Autores: Eduardo Brito (up201806271) e Rita Silva (up201806527)

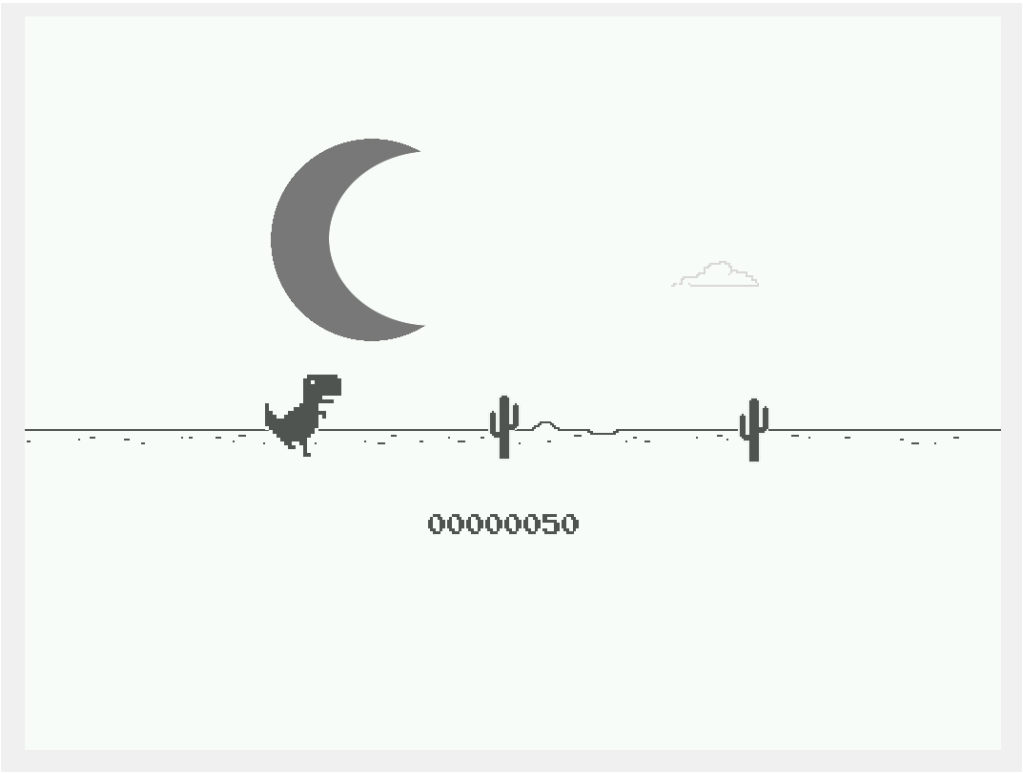
LCOM - PROJECT SPECIFICATION

Inspirado no conceito de jogos de arcada clássicos, o nosso projeto irá replicar o famoso jogo do Chrome Browser e trazer para o ambiente Minix a possibilidade de controlar um T-REX, no meio de um árido deserto, numa corrida interminável contra o tempo. O jogador necessitará de uma grande destreza para se desviar de todos os obstáculos que vão surgindo, enquanto tenta chegar o mais longe possível e salvar-se da extinção.

ESTRUTURA

O Jogo, estruturalmente, está organizado por módulos, contendo, nomeadamente, um Splash Screen que apresenta, de seguida, o Menu Inicial, onde tem as múltiplas opções, escolhidas não só com os botões e posição do Mouse, mas também com as teclas do Keyboard (P – Play , S – Scores, M – Multiplayer, Esc – Exit):

- Singleplayer (Mode) : Modo jogável para um único utilizador, onde o dinossauro é controlado pelo jogador, com algumas teclas do Keyboard que desencadeiam várias ações, saltando (Espaço) ou encolhendo-se (Seta para Baixo), permitindo à personagem esquivar-se dos obstáculos e continuar viva por mais tempo.



- Multiplayer (Mode) : Modo para dois utilizadores, que competirão pelo tempo mais longo de vivência no jogo.
- Scores : Painel de Recordes que indicará os três primeiros tempos mais longos de vivência em jogo, contendo a data real de obtenção do recorde.
- Exit.

DISPOSITIVOS

- Timer : Permite as animações, controlo do tempo jogado, tempos de espera, atualização do ecrã, movimentação dos obstáculos e outras funcionalidades.

- Keyboard : Permite controlar a personagem e as escolhas dos menus através de várias teclas.

- Mouse : Permite navegar e escolher as opções nos Menus, através dos botões e da posição do ponteiro no ecrã.

- Video : Permite toda a interface gráfica do jogo. O modo de vídeo usado foi o 0x117, com resolução 1024x768 e 64 mil (5:6:5) cores.

Foi implementado o conceito de double buffering para gerir as atualizações da interface gráfica do jogo e ainda um terceiro buffer para detetar as colisões das entidades. Todas as animações são geradas com o cálculo de novas posições, ou novas assets, através das interrupções do timer e cada estado da nossa máquina possui uma função que serve exatamente o propósito de calcular todas estas alterações e desenhar no ecrã a interface respetiva. Convém notar que, para isto, foi usada uma biblioteca externa de leitura e representação de bitmaps, que sofreu várias alterações para ser possível a adaptação ao nosso projeto (nomeadamente, a ligação necessária à API desenvolvida no lab5).

- RTC : Através da técnica de pooling, permite apresentar a data real dos recordes e ainda determinar a posição do Sol ou da Lua com base na hora do dia.
- UART : ~~Permite implementar o modo Multiplayer~~. (Não foi implementado)

Dispositivo	Interrupções
Timer	Sim
KBD	Sim
Mouse	Sim
Video Card	Não
RTC	Não
Serial Port	Não

ESTRUTURAS DE DADOS

Além das API desenvolvidas nos labs, o nosso programa contém ainda as implementações necessárias à concretização visual e funcional do jogo.

```
// Possible States for the State Machine
typedef enum {MENU = 0,SINGLEP,MULTIP,SCORES,EXIT,SPLASHSCREEN} State;

// Possible Events for the State Machine
typedef enum {NOTH = 0, SPBTN, MPBTN, SCBTN,EXBTN} Event;

// Functions related to the current State of the State Machine
int stMENU();
int stSINGLEP();
int stMULTIP();
int stSCORES();
int stEXIT();

// State Machine
typedef struct {
  State state;
  Event event;
  int xi,yi; // Mouse Coordinates
  Bitmap * mouse; // Mouse Pointer
  int (*actions[5])();
}MainMenuStM;
```

Destacamos a existência de um único loop de gerenciamento das interrupções dos vários dispositivos usados e a estruturação baseada numa simples máquina de estados. A cada estado é associado um Handler para cada dispositivo e ainda uma função geradora da interface gráfica que atualiza o ecrã a cada interrupção do timer. Cada módulo contém também as estruturas necessárias à agregação da informação, onde se inclui todos os bitmaps usados.

```
/**
 * @brief Game Main Loop
 *
 * @return 0 if OK, 1 if NOT_OK
*/
int mainLoop();

/**
 * @brief Loads all the necessary images and initial data
*/
void LoadBitmaps();

/**
 * @brief Event Handler for Mouse on Main Menu
 *
 * @param p Mouse packet to be parsed
*/
void MenuMouseEventHandler(struct packet p);

/**
 * @brief Event Handler for Keyboard on Main Menu
*/
void MenuKbdEventHandler();

/**
 * @brief Main Menu Graphic Interface
*/
void MenuDrawInterface();
```

```
/**
 * @brief Event Handler for Keyboard on Play State
 */
void PlayKbdEventHandler();

/**
 * @brief Play State Graphic Interface
 */
int PlayDrawInterface();
```

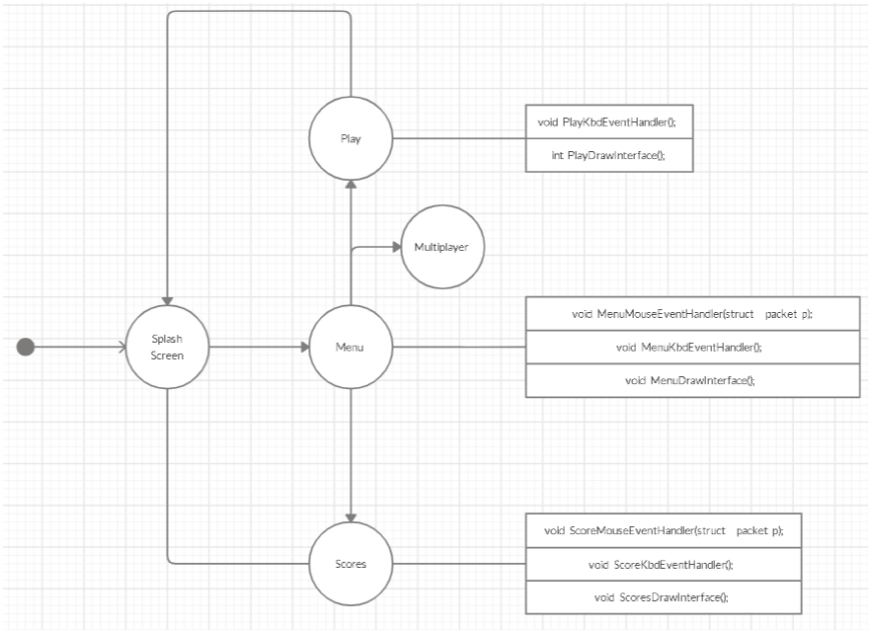
```
/**
 * @brief Event Handler for Mouse on Score
 *
 * @param p Mouse packet to be parsed
*/
void ScoreMouseEventHandler(struct packet p);

/**
 * @brief Event Handler for Keyboard on Score State
*/
void ScoreKbdEventHandler();

/**
 * @brief Draw Digits on Screen
 *
 * @param value to be parsed and drawn on screen
 * @param h X position for the first digit
 * @param v Y position for the first digit
 * @param n number of digits
*/
void drawDigits(int value,int h, int v, int n);

/**
 * @brief Scores Graphic Interface
*/
void ScoresDrawInterface();
```

STATE DIAGRAM



PROJECT DIRECTORY

No nosso diretório principal existem vários subdiretórios, entre eles, “src” que contém o código do projeto, “res” que contém os recursos/assets do jogo, “gimp” onde estão os ficheiros gerados pelo GIMP e “doc” onde está toda a documentação e diagramas respetivos. Para compilar e correr o nosso programa, a partir do diretório principal do projeto, basta inserir o comando “./exe.sh” na linha de comandos do Minix.