Project 1: Jumping Frog (action game)

Event-Driven Programming with Text Graphics

Version 0.94 (early version), November 21, 2024

Michał Małafiejski
Programming Basics 2024/2025

Remarks? jump to me: michal@animima.org

Why?

To acquire the following knowledge or skills:

- basics of programming in C/C++
- basics of text graphics
- event-driven programming
- file operations

How? (technologies)

- You can use any ANSI/ISO compliant C/C++ compiler (Linux, MacOS, Windows).
- You can use one of two text graphics libraries: `*curses.h` (Linux, MacOS, Windows-) or `conio*.h` (Windows)

How? (C/C++)

The following are obligatory requirements:

- use functions and structures with procedural programming
- use standard C or C++ libraries and your own data structures
- generate random numbers with `srand()` and `rand()`
- manipulate on files using the `FILE` structure and C functions that operate on `FILE`: e.g., `fopen`, `fgetc`, `fputc`, `fscanf`, `fprintf`, `fread`, `fwrite`, `fclose`

How? (limitations)

- It is forbidden to use object-oriented programming techniques (C++), in particular: classes, inheritance mechanisms, polymorphism, overloading of functions and operators. It is allowed to use data structures without methods (functions) inside the structures (C).
- It is forbidden to use global variables, except for variables of simple types (e.g. int, float, etc., without pointers) preceded by the `const` specifier.
- The use of the C++ (STL) template library is strictly forbidden, but you may use own template functions.
- The number of characters of any function used (including `main()`) is limited to 2^{10} bytes (comments and whitespaces are ignored).

Note: Failure to comply with the above restrictions will result in disqualification of the project and obtaining 0 points.

What?

- Obligatory (graphics, logic, code, max. 6 pts)
  1. Basic user interface: status, playable area, jumping frog, moving cars, streets (lanes), destination place, colors (1 pt)
  2. Five or more cars constantly moving and randomly appearing which are wrapping or bouncing, detection of collisions between frog and cars (1 pts)
  3. Timer and timing mechanism, cars running at different speeds, frog jumping with time breaks between jumps (1 pt)
  4. Using separate configuration text file with parameters of the game, e.g., size and shape of the frog and cars, playing area, number, types and speed of cars, random seeds, colors, etc (1 pt)
  5. Implementation requirement: well organized code with own structures (`struct`) and functions (2 pt)
- Optional (game and more logic, max. 11 pts)
  7. Some cars are disappearing when reaching the border, new random cars are placed on the street (possible time intervals between) (1 pt)
  8. There are a few static obstacles (e.g., between street lanes) and the speed of some cars is changing during the game (1 pt)
  9. Some cars stop the movement when the frog is close to them (1 pt)
  10. Some cars are friendly cars and catch the frog and move to the another place (1 pt)
  11. Game ranking (think over the concept of points) (1 pt)

What? (cont)

- Optional (harder, max. 15 points)
  12. Stork flying towards the frog (vertical, horizontal and diagonal movement) trying to catch the frog (2 pts)
  13. Build the concept of levels with non-trivial logic (at least three levels) and apply it to the game (2 pts)
- Extra points (max. 18 points)
  14. Recording and replaying the whole game to and from file (`FILE`), respectively (2 pts)
  15. Using `GitHub` account for storing, versioning and presenting all the history, trying to apply the agile methodology to realize the project individually (1 pt)

Scored only any *prefix* from all task list numbers, i.e. a failure to complete task number i results in not scoring subsequent tasks $i+1, i+2, \ldots$

Basics

- Idea: jumping frog trying to bypass cars and reach the goal
- Simple action real-time game for one player
- The frog controlled by the user starts at the bottom line
- The goal is to reach the destination place (e.g., the top line) by moving the frog

Details

- The moving cars must be controlled by the computer, they may have different behaviour (enemy car, friendly car, etc)
- Timer is a crucial part affecting the game logic: cars are constantly moving during the game with possible different speeds, jumps may be related to some time intervals, etc
- Operations on files should be realized using `FILE` and operations on a text file (not a binary one)
- The text graphics should be *reasonable* what means frog and cars should be easily identified, the look of them is not so important but playability should be well-done
- Be creative building the concepts of ranking and levels: consider time, number of jumps, enemy cars, friendly cars, etc
- The concept of status line (area) is also depending on you, but you must show your name, surname and the university ID

Details (cont)

- Static obstacles may be placed between car lanes or on the lanes; they may affect cars (e.g., cars are bouncing or disappearing) or frog (not possible to make a jump)
- Friendly cars catch and move the frog to the another place *on demand*, which means that frog must send a request to do that (e.g., by pressing some key)
- Stork flying towards the frog is moving with a smaller speed than frog can escape by jumping, moreover, it is flying in the air not respecting any ground obstacles and street lanes

Resources

- Lecture notes on event-driven programing
  - Definitely read it, you will get know not anly about the `ncurses` library, but also you learn about interesting details about timing and controlling
  - `https://www.cs.hunter.cuny.edu/~sweiss/course_materials/unix_lecture_notes/chapter_06.pdf`
- Template of a game using `ncurses`
  - `http://gut.animima.org/pp/Projekt1/demogame/win1.c`
  - compilation: `gcc -o win1 win1.c -lncurses`