Universidade Federal de São Paulo (UNIFESP)

Trabalho de Conclusão da Disciplina de Computação Gráfica

Thiago Henrique Leite da Silva, RA: 139920

São José dos Campos, 15 de fevereiro de 2022

Relatório

1. **O Projeto**

Para o trabalho final da disciplina, o projeto final desenvolvido foi a rua de uma cidade, com prédios ao fundo, dois veículos se movimentando, um carro e um ônibus, além de um sol que ilumina a cidade. Também temos na cidade um semáforo para controlar o trânsito. Ao todo foram mais de mil linhas de código implementadas. A seguir veremos como a implementação foi realizada, os requisitos solicitados no projeto e seu funcionamento.

2. **Requisitos**

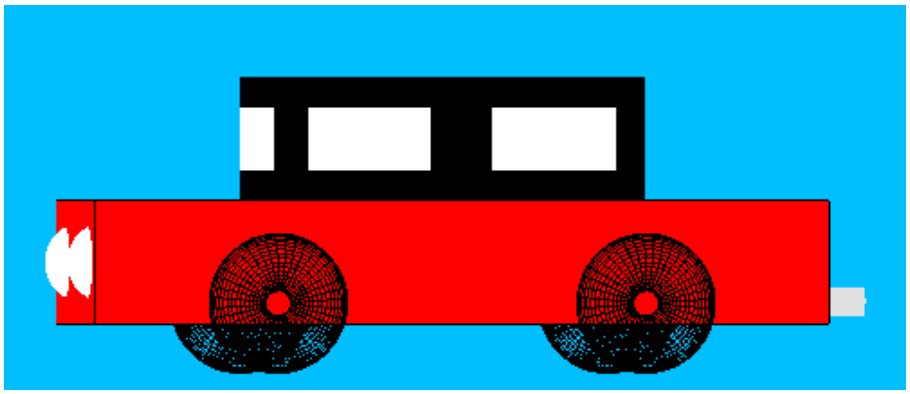
    a. **Deve conter pelo menos 4 objetos complexos que devem se mover de forma diferente.**

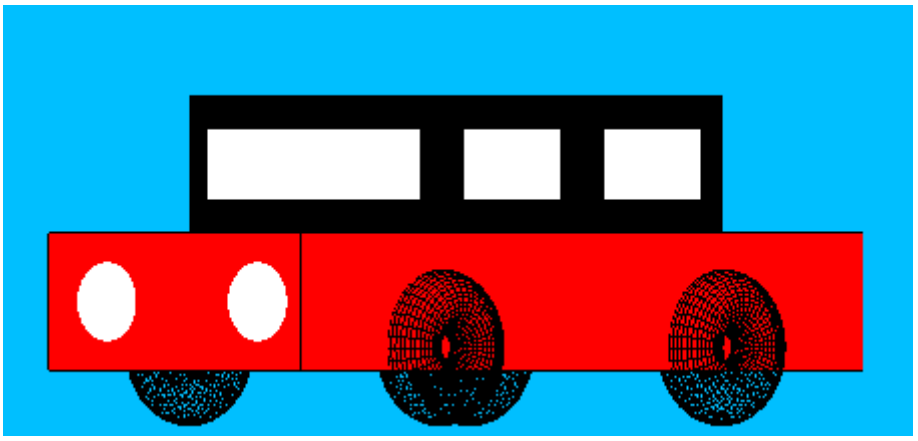
Os objetos complexos implementados no trabalho foram os seguintes:

- Carro

O carro foi construído a partir de uma composição de cubos, sólidos e wire, esferas, um cilindro para o escapamento, e as rodas que foram feitas com Wire Torus. As rodas não são sólidas para permitir uma melhor visualização delas girando, outra feature implementada ao carro. Foi um desafio inicialmente entender quais movimentações precisam ser feitas para girar o Torus sob seu próprio eixo e como uma roda de um carro de fato. Após diversos testes, cheguei a resposta, que é a exibida no trecho do código abaixo.

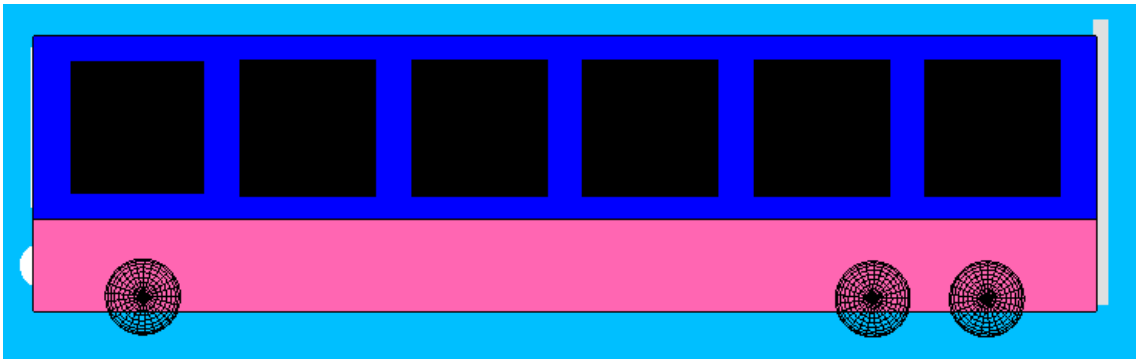
```
// Roda frontal direita
glPushMatrix();
    glTranslatef(0.90, -3.0, 0.55);
    glRotatef(90,1,0,0);
    glRotatef(90,0,0,1);
    glRotatef(angX,1,0,0);
    glRotatef(-90,0,1,0);
    glColor3fv(CL_BLACK);
    glScaled(0.5,0.5,0.5);
    glutWireTorus(0.28,0.40,40,40);
    glTranslatef(-0.90, +3.0, -0.55);
    glRotatef(theta, 0.0, 0.0, 1.0);
    glTranslatef(0.90, -3.0, 0.55);
glPopMatrix();
```

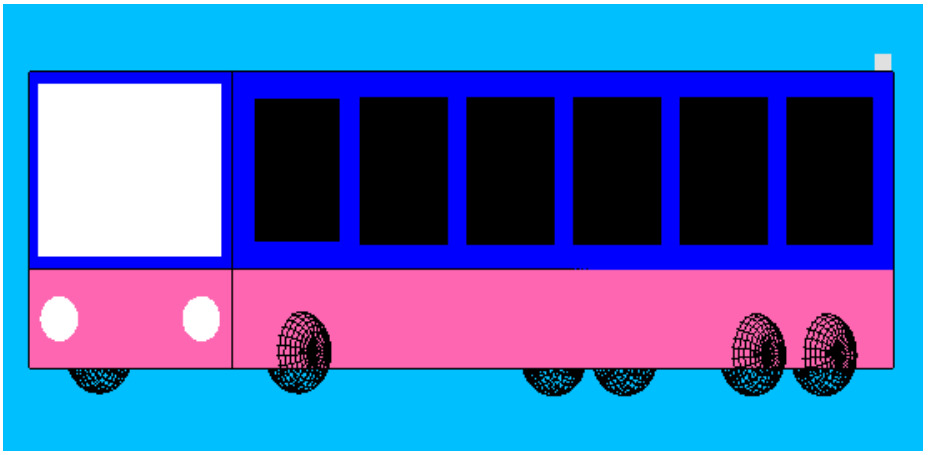
- 



- Ônibus

Em relação ao ônibus, o mesmo também foi construído com os mesmos elementos usados no carro, porém foi necessário muitas adaptações, além de aumentarmos a quantidade de rodas, janelas e o comprimento do veículo. Dessa vez o escapamento foi colocado na parte traseira do automóvel.



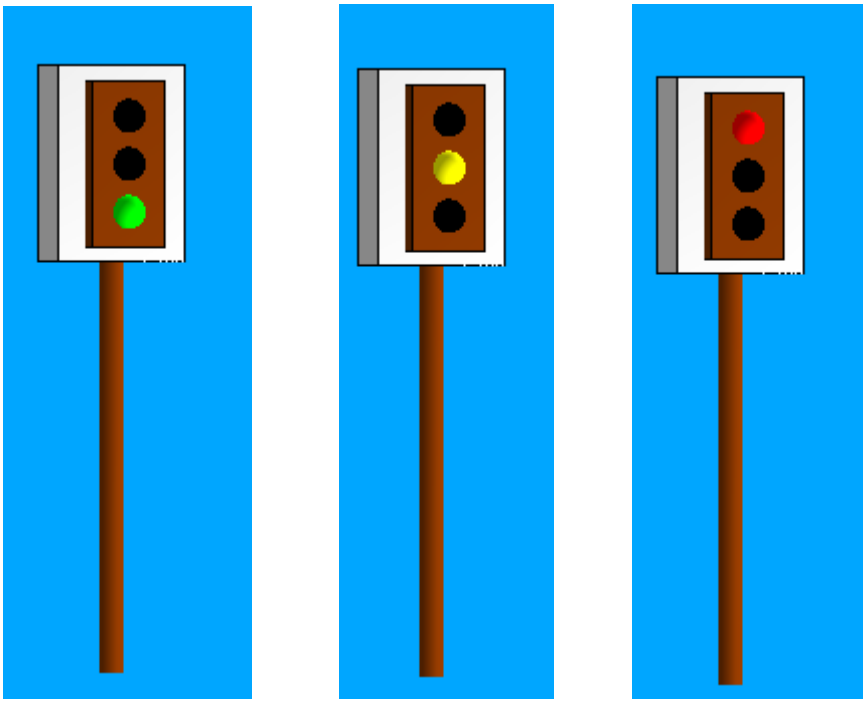


Abaixo temos um trecho do código para implementação do ônibus:

```
void build_bus()
{
    glPushMatrix();
        glMatrixMode(GL_MODELVIEW);
        glLoadIdentity();
        gluPerspective(90.0, 1.0, 1, 1);
        glOrtho(-7.0,7.0,-7.0,7.0,-7.0,7.0);
        glTranslatef(0,0,1.5);

        gluLookAt(
            0.0, 0.0, 1.0,
            0.0, 0.0, 0.0,
            0.0, 1.0, 0.0
        );

        glViewport(0, 0, 800, 800);

        glPushMatrix();
            glRotatef(9, 0, 1, 0);
            glTranslatef(-right, 0.5, -1.0);
            glScalef(1.3, 1.3, 1.3);

            // Carroceria
            glPushMatrix();
                glTranslatef(0, -2.8, -0.2);
                glColor3fv(CL_BLACK);
                glScalef(2.3301, 0.2005, 0.5);
                glutWireCube(4);
            glPopMatrix();

            glPushMatrix();
                glTranslatef(0, -2.8, -0.2);
                glColor3fv(CL_PINK);
                glScalef(2.33, 0.2, 0.5);
                glutSolidCube(4);
            glPopMatrix();

            // Teto
            glPushMatrix();
                glTranslatef(0, -1.6, -0.2);
                glColor3fv(CL_BLACK);
                glScalef(2.3301, 0.4005, 0.5);
                glutWireCube(4);
            glPopMatrix();
```

- Semáforo

O semáforo é uma composição de dois cubos, um para a caixa externa e outro para a caixa interna. Três esferas para os sinais, as mesmas trocam de cor com um comando do usuário, conforme veremos mais adiante. Por fim, a haste é feita por um cilindro. Abaixo vemos os três estados possíveis do semáforo, sendo eles a luz verde, amarela e vermelha acesas individualmente.



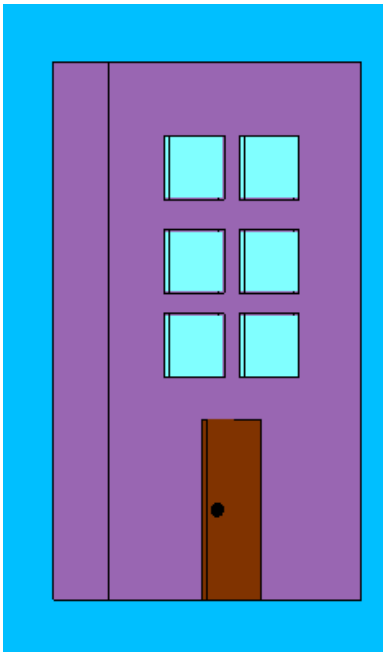
O principal desafio em relação ao semáforo foi pensar na melhor forma de alterar estes estados de forma que fizesse sentido no ambiente projetado. Além de colocar a iluminação nos sinais. Abaixo segue um trecho do código implementado.

```
// Posição inicial do semáforo
GLint greenSignal  = 1;
GLint yellowSignal = 0;
GLint redSignal    = 0;

GLfloat *colorGreenSignal  = CL_GREEN;
GLfloat *colorYellowSignal = CL_BLACK;
GLfloat *colorRedSignal    = CL_BLACK;
```

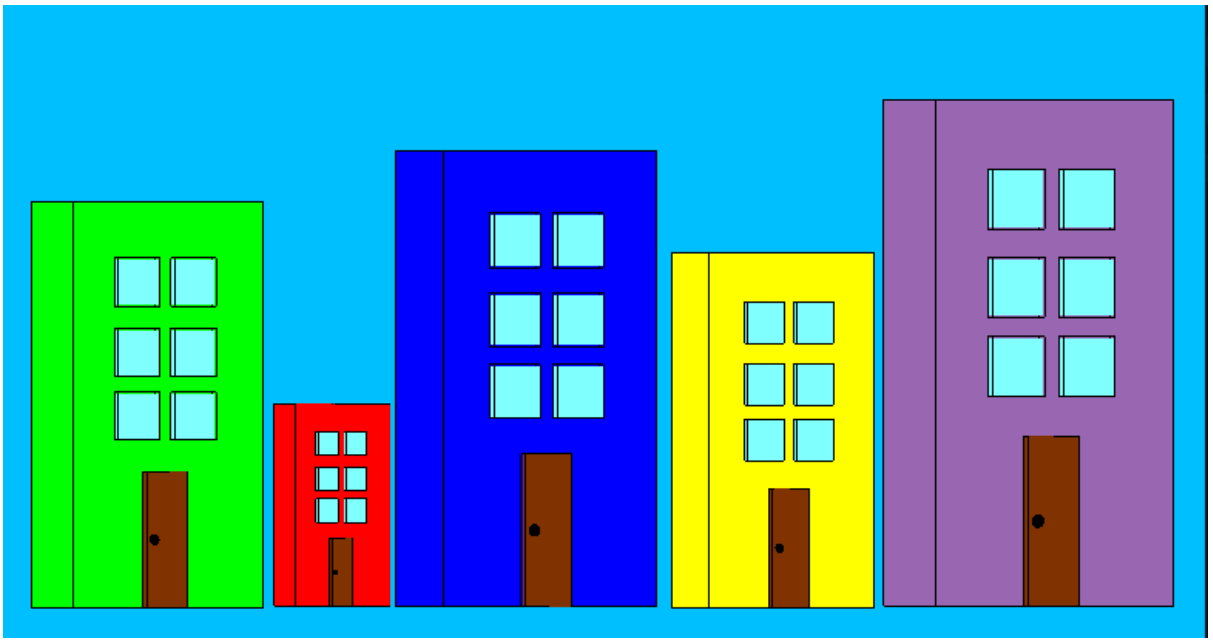
```
            // Sinal vermelho
            glPushMatrix();
                glTranslatef(0, 2.5, 0.3);
                glColor3fv(colorRedSignal);
                glutSolidSphere(0.25, 50, 50);
            glPopMatrix();

            // Sinal amarelo
            glPushMatrix();
                glTranslatef(0, 1.8, 0.3);
                glColor3fv(colorYellowSignal);
                glutSolidSphere(0.25, 50, 50);
            glPopMatrix();

            // Sinal verde
            glPushMatrix();
                glTranslatef(0, 1.1, 0.3);
                glColor3fv(colorGreenSignal);
                glutSolidSphere(0.25, 50, 50);
            glPopMatrix();
        glPopMatrix();
    glPopMatrix();
```

- Apartamento

O prédio é composto quase que exclusivamente por cubos, que foram divididos em: torre, janelas e porta. Por fim, temos uma esfera para a maçaneta da entrada do imóvel. O modelo criado para construirmos um prédio foi o mais genérico possível, para que eu pudesse criar vários deles sem duplicação de código.



Sendo assim, ao final conseguimos construir 5 prédios para preencher nossa cidade.



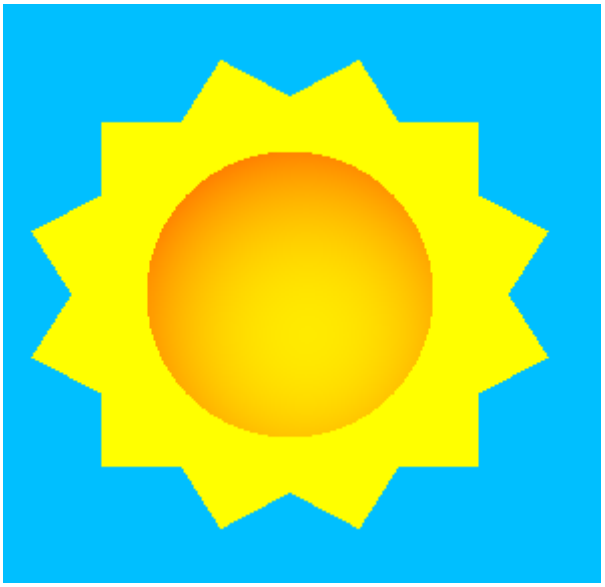
Trecho da função que constrói um prédio:

```
// Contorno da Torre
glPushMatrix();
    glTranslated(5, 0.96, -5.5);
    glRotatef(30,0,1,0);
    glColor3fv(CL_BLACK);
    glScaled(0.7,1.3,0.3);
    glScalef(1.11,1.104,1);
    glutWireCube(4);
glPopMatrix();

// Torre
glPushMatrix();
    glTranslated(5, 0.96, -5.5);
    glRotatef(30,0,1,0);
    glColor3fv(color);
    glScaled(0.7,1.3,0.3);
    glScalef(1.1,1.1,1);
    glutSolidCube(4);
glPopMatrix();
```

O contorno implementado, que é um cubo na forma wire, serve para deixarmos as arestas da figura visível, dando uma perspectiva de profundidade. Esta técnica foi utilizada em todo o código.

- Sol

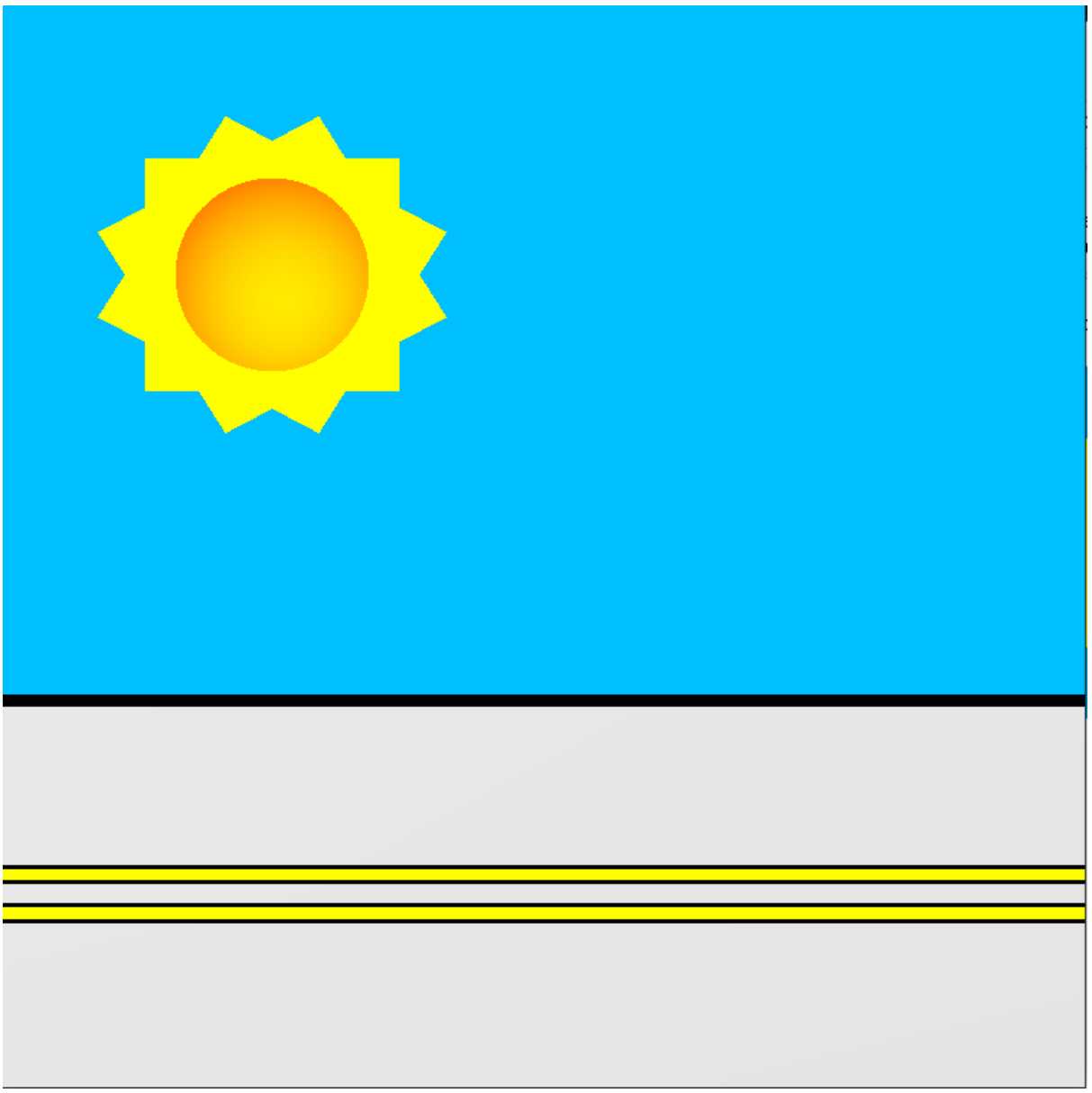
O último elemento que completa o código é o Sol, para preencher o espaço vazio no céu. O sol em si é uma esfera grande laranja, já os raios solares foram feitos por um conjunto de cubos entrelaçados em diferentes posições. Juntamente com o Sol, temos o nosso background, formado por uma via de mão dupla e o céu. O sol é quem comanda a iluminação do projeto.



No trecho de código abaixo veremos a configuração feita para que a iluminação acompanhe o sol. Os parâmetros lightX e lightY são quem determinam a posição da luz.

```
// Sol
glPushMatrix();
    x_ortho = lightX * 7; y_ortho = lightY * 7;
    printf("(%.2f, %.2f)\n", x_ortho, y_ortho);

    glTranslatef(x_ortho, y_ortho, -6.5);

    // Raios
    glPushMatrix();
        glColor3fv(CL_YELLOW);
        glTranslated(0,0,1.0);
        glScaled(11.0,10.0,1.0);
        glutSolidCube(0.30);
        glRotatef(-30,0.0,0.0,1.0);
        glutSolidCube(0.30);
        glRotatef(-120,0.0,0.0,1.0);
        glutSolidCube(0.30);
    glPopMatrix();

    // Centro
    glPushMatrix();
        glColor3fv(CL_ORANGE);
        glScaled(1.0,1.0,1.0);
        glutSolidSphere(1.7, 50, 50);
    glPopMatrix();
glPopMatrix();
```

Abaixo temos a visualização do background do projeto.



b. Deve permitir interação com o teclado e/ou mouse

Foram três as interações implementadas com mouse e teclado:

- [Teclado] letra ‘s’

Troca o sinal do semáforo, que inicialmente é verde. Ao pressionar a letra ‘s’, o mesmo passa para amarelo, ao pressionar novamente, fica vermelho, e por fim, entra em um looping voltando para o verde.

- [Teclado] letra ‘a’

Aumenta a velocidade em que os carros se movimentam a cada clique, até chegar na velocidade máxima.

- [Mouse] Clique com o botão esquerdo

O sol aparece no lugar em que o usuário clicar na tela. Inicialmente o mesmo fica no canto superior esquerdo da tela. Caso o usuário clique nos prédios, o sol aparecerá atrás, e caso o clique seja na região da rua, o sol desaparece.

c. Deve conter algum tipo de animação

A animação implementada envolve algumas ações citadas acima. Ao executar o programa, ele inicia com um carro em movimento e o semáforo verde. Quando o carro sai da tela, entra o ônibus, quando o ônibus sai, entra o carro, e assim eles ficam alternando em um looping infinito; vale ressaltar que cada um anda em uma pista. Quando o semáforo fica amarelo, nada acontece, os carros ainda continuam em movimento, porém quando ele fica vermelho, o veículo que estiver passando na via não avançará o sinal, e ficará ali, parado, até que o semáforo fique verde novamente. Sendo assim, o segundo veículo também não entra na tela. Caso o semáforo seja mudado enquanto o carro passa por ele, o mesmo não irá parar, apenas o próximo que chegar.

Outra animação é referente ao próprio semáforo, que tem suas cores alternando de acordo com os comandos do usuário, pressionando a letra ‘s’ como visto acima.

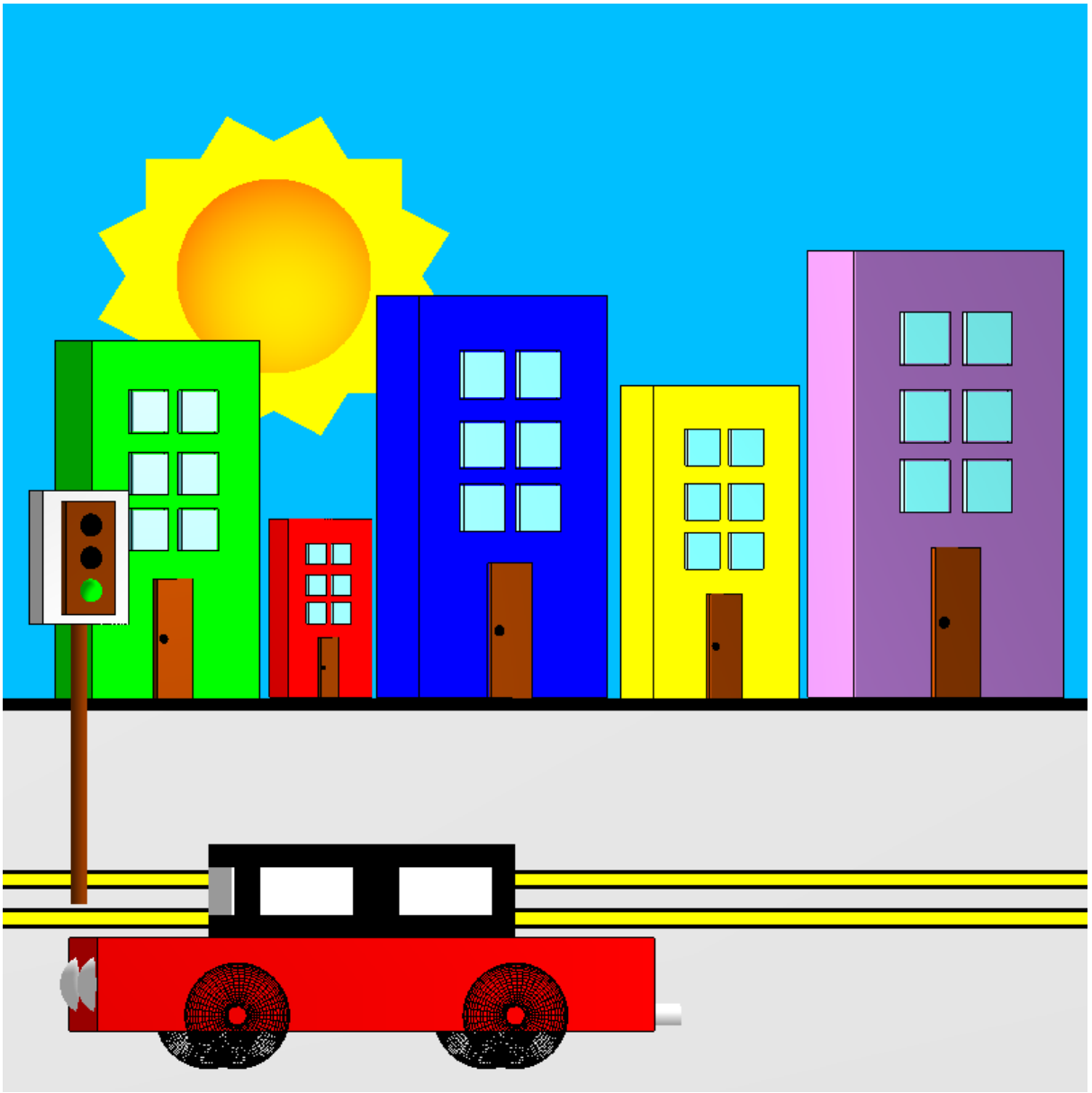
Uma última animação é referente a iluminação que veremos no próximo tópico.

d. Deve usar os recursos da aula sobre modelos de iluminação

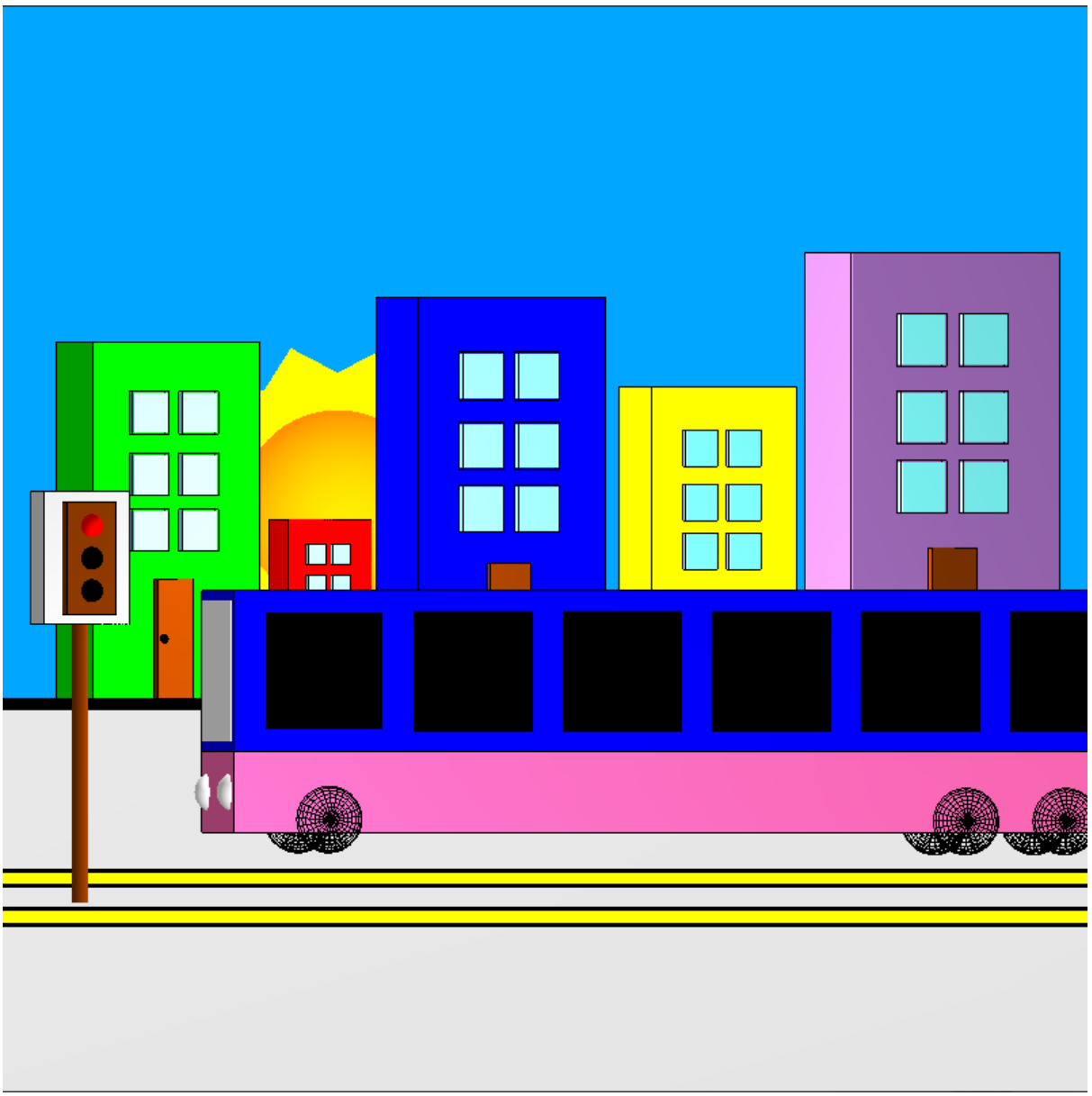
Se tratando da iluminação, procurei fazer algo que o usuário possa interagir. Como mencionado anteriormente, a iluminação acompanha o movimento do sol, que aparece onde o usuário clicar na tela. É perceptível que os prédios mais próximos ao sol ficarão melhor iluminados que aqueles mais distantes. Quando o sol está atrás dos prédios, a cor do céu fica um pouco mais escura, dando a impressão que ficou de tarde, quando o sol fica atrás da rua, não é possível visualizá-lo, e o céu fica ainda mais escuro, dando a impressão de noite. Ou seja, uma iluminação iterativa e coerente.

Segue abaixo as telas finais da aplicação.

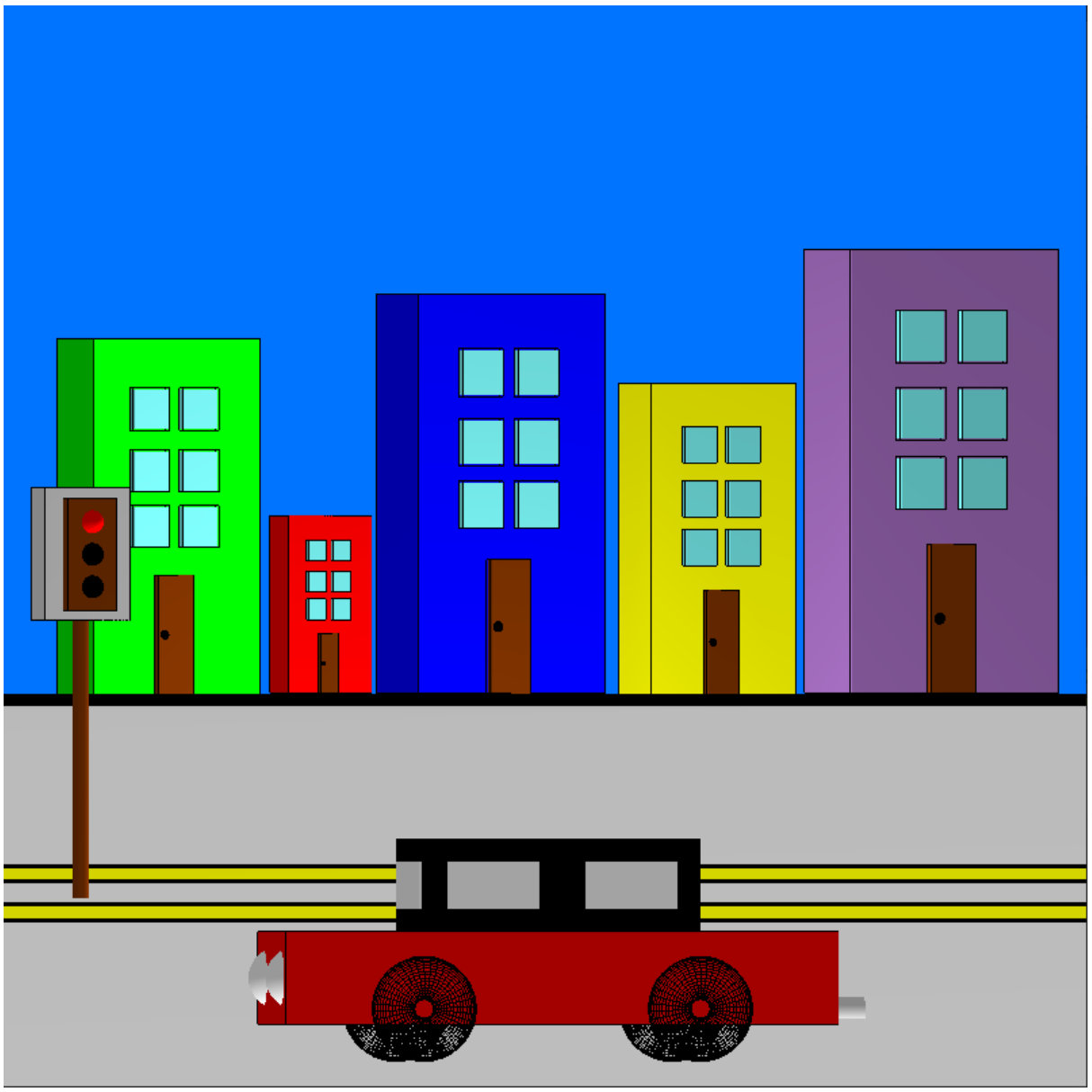
Manhã:

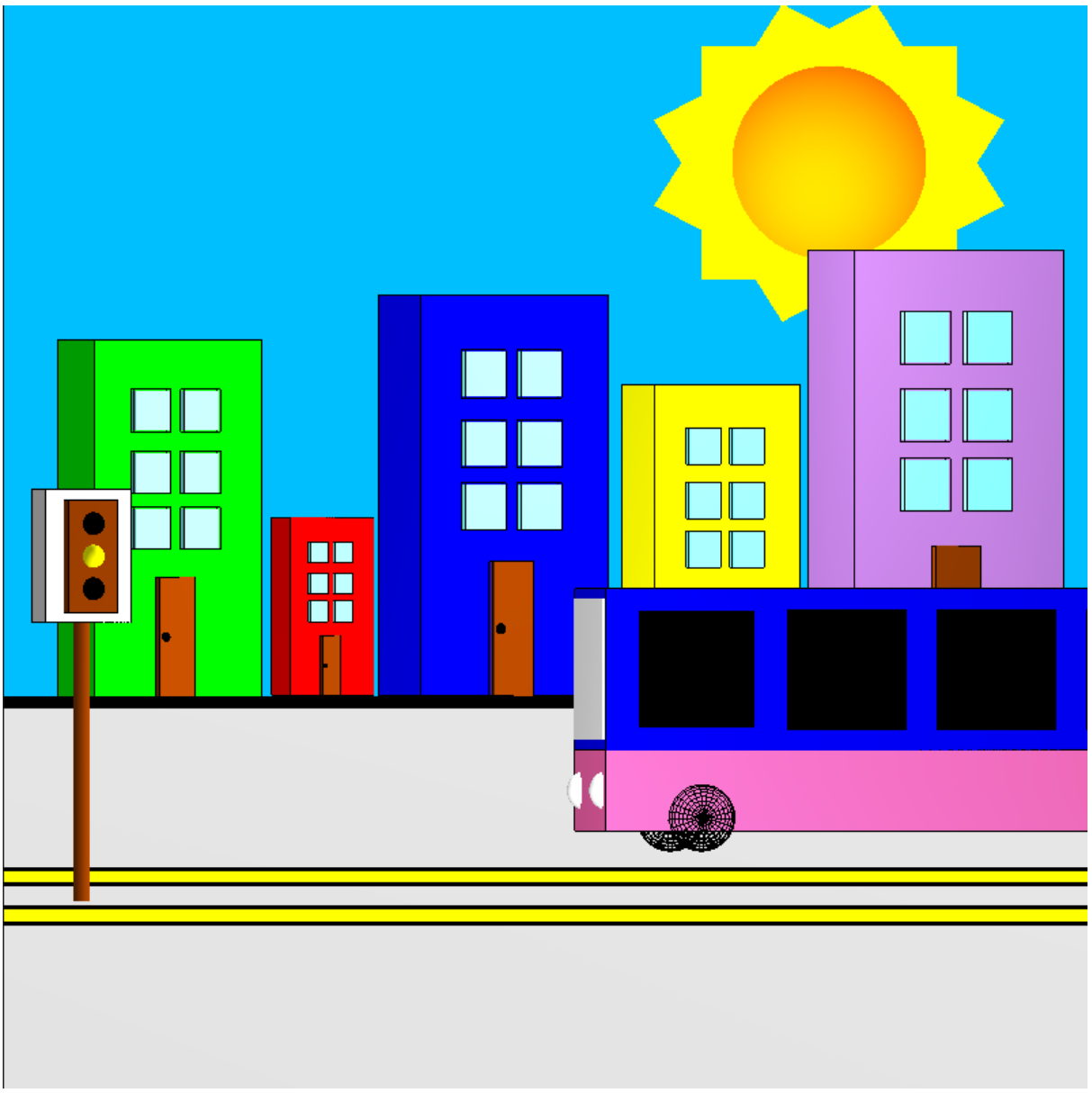


Tarde:



Noite:





Conclusão

Sendo assim, a aplicação desenvolvida cumpre todos os requisitos propostos no trabalho final.