Programmer’s Documentation

**A picture containing object

Description automatically generatedMineZweeper**

Markéta Sauerová

Univerzita Karlova MFF

zimní semestr 2018/2019

Programování I.

Contents

[1. Specification Breakdown 3](#_Toc13355977)

[1.1. Brief description 3](#_Toc13355978)

[1.2. Functional requirements 3](#_Toc13355981)

[1.2.1. Menu 3](#_Toc13355982)

[1.2.2. Game 4](#_Toc13355995)

[2. Project Structure 5](#_Toc13355996)

[2.1. Intro1 5](#_Toc13355998)

[2.1.1. Global variables 5](#_Toc13355999)

# Specification Breakdown

## Brief description

MineZweeper is a modified implementation of the classical game MineSweeper, which is a single-player puzzle video game, written in Pascal, using Wingraph, winmouse and wincrt units for better graphical appearance as well as handling user input.

*Original specification:* <Specification-MineZweeper.pdf> .



## Functional requirements

### Menu

1. CloseGraph request

*At any stage, program must react to close graphical window request and properly end itself. It’s meant to be a prime feature for users.*

1. Ban Console

*Forbid the console window from appearance, so that user can focus only on graphical window.*

1. Process mouse events

*Accurately process mouse clicking to prevent the menu from being uncontrollable.*

1. Input reaction time

*Feedback on mouse input from menu buttons should be fast with a small delay, when creating the notion of pressing the button.*

1. CPU usage management

*Don’t use all the CPU time in main loops, checking for the input. Instead use delay function for resting.*

1. Redrawing graphics

*Minimize graphics bugs, while redrawing or overdrawing window.*

1. Reduce flickering

*Reduce flickering of the animated title to the minimum possible.*

1. Execute new process

*Must be able to transfer prime event handler, close graphical window and exit the old process when executing the game, that is written as a separate program.*

1. File handling

*File managing for passing data between individual programs.*

1. Control memory

*Freeing allocated stuff in memory for instance animations, images. Preventing memory leaks.*



### Game

1. CloseGraph request

*At any stage, program must react to close graphical window request and properly end itself. It’s meant to be a prime feature for users.*

1. Ban Console

*Forbid the console window from appearance, so that user can focus only on graphical window.*

1. Loading time

*Reduce time spends with loading and initializing to shortest possible.*

1. Difficulty settings

*Every variable element of the game, that depends on difficulty, must be changed due to settings from Menu output.*

1. Mine distribution

*Distribution of mines must be random and different for majority of newly started games.*

1. Follow game rules

*Square manipulation as well as losing or winning the game should be implemented according to Minesweeper rules.*

1. Mouse accuracy

*The accuracy of clicking on the specific square must be high, avoiding any unwanted squares to be marked.*

1. Mouse click time

*If the mouse button was pressed the delay when processing should be shortest possible.*

1. Timer

*The counting of the time must start exactly when the user presses any mouse button.*

1. Execute new process

*Must be able to transfer prime event handler, close graphical window and exit the old process when executing the Menu or NicknameWindow, which are written as separate programs.*

1. File handling

*File managing for passing data between individual programs.*

1. Control memory

*Trying to optimize occupied memory space, freeing allocated stuff in memory. Preventing memory leaks.*

1. Redrawing graphics

*Minimize graphics bugs, while redrawing or overdrawing window.*

# Project Structure

Project is structured in two main separate programs written in Pascal using Lazarus IDE, **Game01** as the MineSweeper implementationand **Intro1** as Menu and official starter of the whole game. Individual programs are connected by executing each other and exiting their own process.

Along these, minor program **NameWindow**, was also created separately after consideration of possibilities throughout designing the project. Shortly, it serves for obtaining keyboard input from user. Detailed description is situated below.

As mentioned, programs use **wingraph**, **wincrt** and **winmouse** units, which source files are compiled together with code and located in the project folder. Used code will be explained concisely but not in detail. For further usage or interest, it can be found in wingraph unit documentation.

**Sysutils** unit (various system utilities) is also attached and widely utilized.

Programs try to comply with every functional requirement listed above. In this documentation procedures or functions, strictly fulfilling the requirements, will be **marked with number** of the specified one form the list.

This documentation is organized according to two main programs and minor third is described afterwards. Procedures or functions are divided in **high-level** and **low-level** design producing distinct blocks handling various parts of the project. In these blocks, used data structures, algorithms, input/output or necessary arguments will be represented and explained.



## Intro1

Overall it is official starter of the game. One of two main programs. Alone, provides Menu support for users with specified functionalities. Start Game executes the Game01 process and exits current one, Difficulty handles files for passing required settings, Instructions overdraw graphical window and loads specified instructions and game rules, Highscore displays highest score reached in individual difficulties along with the gamer nickname.

### Global variables

* anim: AnimatType

*The returned handle of an animation from GetAnim procedure with a bitmap image taken from screen in the defined rectangle. Later serves for animating the title (rotation). Freeing the memory happens at the end by FreeAnim procedure.*

* i: integer

*Helps with preserving animation of the title, which is splitted in two procedures maintained in prime loop (it has to be changeable and accessible from more places). Also used as indicator when should unpressing menu button be performed.*

* colors: array [0..4] of ^longword = ( @red,@orange,@green,@Blue,@Purple)

*Main menu colors, used for simplifying possible changes in different procedures (LoadFromFile,* *MenuButtons,* *PressButton,* *UnpressButton).*

* word: array [0..3] of string = ( 'Start Game','Difficulty','Instructions','Highscore')

and word1: array [0..2] of string = ( 'Beginner','Intermediate','Expert')

*Continuing usage of these strings (MenuButtons,* *PressButton,* *UnpressButton, Difficulty).*

* + buttonPressed: Boolean

*For handling the notion of pressing the button at first menu stage. Used in prime loop and changed by procedure ProcessMouseEvents.*

* bitmap: pointer

*Points to allocated memory block with loaded static image (back.bmp). Used throughout the program, freed at the end or when executing Game01 process.*

### Procedure Main()