Biblioteca de Funções para Utilização do Kinect em Jogos Eletrônicos e Aplicações NUI

Gabriel Schade Cardoso Ana Elisa Ferreira Schmidt*

Universidade do Vale do Itajaí (UNIVALI), Ciência da Computação, Brasil

Abstract

This paper presents a library that aims to facilitate developing UI for games based on natural language interactions like gestures, poses and voice commands. Microsoft Kinect is used to capture poses and voice commands that are stored in a predefined catalog. At game time, user's poses and voice commands are matched against the predefined catalog by the KinectScan DLL; when there is a match, events are fired by KinectScan. The publisher-subscriber design allows a loosely-coupled integration between game and library implementations. As a case of study, we propose integration between our library tools and the game engine Unity.

Keywords: Microsoft Kinect, Unity, NUI, Publisher-Subscriber, Integration

Authors' contact:
gabriel3656@gmail.com
*ana.schmidt@univali.br

Resumo

Este trabalho apresenta uma biblioteca de funções para facilitar o uso de interação em jogos por meio de interfaces que procurem utilizar a linguagem natural do jogador (como poses, gestos e reconhecimento de voz). O Microsoft Kinect é utilizado como hardware para captura de poses e comandos de voz que alimentam um catálogo de poses e comandos de voz pré-definidos. Em tempo de jogo, as poses e comandos de voz executados pelo jogador são comparados contra o catálogo pré-definido, sendo gerados eventos quando um comando é reconhecido. Um modelo de publicação baseado no padrão publisher-subscriber é implementado para uma integração fracamente acoplada entre um jogo e as ferramentas propostas. Como estudo de caso é apresentada a integração entre as ferramentas propostas e a engine de jogos Unity.

Palavras-chave: Microsoft Kinect, Unity, NUI, Publisher-Subscriber, Integração

Contato:
gabriel3656@gmail.com
gabriel.schade@hotmail.com
*ana.schmidt@univali.br

1. Introdução

Formas inovadoras de interação têm sido propostas por interfaces focadas no paradigma de interface natural. A NUI (Natural User Interface) visa utilizar uma linguagem natural para a interação entre o humano e o software através de gestos, poses e comandos de voz. A empresa Microsoft desenvolveu o hardware Kinect com o propósito inicial de trazer este tipo de interface para a área de jogos.

O desenvolvimento de jogos tornou-se menos complexo comparado a realidade de alguns anos atrás, devido as diversas ferramentas, frameworks, bibliotecas e engines que auxiliam os desenvolvedores e designers durante o projeto. A ferramenta Unity [Unity 2012] é um exemplo de ferramenta que visa facilitar e agilizar o processo de desenvolvimento de jogos digitais.

A interface é um componente fundamental no resultado da experiência de um usuário, principalmente nos jogos. A interação entre o jogador e o jogo pode alterar o resultado da experiência tanto positivamente quanto negativamente [Oliveira Netto 2004].

O Kinect foi desenvolvido a fim de oferecer uma interface mais flexível, simples e natural ao usuário, criando uma possibilidade de interação com os softwares e jogos do console Xbox 360. Em fevereiro de 2011 a Microsoft anunciou um software development kit (SDK) para o Kinect para a plataforma Windows, permitindo assim que os desenvolvedores pudessem utilizar o Kinect como interface para outros tipos de aplicações.

A integração de um jogo com o hardware do Kinect pode gerar uma experiência muito positiva para o usuário. Como proposta de integração, este trabalho apresenta uma biblioteca de funções para uso do Kinect em aplicações sem que seja necessário conhecimento aprofundado do hardware e de seu SDK.

O design desta biblioteca de integração segue o padrão publisher-subscriber, ou seja, toda a complexidade de comunicação com o Kinect está encapsulada em uma DLL-publisher, a KinectScan, que publica eventos quando comandos são reconhecidos. Existe ainda uma DLL-subscriber que consome os eventos de interesse do lado da aplicação. Desta forma a KinectScan não precisa conhecer a

aplicação final, oferecendo uma integração fracamente acoplada e extensível para qualquer tipo de aplicação, desde que seja criado um cliente que consuma e interprete os eventos disparados, no caso de estudo, a KinectUnity.

2. Kinect

O Microsoft Kinect foi idealizado pelo brasileiro Alex Kipman, inicialmente sob o codinome de projeto natal, fazendo referência à cidade brasileira Natal (RN) e pela palavra ser uma derivação em latim da palavra "nascimento". O dispositivo era utilizado como acessório do console Xbox 360, mas logo a empresa Microsoft disponibilizou sua versão para Windows.

O dispositivo possui componentes que o permitem captar movimentos e comandos de voz do usuário [Ashley and Webb 2012]. Todos os seus recursos podem ser manipulados via software através do Kinect SDK disponibilizada pela Microsoft [Msdn 2012].

2.1 Esqueleto do Usuário

O Kinect possui um processamento interno que utiliza um algoritmo de redes neurais artificiais que mapeia 20 articulações do usuário formando um "esqueleto". O esqueleto mapeado pelo dispositivo é composto das coordenadas tridimensionais, X, Y e Z, de cada articulação, permitindo a representação e manipulação 3D do mesmo.

3. Poses e Gestos

Os movimentos que compõe as expressões corporais das pessoas podem ser classificados em duas principais categorias: poses e gestos [Ashley and Webb 2012].

A diferença entre estes dois tipos de movimento é bastante clara e simples: **gesto** é um movimento do corpo em um espaço de tempo, como por exemplo, um aceno de despedida; já a **pose** é uma forma estática de manter o corpo parado. Apesar de não haver variação do corpo no tempo, é necessário um tempo de espera até que uma pose possa ser considerada ativa [Ashley and Webb 2012; Kean et al 2012]. No contexto deste trabalho, somente poses são armazenadas e posteriormente reconhecidas.

Neste trabalho, uma pose é considerada um conjunto de subposes, onde cada subpose é formada por três articulações. Toda pose deve possuir no mínimo uma subpose para que ela tenha significado; por exemplo, na posição T (manter os braços na horizontal alinhados com os ombros) deve ser verificado as articulações referentes aos dois braços e ombros, fazendo com que esta pose possua mais de uma subpose.

3.1 Detecção de Poses

Para a implementação do algoritmo detecção de poses, considera-se que duas poses são similares se os ângulos entre no mínimo três articulações que compõem a pose candidata são similares aos respectivos ângulos entre as articulações da pose pré-definida. Foram estudadas duas abordagens para o cálculo de ângulo entre articulações: através do **produto escalar** no espaço 3D e através da **lei dos cossenos** no espaço 2D.

No cálculo do ângulo pelo **produto escalar**, efetua-se o reconhecimento da posição relativa entre as articulações através dos ângulos formados entre os vetores de encontros das mesmas [Ashley and Webb 2012; Kean et al 2012; Kipman 2012].

As coordenadas das articulações escolhidas são empregadas para formar dois vetores (V e W) no espaço 3D; o ponto de encontro destes dois vetores formará o ângulo a ser calculado através da fórmula do produto escalar [Foley et al 1997] ilustrada na (Figura 1).

$$cos^{-1}(\frac{V.W}{||V||.||W||})$$

Figura 1: Fórmula do Produto Escalar

Para o cálculo do ângulo pela **lei dos cossenos** também são utilizadas as coordenadas das articulações envolvidas, no entanto os ângulos são calculados no plano 2D. Para tanto, uma etapa de projeção das coordenadas 3D para os planos frontal (XY), lateral (YZ) e superior (XZ) é necessária. Neste trabalho foi selecionada a projeção ortográfica paralela para implementar esta transformação [Foley et al 1997].

As articulações, agora representadas nos planos de projeção, são interligadas por retas formando três triângulos, um em cada plano de projeção. É escolhida uma das articulações como sendo a central, definindo qual dos três ângulos deve ser considerado para efeitos de comparação de poses.

Após ter-se identificado a articulação central, aplica-se a fórmula da lei dos cossenos, ilustrada na (Figura 2), onde a, b e c representam as retas que interligam as articulações e C representa o ângulo da articulação central. Note que o cálculo do ângulo será realizado uma vez para cada triângulo representado nos planos de projeção. Duas poses são consideradas similares quando os valores dos ângulos centrais da pose candidata for similar aos da pose pré-definida em cada um dos planos de projeção.

$$C = cos^{-1}(\frac{a+b-c}{2ab})$$

Figura 2: Fórmula da Lei dos Cossenos

4. UNITY

A Unity é um software que possui diversas ferramentas voltadas ao auxílio de desenvolvedores de jogos; desde programação, oferecendo APIs voltadas para isso; até a publicação, facilitando a publicação do jogo para plataformas diferentes.

Para os desenvolvedores, a engine Unity fornece um mecanismo de *Scripts* para criação da interatividade dos jogos; estes scripts podem ser codificados nas linguagens Javascript, C# e Boo [Unity 2012; Kean et al 2012]. Scripts são pequenos trechos de código escritos pelo desenvolvedor do jogo; através deles é possível criar toda a animação e interação entre os objetos do mundo virtual. Via scripts também é possível definir a interação entre os personagens do jogo e o jogador, através de funções que reconheçam diversos tipos de entrada.

Neste trabalho estas funções de entrada são sobrecarregadas para que a Unity também possa compreender entradas pelo dispositivo Kinect.

5. Biblioteca de funções

A biblioteca de funções que está sendo desenvolvida tem como principal objetivo facilitar o trabalho do desenvolvedor que deseja utilizar o dispositivo Kinect como forma de interface para sua aplicação. Para que isto seja possível esta biblioteca criará um nível mais alto de abstração em relação ao SDK oficial do dispositivo.

Esta biblioteca é formada por três elementos chaves: uma aplicação para registrar as formas de entradas (poses e comandos de voz); uma DLL que irá interagir com o SDK do dispositivo, a KinectScan; e uma DLL para a interação com a Unity, a KinectUnity.

A arquitetura desta biblioteca funciona no padrão publisher-subscriber, ou seja, a DLL KinectScan efetua a comunicação com o Kinect e ao reconhecer algum tipo de entrada válida, adiciona um evento em uma fila de eventos que poderá ser tratado por diferentes tipos de "clientes". Para estudo de caso, neste projeto o "cliente" é a DLL KinectUnity.

Juntamente com a biblioteca também é disponibilizado um conjunto de arquivos de entradas inicial. Todas as entradas que devem fazer parte da aplicação final que está sendo desenvolvida pelo usuário da biblioteca e que não constam no pacote inicial, devem ser registradas a partir da aplicação para captura de entradas.

No estado atual de desenvolvimento, já estão implementadas a aplicação de captura de entradas e a KinectScan, sendo que os testes de reconhecimento de poses ainda estão sendo avaliados. Estamos entrando em fase de implementação e testes da KinectUnity, que possibilitará a integração com um jogo já existente.

5.1 Aplicação para Captura de Entradas

Esta aplicação utiliza as câmeras e microfones do Kinect e algumas informações fornecidas pelo usuário para registrar arquivos XML contendo informações necessárias para que uma entrada possa ser reconhecida posteriormente. Existem dois tipos de arquivos XML para entradas customizáveis: um para registro de pose e outro para registro de comandos de voz.

A aplicação apresenta um formulário com a câmera do Kinect ligada para que o usuário consiga posicionar-se para registrar a pose; após posicionar-se basta o usuário dizer a frase "Record Pose" que a pose será registrada. A aplicação permite alterar e testar uma pose a partir de um arquivo XML de pose.

Esta aplicação também fornece os formulários para cadastro e teste de arquivos XML para grupos de comandos de voz.

5.2 Arquivo XML para Poses

Este arquivo contém informações necessárias para que a pose possa ser aberta pela aplicação e interpretada pela KinectScan em tempo de execução. Segue abaixo a lista das informações contidas no arquivo:

Tag Pose:

- Nome da pose – utilizado para identificação
- Tempo de espera – tempo até que a pose seja considerada ativa
- Conjunto de subposes – Tags do tipo Subpose

Tag SubPose:

- Ângulo – resultado do ângulo calculado
- Margem de erro – margem de erro a ser considerada do arquivo para o tempo de execução
- Articulações – Tags do tipo Articulação

Tag Articulação:

- X – coordenada X no espaço 3D
- Y – coordenada Y no espaço 3D
- Z – coordenada Z no espaço 3D
- Id – identificação da articulação

5.3 Arquivos XML para Comandos de Voz

Os comandos de voz são agrupados no arquivo de acordo com seu contexto, por exemplo: os comandos "Novo Jogo", "Continuar" e "Sair" devem estar no mesmo grupo pois pertencem ao mesmo contexto (menu inicial); já os comandos "Jump", "Run" devem estar em outro arquivo, pois estes comandos pertencem a outro contexto (jogo em andamento).

Segue a seguir a lista das informações contidas no arquivo:

- Nome do grupo – nome para identificação
- Nível de confiança – nível de confiança que será necessário atingir para o comando ser considerado correto.

- Palavras – comandos reconhecíveis.

5.4 KinectScan

Esta é a DLL que efetua a interação diretamente com o dispositivo Kinect através de seu SDK. A aplicação cliente criada pelo desenvolvedor deve possuir esta DLL incorporada no projeto, pois a KinectScan possui a responsabilidade de efetuar a varredura do usuário durante a execução da aplicação final que visa ter o Kinect como interface.

A KinectScan possui uma lista de eventos e métodos para que a aplicação cliente interaja com o sensor. Os eventos fornecidos pela DLL são: imagem da câmera RGB; informações do esqueleto do usuário; detecção do gesto "Swype"; detecção de poses; detecção de comandos de voz; alteração do estado do sensor e alteração do estado de um evento.

A DLL possui uma fila destes objetos de eventos. Esta fila é alocada em um espaço de memória compartilhado com a aplicação cliente. Cada vez que um novo evento é reconhecido, este é inserido na fila de eventos e a aplicação cliente é notificada que um novo evento foi adicionado na fila.

A KinectScan também possui algumas operações que iniciam, pausam e param os eventos e a conexão com o sensor. Os principais métodos fornecidos são:

- Initialize – Liga a conexão com o sensor.
- Start – Inicia um evento.
- Stop – Encerra um evento.
- Pause – Pausa um evento.
- Resume – Retoma um evento.
- Status – Retorna o estado de um evento.
- Terminate – encerra a conexão com o sensor.

Quando um evento é iniciado através do método "Start", a KinectScan inicializa os serviços do Kinect e começa a processar as informações necessárias para o reconhecimento da pose ou comando de voz associado a este evento; quando este for detectado o evento é enviado para a aplicação cliente.

No estado de pausa todos os serviços com o Kinect e processamento interno permanecem, porém o evento pausado não é mais adicionado na fila de eventos. Para interromper o processamento e desligar os serviços do sensor é necessário utilizar o método "Stop".

A KinectScan obtém apenas as informações necessárias para disparar os eventos assinados pela aplicação cliente. Para disparar eventos de poses e de comandos de voz, a DLL realiza o *matching* contra as entradas previamente gravadas pelo desenvolvedor.

Ao instanciar um objeto da KinectScan o próprio objeto cria uma thread separada da aplicação para efetuar todo o processamento do Kinect, bem como o *matching* de poses e comandos de voz. Desta forma a aplicação final não interrompe seu processamento enquanto os eventos são reconhecidos.

5.5 KinectUnity

A KinectUnity é a DLL que serve como estudo de caso para criação de um cliente para a KinectScan. Esta DLL faz a ligação entre um evento disparado pela KinectScan para um método que indique qual o tipo de entrada foi acionada através dos scripts C# da Unity.

Esta DLL possui códigos que interajem diretamente com as classes que manipulam a entrada de dados nos projetos da Unity, sobrecarregando uma classe chamada Input.

Propõem-se selecionar o projeto de um jogo já desenvolvido na engine Unity para efetuar a integração com a interface do Kinect utilizando a biblioteca que será criada. O projeto selecionado dará origem ao pacote inicial de poses e comandos de voz, além de servir de exemplo aos desenvolvedores.

Referências

ASHLEY, James; WEBB, Jarret, 2012. Beginning Kinect Programming with the Microsoft Kinect SDK 1. ed., Apress.

FOLEY, James D; et al., 1997. Computer Graphics: Principles and Practice 2 ed., Boston: Addison-Wesley.

KEAN, Sean; HALL, Jonathan; PERRY, Phoenix, 2012. Meet the Kinect 1. ed., Apress.

KIPMAN, Alex; MOORE, Richard; SHARP, Toby; COOK, Mat; et al. Real-Time Human Pose Recognition in Parts from Single Depth Images. Disponível em: http://research.microsoft.com/pubs/145347/BodyPartRecognition.pdf [Acessado em 26 de abril, 2012].

MSDN. Microsoft.Kinect. 2012. Disponível em: http://msdn.microsoft.com/en-us/library/hh855419.aspx [Acessado em 24 de abril, 2012].

OLIVEIRA NETTO, Alvin Antônio de Oliveira, 2004. IHC: Interação Humano Computador. Modelagem e Gerência de Interfaces com o Usuário 1. ed. Florianópolis: VisualBooks.

PREECE, Jenny; ROGERS, Yvonne; SHARP, Helen; BENYON, David; HOLLAND, Simon; CAREY, Tom, 1994. Human-Computer Interaction 1. ed. Nova Jersey: Addison-Wesley.

SANTOS, Rafael. "Desenvolvimento de games no Brasil Vale a Pena?". 2012. Disponível em: http://www.techtudo.com.br/platb/desenvolvimento/2011/06/11/desenvolvimento-de-games-no-brasil-vale-a-pena [Acessado em 15 de março, 2012].

SHNEIDERMAN, Ben, 1998. Designing the User Interface: Strategies for Effective Human-Computer Interaction 3. ed. Nova Jersey: Addison Wesley.