3D-Car Animation

CS352: Computer Graphics & Visualization Lab

Project Report

Course Instructor:
Dr. Somnath Dey

Submitted By:
Annavarapu Bhanuteja– 200001004

Katta Jayanth Kumar – 200001034

Shaik Wanhar Aziz - 200001072

Introduction:

3D-car animation, as the name suggests, is about a 3D-simulation of a car. In this project we are planning to implement how a car looks, its basic functionalities and a simple background so that it looks exactly like it's in the real world. When it comes to the looks of a car, it should have a color, a few windows, two pairs of wheels, an exhaust and headlights. When it comes to basic functionalities of a car it means moving back and forth, rotation of the wheels while moving the car, turning on and turning off the headlights and many more things that can be done as per user's choice. When it comes to background, we mean the surroundings of the car, like a green field, a road and some blocks of dividers at the edges of the road, how the car looks in day-mode and night-mode are also included. To implement all these features we used the functionalities of pre-built functions in OpenGL. OpenGL is a powerful graphics library in c++ which helps in creating 3D animations and visual effects. We aim to leverage the capabilities and functionalities of OpenGL to create a realistic and interactive car model that the user can control. Since OpenGL allows the user to interact with the car model, we will add some miscellaneous features like changing the color of the car, toggling between day and night modes, toggling between headlights on and off modes. There will be a menu option where we can implement this functionality so that the users can choose what to do and how to interact with the car. Now that we are talking about menu options, let's also discuss the sub-menu options in the dropdown menu. The only dropdown menu is of color, here we'll have various color options like: blue, red, green, black, yellow, gray. These options will be present in the user-interface and our user must have a mouse to select these features. All these features mentioned above can be used only with the help of a mouse and a keyboard.

The Technical Specifications for the project 3D Car Animation is as follows :

Hardware Specifications:

1. 4 GB ram minimum.
2. minimum of 2.1GHz processor.
3. Linux operating system.

Software Specifications:

1. Text Editor.
2. User Interface.
3. Libraries.

Text Editor:

As this is a 3D-simulation, we are planning to use c++ as our primary programming language and since we have to code this somewhere, we are planning to use Visual Studio Code as our primary text editor to write and edit our code. VS code is a user-friendly platform where we can easily code and see the output of our car easily.

User Interface:

The user interface shows the output of the car and a menu for the user so that he/she can change the background, toggle between various modes and do various tasks with the car, to use these features the user must use mouse and keyboard.

Libraries Used:

1. OpenGL libraries: GL/glut.h,
2. math library: math.h
3. standard library: stdlib.h, stdio.h

With the help of the above libraries we can implement all the above functionalities of a car and simulate a 3D car in a 2D display. Let's talk more about the libraries in general.

GL/glut.h: GL/glut.h is a header file in the OpenGL Utility Toolkit (GLUT) library that provides a set of functions for creating and managing windows, handling events such as mouse and keyboard input, and rendering 3D graphics in OpenGL. The GLUT library simplifies the process of creating OpenGL applications and makes it easier to develop portable and platform-independent code. The GL/glut.h header file contains prototypes for various functions, including glutInit(), glutCreateWindow(), glutDisplayFunc(), and glutMainLoop(), which are essential for creating an OpenGL window, setting up a display function to render graphics, and running the main loop to handle events and update the display. To summarize this, GL/glut.h is a crucial library for developing OpenGL applications that simplifies window creation, event handling, and rendering of 3D graphics.

math.h:

The math.h library is a standard C++ header file that provides a set of mathematical functions and constants. It is included in the C++ standard library. Some of the commonly used functions in the math.h library that we are planning to use are: trigonometric functions such as sin(), cos(), and tan() and algebraic functions such as sqrt(), pow(). The library also includes constants such as pi and e. In addition to the basic mathematical functions, the math.h library also provides functions for rounding, random number generation, and complex arithmetic.

Standard library:

The stdio.h and stdlib.h header files are standard C++ libraries that provide fundamental functionalities for input/output and general-purpose programming respectively. The stdio.h library provides input/output (I/O) functions for reading and writing data to the console. We are going to use this library to implement the general things like loops, statements, usage of data-types etc. These functions are essential for most console-based programs, and are used extensively for debugging, logging, and user interaction. The stdlib.h library provides a wide range of general-purpose functions, including memory allocation and deallocation, string manipulation, and conversion of values between different data types. Some of the commonly used functions in this library include atoi(), atof(), and exit().

Execution of the project:

To execute the project we must run the following command in the command prompt,
<compiler used for c++> <file-name> -o <executable-file-name(as per user)> <libraries with which we want the output> (-lglut -lGLU -lGL)
as per the above mentioned manner the command we used was: g++ car.cpp -o car -lglut -lGLU -lGL
After the compilation process is done, we'll receive an executable file, which we can find in our project directory. We now need to run this executable file to see our output. To see our car we must run the following command in the command prompt.
./<executable file name>
In our case it is ./car
After running this command we can see the car as output. As mentioned above the user has the liberty to interact with this car using the menu feature. To access this menu feature the user must have a mouse and must click the right-click button in the mouse, once done, he can see the menus and submenus which consists of a list of various features by which he/she can interact with the car. The menu contains the following:

1. car model mode
2. car driving mode
3. wheel effect
4. lighting on
5. lighting off
6. car colors
    a. blue
    b. red
    c. green
    d. black
    e. yellow
    f. gray
7. day mode
8. night mode

The User can choose among anything from the above options and as per the selected option from the menu the car's functionality will change. To select a particular option the user must click on the left-click button on the mouse and he can then visualize the car with that particular functionality. With this the mouse functionality is done. Now let's talk about keyboard functionality. We are going to use the keyboard in various situations like movement of the car, rotation of wheels, geo-metric rotation of the car w.r.t to various axes like X, Y, Z axes respectively, scaling-up and scaling-down the car, moving the car up and down in the vertical direction and many more functions can be done using the keyboard. Here we'll see what each key does specifically.

The keys used to interact with the car are:

- If we press the x key the car will rotate about X-Direction by 5 degrees Anticlockwise.
- If we press the X key the car will rotate about X-Direction by 5 degrees Clockwise.
- If we press the y key the car will rotate about Y-Direction by 5 degrees AntiClockwise.
- If we press the Y key the car will rotate about Y-Direction by 5 degrees Clockwise.
- If we press the z key the car will rotate about Z-Direction by 5 degrees Anti Clockwise.
- If we press the Z key the car will rotate about Z-Direction by 5 degrees Clockwise.
- If we press the u key the car will move in Y-Direction by 0.2 units above .(moving up)
- If we press the U key the car will move downwards in Y-Direction by 0.2units.(moving down)
- If we press the f key the car will move in positive Z-Direction by 0.2 units. (moving towards the user)
- If we press the F key the car will move in negative Z-Direction by 0.2 units. (moving towards the screen)
- If we press the a key the car will move in positive X-Direction by 0.2 units . (moving right)
- If we press the d key the car will move in negative X-Direction by 0.2 units. (moving left)

Functionalities Implemented:

The many functionalities of the simulated car are: movement of the car, rotation of the car, rotation of the wheels, toggle of the headlights, color of the car. The functionalities of the background are: day mode, night mode, green background, road, brown bricks at the edges of the road.
Before we deep-dive into the functionalities of the car, we must see how we are going to actually initialize OpenGL and leverage its functionality of rendering 3D graphics. The function that actually does this initialization is: initGL() function and transform() function.

```
GLvoid InitGL(GLfloat Width, GLfloat Height)
{
   glClearColor(1.0, 1.0, 1.0, 1.0);
   glLineWidth(2.0);
   Transform(Width, Height);
   t = gluNewQuadric();
   gluQuadricDrawStyle(t, GLU_FILL);
}
```

The InitGL() function is used to initialize the graphics rendering environment. It takes two parameters, Width and Height, which represent the dimensions of the viewport in pixels. The first line of the function, glClearColor(), sets the color used to clear the frame buffer. In this case, it sets it to white, with all color components set to 1.0. The next line, glLineWidth(), sets the width of lines drawn using OpenGL. In this case, it sets it to 2.0 pixels. The following line, Transform(), is a user-defined function that sets up the projection matrix for 3D rendering. The next line creates a new quadric object using gluNewQuadric(). Quadrics are primitive shapes in OpenGL that include spheres, cones, and cylinders. The following line, gluQuadricDrawStyle(), sets the draw style of the quadric. In this case, it is set to GLU_FILL, which means the object will be filled with a solid color. The next three lines enable lighting in the scene, specify the properties of the light source, and set its position. The first line, glEnable(GL_LIGHTING), enables lighting calculations in the scene. The second line, glEnable(GL_LIGHT0), enables the first light source in the scene. The next four lines specify the properties of the light source, including its ambient, diffuse, and specular colors, and its position in the scene. Overall, the InitGL() function is required for setting up the graphics rendering environment, including the projection matrix, quadrics, and lighting sources.We called it once at the beginning of the program to initialize the graphics rendering environment.
Now that the rendering environment is initialized we'll see how the menu option actually works and how it helps users to interact with the car.

Menu-
We can navigate through different functionalities using our menu which is created using the glutCreateMenu() function. Menu items are added using the glutAddMenuEntry() function. The colors menu is added as a submenu using the glutAddSubMenu() function. We are enabling this menu using right click which is added using the glutAttachMenu(GLUT_RIGHT_BUTTON) function.

Car Model- We used the glBegin(GL_QUADS) function to make quadrilaterals of the required sizes by mentioning the four vertices of the quadrilateral using glVertex3f() function.

We have made the car by making the front body, back body, windows, windshield, doors, wheels, exhaust.
Wheels were made by making tori using the glutSolidTorus() function which takes inner radius, outer radius, number of sides and number of rings needed to approximate as arguments.
Exhaust is made using the gluCylinder() function, it takes 5 arguments - a pointer to quadric object, base radius, top radius, height, number of slices and stacks.

Different viewing angles-

The car and scenery can be viewed from different angles. On pressing keyboard keys we are able to change the viewing angles, this is done through glutKeyboardFunc() function. When a keyboard event is recorded according to the case list, variables xangle, yangle, zangle are changed and the glRotatef() function is applied on them. It takes four arguments, the angle of rotation in degrees, the x, y, and z components of the vector representing the axis of rotation.

Driving mode-

Driving mode was implemented by using some global variable with the menu , so when we press that menu entry of driving mode the variable will activate the driving mode and OpenGL will redisplay our screen with the help of the function glutPostRedisplay() so that when we rendering now the car we will also render the green surroundings the road and a median by using the global variable as the condition variable, here we are creating green surroundings and the road using GL_QUADS . This is the way we implemented Driving mode.

Lighting-

Lighting was done by creating a light source with ambient light , diffusive light , specular light setting at positions and making sure that our car looks bright and we used lighting for the lights of the car . the code for the lighting is

```
glEnable(GL_LIGHTING);
   glEnable(GL_LIGHT0);
   GLfloat ambientLight[] = {0.2f, 0.2f, 0.2f, 1.0f};
   GLfloat diffuseLight[] = {0.8f, 0.8f, 0.8, 1.0f};
   GLfloat specularLight[] = {0.5f, 0.5f, 0.5f, 1.0f};
   GLfloat position[] = {1.5f, 1.0f, 4.0f, 1.0f};
   glLightfv(GL_LIGHT0, GL_AMBIENT, ambientLight);
   glLightfv(GL_LIGHT0, GL_DIFFUSE, diffuseLight);
   glLightfv(GL_LIGHT0, GL_SPECULAR, specularLight);
   glLightfv(GL_LIGHT0, GL_POSITION, position);
```

we initialize in the initGL function so that it will act like a light source for the car. in our code this is the only light source.

Ambient light, diffuse light, and specular light are components of light in OpenGL that affect how objects are illuminated in a 3D scene. Ambient light is general illumination in a scene that affects all objects equally. Diffuse light simulates the effect of light coming from a light source that is scattered uniformly in all directions causing an object to appear brighter. Specular light simulates the effect of light coming from a point source bouncing of a surface in a specific direction based on angle of incidence.

Wheel Effect-

When the car moves forward, the wheel rotates in anti-clockwise direction and when it goes back, the wheel rotates in clockwise direction. This effect is generated by using rotation and translation functions. When the key is pressed to move forward, the blocks on the boundary of the road are translated to move backward using the glTranslatef() function which takes distance in x,y,z direction to be translated as arguments. The spikes of the wheel are rotated by a small angle using glRotatef() function which generates the effect of the car being moved forward.

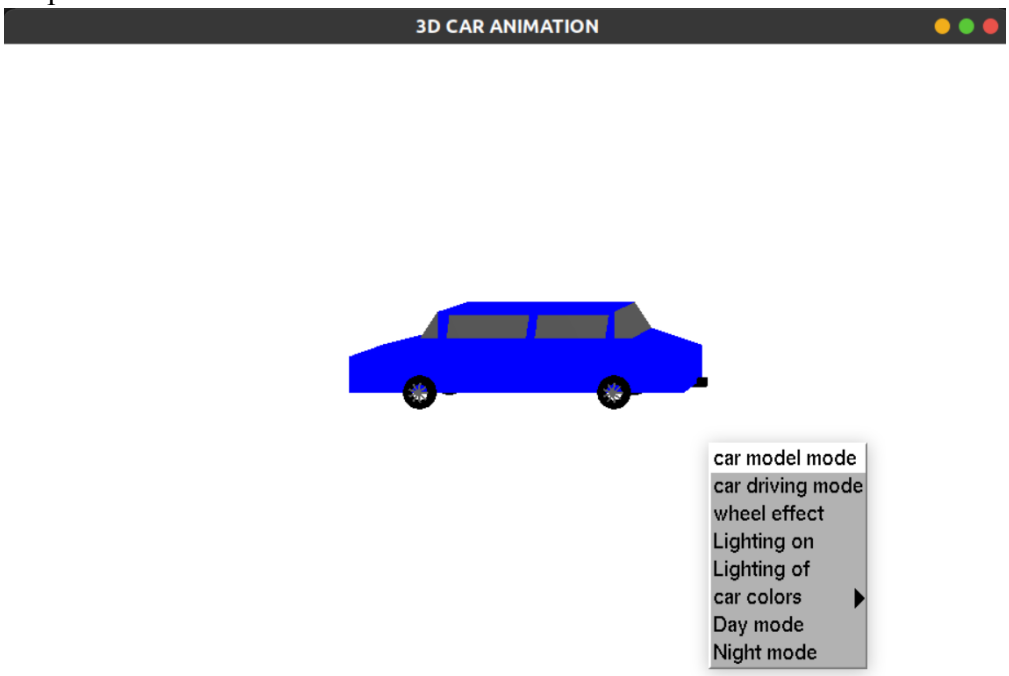
Night Mode-

Driving mode was implemented using some global variable with the menu, when the user presses the menu entry of night mode the variable will activate the night mode and OpenGL will redisplay our screen with the help of the function glutPostRedisplay().

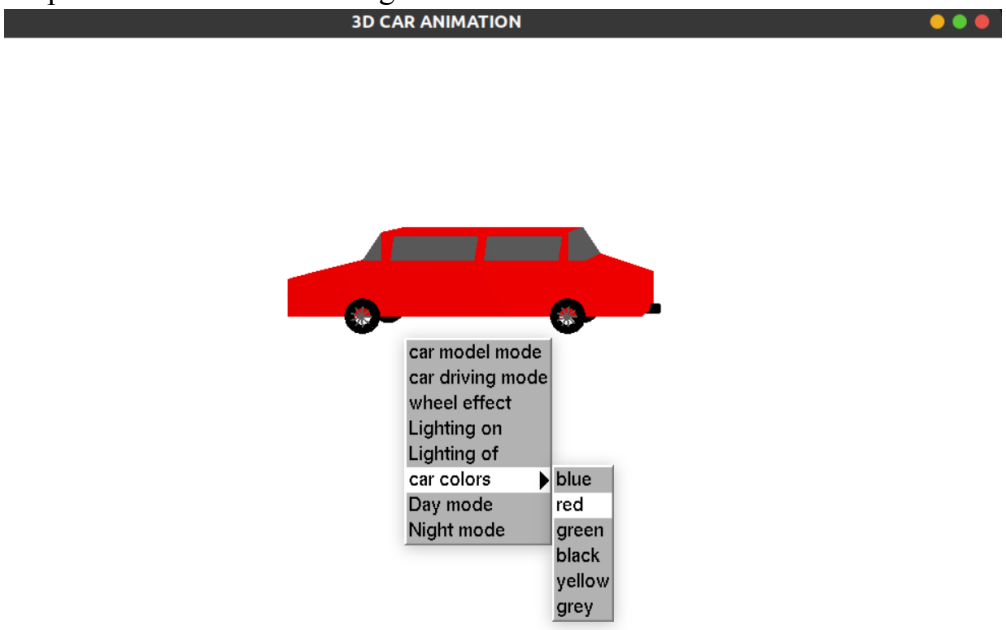
Car colors-
Car color was implemented by a color sub menu where we kept ids for each color and when user chooses a particular color via mouse that color's particular id's code is executed and via glutPostRedisplay() the car is re-displayed on the screen with the color the user choose.

Output Window:

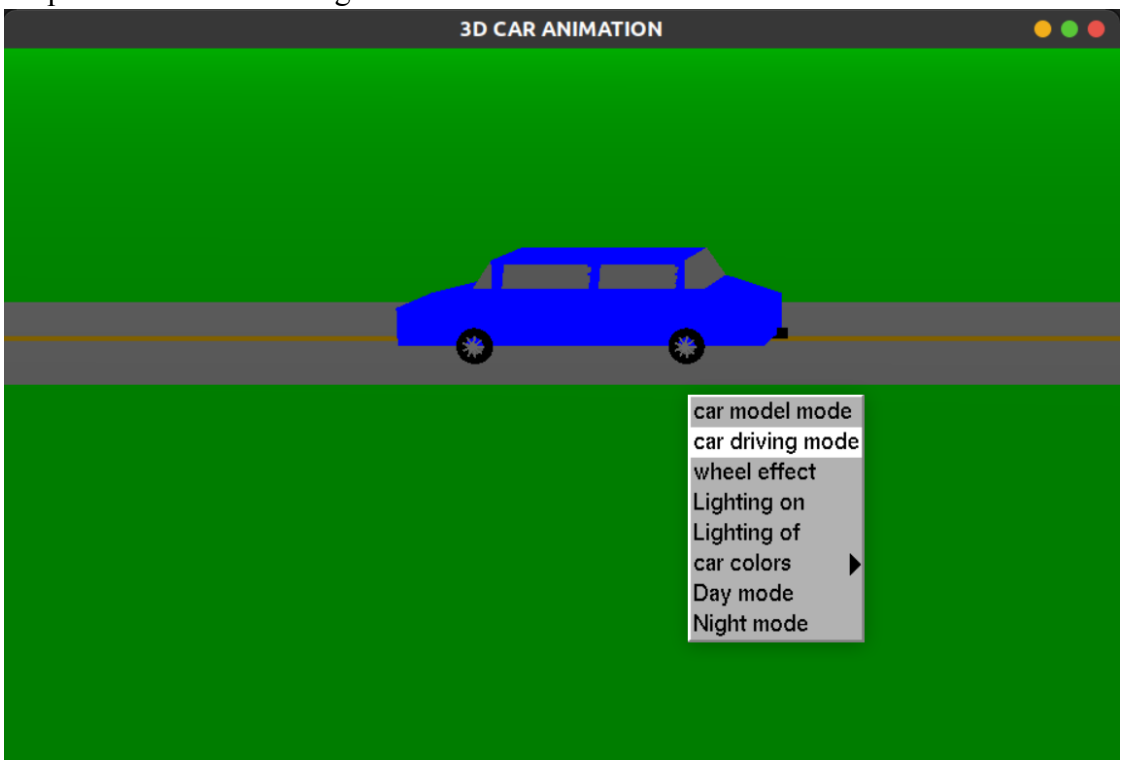
Output of car with menu:



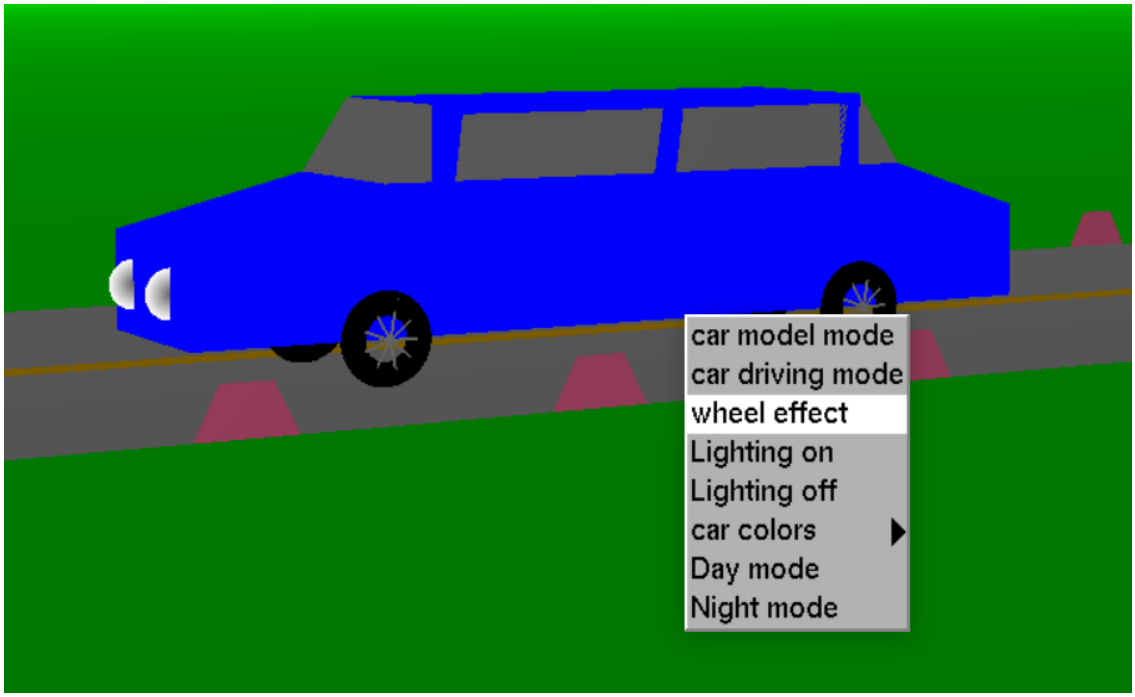
Output of car after color change from menu:



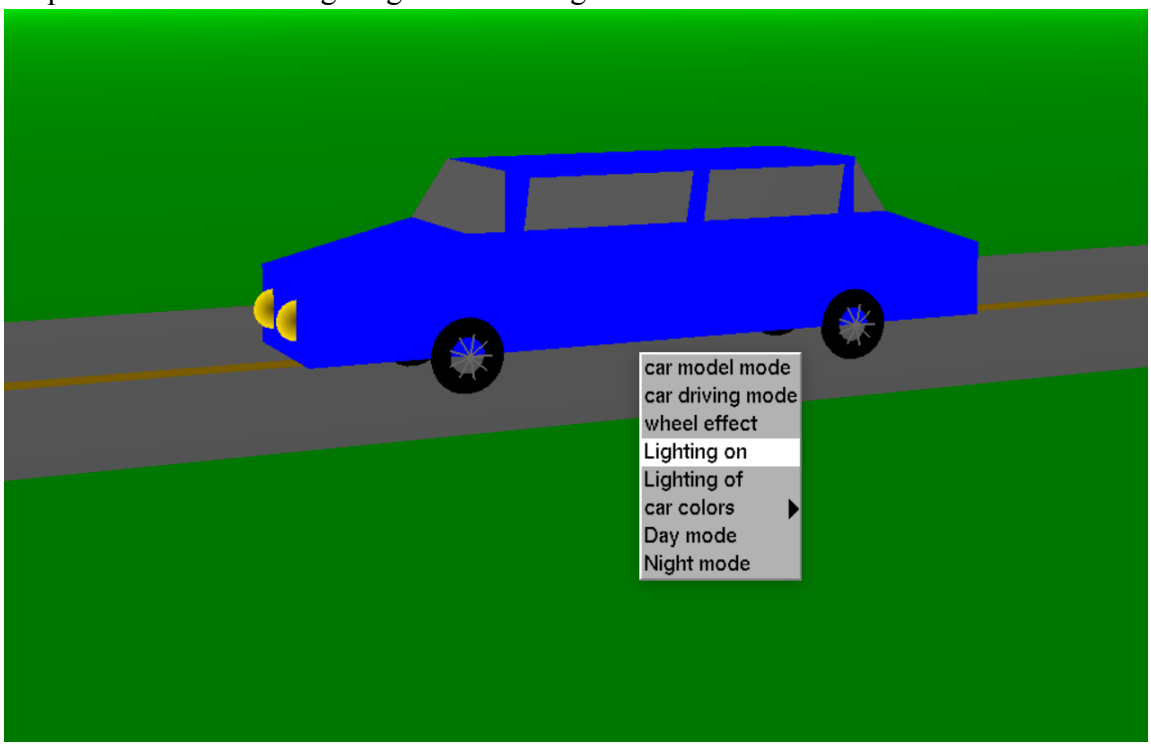
Output of the car in driving mode:



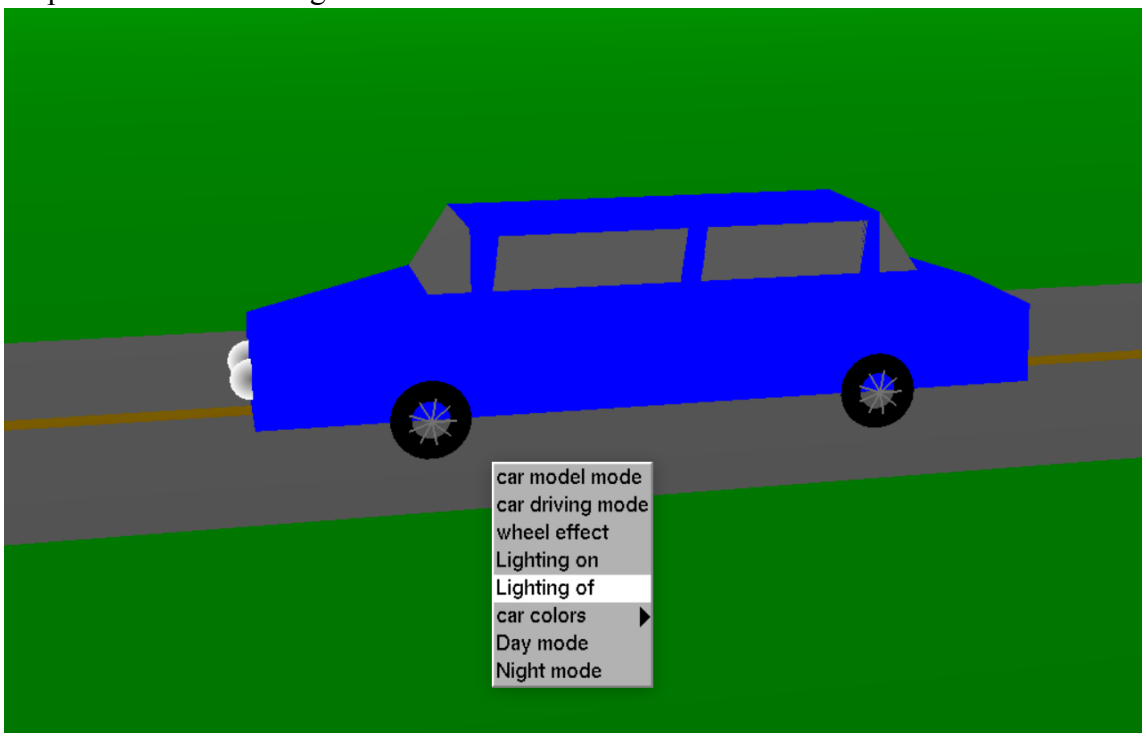
Output of the car in wheel effect:



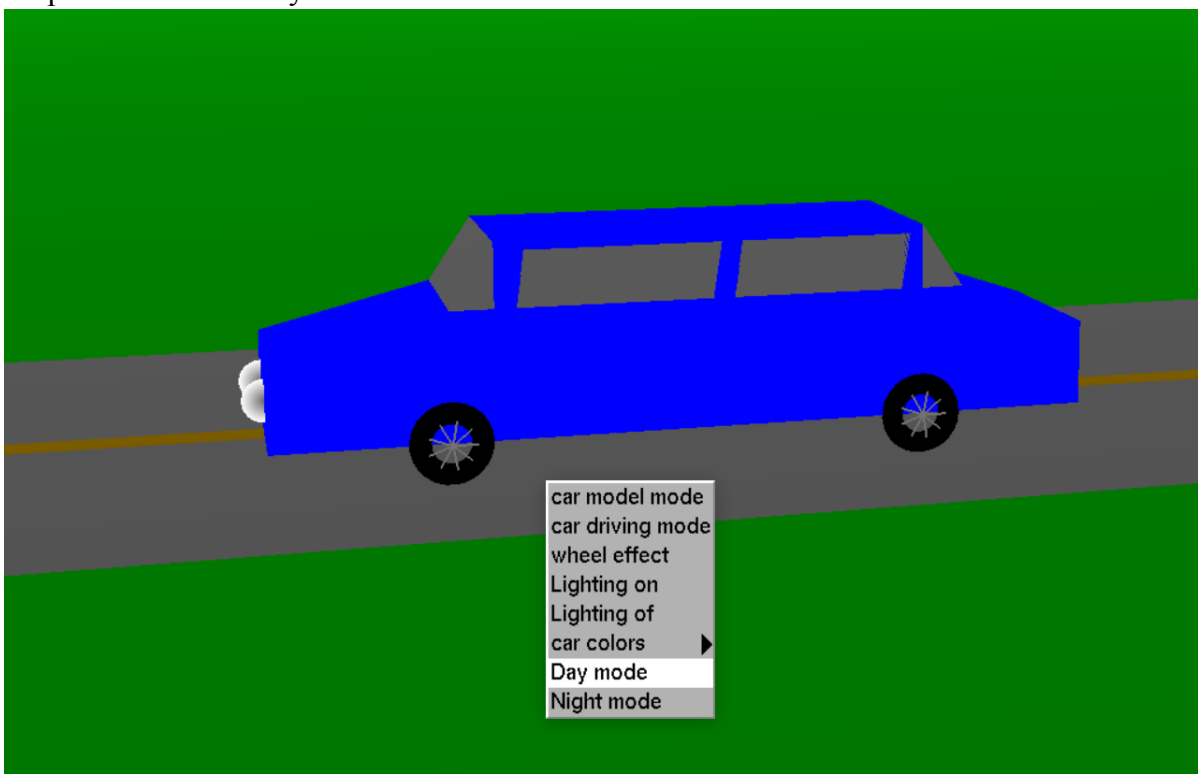
Output of the car when lighting of the headlights is on:



Output of the car after lights off:



Output of the car in day mode:



Output of the car in night mode: