Lá, bem, bem longe

É uma época de guerra civil. Naves espaciais rebeldes, atacando de uma base secreta, obtiveram sua primeira vitória contra o maligno Império Galáctico.

Durante a batalha, espiões rebeldes conseguiram roubar os planos secretos da arma definitiva do Império, a ESTRELA DA MORTE, uma estação espacial blindada com poder suficiente para destruir um planeta inteiro.

Perseguida pelos sinistros agentes do Império, a Princesa Leia corre para casa a bordo de sua nave estelar, guardiã dos planos roubados que podem salvar seu povo e restaurar a liberdade na galáxia... (Lucas, George (Diretor). Star Wars: Episódio IV - Uma Nova Esperança. Lucasfilm, Twentieth Century Fox, 1977.)

As batalhas são travadas por meio de ogivas que visam destruir diversos tipos de equipamentos. Também existem drones espiões. As ogivas podem ser lançadas a uma certa distância em lotes ou individualmente. Também é possível enviar um comando especifico para que uma ogiva exploda no local onde está. Sua tarefa, como fiéis súditos da princesa Léia, é aferir e contabilizar os resultado dos ataques dos rebeldes.

Ogivas, lançadores, equipamentos, etc são representados por formas geométricas simples (círculos, retângulos, linhas e textos) dispostos num campo bidimensional, como descrito na próxima seção.

A Entrada

***ATENÇÃO**: Não esqueça de ler a descrição geral dos projetos (Sala de Aula).*

A entrada do algoritmo será basicamente um conjunto de formas geométricas básicas (retângulos, círculos, etc) dispostos numa região do plano cartesiano .

Considere a Ilustração 1. Cada forma geométrica é definida por uma coordenada âncora (marcada, na figura, por um pequeno ponto vermelho) e por suas dimensões. A coordenada âncora do círculo é o seu centro e sua dimensão é definida por seu raio (r, na figura). A coordenada âncora do retângulo é seu canto inferior esquerdo[1] e suas dimensões são sua largura (w) e sua altura (h). A coordenada âncora de um texto, normalmente, é o início do texto, porém, pode ser definida como o meio ou o fim do texto. Por fim, uma linha é determinada por duas âncoras em suas extremidades. As coordenadas que posicionam as formas geométricas são valores reais.

Cada forma geométrica é identificada por um número inteiro.

1 Note que o plano cartesiano está desenhado "de ponta-cabeça" em relação à representação usual.

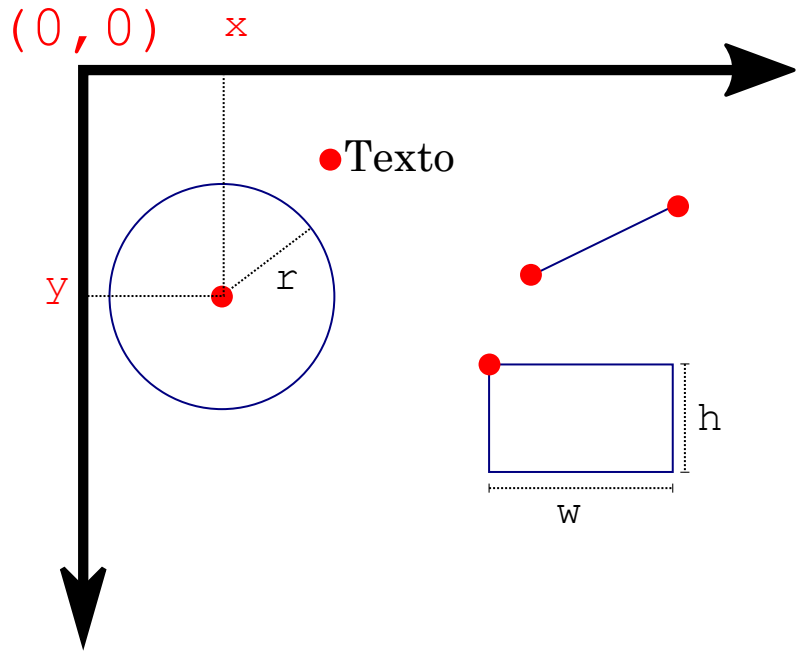

Ilustração 1: Formas no plano

As tabelas abaixo mostram os formatos dos arquivos de entrada (.geo e .qry). Cada comando tem um certo número de parâmetros. Os parâmetros mais comuns são:

- `i,j,k`: número inteiro, maior ou igual a 1. Identificador de uma forma geométrica.
- `r`: número real. Raio do círculo.
- `x,y`: números reais. Coordenada (x,y).
- `cor`: string. Cor válida dentro do padrão SVG.[2]

comando	parâmetros	descrição
c	`i x y r corb corp`	*Cria um círculo com identificador i: (x,y) é o centro do círculo; r, seu raio,* corb *é a cor da borda e* corp *é a cor do preenchimento*
r	`i x y w h corb corp`	*Cria um retângulo com identificador i: (x,y) é a âncora do retângulo, w é a largura do retângulo e h, a altura.* corb *é a cor da borda e* corp *é a cor do preenchimento*
l	`i x1 y1 x2 y2 cor`	*Cria uma linha com identificador i, com extremidades nos pontos (x1,y1) e (x2,y2), com a cor especificada. A linha está orientada da origem (x1,y1) para outra extremidade, destino (x2,y2).*

2 http://www.december.com/html/spec/colorsvg.html.
https://www.w3.org/Graphics/SVG/IG/resources/svgprimer.html

t	`i x y corb corp a "txto"`	*Cria o texto* txto *com identificador i, nas coordenadas (***x,y***) e com a cores indicadas.* corb *é a cor da borda e* corp *é a cor do preenchimento. O parâmetro* **a** *determina a posição da âncora do texto:* **i**, *no início;* **m**, *no meio,* **f**, *no fim.* *O texto txto é o último parâmetro do comando. Pode incluir espaços em branco. O texto é delimitado por aspas, porém, as aspas não fazem parte dele.*
ts	`fFamily fWeight fSize`	*Muda o estilo dos textos (comando t) subsequentes.* *font family: sans (sans-serif), serif, cursive; font weight (n: normal, b: bold, b+: bolder, l: | lighter)*
comandos .geo		

O programa a ser executado está no arquivo .qry que contém comandos do processador.

comando	parâmetros	descrição
selr	n x y w h	*Seleciona as formas inteiramente contidas dentro do retângulo* (x,y,w,h) *e as identifica conjuntamente com o número* n (0..99). *TXT: reportar o identificador e o tipo da forma selecionada* *SVG: colocar uma pequena circunferência vermelha na âncora da forma selecionada. Desenhar o retângulo (x,y,w,h) em vermelho.*
seli	n x y	*Similar a* selr*, porém, seleciona uma única forma, cuja âncora é a coordenada (x,y).*
disp	id n	*Dispara as ogivas da seleção n na direção dada pela linha de identificador id. A distância depende da área da forma disparada. Destrói as formas para as quais a âncora da forma disparada é interna à forma atingida (para mais detalhes, veja seção específica). As ogivas são destruídas.* *TXT: reportar identificador, tipo da forma disparada, a posição inicial, a posição final, a distância de lançamento e o identificador, tipo e posição das respectivas formas atingidas.* *SVG: colocar um pequeno # no local da explosão da ogiva e um pequeno x nas âncoras das formas destruídas.*
transp	id x y	*Move a forma de identificador id para a posição (x,y).* *TXT: reportar tipo e posição original da forma.*
cln	n dx dy	*Clona as formas da seleção n. Os clones são transladados por dx, dy. O clone terá as cores de borda e preenchimento trocadas.*
spy	id	*TXT: Reporta dados sobre as formas inteiramente contidas dentro do retângulo de identificador id ou exatamente*[3] *na coordenada do texto de identificador id.*
cmflg	id cb cp w	*Semelhante ao comando spy, porém, muda a cor da borda para cb, preenchimento cp e largura da borda para w px.*
blow	id	*A ogiva cujo identificar é id explode. A explosão produz os mesmos efeitos da explosão do comando* disp. *TXT: reporta os atributos da figura removida e os atributos das figuras atingidas*
Comandos .qry		

3 Lembre-se do epsilon.

Seleção e Disparo

A Ilustração 2 exemplifica as operações de seleção espacial e disparo nos comandos abaixo:

```
selr 9 4.0 5.0 50.0 87.0
disp 7 9
```

Note que a ilustração apresenta algumas anotações (em vermelho) que não fazem parte do resultado, mas apenas explicam a semântica dos referidos comandos.

A ilustração apresenta 4 círculos, 3 retângulos, 3 linhas e 3 textos. Note a origem de cada linha está marcada (anotada) com um pequeno ponto vermelho. Também, uma das linhas está anotada com seu identificador (id:7) em vermelho.

O primeiro comando (`selr`) seleciona as figuras contidas no retângulo com âncora em (4.0, 5.0), largura 50.0 e altura 87.0 (retângulo de borda vermelha tracejada) e identifica esta seleção com o número 9. Note que os eixos x e y estão anotados na ilustração e também alguns pontos nestes eixo.

O segundo comando (`disp`) dispara uma ogiga, a partir de cada figura selecionada a uma certa distância na direção determinada pela linha cujo identificador é 7. A distância do lançamento corresponde à area da figura selecionada. Por exemplo, o círculo verde possui uma área de aproximadamente 23,1 u^2. Portanto, a ogiva será lançada a uma distância de 23,1 u. Já o circulo azul, com uma área de aproximadamente 63,5u^2, será lançado a uma distância de 63,5 u. Note que a direção/sentido dos lançamentos estão anotados em algumas das figuras selecionadas com uma seta vermelha. Para o cálculo da "área" de linhas e textos, convenciona-se:

- linha: distância do lançamento é 10*comprimento da linha
- texto: 12 * número de caracteres

Obs.: Um ponto é interno a outro ponto se ambos são iguais.[4]

4 Lembrar do epsilon

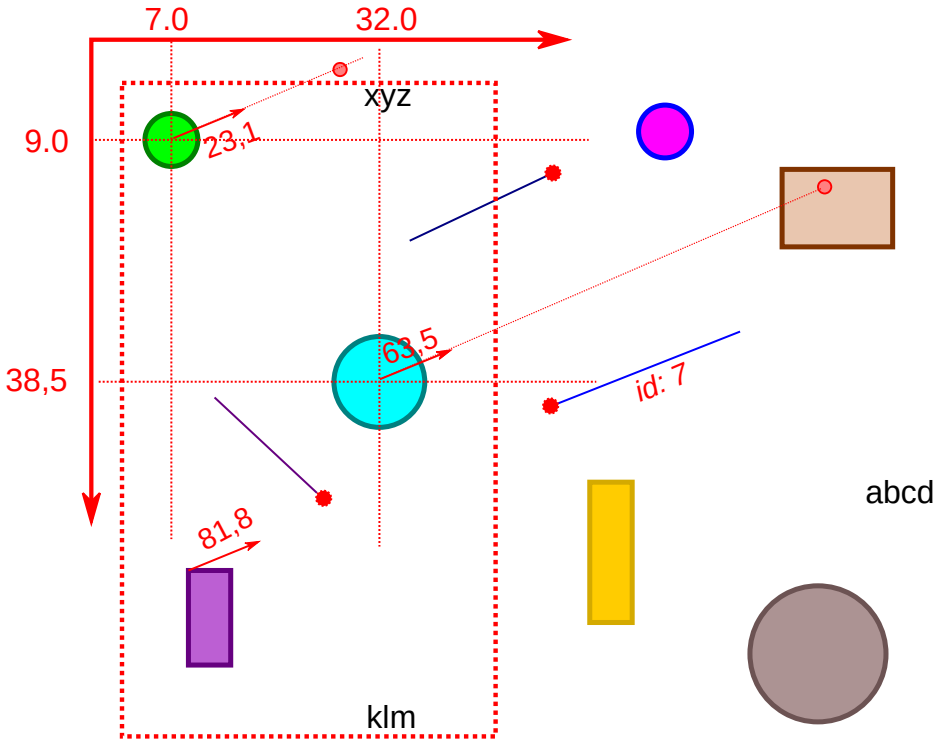

Ilustração 2: Seleção seguida de disparo

A Implementação

Implementar os TADs postados em nossa Sala de Aula.

É **terminantemente proibido** declarar structs nos arquivos de cabeçalho (.h).

O programa deve estar bem modularizado (arquivos .h e .c). Cada estrutura de dados deve estar em um módulo separado. O arquivo .h deve estar muito bem documentado (lembre-se que é um "contrato").

As formas **devem** ser armazenadas em árvore SmuTreap. A árvore **deve** ser implementada de acordo com o arquivo de cabeçalho provido no Classroom. Note que a chave desta árvore é a cordenada da forma. O desenho do arquivo svg com as formas **deve** ser implementado usando alguma das operações de percurso definas no .h da árvore. As buscas por região **devem** ser otimizadas, levando em conta os bounding boxes das sub-árvores, de forma a apenas "descer" na sub-árvores que possam conter formas na região de busca.

A Saída

Devem produzidos os arquivos .txt e .svg resultantes do processamento dos arquivos .geo e .qry conforme anteriormente especificado.

Também devem ser produzido (para cada execução) o arquivo .dot que desenhe o resultado final da árvore. Veja:

- https://graphviz.org/
- https://graphviz.org/doc/info/lang.html
- https://graphviz.org/Gallery/directed/hello.html
- https://graphviz.org/Gallery/directed/Genetic_Programming.html

Avaliação

Espera-se uma atitude pró-ativa para a aquisição dos conhecimentos (i.e., estudo) para resolver o problema proposto.

A avaliação consistirá da execução dos testes e da inspeção de código. A nota será proporcional ao número de testes bem sucedidos, aplicados os descontos escritos abaixo.

ATENÇÃO: Os focos do projeto são: implementação e uso da Estrutura de Dados proposta e modularização bem feita. Hipoteticamente, se todo o resto estiver bem feito e esta parte mal feita, a nota será muito baixa.

ATENÇÃO: Caso algum tipo de FRAUDE seja detectada: Nota ZERO a TODOS os envolvidos.

Critério	Desconto
Escrever struct em arquivo .h	2.5
Modularização Pobre: .h mal projetado, mal documentado	até 1.5
Não implementado conforme especificado	tipicamente até 4.5, mas em casos graves, o desconto pode ser total
• não implementar estruturas pedidas ou implementação não funcional (gravíssimo) • não considerar bounding boxes na busca por região (grave) • não usar alguma função de percurso da árvore para gerar o arquivo .svg • faltar arquivo dot ou dot incorreto (p.ex., nós não pintados conforme solicitado)	
Procedimentos extensos e/ou complicados	até 1.0
Não usar o makefile provido	Em geral, nenhum desconto. Mas, se ocorrer problema de compilação ou execução que poderia ter sido evitado pelo uso do modelo do makefile provido, a consequência pode ser grave.
Erro de compilação	Nenhum teste será executado. Portanto, nota Zero, especialmente se for causado por negligência do aluno.

O Que Entregar

Submeter no Classroom o arquivo .zip com os fontes , conforme descrito na descrição geral.

RESUMO DOS PARÂMETROS DO PROGRAMA TED

Parâmetro / argumento	Opcional	Descrição
`-e` ***`path`***	S	Diretório-base de entrada (**BED**)
`-f` ***`arq`*`.geo`**	N	Arquivo com a descrição da cidade. Este arquivo deve estar sob o diretório **BED**.
`-o` ***`path`***	N	Diretório-base de saída (**BSD**)
`-q` ***`arqcons`*`.qry`**	S	Arquivo com consultas. Este arquivo deve estar sob o diretório **BED**.

RESUMO DOS ARQUIVOS PRODUZIDOS

-f	-q	comando com sufixo	arquivos
arq.geo			arq.svg arq.dot arq.csv
arq.geo	***arqcons***.qry		arq.svg arq-arqcons.svg arq-arqcons.txt arq-arqcons.dot arq-arqcons.csv
arq.geo	***arqcons***.qry	***sufx***	arq.svg arq-arqcons.svg arq-arqcons.txt arq-arqcons-sufx.[svg|txt][5]

ATENÇÃO:
* os fontes devem ser compilados com a opção `-fstack-protector-all`.
* adotamos o padrão C99. Usar a opção `-std=c99`.

5 Podem ser produzidos os respectivos arquivos .svg e/ou .txt, dependendo da especificação do comando.