


| | |
|---|---|
|  | <p>Centro Tecnológico Departamento de Informática</p> |
| <p>Disciplina: Computação Gráfica</p> | <p>Código: INF09282 e INF09284</p> |
| <p>Prof. Thiago Oliveira dos Santos</p> | |

Trabalho Curto 2

1 Introdução

Esse trabalho tem como objetivo aprimorar o conhecimento dos alunos em relação ao tópico de formato de arquivos gráficos e interatividade usando dispositivos gráficos e representação da informação visual.

Para isso, o aluno deverá implementar um programa que lerá, de um arquivo do tipo Scalable Vector Graphics (SVG), informações descrevendo uma arena e desenhará a arena com seus respectivos elementos na tela. A arena deverá ser estática, exceto pelo personagem do jogador que deverá se mover pela arena, voar e colidir com os objetos dela. Esse trabalho é o primeiro passo em direção a um jogo de guerra de aviões. O trabalho deverá ser implementado em C++ (ou C) usando as bibliotecas gráficas OpenGL e GLUT (freeglut).

2 Especificação das Funcionalidades


Ao rodar, o programa deverá ler, de um arquivo de configurações (denominado “config.xml”), as configurações necessárias para suas tarefas. O arquivo de configurações deverá estar no formato xml e será fornecido juntamente com a especificação do trabalho. A localização do arquivo “config.xml” será fornecida pela linha de comando ao chamar o programa. Por exemplo, se o arquivo estiver dentro de uma pasta chamada “Test1” localizada na raiz, basta chamar o programa com “/Test1/” como argumento (outros exemplos de caminhos possíveis “../Test1/”, “../../Test1/”, etc.). As informações contidas nesse arquivo servirão para ler o arquivo SVG contendo as informações da arena.

O arquivo de configurações deverá conter uma tag xml global <aplicacao> com uma sub-tag específica para descrever o arquivo de entrada da arena, denominada <arquivoDaArena>, e uma para descrever o jogador, denominada <jogador>. A tag <arquivoDaArena> terá outras subtags para descrever o nome, a extensão, e o caminho do arquivo descrevendo a arena (<nome>, <tipo> e <caminho> respectivamente). Perceba que o nome e o caminho do arquivo SVG descrevendo a arena será especificado por esses 3 atributos combinados, e que os caminhos do arquivo podem contemplar combinações do tipo: “../Test1/”, ou “./Test1/”, ou “~/Test1/”, entre outros. **Todos os caminhos relativos do arquivo da arena serão em relação ao diretório de execução do trabalho.** A tag <jogador> terá um atributo para descrever a velocidade do jogador (i.e., “vel”). Ela definirá o quanto o jogador se move a cada frame e estarão em unidades por milissegundos (como mostrado no laboratório). As velocidades do jogador e do tiro deverão ser invariantes a velocidade da máquina (i.e., o jogador com uma mesma informação no arquivo de configurações deve se movimentar com a mesma velocidade em máquinas mais lentas ou mais rápidas)¹. Percebam que os valores dados como exemplo foram escolhidos “aleatoriamente” e, portanto, podem não representar valores ótimos para teste. Ajuste os valores no *config.xml* para funcionar bem nas máquinas do laboratório.

Exemplo do arquivo config.xml com um arquivo SVG dado por: ../Test1/arena.svg

```
<aplicacao>
  <arquivoDaArena>
    <nome>arena</nome>
    <tipo>svg</tipo>
    <caminho>coloque o seu caminho aqui para testar</caminho>
  </arquivoDaArena>
  <jogador vel="0.1"></jogador>
</aplicacao>
```

¹ Para testar o correto funcionamento em velocidades de máquina diferentes, pode-se utilizar alguma função que coloque o *idle* em *sleep* por um curto tempo.

| | | |
|---|---|-----------------------------|
|  | Centro Tecnológico Departamento de Informática | |
| Disciplina: Computação Gráfica | | Código: INF09282 e INF09284 |
| Prof. Thiago Oliveira dos Santos | | |

Após ler as informações do arquivo de configurações, o programa deverá ler e interpretar os elementos da arena do arquivo do tipo SVG informado no arquivo de configurações e, desenhá-los na tela. A arena será composta por uma série de elementos (ver Figura 1): uma região de voo descrita por um círculo azul; um círculo verde representando o personagem do jogador; pelo menos um círculo vermelho representando inimigos voadores; pelo menos um círculo laranja representando inimigos terrestres; e uma linha preta representando a pista de decolagem. O avião do jogador estará sempre pousado no início da pista de decolagem. Todos esses elementos estarão contidos no arquivo SVG descrevendo a arena. Um arquivo SVG será fornecido como exemplo juntamente com a descrição do trabalho.

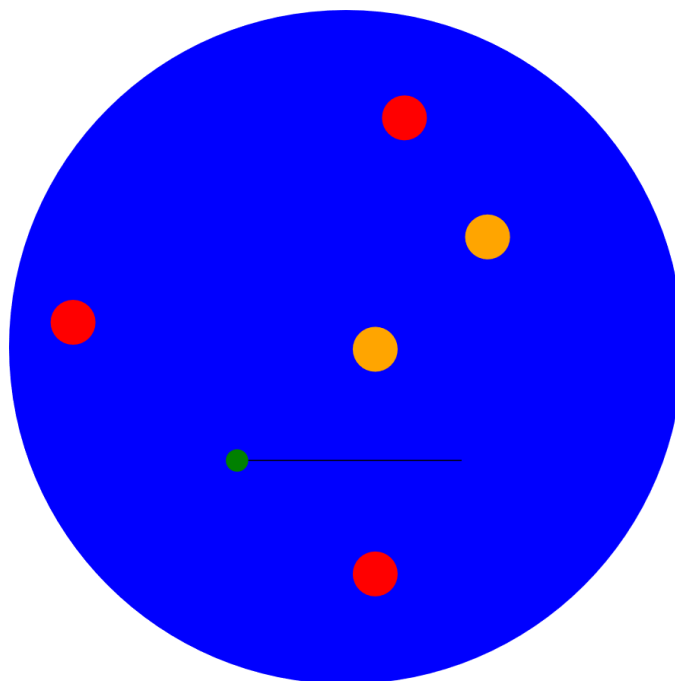



Figura 1: Exemplo de arena com seus elementos.

Cada um dos elementos da arena, contidos no SVG, tem informação suficiente para desenhá-los na tela (ex. posição, tamanho, cor, etc.). A arena deverá ser desenhada em uma tela do tamanho do diâmetro do círculo representando a região de voo, de forma que ele esteja todo contido na janela. A Figura 1 mostra o que deveria ser impresso com o arquivo fornecido como exemplo. Os outros elementos da arena deverão ser desenhados em cima da arena considerando as posições, tamanhos e cores lidos do arquivo SVG.

Todos os objetos serão estáticos exceto pelo personagem do jogador (i.e., círculo verde). O círculo do jogador deverá se mover ao pressionar as teclas do “w”, “s”, “a” e “d” respectivamente para cima, para baixo, para esquerda e para direita. Os movimentos deverão ser contínuos e permitir combinações de teclas para movimentos na diagonal (ex. ao manter “a” e “w” pressionados, o objeto deverá ser mover na diagonal esquerda para cima), conforme implementado no laboratório.

Colisões deverão ser tratadas para o personagem do jogador. Ele não poderá sair da região de voo (ou seja, extrapolar o círculo azul). Além disso, o círculo do jogador não poderá sobrepor o círculo do inimigo voador (ou seja, a colisão com os inimigos voadores deverá ser tratada). O círculo do jogador deverá poder passar por cima do inimigo terrestre.

O personagem do jogador deverá decolar da pista para começar a se mover pela arena. Portanto, ao iniciar o jogo ele estará parado no início da pista de decolagem. Ao se pressionar a tecla “u”, o processo de decolagem deverá iniciar e deverá durar 4 segundos. Durante a decolagem, o círculo deverá se mover até o final da pista aumentando a velocidade e, a partir da metade da pista, aumentando o seu tamanho, que deverá chegar ao dobro do raio inicial quando chegar ao final da pista e atingir a velocidade estabelecida no arquivo de configurações. Durante a decolagem, as teclas não deverão afetar o movimento. O jogador ganha o controle do personagem quando ele chegar ao final da pista.

| | | |
|---|---|-----------------------------|
|  | Centro Tecnológico Departamento de Informática | |
| Disciplina: Computação Gráfica | | Código: INF09282 e INF09284 |
| Prof. Thiago Oliveira dos Santos | | |

3 Regras Gerais

O trabalho deverá ser feito individualmente. Trabalhos identificados como fraudulentos serão punidos com nota zero. Casos típicos de fraude incluem, mas não se restringem à cópia de trabalhos, ou parte deles, assim como trabalhos feitos por terceiros. Caso seja necessário confirmar o conhecimento do aluno a respeito do código entregue, o professor poderá pedir ao aluno para apresentar o trabalho oralmente em um momento posterior. A nota da apresentação servirá para ponderar a nota obtida no trabalho.

3.1 Entrega do Trabalho

O código deverá ser entregue por email (para: todsantos@inf.ufes.br) dentro do prazo definido no portal do aluno. Trabalhos entregues após a data estabelecida não serão corrigidos.

A entrega do trabalho deverá seguir estritamente as regras a seguir. O não cumprimento acarretará na **não correção do trabalho** e respectivamente na atribuição da nota zero.

- Assunto da mensagem: [CG-2019-2] <tipo do trabalho>. Onde, <tipo do trabalho> pode ser TC1, TC2, TC3 e representa respectivamente trabalho curto 1, 2, 3, etc, ou TF para o trabalho final.
- Anexo da mensagem: arquivo zippado (com o nome do autor, ex. FulanoDaSilva.zip) contendo todos os arquivos necessários para a compilação do trabalho;
- Não enviar arquivos já compilados, inclusive bibliotecas!
- O diretório deverá necessariamente conter um makefile que implemente as seguintes diretivas "make clean" para limpar arquivos já compilados, "make all" para compilar e gerar o executável. O executável deverá ser chamado *trabalhocg*.

Lembre-se que a localização do arquivo config.xml será passada via linha de comando e, portanto, não se deve assumir que haverá um arquivo desses na pasta do executável. Seja cuidadoso ao testar o seu programa, isto é, não teste com o arquivo no diretório do programa, pois você pode esquecer de testá-lo em outro lugar posteriormente.

3.2 Pontuação

O trabalho será pontuado conforme a tabela abaixo. Bugs serão descontados caso a caso.

| Funcionalidade | Peso |
|--|------|
| Ler e desenhar a arena e seus elementos corretamente | 2 |
| Responder corretamente aos movimentos | 2 |
| Tratamento da colisão com inimigos | 2 |
| Tratamento da colisão com as bordas da pista | 2 |
| Decolar corretamente | 2 |

4 Erratas

Qualquer alteração nas regras do trabalho será comunicada em sala e no portal do aluno. É de responsabilidade do aluno frequentar as aulas e se manter atualizado.