Universidade Tecnológica Federal do Paraná
Departamento Acadêmico de Informática
ICSF13 – Fundamentos de Programação 1
Prof. Bogdan Tomoyuki Nassu
Profa. Leyza Baldo Dorini
Prof. Daniel Fernando Pigatto

Trabalho 3

I) Descrição geral

Lançado no final de 2006, o Nintendo Wii foi um console de videogame que concorreu com o Playstation 3 (da Sony) e o Xbox 360 (da Microsoft) na chamada "7ª geração dos videogames". Mesmo tendo poder computacional inferior aos seus concorrentes, o Wii vendeu mais unidades: mais de 100 milhões de consoles no total, sendo um dos videogames mais vendidos da história.

O sucesso do Wii se deveu, em grande parte, ao seu esquema de controle, radicalmente diferente daquilo que se encontrava em produtos anteriores e contemporâneos. O controle, apelidado de Wiimote (ver Fig. 1), traz dentro de si uma série de sensores de movimento e inclinação (acelerômetros e giroscópios), que permitem que os jogos sejam controlados por movimentos do próprio controle, e não apenas pelos tradicionais botões e alavancas. Aliado a jogos simples e acessíveis, os controles de movimento do Wii criaram um apelo junto a um público "casual", que normalmente não consumia jogos eletrônicos.

Figura 1: Wiimote

Além de ser um controle por movimento, o Wiimote também pode ser usado como um "apontador". O jogador aponta o controle para a tela e movimenta um cursor / mira, de forma similar a um mouse. Esta forma de controle é realizada com o auxílio da *sensor bar*, uma pequena barra que é posicionada logo acima ou abaixo do televisor, como mostra a Fig. 2.

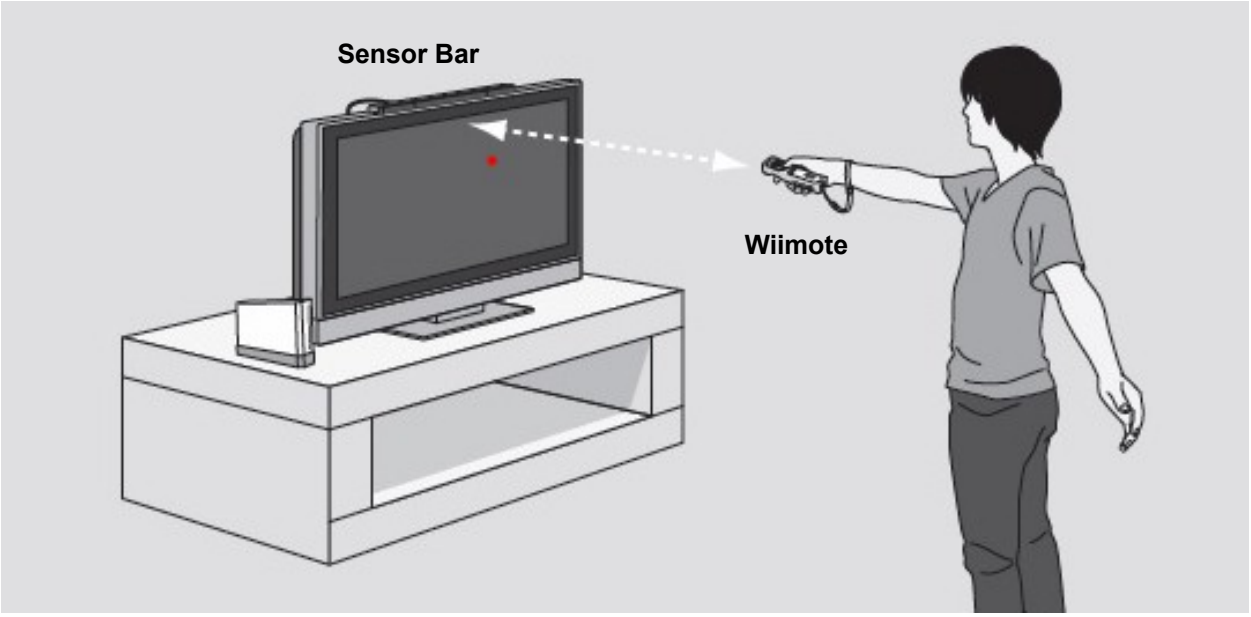

Figura 2: Wiimote + sensor bar.

Apesar da aparência, o conjunto Wiimote + sensor bar é muito diferente de um controle remoto convencional. Um controle remoto convencional tem um LED infravermelho na ponta, que pisca segundo algum padrão quando um botão é pressionado. O padrão é então reconhecido por um sensor presente no aparelho (por exemplo, no televisor). Já o controle do Nintendo Wii funciona no sentido inverso: apesar do nome, a sensor bar não possui sensores – ela tem dois conjuntos de LEDs infravermelhos, um em cada ponta. Na ponta do controle, há um sensor, que funciona basicamente como uma câmera de baixa resolução, sensível apenas à faixa de frequência dos LEDs presentes na sensor bar. Com isso, o Wiimote "enxerga" uma imagem escura, com 2 regiões claras que correspondem aos conjuntos de LEDs da sensor bar, como mostra a Fig. 3.


Figura 3: sensor bar "vista" pelo Wiimote (simulação).

Suponha que o ano é 2004, e a sua equipe trabalha no centro de pesquisas da Nintendo, no "Project Revolution" - que viria a se tornar o Wii. Vocês estão trabalhando no desenvolvimento do algoritmo para detectar o sensor bar. A sua equipe deve criar uma função com o seguinte protótipo:

```
double detectaSensorBar (Imagem1C* img, Coordenada* l, Coordenada* r);
```

Esta função recebe como parâmetro de entrada um ponteiro para uma imagem do tipo `Imagem1C`. Esta imagem é semelhante àquela mostrada na Fig. 3: escura e com 2 regiões mais claras. O objetivo da função é processar a imagem e localizar os centros dos dois agrupamentos de LEDs. Considere que a imagem de entrada pode ser modificada pela função (i.e. o conteúdo da imagem pode ser destruído).

Neste ponto do projeto, o sensor que será colocado no Wiimote ainda não foi escolhido, portanto a sua função deve ser capaz de trabalhar com imagens de resolução arbitrária (no Wii real, a resolução é de 128x96 pixels). Note também que a imagem não é simplesmente "preta com círculos brancos". O ambiente pode criar reflexos, podem existir outras fontes de luz infravermelha, e a necessidade de manter o custo do sensor baixo implica na presença de ruído. Além disso, a luz dos LEDs cria um efeito "borrado" ao seu redor.

Além da imagem de entrada, a função recebe 2 parâmetros de saída, `l` e `r`. Eles são ponteiros para variáveis do tipo `Coordenada`, uma `struct` declarada no arquivo trabalho3.h. Estes parâmetros devem ser preenchidos com as coordenadas dos centros dos agrupamentos esquerdo (`l`) e direito (`r`). Ao final, a função deve retornar o ângulo entre os dois agrupamentos, em radianos, no intervalo (-π/2, +π/2).

Para criar o trabalho, será fornecido um "pacote" contendo os seguintes arquivos:

- gerador_de_testes.c, gerador_de_testes.h: módulo gerador de testes. É invocado pelo programa testador – você não precisa se preocupar com o que existe dentro desses arquivos.

- imagem.c e imagem.h: contêm a declaração e a implementação de tipos e funções para manipulação de imagens no formato bmp.

- trabalho3.h: contém a declaração da função `detectaSensorBar` e do tipo `Coordenada`. Este arquivo deve ser incluído (através da diretiva `#include`) no arquivo contendo a implementação da sua função.

- main.c: contém uma função `main` (i.e. um programa) que testa o trabalho. No começo do arquivo, existem 3 macros, que podem ser usadas para controlar os testes:

`RANDOM_SEED_OFFSET`: é a semente usada pelo gerador de números aleatórios. Modifique este valor para gerar casos de teste diferentes. Os testes finais serão realizados com um valor diferente daquele presente no arquivo original.

`N_TESTES`: número de testes a realizar.

`SALVA_INTERMEDIARIOS`: se tiver valor diferente de 0, as imagens passadas como parâmetro para a função central serão salvas.

A montagem do projeto, a forma de se representar e manipular imagens, e as estratégias que podem ser usadas para resolver o problema serão discutidos em aula.

II) Equipes

O trabalho deve ser feito em equipes de 3 pessoas. Equipes menores só serão aceitas se devidamente justificadas e previamente autorizadas pelos professores. A justificativa deverá ser um motivo de força maior: questões pessoais como "prefiro trabalhar sozinho" ou "não conheço ninguém da turma" não serão aceitas. Cada equipe deve apresentar sua própria solução para o problema. Indícios de fraude (cópia) podem levar à avaliação especial (ver item V).

III) Entrega

O prazo de entrega é 11/12/2023. Trabalhos entregues após esta data terão sua nota final reduzida em 0,00025% para cada segundo de atraso. Cada equipe deve entregar, através da página da disciplina no Moodle, dois arquivos (separados, sem compressão):

- Um arquivo chamado *t3-x-y-z.c*, onde *x, y* e *z* são os números de matrícula dos alunos. O arquivo deve conter a implementação da função `detectaSensorBar` (com cabeçalho idêntico ao especificado), assim como de quaisquer outras funções auxiliares usadas no trabalho. Funções da biblioteca-padrão também

podem ser usadas. Os autores do arquivo devem estar identificados no início, através de comentários. Separe as sub-tarefas em funções, para organizar o seu código.

IMPORTANTE: as funções pedidas não envolvem interação com usuários. Elas não devem imprimir mensagens (por exemplo, através da função `printf`), nem bloquear a execução enquanto esperam entradas (por exemplo, através da função `scanf`). O arquivo também não deve ter uma função `main`.

- Um arquivo no formato PDF chamado *t3-x-y-z.pdf*, onde *x, y* e *z* são os números de matrícula dos alunos. Este arquivo deve conter um relatório breve (em torno de 2 a 3 páginas), descrevendo em detalhes a contribuição de cada membro da equipe, os desafios encontrados, e a forma como eles foram superados. Não é preciso seguir uma formatação específica. Os autores do arquivo devem estar identificados no início.

IV) Avaliação (normal)

Todos os testes serão feitos usando a IDE Code::Blocks. Certifique-se de que o seu trabalho pode ser compilado e executado a partir dela.

Os trabalhos serão avaliados por meio de baterias de testes automatizados. Os testes estão implementados no arquivo main.c fornecido. A nota final será composta por uma nota base e descontos aplicados por conta de problemas encontrados. A nota base será decidida por uma "competição" envolvendo todos os trabalhos entregues, além de uma implementação de referência feita pelo professor, sem otimizações ou "truques" sofisticados. As notas específicas serão atribuídas com base na colocação de cada trabalho na competição. Todos os trabalhos que obtiverem desempenho superior à implementação de referência terão a nota base superior a 100.

O desempenho será medido objetivamente, com base em estatísticas computadas a partir das distâncias entre os centros detectados e os centros reais. Se quiser entender o cálculo da pontuação, verifique o mesmo no arquivo main.c. O tempo de execução será usado como critério de desempate, e também para gerar um multiplicador no intervalo [0.8, 1], definido de acordo com a velocidade de resposta de cada solução comparada à referência. Se vários trabalhos tiverem desempenho similar, os parâmetros do gerador de testes poderão ser modificados, de forma a produzir imagens de pior qualidade.

Os descontos sobre a nota base serão aplicados com base nos seguintes critérios:

IV.a) Compilação e corretude. Cada erro que impeça a compilação do arquivo ou que cause um erro de execução será corrigido pelo professor. A cada erro corrigido, a nota da função será multiplicada por um valor, que pode chegar a 0.7 em casos de erros facilmente detectáveis. Erros cuja correção seja complexa demais serão tratados como fracassos para os casos de teste afetados.

IV.b) Atendimento da especificação. Serão descontados até 10 pontos para cada item que não esteja de acordo com esta especificação, tais como nomes de arquivos e funções fora do padrão.

IV.c) Documentação. Descreva através de comentários o funcionamento das suas funções. Não é preciso detalhar tudo linha por linha, mas forneça uma descrição geral da sua abordagem, assim como comentários sobre estratégias que não fiquem claras na leitura das linhas individuais do programa. Uma função sem comentários terá sua nota reduzida em até 25%.

IV.d) Organização e legibilidade. Use a indentação para tornar seu código legível, e mantenha a estrutura do programa clara. Evite construções como *loops* infinitos terminados somente por `break`, ou *loops* `for` com varías inicializações e incrementos, mas sem corpo. Evite também replicar blocos de programa que poderiam ser melhor descritos por repetições. Um trabalho desorganizado ou cuja legibilidade esteja comprometida terá sua nota reduzida em até 30%.

IV.e) Variáveis com nomes significativos. Use nomes significativos para as variáveis – lembre-se que `n` ou `x` podem ser nomes aceitáveis para um parâmetro de entrada que é um número, mas uma variável representando uma "soma total" será muito melhor identificada como `soma_total`, `soma` ou `total`; e não como `t`, `aux2` ou `foo`. Uma função cujas variáveis internas não tenham nomes significativos terá sua nota reduzida em até 15%.

IV.f) Siga as convenções para nomear constantes, funções e variáveis. Um trabalho que tenha identificadores fora das convenções pode ter sua nota reduzida em até 10%.

IV.g) Organize seu código e use funções, tendo em mente a modularização e a legibilidade. Um trabalho monolítico ou tendo funções muito longas/confusas terá sua nota reduzida em até 25%.

IV.h) Desempenho. Se a estrutura lógica de uma função for pouco eficiente, por exemplo, realizando muitas operações desnecessárias ou redundantes, a sua nota pode ser reduzida em até 20%.

V) Avaliação (especial)

Indícios de fraude ou de divisão desigual do trabalho podem levar a uma avaliação especial, com os alunos sendo convocados e questionados sobre aspectos referentes aos algoritmos usados e à implementação. Além disso, alguns alunos podem ser selecionados para a avaliação especial, mesmo sem indícios de fraude, caso exista uma grande diferença entre a nota do projeto e a qualidade das soluções apresentadas em listas de exercícios, ou se a explicação sobre o funcionamento de uma função for pouco clara.

VI) Apoio

Os professores estarão disponíveis para tirar dúvidas a respeito do trabalho, nos horários de atendimento previstos (e em casos excepcionais, fora deles). A comunicação por e-mail pode ser usada para pequenas dúvidas sobre aspectos pontuais.