

## STATE: Initial config

- Set default config values
  - Ask for username
- OUT: Username

## STATE: TOP Scores

- IN: Username
- Show TOP scores
  - Show score of the user

## STATE: (Re)Start

- Direct user to the menu they want (game, TOP scores, settings)

## STATE: Settings

- IN: Config
- Show list of configurable things and their state
  - Ask for changes or start
- OUT: Config

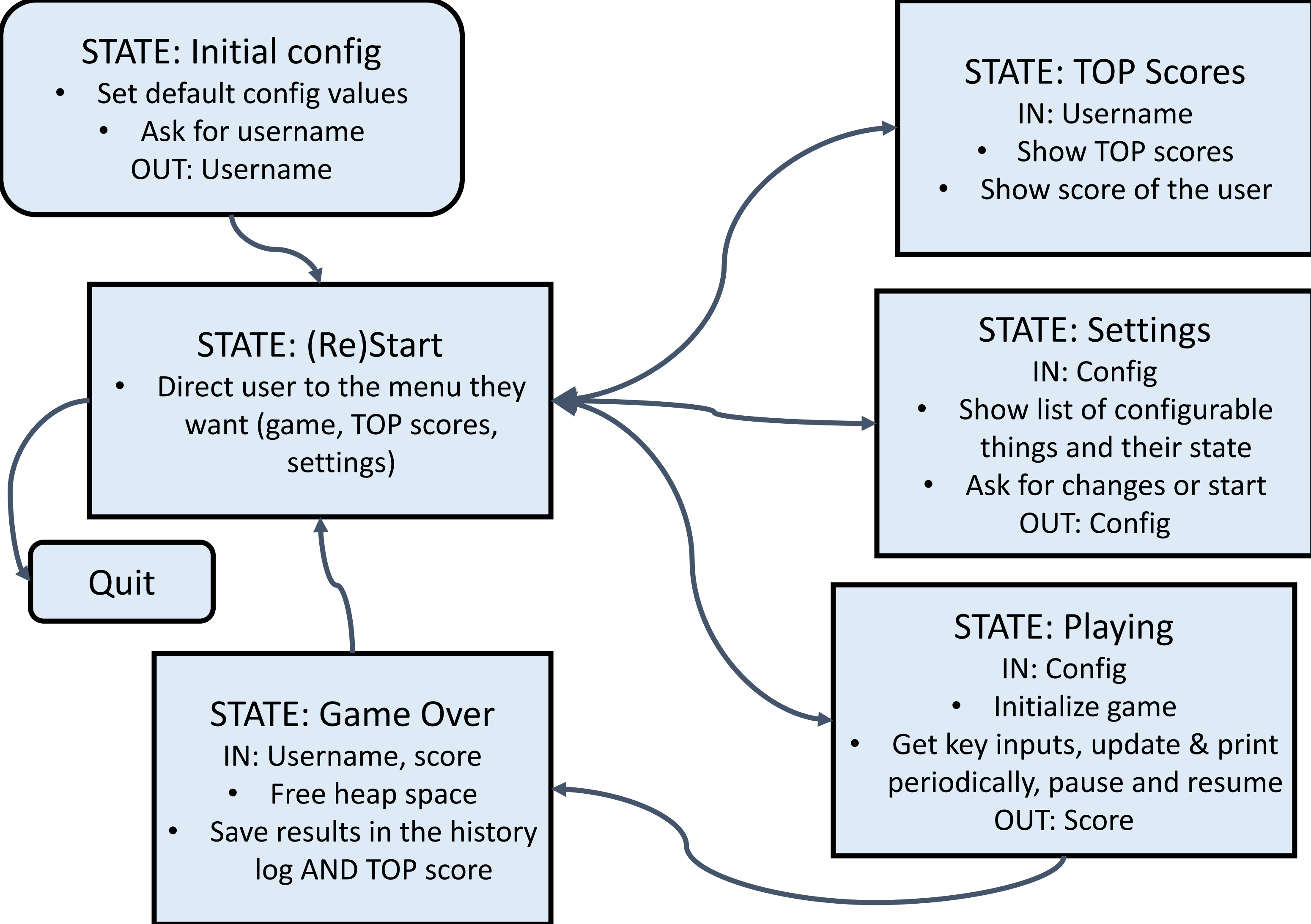
## STATE: Playing

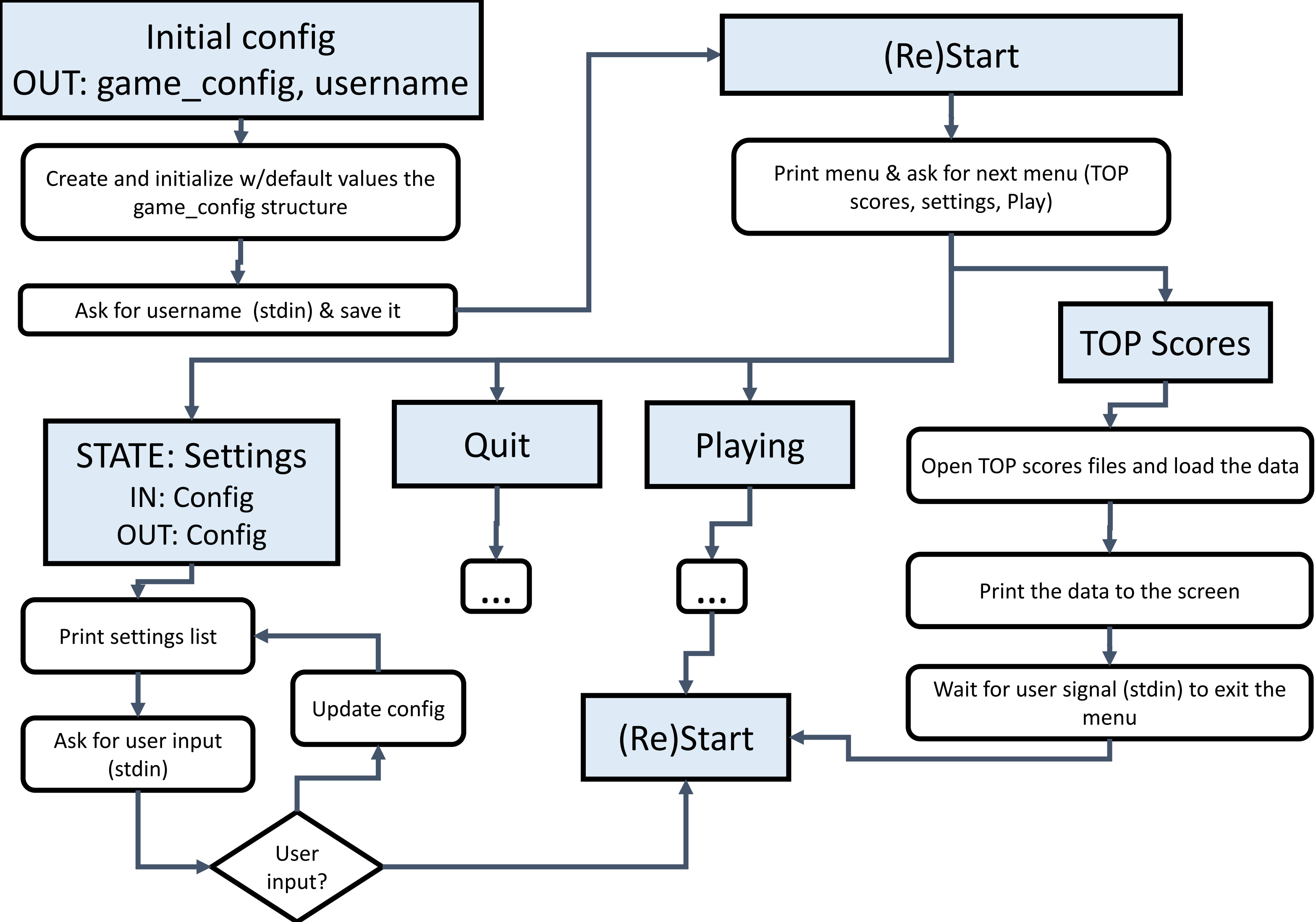
- IN: Config
- Initialize game
  - Get key inputs, update & print periodically, pause and resume
- OUT: Score

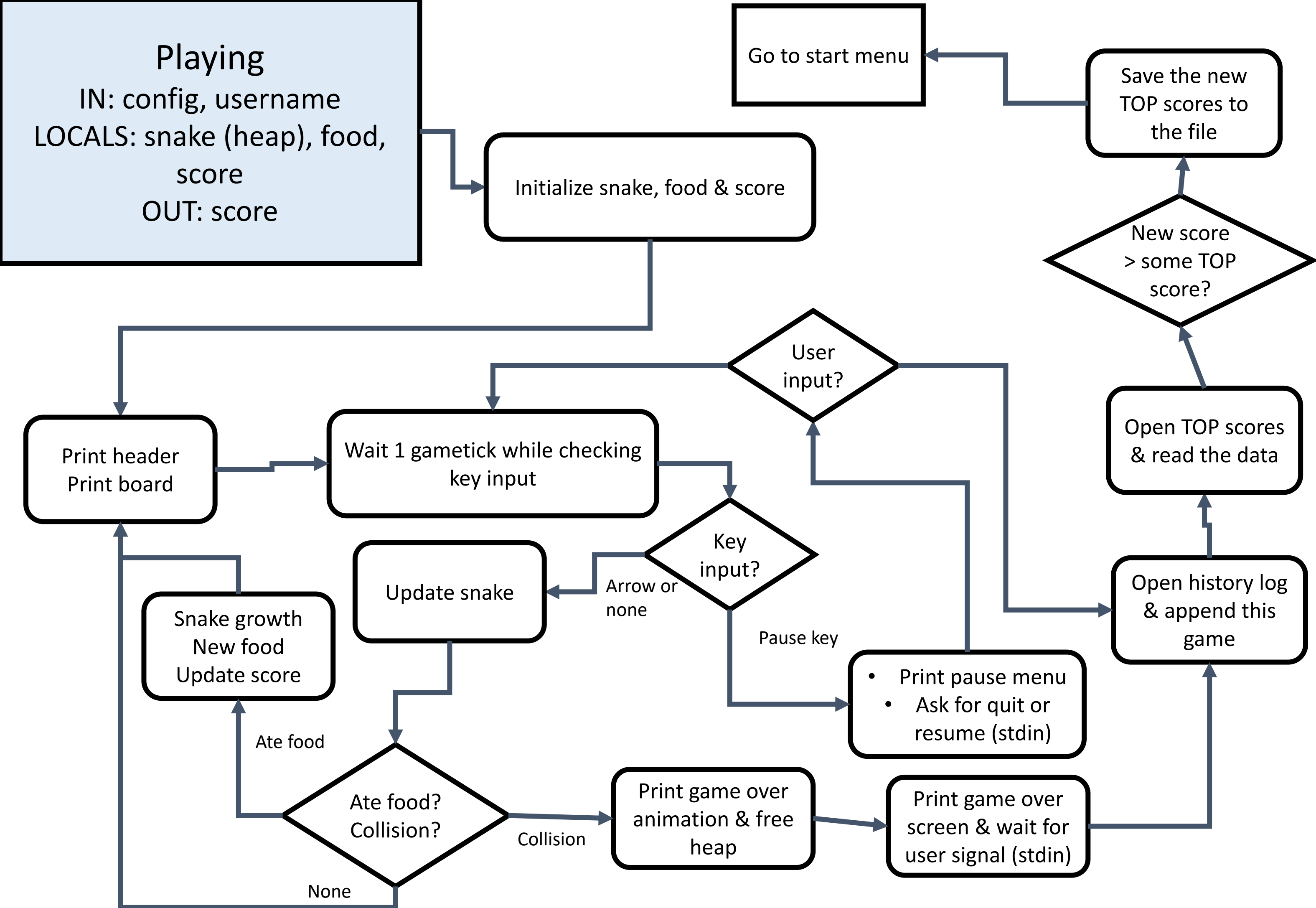
## STATE: Game Over

- IN: Username, score
- Free heap space
  - Save results in the history log AND TOP score

Quit







## Game rules

- Typedef enum { UP = N, DOWN = S, ..., PAUSE } key\_t;
- key\_t getInputInGametick (uint gametick\_length);  
Checks for a key input during a whole gametick

```
typedef struct { uint x; uint y } food_t;  
• food_t newFood(uint width, uint height);  
• Macro OCURREDCOLLISION(phead)
```

- gamestate\_t runOneGameTick(snake\_t\* snake, food\_t\* food, uint\* score, const info\_t\* gameinfo);

## Menus

- gamestate\_t pause();  
Prints the pause menu and waits for user input to resume or exit, and returns it
- void initGame(settings\_t\* settings, char\* user\_buffer);  
Initializes the default settings and asks the player for the name
- gamestate\_t printStart();  
Prints the start module and asks for the menu (play, TOP scores or settings)
- void printTopScore(const char\* users\_buffer[], const uint scores\_buffer[]);

## Graphics

- void printGameInit(const part\_t\* phead, uint width, uint height, food\_t food);
- void printBoard(const part\_t\* phead, food\_t food, uint width, uint height);  
Prints the board of the game
- void printHeader(const char user[], uint score, uint lives);
- void printGameOver(uint score);
- void eraseScreen();

## Body

- int initSnake(snake\_t\* snake, int init\_x, int init\_y, direction\_t init\_orient, uint init\_length);  
Initializes the snake with some initial position, orientation and length.  
Returns 0 on default or error code if malloc fails.
- int isInsideSnake(int x, int y, const part\_t\* phead);  
Boolean function: returns 1 if there's a part of the snake in the position (x, y), 0 if not
- void update(snake\_t\* snake, direction\_t dir);  
Moves the snake one step into the direction dir, not increasing it's size
- int newNode(snake\_t\* snake);  
Creates a new part behind the tail and saves it in the snake as the new tail.  
Returns 0 on default or error code if malloc fails
- void freeAll(part\_t\* phead);  
Frees the memory used for the snake on the heap.

## progdata.h

```
typedef struct {  
    uint board_width;  
    uint board_height;  
    uint init_snake_length;  
    clock_t tick;  
    char username[MAX_USER_LENGTH];  
    ...  
} settings_t;  
  
typedef enum {  
    PLAYING, PAUSE, GAMEOVER, START,  
    TOPSCORES, SETTINGS  
} gamestate_t;  
  
// + constants and other things
```

## History

```
int saveThisMatch(char user[], uint score);  
  
int appendMatchToTopScore(char user[], uint score);  
  
int getTopScore(char* users_buffer, uint* scores_buffer);
```

