	<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 3

### 1 Introdução

Esse trabalho tem como objetivo aprimorar o conhecimento dos alunos em relação ao tópico de transformações afins.

Para isso, o aluno deverá implementar um programa que coloque um avião visto de cima na arena implementada no trabalho curto 2. Nesse trabalho, a arena será estática, ou seja, não haverá movimentações, exceto do jogador. O jogador será controlado pelo teclado e pelo mouse, e poderá atirar com o canhão da frente, soltar bombas, decolar, e voar pela arena. 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 um multiplicador de velocidade do jogador (i.e., “vel”) e também um atributo para descrever um multiplicador de velocidade do tiro do jogador (i.e., “velTiro”). Esses valores deverão ser multiplicados pela velocidade final atingida pelo avião após o término da decolagem para definir a velocidade do avião e do tiro respectivamente.


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

```
<aplicacao>
  <arquivoDaArena>
    <nome>arena</nome>
    <tipo>svg</tipo>
    <caminho>../Test1/</caminho>
  </arquivoDaArena>
  <jogador vel="2.0" velTiro="1.0"></jogador>
</aplicacao>
```

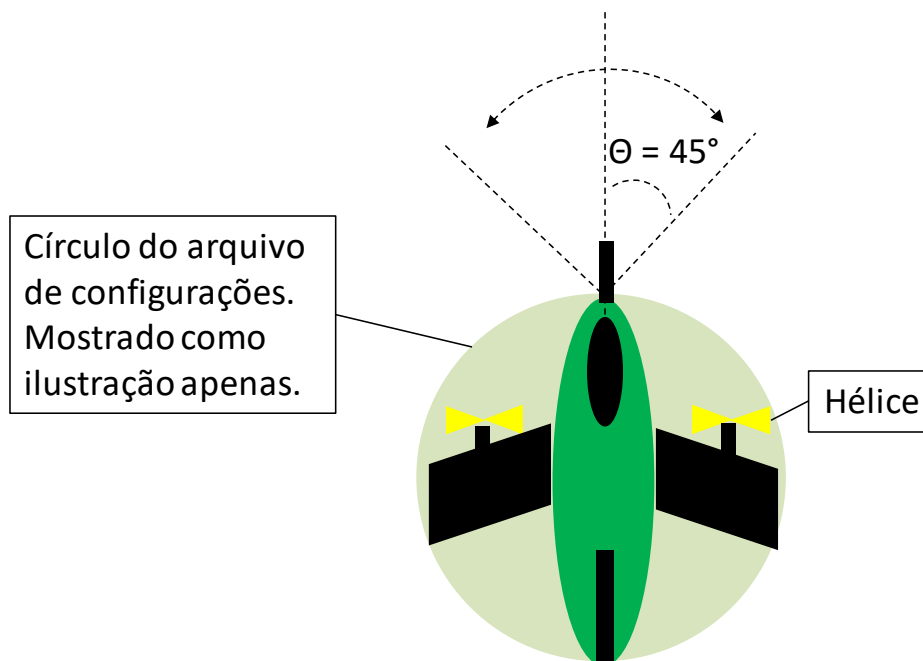
Após ler as informações do arquivo de configurações, o programa deverá carregar os elementos da arena do arquivo do tipo SVG respectivo e desenhá-los na tela (assim como feito no trabalho curto 2), exceto o círculo do personagem do jogador que terá o avião do jogador da Figura 1, ao invés de um círculo, apontando na direção da pista.

#### Jogador

O jogador deverá ser capaz de se movimentar pela arena e detectar as colisões com os inimigos voadores. Ao colidir com um inimigo voador, o jogo deverá terminar com a morte do jogador. O jogo poderá ser reiniciado a qualquer momento ao clicar “r”. O jogador deverá estar contido no círculo definido no arquivo de configurações (o canhão pode ficar de fora). O círculo servirá para calcular a colisão. O

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


jogador será composto principalmente por uma elipse verde, asas e suas hélices e um canhão, ver Figura 1 como exemplo.



*Figura 1: Avião com canhão (barra preta) que deve se movimentar de um lado para o outro, sendo no máximo a 45 graus para ambos os lados. As hélices são representadas pelos triângulos amarelos que girão com o movimento do avião.*

#### Voar

O jogador deverá se mover sempre para frente com a velocidade atingida após a decolagem. As teclas “+” e “-” servirão para aumentar e diminuir a velocidade de voo respectivamente. Os direcionais “a” e “d” controlarão respectivamente a curva do avião para a esquerda e para a direita em relação ao jogador. O avião deverá fazer curva como um arco e somente enquanto os direcionais estiverem pressionados. Portanto, ao soltar os direcionais o avião deverá manter o curso em trajetória retilínea. Note que o avião não deverá girar entre o próprio eixo e sim fazer uma curva suave. Adicionalmente, a curva deverá ser proporcional a velocidade atual do avião de forma que a velocidade do avião não afete a forma como sua curva é feita. Todos os movimentos do jogador deverão ser invariantes a velocidade da máquina. Ao atingir a borda da arena, o avião deverá ser transportado para o lado oposto da arena, considerando a direção da trajetória do avião. Isto é, ao chegar no final da arena por um lado, ele aparece no outro (ver exemplo na Figura 2). Isto permitirá que o avião fique voando continuamente sem parar ao chegar no final da arena.

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

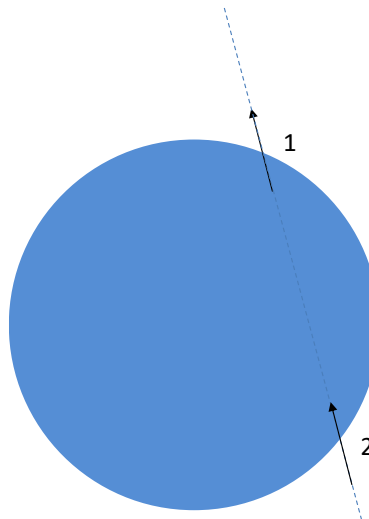


Figura 2: Exemplo do avião saindo da arena no ponto 1 e reentrando no ponto 2.

#### *Hélices*

As hélices serão usadas para representar o movimento do avião. Elas deverão girar continuamente com a velocidade proporcional a velocidade do avião.

#### *Decolagem*

O jogador deverá implementar as mesmas funcionalidades de decolagem do segundo trabalho, ou seja, ao apertar a tecla “u”, ele deverá decolar conforme definido no trabalho 2. Porém, após a decolagem, o avião deverá continuar voando com a velocidade atingida.

#### *Atirar*


O avião terá um canhão na sua dianteira e ele poderá se mover 45 graus para cada lado de acordo com o movimento do mouse, ou seja, mover o mouse para esquerda gira o canhão para a esquerda do jogador e mover para a direita gira para a direita do jogador. O jogador deverá atirar na direção para a qual o canhão estiver apontando sempre que o botão esquerdo do mouse for clicado, e uma vez atirado, o tiro se torna independente do avião, ou seja, alterações na rota do avião não devem influenciar a rota do tiro que deve continuar em linha reta. Perceba que tiros que saírem da arena podem ser descartados. A velocidade do tiro deverá levar em conta a informação vinda do arquivo de configurações e deverá aumentar juntamente com a velocidade do avião quando ela for alterada. Tiros não devem interagir com os obstáculos nesta versão do trabalho.

#### *Bombardear*

O avião poderá soltar bombas ao clicar com o botão direito do mouse. A bomba (representada por um círculo com diâmetro da largura do meio do avião) sairá do meio do avião e seguirá uma trajetória retilínea com a velocidade do avião caindo por 4 segundos até atingir o chão e sumir. O tamanho da bomba deverá diminuir até a metade do seu diâmetro, ao atingir o chão, para representar a sua queda. Note que a bomba não será vista se o avião continuar voando em trajetória retilínea após soltar a bomba. Bombas não devem interagir com os obstáculos nesta versão do trabalho.

O aluno deverá utilizar os conceitos de *double buffer* e variável de estado das teclas para interação com teclado (como visto em aula e feito no TC2). A utilização de conceitos de modularização (e.g. usando classes para representar os objetos da cena) facilitará a implementação dos trabalhos seguintes, como por exemplo uma classe jogador com funções moverParaFrente, girar, etc. Evite usar variáveis globais dentro das funções, de preferência à passagem por parâmetros, mesmo que os argumentos venham de variáveis globais. Restrinja o uso de variáveis globais às chamadas mais externas.

### **3 Regras Gerais**

	<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>	

O trabalho deverá ser feito individualmente. Trabalhos identificados como fraudulentos serão punidos com nota zero. Casos típicos de fraude incluem, mas não se restringem às cópias de trabalhos, ou parte dele, 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.

## 4 Pontuação

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

Funcionalidade	Peso
Jogador	1
Colisão com inimigos	1
Reiniciar jogo	1
Voar	1
Transporte para o lado oposto da arena	1
Canhão	1
Atirar com canhão	1
Bombardear	1
Hélices	1
Decolar	1

## 5 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.