Snake++: Datatypes and function flowcharts
10.11.2023

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;

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;

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)

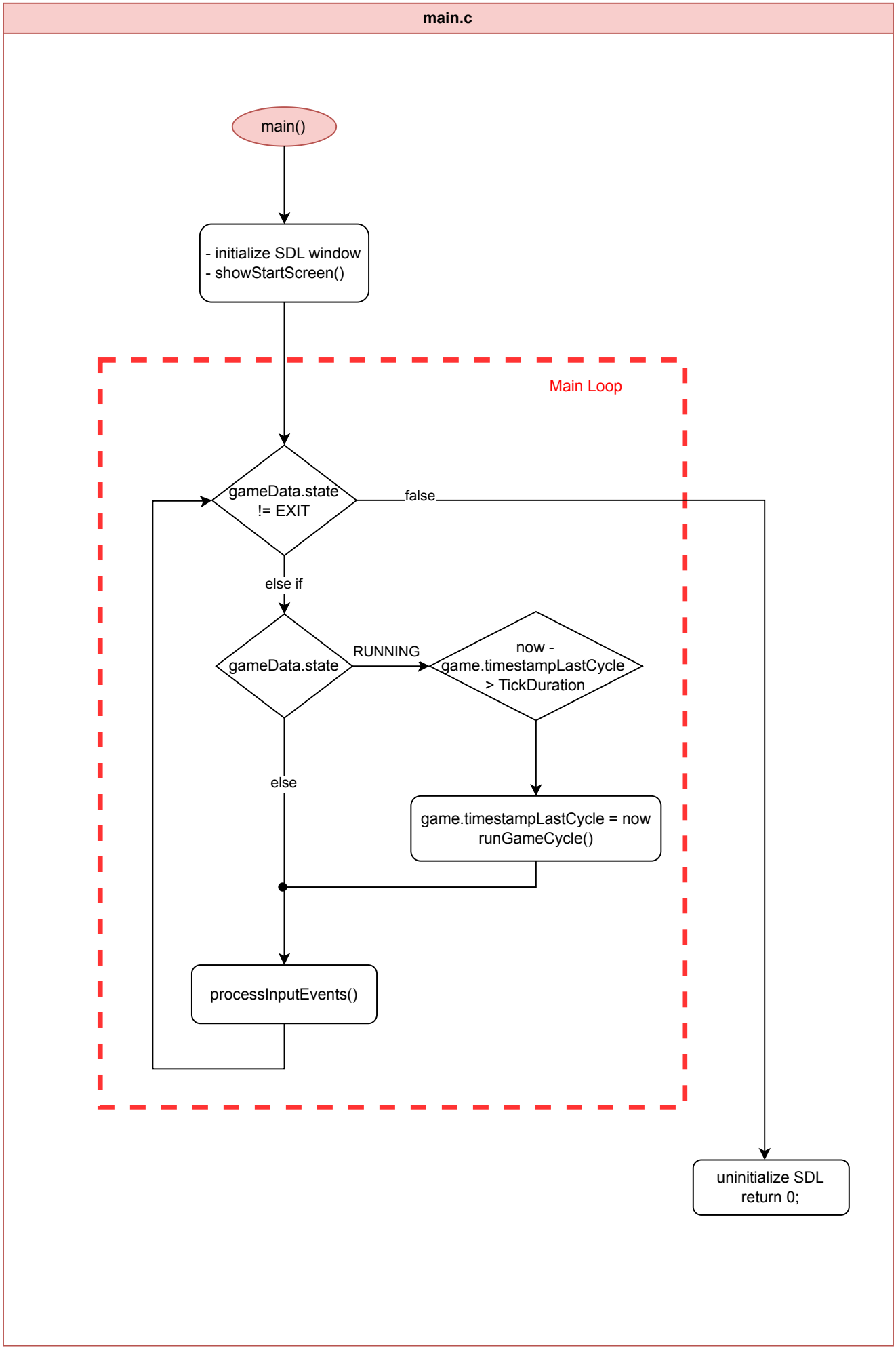
```
int posX;
int posY;

int targetX;
int targetY;

char * color;
```

struct collision_t (map.h)

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


main.c
main()
- initialize SDL window
- showStartScreen()
Main Loop
gameData.state
!= EXIT
false
else if
gameData.state
RUNNING
now -
game.timestampLastCycle
> TickDuration
else
game.timestampLastCycle = now
runGameCycle()
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

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()

game.state = MENU
activeMenu = START
[show initial menu frame]

optional: if custom selection menu implemented: loadMap(name)

showLeaderboard()

game.state = MENU
activeMenu = LEADERBOARD
[show initial menu frame]

showSettings()

game.state = MENU
activeMenu = SETTINGS
[show initial menu frame]

showPauseScreen()

game.state = MENU
activeMenu = PAUSE
[show initial menu frame]

menuHandleInput(SDL_Event)

activeMenu

- StartScreen
- LEADERBOARD
- PAUSE
- SETTINGS

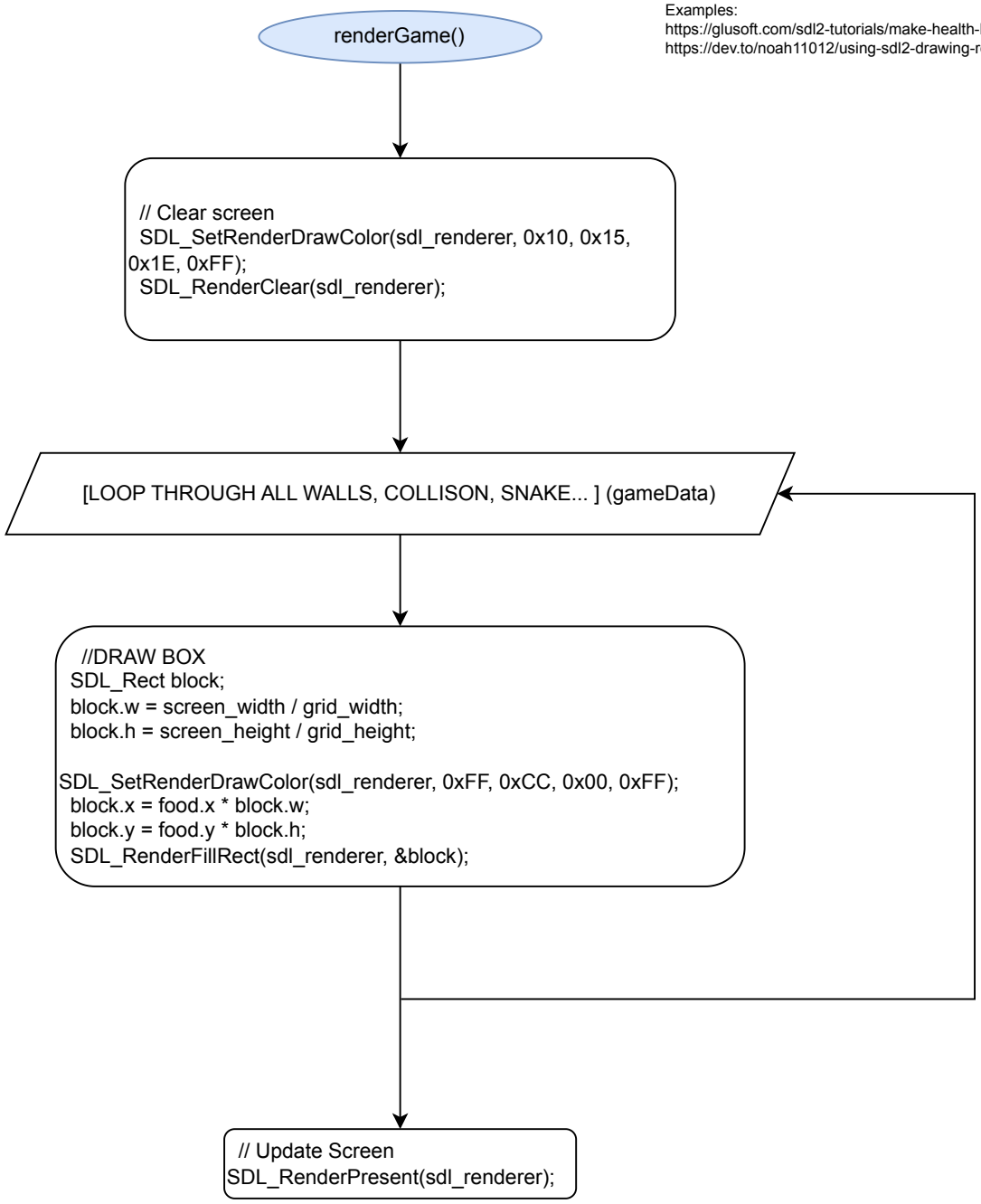
switch case for each used key:
change menu item
change value
save to global config_t config

game.state = RUNNING

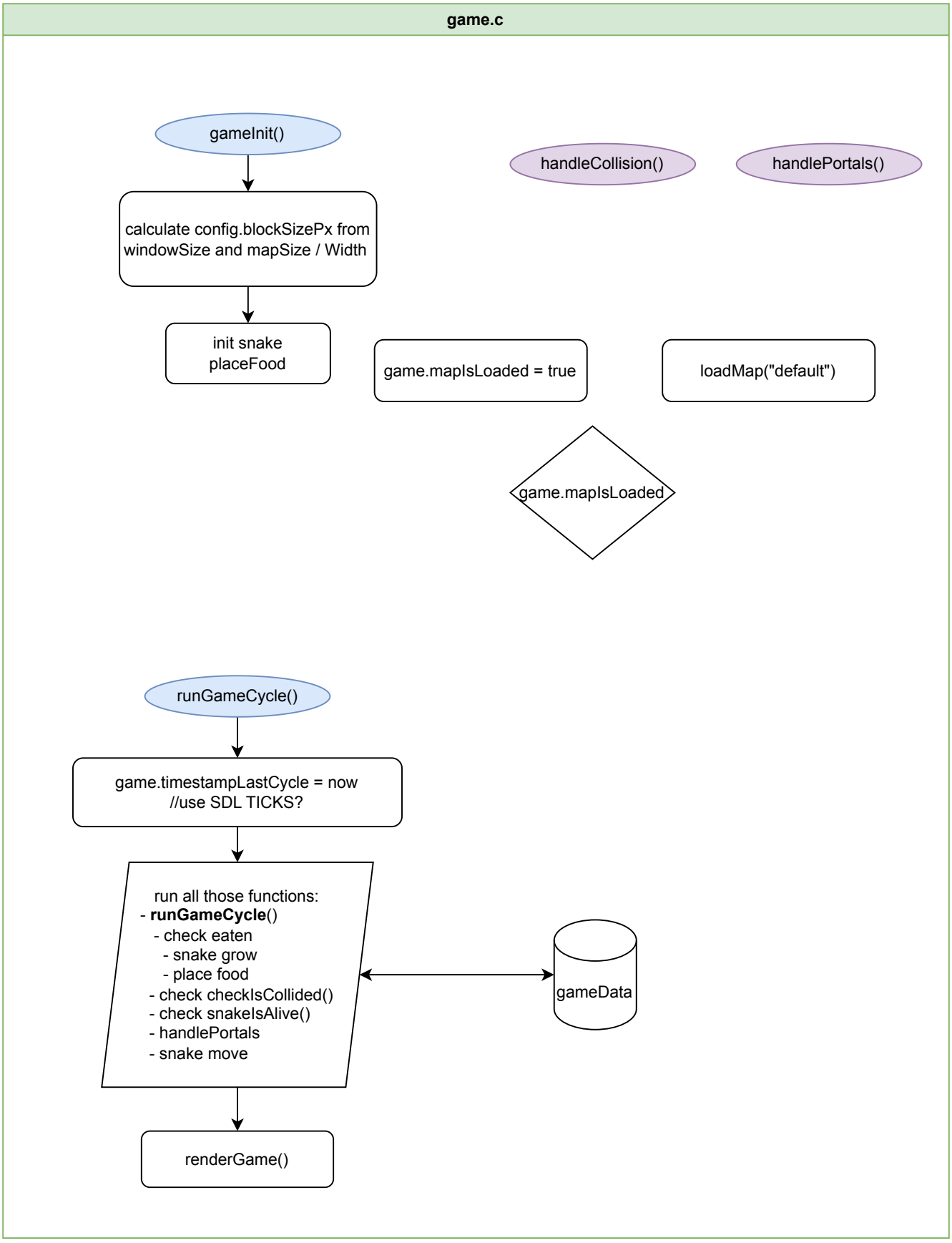
render.c

:

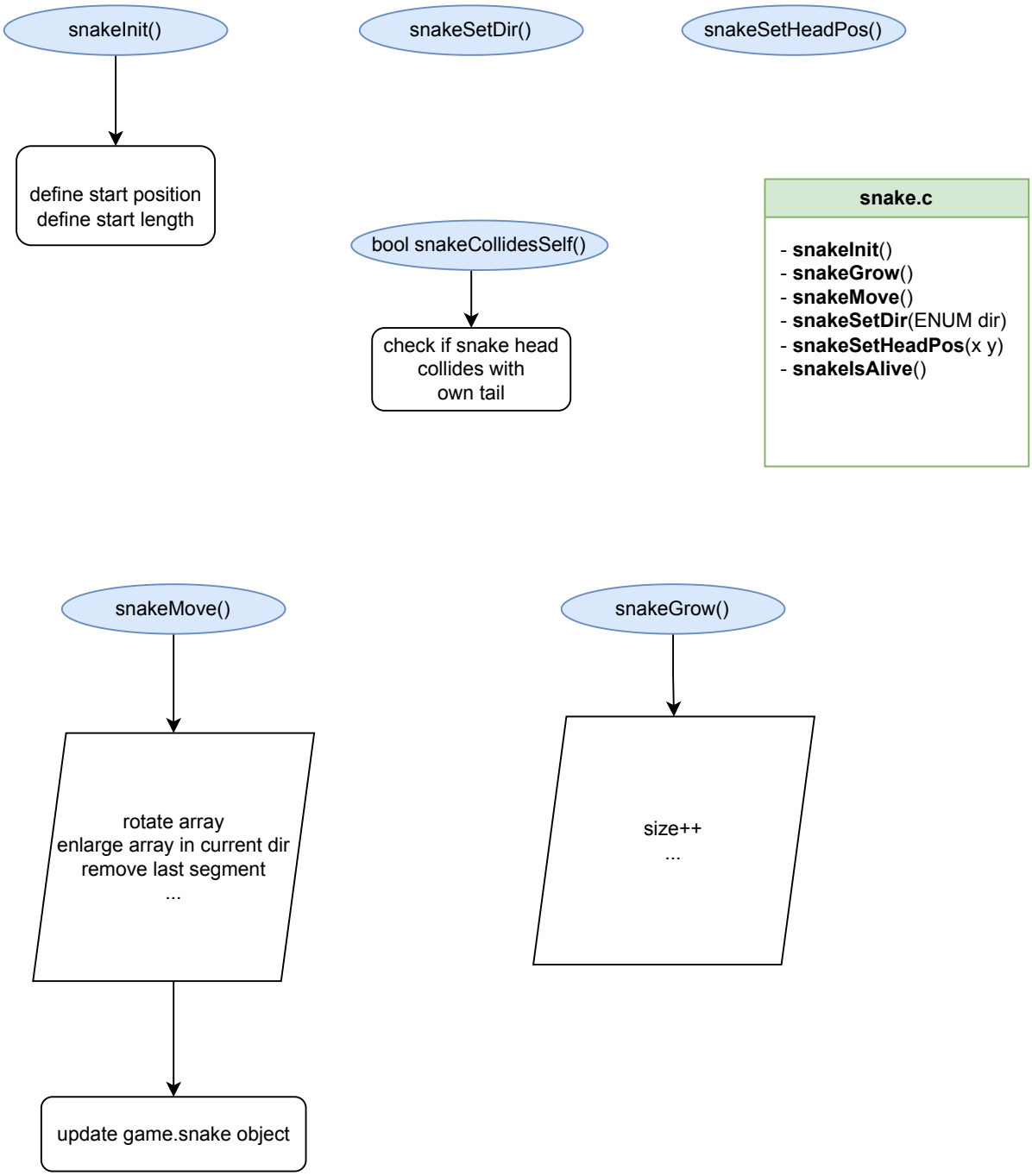
Examples:
https://glusoft.com/sdl2-tutorials/make-health-bar-sdl/
https://dev.to/noah11012/using-sdl2-drawing-rectangles-3hc2



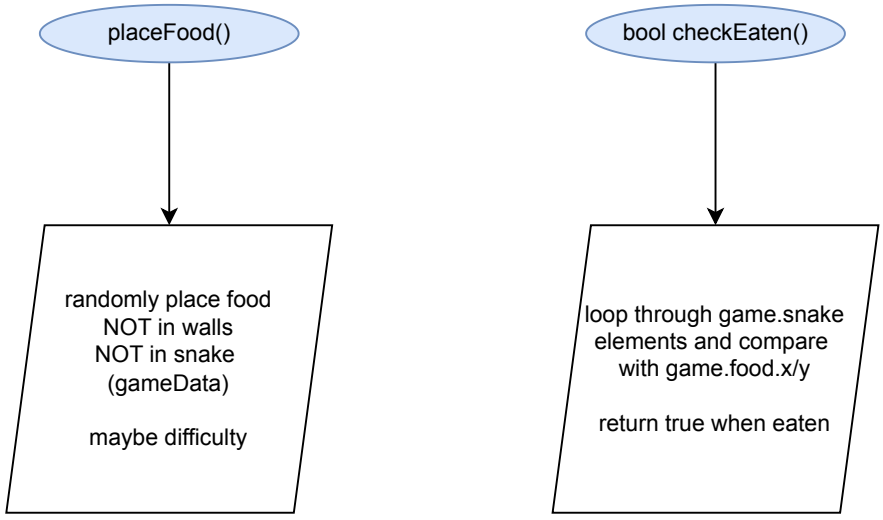
game.c


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()

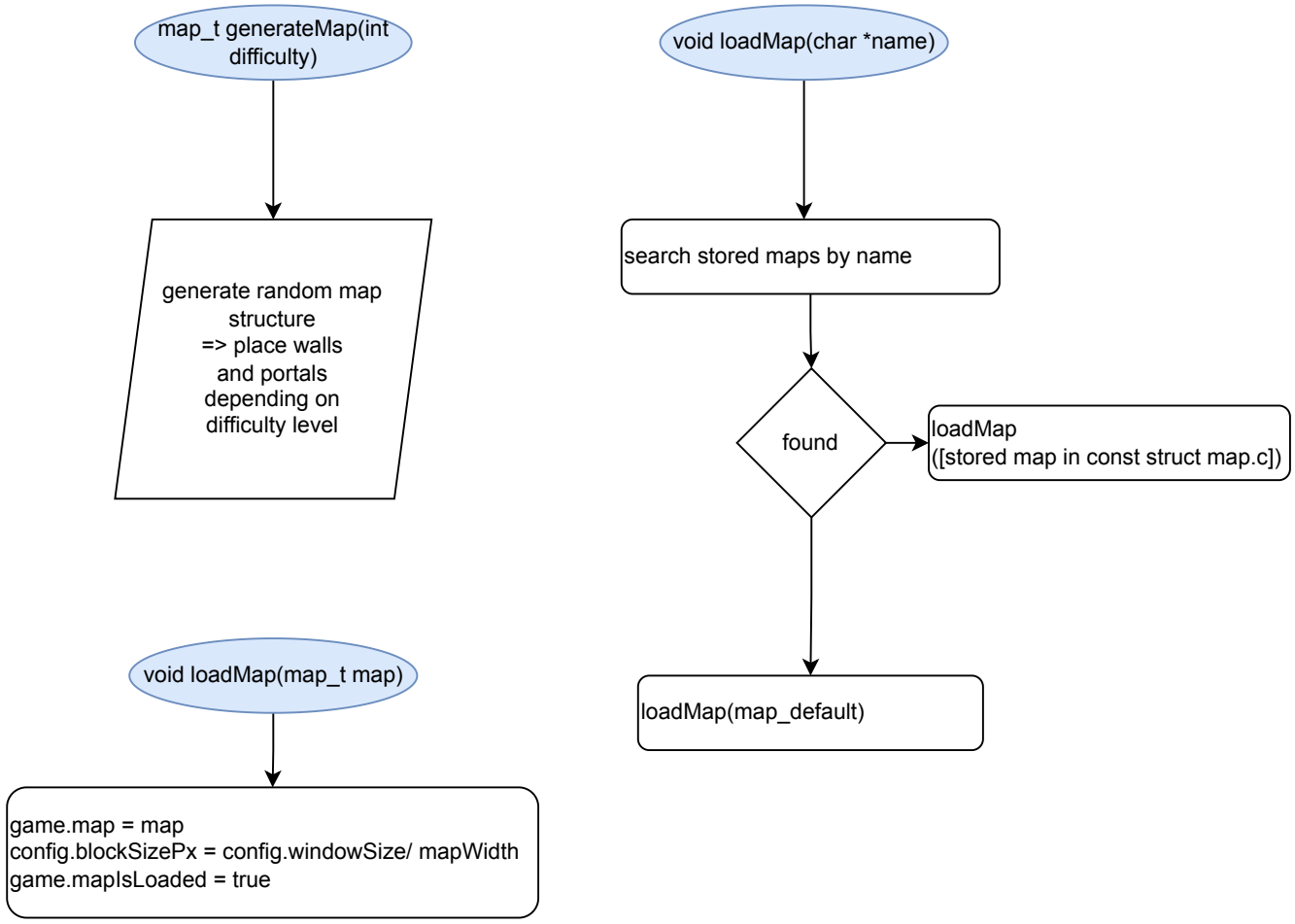
snake.c


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