	<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 uma pessoa vista de cima na arena implementada no trabalho curto 2. Nesse trabalho, a arena será estática, ou seja, não haverá movimentações ou interações de objetos, exceto do jogador. O jogador será controlado pelo teclado e pelo mouse, e poderá atirar, pular e andar pela arena, considerando as colisões do TC2. 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.

Além das tags já especificadas no trabalho curto 2, o arquivo de configurações deverá conter uma sub-tag específica para descrever parâmetros do jogador, denominada <jogador>. A tag <jogador> terá atributos para descrever a velocidade do tiro, e do jogador (“velTiro” e “vel” respectivamente). Elas definirão o quanto um tiro e o jogador se movem a cada frame e deverão estar em unidades por milissegundos (como mostrado no laboratório). Percebam que os valores dados como exemplo foram escolhidos aleatoriamente e, portanto, podem não representar valores ótimos para teste.


Exemplo do arquivo config.xml

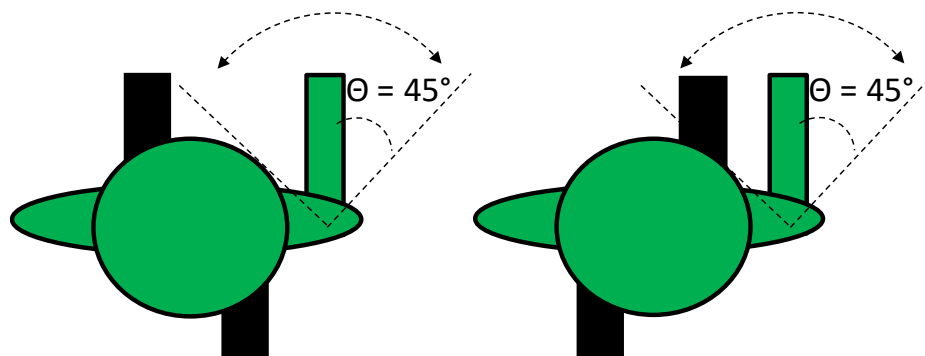
```
<aplicacao>
  <arquivoDaArena nome="arena" tipo="svg" caminho=" caminho para testar"></arquivoDaArena>
  <jogador velTiro="0.2" vel="0.1"></jogador>
</aplicacao>
```

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

#### *Jogador*

O jogador deverá ser capaz de se movimentar pela arena e detectar as colisões e realizar as ações já implementadas no trabalho anterior. Ele iniciará na posição definida pelo centro do círculo verde na arena e terá o tamanho aproximado do círculo que o representa, ou seja, o jogador deverá estar contido no círculo. O jogador será composto por uma cabeça, ombros, um braço que se movimentará para atirar, e dois pés que deverão se alternar indo para frente e para trás enquanto o jogador se move, ver Figura 1 como exemplo.

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



*Figura 1: Jogador com o pé a esquerdo a frente e o pé direito atrás (esquerda) e jogador com o pé direito à frente e pé esquerdo atrás (direita).*

#### Andar

O jogador deverá se mover para frente e para trás ao pressionar as teclas do “w” e “s” respectivamente. Os pés do jogador deverão se alternar indo para frente e para trás enquanto ele se movimentar. A velocidade do jogador deverá levar em conta a informação de velocidade vinda do arquivo de configurações. O jogador deverá girar em torno de seu centro ao se pressionar as teclas direcionais “a” e “d”. Os direcionais “a” e “d” controlarão respectivamente o giro no sentido anti-horário e horário em relação ao jogador. O programa deverá permitir combinações de teclas para andar e girar ao mesmo tempo, sendo que a componente de andar deverá ser mais influente do que a componente de girar (isto é, fazendo com que o jogador ande fazendo uma curva aberta e não fique girando em torno do próprio eixo). Essa proporção deverá levar em conta a informação de velocidade do jogador vinda do arquivo de configurações.

#### Pular


O jogador deverá implementar as mesmas funcionalidades de pulo do segundo trabalho, ou seja, ao apertar a tecla “p” em qualquer lugar da arena, ele deverá pular. Para simular o pulo, o jogador deverá aumentar o seu tamanho continuamente do tamanho original a até 1.5 vezes o tamanho original. Assim como no segundo trabalho, o pulo servirá para ultrapassar, ou subir nos, obstáculos baixos. Uma vez em cima do obstáculo baixo, o jogador deverá se movimentar normalmente. Uma vez fora dele, o jogador deverá pular novamente para subir nele.

Um pulo deverá durar 2 segundos, portanto o tamanho máximo deverá ser atingido em aproximadamente 1 segundo (sendo 1 segundo crescendo e outro diminuindo). Ao manter a tecla “p” pressionada, o jogador deverá ficar pulando continuamente, ou seja, dar um pulo atrás do outro.

#### Atirar

O jogador deverá atirar na direção em que o braço estiver apontando sempre que o botão esquerdo do mouse for clicado. O braço do jogador será controlado pelo deslocamento lateral do mouse, ou seja, arrastar o mouse para a direita move o braço no sentido horário e arrastar para a esquerda move no sentido anti-horário em relação ao jogador. O braço não deverá mover mais do que 45 graus do centro, ver Figura 1. Perceba que tiros que saírem da arena podem ser descartados. O tiro poderá ser representado por um círculo pequeno, ou um ponto. A velocidade do tiro deverá levar em conta a informação vinda do arquivo de configurações. Tiros não devem ultrapassar os obstáculos baixos, ou seja, eles devem sumir ao atingi-los. Os obstáculos altos não precisam ser tratados no momento. O jogador não deve atirar enquanto pula ou enquanto estiver em cima do obstáculo.

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.

	<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>	

### 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 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: todosantos@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-2017-2] <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	2
Pular	2
Andar	2
Atirar	2
Mover braço	2

### 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.