*C Graphics*

Concept:

Graphics programming in C used to drawing various geometrical shapes (rectangle, circle eclipse etc.), use of mathematical function in drawing curves, colouring an object with different colours and patterns and simple animation programs like jumping ball and moving cars.

To begin with we have declared two variables of int type gd and gm for graphics driver and graphics mode respectively.

DETECT is a macro defined in "graphics.h" header file, then we have passed three arguments to initgraph function first is the address of gd, second is the address of gm and third is the path where your BGI files are present.

Initgraph function automatically decides an appropriate graphics driver and mode such that maximum screen resolution is set, getch helps us to wait until a key is pressed, closegraph function closes the graphics mode, and finally return statement returns a value 0 to main indicating successful execution of the program

**Colours in C Graphics Programming:**

There are 16 colours declared in graphics.h header file. We use colours to set the current drawing colour, change the colour of background, change the colour of text, to colour a closed shape etc (Foreground and Background Colour). To specify a colour, can either use color constants like setcolour(RED), or their corresponding integer codes like setcolour(4).

CAR

Description:

The code showcases the capabilities of graphics.h library, creating a visually appealing scene of a moving car with intricate details. The car's design, complete with wheels, mirrors, and headlights

1. **Header**
2. The #include<graphics.h> statement is part of the Turbo C/C++ programming environment and is used to include the graphics library header file. This header file provides functions and declarations that enable the usage of graphics-related functions in C or C++ programs. It is commonly used for creating graphical applications, animations, and games by providing functions for drawing shapes, lines, and manipulating pixels on the screen.
3. The #include<conio.h> statement is used to include the console input/output (conio) header file in C and C++ programs. It provides functions that allow interaction with the console, enabling features like reading keyboard input and controlling the console screen. Commonly used functions from conio.h include getch() for reading a single character from the keyboard without echoing it to the screen and clrscr() for clearing the console screen. It's important to note that conio.h is not a standard part of C or C++, and its usage is specific to certain compilers and environments like Turbo C/C++
4. The #include<dos.h> statement is used to include the DOS (Disk Operating System) header file in C and C++ programs. This header file provides functions that allow low-level programming interactions with the hardware and the DOS operating system on x86 architecture. Functions from dos.h are specific to DOS-based systems and are not part of the standard C or C++ libraries.
5. The #include<stdlib.h> statement is used to include the Standard Library header file in C and C++ programs. This header file provides functions that involve memory allocation, random number generation, and other general-purpose utilities. Commonly used functions from stdlib.h include malloc() and free() for dynamic memory allocation and deallocation, rand() for generating random numbers, and exit() for terminating a program. The functions in stdlib.h are part of the standard C and C++ libraries and provide essential functionalities for many programs.
6. The #include<process.h> statement is used to include the process control operations header file in C and C++ programs. This header file provides functions for program control flow, process execution, and termination. Commonly used functions from process.h include exit() for terminating the program, \_getpid() for obtaining the process ID, and \_getch() for reading a character from the console without echoing it to the screen.
7. **Graphics Initialization:**

**Initialize the graphics system using initgraph.**

1. This function initializes the graphics system. The parameters are pointers to integers (gdriver and gmode) and a string specifying the path to the BGI (Borland Graphics Interface) driver file. This line sets up the graphics system for use.
2. Graphics code: After calling initgraph, you can write your graphics code using functions like line, circle, etc., to draw shapes on the screen.
3. closegraph();: This function is used to close the graphics mode and deallocate resources when you're done with the graphics operations.

**Set the background colour to black using setbkcolor.**

This function is used to set the current background color for drawing operations on the screen.

1. **Drawing the Car:**

**Uses various graphics functions (lines, ellipses, etc.) to draw the shape of a car.**

1. line(x1, y1, x2, y2): Draws a line between the points (x1, y1) and (x2, y2)
2. ellipse(x, y, start\_angle, end\_angle, xradius, yradius): Draws an ellipse with center at (x, y), starting and ending angles, and x and y radii.
3. floodfill(x, y, border): Fills the area around the point (x, y) with the current fill color until it reaches a border colour.
4. setcolor(color): Sets the drawing color to the specified color. Colors can be constants like RED, BLUE, YELLOW, etc.
5. setfillstyle(style, color): Sets the fill pattern and color for filled shapes. The style parameter determines the fill pattern, and color specifies the fill color.
6. **Text Display:**

**Displays text on the screen using outtextxy to prompt the user to press keys.**

outtextxy(x, y, "Hello, World!");: Displays the text "Hello, World!" at the specified coordinates (x, y).

Instructions are given for pressing keys 'L', 'H', 'T', and 'P'.

1. **User Input:**

**It waits for user input in an infinite loop using while(1) and checks for key presses using kbhit().**

The kbhit() function returns a non-zero value if a key has been pressed, and zero otherwise.

The while (!kbhit()) loop continues to execute as long as no key has been pressed.

Once a key is pressed, the loop exits, and you can handle the key press as needed.

1. **Key Press Handling:**

If '1' is pressed, the program exits.

If 'H' is pressed, a horn sound is played using sound() and nosound().

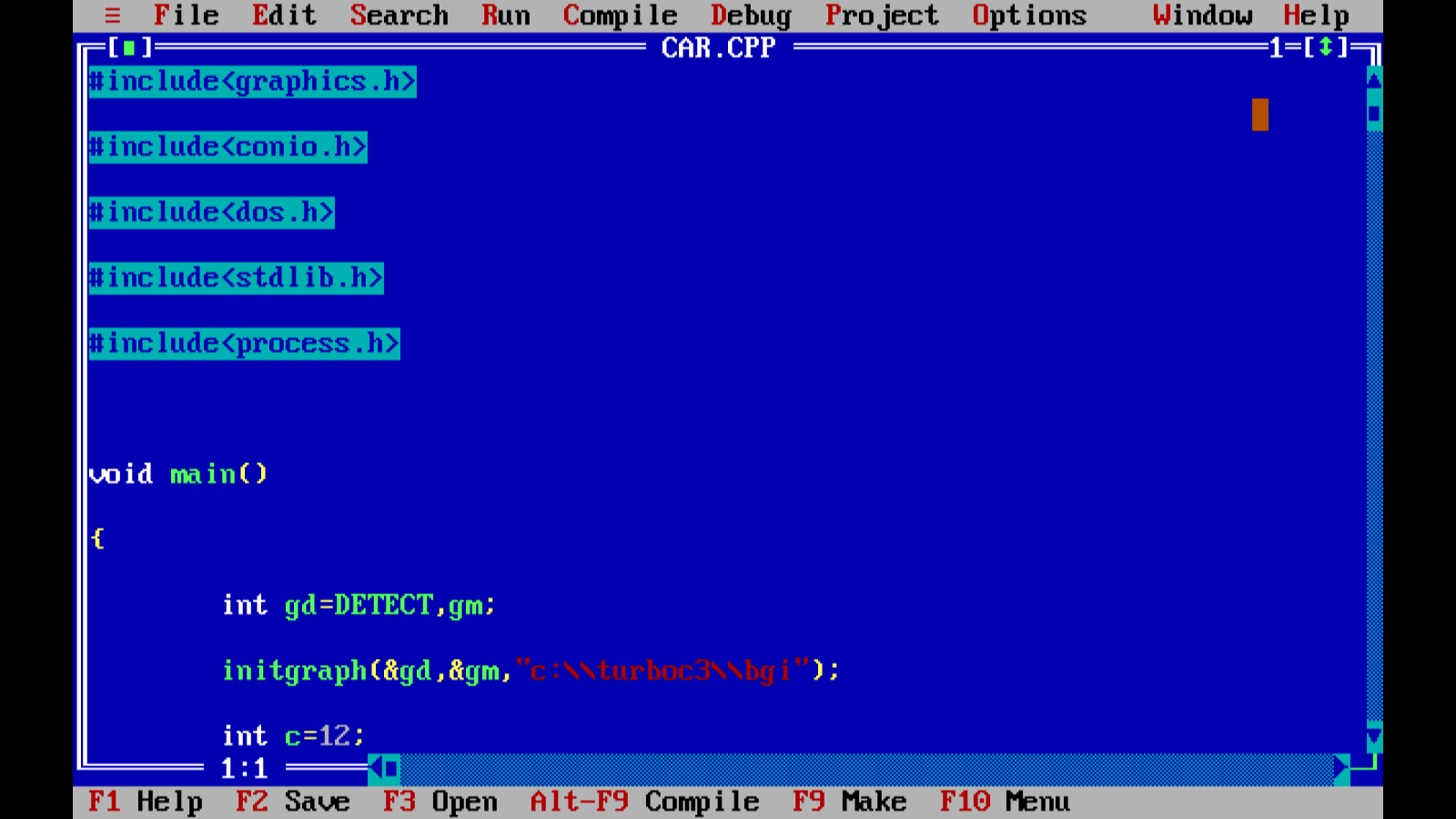
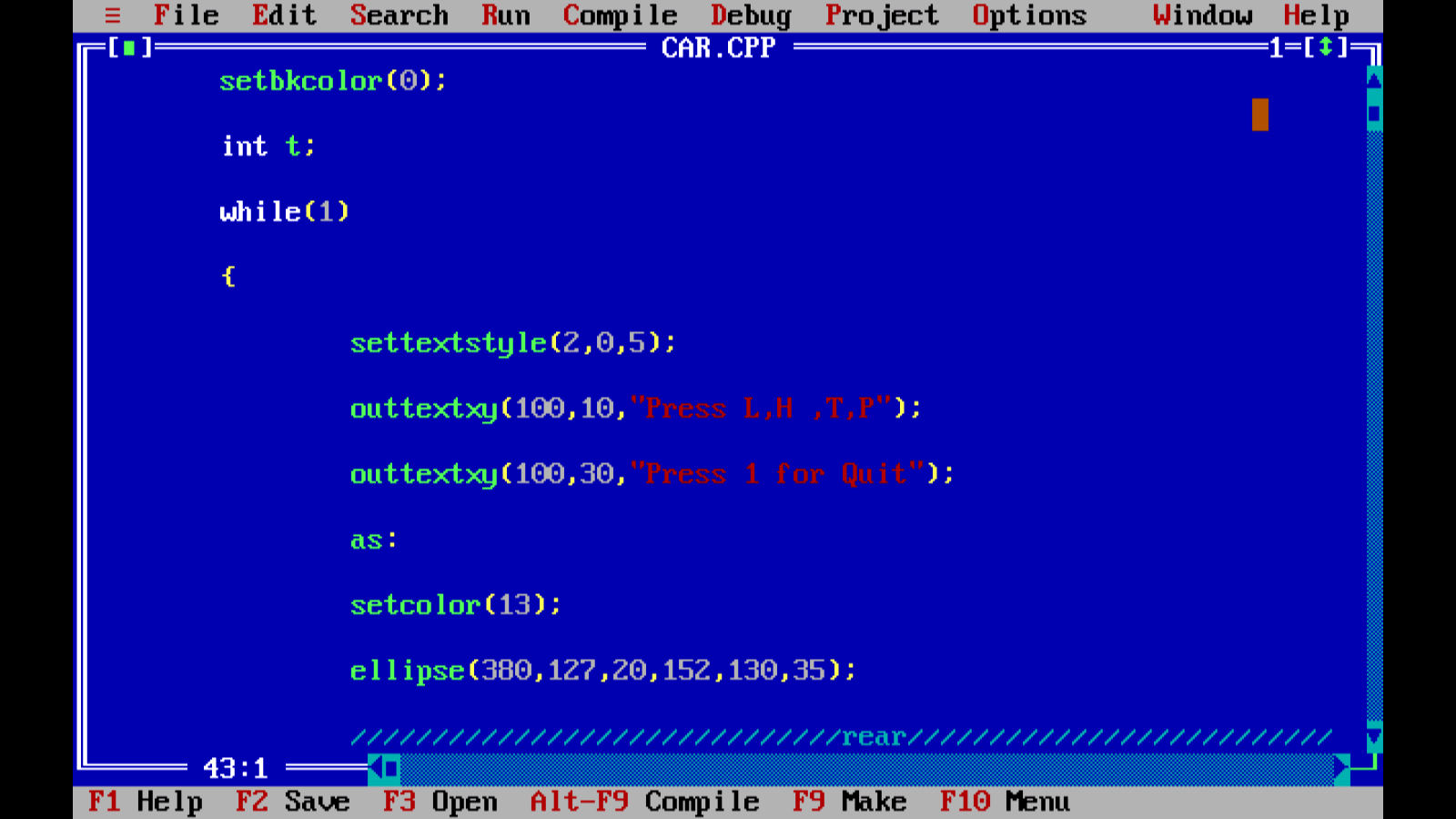
If 'T' is pressed, a colorful animation is displayed on the screen.

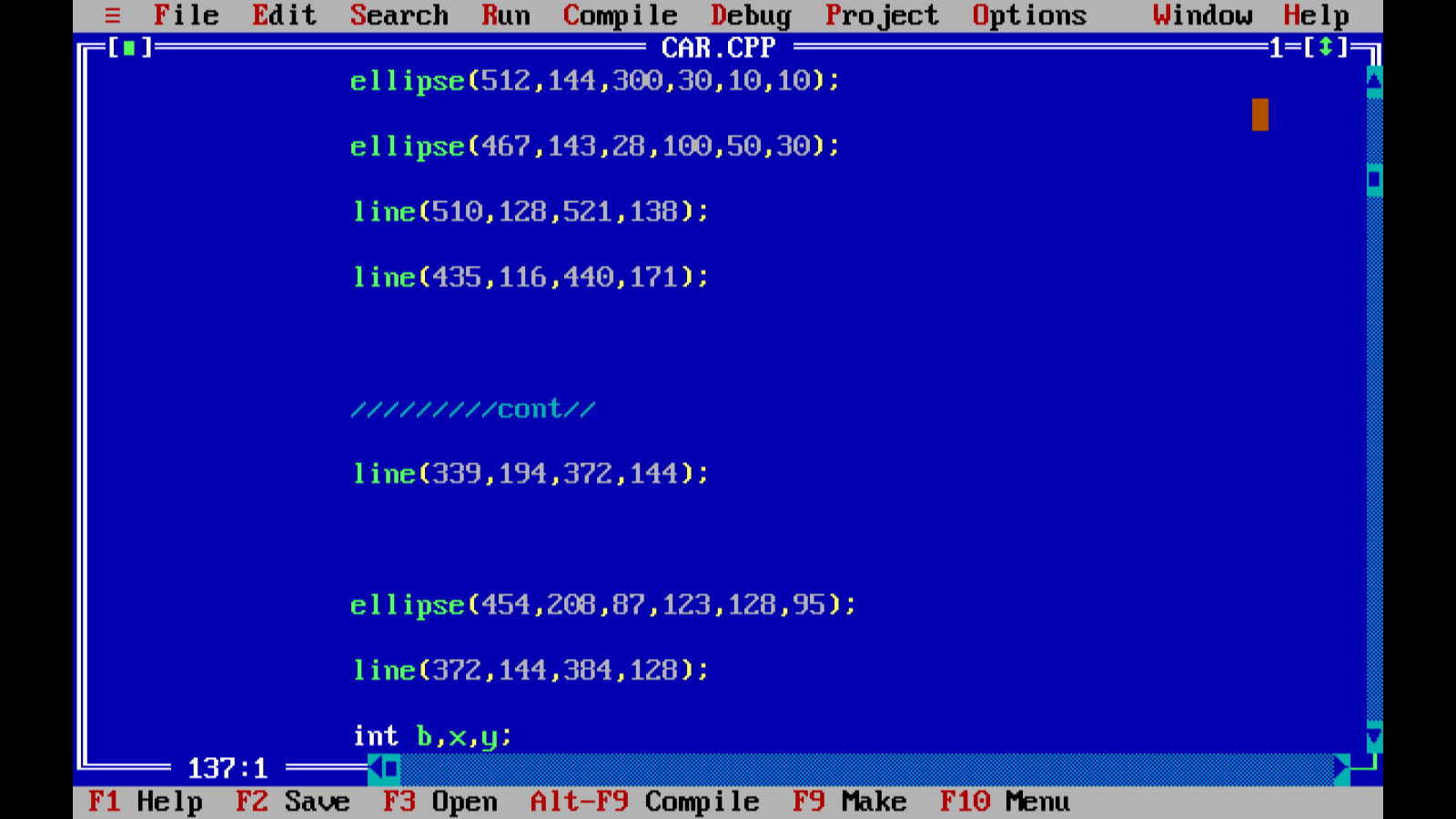
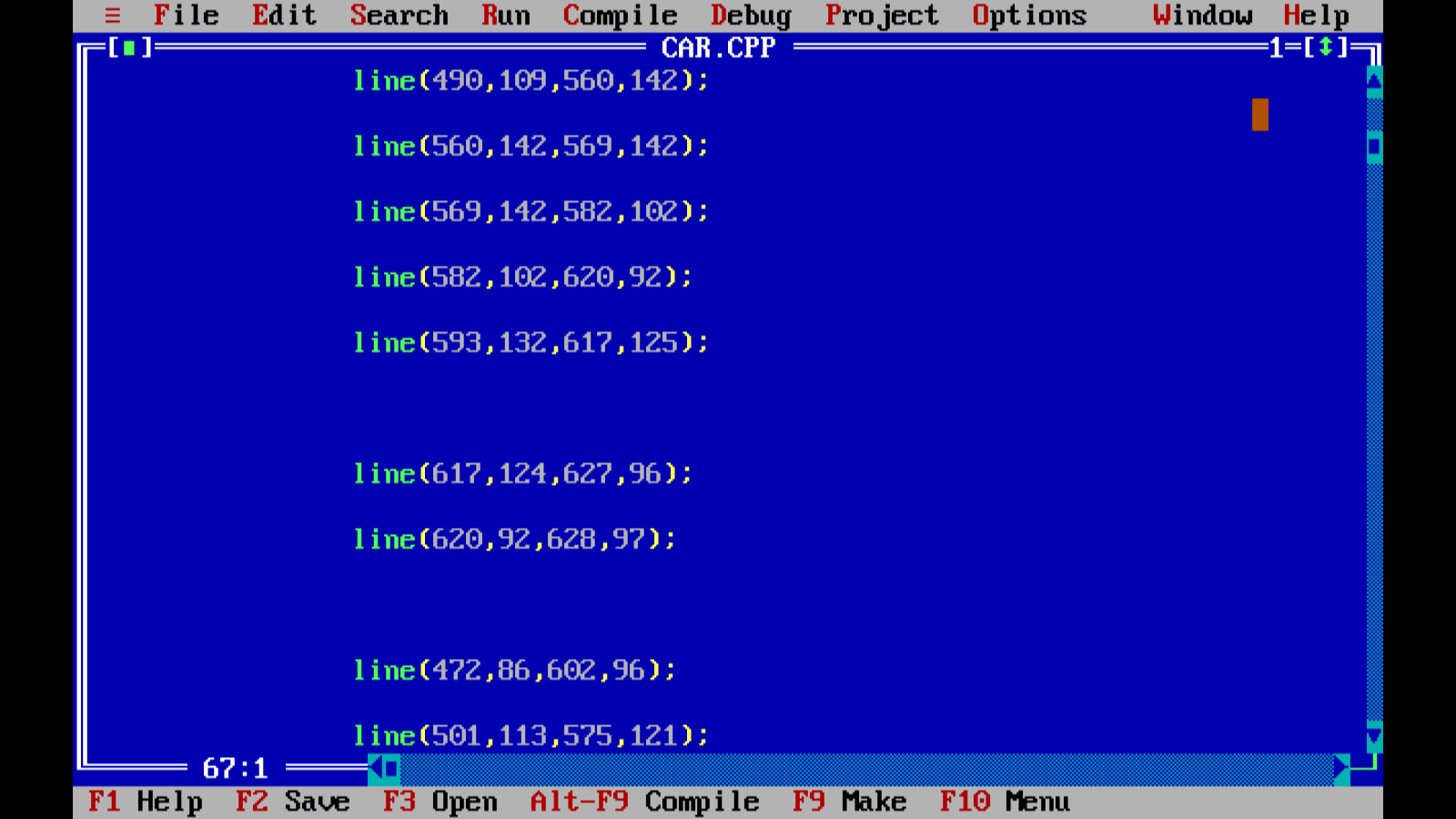
If 'L' is pressed, the lights of the car and road are animated.

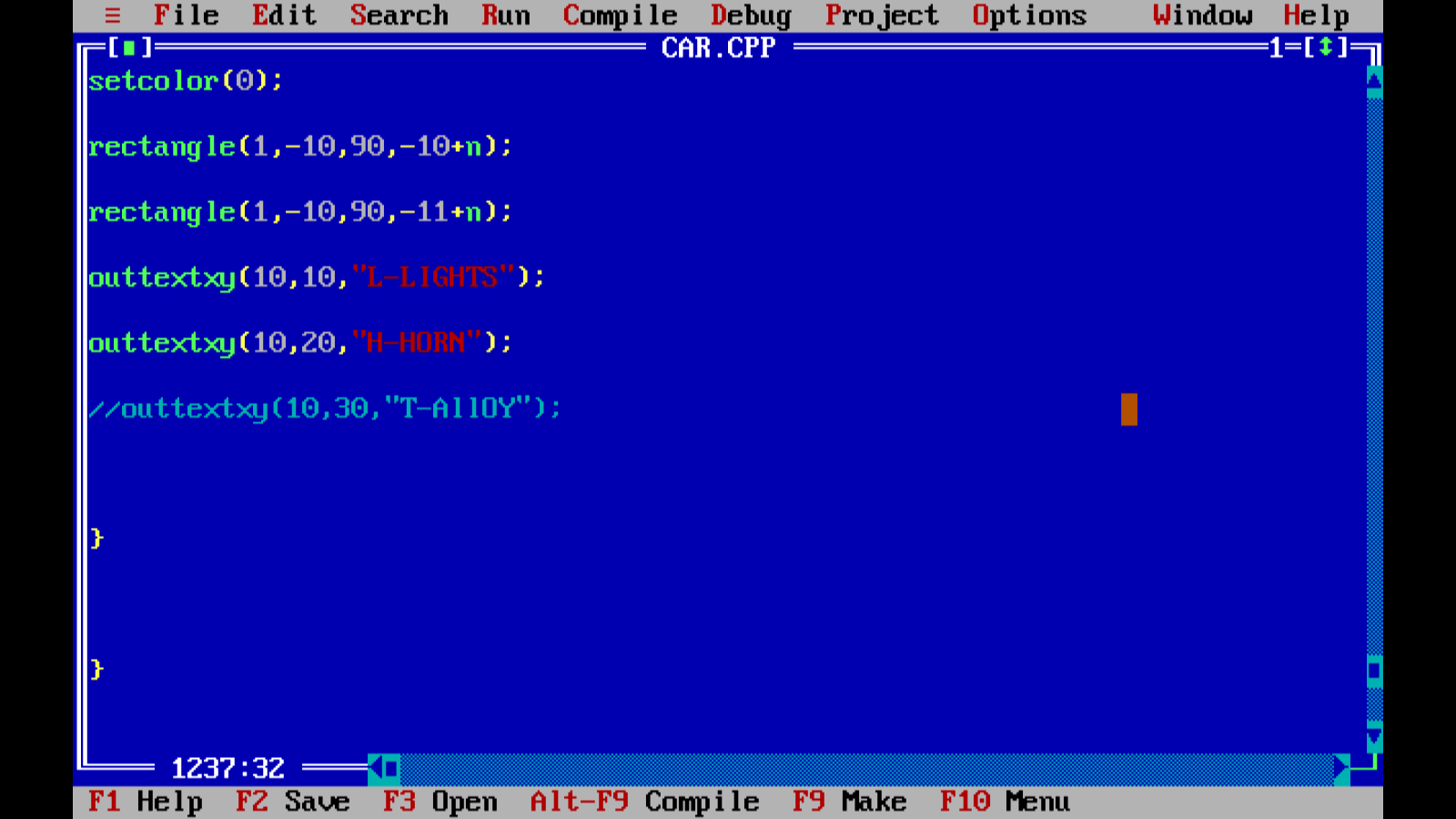
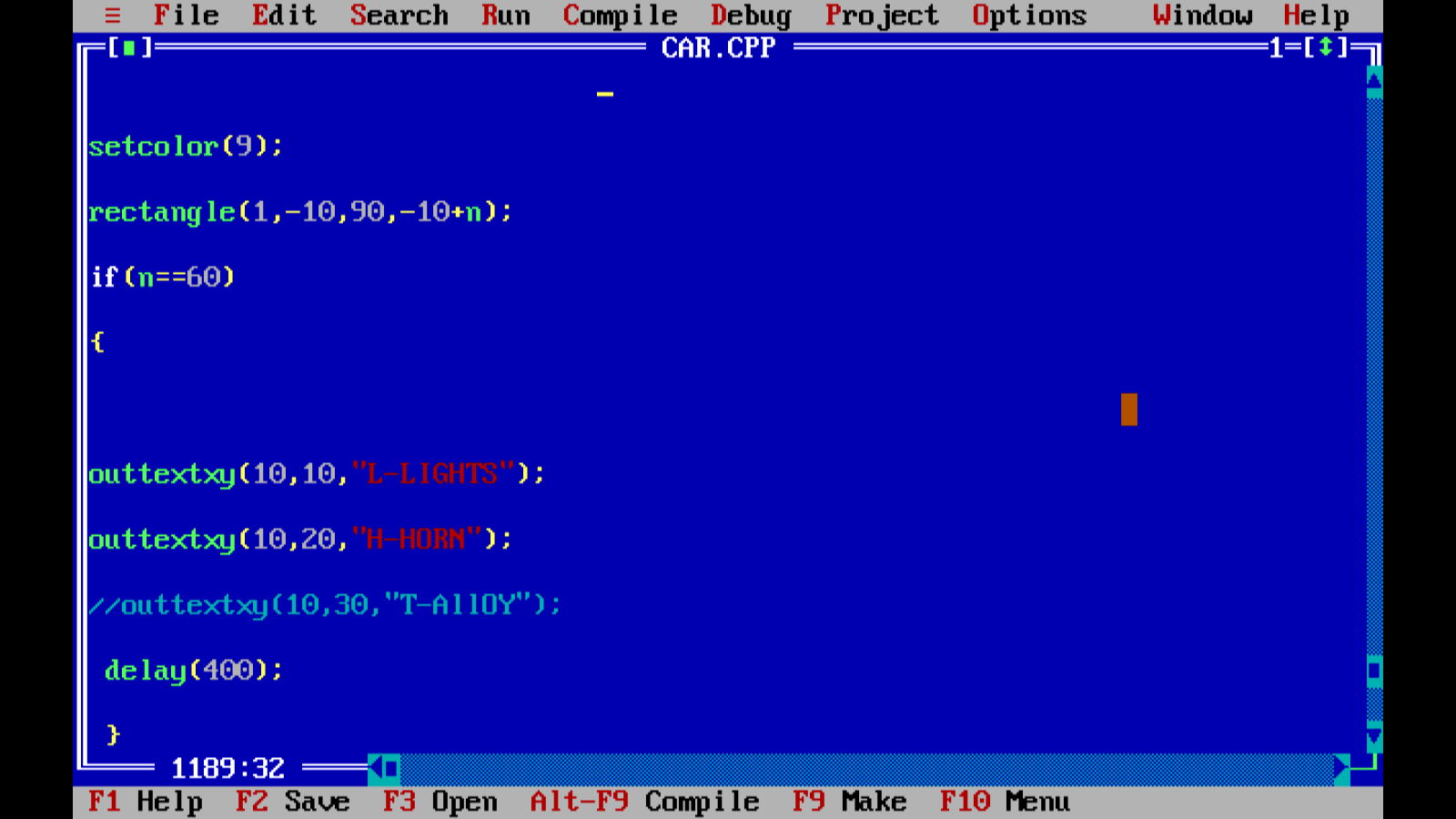
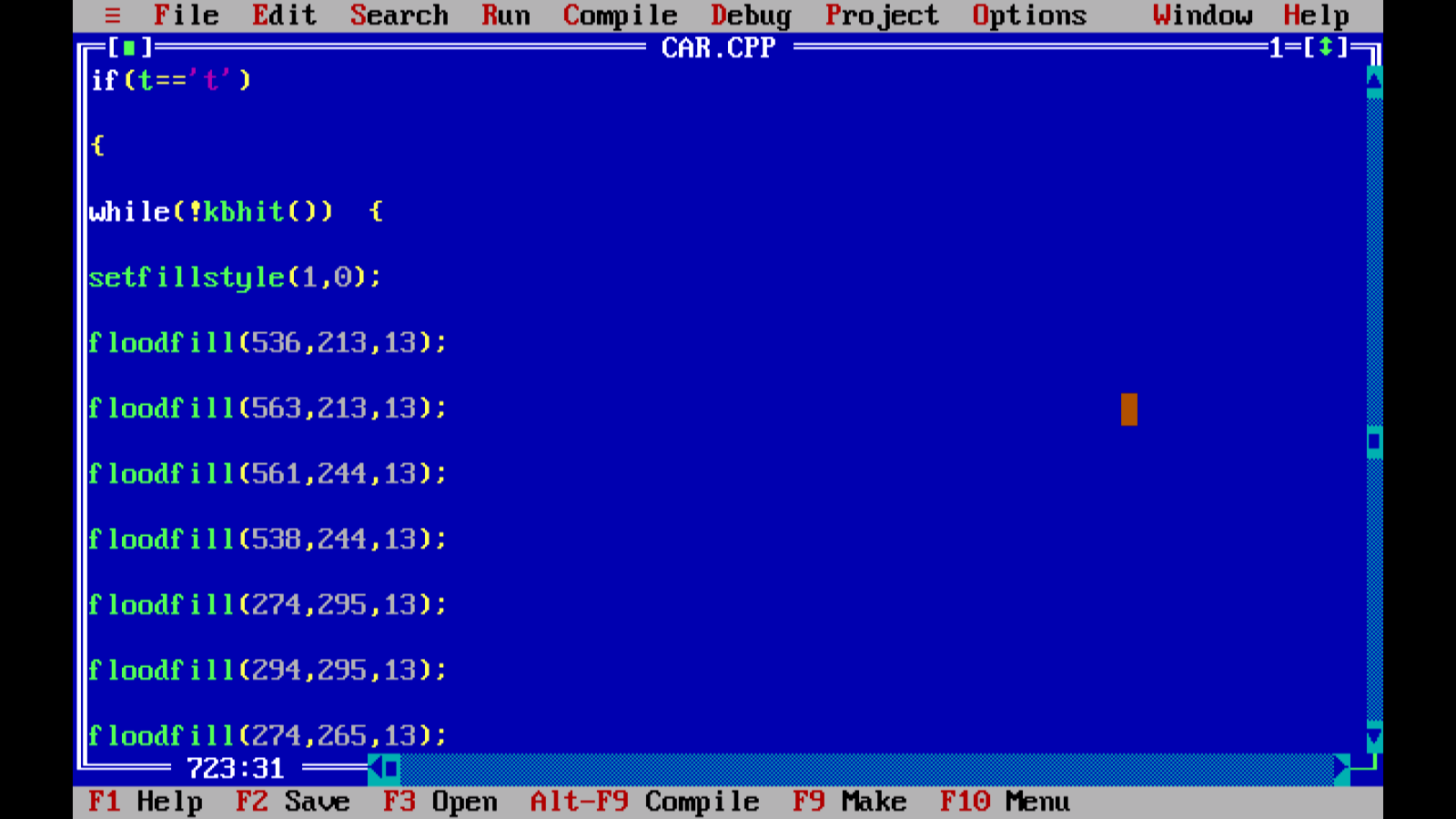
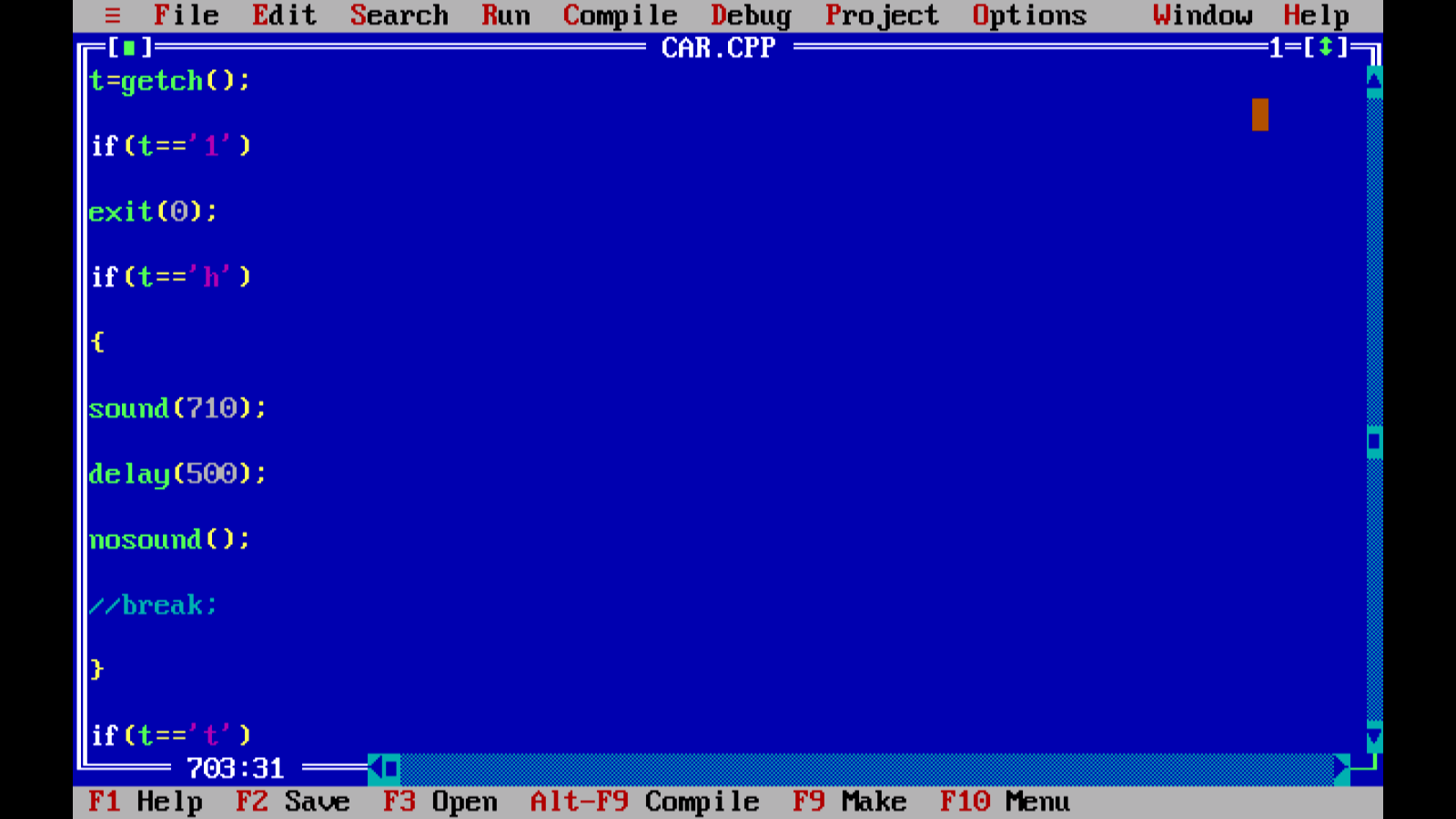
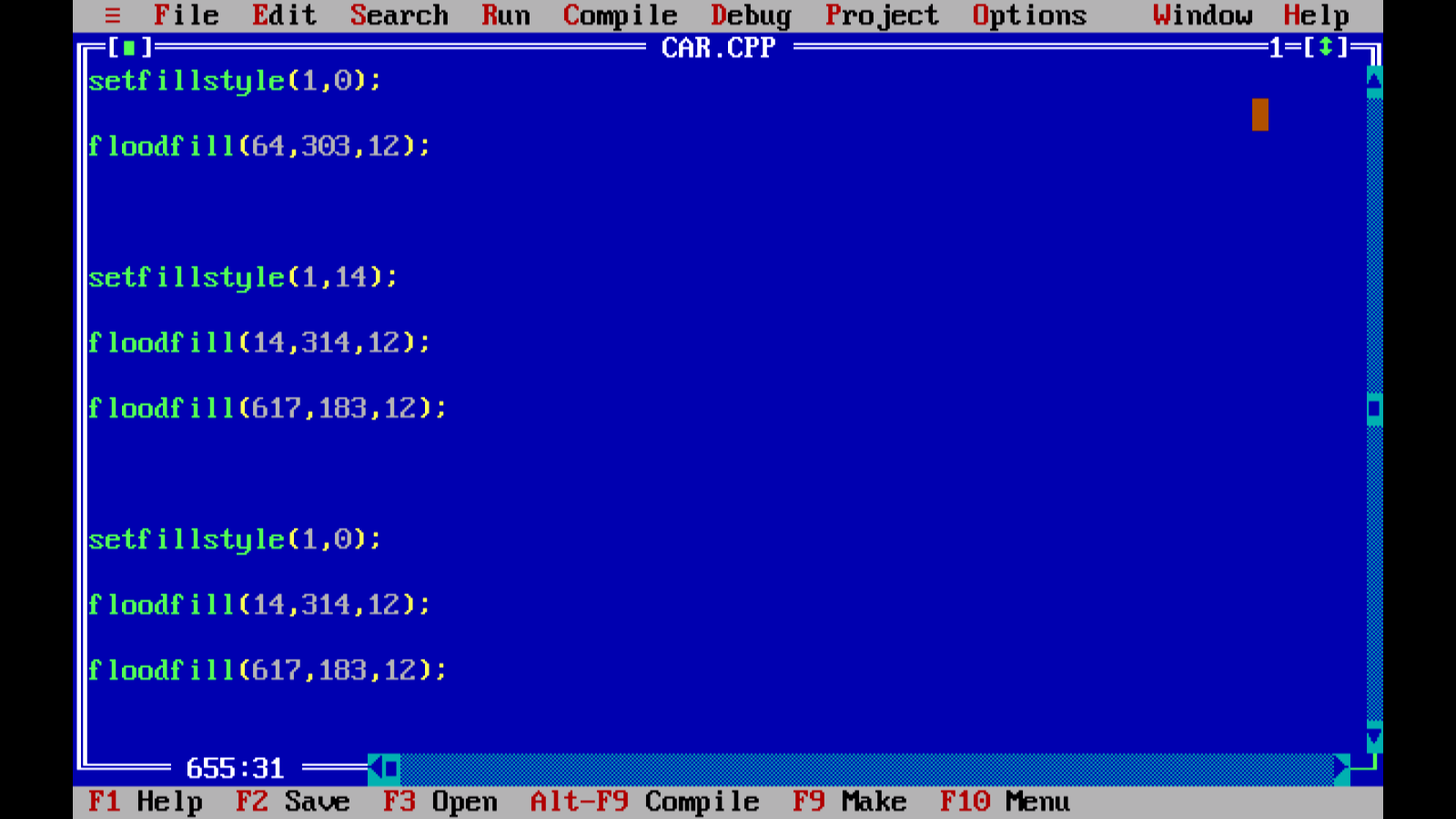
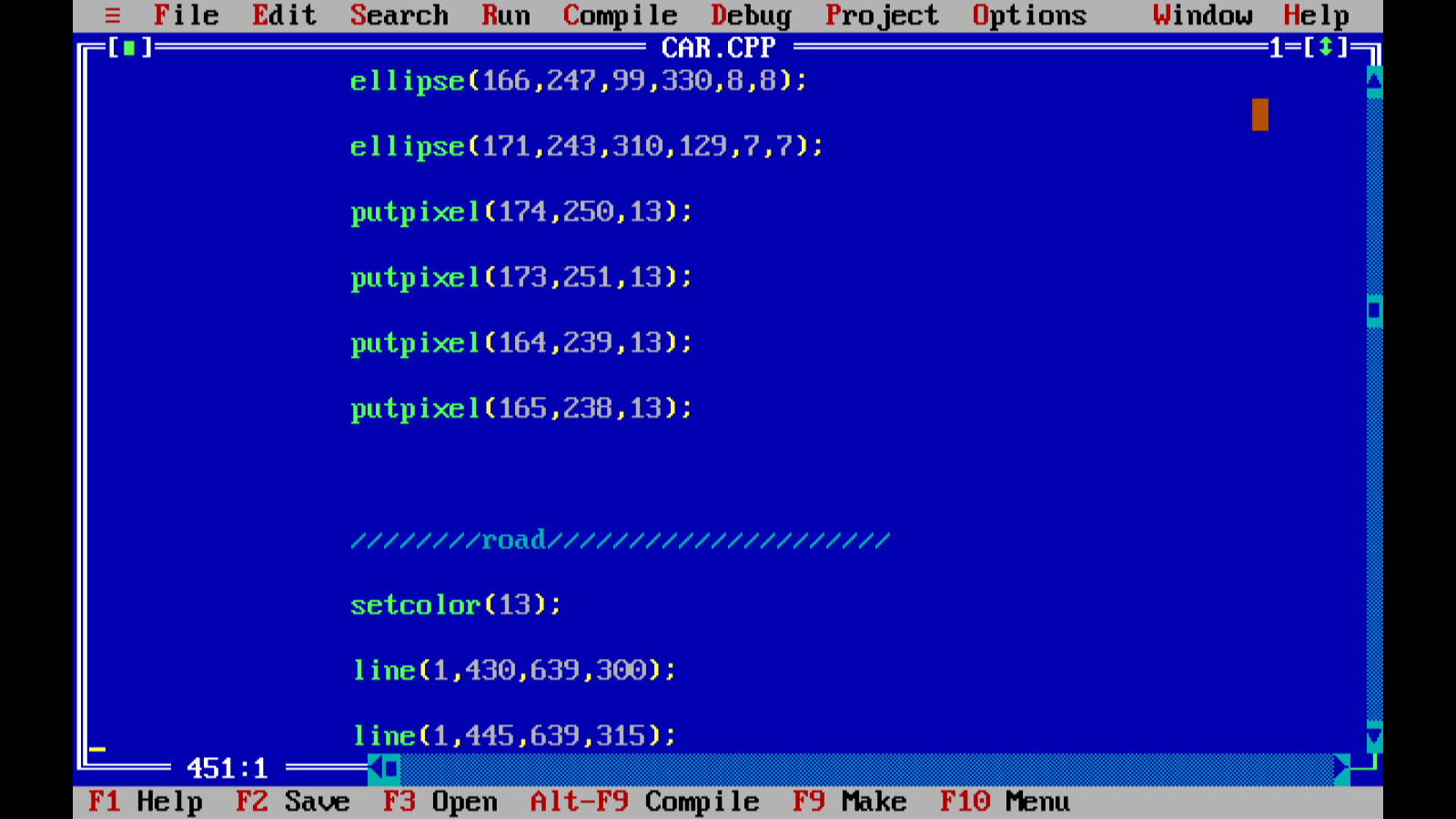
1. **End of Program:**

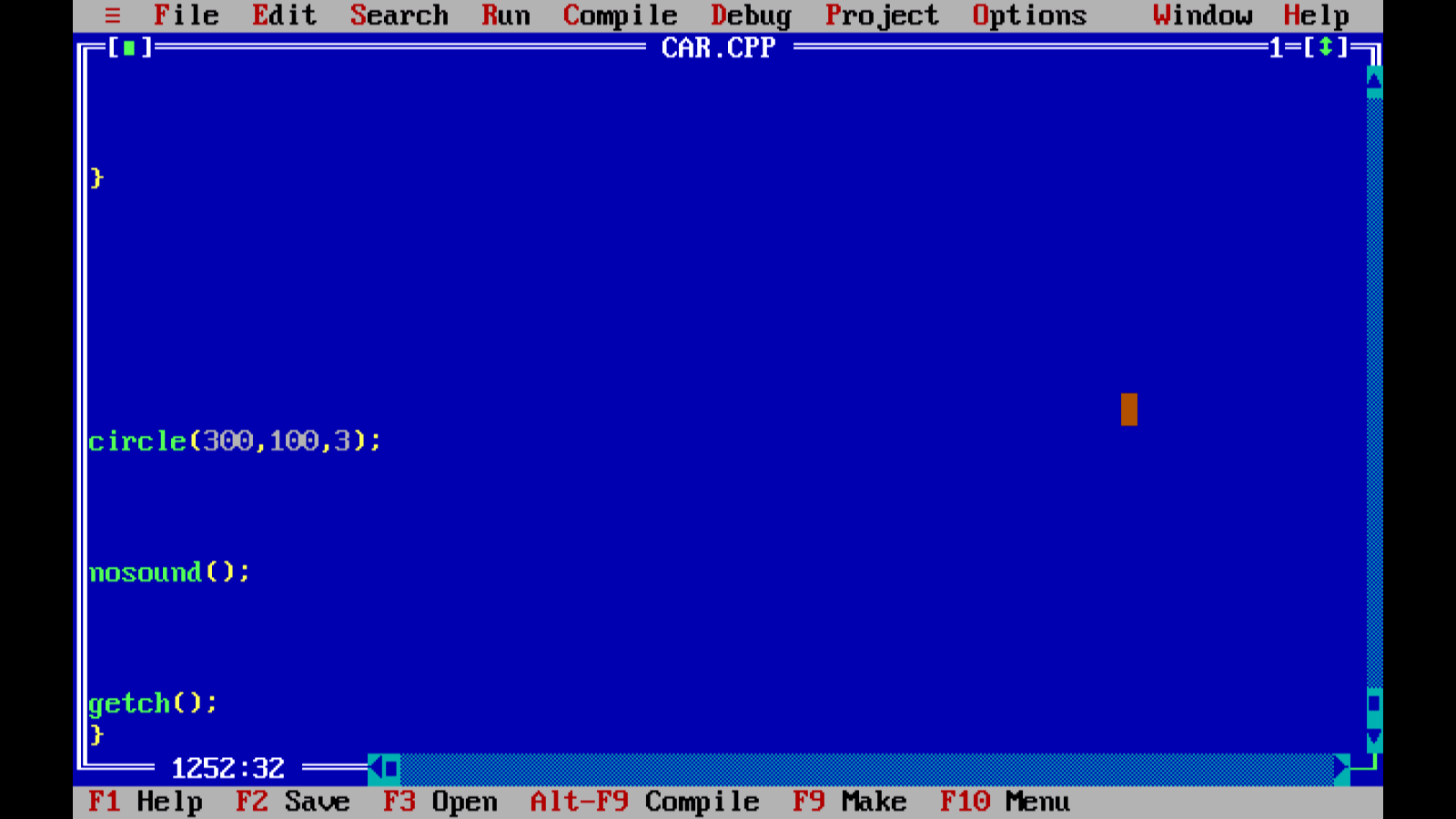
The program ends with a small graphical circle and a beep sound.

**Code:**

****







Output: