Paint

Computação Gráfica

Tópicos

- Estruturas básicas
- Transformações básicas
- Interação com o usuário
- Save
- Formato do txt
- Load
- Animação

Estruturas básicas

```cpp
struct ponto {
    double x;
    double y;
    vector<double> cor = {0.0, 0.0, 0.0};
};

typedef vector<ponto> pontos;
typedef vector<ponto> reta;
typedef vector<reta> retas;
typedef vector<ponto> poligono;
typedef vector<poligono> poligonos;

struct draws {
    pontos lista_pontos;
    retas lista_retas;
    poligonos lista_poligonos;
};

typedef stack<vector<vector<double>>> operacoes;
```

As estruturas que utilizamos foram:

- ponto; guarda as coordenadas x e y e a cor (que por padrão é preto)
- vector; da biblioteca vector do c++, é uma arraylist, simplifica muito as manipulações dos dados.
- reta; um vector de pontos.
- polígono; um vector de pontos.
- draws; guarda vectors de pontos, retas e polígonos.
- operacoes; um stack de matrizes, guarda as matrizes das operações que serão realizadas, antes de calcular a matriz composta.

Transformações básicas

```cpp
vector<vector<double>> matrizRotacional(double theta) {
    vector<vector<double>> matriz(3, vector<double>(3));

    matriz[0][0] = cos(theta);
    matriz[0][1] = -sin(theta);
    matriz[0][2] = 0;

    matriz[1][0] = sin(theta);
    matriz[1][1] = cos(theta);
    matriz[1][2] = 0;

    matriz[2][0] = 0;
    matriz[2][1] = 0;
    matriz[2][2] = 1;

    return matriz;
}
```

As funções básicas incluem criação das matrizes de transformação.

Transformações básicas

```cpp
vector<vector<double>> calcular_matriz(operacoes &pilha) {
    vector<vector<double>> matriz(3, vector<double>(3));
    vector<vector<double>> auxiliar(3, vector<double>(3));

    matriz[0][0] = 1;
    matriz[0][1] = 0;
    matriz[0][2] = 0;

    matriz[1][0] = 0;
    matriz[1][1] = 1;
    matriz[1][2] = 0;

    matriz[2][0] = 0;
    matriz[2][1] = 0;
    matriz[2][2] = 1;

    while(!pilha.empty()) {
        vector<vector<double>> temporario = pilha.top();
        pilha.pop();

        for(int i = 0; i < 3; i++) {
            for(int j = 0; j < 3; j++) {
                double produto = 0.0;
                for(int k = 0; k < 3; k++)
                    produto += matriz[i][k] * temporario[k][j];

                auxiliar[i][j] = produto;
            }
        }

        for(int i = 0; i < 3; i++) {
            for(int j = 0; j < 3; j++)
                matriz[i][j] = auxiliar[i][j];
        }
    }

    return matriz;
}
```

Funções como calcular a matriz composta.

Transformações básicas

```cpp
bool selecionar_area(poligono &p, double mx, double my) {
    int qtd = 0;

    for(int i = 0; i < p.size(); i++) {
        int j = (i + 1) % p.size();

        if(my == p[i].y) {
            if(my > p[(i - 1 + p.size()) % p.size()].y)
                qtd++;
            if(my > p[j].y)
                qtd++;
        }
        else if(p[i].y > my && p[j].y > my)
            continue;
        else if(p[i].y < my && p[j].y < my)
            continue;
        else if(p[i].x < mx && p[j].x < mx)
            continue;
        else if(p[i].x > mx && p[j].x > mx && ((p[i].y > my && p[j].y < my) || p[i].y < my && p[j].y > my))
            qtd++;
        else {
            int xi = p[i].x + (my - p[i].y) * (p[j].x - p[i].x) / (p[j].y - p[i].y);
            if(xi > mx)
                qtd++;
        }
    }

    return qtd % 2 == 1;
}
```

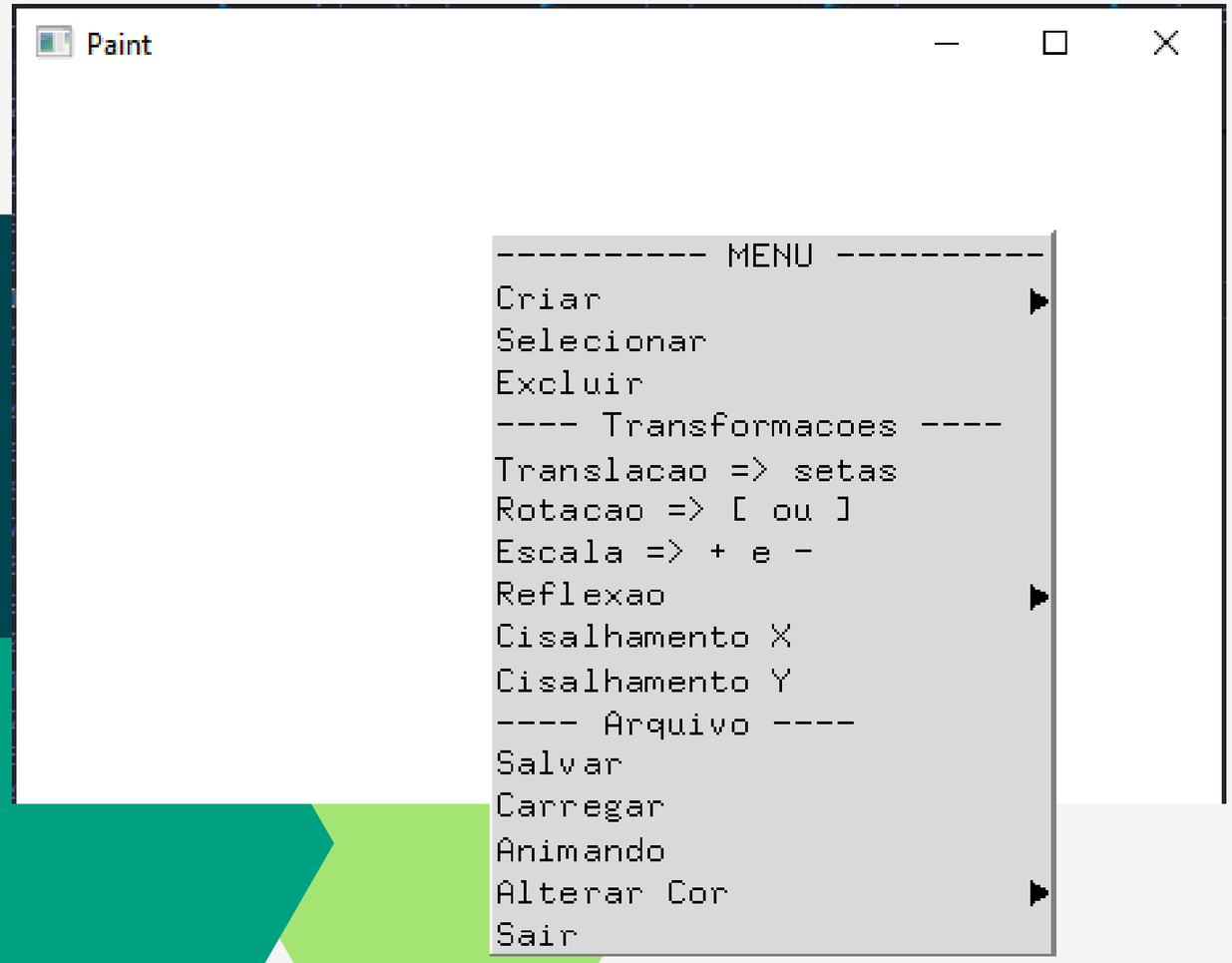
Selecionar os objetos.

Transformações básicas

```cpp
void rotacionar_r(reta &r, double theta, pair<double, double> centroide) {

    operacoes rot;

    rot.push(matrizTransacional(-centroide.first, -centroide.second));
    rot.push(matrizRotacional(theta));
    rot.push(matrizTransacional(centroide.first, centroide.second));

    vector<vector<double>> matriz = calcular_matriz(rot);

    calcular_novo_ponto(matriz, r[0]);
    calcular_novo_ponto(matriz, r[1]);
}
```

Aplicar as transformações.

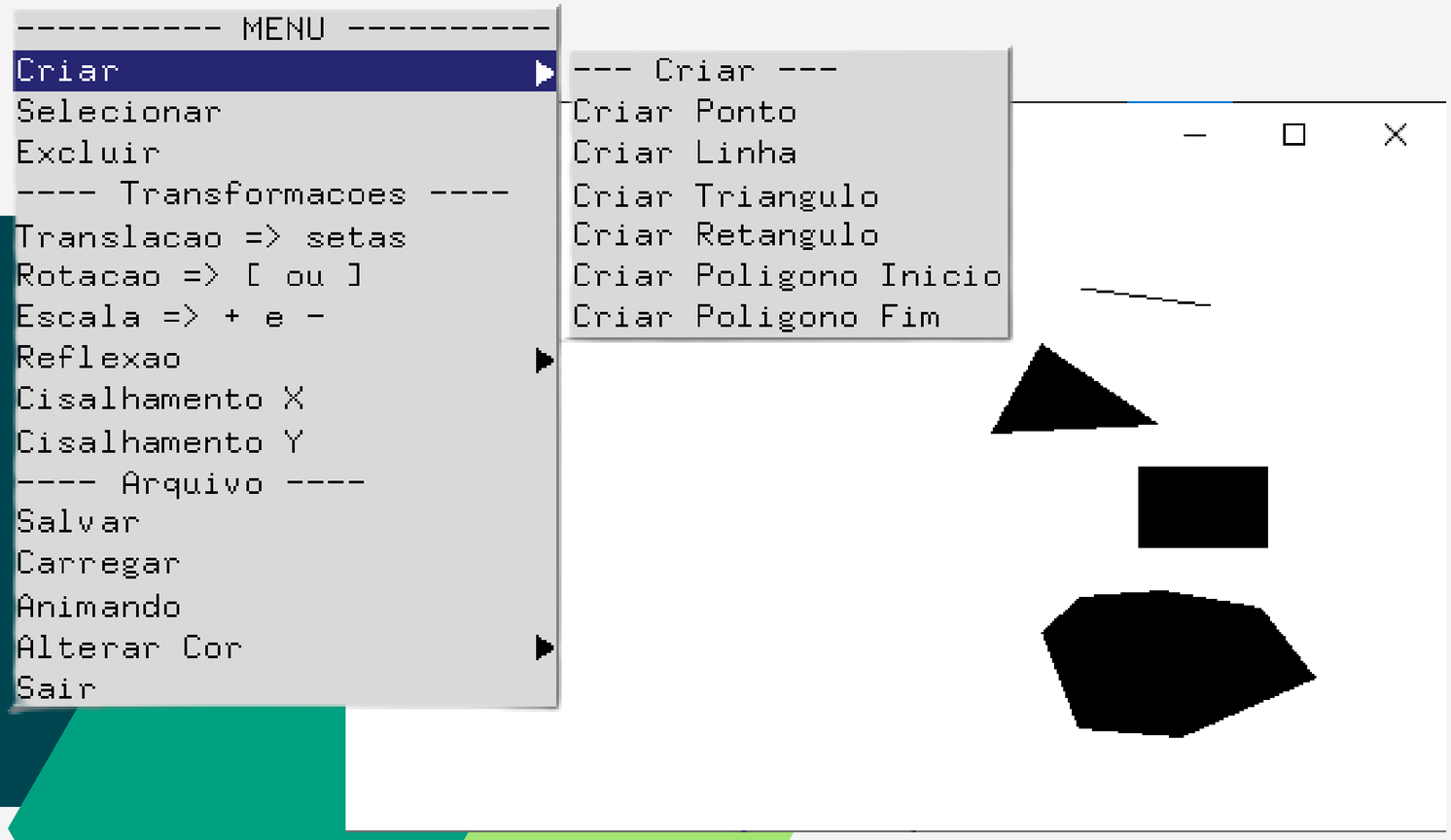
Interação com o usuário



A principal forma de interação é por menu

- 500 x 300
- Divisão em 3 seção
  - Principal/Criação
  - Transformações
  - Arquivos/Adicionais
- Inicialmente, ele faz os desenhos na cor preta.

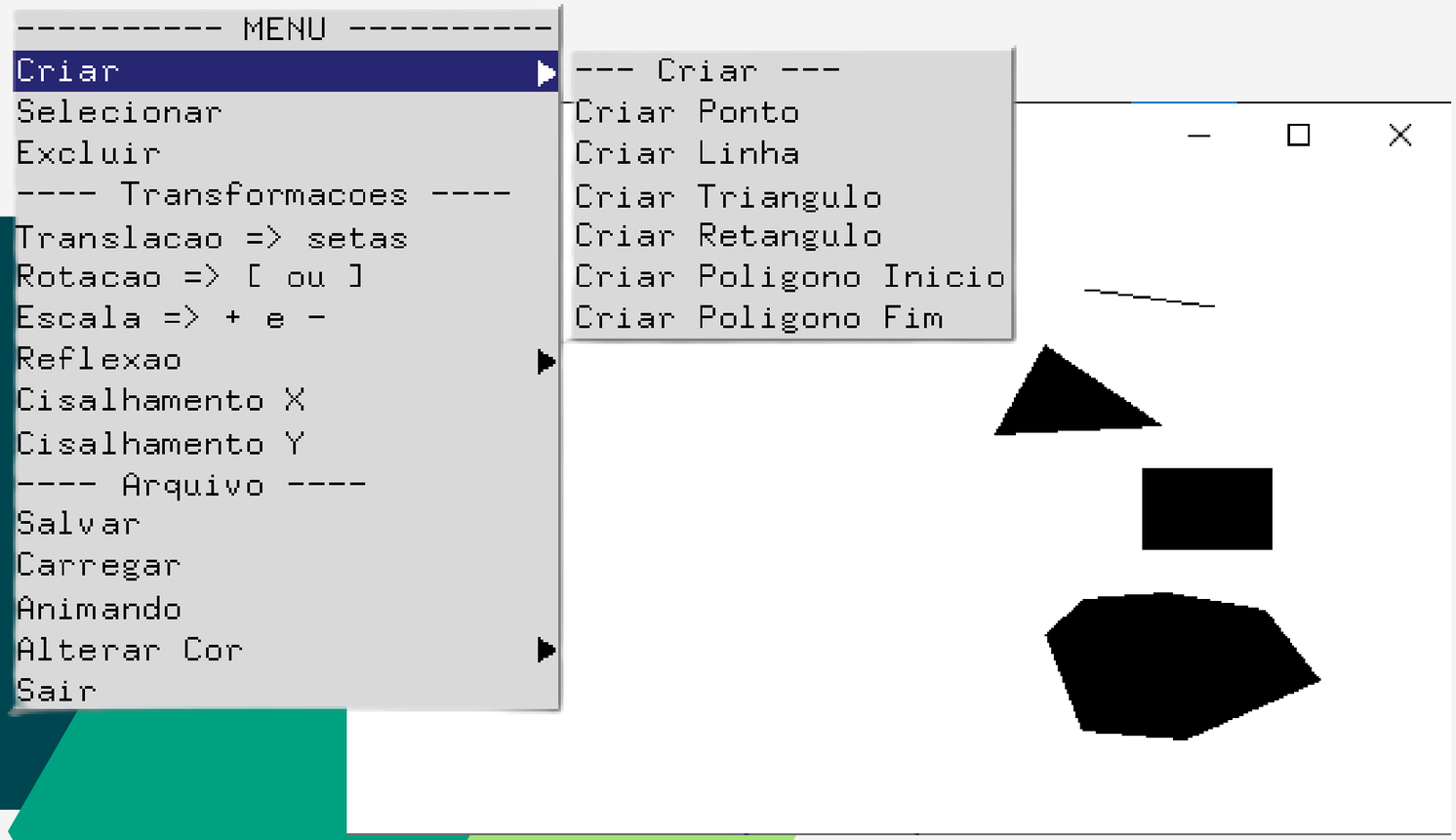
Interação com o usuário



Criação de objetos

- Inicia a criando pontos
- Seleciona a opção
- ponto → 1 click
- linha → 2 click
- Triangulo → 3 click
- Retangulo → 2 click
- Polígono:
  - Inicia a criação de ponto
  - Finaliza a criação de ponto

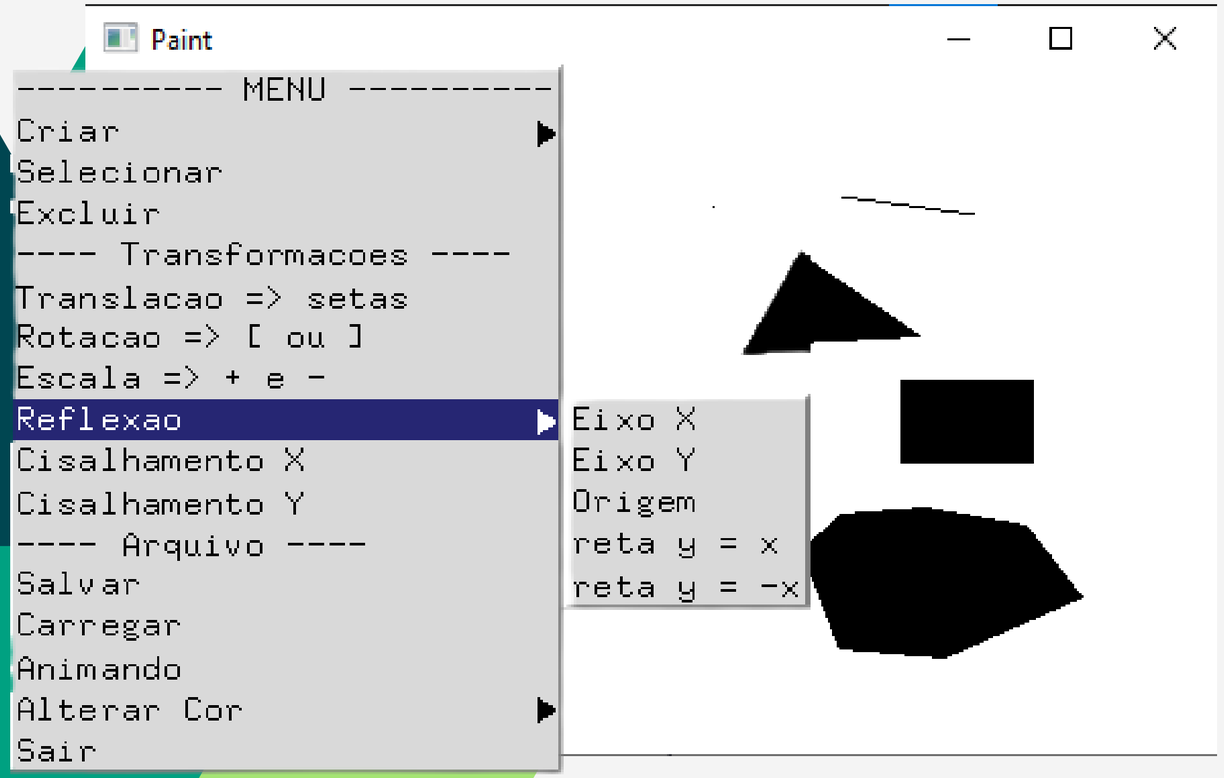
Interação com o usuário



Selecionar: Você deve clicar na opção selecionar e clicar no objeto que você quer escolher.

Excluir: Após o objeto selecionado, ao apertar excluir, o objeto será apagado.

Interação com o usuário



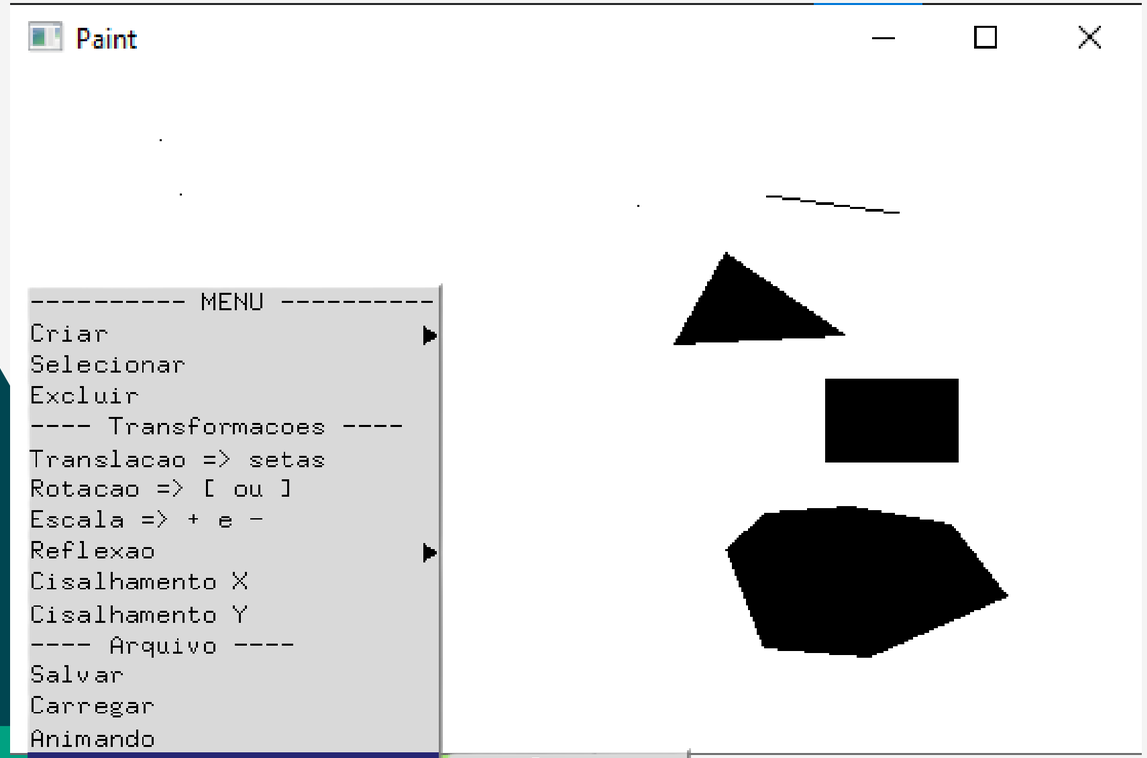
Transformações

Com o objeto selecionado, poderá fazer as 3 transformações básica.

Para realizar a reflexão e o cisalhamento, deve está com o objeto selecionado e aperta a opção da transformação escolhida.

Cuidado, a escala é a tecla +

Interação com o usuário



Adicionais

Salvar precisa apenas clicar na opção e será criado ou sobreescrito um arquivo save.txt, **Carregar** precisa de uma arquivo save.txt, que seá lido.

Animando: Chama a animação

Alterar Cor: muda a cor que você vai poder escrever e não a cor do objeto selecionado

Save

```
11  void save_objects(draws &structure_list) {
12      std::cout << "Salvando..." << std::endl;
13      std::ofstream arquivo("save.txt");  // Abrir o arquivo de forma adequada
14
15      if (!arquivo.is_open()) {
16          std::cerr << "Erro ao abrir o arquivo para salvar!" << std::endl;
17          return;
18      }
19
20      // Salvando os pontos
21      for (const ponto &p : structure_list.lista_pontos) {
22          arquivo << p.x << " " << p.y << " ";  // Escreve as coordenadas
23          for (double c : p.cor) {
24              arquivo << c << " ";  // Escreve os componentes de cor
25          }
26          arquivo << std::endl;  // Quebra de linha após cada ponto
27      }
28      arquivo << "p" << std::endl;
```

- Abrir o arquivo
- Colocar as informações dos pontos e cores no arquivo separados por um espaço
- Após finalizar a estrutura colocar o caractere correspondente a pereira letra na linha inferior
- Repetir para as demais estruturas

Formato do txt

-50 90 1 0 0
p
100 90 1 1 0 20 80 1 0 1
r
333 146 0 0 0 333 146 0 0 0 303 89 0 0 0 f
414 97 0 0 0 286 133 0 0 0 336 80 0 0 0 f
311 203 0 0 0 415 177 0 0 0 376 240 0 0 0 f
p

Load

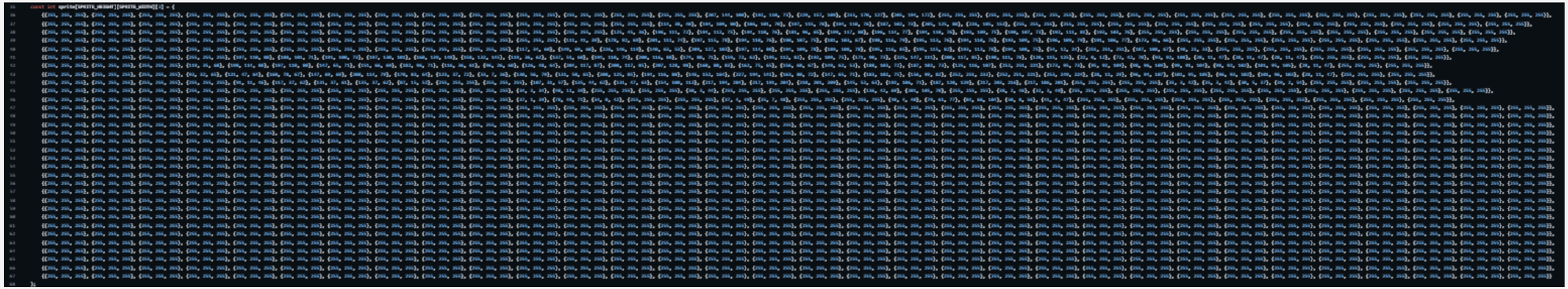
```
void load_objects(draws &structure_list) {
    std::cout << "Carregando..." << std::endl;
    std::ifstream arquivo("save.txt");
    std::string linha;
    int modo = 1;
    std::string str = "";

    while (std::getline(arquivo, linha)) {
        if (linha.length() == 1) {
            modo++;
        }

        switch (modo) {
            case 1: {
                int x, y;
                std::vector<double> c;
                int nunV = 1;

                for (int i = 0; i < linha.length(); i++) {
                    int numero = linha[i] - '0';
                    if (numero >= 0 && numero <= 9) {
                        str = str + linha[i];
                    }

                    if (linha[i] == ' ' || i == linha.length() - 1) {
                        if (!str.empty()) {  // Verifica se a string não está vazia antes de usar
                            switch (nunV) {
                                case 1:
                                    x = std::stoi(str);
                                    break;
                                case 2:
                                    y = std::stoi(str);
                                    break;
                                case 3:
                                case 4:
                                    c.push_back(std::stoi(str));
                                    break;
                                case 5:
                                    c.push_back(std::stoi(str));
                                    create_point(x, y, c, structure_list);
                                    c.clear();
                                    break;
                            }
                            str = "";  // Limpa a string após cada conversão
```

- Abrir o arquivo
- **Percorrer a linha pegando as informações e alocando em variáveis**
- Desenhar a estrutura com as informações coletadas
- Repetir para as demais estruturas

Animação

Dificuldade inicial de exportar imagem

Por conta da dificuldade de alocar a imagem optei por exportar um sprite composto por várias variáveis contendo o RGB do pixel específico



Animação

Controlando o movimento da animação

```
70  void update(int value) {
71      if(fim == false){
72              if(muv){
73              spriteX += 5.0f;
74              spriteY += 5.0f;
75              muv = false;
76          }else{
77              spriteX += 5.0f;
78              spriteY -= 5.0f;
79              muv = true;
80          }
81
82          if (spriteX > WINDOW_WIDTH) {
83              fim = true;
84              spriteX = -50.0f;
85          }
86
87      }
88
89      glutPostRedisplay();
90      glutTimerFunc(100, update, 0);
91  }
```

- Espera a variável fim ser desativada para movimentar o personagem para cima e para direita
- Movimenta o personagem para baixo e para direita
- Continua o lup ate o personagem sair da tela
- Ativa a variável fim e devolve o objeto a posição original fora da tela

Tem alguma pergunta?

Alan Gabriel Silva Oliveira

alan.gabriel@aluno.ufca.edu.br

Pedro da Silva Viana

pedro.viana@aluno.ufca.edu.br

Felipe Rodrigues Ferraz de Alencar

ferraz.felipe@aluno.ufca.edu.br