Report On

Car Animation

Submitted in partial fulfillment of the requirements of the Course project in Semester III of

Second Year Computer Engineering

by
Pratik Avhad (Roll No. 01)
Priyanka Bhandari (Roll No. 02)
Swarup Kakade (Roll No. 19)

Supervisor
Mrs. Sneha Yadav

University of Mumbai

Vidyavardhini's College of Engineering & Technology

Department of Artificial Intelligence and Data Science

(2023-24)

Vidyavardhini's College of Engineering & Technology
Department of Artificial Intelligence and Data Science

CERTIFICATE

This is to certify that the project entitled “Car Animation” is a bonafide work of "Pratik Avhad (Roll No. 01), Priyanka Bhandari (Roll No. 02), Swarup Kakade (Roll No. 19)" submitted to the University of Mumbai in partial fulfillment of the requirement for the Course project in semester III of Second Year Artificial Intelligence and Data Science engineering.

Supervisor

Mrs. Sneha Yadav

Dr. Tatwadarshi P. N.
Head of Department

Abstract

In our course project titled "Car Animation" we delved into the realm of computer graphics, aiming to create an engaging and visually stimulating experience. The project encompasses various elements of graphic programming, animation, and interactivity, demonstrating the application of fundamental concepts in a practical context.

The project begins by introducing the audience to basic graphic primitives, illustrating their usage through the implementation of a colorful and dynamic landscape. Key components such as circles, rectangles, and ellipses are utilized to create a visually appealing backdrop. Building upon these foundational elements, the project explores advanced graphical concepts such as filling shapes with vibrant colors, creating animated objects, and implementing user interactions.

One of the highlights of our project is the development of an animated scene featuring a moving car, a radiant sun, and scenic trees. Through careful manipulation of graphical elements and animation techniques, we brought the static environment to life. The car, adorned with bright colors, moves smoothly across the screen, capturing the viewer's attention. Simultaneously, the sun shines brightly, and trees sway gently, adding a touch of realism and dynamism to the scene. User interaction is incorporated, allowing viewers to control the animation speed, providing an immersive experience.

Additionally, the project explores the mathematical concepts behind graphical transformations, including rotation and translation. Through the implementation of these transformations, objects within the scene seamlessly move and rotate, enhancing the overall visual aesthetics. Moreover, the utilization of trigonometric functions for animating the sun's rays demonstrates the integration of mathematics and computer graphics.

Furthermore, the project emphasizes the significance of optimization and efficiency in graphical programming. Techniques such as double buffering and frame rate control are employed to ensure smooth and flicker-free animations. This optimization not only enhances the visual quality of the project but also provides a seamless user experience.

In summary, our course project showcases the fusion of creativity, technical proficiency, and mathematical concepts in the domain of interactive graphics. By combining fundamental graphical primitives, animation techniques, user interactions, and mathematical computations, we have successfully created an immersive and visually captivating experience. This project serves as a testament to the potential of computer graphics in creating interactive and engaging digital environments.

Table of Contents

The Block Diagram:

Working:

1. Initialization: Graphics system is initialized. Text "Press any key to view the moving car" is displayed.
2. User Input: Program waits for a key press.
3. Drawing Car Elements: A yellow circle is drawn as the car's front. A brown rectangle represents the car's body. Two wheels are drawn with green filling. 4. Animation Loop: The car moves horizontally across the screen in increments. Car elements are re-drawn with updated positions in each iteration. A delay of 100 milliseconds between iterations creates a smooth animation. Animation continues until the car reaches a specific position (x=420).
5. Clearing and Redrawing: Viewport is cleared, and car elements are redrawn in updated positions.
6. Completion: Animation stops when the car reaches x=420. Screen is cleared after a brief delay (500ms).
7. Closure: Graphics system is closed.

In essence, this project showcases a basic animation where the car moves horizontally, providing a visual demonstration of graphics programming.

Description

1. Header Files:

The code includes necessary header files like `graphics.h`, `dos.h`, and `conio.h` for handling graphics, input/output, and DOS functions respectively.

2.Function Definitions:

- `drawSun(int x, int y)`: This function draws a yellow sun with rays emanating from it. It uses trigonometric functions (cosine and sine) to calculate the end points of the rays based on the sun's position (x, y) and angle.

- `drawTree(int x, int y)`: This function draws a tree with a brown trunk and green foliage. It utilizes graphical functions to draw rectangles and ellipses representing the trunk and leaves of the tree.

3.Main Function:

- Initialization: - Initializes graphics mode using `initgraph()`.
  - Displays a message prompting the user to press any key to view the moving car. - Waits for user input using `getch()`.
- Drawing Initial Scene:
  - Draws a yellow circle representing the sun using `circle()` and `fillellipse()` functions.
  - Calls the `drawTree()` function to draw a tree on the screen.
- Animating the Car
  - Enters a loop where the car is animated across the screen:
  - Draws the car using rectangles and circles, changing its position in each iteration to simulate movement.
  - Utilizes `clearviewport()` to erase the previous frame, creating the illusion of movement.
  - Draws the sun and tree in each iteration to maintain their presence throughout the animation.
  - Delays for a short period using `delay()` to control the animation speed. - User Interaction:
  - After the animation is complete, waits for user input again using `getch()`. - Clears the screen, ending the graphical display.

4. Graphics Functions:

- The code uses various graphics functions such as `rectangle()`, `circle()`, `fillellipse()`, `setcolor()`, `floodfill()`, and `line()` to draw shapes, set colors, and fill areas.

- The `clearviewport()` function erases the current viewport, allowing for smooth animation.

5.Optimization and Closing:

- Double buffering and frame rate control techniques are used for smooth and flicker-free animations.

- The graphics mode is closed using `closegraph()` after the animation is complete.

In summary, the code initializes a graphics window, draws an initial scene with a sun and tree, animates a car moving horizontally across the screen, and allows user interaction to control the animation speed. Throughout the code, various graphical functions are utilized to create a visually appealing and interactive experience.

Module Description

The functionality is divided into two distinct modules: Graphics Functions and Main Program.
1. Graphics Functions:
This module contains functions responsible for drawing various graphical elements on the screen. These functions utilize the graphics.h library to create shapes, colors, and animations.

- drawSun(int x, int y)`:
- Draws a yellow sun with rays emanating from it.
- Parameters: `x` and `y` represent the coordinates of the sun's center.

-drawTree(int x, int y):
- Draws a tree with a brown trunk and green foliage.
- Parameters: `x` and `y` represent the coordinates of the tree's position.

2. Main Program:
The main program module orchestrates the overall flow of the application. It initializes the graphics environment, handles user interactions, and animates the scene. - Initialization:
- Initializes the graphics mode using `initgraph()`.
- Displays a message prompting the user to press any key to view the moving car. - Waits for user input using `getch()`.

- Drawing Initial Scene:
- Draws the initial scene, including the sun and tree, using the `drawSun()` and `drawTree()` functions.
-Animating the Car:
- Enters a loop where the car is animated across the screen:
- Draws the car, changes its position in each iteration to simulate movement, and utilizes `clearviewport()` to erase the previous frame.
- Draws the sun and tree in each iteration to maintain their presence throughout the animation.
- Delays for a short period using `delay()` to control the animation speed. - User Interaction:
- After the animation is complete, waits for user input again using `getch()`. - Clears the screen, ending the graphical display.

- Graphics Functions:
- Utilizes various graphics functions such as `rectangle()`, `circle()`, `fillellipse()`, `setcolor()`, `floodfill()`, and `line()` to draw shapes, set colors, and fill areas. - Uses `clearviewport()` to erase the current viewport, allowing for smooth animation. - Optimization and Closing:
- Implements double buffering and frame rate control techniques for smooth and flicker-free animations.
- Closes the graphics mode using `closegraph()` after the animation is complete.

Brief description of software & hardware used and its programming:

The provided code is written in C and utilizes the Turbo C/C++ compiler, which is a popular integrated development environment (IDE) for C and C++ programming. Here's a brief description of the software, hardware, and programming language used:

Software:

1. Turbo C/C++ Compiler:

Description:Turbo C/C++ is a compiler and integrated development environment (IDE) for the C and C++ programming languages. It was widely used in the MS-DOS and early Windows environments. Turbo C/C++ provides a simple and user-friendly interface for writing, compiling, and executing C and C++ programs. Features: Turbo C/C++ provides a text editor, compiler, linker, and debugger, allowing developers to write and test their programs within a single environment.

Note: While Turbo C/C++ was popular in the past, more modern and updated compilers and IDEs are available today, such as GCC (GNU Compiler Collection), Visual Studio, and Code::Blocks.

2. Graphics Library:

Description: The code uses the graphics.h library, which is specific to Turbo C/C++. This library provides functions for graphics programming, allowing developers to create graphical applications, draw shapes, and handle graphical elements. Functions Used: Functions like `initgraph()`, `circle()`, `rectangle()`, `floodfill()`, and others from the graphics.h library are used in the code to draw and manipulate graphical elements.

Hardware:

The specific hardware requirements for running Turbo C/C++ are generally minimal. It can run on older hardware configurations and does not demand significant computing resources. Any standard desktop or laptop computer with a compatible MS-DOS or Windows operating system could run the Turbo C/C++ compiler and execute the code.

Programming Language:

C Programming Language:

Description: The code is written in the C programming language. C is a procedural programming language developed in the early 1970s. It is known for its efficiency, flexibility, and low-level memory manipulation capabilities. C is widely used in systems programming, embedded systems, and game development due to its performance and control over hardware. Features Used: The code makes use of C language features such as functions, loops, conditional statements, mathematical calculations, and user-defined functions to create the graphical animations and interactions.

In summary, the code is written in C programming language and utilizes the Turbo C/C++ compiler and graphics.h library for creating graphical animations. It can run on standard desktop or laptop computers without requiring high-end hardware resources. However, it's important to note that Turbo C/C++ and graphics.h are considered outdated in the context of modern software development, and developers often use more contemporary tools and libraries for graphics programming in today's programming landscape.

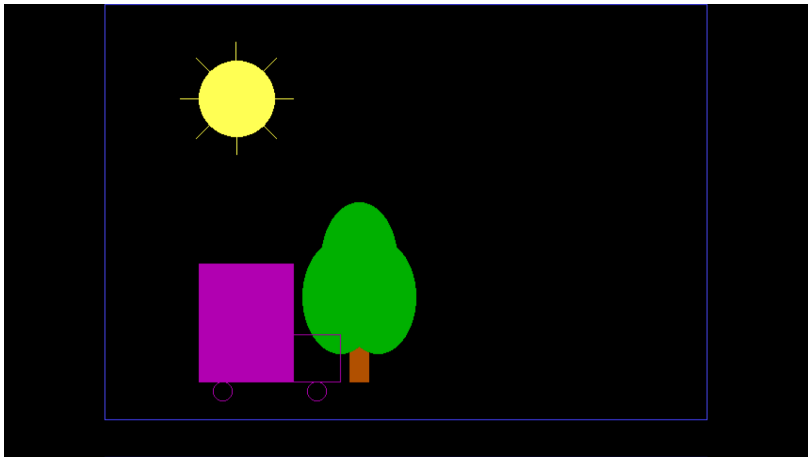
Code

```
#include <graphics.h>
#include <dos.h>
#include <conio.h>
#include<math.h>
void drawSun(int x, int y) {
 setcolor(YELLOW);
  setfillstyle(SOLID_FILL,YELLOW);
  fillellipse(x,y,40,40);
  int rayLength =60;
  for(int angle=0;angle<360;angle+=45){
  int endX=x+rayLength * cos(angle*3.14/180);
   int endY=y+rayLength * sin(angle*3.14/180);
   line(x,y,endX,endY);
   }
   }
 void drawTree(int x,int y){
 setcolor(BROWN);
 rectangle(x+60,y+100,x+80,y+200);
 setfillstyle(SOLID_FILL,BROWN);
 floodfill(x+70,y+150,BROWN);
 setcolor(GREEN);
  setfillstyle(SOLID_FILL,GREEN);
  fillellipse(x+70,y+70,40,60);
  fillellipse(x+90,y+110,40,60);
  fillellipse(x+50,y+110,40,60);
  }
int main()
 {
   int i, j = 0, gd = DETECT, gm;
   initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");
   settextstyle(DEFAULT_FONT,HORIZ_DIR,2);
   outtextxy(25,240,"Press any key to view the moving car");
   getch();
   setviewport(0,0,639,440,1);
   setcolor(9);
   rectangle(0,0,639,440);
   setcolor(YELLOW);
   circle(80,80,40);
   drawTree(200,200);
   for( i = 0 ; i <= 420 ; i = i + 10, j++ )
   {
          setcolor(j);
          rectangle(50+i,275,150+i,400);
          rectangle(150+i,350,200+i,400);
          setfillstyle(SOLID_FILL,j);
          floodfill(100+i,300,j);
          circle(75+i,410,10);
```

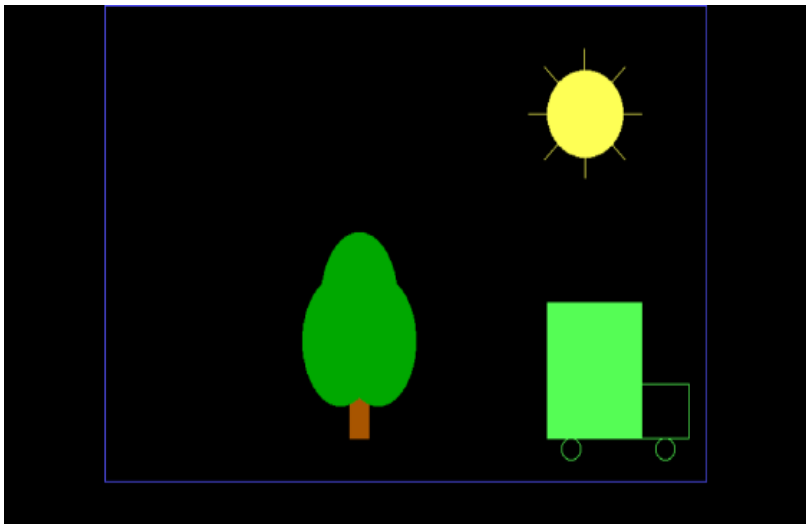
```
circle(175+i,410,10);
delay(100);
if( i == 420 )
break;
clearviewport();
setcolor(9);
rectangle(0,0,639,440);
setcolor(YELLOW);
drawTree(200,200);
drawSun(100+i,100);
}
getch();
cleardevice();
delay(500);
closegraph();
return 0;
}
```

Result:

Before:-



After:-



Conclusion

The provided code serves as a compelling demonstration of interactive graphics programming, incorporating fundamental concepts of shapes, colors, animations, and user interactions. Through the implementation of a dynamic scene featuring a moving car, radiant sun, and scenic trees, the project showcases the integration of graphical elements and mathematical computations. By employing optimization techniques, the code ensures a smooth and visually pleasing experience for the viewer. This project highlights the creative potential of computer graphics, making it an excellent learning resource for students and enthusiasts interested in the field of interactive visual programming.

References

1. Akenine-Moller, T. and E. Haines (2002) Real-Time Rendering, A.K. Peters.

2. (Order from amazon , order from Barnes and Noble , compare at bigwords , compare at CampusBooks4Less , order from Chegg , or search eFollett)

3. Angel, E. (2005) Interactive Computer Graphics: A Top-Down Approach with OpenGL , Addison Wesley.

4. (Order from amazon , order from Barnes and Noble , compare at bigwords , compare at CampusBooks4Less , order from Chegg , or search eFollett)

5. Farin, G. and D. Hansford (2004) Practical Linear Algebra: A Geometry Toolbox, AK Peters.

6. (Order from amazon , order from Barnes and Noble , compare at bigwords , compare at CampusBooks4Less , order from Chegg , or search eFollett)

7. Hearn, D. and M.P. Baker (2003) Computer Graphics with OpenGL, Prentice Hall.

8. (Order from amazon , order from Barnes and Noble , compare at bigwords , compare at CampusBooks4Less , order from Chegg , or search eFollett)