interrupção**
Timer	Atualizar o movimento dos objetos do jogo e a sua ilustração. Controla o tempo decorrido, como o tempo para o carregamento de balas ou tempo para aumento de dificuldade do jogo.	Sim
Teclado	Controlar o movimento do jogador.	Sim
Rato	Navegação pelo menu e disparo de tiros.	Sim
Placa gráfica	Exibir o menu e o jogo.	Não
Real Time Clock	Selecionar o fundo de ecrã e mostrar o dia e hora atual no menu principal.	Não

Timer

O *timer* tem diversas funcionalidades. A cada interrupção do *timer*, as funções responsáveis por desenhar o rato, o fundo do ecrã, a personagem, o helicóptero, as respectivas balas e todas as imagens que aparecem no ecrã são chamadas. A cada interrupção a hora atual também é atualizada, para ser possível confirmar as horas a serem demonstradas no menu principal e o segundo plano a ser selecionado.

Além disso, o *timer* também é utilizado para chamar as funções para atualizar o movimento do helicóptero (gerado de forma automática), das balas e do jogador. A utilização do *timer* para atualizar a posição da personagem foi necessária para que este se tornasse mais natural, desta forma, enquanto uma tecla está pressionada, o jogador irá continuar o seu movimento até deixar de ser pressionada, mesmo que outras teclas sejam premidas.

As interrupções do *timer* são também utilizadas para controlar a frequência com que o jogo aumenta a dificuldade (a cada 30 segundos de jogo o helicóptero torna-se mais rápido e a disparar com maior frequência e as suas balas terão maior velocidade), controlar o tempo de carregamento da arma (a cada 10 tiros disparados o soldado precisa de 2 segundos para carregar a arma com outras 10 balas) e o tempo para atualizar a frame do soldado, concedendo-lhe um movimento mais natural.

Teclado

O teclado é utilizado, maioritariamente, para controlar a posição do jogador. Ao receber um scancode, é verificado se este é de alguma das teclas relevantes para o jogo, na circunstância de ser o *makecode* da tecla A ou D, o jogador irá continuar esse movimento até ao respectivo *breakcode* ser passado, e no caso de receber o *makecode* da tecla W o jogador iniciará um salto até cair no chão ou numa das plataformas.

O teclado é também utilizado para verificar a condição de saída do jogo. Quando a tecla esc é pressionada o jogo termina e desliga.

Rato

A cada interrupção do rato, a posição do cursor é atualizada. O rato é utilizado para selecionar os itens do menu, através de um clique no botão esquerdo, caso o rato se encontre numa zona do menu que pode ser selecionada, esta irá ficar destacada.

A sua principal função é disparar tiros, comparando a posição do jogador e do cursor, e assim definir o ângulo necessário para lançar a bala, e, posteriormente, definir a sua velocidade no eixo das abscissas e das ordenadas.

Placa gráfica

O modo escolhido para desenhar imagens (bitmaps) é o 0x11A, com resolução de 1280x1024, que utiliza o formato de píxel de 5:6:6, ocupando assim 2 bytes por cor.

Foi utilizado o formato de *double buffering* para tornar o jogo mais fluido e a animação do sprite do jogador. Quanto à detecção de colisões, achamos mais adequado realizá-la através da posição de cada objeto, em substituição da placa gráfica.

O load dos sprites é realizado todo no início do jogo, para que este se torne mais natural no seu decorrer.

Real Time Clock

O real time clock é utilizado para definir o fundo do ecrã do jogo, no caso de ser uma hora diurna o fundo será durante o dia, de outra forma, o fundo será noturno e para mostrar ao jogador a hora e dia atual no menu principal.

Organização do código

Módulo do *timer*

Ficheiro com as funções desenvolvidas no laboratório 2 para controlo do *timer* usadas no projeto.

Módulo do *keyboard*

Ficheiro com as funções desenvolvidas no laboratório 3 e 4 para controlo do teclado e rato e funções extra relevantes para o movimento da personagem. Este módulo contém também a estrutura do rato utilizada no projeto:

```
typedef struct{
  uint16_t x;
  uint16_t y;
} Mouse;
```

Módulo do vídeo

Ficheiro com as funções desenvolvidas no laboratório 5 para controlo da placa gráfica.

Além destas, também tem a função `vg_load_xpm`, na qual todos os sprites utilizados no projeto são carregados ao iniciar o jogo, e tanto o bitmaps como os `xpm_image_t` são guardados em dois arrays. Desta forma, quando é necessário ilustrar um dos sprites na tela, chama-se a função `vg_draw_xpm` com o id do sprite, usando a macro correspondente, definida em xpm_id.h. Este macro corresponde à localização das informações do sprite nos arrays.

A função `free_buffer` é utilizada para libertar a memória do segundo buffer e de todas as sprites.

Módulo utils

Ficheiro com funções realizadas ao longo dos laboratórios e utilizadas nos restantes ficheiros.

Módulo de *interrupt handler*

O ficheiro ih.c contém a função necessária para inicializar o jogo, que dá subscribe a todos os dispositivos com os quais são utilizados interrupções, inicializa a placa gráfica e carrega todos os sprites.

Neste módulo estão as funções responsáveis por lidar com as interrupções de cada dispositivo em cada estado de jogo.

A função responsável por fechar o jogo também se encontra neste módulo, responsável por dar unsubscribe a todos os dispositivos, fechar a placa gráfica e libertar a memória que foi alocada anteriormente.

Módulo de menu

Neste módulo encontra-se definido o enum do estado do jogo GameState, podendo ser uma das seguintes opções:

```
typedef enum {
    MAINMENU,
    INSTRUCTIONS,
    GAME,
    GAMEOVER
} GameState;
```

Encontram-se neste módulo a função responsável por repor os dados do jogo quando o jogador pressiona “Play Again” no menu.

Além disso, contém as funções para alterar o estado atual do jogo quando o jogador pressiona alguma das opções do menu que sejam relevantes, como “Start” ou “Play Again”.

Módulo bullet

Módulo com a estrutura utilizada para guardar a informação de uma bala:

```
typedef struct {
    uint16_t x;
```

```
    uint16_t y;
    int16_t vx;
    int16_t vy;
    bool in_game;
} Bullet;
```

E as funções utilizadas para criar, atualizar a posição, verificar colisões e desenhar todas as balas do jogo.

Módulo helicopter

Módulo com a estrutura utilizada para guardar informações relevantes do helicóptero:

```
typedef struct {
  uint16_t x;
  uint16_t y;
  uint16_t vx;
  uint16_t vy;
  uint8_t hp;
  bool alive;
} Helicopter;
```

Inclui também as funções que geram o movimento, o disparar, o aumento da dificuldade do helicóptero, assim como as funções responsáveis por gerar a morte do helicóptero, incluindo a explosão quando esta ocorre e gerar um novo helicóptero após o anterior ser destruído.

Módulo platforms

Neste ficheiro encontram-se as funções relativas às plataformas, incluindo as funções para inicializar, definindo a posição de cada plataforma, para obter as suas localizações, atualizar as posições e a função para desenhar as plataformas.

Módulo player

Inclui a estrutura utilizada para guardar a informação do soldado:

```
typedef struct {
    uint16_t x;
    uint16_t y;
    uint8_t hp; // between 0 and 100
    uint8_t frame;
} Player;
```

Contém as funções responsáveis por movimentar, disparar e desenhar o jogador. As funções responsáveis por desenhar a barra que mostra a restante vida do jogador e que verifica a sua colisão com as plataformas também se encontram neste módulo.

Módulo main

Contém a função main do projeto e a função responsável pelo ciclo principal do jogo, que chama as funções para iniciar, lidar com as interrupções e fechar o jogo.

Módulo canvas

Contém as funções que desenham aspetos comuns do jogo, como o menu, o segundo plano, o rato e a data e horas.

Módulo xpm_sprites.h

Ficheiro que contém todas as structures do tipo `xpm_row_t` com as sprites utilizadas no jogo.

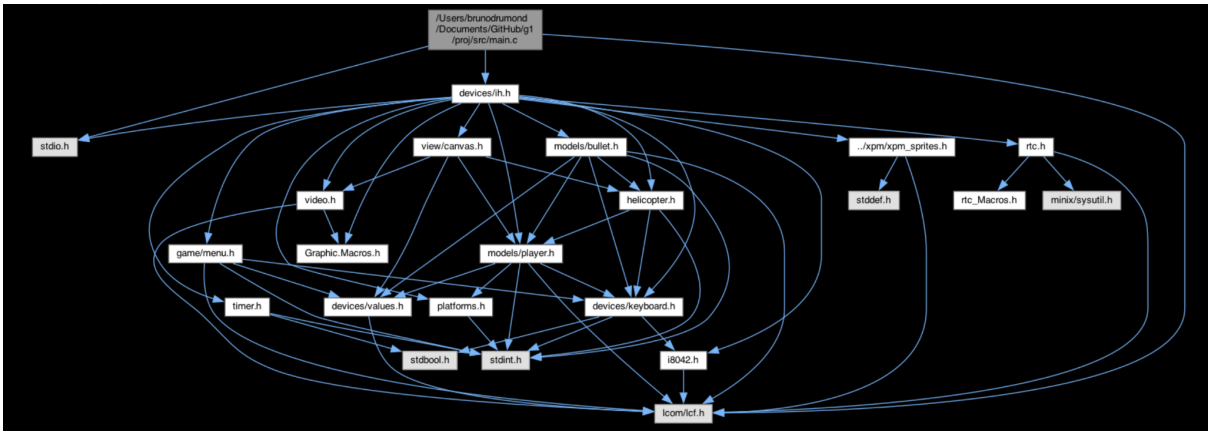
Módulo xpm_id.h

Ficheiro com os macros dos identificadores de cada sprite, que correspondem à sua localização nos arrays de bitmaps e images.

Pesos relativos

Ficheiro	Peso
i8042	3%
i8254	3%
ih	14%
keyboard	11%
rtc	5%
timer	1%
utils	1%
video	12%
xpm_id	2.5%
menu	5%
bullet	9,5%
helicopter	8%
platforms	2%
player	13%
canvas	6%
values	1%

Grafo de chamadas de funções



Detalhes da implementação

Durante a implementação do projeto, pudemos aplicar vários tópicos abordados nas aulas de forma criativa, demonstrando nosso domínio dos conceitos da unidade curricular. Alguns desses tópicos incluíram máquinas de estado, utilização dos diversos dispositivos e geração de frames.

Uma parte fundamental da implementação envolveu o uso de uma máquina de estados para verificar e controlar o estado atual do jogo. Essa abordagem permitiu a transição suave entre os diferentes menus, a jogabilidade principal e a tela de "game over".

A detecção de colisões foi um aspecto crucial do jogo. Implementamos um sistema que verifica continuamente as colisões a cada movimento do jogador, garantindo que ele permaneça dentro dos limites permitidos. Além disso, ao atualizar as balas, verificamos se elas colidem com o elemento ou se saem do ecrã do jogo.

Para controlar a velocidade das balas em diferentes direções, utilizamos funções matemáticas disponíveis no header "math.h". Essas funções permitiram-nos calcular e ajustar a velocidade das balas separadamente para os eixos X e Y.

No que diz respeito aos menus, cada um deles foi equipado com uma função para verificar a posição do cursor e detectar se o cursor está sobre algum botão. Em caso afirmativo, o estado atual do jogo é alterado de acordo, permitindo uma navegação fluente entre as opções e uma interação intuitiva com o usuário.

Além dos tópicos abordados nas aulas, também tivemos que aprender e aplicar conceitos adicionais por conta própria. Um exemplo disso foi a utilização do Real-Time Clock (RTC) para definir a configuração de tempo do jogo, alternando entre o dia e a noite e ilustrar a data e hora do momento. Essa funcionalidade trouxe um elemento adicional de imersão ao ambiente de jogo, criando uma atmosfera visualmente diferenciada.

Uma característica interessante que implementamos no jogo foi a utilização de plataformas. As plataformas permitem ao jogador se aproximar ou se distanciar de diferentes pontos do cenário, adicionando uma camada estratégica às suas habilidades de disparo. Essa mecânica de plataformas adicionou uma dinâmica interessante ao jogo, permitindo ao jogador adaptar sua estratégia de combate de acordo com a posição e localização do inimigo.

Uma estratégia eficiente que adotamos para manter nosso código limpo e legível foi a utilização de dois arquivos separados contendo macros (#define) para definir os valores constantes utilizados em nosso projeto. Essas macros permitiram-nos atribuir nomes significativos aos valores e reutilizá-los em todo o código-fonte, tornando mais fácil entender e modificar esses valores quando necessário. As únicas ocasiões em que utilizamos

diretamente valores foi quando consideramos irrelevante a criação de macros para a situação, como em ciclos “for”.

No geral, a implementação deste projeto permitiu-nos aplicar uma ampla gama de tópicos cobertos nas aulas, bem como explorar e aprender conceitos adicionais por conta própria. Ao superar os desafios e empregar soluções criativas, pudemos criar um jogo envolvente e divertido que demonstra nossa compreensão e aplicação dos conceitos aprendidos ao longo da unidade curricular.

Conclusões

No decorrer deste projeto, inicialmente planejamos implementar o uso da *serial port* para permitir que o segundo jogador controlasse o helicóptero, além de incorporar algumas funcionalidades adicionais, como possíveis melhorias na arma do jogador.

Durante o processo de desenvolvimento, encontramos alguns desafios técnicos que dificultaram a integração efetiva da *serial port* no jogo. Apesar dos nossos esforços, não conseguimos alcançar uma solução completamente funcional dentro do escopo e prazo estabelecidos para o projeto.

Ao longo do projeto, aprendemos valiosas lições sobre planeamento, gerenciamento de recursos e a importância de adaptar-se às circunstâncias. Reconhecemos a necessidade de definir prioridades claras e estabelecer uma comunicação eficiente para garantir um melhor desenvolvimento de futuros projetos.

Apesar de não termos alcançado todas as funcionalidades inicialmente planeadas, estamos satisfeitos com os resultados obtidos e a aprendizagem adquirida. Este projeto serviu como uma base sólida para futuros projetos, permitindo-nos explorar novas possibilidades e melhorar continuamente nossas habilidades de desenvolvimento de jogos e trabalhos de equipa.