APO Snake Documentation

Jiří Povolný, povolji2@fel.cvut.cz
Marian Krotil, krotima1@fel.cvut.cz

June 13, 2020

1 Introduction

Regarding the semestral work in APO we've chosen Snake video game, the first inspiration example from APO CourseWare Wiki. This project implemented in *C/C++* represents Snake video game on MicroZed board. In the game there are two snakes (player or AI controlled) substituted by line which grows in length if the snake eats food. The game ends when any of the snakes collides with the screen border, other snake, or itself.

2 Installation

First of all you have to download semcode folder from our git repository or from BRUTE. Then if you are connected on postel.felk.cvut.cz you can compile the whole program by using the present Makefile (the same as the mzapo_template Makefile with added source libraries). The default MicroZed board chosen in Makefile has following IP: `192.168.202.213`, which should be the seventh board from the left on the YouTube stream. There are also 7 shell scripts: `semcode‣X.sh` where X corresponds to the Xth board from the left on stream. If nothing goes wrong, board should turn on and show loading screen. Next instructions are described in Manual section.

3 Manual

3.1 Loading screen

To skip loading screen press F.

3.2 Menu

After that menu will open. The menu is controlled by keyboard. To move up press W, to move down press S, to select highlighted button press F, to

exit the game press [X]. Menu contains four buttons and the selected button is highlighted. [DEMO] Button starts Demo mode (AI vs AI), [STANDARD] button starts Standard mode (Player vs AI), [OPTIONS] button opens game options and [QUIT] button exits the game. Only two buttons are shown simultaneously on the display, to get to the next ones you have to scroll down using the [S] key.

3.3 Options menu

To access the options, select [OPTIONS] button in Menu. The control is the same as in the Menu. To move up press [W], to move down press [S], to select highlighted button press [F], to return to the menu press [X]. Options include five buttons with specific functionality. [SPEED] Button sets value of the snake speed, the number of turned on LEDs in LED line corresponds to the chosen speed value. Speed allows setting from 1 to 5 which corresponds to 1, 0.5, 0.33, 0.25 and 0.2 in seconds per move. [SIZE] Button sets size of the game tiles in pixels^2, values are 16, 20 and 32. The size of the tile will also affect the maximum width and height of the game board. [COLOR 1] Button and [COLOR 2] button set the colours of first and second snake respectively. [BACK] Button returns back to the menu.

3.4 Game modes

The game offers two modes, Demo mode and Standard mode. In both modes, snakes are spawned close to the upper left and lower right corners of the game board. At the start of the game food is generated in the middle of the game board then it is generated randomly. The right side of the screen shows elapsed time (middle portion of the screen) and score of the snakes according to the snake color (upper and lower portions of the screen). If snake is alive and moving his respective LED diode is green. When the snake eats food his respective LED diode will turn blue. If a snake dies his respective LED diode turns red and game over screen appears.

3.4.1 Demo mode

To access demo mode, select [DEMO] in Menu. Demo mode creates two snakes with simple AI and lets them play. Demo mode ends if any snake dies or if [X] is pressed.

3.4.2 Standard mode

To access demo mode, select [STANDARD] in Menu. Standard mode creates two snakes, one with AI and other is controlled by player. Player controls the first snake by pressing keyboard keys, To move up press [W], to move down press [S], to move left press [A], to move right press [D]. This mode ends if any snake dies or if [X] is pressed.

3.5 Game over

This occurs when snake dies. The score of both snakes is drawn on the screen. To return back to Menu, press X.

3.6 Credits

Credits screen is showed when the whole game ends. On the screen there are names of authors. To skip the credits, press X.

4 Programming structure

All functions and libraries are described in their respective headers.

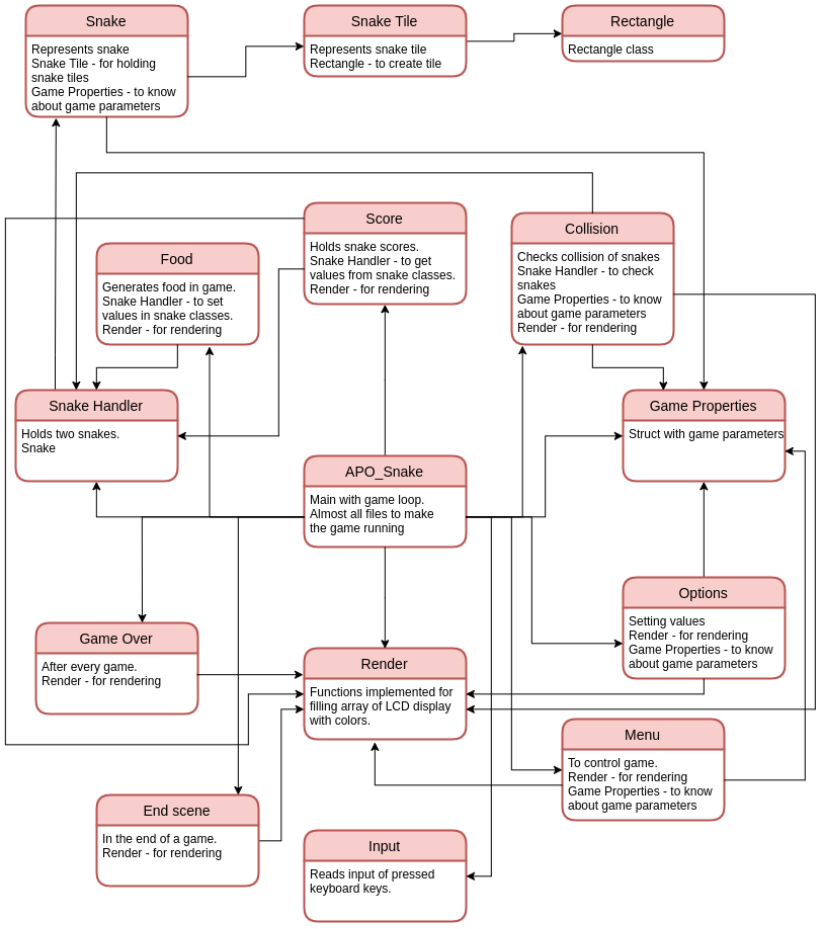

Figure 1: Object structure

4.1 APO_Snake

```
1 int main(int argc, char* argv[]);
2 void update_demo_mode(Score* score, Game_Properties
    game_properties);
3 void input();
4 void update_standard_mode(Score* score, Game_Properties
    game_properties);
5 void update_menu_option(Game_Menu* menu, Menu_Options*
    menu_options);
6 void update_game_over(Score* score);
7 void update_end_scene();
8 void loading();
```

4.2 Collision

```
1 bool collision_update(Snake_Handler handler,
    Game_Properties gm, unsigned char* mem_base);
```

4.3 Food

```
1 class Food
2     Food(Snake_Handler* handler2, int x, int y, int
        border2, int color2, int max_x2, int max_y2, int
        tile_size2)
3     void start(int, int);
4     void add_new_food();
5     bool was_food_eaten(unsigned char* mem_base);
6     void update(unsigned char* mem_base);
7     void fill_array(uint16_t* fb, int LCD_width);
```

4.4 Menu

```
1 class  Game_Menu
2     Game_Menu();
3     int menu_selection(char key_pressed);
4     void menu_fill_array(uint16_t* fb, unsigned char*
        parlcd_mem_base, Game_Properties gp);
```

4.5 Rectangle

```
1 class Rectangle
2     Rectangle(int startX, int startY, int width, int
          height);
3     bool intersects(Rectangle *rect);
```

4.6 Render

```
1  int char_width(font_descriptor_t* fdes, int ch);
2  void draw_pixel(int xTile, int yTile, uint16_t colour,
       uint16_t* pixels);
3  void draw_char(int xTile, int yTile, font_descriptor_t*
       fdes, int ch, uint16_t colour, uint16_t* pixels);
4  void draw_string(int xTile, int yTile, uint16_t colour,
       char* ch, int size_of_string, uint16_t* pixels);
5  void draw_rect(int x, int y, int width, int height, int
       border, uint16_t colour, uint16_t* pixels);
6  void draw_sinus(int x, int y, int size, uint16_t colour,
       uint16_t* pixels);
7  void draw_lineX(int xTile, int yTile, int size, uint16_t
       colour, uint16_t* pixels);
8  void draw_lineY(int xTile, int yTile, int size, uint16_t
       colour, uint16_t* pixels);
9  void draw_filled_rect(int x, int y, int width, int height
       , uint16_t colour, uint16_t* pixels);
10 void clear_array(uint16_t colour, uint16_t* pixels);
11 void draw(unsigned char* parlcd_mem_base, uint16_t*
       pixels);
12 void led_line(uint32_t val_line, unsigned char* mem_base)
       ;
13 void led_RGB1(uint32_t colour, unsigned char* mem_base);
14 void led_RGB2(uint32_t colour, unsigned char* mem_base);
```

4.7 Snake

```
1 class Snake
2     Snake(int x, int y, uint16_t new_color,
          Game_Properties gp, bool is_player);
3     bool is_new_direction_correct(Snake_Tile::Direction
          new_dir, Snake_Tile::Direction prev_dir);
4     Snake_Tile::Direction set_direction();
5     void move_tiles();
```

```
6       Snake_Tile::Direction choose_move2();
7       void set_moves();
8       bool is_tile_occupied(int x, int y);
9       void set_opposite_snake(Snake* op_snake);
10      void fill_array(uint16_t* fb, int LCD_width);
11      void update();
12      std::vector<Snake_Tile> get_snake_tiles();
13      Snake_Tile get_tile(int pos);
14      void add_snake_tile();
15      size_t get_size();
16      void set_position_of_food(int new_food_x, int
            new_food_y);
17      void delete_snake_tiles();
18      void set_direction(Snake_Tile::Direction dir);
19      uint16_t get_color();
```

4.8 Snake_Handler

```
1   class Snake_Handler
2       Snake_Handler();
3       void delete_snakes();
4       void add_snake(Snake* snake);
5       void update();
6       void fill_array(uint16_t* fb, int LCD_width);
```

4.9 Snake_Tile

```
1   class Snake_Tile
2       Snake_Tile();
3       Snake_Tile(int start_x, int start_y);
4       int get_x();
5       int get_y();
6       void set_x(int new_x);
7       void set_y(int new_y);
8       Direction get_direction();
9       void set_direction(Direction new_dir);
10      const char* print_direction();
11      Rectangle* get_occupied_field(int w, int h);
12      bool intersects(Snake_Tile tile, int tile_size);
```

4.10 end_scene

```
class End_Scene
    End_Scene();
    int get_max();
    void end_scene_update();
    void end_scenefill_array(uint16_t* fb, unsigned char*
        parlcd_mem_base);
```

4.11 game_over

```
int game_over_update(char keypressed);
void game_over_fill_array(uint16_t* fb, unsigned char*
   parlcd_mem_base, int score_1, int score_2, uint16_t
   color_1, uint16_t color_2);
```

4.12 input

```
void PrepareKeyboardTtySettings();
int kbhit();
char getch();
```

4.13 options

```
class Menu_Options
    Menu_Options()
    int options_selection(char key_pressed, int& speed,
        int& size_of_tile, uint16_t& snake_1_color,
        uint16_t& snake_2_color, unsigned char* mem_base);
    void options_fill_array(uint16_t* fb, unsigned char*
        parlcd_mem_base, Game_Properties gp);
```

4.14 score

```
class Score
    Score();
    void update(Snake_Handler handler);
    void score_fill_array(uint16_t* fb, unsigned char*
        parlcd_mem_base);
    void reset();
    int get_score_1();
    int get_score_2();
```

```
8       uint16_t get_color_1();
9       uint16_t get_color_2();
```