



# GEOMETRIA COMPUTACIONAL

## AVANÇO DE FRONTEIRA

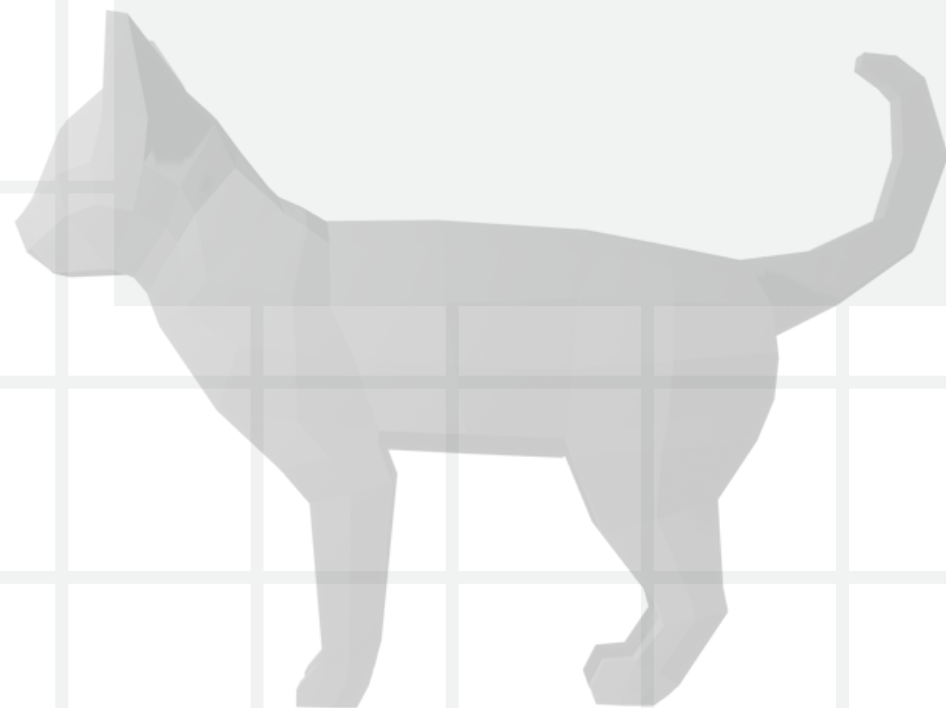
**Luis Fernando Bastos Rego - 470043**

**Samuel Vieira de Paula Farias - 498844**

# INTRODUÇÃO


O algoritmo de Avanço de Fronteira é um dos algoritmos de triangulação, que pode receber diversos critérios para a produção da triangulação.

Nesta apresentação iremos mostrar como foi feito a implementação desse algoritmo.

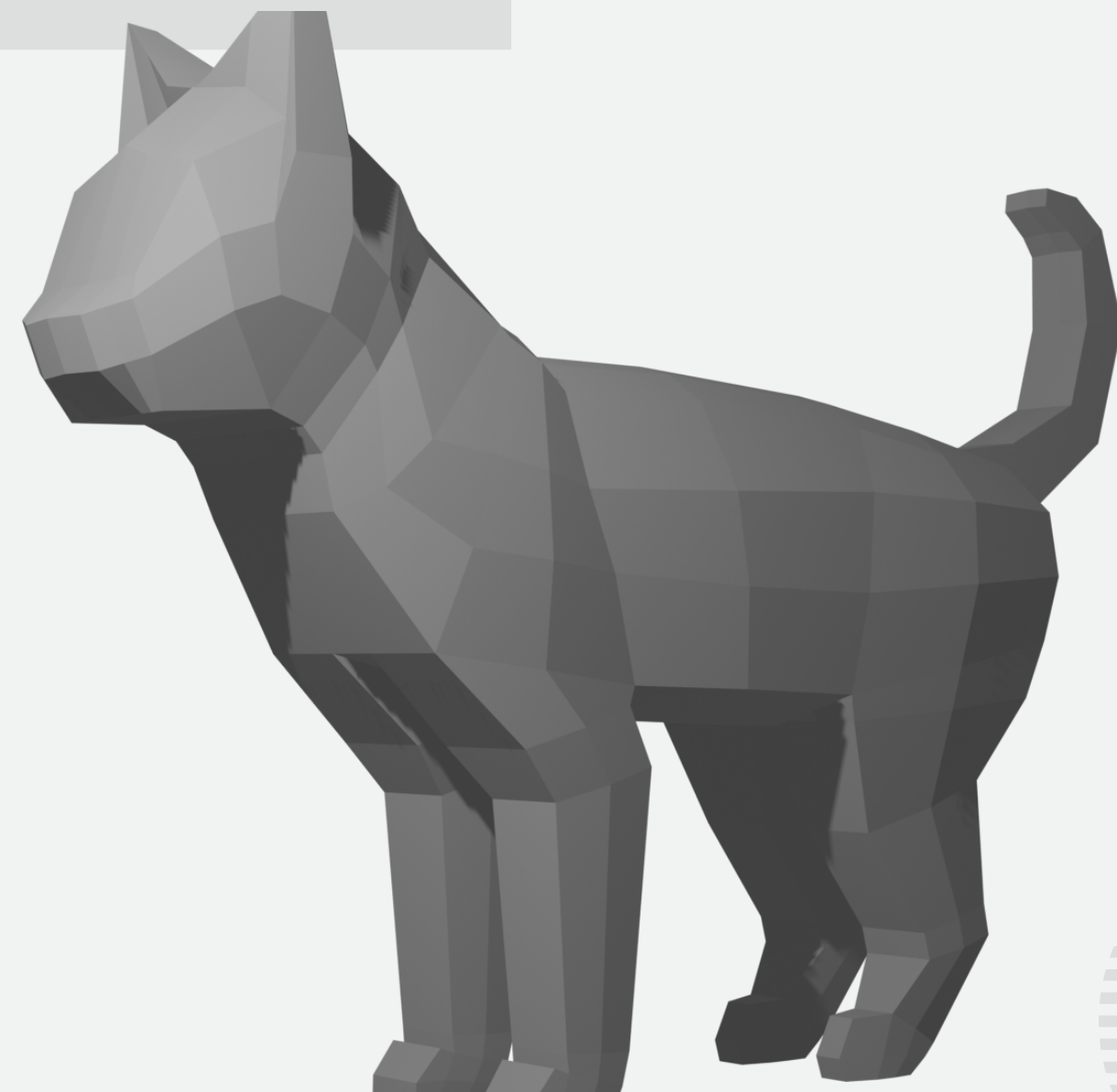
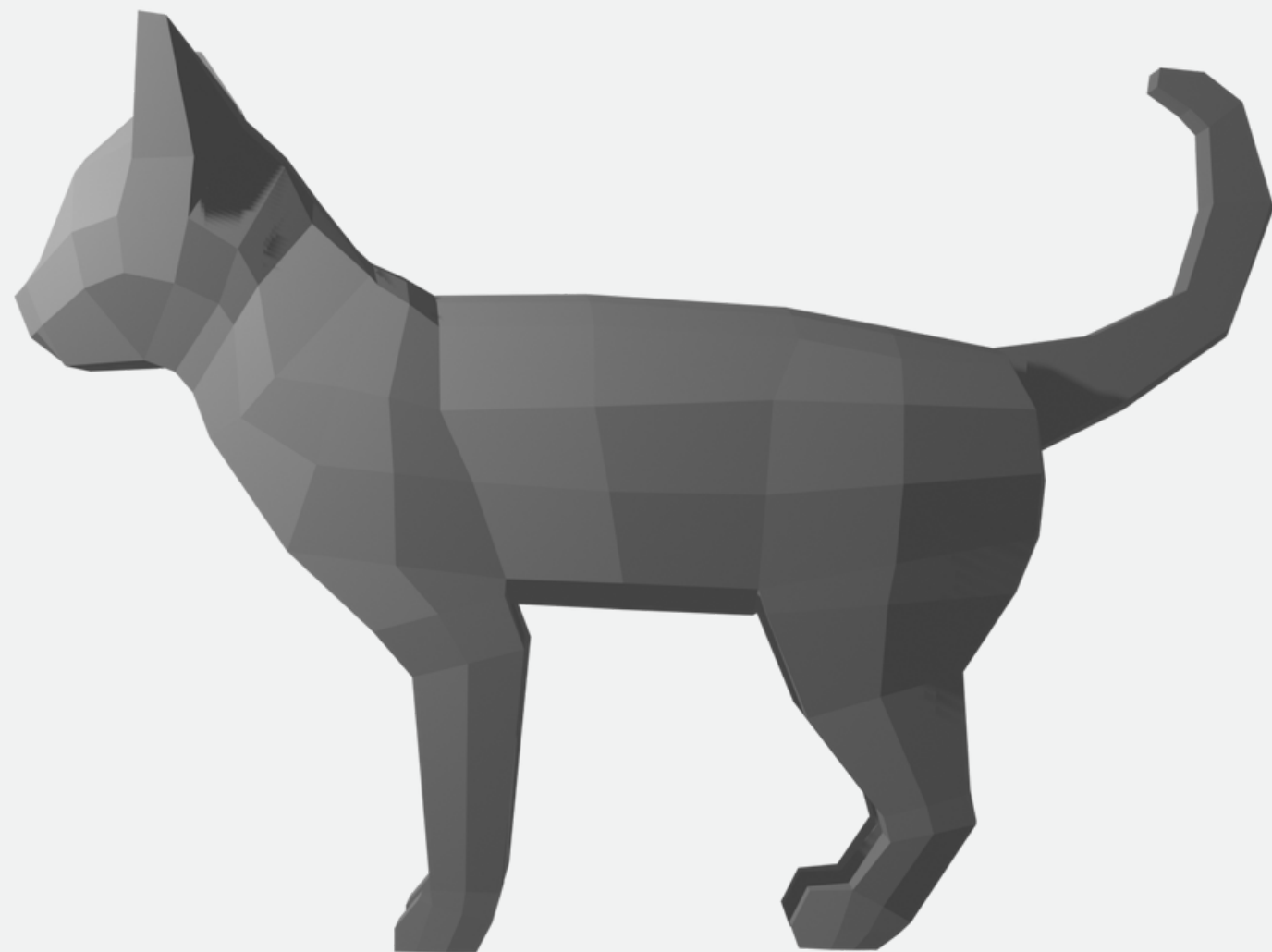




# **METODOLOGIA**

- Foram usados os pseudocódigos e explicações disponibilizados nos slides da disciplina;
  - Códigos escritos na linguagem C++;
  - Blender utilizado para renderização e modelagem;
  - Tema é uma projeção de um modelo 3D, disponibilizado na bibliografia deste slide;
- 

# TEMA



# PLANEJAMENTO

```
graph TD; A[PLANEJAMENTO] --- B[PARTE 1]; A --- C[PARTE 2]; A --- D[PARTE 3];
```

## PARTE 1

Executar o algoritmo de Graham com o tema

## PARTE 2

Produção da triangulação com o Avanço de Fronteira

## PARTE 3

União das triangulações produzidas

# PRÉ-PROCESSAMENTO

## TRANSIÇÃO DO 3D PARA O 2D

Foi utilizado o Blender para realizar o "flattening" do modelo em torno do plano sagital, ou seja, a forma do gato em perfil

## REMOÇÃO E MODIFICAÇÃO DE VÉRTICES INDESEJÁVEIS

Para facilitar a divisão entre partes convexas, foram removidos ou modificados alguns vértices indesejáveis

# PSEUDOCÓDIGO

FrontierAdvance(pontos, arestas, critério):

queue ← adiciona(arestas), arestas estão orientadas todas de acordo com o fecho convexo.

enquanto a queue não está vazia:

atual ← queue.pop()

se atual ainda não foi checada 2 vezes:

melhor ← escolhePonto(critério), onde critério é uma função/objeto relativo ao critério que se foi escolhido, no nosso caso, Delaunay.

aresta1 ← fazAresta(atual.primeiro, melhor)

aresta2 ← fazAresta(melhor, atual.segundo), ambas inicializadas como checada 1 vez.

queue.push(aresta1)

queue.push(aresta2)

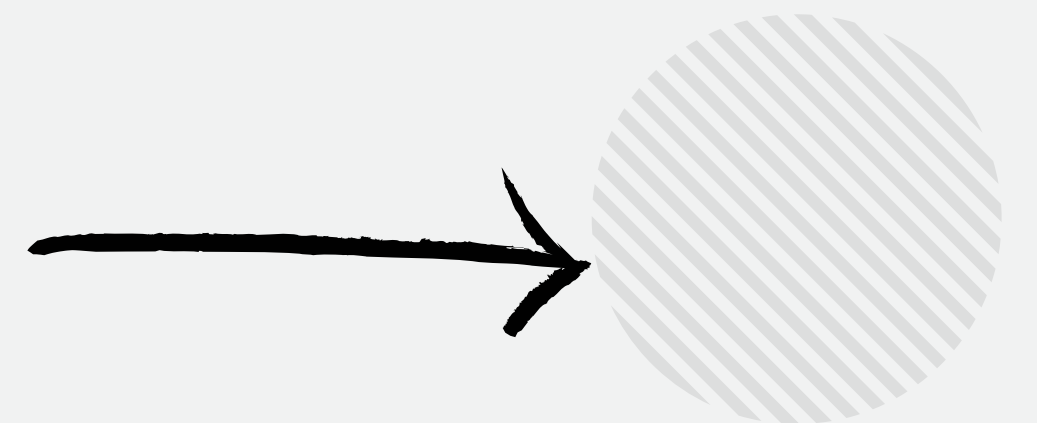
coloca na triangulação as arestas atual, aresta1 e aresta2

# COLAR

O algoritmo correspondente ao "colar" consiste em criar um vetor de vértices únicos  $T$ , no qual é preenchido com a leitura de vértices de cada objeto após a execução do algoritmo.

Cada triangulação retorna um vetor de arestas, que são pares de inteiros que representam índices relativos a cada divisão convexa.

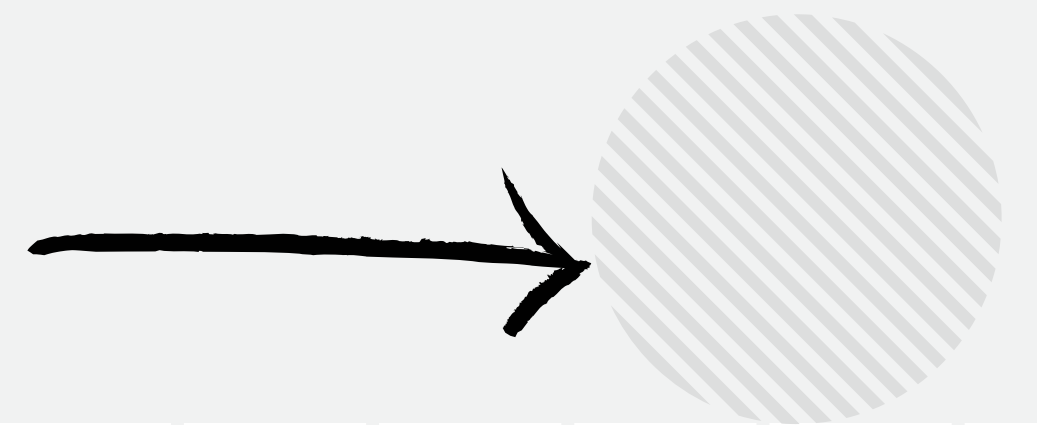
Cada adição em  $T$  retorna um índice absoluto, que é substituído em cada triangulação.





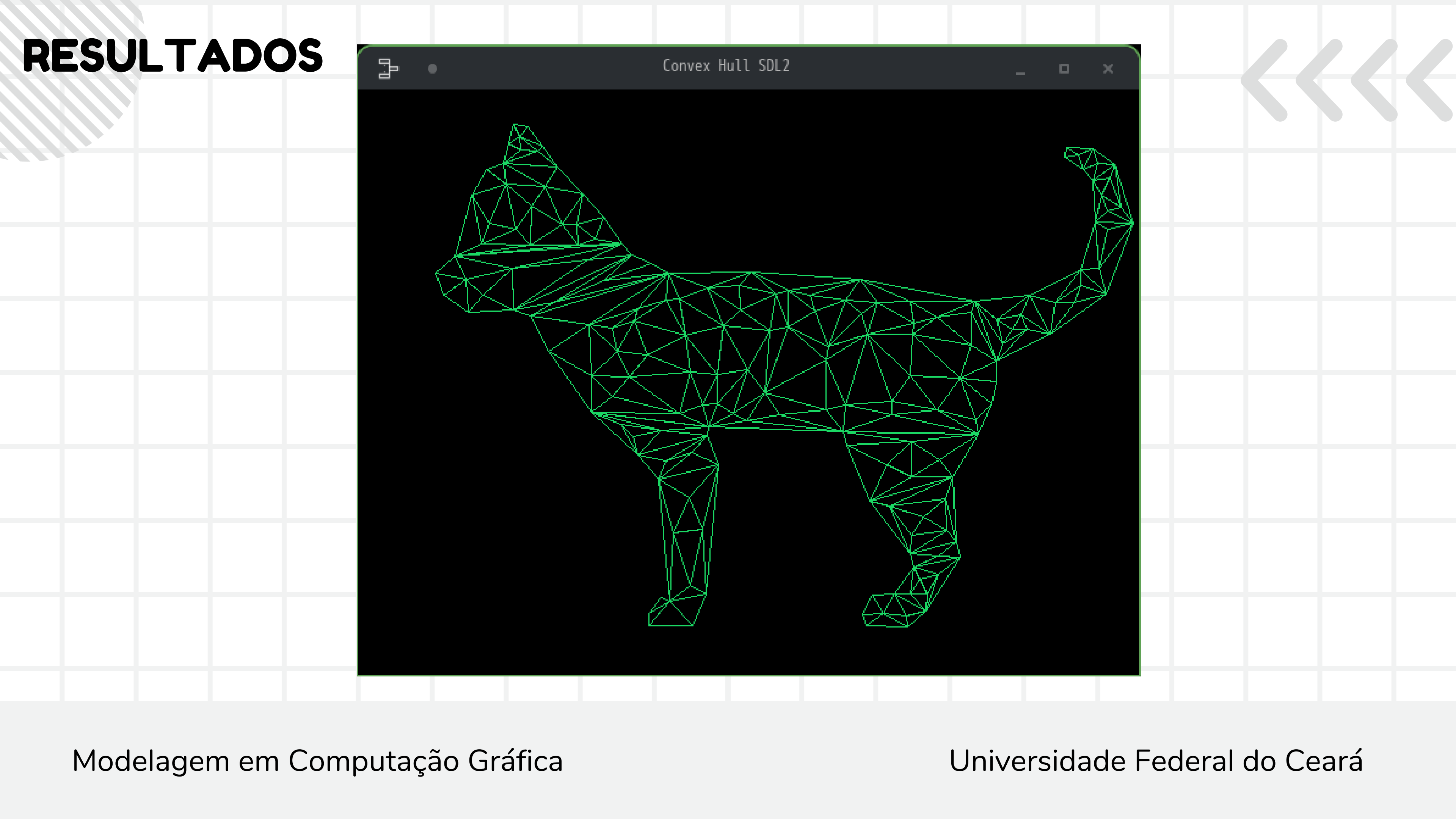
# COLAR

Com cada triangulação agora possuindo índices absolutos, basta construir uma matriz, que indica que se a posição  $u,v$  possuir o valor true, a aresta  $(u,v)$  já foi adicionada, assim não teremos arestas duplas.

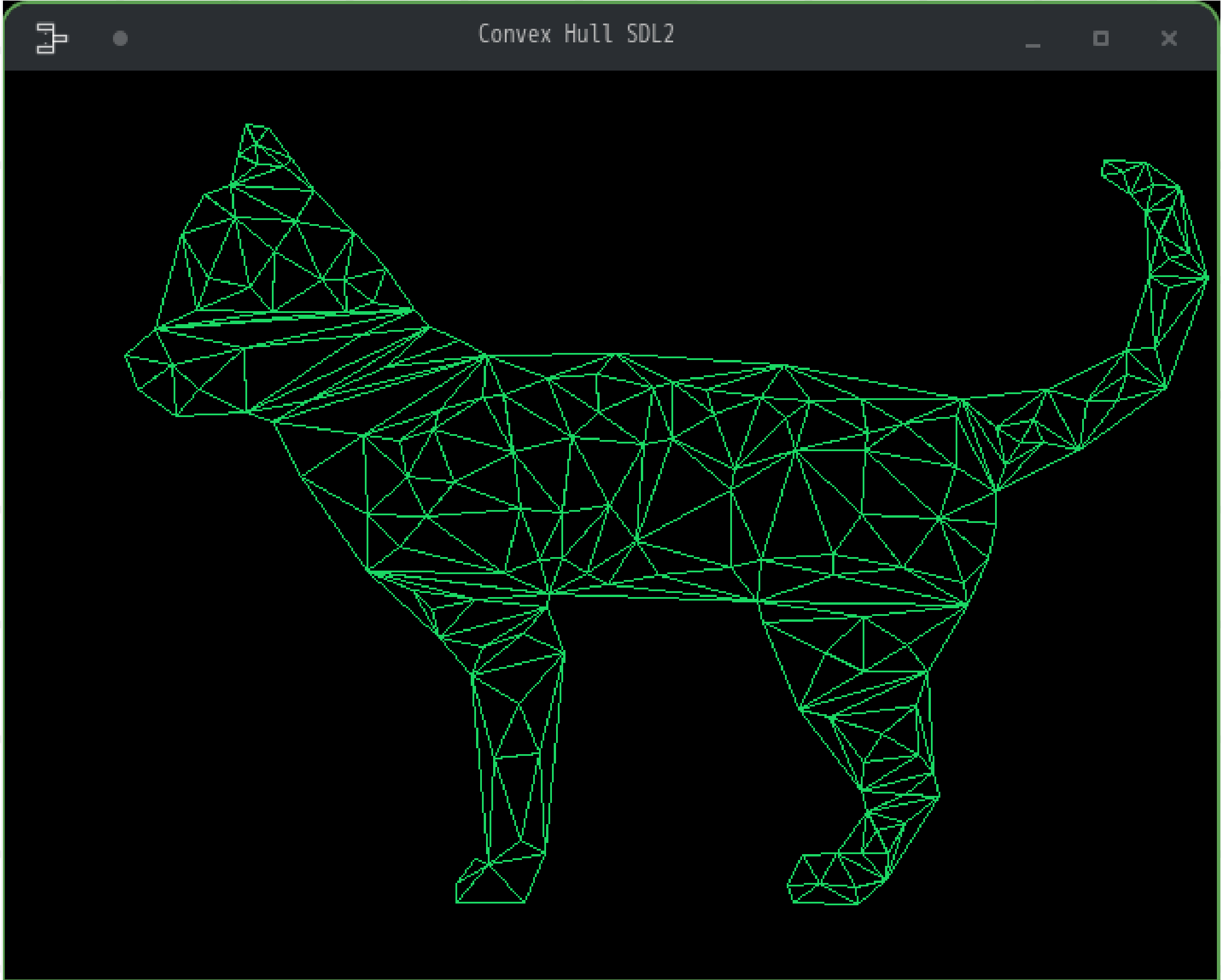


# COLAR

- Se  $\text{matrix}[i][j] == F, \text{matriz}[i][j]$  e  $\text{matriz}[j][i] = V$   
(A aresta ainda não existe, logo, adiciona)
- Se  $\text{matrix}[i][j] == V, \text{matriz}[i][j]$  e  $\text{matriz}[j][i] = F$   
(A aresta já existe e foi encontrada uma igual, logo, é ignorada)




# RESULTADOS





# BIBLIOGRAFIA

- 🔍 Slides da disciplina disponibilizados no email.
  - 🔍 CARVALHO, Paulo C. P., FIGUEIREDO, L.H. Introdução à Geometria Computacional, Rio de Janeiro: IMPA, 1991
  - 🔍 Modelo utilizado no tema: [link](#)
- 

# OBRIGADO



Apresentação feita por Luis Fernando e Samuel Vieira

Repositório em <https://github.com/samue1v/computational-Geometry/>