Developing games in C language using allegro.h graphics and sounds library

Alexei DVORAC, Mihail KULEV
Technical University of Moldova
lexhxas@gmail.com, mkmk54@mail.ru

***Abstract* — The DTT ("Destroy the Tower") software is a simple 2D game created using DEV-C++ IDE and the allegro.h library. This project shows the basic possibilities of the previously mentioned graphics and sounds library, as well as of an extension for it, for working with greater variety of graphic objects (images).**

***Index Terms* — allegro.h, animation, game, graphics, sounds.**

I. INTRODUCTION

The purpose of this project was to find libraries that allow us working with graphics in C language. Afterwards, study the possibilities of these libraries for working with images, creating animations and developing simple 2D games in C language.

A simple idea of a game was chosen to do the mentioned tasks.

The game that we will talk about: "Destroy the Tower" (or DTT). The development process has been divided in four steps:

1. Creating the graphics.
2. Connecting all the graphic elements and adding sounds.
3. Implementing a formula for projectile parabolic motion.
4. Creating a friendly user interface (UI).

II. THE ALLEGRO LIBRARY.

Allegro is a free and open source software library for video game development. The functionality of the library includes support for basic 2D graphics, image manipulation, text output, audio output, midi music, input and timers, as well as additional routines for fixed-point and floating-point matrix arithmetic, Unicode strings, file system access, file manipulation, data files, and (limited, software-only) 3D graphics.

As of version 4.0, programs that use the library work on DOS, Microsoft Windows, BeOS, Mac OS X, and various Unix-like systems with (or without) X Window System, abstracting their application programming interfaces (APIs) into one portable interface. There is also an independent port of Allegro onAmigaOS.[1]

III. CREATING GRAPHICS.

The biggest part of images has been created using the graphics software Adobe Photoshop CS4. The other ones were completely copied or included, as elements for other images, from the internet. None of the images were left without being worked. Also, every little graphic object has been saved in .BMP or .PNG format, depending on the further applications of this one.

IV. METHODS THAT WERE USED.

To create animations a sequence of images that were shown with a delay has been selected, specified in order to provide the best performance for an animated object.

For surfing the menus I've included a counter of the respective menu option, which was increasing or decreasing depending on the keys pressed, and after the ENTER key was pressed, the corresponding image of the menu has been shown, depending on what image was shown previously and the value of the counter.

The last and the most important technique that has been used, was buffering. It was extremely helpful when creating animations using .PNG objects, as buffering allowed to avoid blinking of the screen.

Buffering is a method that uses the simple functions as for drawing on the screen, with the difference that the selected images are drawn on a virtual screen in the memory of the computer. Later, this buffer (which is also a graphic object – an image) is drawn itself on the screen as a simple image.

V. RESULTS OF WORK.

As a result of working process a playable version of the game has been obtained. The main problems encountered were: the blinking screen when creating animations using .PNG images and implementing a working formula for parabolic motion of the projectile.

Fig. 1 Loading screen.

The source code for buffering procedure is shown within Appendix A. There, we have a previously created BITMAP object (800x600) named buffer, to which we draw consecutively the objects of the game screen, afterwards we just draw the buffer to the screen.

Fig. 2 The menu.

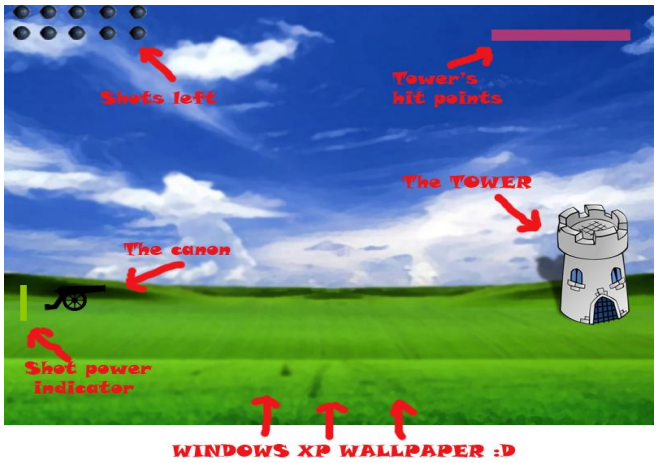

Fig. 3 In-game screen.

The user interface that has been created is quite simple and easy to understand, what are the main characteristics of an user-friendly interface.

There are four options: start the game, help, developers and exit.

To navigate you use the arrow keys, for selecting an option you press ENTER, to go a step back or exit - press ESC.

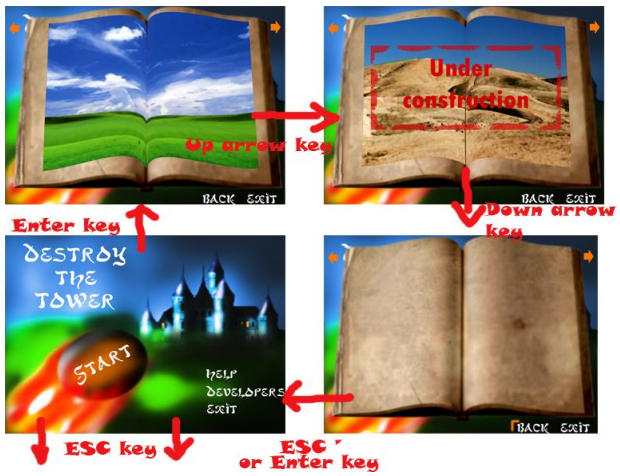

Fig. 4 An example of navigation in the menu

The sequence of the code used for implementing the parabolic motion formula is:

```
x+=xvel*timestep*cos(alpha)+10;
y-=(x-50)*tan(alpha)-0.5*g*(pow((x-
50),2))/pow((power*cos(alpha)),2);
rest(ff);
```

VI. CONCLUSION

Even with a lot of time being spent on this project, it still has many problems that could be corrected in future. However, the result of the work – the game, shows us the basic possibilities offered by the allegro library for working with graphics in C language, although these are very large and a more thorough study of this library, especially with the use of OpenGL could be a great advantage.

APPENDIX A

```
draw_sprite(buffer, background1,0,0);


draw_trans_sprite(buffer,
     tower,650,250);


draw_trans_sprite(buffer,
  bullet[shots], 0, 0);


draw_trans_sprite(buffer,bomb,(int)
          x,(int) y);


rectfill(buffer,600,50, xlife,ylife,
        makecol(170,57,120));


rectfill(buffer,20,450,27,ypower,makeco
        l(150,200,0));


draw_trans_sprite(buffer, canon, 50,
        365);


draw_sprite(screen,buffer,0,0);
```

ACKNOWLEDGMENTS

The process of game development has been supported by a graphic designer Mihail Kalin and a beta-tester Pavel Craitor.

REFERENCES

[1] Allegro library. http://en.wikipedia.org/wiki/Allegro.h