custom data types

struct config_t (config.h)

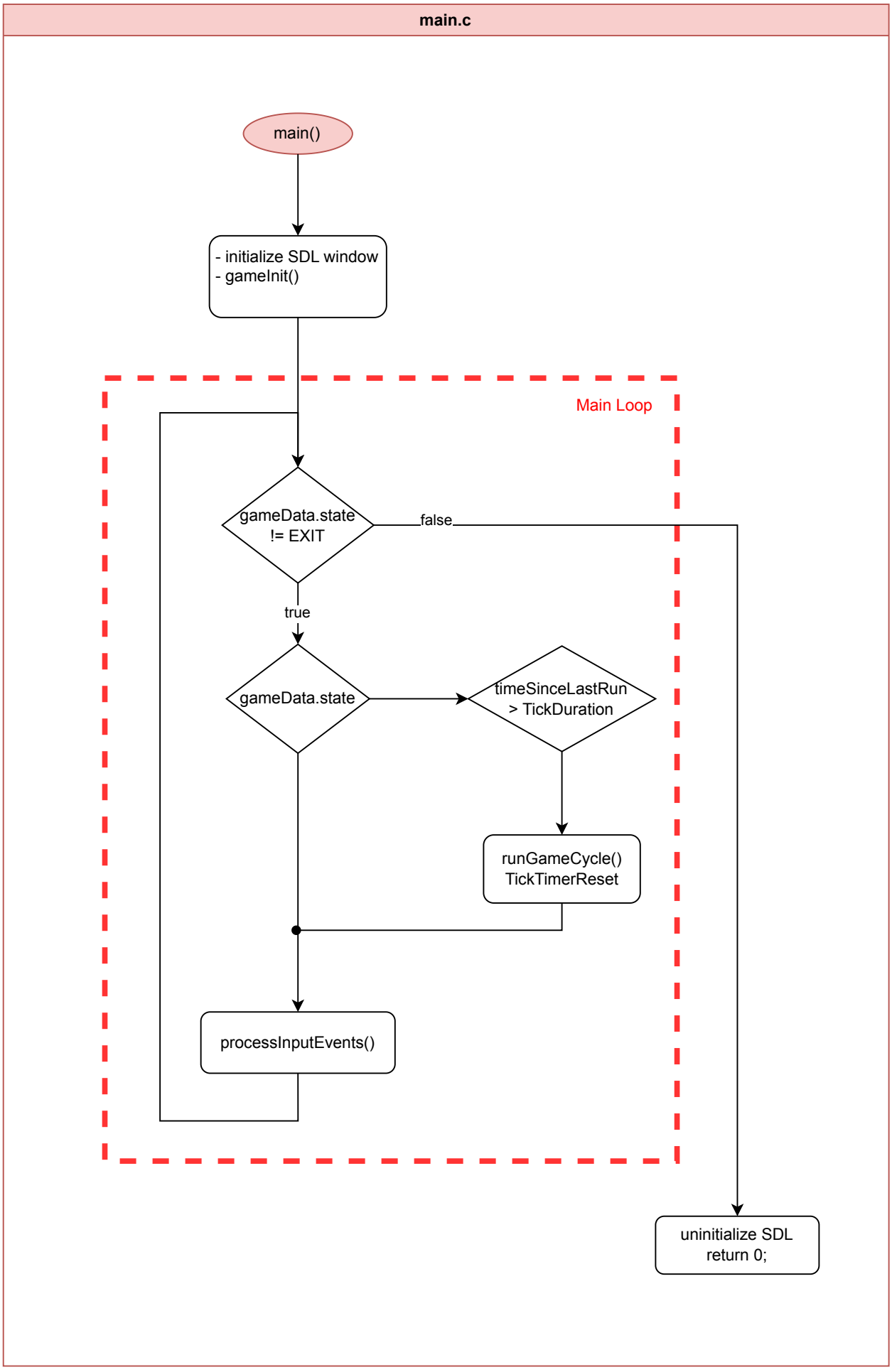
```
int mapWidth;
int mapHeight;
int blockSizePx; //pixle size of one block

int cycleDurationMs;
int difficulty; //0-3
int snakeDefaultLength;

const char * leaderboardFilename;

bool debug; //enable debug output
```

struct snake_t (snake.h)

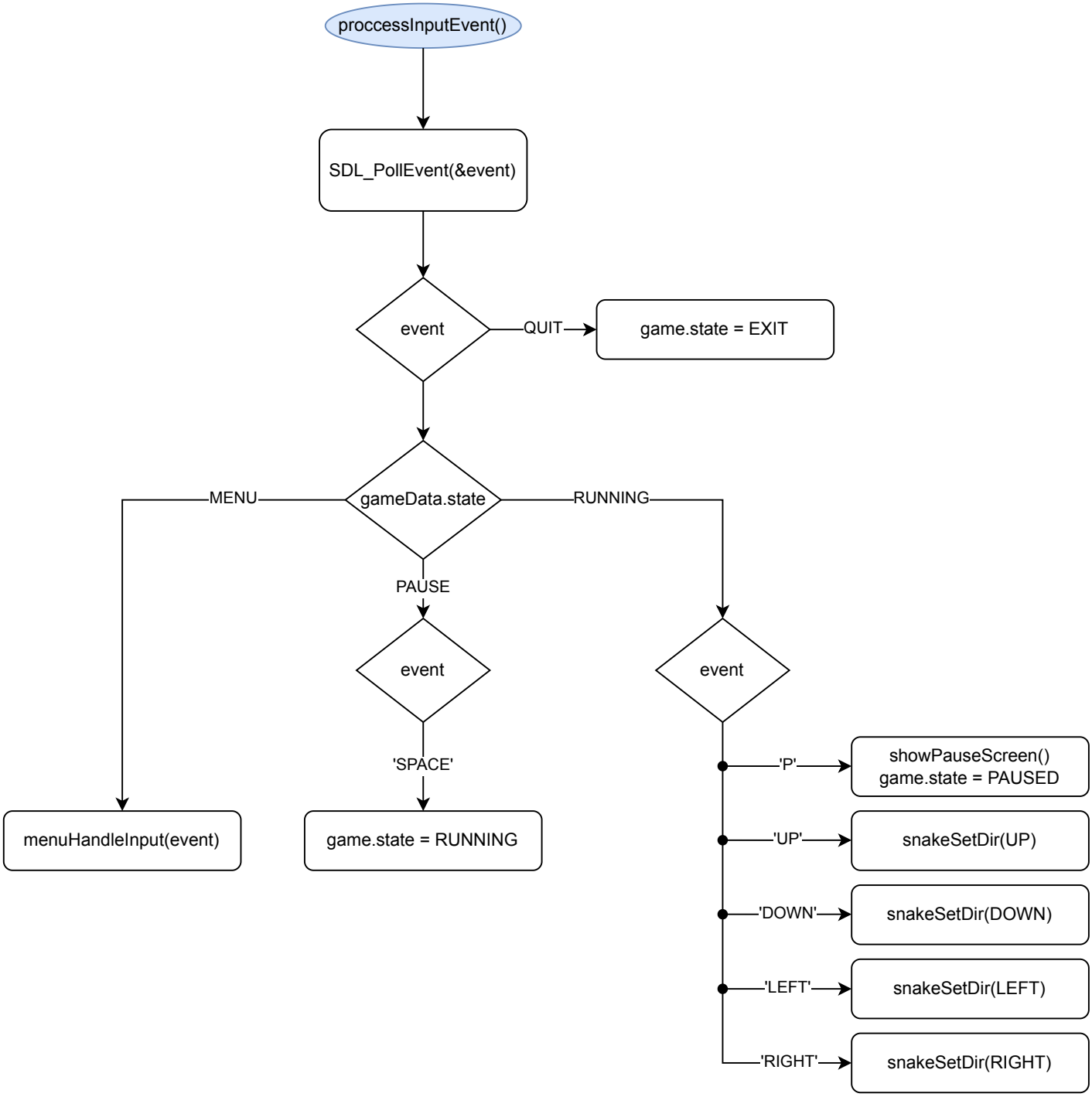
```
int length;
int headX, headY;

enum snakeDirection_t{DOWN=0, UP, LEFT, RIGHT};
snakeDirection_t direction;
int tail[512][2] = {0};

bool isAlive;
```

struct gameData_t (game.h)

```
snake_t snake;

SDL_Renderer *sdlRenderer
SDL_Window *sdlWindow

int mapCollisions[MAX_MAP_SIZE][MAX_MAP_SIZE];
int mapPortals[MAX_MAP_SIZE][MAX_MAP_SIZE];

int foodX, foodY;
int lifesRemaining;

int timestampLastCycle;
bool isPaused;

typedef enum gameState_t {PAUSED=0, MENU, RUNNING};
gameState_t gameState;
```

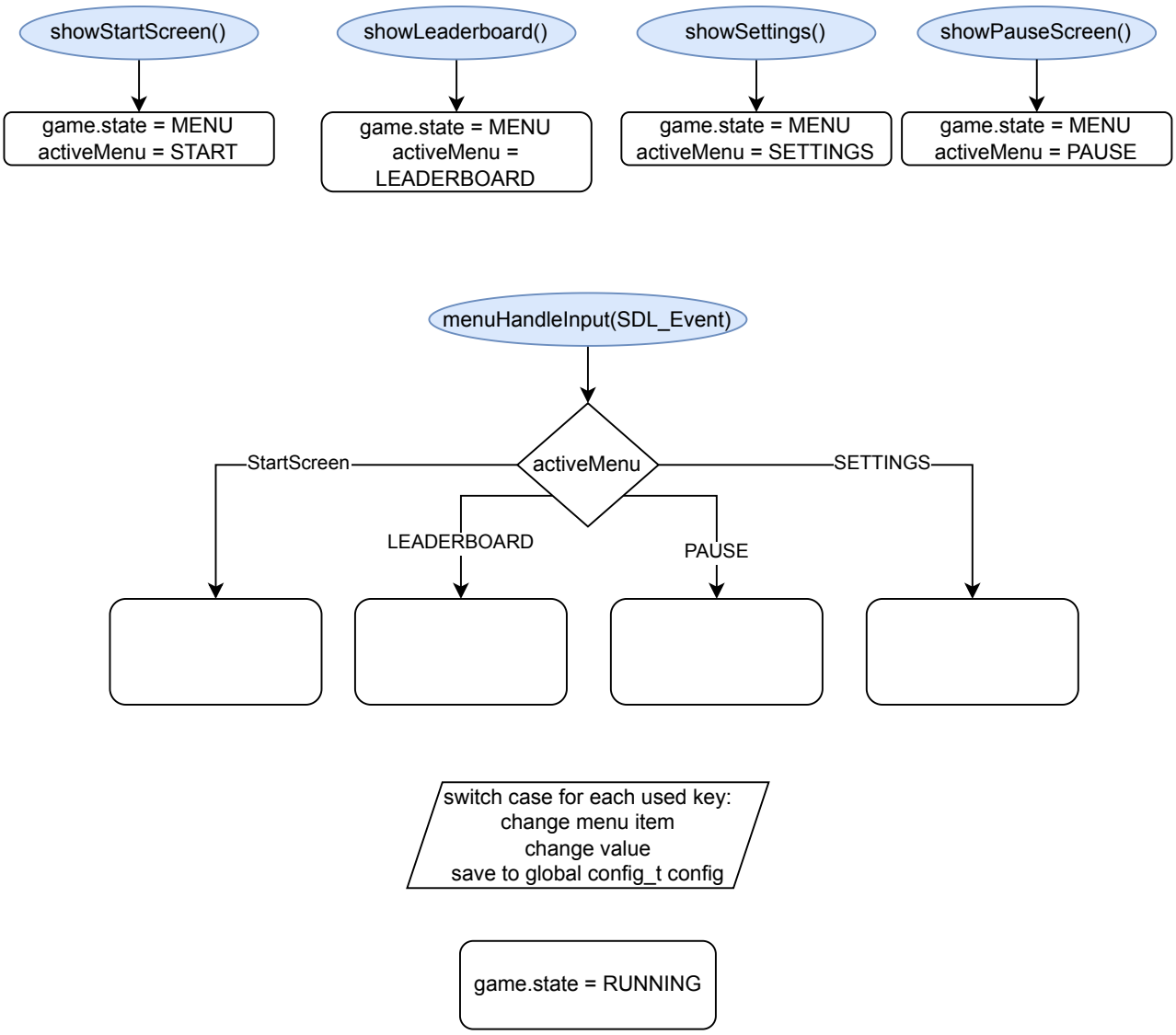

main.c
main()
- initialize SDL window
- gameInit()
Main Loop
gameData.state
!= EXIT
false
true
gameData.state
timeSinceLastRun
> TickDuration
runGameCycle()
TickTimerReset
processInputEvents()
uninitialize SDL
return 0;

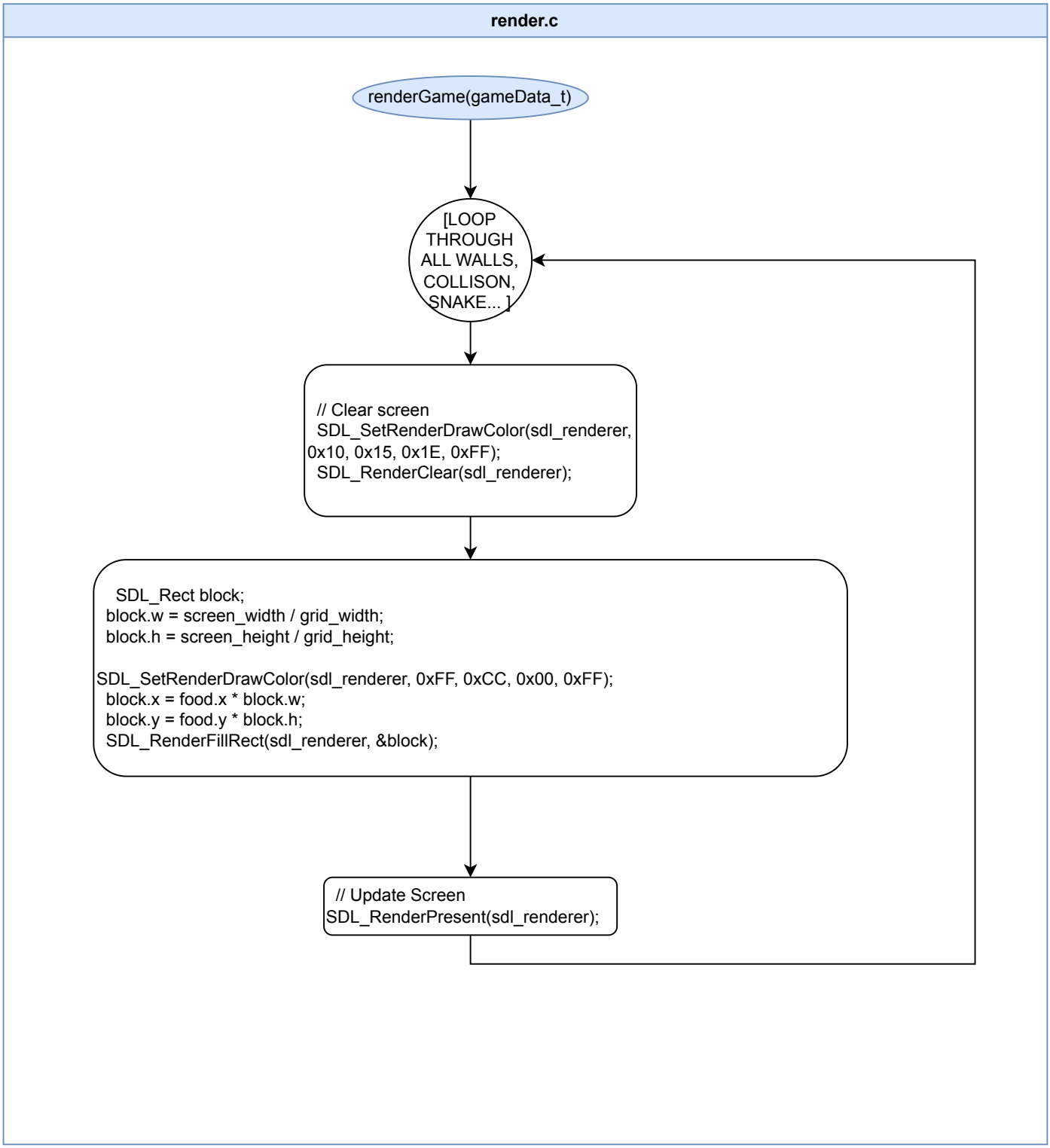
input.c


proccessInputEvent()
SDL_PollEvent(&event)
event
QUIT
game.state = EXIT
gameData.state
MENU
RUNNING
PAUSE
event
event
'SPACE'
'P'
showPauseScreen()
game.state = PAUSED
menuHandleInput(event)
game.state = RUNNING
'UP'
snakeSetDir(UP)
'DOWN'
snakeSetDir(DOWN)
'LEFT'
snakeSetDir(LEFT)
'RIGHT'
snakeSetDir(RIGHT)

menu.c

enum menus_t = {NONE=0, START, SETTINGS,
LEADERBOARD, PAUSE}
menus_t activeMenu = NONE




render.c
renderGame(gameData_t)
[LOOP THROUGH ALL WALLS, COLLISON, SNAKE...]
// Clear screen
SDL_SetRenderDrawColor(sdl_renderer, 0x10, 0x15, 0x1E, 0xFF);
SDL_RenderClear(sdl_renderer);
SDL_Rect block;
block.w = screen_width / grid_width;
block.h = screen_height / grid_height;
SDL_SetRenderDrawColor(sdl_renderer, 0xFF, 0xCC, 0x00, 0xFF);
block.x = food.x * block.w;
block.y = food.y * block.h;
SDL_RenderFillRect(sdl_renderer, &block);
// Update Screen
SDL_RenderPresent(sdl_renderer);

game.c

gameInit()

handleCollision()

handlePortals()

init snake

runGameCycle()

run all those functions:
- snake move
- check eaten
- snake grow
- place food
- handleCollision
- handlePortal

renderGame()

game.timestampLastCycle = now
//use SDL TICKS?

snake.c

snakeInit()

snakeSetDir()

snakeSetHeadPos()

snake.c

- **snakeInit**()
- **snakeGrow**()
- **snakeMove**()
- **snakeSetDir**(ENUM dir)
- **snakeSetHeadPos**(x y)
- **snakeIsAlive**()

snakeMove()

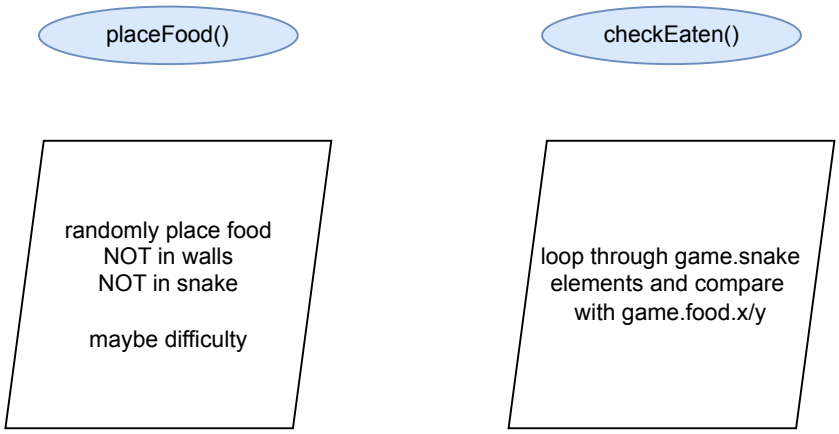
rotate array
enlarge array in current dir
remove last segment
...

snakeGrow()

size++
...

update game.snake object

food.c


placeFood()
checkEaten()
randomly place food
NOT in walls
NOT in snake
maybe difficulty
loop through game.snake
elements and compare
with game.food.x/y