Relatório do jogo Dave, The Jumper

Projeto para Laboratório de Computadores

Turma 6 Grupo 11
Carolina Rosemback Guilhermino – up201800171
Victor Saldanha Nunes – up201907226

Este relatório irá conter a explicação dos mais importantes mecanismos e detalhes sobre a produção do jogo “Dave, The Jumper”.

Índice

1. Instruções para o Usuário

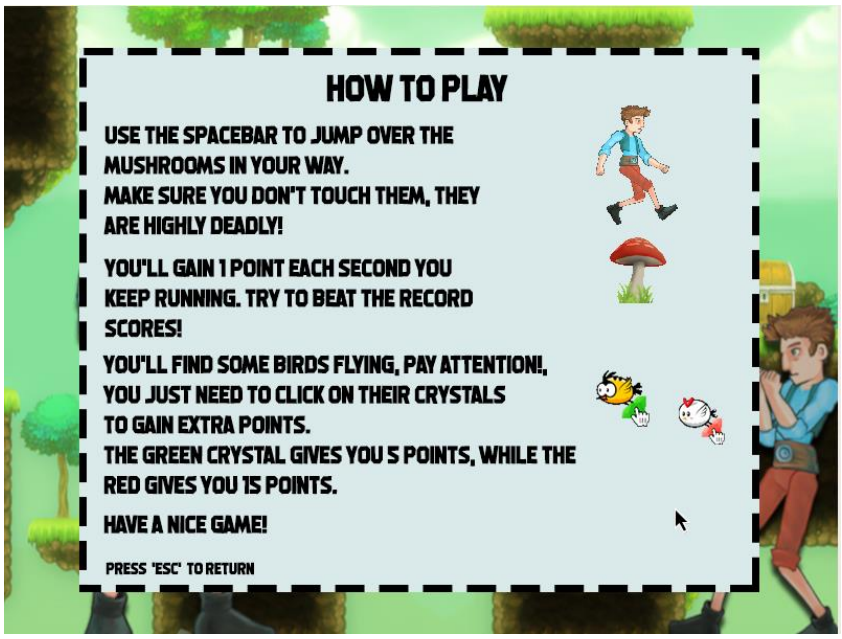
1.1. Menu Inicial

Este é o menu principal do jogo. Nele podemos escolher uma das opções representadas pelos botões. Para selecionar um botão, podemos usar tanto o teclado, usando as setas verticais ou as teclas W e S para navegar entre os botões e apertando ENTER para efetivar a escolha, quanto o rato, colocando o cursor em cima do botão pretendido e clicando com o botão esquerdo.

Detalhes Relevantes: Note a mudança de aparência do cursor, se tornando uma pequena mão, quando ele está em cima de um objeto clicável (neste caso, os botões). Este detalhe está presente em todo o jogo, o cursor sempre irá indicar se um objeto é clicável ou não.
Vejamos detalhadamente o que há em cada uma das opções do menu principal.

1.2. Instruções

Esta opção tem a função de explicar ao jogador como se jogar o jogo.



Como podemos ver, o jogo consiste em manter-se vivo o máximo de tempo possível, saltando sobre os cogumelos sem tocá-los. A cada segundo vivo na partida, o jogador ganha 1 ponto.

Para saltar, basta pressionar a barra de espaço, só sendo permitido um pulo simples, não havendo double jumping. Caso haja qualquer tipo de colisão do personagem com algum cogumelo, o personagem irá morrer e a partida irá se encerrar.

O jogo também conta com pássaros que carregam cristais consigo. Basta clicar nestes cristais para ganhar pontos extras, o cristal verde vale 5 pontos, e o cristal vermelho (mais raro) vale 15 pontos. A coleta de cristais são apenas um bônus, o jogador não perde a partida se não os coletar.

1.3. Play

1.3.1. O jogo

Vejamos agora o jogo propriamente dito.

Como podemos ver, a pontuação atualizada do jogador é mostrada no canto superior direito da tela ao longo do jogo. A partir do momento em que o personagem morre, mesmo enquanto ele ainda está no processo de queda, o jogador para de receber 1 ponto por segundo. Também já não é mais possível clicar nos cristais presentes na tela.

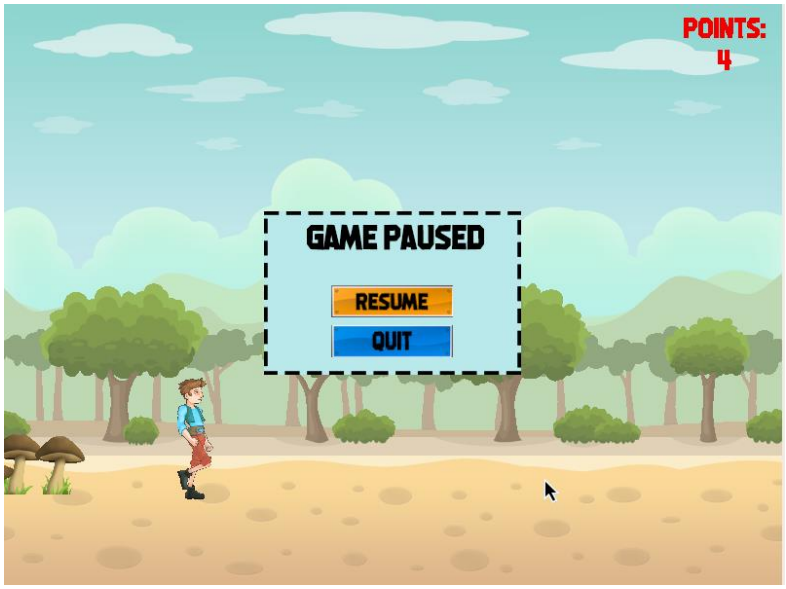
jogabilidade é exatamente como foi descrita na secção das instruções.

Detalhes Relevantes:

1. A pontuação apresentada no canto superior direito está sempre atualizada e centralizada, com base na largura que ocupa os dígitos.
2. Quando clicados, os cristais somem e há um efeito de uma pequena explosão brilhante indicando que o cristal foi coletado com sucesso.

1.3.2. Opção de pausar

Ressalta-se que durante o jogo é possível pausar, pressionando a tecla ESC.



A função de pausar não está disponível quando o personagem está morrendo ou já está morto, e obviamente o jogador não receber 1 ponto por segundo enquanto o jogo estiver pausado.

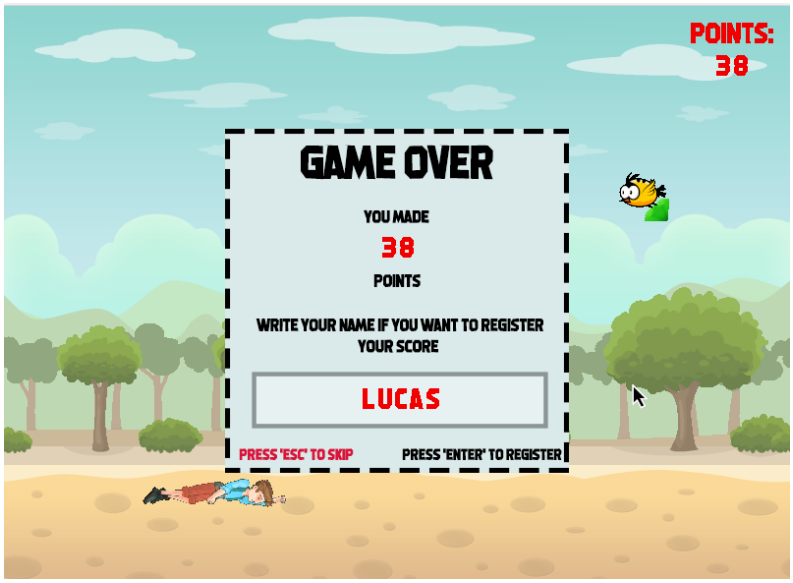
O menu de pausa conta com 2 botões: RESUME e QUIT. Pode-se usar tanto o rato quanto o teclado para selecionar os botões, tal qual no menu principal.

O botão de RESUME da continuidade ao jogo. Também é possível voltar ao jogo simplesmente pressionando a tecla ESC de novo. O botão de QUIT indica que o jogador deseja abandonar a atual partida, sem registrar seus pontos. Ao clicar nesta opção, ainda há uma tela de confirmação por uma questão de segurança. Para confirmar basta escolher YES, para retornar ao menu de pausa basta escolher NO ou simplesmente pressionar ESC.

1.3.3. Janela de Game-Over

Quando o personagem morre, aparece a tela de game-over. Esta tela mostra a pontuação que o jogador fez na partida, e pergunta se o

jogador deseja registrar a sua pontuação. Caso não queira registrar, basta pressionar a tecla ESC. Para registrar, basta escrever o nome ou pseudônimo (é possível escrever tanto letras quanto números) que deseja e pressionar o ENTER para confirmar, não sendo possível registrar com um nome vazio.



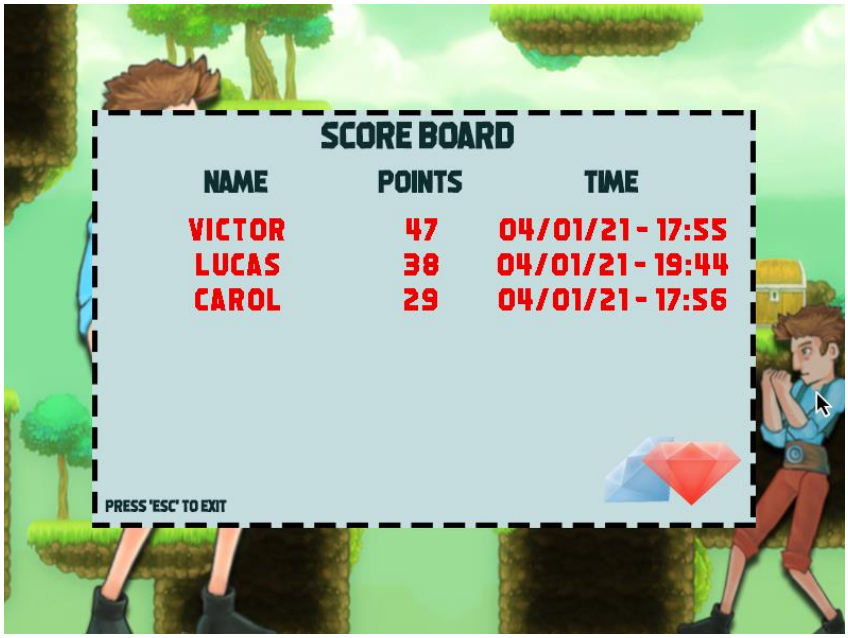
Detalhes Relevantes:

1. O nome do jogador fica sempre centralizado automaticamente conforme o mesmo vai escrevendo as letras, sendo possível deletar caracteres pressionando o BACKSPACE.
2. A medalha de HighScore, no canto superior direito da janela, apenas aparece se o jogador nesta partida ultrapassar a maior pontuação registrada até então.
   Ou seja, se registrar esta pontuação, certamente estará em primeiro lugar no Scoreboard. Em partidas ordinárias, esta medalha não aparece.

1.4. Scoreboard

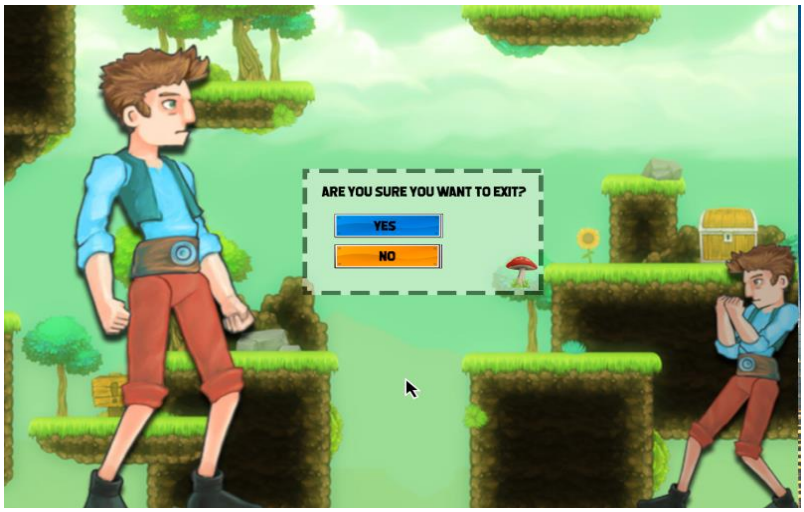
Esta janela mostra as 5 melhores pontuações registradas, em ordem do melhor ou pior (em caso de empate, prevalece o jogador que alcançou aquela pontuação primeiro). Cada linha possui informações da

pontuação, do nome ou pseudônimo do jogador e a data e hora que aquela pontuação foi alcançada.



1.5. Sair

Nesta janela há simplesmente uma confirmação se o jogador quer realmente sair do jogo, e se ele confirmar o programa se encerra. Vale lembrar que mesmo o programa se encerrando e sendo aberto posteriormente outra vez, os recordes no Scoreboard continuam salvos visto que há a utilização de registro destes recordes num ficheiro de texto exterior.



2. Status do Projeto

Esta secção mostrará em detalhes quais foram os dispositivos de entrada e saída utilizados, como eles foram utilizadas e como eles interagem para formar o resultado do jogo.

Dispositivos	Funcionalidade	Interrupção
Timer	Controlar o frame rate do jogo Acrescentar pontos por tempo vivo ao jogador e checar colisões Controlar geração de pássaros e cogumelos	Sim
KBD	Navegação nos menus e na pausa do jogo Realizar saltos durante o jogo Permitir o jogador escrever o nome para registro da pontuação	Sim
Mouse	Navegação nos menus e na pausa do jogo Clicar nos cristais durante o jogo	Sim
Video Card	Exibição de menu e janelas Exibição do cenário dinâmico e todos os objetos nele incluso Em suma, toda a parte visual do jogo	Não
RTC	Coletar as informações de data e hora do registro de certa pontuação	Não

Para um esclarecimento melhor sobre esta secção, vale explicar como foi estruturado a parte para lidar com os dispositivos de entrada e saída. Esta parte está dividida entre os ficheiros game.c e events.c.

O ficheiro game.c é onde as interrupções são recebidas, ligando a respectiva variável global de ocorrência de um evento neste dispositivo. Estes eventos são tratados no ficheiro events.c, que fará as ações resultantes daquele evento naquele estado de jogo em específico. Depois de atendido o evento, a variável global de ocorrência de evento daquele dispositivo volta a ser falso.

2.1. Timer

Usamos para o timer o interrupt handler normal usado no laboratório. Ele gera 60 interrupções por segundo, mas só ativa variável global **timer_event_ocurred** a cada duas interrupções, ou seja, há 30 eventos do timer a cada segundo, isto faz com que o frame rate do jogo seja 30 FPS. Os eventos do timer são usados principalmente para desenhar as sprites necessárias na tela, dependendo do modo de jogo em que se estiver.

Os eventos do timer também acionam funções de geração aleatória de cogumelos e pássaros, é também responsável por controlar o tempo para o jogador receber 1 ponto a cada segundo que permanecer vivo e checar colisões contra cogumelos.

Funções relevantes para desenhar os menus:
(usadas nas linhas 624-656 do events.c)

void draw_menu_buttons(MenuOptions menu_option);
void draw_scoreboard_window();
void draw_instructions_window();
void draw_exit_window();
void draw_yes_no_buttons(YesNoOptions exit_option);
void draw_pause_window();
void draw_pause_buttons(PauseOptions pause_option);
void draw_quit_window();
void draw_gameover_window();

void draw_highscore_medal(int points);

(todas as funções anteriores foram implementadas em menu.c)

Funções relevantes para desenhar as sprites no jogo:
(usadas nas linhas 657-719 do events.c)

void draw_moving_scenario(bool move);
(implementada em scenario.c linha 14)

void draw_character(Character* character, bool move);
(implementada em character.c linha 94)
void draw_mushrooms(bool move);
(implementada em mushroom.c linha 96)

void draw_birds_and_crystals(bool move);
(implementada em birds_and_crystals.c linha 183)

Funções relevantes para geração de cogumelos e pássaros:
(usadas nas linhas 672, 677, 684 do events.c)

void mushroom_generator(int player_points);
Uma maior pontuação causa uma maior frequência na geração de cogumelos (máximo de 5 cogumelos na tela, e gerados de modo a evitar ao máximo um cenário de morte certa)
(implementada em mushroom.c linha 47)

void bird_and_crystal_generator();
(implementada em birds_and_crystals.c linha 139)

mushrooms_collision_detection(Character* dave);
(implementada em mushroom.c linha 116)

Funções relevantes para desenhar outras sprites:

void draw_cursor(Cursor* cursor);
Usada em todo o projeto
(implementada em mouse.c linha 143)

void draw_number_centralized(int number_points, int x_initial, int x_final, int y);
(implementada em letters_and_numbers.c linha 64)

void draw_word_centralized(char* word, int x_initial, int x_final, int y);
(implementada em letters_and_numbers.c linha 104)

2.2. Teclado

O teclado tem a função de navegar entre os menus, fazer o personagem saltar, pausar o jogo e escrever o nome do jogador ao final do jogo. Cada scancode coletado gera um evento no teclado, e as ações deste evento podem ser das mais variadas dependendo do estado do jogo em que se está.

Somando-se todas as situações, as teclas que potencialmente podem realizar ações no projeto são: Todas as letras e números (tanto as teclas numéricas em cima das letras quanto as da direita perto do Num Lock), ENTER, ESC, BACKSPACE, SPACEBAR e as setas verticais.

No geral os eventos de tecla chamam poucas funções. A importância maior está na mudança de variáveis globais que representam estados do jogo, nomeadamente **game_mode**, **menu_option**, **pause_option** e **yes_no_option**.

Também há a alteração do atributo status do objeto Character para JUMPING para provocar o salto.

Funções relevantes para o jogador escrever o seu nome:

void name_write_char(char letter);
(implementada em letters_and_numbers.c linha 155)
void name_delete_char();
(implementada em letters_and_numbers.c linha 168)

void register_score(int points, char *name, int day, int month, int year, int hours, int minutes);

Invocada após o jogador escrever o seu nome, para registrar de fato o registro, tanto nos vetores de registros quanto no ficheiro de texto através da função **write_scores_to_file()**(implementada em scoreboard.c linha 11)

2.3. Rato

O rato também apresenta funções relevantes. Ele possui o papel de auxiliar na navegação dos menus, representando uma alternativa ao teclado.

Durante o jogo, ele também desempenha a importante função de permitir que o jogador colete os cristais dos pássaros e assim ganhe pontos extras.

As propriedades do rato utilizadas foram a sua movimentação e o botão esquerdo. É possível visualizar e movimentar o rato em qualquer janela do jogo, e sempre que for possível clicar em um objeto, o cursor se tornará uma pequena mão.

Assim como o teclado, o mouse também pode modificar as variáveis globais de estado do jogo **game_mode**, **menu_option**, **pause_option** e **yes_no_option**.

Funções relevantes para transformação do modo do cursor:

void become_regular_cursor(Cursor* c);
(implementada em mouse.c linha 155)

void become_hand_cursor(Cursor* c);
(implementada em mouse.c linha 147)

Funções relevantes para movimentação do cursor:

void update_cursor(Cursor* c);
(implementada em mouse.c linha 121)

void move_cursor_sprite(Sprite* sp);
(implementada em sprite.c linha 159)

Funções relevantes para a coleta de cristais:

bool cursor_over_crystal(int x, int y, Crystal* c);

(implementada em birds_and_crystals.c linha 221)

void click_on_crystal(Crystal* c);
(implementada em birds_and_crystals.c linha 241)

2.4. Placa Gráfica

Para este jogo, estamos usando o modo de vídeo 0x115, tendo assim uma resolução de 800x600 px, e usando o modo de cor de 24 (8:8:8) bits por pixel, possibilitando um total de 2^24 cores.

O projeto também utiliza a técnica de double buffering, sendo a imagem toda carregada antes no endereço **double_buffer** e posteriormente escrita de uma só vez no endereço **video_mem**, através da função **write_buffer_to_videomem()** (implementada em graphic.c linha 113).

O projeto também conta com um diverso número de animações de sprites, através do uso de alguns arrays de sprites. Possui também um detector de colisão entre o personagem e os cogumelos.

Funções Relevantes Para Desenhar Sprites:

void draw_sprite(Sprite* sp);
void draw_sprite_and_background(Sprite* sp);
void draw_sprite_no_packman_effect(Sprite* sp);
void erase_sprite(Sprite* sp);
void move_sprite(Sprite *sp);
(implementadas em sprite.c)

int drawPixel(uint16_t x_pos, uint16_t y_pos, uint32_t color);
(implementada em graphic.c linha 101)

Funções Relevantes Para Configurar a Placa Gráfica:

int init_mode(uint16_t mode);
int my_get_mode_info(uint16_t mode, vbe_mode_info_t *mode_info);
(implementadas em graphic.c)

Funções Relevantes Para Detecção de Colisão e Double Buffer:

void write_buffer_to_videomem();
(implementada em graphic.c linha 113)

bool collition_detection(Sprite *sp1, Sprite *sp2);
(implementada em sprite.c linha 170)

2.5. RTC

O RTC foi usado para obter a data e a hora exata em que o jogador jogou o jogo. Caso sua pontuação esteja dentro dos valores de recorde, ele poderá escrever se nome para estar registrado ao scoreboard. Junto a seu nome e pontuação, estará a data (em formato dd/mm/aa) e a hora (em formato hh:mm) em que foi obtido tal recorde.

Funções relevantes obter informações de data e hora:

void getDate();
(implementada em rtc.c linha 42)

void getTime();
(implementada em rtc.c linha 61)

3. Organização e Estrutura de Código

Está secção trará informações sobre cada módulo do qual o projeto foi dividido, trazendo uma visão holística da estrutura e organização do código do projeto.

3.1. Módulos

proj.c – Ficheiro que contém a função main do projeto. Neste ficheiros basicamente ocorre a inicialização do modo de vídeo, subscrição de interrupções dos dispositivos de E/S, chama a função **start_dave_the_jumper()** (função está que dá começo ao jogo “Dave,

The Jumper" e só acaba ao jogador sair do jogo), desinscreve os dispositivos de E/S e encerra o modo de vídeo, voltando ao modo texto do MINIX.
Peso: 2% *Autor: Carolina Rosemback*

game.c – Ficheiro que contém a função que chama o driver_receive **game_main_loop()**, ele que se encarrega de receber as interrupções e ligar as variáveis globais de evento para serem tratadas posteriormente no ficheiro events.c. Além disso, ela também inicializa todas as sprites e estruturas usadas no jogo, e ao final as destroem.
Peso: 6% *Autor: Victor Nunes*

events.c – Ficheiro que se encarrega de realizar as ações necessárias tendo em conta o evento que acabou de acontecer e o estado do jogo atual, incluindo os elementos que devem ser desenhados na tela. Este ficheiro é o mais importante do projeto.
Peso: 26% *Autor: Victor Nunes*

Principais Estruturas de events.h:

```
typedef enum {
  MENU_SCREEN,                 //Estar no menu principal
  SCOREBOARD_SCREEN,           //Estar na tela de ScoreBoard
  INSTRUCTIONS_SCREEN,         //Estar na tela de instruções
  EXIT_SCREEN,                 //Estar na tela de sair do jogo
  PLAYING_SCREEN,              //Estar jogando uma partida
  PAUSE_SCREEN,                //Estar na tela de pausa
  QUIT_SCREEN,                 //Estar na tela de abandonar a partida
  GAMEOVER_SCREEN,             //Estar na tela de game-over
  CLOSE_GAME                   //Efetivamente fechar o jogo
} GameModes;                   //Representa os diversos estados do jogo
```

graphic.c – Ficheiro que trata de tudo relacionado com a inicialização da placa de vídeo e seu encerramento, e contém a função drawPixel.
Peso: 3% *Autor: Victor Nunes*

sprite.c – Ficheiro muito importante que contém todas as funções relacionadas a sprites, incluindo diferentes modos de desenhá-las no

ecrã. A relevância deste ficheiro é grande pois suas funções são usadas por diversos outros, como menu.c, character.c, mushroom.c, birds_and_crystals.c, entre outros.
Peso: 10% Autor: Victor Nunes

rtc.c – Ficheiro que contém tudo relacionado com a RTC, possuindo também as funções para buscar a data e hora do momento.
Peso: 1% Autor: Carolina Rosemback

timer.c – Ficheiro que contém o interrupt handler do timer, que é usado no projeto.
Peso: 1% Autor: Nunes/Rosemback

keyboard.c - Ficheiro que contém o interrupt handler do teclado, que é usado no projeto.
Peso: 3% Autor: Nunes/Rosemback

mouse.c - Ficheiro que contém o interrupt handler do rato, assim como funções relacionadas à estrutura cursor.
Peso: 5% Autor: Nunes/Rosemback

menu.c – Ficheiro que trata de desenhar as janelas e botões dos diversos tipos de menu durante o jogo, dependendo do estado do jogo.
Peso: 8% Autor: Victor Nunes

Principais Estruturas de menu.h:

```
typedef enum {
    PLAY,                  //Botão Play em seleção
    SCOREBOARD,            //Botão Scoreboard em seleção
    INSTRUCTIONS,          //Botão Intructions em seleção
    EXIT                   //Botão EXIT em seleção
} MenuOptions;             //Botão do menu principal em seleção

typedef enum {
    YES,                   //Botão YES em seleção
    NO                     //Botão NO em seleção
} YesNoOptions;            //Botão do menu de exit/quit em seleção
```

```
typedef enum {
        RESUME,                 //Botão RESUME em seleção
        QUIT                    //Botão QUIT em seleção
} PauseOptions;                 //Botão do menu de pausa em seleção
```

birds_and_crystals.c – Ficheiro que contém tudo relacionado com o desenho e geração de pássaros e cristais, assim como o efeito de brilho.
Peso: 8% *Autor: Victor Nunes*

Principais Estruturas de birds_and_crystals.h:

```
typedef struct {
  Sprite sprite;                //Sprite do pássaro
  bool in_game;                 //Se o pássaro está no cenário ou não
  bool special;                 //Se é um pássaro especial ou não
  int frame;                    //Qual frame deve-se desenhar o pássaro
} Bird;                         //Estrutura Pássaro

typedef struct {
  Sprite sprite;                //Sprite do cristal
  bool in_game;                 //Se o cristal está no cenário ou não
  bool special;                 //Se é um cristal especial (vermelho) ou não
} Crystal;                      //Estrutura Cristal

typedef struct {
  Sprite sprite;                //Sprite do efeito de brilho
  int frame;                    //Qual frame está o efeito de brilho
} Shine_Effect;                 //Estrutura Efeito de Brilho
```

character.c - Ficheiro que contém tudo relacionado com o personagem, incluindo desenha-lo no ecrã como base no seu status.
Peso: 6% *Autor: Nunes/Rosemback*

Principais Estruturas de character.h:

```
typedef enum {
  RUNNING,                  //Status correndo
  JUMPING,                  //Status pulando
  DYING                     //Status morrendo
} CharacterStatus;          //Status do Personagem

typedef struct {
  CharacterStatus status;   //Status do personagem
  int frame;                //Qual o frame em que o personagem está
  Sprite* sprite;           //Sprite do personagem
} Character;                //Estrutura Personagem
```

mushroom.c - Ficheiro que contém tudo relacionado com o desenho e geração de cogumelos durante o jogo.
Peso: 8% *Autor: Victor Nunes*

Principais Estruturas de mushroom.h:

```
typedef struct {
  Sprite sprite;            //Sprite do cogumelo
  bool in_game;             //Se o cogumelo está no cenário ou não
} Mushroom;                 //Estrutura Cogumelo
```

scenario.c – Ficheiro que trata do desenho do cenário durante o jogo.
Peso: 3% *Autor: Nunes/Rosemback*

scoreboard.c – Ficheiro que trata da leitura e escrita de um ficheiro de texto relacionado com as pontuações previamente registradas, assim como a impressão no ecrã do scoreboard das 5 melhores pontuações registradas.
Peso: 4% *Autor: Victor Nunes*

letters_and_numbers.c – Ficheiro que trata de desenhar no ecrã palavras ou números de uma maneira centralizada. É muito útil no momento que o jogador está a escrever seu nome na tela de registro e

para desenhar a pontuação atualizada no canto da tela. Em suma, muito útil para desenhar na tela palavras e números que podem variar.
Peso: 5% *Autor: Victor Nunes*

utils.c – Ficheiro que contém a implementação da função util_sys_inb e entre outras.
Peso: 1% *Autor: Carolina Rosemback*

3.2. Código Externo

Em relação a código externo, este projeto apenas usou uma simples função chamada char* itoa(int val, int base), que foi alterada para servir apenas para 10, e transformar inteiros em string. Ela está implementada em scoreboard.c linha 51 .
Fonte: https://www.strudel.org.uk/itoa/

4. Detalhes da Implementação

Nesta secção vamos descrever com mais detalhes como se foi dada a implementação do jogo.

O coração do jogo está propriamente nos ficheiros game.c e events.c, pois usamos principalmente um sistema de eventos e máquina de estados. As interrupções recebidas no ficheiro game.c geram eventos que são tratados no ficheiro events.c. Estes eventos podem gerar consequências diferentes dependendo do estado em que o jogo se encontra, utilizando uma espécie de máquina de estados. É através disso que o jogo flui entre os seus diversos estados.

A geração de frames também se encontra aí. Os eventos do timer geram as ações de desenhar as sprites necessárias para construir o frame, usando para isso, o estado com o qual o jogo se encontra. Os eventos do timer são gerados 30 vezes por segundo, sendo esse, consequentemente, o frame rate do jogo.

As partes que mais requereram certa engenhosidade foram a geração aleatória de cogumelos e pássaros com cristais e sua impressão na tela, assim como a animação do brilho na tela. Para isso, utilizamos

atributos dentro da estrutura desses objetos para indicar se eles estão no jogo (in_game) e em qual frame tal objeto se encontra, para assim imprimir a sprite correta no ecrã.

A utilização do cursor no projeto também requereu umas mudanças importantes e adaptações para tornar mais objetivas as funções do lab4. Com essas mudanças, nem houve a necessidade de utilizar a estrutura Packet.

Uma parte interessante que requereu um pouco de engenhosidade foram as funções dentro de letters_and_numbers.c. Não foi muito trivial a implementação das funções de imprimir na tela em forma de sprites palavras e números arbitrários que eram dados como argumentos (funções **void draw_word_centralized(char* word, int x_initial, int x_final, int y)** e **void draw_number_centralized(int number_points, int x_initial, int x_final, int y)**). Mas creio que foi possível implementá-las bem e de uma maneira bem mais simples e elegante do que a versão inicial.

O mecanismo de detecção de colisão também foi algo interessante de ser implementado, sendo sua função averiguar se há alguma sobreposição de pixel entre as duas sprites que ele está comparando (ignorando, claro, os pixels que representam o CHROMA_KEY).

5. Conclusões e Avaliação da UC

Com a conclusão deste projeto, posso dizer que a experiência com esta UC foi positiva. Este projeto, apesar de trabalhoso, foi muito agradável de realizar e o projeto mais divertido que experimentamos na faculdade até agora.

No que consta a críticas, está UC é realmente diferente das outras, e infelizmente é muito fácil acabar ficando para trás e não acompanhar o andamento da matéria. Isto aconteceu no lab2 e no lab5.

De fato, este foi um ano bem atípico, podendo levar a um menor desempenho naturalmente. Mas dedicar um pouco da aula teórica para uma parte mais prática dos conceitos seria de grande ajuda. Isto acabou fazendo falta principalmente na parte de UART.

Mas como uma avaliação geral, foi uma cadeira interessante de se fazer e dá uma boa noção ao aluno de como interagem dos dispositivos de E/S com o computador.

6. Créditos das Imagens Utilizadas

-Personagem (Dave): https://opengameart.org/content/platform-character
-Cenário: https://raventale.itch.io/parallax-background
-Cogumelos: https://raventale.itch.io/plants-minipack-for-nature-basic-tile-set
-Regular Bird: https://opengameart.org/content/bevouliin-free-bee-flappy-bird-sprite-sheets
-Special Bird: https://opengameart.org/content/bevouliin-free-flappy-chicken
-Cristais: https://raventale.itch.io/loot-assets-for-nature-tile-set
-Efeito de Brilho: https://opengameart.org/content/fireworks-effect-spritesheet
-Medalha HighScore: https://www.dreamstime.com/new-high-score-golden-game-icon-gaming-decoration-element-shining-letters-decorative-star-idea-red-coloured-emblem-image147203181