REVISTA DE COMPUTAÇÃO E SISTEMAS (RCS)
ISSN XXXX-XXXX
VOLUME 1 – NÚMERO 1 – ABRIL/2016
http://www.revistacomputacaoesistemas.net

Framework de Interface Gestual para Windows Phone 7

Hugo F. De Campos, Gabriel M. Matsuda, Ricardo M. dos Santos, Luciano Silva

Faculdade de Computação e Informática – Universidade Presbiteriana Mackenzie
Caixa Postal 930 – 01302-907 – São Paulo – SP – Brazil

hugofcampos@gmail.com, gabriel.matsuda@gmail.com,
ricardo.machezini@gmail.com, luciano.silva@mackenzie.br

Abstract. *The Gesture Interface field is one of the most promising fields in Human-Computer Interaction research area. Nowadays, the number of people that have access to computer devices is rising incredibly. This level of access to technology creates the need of having new methods that allow the interaction between the user and the application and also, that this interaction to be done intuitively, in way that the user should not need any previous knowledge. Everyday new technologies are created and these technologies provide new means of interaction. It makes that the treatment of these new technologies becomes even more wide and complex. This work proposes the development of a framework to the Microsoft Windows Phone 7 platform, in order to help the development of electronic games that implement the Gesture Interface. A tool that allows to automate the process of detection, treatment and handling of gestures, such as touch and accelerometer, improving the game development process, using the Microsoft XNA Framework.*

Resumo. *O campo Interfaces Gestuais é um dos campos de pesquisa mais promissores da área de Interação Humano-Computador. Atualmente, o número de pessoas com acesso a dispositivos computacionais cresce incrivelmente. Tanto acesso a tecnologia cria a necessidade de novas formas de fazer com que o usuário interaja com a aplicação e que seja feito uma forma intuitiva que não necessite de conhecimento prévio do usuário. A cada dia novas tecnologias são criadas e elas disponibilizam novos meios de interação. Isso faz com que o tratamento desses recursos se torne mais amplo e complexo. Neste trabalho, propõe-se o desenvolvimento de um framework para a plataforma Microsoft Windows Phone 7, para auxiliar no desenvolvimento de jogos eletrônicos que implementem Interfaces Gestuais. Uma ferramenta desse tipo permite automatizar o processo de detecção, tratamento e manipulação de gestos, por exemplo de gestos do tipo toque e acelerômetro, facilitando e organizando o processo de desenvolvimento de jogos, utilizando como base o XNA Framework.*

1. Introdução

Dentro da área de Interação Humano-Computador, um dos campos de pesquisa mais promissores é o de Interfaces Gestuais. À medida que surgem novas tecnologias em termos de *hardware* para interação entre o homem e a máquina, surgem também novas formas de interagir com o computador. Interface Gestual é a técnica que permite ao usuário interagir com computadores usando os movimentos de seu corpo.

Na área de jogos eletrônicos, a jogabilidade é um dos requisitos mais importantes a serem considerados para o desenvolvimento do jogo. Em plataformas convencionais de jogo, como Playstation e XBOX, a interação do jogador com o jogo é feita por meio de dispositivos de *hardware* específicos para controle (*joysticks*). Já em plataformas não convencionais, em que não existe um dispositivo *hardware* específico, uma boa Interface Gestual deve ser aplicada para se obter uma boa jogabilidade.

Em dispositivos móveis, diversas tecnologias são utilizadas para a interface de controle de usuário. Em especial, dispositivos de telefonia móvel são exemplos dessa aplicação e representam um mercado crescente de aquisição pela popularização dos *smartphones*. Essa plataforma ganhou notoriedade após o lançamento do primeiro modelo do iPhone, da Apple, que substituía os teclados numéricos ou do tipo "Querty" por um conjunto de sensores que proporcionavam maior interação com os usuários. Desses sensores, destacavam-se a tela sensível ao toque e o acelerômetro. Após a introdução desse conceito, diversas outras plataformas surgiram com a mesma proposta, das quais se destacam os celulares com sistema operacional Android e, mais recentemente, Windows Phone 7.

Com a popularização dos *smartphones*, o mercado de jogos para essa nova plataforma se torna cada vez mais explorado. E dado a constante evolução dos equipamentos, em relação aos seus recursos tecnológicos de entrada de usuário, principalmente sensores, a utilização desses recursos foi tomada como cenário para este trabalho.

O objetivo deste trabalho é a criação de um *framework,* baseado no XNA Framework da Microsoft, para automatizar a detecção, o tratamento e a manipulação de gestos a fim de facilitar o desenvolvimento de jogos para a plataforma Windows Phone 7. O XNA Framework atualmente não fornece componentes nativos, já agregados para detecção, tratamento e manipulação de gestos. Portanto, um *framework* que automatizasse esse processo possibilitaria a abstração da lógica de tratamento de gestos da lógica do jogo em si, aumentando a organização e coesão do código do jogo.

2. Interfaces Gestuais

Na busca da maior interatividade entre jogos eletrônicos e seus jogadores, requisito fundamental para o sucesso do jogo, dois fatores se fazem indispensáveis: a plataforma e a técnica de controle.

A plataforma representa o tipo de equipamento em que este jogo será reproduzido. Pela plataforma define-se qual será o ambiente que o jogador deverá ser inserido para a prática do jogo. Esse aspecto implicará, invariavelmente, nas primeiras restrições em relação ao tipo de controle que será utilizado. Uma plataforma fixa,

como um console, por exemplo, pode impor uma restrição ao número de jogadores simultâneos por uma questão espacial. Da mesma maneira, uma plataforma que utiliza movimentação deverá sofrer limitação na utilização do espaço para o jogo.

A técnica diz respeito a como o jogo irá coletar as informações do jogador para a interação com o ambiente virtual do jogo. Semelhante ao que acontece com a plataforma, a técnica representa outro conjunto de restrições em relação ao jogo. Um controle feito por botões, por exemplo, tem a limitação de número e complexidade de combinações que podem ser feitas. Já jogos que utilizam movimentação devem levar em consideração a natureza do movimento e a viabilidade de utilizá-lo em relação ao espaço físico.

A partir da análise desses dois pontos, é possível estudar a melhor forma de interação entre jogador e jogo. Em particular, este trabalho empenha-se na utilização de gestos para a interação. Para isso, será utilizado o conceito de Interface Gestual, que estuda o comportamento de gestos na interação Humano-Computador. Definir o que é um gesto é algo que muitos pesquisadores tentaram fazer de diversas formas. Todos dizem saber o que um gesto é, mas nenhum pôde definir precisamente o que é [Corradini & Cohen, 2002]. Os movimentos de um corpo definem um aspecto do que é um gesto, embora a maioria das definições tente separar gestos de outros tipos de movimentação humana.

Interface Gestual é a interface do computador que permite ao usuário interagir com computadores usando os movimentos de seu corpo [Cheng et al., 2011]. É um dos campos mais promissores para pesquisadores da área de Interação Humano-Computador, pois faz com que o computador responda melhor as habilidades de cada pessoa, fazendo assim o uso mais intuitivo e prazeroso do que o padrão *mouse* e teclado. Como exemplos de aplicações de interfaces gestuais, pode-se ter:

- **Reconhecimento de *pointing directions***: por meio de uma câmera, o robô capta as imagens do ambiente, reconhece a posição das mãos e da cabeça do usuário e assim consegue estimar a direção para onde este está apontando [Nickel & Stiefelhagen, 2007], conforme mostra a Figura 1:

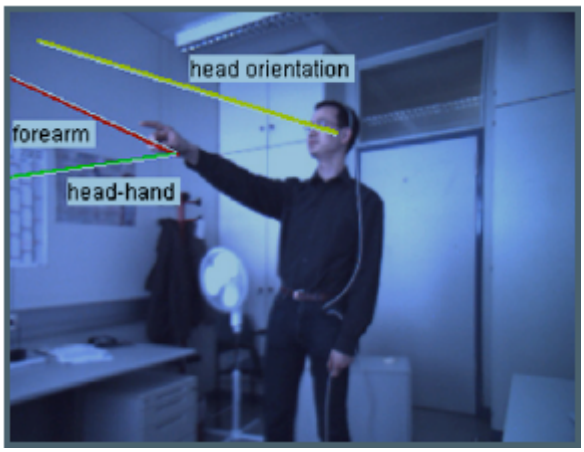

Figura 1: Diferentes abordagens para estimar a *pointing direction*.

Fonte: Nickel & Stiefelhagen, 2007.

- **Reconhecimento de expressões faciais:** por meio de uma câmera, o robô capta as imagens do rosto do usuário, extrai as características faciais e então classifica as expressões em um tipo [Huang & Lin, 2008], como mostra a Figura 2, próxima página.

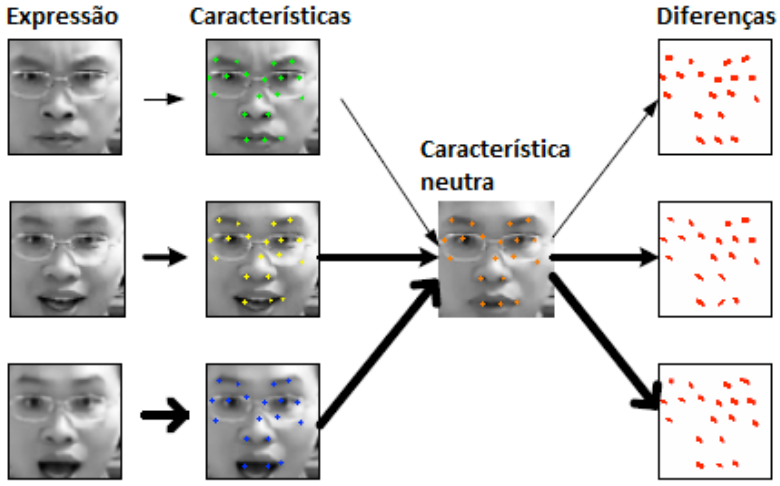

Figura 2 – Diferenças entre expressão facial neutra e outras expressões

Fonte: Huang & Lin, 2008.

3. Interfaces Gestuais aplicadas a Jogos Eletrônicos

Conforme descrito na seção anterior, dois fatores são decisivos para o sucesso de um jogo eletrônico do ponto de vista de interatividade entre o jogador e o jogo: a plataforma e técnica de controle.

O objetivo desta seção é apresentar alguns exemplos de jogos eletrônicos de plataformas atuais e destacar as formas com as quais as aplicações de interfaces gestuais foram aplicadas para solucionar o problema da interação entre o jogador e o jogo eletrônico.

3.1. Brothers In Arms® Hour of Heroes

Neste jogo de tiro em terceira pessoa para iPhone, o jogador controla um soldado avançando pelo campo de guerra. Na tela *multitouch*, além do botão de tiro, é apresentado um *joystick* virtual com o qual o jogador consegue controlar os movimentos do personagem pelo cenário. Ao tocar nas outras partes da tela o jogador consegue alterar a posição da câmera [Gameloft, n.d.].

A Figura 3 mostra uma das telas do jogo Brothers In Arms:

Figura 3 – Brothers In Arms® Hour of Heroes

Fonte: Gameloft, n.d.

3.2. Rise of Glory

Este jogo para Windows Phone 7, tem por objetivo realizar diversas missões, que podem ser do tipo ir de um determinado ponto a outro, passar por um conjunto de anéis (*checkpoints*) em um tempo determinado, e destruir aviões inimigos [Microsoft, Rise of Glory, n.d.].

O controle do jogo é dividido em dois, uma parte pela interface *touch* que controla as armas e funções básicas do avião, como ligar e desligar o motor. A outra parte é feita por acelerômetro, que controla o manche do avião conforme a movimentação do próprio aparelho. Rotacionar o *smartphone* em relação à tela nos sentidos horário e anti-horário, faz com que o avião gire em torno de seu próprio eixo. Já ao incliná-lo para frente e para trás, faz com que o avião empine ou mergulhe.

Figura 4 – Rise of Glory

Fonte: Microsoft, n.d.

4. XNA Gesture Interface Framework

Nesta seção é proposto um *framework* que tem como objetivo disponibilizar ao usuário que procura desenvolver jogos para plataforma Windows Phone 7 um conjunto de ferramentas que automatize a detecção, o tratamento e a manipulação de gestos para ser utilizado durante a lógica de programação do jogo.

Para isso, antes da implementação, foi necessário fazer um estudo para levantar os requisitos e identificar como o *framework* seria elaborado.

4.1. Requisitos do *Framework*

Assim como em todo desenvolvimento de *software*, para o desenvolvimento do *framework* proposto por este trabalho foi necessário definir os requisitos. A lista a seguir mostra os requisitos finais propostos para o *framework*.

- Reconhecer gestos pré-estabelecidos por uma biblioteca de suporte;
- Ser capaz de armazenar e reconhecer uma sequência de gestos;
- Ser capaz de guardar o último gesto reconhecido;
- Possibilitar ao usuário estender o próprio *framework* para implementação do jogo utilizando os recursos disponibilizados pelo *framework*;

4.2. Diagrama de Classes

Nesta seção apresentamos o diagrama de classes do *framework* desenvolvido juntamente com a descrição de cada classe:

- **Gesture**: Classe que denota o gesto genérico para o sistema, sendo extensível para que a implementação dê cobertura para qualquer tipo de gesto seja ele de um tipo *touch* ou de uma leitura do acelerômetro.
- **State**: Classe que denota o estado atual do dispositivo, para que seja possível a compreensão de gestos complexos ou sequenciais, como por exemplo, *double tap* e *pinch*.
- **Manager**: Gerenciador de recursos do *framework* contém a inicialização do sistema e mantém a referência dos membros essenciais para o *framework* (Gesture List, State, ActionMediator e Monitor).
- **GestureList**: Classe que lista as ações aceitas pelo sistema, contém as ferramentas para a manipulação da mesma, é necessária para suportar gestos sequenciais ou complexos, como por exemplo, *double tap* e *pinch*.
- **ActionMediator**: Classe que recebe os dados de entrada, e faz todo o trabalho de tratamento daquela entrada, ou seja, é a classe que toma a ação quando um

gesto é obtido pelo sistema, e faz a atualização do estado do sistema de acordo com os gestos válidos aceitos.

- **Monitor**: Classe que espera o evento, quando o evento ocorre o Monitor repassa para que este evento seja tratado pela classe Mediator.

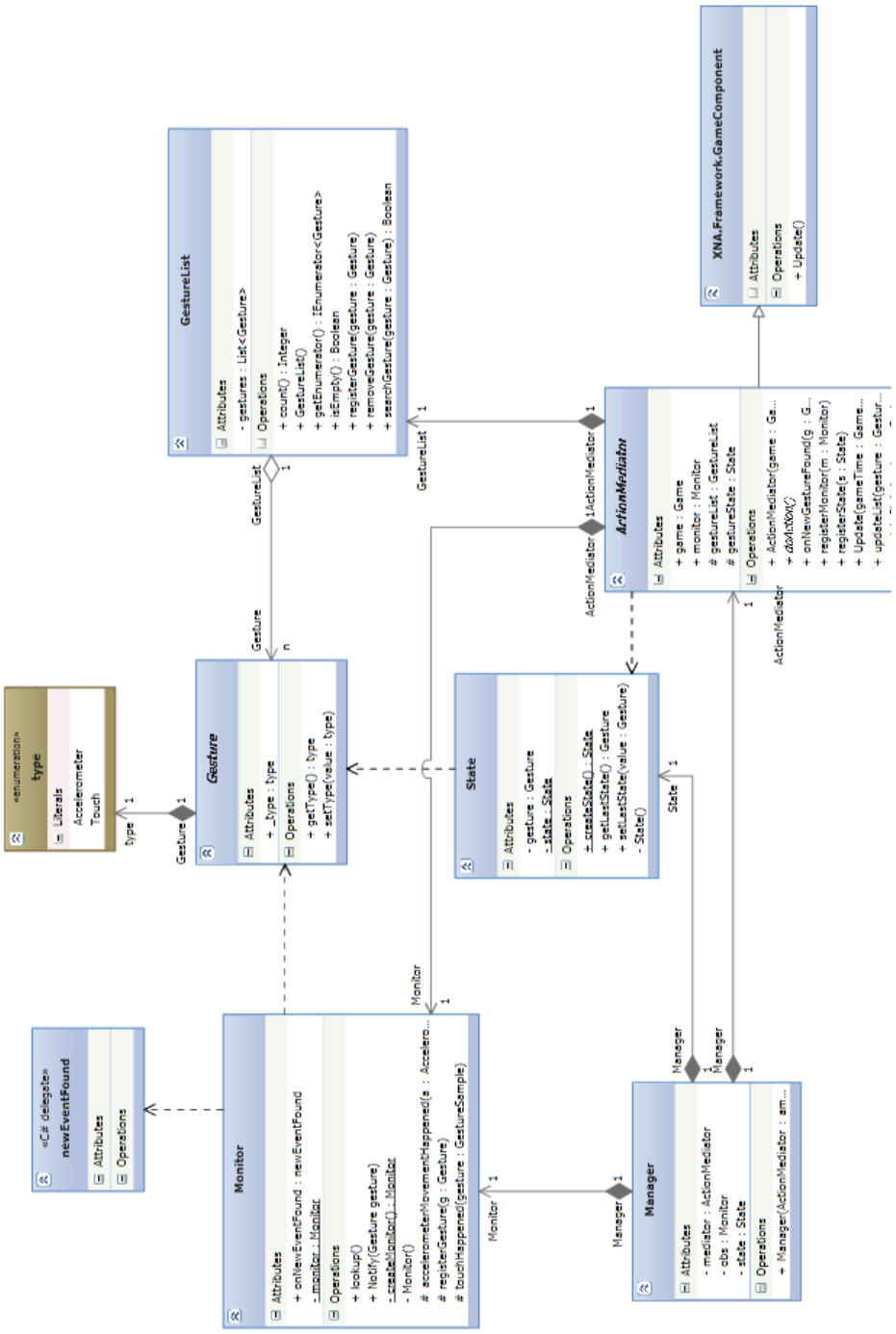

Figura 5 – Diagrama de Classes

.

4.3. Diagramas de Sequência

Nesta seção, são apresentados os diagramas de sequência do *framework* proposto. O primeiro diagrama, apresentado a seguir, mostra a sequência na qual os componentes do *framework* são instanciados a partir do início para a monitoração de gestos que serão capturados pelo *framework*.

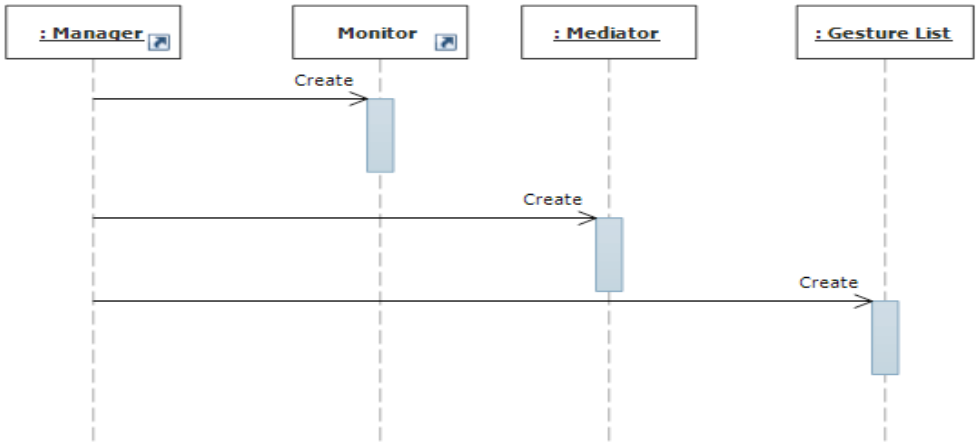

Figura 6 – Diagrama de Sequência I

O Manager é o primeiro componente a ser instanciado, tendo como única função instanciar os demais componentes para o funcionamento do *framework*. Logo após ser criado, o Manager cria o Monitor que será responsável por receber as interrupções geradas pelas interfaces do dispositivo. Em seguida, cria o Mediator que é responsável por receber os gestos do Monitor e fazer a chamada das ações que serão relacionadas a cada gesto. O Mediator também é responsável por atualizar a lista de gestos e o estado do *framework*. Por fim, o Manager cria a GestureList que é a estrutura que irá armazenar os gestos capturados pelo Monitor e registrados pelo Mediator.

O próximo diagrama mostra a sequência que o *framework* segue após ser inicializado pelo Manager para fazer o tratamento do gesto a partir de sua leitura.

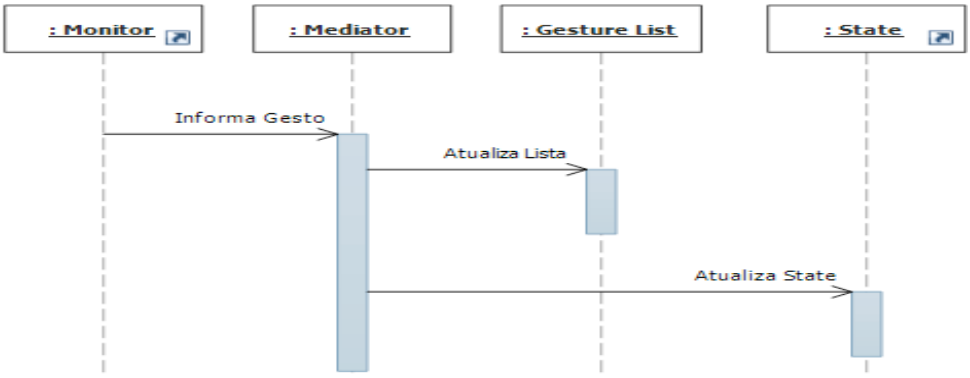

Figura 7 – Diagrama de Sequência II

A técnica proposta para o desenvolvimento do *framework* para fazer a detecção dos gestos foi a técnica de *Polling*. A técnica de *Polling* consiste em fazer com que o próprio sistema verifique regularmente cada dispositivo para obter a informação necessária para processamento [Tanenbaum & Woodhull, 2000]. Porém como a proposta inicial do *framework* é monitorar somente dois recursos do dispositivo (tela *multitouch* e acelerômetro) e devido à facilidade de implementação, a técnica de *Polling* foi escolhida.

Após receber o gesto monitorado pelo sistema, o Monitor informa o gesto ao Mediator. Este por sua vez faz a chamada das ações que serão relacionadas a cada gesto, conforme programado pelo usuário que faz a extensão do *framework*. O Monitor também é responsável por realizar a inserção do gesto na lista de gestos e fazer a atualização do estado do *framework*.

5. Implementação

A implementação foi baseada no XNA Framework da Microsoft. O *framework* desenvolvido foi criado como uma *library* para XNA. Dessa maneira ele poderá ser importado em qualquer projeto do para a plataforma Windows Phone 7 utilizando XNA Framework.

O ambiente de desenvolvimento utilizado para a produção do *framework* foi o Visual Studio 2010, juntamente com o Windows Phone SDK e o XNA Game Studio 4.0.

5.1. Generalização dos gestos

A primeira estrutura criada foi a classe `Gesture`, que generaliza a manipulação de qualquer interação com o dispositivo. Essa classe se utiliza de uma enumeração para determinar qual o tipo de origem de sua criação, permitindo a recuperação posterior de sua estrutura original.

Para o desenvolvimento desse trabalho, foram abordados as entrada `Touch` e `Accelerometer`, ambos fornecidos nativamente pelo XNA Framework. A escolha desses dois elementos se deve a serem comumente usados em aplicações para dispositivos móveis, principalmente para jogos.

```
public abstract class Gesture
{
    private type _type;

    public type Type
    {
        get { return _type; }
        set { _type = value; }
    }
}
```

A classe abstrata `Gesture` permite que diversos mecanismos de entrada sejam tratados de maneira genérica. Para isso, é necessário apenas criar uma nova classe que estenda a classe `Gesture` e que dê suporte a essa nova entrada.

O método público `Type` é responsável por informar o tipo de gesto encapsulado pelo objeto e será utilizado principalmente para o posterior tratamento desse gesto pela classe `ActionMediator`.

5.2. Detectando os gestos

O primeiro passo para a identificação do evento é detectá-lo. Os dispositivos móveis com Windows Phone 7 possuem diversas interfaces de interação, como tela sensível a toque, acelerômetro, bússola, etc.

A classe `Monitor` dispõe de um mecanismo que dá ao utilizador do *framework* a possibilidade de utilizar qualquer tipo de entrada de dados, delegando a responsabilidade de manipulação da lista ao *framework*.

```
public class Monitor
{
public delegate void newEventFound(Object sender, Gesture g);
    public newEventFound onNewEventFound;

    protected void registerGesture(Gesture g)
    {
        if (onNewEventFound != null)
        {
            onNewEventFound(this, g);
        }
    }

    public void lookup()
    {
        //… implementação da lógica de detecção dos gestos
    }
   //…
}
```

A construção da classe possui um atributo do tipo *delegate*, o `newEventFound`, responsável por determinar a assinatura do método que poderá ser disparado pelo acontecimento de um gesto. Esse atributo *delegate* será configurado pela instância da classe `ActionMediator`, que adiciona um método

próprio ao atributo, dizendo que seu método `onNewGestureFound()` será responsável pelo tratamento do gesto encontrado.

Para a detecção de um gesto, o mecanismo proposto utiliza o método `lookup()`. Esse método possui a lógica de detecção das entradas de usuário, que devem ser analisadas e, se assim decidir, enviadas para tratamento do *framework*.

O método `registerGesture` é chamado quando uma entrada de usuário foi detectada. Sua responsabilidade é repassar a chamada ao método configurado pelo atributo *delegate* da classe `Monitor`. Dessa forma, o tratamento desse gesto é delegado ao `ActionMediator`.

5.3. Manipulando os gestos

Para tratar uma grande quantidade de elementos do tipo `Gesture`, foi criada a classe `GestureList`, que contém mecanismos que manipulam uma lista de gestos sequenciais.

```
public class GestureList
{
    private List<Gesture> gestures;

    public GestureList()
    {
        gestures = new List<Gesture>();
    }

    public IEnumerator<Gesture> GetEnumerator()
    {
        foreach (Gesture g in this.gestures)
        {
            yield return g;
        }
    }
}
```

A implementação foi feita utilizando um objeto do tipo `List` para manter todos os gestos encontrados. Foram definidos então, os métodos para a manipulação dessa lista, responsáveis por inserir, remover e contar elementos contidos por ela. Há disponível, também, o método `GetEnumerator`, que foi sobrescrito para que a classe pudesse ser iterada de maneira simplificada pela aplicação.

A estrutura responsável por manipular os gestos armazenados é a classe `ActionMediator`. Trata-se de uma classe abstrata que deve ser reescrita pelo utilizador do *framework* e que herda as características da classe `GameComponent` do XNA Framework, podendo por esse motivo ter sua execução atrelada a do jogo.

```
public abstract class ActionMediator : GameComponent
{
    //… declarações de atributos
    public ActionMediator(Game game) : base(game)
    {
        //… inicializações
    }
    public abstract void doAction();
    //…
}
```

A implementação da classe `ActionMediator`, que estende a classe `GameComponent`, obriga a passagem de um objeto do tipo `Game` em sua instanciação, guardando essa referência em um atributo interno. Isso permite que, quando a classe for estendida e o método `doAction` for reescrito, o usuário terá acesso aos recursos do objeto da classe `Game` que foi passado. Isso é de vital importância para que a interação desejada possa ser repassada aos elementos do jogo.

5.4. Utilizando o Framework

O uso do *framework* é feito por meio do controle realizado pela classe `Manager`. A classe `Manager` tem seu construtor que solicita instâncias de `ActionMediator` e `Monitor` para ser inicializada. Isso se deve ai fato de que tais classes podem (e no caso do `ActionMediator` deve) ser estendidas para se agregar funcionalidade. Sendo assim, o construtor da classe carrega as dependências e faz os ajustes para ligá-las. Isso é feito por meio dos registros de `Monitor` e `State` pela instância da classe `Mediator`. A partir de então, os recursos estarão disponíveis para o funcionamento do *framework*.

```
public class Manager
{
    //… declarações de atributos
    public Manager(ActionMediator am, Monitor m)
    {
        //… inicializações das dependências
    }
}
```

A instanciação de um objeto `Manager`, portanto, garante o acesso aos recursos fornecidos pelo *framework*. Tendo as dependências criadas, o componente do jogo representado pela instância do `ActionMediator` pode ser adicionado como componente do jogo:

```
//… dentro de um jogo
manager = new Manager(this.actionMediator,
Monitor.createMonitor);
Components.Add(actionMediator);
```

6. Testes

O *framework* desenvolvido foi aplicado ao sistema InputToyWP7 disponibilizado no App Hub [App Hub, n.d.], que faz uso de entradas de toque e acelerômetro padrão. Para exemplificar o uso do *framework*, o sistema de entrada padrão foi substituído utilizando *framework* desenvolvido. Este teste foi submetido em um emulador do sistema do Windows Phone 7, desenvolvido pela Microsoft, com o fim de exibir os aplicativos sem a necessidade de fazer o *deployment* no dispositivo. Como resultado obteve-se: o sistema funcionou da mesma maneira, sem nenhuma deficiência aparente, mas também sem nenhum ganho de desempenho evidente.

A aplicação consiste em uma tela com fundo preto que ocupa todo espaço disponível na tela do dispositivo. Dois botões, ambos acionados por toque, são disponibilizados para o acesso a tela de ajuda e para pausar o a animação. A aplicação espera a interação do usuário por meio de um ou mais toque em qualquer posição de sua tela. No momento que ocorre um toque, na posição onde este foi detectado, deve aparece uma imagem representando uma fagulha (*sparkle*):

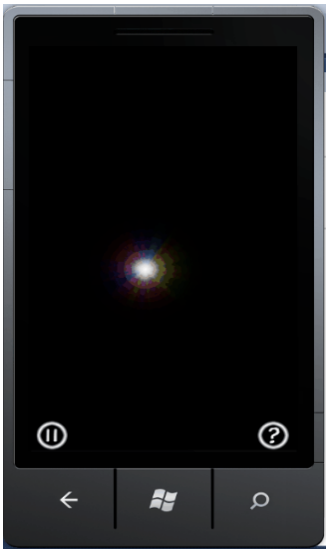

Figura 8 – Imagem da fagulha causada pelo toque

Tendo a referência ao *framework* disponível, a modificação da lógica original pôde ser realizada. A aplicação original é implementada sob uma extensão da classe `Game`, nativa do XNA.

Primeiramente, como é requisito para o uso do *framework*, é necessário a criação de uma classe que estenda a classe `ActionMediator`, que implementará a lógica da aplicação.

```
Class MyActionMediator : ActionMediator
{

    public MyActionMediator(Game game) :base(game) { }

    public override void doAction()
    {
        Game1 game1 = (Game1)this.game;

        //… implementação da lógica de manipulação dos gestos
    }
}
```

A classe `MyActionMediator` estende a classe `ActionMediator`, descrita no *framework*, e é composta por um construtor, que somente chama o construtor da classe pai, e a implementação do método `doAction`.

O método `doAction` recebeu toda a lógica de manipulação de gestos implementada pelo jogo original. No método foram feitos os tratamentos de *multitouch*, tanto para a inclusão de novas imagens de fagulha quanto em relação à interação com os botões do menu. No método `doAction`, os gestos que não foram utilizados pelo método descrito anteriormente, devem ser manipulados para a geração das fagulhas.

Para isso, é criada uma nova instância da classe `Sparkle`, original da aplicação, passando-se como parâmetro de inicialização as posições do toque, nos eixos X e Y, e o tempo atual do jogo, usado para a animação da fagulha. O tratamento de cada gesto encontrado na lista é iniciado pela verificação da posição do gesto. Isso é necessário para que se encontre gestos localizados sobre os botões da aplicação (*pause* e *help*). Os demais gestos devem ser manipulados para a geração das fagulhas.

Para isso, é criada uma nova instância da classe `Sparkle`, original da aplicação, passando-se como parâmetro de inicialização as posições do toque, nos eixos X e Y, e o tempo atual do jogo, usado para a animação da fagulha.

Após a implementação da extensão da classe `ActionMediator`, o *framework* já pode ser usado. Para isso, devem ser criadas as variáveis necessárias para seu funcionamento:

```
Manager manager;
ActionMediator actionMediator;
```

E depois, instanciá-las:

```
this.actionMediator = new MyActionMediator( this );
this.manager = new Manager(this.actionMediator,
Monitor.createMonitor);
Components.Add(this.actionMediator);
```

Nesse ponto é importante notar a criação da instância da classe `ActionMediator`, que recebe a referência `this` como parâmetro. Nesse contexto, a referência `this` representa o próprio jogo e pode ser passada como parâmetro para o `ActionMediator`, que é um `GameComponent`.

Por fim, para que o componente (`ActionMediator`) seja incluído no fluxo de execução do jogo, é necessário adicioná-lo à lista de componentes desse jogo. Como o `ActionMediator` é um `GameComponent`, o XNA Framework permite que o adicione ao atributo `Components`, automatizando esse processo.

Ao final da implementação e dos testes, podemos observar que o *framework* é capaz de detectar, tratar e manipular gestos de forma automática, permitindo ao usuário reescrever a logica de manipulação dos gestos, e cumpriu o requisito de tratamento de gestos sequenciais, embora não seja capaz de dar suporte a dispositivos como câmera, bússola, microfone, pois estavam fora do escopo do trabalho devido às restrições de tempo para o desenvolvimento.

7. Conclusões e Trabalhos Futuros

Interface Gestual é a interface do computador que permite ao usuário interagir com computadores por meio de gestos. Dentro da área de Interação Humano-Computador é um dos campos de pesquisa mais promissores, buscando novas formas de interação mais intuitivas entre o usuário e o computador.

No mercado de jogos eletrônicos, um requisito fundamental para o sucesso de um jogo é a técnica de controle aplicada. Dessa forma, em plataformas não convencionais de jogos (que não se utilizam de um *joystick*) tais como aparelhos celulares e *tablets*, uma boa Interface Gestual deve ser aplicada para se obter uma boa jogabilidade.

Este trabalho se propôs a apresentar um *framework* que disponibilize ao usuário que procura desenvolver jogos para a plataforma Windows Phone 7 um conjunto de ferramentas que automatize a detecção, o tratamento e a manipulação de um gesto individual ou de uma sequência de gestos durante o jogo. Dessa forma é possível abstrair a lógica de programação de detecção, tratamento e manipulação de gestos da lógica do próprio jogo.

O XNA Gesture Interface Framework, proposto por este trabalho, disponibiliza um conjunto de ferramentas que podem ser adicionadas ao projeto do jogo em forma de uma biblioteca que automatiza a detecção, o tratamento e a manipulação de gestos durante o jogo, um recurso que não era disponível pelo XNA

Framework de maneira centralizada e genérica. Além disso, o XNA Gesture Interface Framework é extensível, permitindo que o usuário adicione novas funcionalidades para novos recursos agregados a plataforma Windows Phone 7 sem a necessidade de se criar um nova versão do *framework*.

O XNA Gesture Interface Framework desenvolvido disponibilizou suporte somente a dois dispositivos de entrada. A plataforma Windows Phone 7 por sua vez, fornece outros recursos nativos, como câmera, microfone, bússola e pêndulo, que podem ser utilizados para detectar gestos. Caso o *framework* seja adaptado aumentando o número de dispositivos suportados, como os citados anteriormente, a aplicação pode sofrer problemas de desempenho, uma vez que o *framework* precisará monitorar vários dispositivos de entrada constantemente, ou seja, a cada *loop* de processamento do jogo.

Uma possibilidade de trabalho futuro é aumentar o número de dispositivos de entrada suportados pelo XNA Gesture Interface Framework. O sistema atual fornece suporte somente a dois recursos (o *touchscreen* e o acelerômetro), porém a plataforma Windows Phone 7 possui os demais recursos nativos que poderiam vir a ser suportados pelo *framework*.

Outra possibilidade de trabalho futuro é a adaptação do XNA Gesture Interface Framework para um sistema baseado em interrupções geradas pelos dispositivos. Essa técnica consiste em deixar que o dispositivo dispare um sinal de interrupção no sistema cada vez em que este necessite de um serviço. Dessa forma o sistema interrompe o processamento corrente e atende a interrupção gerada pelo dispositivo. Essa adaptação forneceria uma melhora de desempenho ao *framework* em relação ao sistema atual baseado na técnica de *Polling*, uma vez que o sistema deixaria de monitorar os dispositivos constantemente, obtendo ganhos em tempo de processamento.

Por fim, mais uma possibilidade de trabalho futuro com base no XNA Gesture Interface Framework apresentado por este trabalho é a mudança do próprio *framework* para outras plataformas, além da Windows Phone 7, que também se utilizem do XNA Framework, como PC e XBOX, dando suporte, por exemplo, ao Microsoft Kinect. A adaptação do *framework* para uma plataforma com mais recursos, como o Microsoft Kinect, possibilitaria agregar outras funcionalidades como captura de gestos por câmera infravermelho e vetor de microfones ao XNA Gesture Interface Framework.

Referências Bibliográficas

CHENG, H.; CHEN, A. M.; RAZDAN, A.; BULLER, E. (2011). “Contactless Gesture Recognition for Mobile”, In: MIAA 2011.

CORRADINI, A. e COHEN, P. (2002). “Multimodal Speech-Gesture Interface for Handfree Painting on a Virtual Paper Using Partial Recurrent Neural Networks as Gesture Recognizer”.

GAMELOFT. Brothers in Arms: Hour of Heroes - Official iPhone Game, Top Action Game for iPhone. Disponível em: < http://www.gameloft.com/iphone-games/brothers-in-arms-hour-of-heroes/>. Acesso em: 03 de fevereiro de 2012.

HUANG, X. e Y, Lin. (2008). "A vision-based hybrid method for facial expression", In: Ambi-Sys '08: Proceedings of the 1st International Conference on Ambient Media and Systems.

MICROSOFT. App Hub - Develop for Windows Phone & XBOX360. Disponível em: < http://create.msdn.com/en-US/ >. Acesso em: 02 de maio de 2012.

MICROSOFT. Rise of Glory. Disponível em: < http://www.windowsphone.com/pt-br/apps/c623f41b-af04-e011-9264-00237de2db9e >. Acesso em: 04 de fevereiro de 2012.

NICKEL, K. e STIEFELHAGEN, R. (2007). "Visual recognition of pointing gestures for human–robot interaction", In: Image and Vision Computing, v. 25, n. 12, 2007, p. 1875-1884.

TANENBAUM, A. e WOODHULL, A. Sistemas Operacionais: projeto e implementação. 2. ed. Porto Alegre: Editora Bookman, 2000.