Snake++: Datatypes and function flowcharts
12.12.2023

custom data types

struct config_t (config.h)

```
const int windowSize;

map_t map;
int blockSizePx //pixel size of one block calculated
    by loadMap() using windowSize and mapSize

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

const char * leaderboardFilename;
const char *defaultMapName;

bool debug;    //enable debug output
```

struct snake_t (snake.h)

```
int length;
int headX, headY;

snakeDirection_t direction;
int tail[MAX_MAP_FIELDS][2] = {0};

bool isAlive;
```

enum snakeDirection_t (snake.h)

```
enum snakeDirection_t {DOWN=0, UP, LEFT, RIGHT};
```

struct gameData_t (game.h)

```
snake_t snake;
map_t map;

SDL_Renderer *sdlRenderer;
SDL_Window *sdlWindow;

bool mapIsLoaded;

int foodX, foodY;
int lifesRemaining;

int timestampLastCycle;
bool isPaused;

gameState_t gameState;
```

enum gameState_t (game.h)

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

struct map_t (map.h)

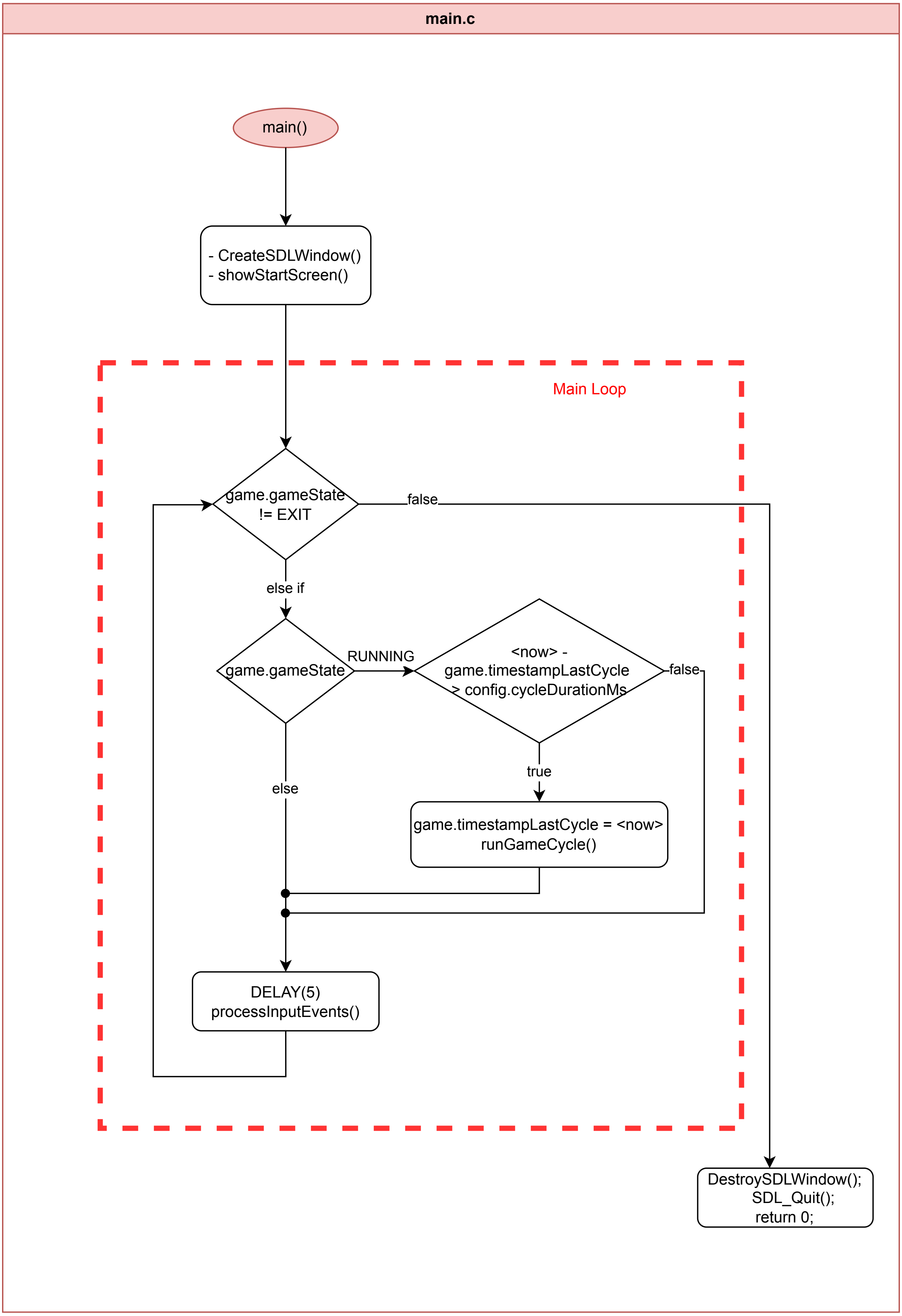
```
int width; //note: fixed 1:1 aspect ratio
int height;

const char*[128] name;
collision_t collisions[MAX_MAP_FIELDS];
int collisionCount;
portal_t mapPortals[MAX_MAP_FIELDS];
int portalCount;
```

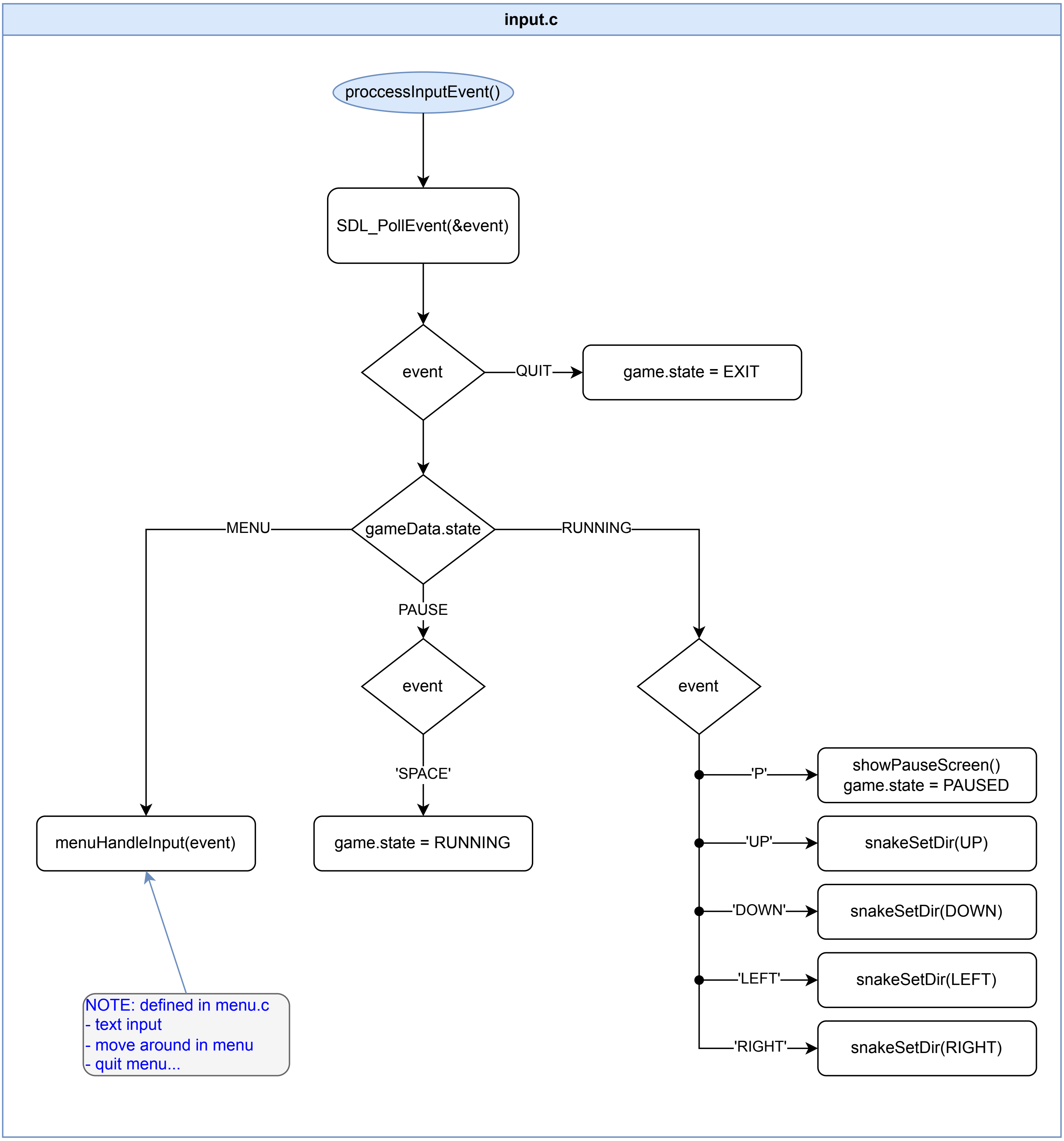
struct portal_t (map.h)

```
int posX;
int posY;

int targetX;
int targetY;

char * color;
```

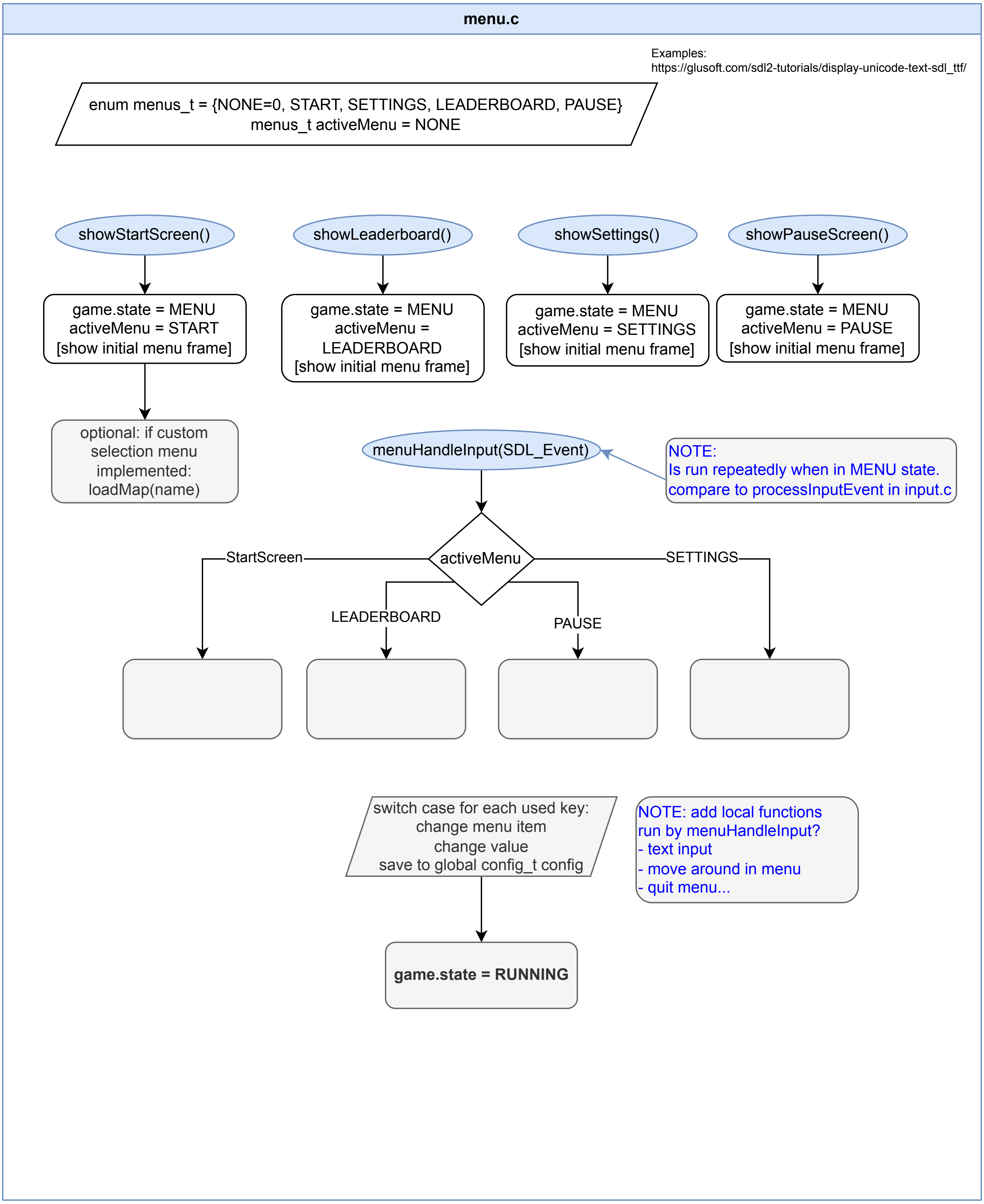
struct collision_t (map.h)

```
int posX;
int posY;
```


main.c
main()
- CreateSDLWindow()
- showStartScreen()
Main Loop
game.gameState
!= EXIT
false
else if
game.gameState
RUNNING
<now> -
game.timestampLastCycle
> config.cycleDurationMs
false
true
game.timestampLastCycle = <now>
runGameCycle()
else
DELAY(5)
processInputEvents()
DestroySDLWindow();
SDL_Quit();
return 0;



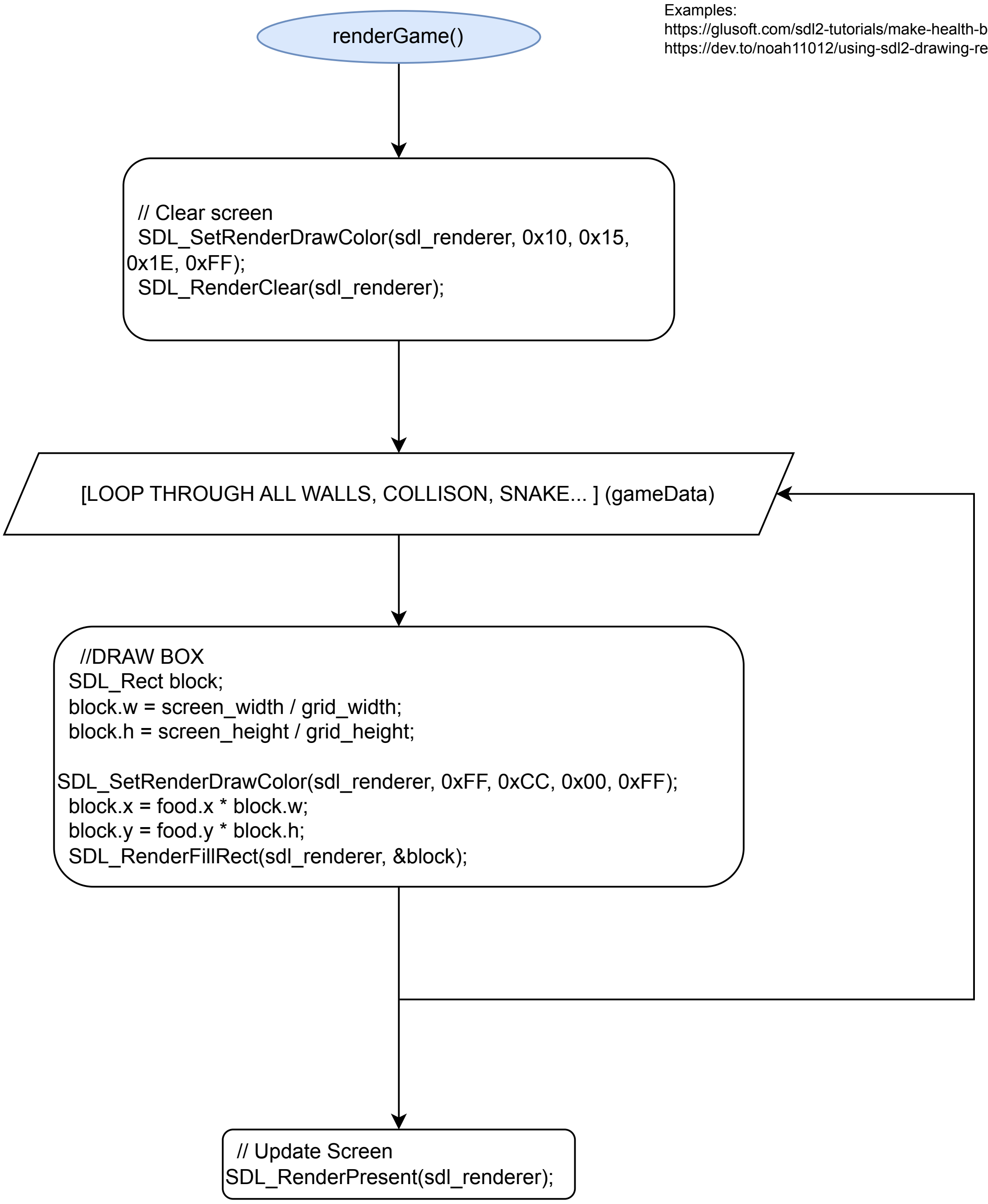
NOTE: defined in menu.c
- text input
- move around in menu
- quit menu...


menu.c
Examples:
https://glusoft.com/sdl2-tutorials/display-unicode-text-sdl_ttf/
enum menus_t = {NONE=0, START, SETTINGS, LEADERBOARD, PAUSE}
menus_t activeMenu = NONE
showStartScreen()
showLeaderboard()
showSettings()
showPauseScreen()
game.state = MENU
activeMenu = START
[show initial menu frame]
game.state = MENU
activeMenu =
LEADERBOARD
[show initial menu frame]
game.state = MENU
activeMenu = SETTINGS
[show initial menu frame]
game.state = MENU
activeMenu = PAUSE
[show initial menu frame]
optional: if custom
selection menu
implemented:
loadMap(name)
menuHandleInput(SDL_Event)
NOTE:
Is run repeatedly when in MENU state.
compare to processInputEvent in input.c
activeMenu
StartScreen
LEADERBOARD
PAUSE
SETTINGS
switch case for each used key:
change menu item
change value
save to global config_t config
NOTE: add local functions
run by menuHandleInput?
- text input
- move around in menu
- quit menu...
game.state = RUNNING

render.c

:

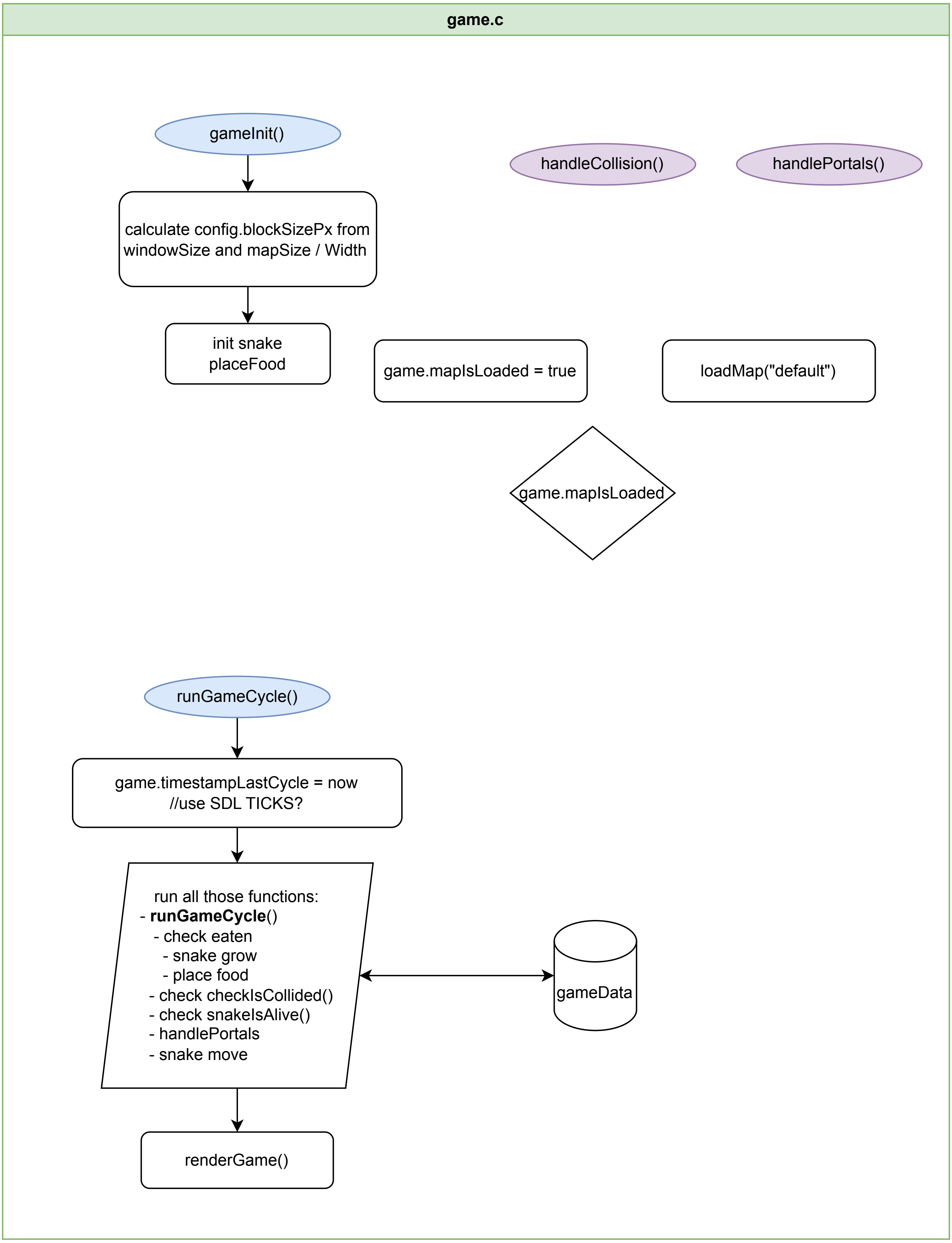
Examples:
https://glusoft.com/sdl2-tutorials/make-health-bar-sdl/
https://dev.to/noah11012/using-sdl2-drawing-rectangles-3hc2

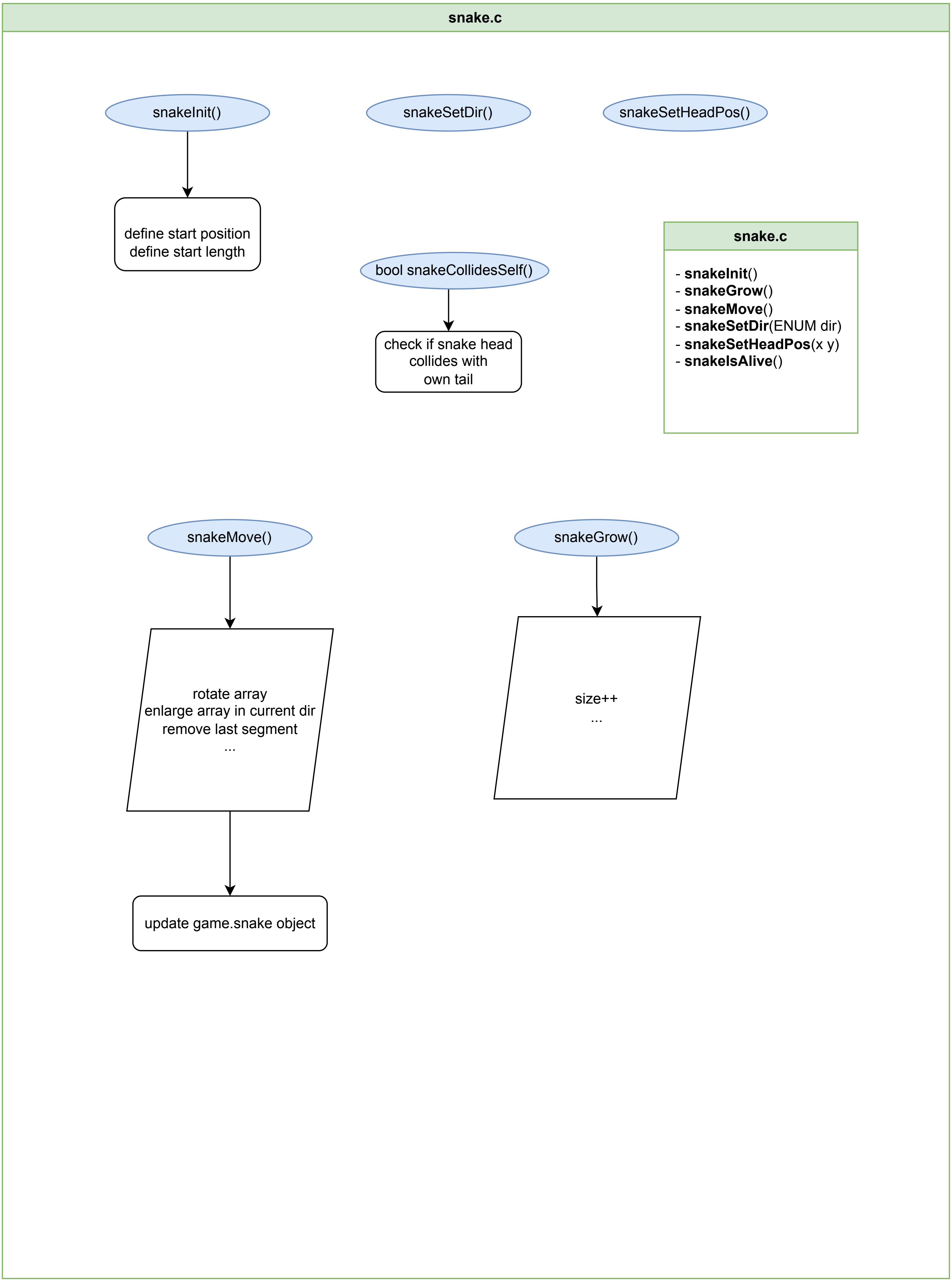


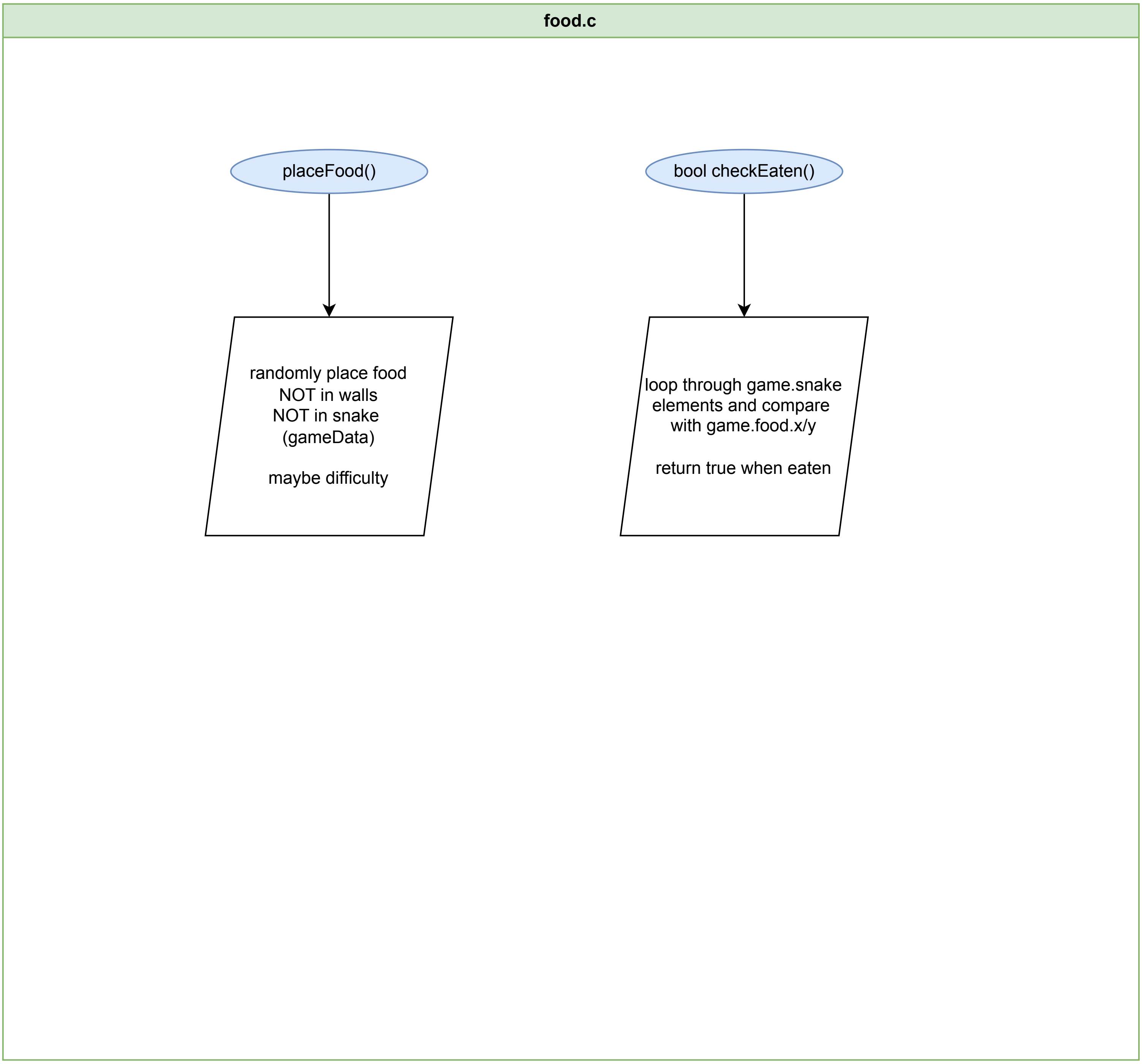
```
 // Clear screen
 SDL_SetRenderDrawColor(sdl_renderer, 0x10, 0x15,
0x1E, 0xFF);
 SDL_RenderClear(sdl_renderer);
```

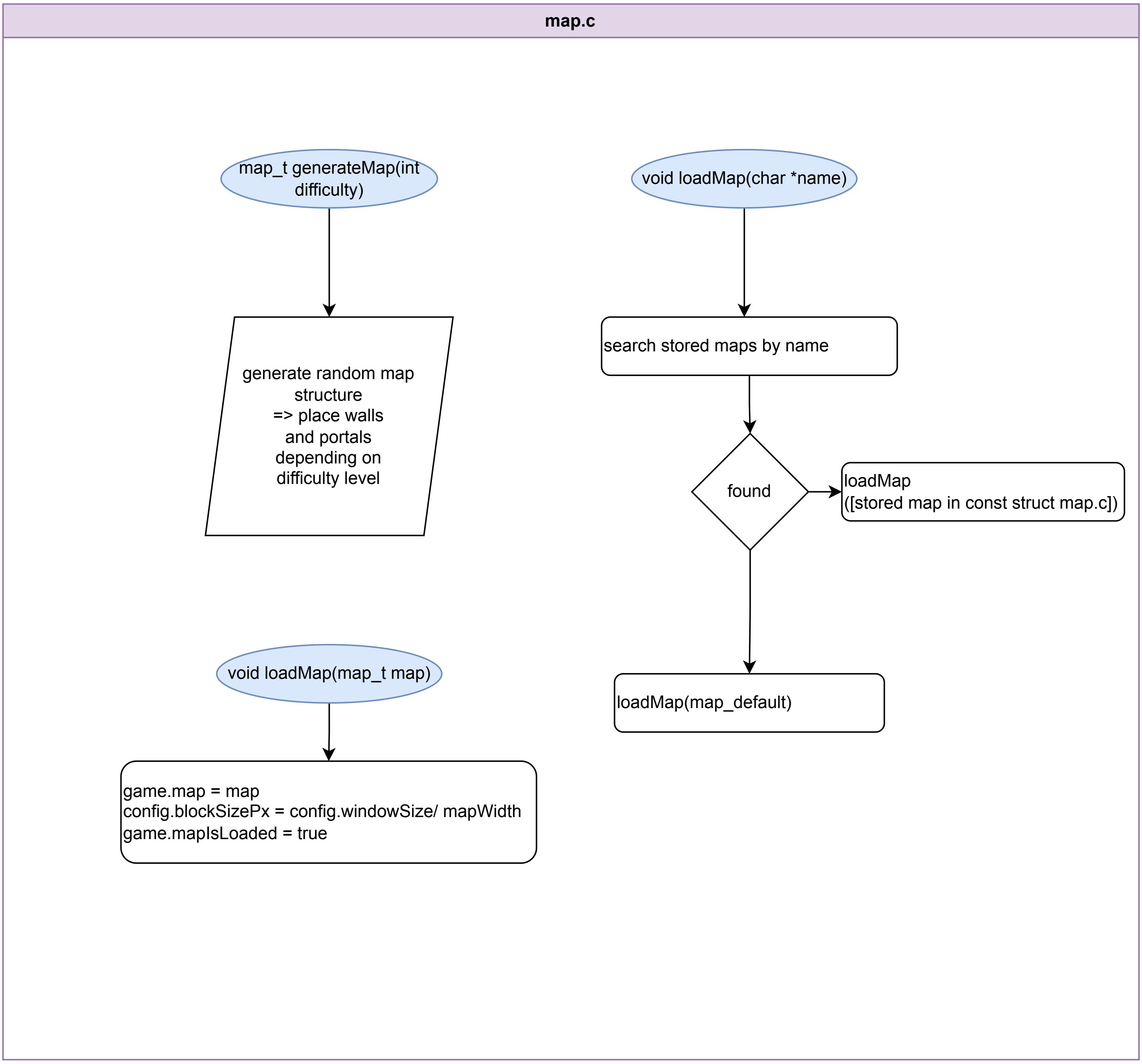
```
   //DRAW BOX
  SDL_Rect block;
  block.w = screen_width / grid_width;
  block.h = screen_height / grid_height;

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()
calculate config.blockSizePx from windowSize and mapSize / Width
init snake
placeFood
handleCollision()
handlePortals()
game.mapIsLoaded = true
loadMap("default")
game.mapIsLoaded
runGameCycle()
game.timestampLastCycle = now
//use SDL TICKS?
run all those functions:
- **runGameCycle**()
- check eaten
- snake grow
- place food
- check checkIsCollided()
- check snakeIsAlive()
- handlePortals
- snake move
gameData
renderGame()


snake.c
snakeInit()
define start position
define start length
snakeSetDir()
snakeSetHeadPos()
bool snakeCollidesSelf()
check if snake head collides with own tail
snake.c
- **snakeInit**()
- **snakeGrow**()
- **snakeMove**()
- **snakeSetDir**(ENUM dir)
- **snakeSetHeadPos**(x y)
- **snakeIsAlive**()
snakeMove()
rotate array
enlarge array in current dir
remove last segment
...
update game.snake object
snakeGrow()
size++
...


food.c
placeFood()
randomly place food
NOT in walls
NOT in snake
(gameData)
maybe difficulty
bool checkEaten()
loop through game.snake
elements and compare
with game.food.x/y
return true when eaten


map.c
map_t generateMap(int difficulty)
generate random map structure
=> place walls and portals depending on difficulty level
void loadMap(char *name)
search stored maps by name
found
loadMap
([stored map in const struct map.c])
loadMap(map_default)
void loadMap(map_t map)
game.map = map
config.blockSizePx = config.windowSize/ mapWidth
game.mapIsLoaded = true