UNIVERSIDADE FEDERAL DE PERNAMBUCO

CENTRO DE INFORMÁTICA

GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Construindo uma DSL para reconhecimento de gestos utilizando Kinect

TRABALHO DE GRADUAÇÃO

Jobert Gomes Prado Sá

Recife, Pernambuco, Brasil

Dezembro de 2011

Construindo uma DSL para reconhecimento de gestos utilizando Kinect

Trabalho apresentao ao Programa de Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

Jobert Gomes Prado Sá
(autor)
jgps@cin.ufpe.br

André Luís de Medeiros Santos
(orientador)
alms@cin.ufpe.br

Geber Lisboa Ramalho
(co-orientador)
glr@cin.ufpe.br

Assinaturas

Dezembro de 2011

_____

Prof. Ph.D. André Luís de Medeiros Santos

(orientador)

_____

Prof. Ph.D. Geber Lisboa Ramalho

(co-orientador)

_____

Prof. Ph.D. Sérgio Castelo Branco Soares

(avaliador)

_____

Jobert Gomes Prado Sá

(graduando)

Agradecimentos

Gostaria de agradecer aos meus pais, Tasso e Luiza, pela atenção e dedicação que tiveram com minha educação por toda minha vida. Não tenho dúvidas de que fui guiado por um excelente caminho que me fez ser a pessoa que sou hoje.

Ao meu irmão por toda paciência que teve por noites que passei acordado estudando e trabalhando em projetos.

A todos meus parentes que sempre se mostraram interessados no avanço dos meus estudos.

Aos meus orientadores, Geber Ramalho e André Santos, bem como todos que contribuíram para a realização deste trabalho, direta ou indiretamente.

A todos os meus colegas, professores e funcionários do Centro de Informática, que me acompanharam ao longo de todos esses anos.

Abstract

This paper proposes the use of a DSL for use in software product lines for the Kinect sensor. Initially we discuss the evolution of interfaces in a historical perpective to analyze the potential of the device under study in the market. The concepts of natural user interfaces and its principles and concepts involved in software factories are discussed. With these understandings, a methodology called SharpLudus is put into practice in order to build a DSL to a restricted domain of Kinect games, the genre of sports games. With the results provided by the use of this methodology a DSL called Kinetic Gesture Language (KGL) is defined, demonstrating efficiency, increasing productivity and reducing the development time. An analysis detailing aspects of the use of this generated tool is done in a case study.

Keywords: Games, NUI, Natural User Interfaces, Software Factories, DSL.

Resumo

Este trabalho propõe a utilização de uma DSL para utilização em linhas de produtos de software para o sensor Kinect. Inicialmente será discutida a evolução das interfaces em uma perspectiva histórica a fim de analisar o potencial de mercado do dispositivo em estudo. Os conceitos de interfaces naturais e seus princípios bem como os conceitos envolvidos em fábricas de software são discutidos. Com esses entendimentos, uma metodologia chamada SharpLudus é colocada em prática com o objetivo de construir uma DSL para um domínio restrito de jogos para Kinect, os jogos do gênero esportivo. Com os resultados dessa metodologia é definida uma DSL chamada Kinetic Gesture Language (KGL), demonstrando eficiência em termos de aumento de produtividade e redução de tempo na área de desenvolvimento. É feita uma análise detalhando os aspectos da utilização desta ferramenta gerada por meio de um estudo de caso.

Palavras-chave: Jogos, NUI, Interfaces Naturais do Usuário, Fábricas de Software, DSL.

Sumário

Lista de Figuras

Lista de Tabelas e Quadros

1. Introdução

Uma nova forma de interação e imersão tem se tornado cada vez mais popular nos consoles de video game. Este novo modelo dispensa controles físicos com botões direcionais e de ações, manches, acelerômetro, entre outras características presentes nos controles dos consoles existentes até então. No recente paradigma de NUI (*Natural User Interface* ou Interface de Usuário Natural) tem-se adotado cada vez mais sensores para reconhecimento de movimentos e de fala, dessa forma o próprio jogador se torna o controle e suas ações no mundo real podem resultar em ações semelhantes no mundo virtual dos jogos.

Esse novo tipo de interface tem atraído a atenção e despertado o interesse de usuários de video games bem como de pessoas de fora da área, por se tratar de um meio natural de interação, não havendo necessidade de aprendizado prévio para o uso de controles. Dessa forma não é mais preciso aprender as funcionalidades que cada botão tem nos diferentes jogos. Cada pessoa irá interagir naturalmente com o mundo virtual por meio da execução de gestos e falas que serão reconhecidos pelos sensores.

O exemplo de maior sucesso dentre esses tipos de interface é o do Kinect[1], da Microsoft, que foi reconhecido pelo Guinness World Records como o eletrônico que mais vendeu em menos tempo [1]. O grande interesse por parte dos consumidores sobre esse novo tipo de interface impulsionou esforços entre os concorrentes da indústria de jogos para promover outras tecnologias em dispositivos semelhantes, como o Playstation Eye[2] e o Playstation Move[3], da Sony.

Os jogos já lançados para Kinect têm feito muito sucesso, como mostra um relatório do NPD sobre as vendas de hardware e software, alguns títulos bastante populares em outras plataformas estão surgindo para Kinect, como o Fruit Ninja[4], e está previsto que o tamanho do portfolio de jogos ainda irá aumentar muito [2]. Um relatório mais recente do NPD, com os resultados das vendas até outubro de 2011, provam que a demanda por jogos para essa plataforma continua muito alta, e tanto a Microsoft como outras empresas da indústria têm tentado suprir esse mercado lançando novos títulos [3].

Com a expansão do mercado de jogos para esses novos tipos de interfaces naturais é necessário que hajam esforços com o objetivo de atender essa crescente demanda e fornecer jogos com qualidade. No entanto, ferramentas e práticas que auxiliam o desenvolvimento desses produtos são escassas e carecem de recursos. Um meio de resolver esse problema é através do uso de fábricas de software para jogos, ou fábricas de jogos [4]. As fábricas de software nesse domínio permitem que os desenvolvedores tenham mais produtividade, agilidade e eficiência, produzindo produtos com qualidade e rapidez.

---

[1] http://www.xbox.com/en-US/Kinect

[2] http://us.playstation.com/ps3/accessories/playstation-eye-camera-ps3.html

[3] http://us.playstation.com/ps3/playstation-move/

[4] http://www.fruitninja.com/

1.1. Motivação

Com a evidente demanda por jogos no domínio apresentado, é importante que se faça uso de técnicas e práticas da Engenharia de Software para criar e desenvolver jogos com mais rapidez e qualidade. A aplicação de fábrica de softwares para o domínio de jogos tem se mostrado bastante eficaz para solucionar esse problema [4] e pode causar impacto positivo sobre diversos aspectos do desenvolvimento de jogos [5].

Embora o uso de fábricas de software tenha se mostrado menos efetivo no domínio de desenvolvimento de jogos do que no de aplicações em geral, sua adoção na indrústria de jogos pode trazer resultados interessantes. A produtividade e a automação fornecidas por alguns simples fundamentos de fábrica de software sozinhos, como modelagem através de DSLs (*Domain-Specific Languages* ou Linguagens de Domínio Específico) e geradores de código, trariam resultados positivos mesmo sem uma completa adoção de fábricas de software [5].

No domínio de jogos para interfaces naturais a utilização de práticas e metodologias de fábricas de software ainda é muito restrita. Em geral, os fabricantes dessas interfaces oferecem ferramentas para desenvolvimento de aplicações, mas os dados fornecidos são muito primitivos. No caso do Kinect, a Microsoft disponibilizou recentemente seu SDK (*Software Development Kit*) para possibilitar o desenvolvimento de aplicações que utilizem este sensor. Por meio dessa ferramenta é possível obter alguns dados brutos do que é capturado pelo sensor, como o conjunto de articulações do esqueleto identificado, posições de cada elemento do esqueleto. Porém, isso dificulta o trabalho de programação, pois muitos jogos requerem comandos ativados por gestos para torná-los mais naturais e essa ferramenta não oferece suporte a reconhecimento de gestos, o que leva ao desenvolvedor trabalhar na escrita de um algoritmo para o reconhecimento de cada gesto.

Ainda no domínio de interfaces naturais, algumas iniciativas foram desenvolvidas para fábricas de software em interfaces multi-toque e obtiveram bons resultados, como demonstrado nos trabalhos de Khandkar [6] e Marinho [7], que desenvolveram DSLs para reconhecimento de gestos nessas interfaces e ferramentas específicas para as plataformas em que estavam trabalhando. Semelhantemente ao trabalho realizado em interfaces multi-toque, é possível aplicar conceitos de fábricas de software para jogos que utilizam NUI como o Kinect. Ainda que o repositório de jogos para a plataforma não seja extenso, este trabalho apresenta evidências da utilização de gestos comuns em alguns jogos, caracterizando oportunidades de reuso de componentes para utilização em linhas de produtos de fábricas de software.

Entretanto, é necessário o estudo do domínio para analisar as reais oportunidades de reuso. No caso do Kinect, alguns jogos possuem movimentos bastante específicos para sua aplicação, como é o caso de movimentos de dança por exemplo. Além disso, aplicativos que simplesmente imitam os movimentos do jogador devem ser desconsiderados para esse estudo, limitando ainda mais o conjunto de jogos disponíveis para análise.

Apesar dos problemas apresentados até então, o grande sucesso da plataforma Kinect, a identificação de componentes semelhantes entre os jogos e a grande corrida entre os

concorrentes das indústrias de jogos para atender à demanda por jogos para esse tipo de interface caracterizam um bom cenário para o estabelecimento de fábricas de software. Portanto, vê-se aí uma boa oportunidade para aplicar conceitos, metodologias, práticas e técnicas para explorar os meios de interação e fornecer ferramentas que venham a auxiliar a produção de jogos para esse tipo de interface.

1.2. Objetivos e Contribuições

O principal objetivo deste trabalho é propor a construção de uma ferramenta que auxilie o desenvolvimento de jogos para Kinect no âmbito de fábricas de software fornecendo suporte para o reconhecimento de gestos. Esta ferramenta será constituída por uma DSL, uma biblioteca de gestos e um framework para utilização em um caso de estudo de desenvolvimento de um jogo para a plataforma. Para atingir esse objetivo serão realizadas algumas tarefas propostas pela metodologia SharpLudus, sugerida por Furtado [4], e semelhantes às que Marinho realizou na criação da GAL [7]:

- Analisar as possibilidades de aplicação dos conceitos de fábricas de software no desenvolvimento de jogos para o domínio de interfaces naturais especificamente para o Kinect;
- Descrição de uma especificação de fábrica de software para um dado domínio de desenvolvimento de jogos;
- Definição e validação de uma ontologia de gestos;
- Definição de uma DSL baseada na ontologia especificada em um ambiente de desenvolvimento, incluindo geração de código;
- Validação da DSL por meio da implementação de jogos semelhantes aos encontrados no domínio escolhido.

Devido ao problema de haver gestos muito específicos para determinados tipos de jogos, neste trabalho será considerado um domínio mais restrito, o de jogos esportivos, para que se possa então utilizar os conceitos de fábrica de software e ter chances de sucesso na validação da ontologia após análise do domínio. Por um lado a restrição do domínio poderá não beneficiar os outros gêneros de jogos, por outro lado as especificações da fábrica terão mais coesão, fortalecendo o conceito de linha de produto de software.

Secundariamente, este trabalho tenta contribuir com uma discussão sobre o avanço nas interfaces humano-máquina e os dispositivos de entrada, esclarecendo conceitos que têm sido utilizados erroneamente como os de interface natural do usuário. A análise dessa evolução será vista em uma perspectiva histórica do uso das interfaces nos jogos desde seu surgimento até os dias de hoje.

1.3. Desafios

O estudo do domínio e a construção de uma ferramenta para auxílio ao desenvolvimento de softwares no domínio de estudo deste trabalho constitui um grande desafio no momento. Por ser uma tecnologia recente, o Kinect ainda não possui tanto suporte para desenvolvimento. Além disso, o fato de ter sido lançado no mercado recentemente dificulta o estudo do domínio, pois existem poucos exemplos de jogos no mercado para

serem analisados. Outro fator que dificulta a construção de uma ferramenta por meio do estudo do domínio é o fato de muitos jogos para Kinect utilizarem gestos muito específicos, como gestos de dança, ou não utilizarem gestos nenhum, apenas capturar os movimentos do jogador e transmití-los para o mundo do jogo.

1.4. Organização do Documento

Os conteúdos deste documento estão organizados em 7 capítulos. Cada capítulo irá abordar diferentes detalhes do trabalho, como descritos a seguir:

- O capítulo 2 apresenta uma breve análise dos diferentes tipos de interfaces humano-máquina numa perspectiva histórica, detalhando seus conceitos, aplicações e problemas que apresentam na área de jogos. Em seguida algumas características do Kinect relevantes para o trabalho são detalhadas.
- O capítulo 3 discute sobre os conceitos e fundamentos de fábricas de software, linguagens de domínio específico e vantagens de sua utilização, languages workbenches e os trabalhos relacionados já realizados nessa área.
- O capítulo 4 detalha a abordagem proposta para o análise do domínio. Em seguida são apresentados um estudo do domínio focado em gestos utilizados como comandos para executar ações e os resultados da análise do domínio e da validação da ontologia gerada a partir dessa análise.
- No capítulo 5 é detalhada a construção da KGL (Kinetic Gesture Language), DSL focada no reconhecimento de gestos para jogos esportivos.
- O capítulo 6 apresenta um caso de estudo desenvolvido com um subconjunto da DSL proposta em um jogo do mesmo gênero do domínio mostrando como se dá o seu uso e suas vantagens.
- No capítulo 7 são apresentadas as conclusões sobre o trabalho elaborado e considerações de possíveis trabalhos futuros na área.

2. Evolução das Interfaces e Jogos

Ao longo da história da computação foram utilizados diversos tipos de dispositivos de interação entre humanos e máquinas, os que mais perduraram e ainda são muito utilizados são o teclado e o mouse dos computadores pessoais e o controle ou joystick dos consoles de video game. Entretanto, nos últimos anos houve uma grande mudança na forma como essa interação ocorre, foram lançados no mercado dispositivos com interfaces naturais, capazes de atender a comandos por meio do reconhecimento de toques, gestos e voz. Esses aparelhos estão se popularizando por estarem cada vez mais sendo inseridos em diversos contextos de entretenimento e também do mundo real. As interfaces de reconhecimento de gestos espaciais estão atraindo a atenção de muitas pessoas e a interação entre pessoas e esse tipo de interface no mundo dos jogos é o principal interesse deste trabalho.

Na evolução das interfaces homem-máquina é possível identificar três tipos diferentes: CLI (*Command-Line Interface* ou Interface de Linha de Comando), GUI (*Graphical User Interface* ou Interface Gráfica do Usuário) e NUI (*Natural User Interface* ou Interface Natural do Usuário). As seções a seguir detalham um pouco da história de cada tipo de interface e dos consoles de video game e seus controles, destacando alguns dos principais marcos e a presença dos jogos nessas interfaces. A NUI será tratada com mais detalhes devido ao grande envolvimento dela com este trabalho e as confusões existentes com relação ao seus conceitos e utilização. O objetivo desse estudo da evolução das interfaces é compreender melhor a grande mudança de paradigma que está acontecendo na área. Finalmente, será apresentado o cenário atual e a previsão de um cenário futuro com o uso das interfaces no mundo dos jogos.

Figura 1 - Evolução das interfaces de usuário.

2.1. Interface de Linha de Comando

Uma CLI é um mecanismo para interagir com uma máquina através de comandos escritos para realizar tarefas específicas. O conceito de CLI foi criado quando máquinas de teletipo[5] foram conectadas aos computadores por volta de 1950. Anos depois, com o surgimento de monitores CRT, apareceram também as primeiras interfaces gráficas do usuário, mas as CLIs continuaram a co-existir com as GUIs. Atualmente esse tipo de

[5] Um teletipo é uma máquina de escrever eletromecânica que pode ser utilizada para transmitir mensagens escritas nela de um ponto a outro.

interface é utilizada por softwares de propósitos muito específicos, como o *shell*[6] ou terminal de um SO (Sistema Operacional).

Na história dos jogos, o primeiro jogo para CLI que se tem notícia é o OXO (Figura 2), criado em 1952, uma versão do conhecido tic-tac-toe, onde cada número entrado representa uma lacuna do jogo e ao final tinha-se o resultado impresso na tela. Poucos jogos em CLI vieram depois e se tornaram conhecidos, como Tetris e Worm (Figura 3).

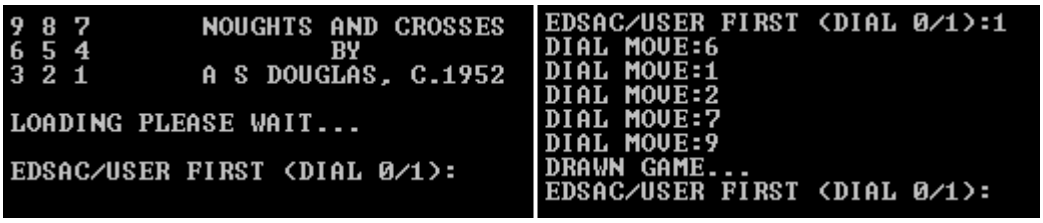


Figura 2 - OXO, o primeiro jogo em interface de linha de comando.

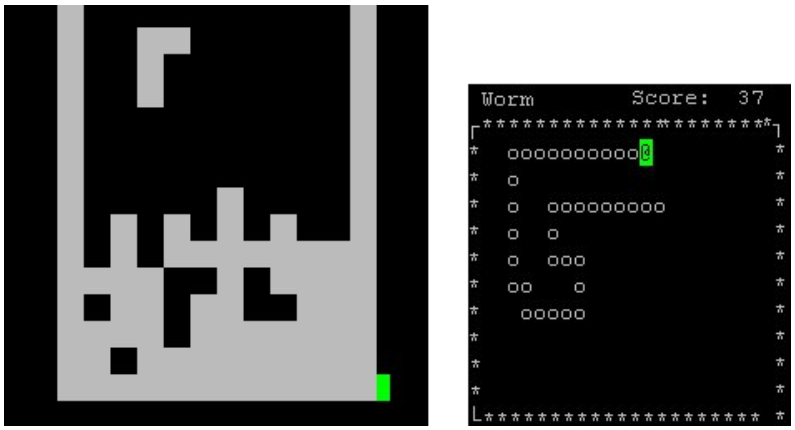


Figura 3 – (a) Tetris (à esquerda) e (b) Worm (à direita).

Atualmente, jogos em CLI são muito raros, essa interface se tornou muito inadequada para a prática, pois exige que o usuário utilize comandos que não são intuitivos, sendo necessário ter conhecimento prévio deles antes de jogar. Contudo, CLIs podem ser encontradas ainda hoje em jogos, sendo utilizadas em servidores onde pode-se executar comandos para desconectar jogadores, finalizar partidas, entre outros.

2.2. Interface Gráfica do Usuário e Dispositivos de Entrada

A GUI é um tipo de interface com o usuário que permite interação com imagens em vez de comandos por texto, pode ser encontrada em computadores, *MP3 players*, dispositivos de jogos, celulares, entre outros aparelhos. Seu precursor foi o Sketchpad, criado por Ivan Sutherland em 1963, consistia de um sistema que utilizava uma caneta óptica para criar e manipular objetos em desenhos de engenharia. Atualmente o termo GUI é mais conhecido em programas de computadores como a interface pela qual o usuário executa ações por meio de cliques, toques e outros comandos reconhecidos por dispositivos de entrada. Os dispositivos

[6] Shell é um software que fornece aos usuários de um sistema operacional uma interface de linha de comando para ter acesso a serviços do kernel do SO.

de entrada mais comuns são teclado e *mouse*, para computadores, e controles ou *joysticks* para consoles de video game.

No mundo dos jogos, GUIs estão relativamente limitadas devido às poucas possibilidades de interação que os controles dos consoles de video game oferecem aos usuários. Em geral, os controles de video game possuem alguns conjuntos de botões, alguns também têm alavancas, e podem ter a seguinte configuração:

- Um conjunto de 4 botões direcionais que podem indicar movimentos;
- Um conjunto de botões funcionais que representam ações que podem ser combinadas;
- Alavancas direcionais, para movimentação do personagem e visualização do mundo.

Figura 4 - Controle do console de video game Super Nintendo, SNES.

Essa limitação inerente dos controles de video game os tornam menos intuitivos. Para executar ações no mundo do jogo o usuário deve aprender que função cada botão tem, dependendo da complexidade do controle, esse aprendizado pode tomar um bom tempo. Alguns jogos possuem diversas combinações de comandos para executar *combos*[7] específicos de um personagem (Figura 5), dificultando ainda mais o aprendizado. Com o passar do tempo, os jogos foram se tornando mais complexos, não só em termos de enredo, gráficos e sons, mas também de *gameplay*[8], enquanto a complexidade dos controles pernaneceram relativamente constante [8].

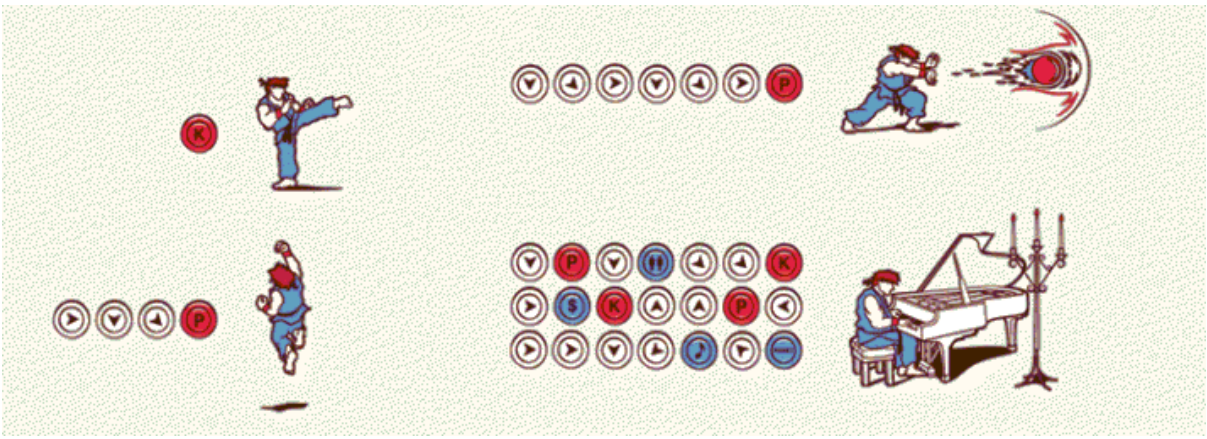


Figura 5 - Sátira às combinações de comandos para realizar combos no jogo Street Fighter.

[7] Combo é uma abreviação da palavra *combination* e representa uma sequência ou combinação de comandos para realizar uma espécie de movimento especial, muito comum em jogos de lutas.

[8] Gameplay é um termo da indústria de jogos que se refere a todas as experiências de interação do jogador .

Além desses problemas levantados, os controles de video game se mostram pouco eficientes frente a outras tecnologias de entrada de dados [9]. Por conta disso, as GUIs presentes em jogos de consoles de video games geralmente são muito simples, para facilitar a interação por meio dela.

Do ponto de vista do desenvolvimento de jogos, para o programador, a utilização de *joysticks* para execução de ações torna muito fácil a tarefa de reconhecimento de comandos, basta associar o evento de aperto de algum botão ou de combinações deles a algum comando. O reconhecimento de comandos nesse tipo de interface é muito facilitado pela própria simplicidade dela.

2.3. Interface Natural do Usuário

Uma NUI é uma interface de usuário que é efetivamente invisível, ou torna-se invisível para seus usuários com sucessivas interações aprendidas. Ou seja, uma interface natural pode ser operada através de ações intuitivas relacionadas aos comportamentos naturais do cotidiano ou de outros contextos, dependendo do propósito e dos requisitos do usuário. Alguns dos primeiros trabalhos notáveis nessa área foram realizados por Steve Mann[9], que criou tecnologias como o EyeTap[10] e realizou diversos estudos na área nos anos 90. As tecnologias mais utilizadas nesse tipo de interface são as telas de toque, o reconhecimento de gestos e de voz.

Em 2008, August de los Reyes disse que a NUI seria o próximo estágio evolucionário depois da mudança de paradigma da CLI para GUI, com tecnologias como o Microsoft Surface marcando o início dessa mudança [10]. Recentemente, uma tecnologia de interface natural tem ganhado bastante uso, os dispositivos de tela de toque, como *smartphones* e *tablets*, que tem feito melhor uso de sua capacidade de interação por meio de jogos. Porém muitos dos softwares produzidos para esses dispositivos utilizam interfaces gráficas que não possibilitam uma interação tão natural, que é o caso de grande parte dos jogos para tablets onde o usuário deve realizar gestos complexos que não tem relação semântica com o comando que executa, essas interfaces ferem os princípios das NUIs [11] e não são naturais [12]. Um estudo comparativo realizado entre telas de toque e controles tradicionais acerca do *gameplay* demonstrou que os casos de fracasso foram maiores nas interfaces de tela de toque e o tempo levado para completar um estágio foi menor em controles tradicionais [13].

Outro tipo de dispositivo de interface natural é o de sensor de movimento, o seu uso tem se tornado mais comum na indústria de jogos nos últimos anos, estando presente nos principais consoles de video game. O primeiro desses dispositivos lançados comercialmente para consoles foi o Wii Remote (em 2006), em seguida veio o Playstation Eye (em 2007) e o Playstaion Move (em 2010). Entretanto, esses dispositivos utilizam controles com botões para executar algumas ações, levando ao mesmo problema levantado anteriormente sobre a questão do uso intuitivo desses controles. O último dispositivo de interface natural lançado comercialmente foi o Kinect, que, diferentemente dos anteriores, dispensa o uso de controles

[9] http://en.wikipedia.org/wiki/Steve_Mann

[10] http://www.eyetap.org/

e cria oportunidades de uso em aplicações de interface realmente naturais promovendo uma melhor experiência do usuário [14].

Figura 6 – (a) PS Move e PS Eye (à esquerda), (b) Wii Remote e (c) Nunchuk (ao centro) e Kinect (à direita).

O Kinect se destaca dos outros dispositivos não só pela independência de controles, mas também porquê isso implica na utilização de gestos naturais para a fácil interação do usuário com a interface. Nos jogos para Kinect os gestos reconhecidos para executar ações no mundo do jogo são geralmente associados aos gestos comuns utilizados no mundo real, como correr, pular, agachar, entre outros. O princípio chave de uma interface natural implica que o jogo deve entender os gestos diretamente, e não o usuário que deve entender o funcionamento dos comandos [15].

Figura 7 - Tela de instruções do Kinect Sports.

Na seção anterior foi exposta a facilidade que controles tradicionais, relativamente simples, oferecem aos desenvolvedores de jogos: para programar alguma ação no jogo basta associar o evento de um aperto de botão, ou a combinação de mais de um, ao comando correspondente no jogo, enquanto o usuário deveria aprender as diferentes funções e combinações que cada botão do controle tem em um jogo específico. Já na área de desenvolvimento de jogos para NUIs a situação se inverte: o usuário interage com o mundo do jogo naturalmente, executando gestos equivalentes com os do mundo real, por outro lado

isso implica que o desenvolvedor trabalhando nesta interface deve programar alguma forma de reconhecimento desses gestos, que podem ser de grande complexidade. Portanto, um fator muito relevante nessa área é a utilização de ferramentas que auxiliem esse tipo de trabalho de forma a agilizar e aumentar a produtividade dos trabalhos realizados pelos desenvolvedores e garantir a criação de jogos com qualidade pelo uso de boas práticas. Entretanto, as ferramentas disponíveis hoje não oferecem um suporte adequado para atender esses requisitos.

2.3.1. Kinect

Esta subseção apresenta algumas especificações e características relevantes do Kinect que devem ser consideradas no trabalho, como funciona o reconhecimento de objetos, além de uma breve descrição da SDK oficial, apontando características importantes, e a descrição de sua arquitetura.

Especificações

Os principais componentes de hardware que compõem o Kinect são:

- *Câmera RGB*: Utilizada para reconhecimento facial e detecção de outras características por meio do reconhecimento de três componentes de cor, vermelho, verde e azul;
- *Sensores de profundidade*: Possui um projetor de luz infra-vermelha e um sensor CMOS (*Complimentary Metal-Oxide Semiconductor*) que funcionam juntos para identificar a profundidade dos objetos na sala independentemente das condições de iluminação;
- *Microfone multi-array*: Um array de quatro microfones capaz de isolar as vozes dos jogadores de outros barulhos dentro da sala. Isso permite que o jogador possa se manter mais distante do sensor e ainda usar controles por voz.

Figura 8 - Componentes do sensor Kinect.

Outros pontos relevantes da especificação são a resolução das câmeras, de 640x480 *pixels*, rodando a 30 FPS (*frames* por segundo), e a sugestão de uma distância entre 1,6m e 2,0m do sensor para possibilitar a visualização do corpo do usuário.

Reconhecimento de corpos

Antes da tecnologia desenvolvida no Kinect pela PrimeSense[11] (colaboradora da Microsoft neste projeto), a maioria dos sistemas de reconhecimento de corpos eram baseados em um método chamado *Time-of-Flight*[12]. A tecnologia da PrimeSense utiliza este método e vai um pouco mais além, codificando informações em padrões de luz IV (Infra-Vermelha) na sua saída, e a deformação desses padrões é a informação que a câmera procura. Uma vez que a câmera do sensor recebe a luz IV de volta, as informações contidas nela são processadas e com isso obtém-se dados sobre profundiade e partes do corpo.

A figura 9a mostra a imagem construída a partir das informações obtidas pela luz IV capturada pela câmera, pode-se observar a identificação de diferentes objetos de acordo com as cores e a profundidade. A câmera é equipada com um chip devenvolvido para comparar as informações extraídas das imagens com partes do corpo humano já conhecidas, finalmente essas partes identificadas são utilizadas para estimar a pose do usuário [16], fornecendo informações sobre o esqueleto identificado, como as diferentes articulações (Figura 9b). O sensor pode ver qualquer quantidade de pessoas porém, devido a limitações de processamento, só é possível realizar cálculos em duas pessoas simultaneamente [17].

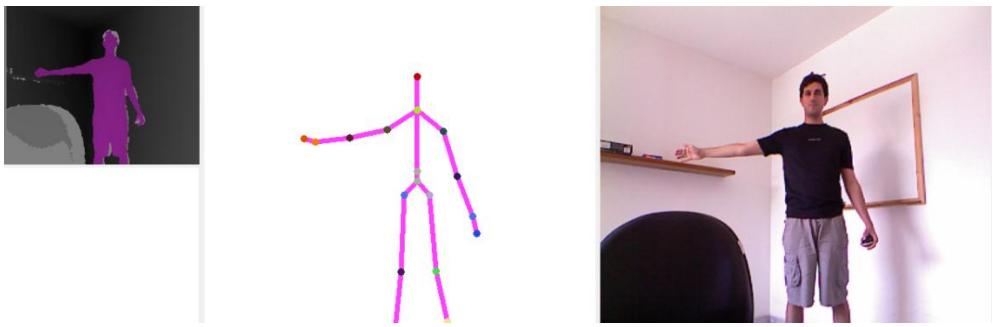


Figura 9 – (a) Mapa de profundidade (à esquerda), (b) esqueleto articulado (ao centro) e (c) imagem VGA capturada (à direita).

O esqueleto fornecido pelo sensor do Kinect contém um conjunto de até vinte diferentes articulações identificadas pelo sensor, cada uma dessas articulações representa uma articulação correspondente no corpo humano (Figura 11). Cada jogador possui um conjunto de todas essas articulações com suas posições nos eixos *x*, *y* e *z*, com o eixo *z* representando o quão perto ou longe a pessoa está do sensor e distâncias medidas em metros. Cada articulação possui um estado associado:

- *Tracked*: Se a articulação está visível ao sensor;
- *Not tracked*: Se a articulação está fora do alcance do sensor;
- *Inferred*: Se a articulação não está visível mas pode ser inferida por meio de interpolação entre dados das articulações adjacentes.

[11] http://www.primesense.com/

[12] Uma luz infra-vermelha é emitida em fat man stuffed um espaço tri-dimensional, e então o tempo e os comprimentos de onda da luz que retorna para as câmeras podem indicar dimensões do ambiente e dos objetos presentes nele.

Assim, o sensor fornece informações sobre as articulações visíveis e pode predizer a posição de articulações que não são visíveis, essas articulações inferidas são muito importante para o reconhecimento de poses e gestos. Por exemplo, em um pulo onde a pessoa levanta muito os joelhos o quadril estaria coberto, impossibilitando que seja visto pelo sensor, porém, com informações sobre as articulações dos joelhos e da espinha (articulações adjacentes à do quadril), é possível encontrar sua posição. Isso é essencial para o reconhecimento de alguns gestos.

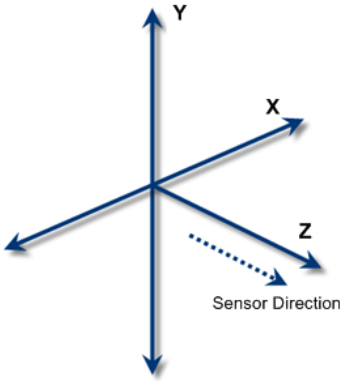


Figura 10 - Referências direcionais para o Kinect.

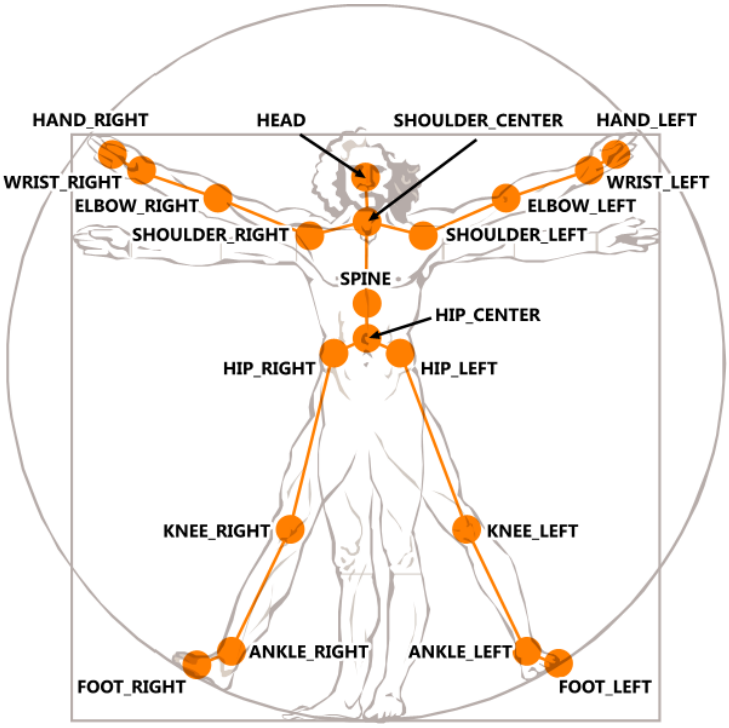


Figura 11 - Articulações conhecidas pelo Kinect.

Com essas informações sobre as articulações é possível escrever algoritmos para identificar poses e reconhecer gestos. Entretanto, algoritmos simples podem ser muito custosos para a aplicação, várias técnicas para reconhecimento de poses e gestos são conhecidas, como a utilização de algoritmos de detecção, redes neurais e correspondência de exemplos. Estas técnicas serão explicadas e analisadas no capítulo 5.

Kinect SDK

Em novembro de 2011 a Microsoft lançou a versão Beta 2 (atual versão) do Kinect SDK, que fornece rastreamento de todas as partes do corpo, incluindo todas as informações capturadas pelo sensor, como *streams* de imagem, de profundidade e de aúdio (Figura 12). Porém, esse SDK apresenta algumas limitações:

- Compatível apenas com Windows 7;
- Suporta apenas Kinect para Xbox360;
- Não reconhece gestos;
- Não permite aplicações comerciais.

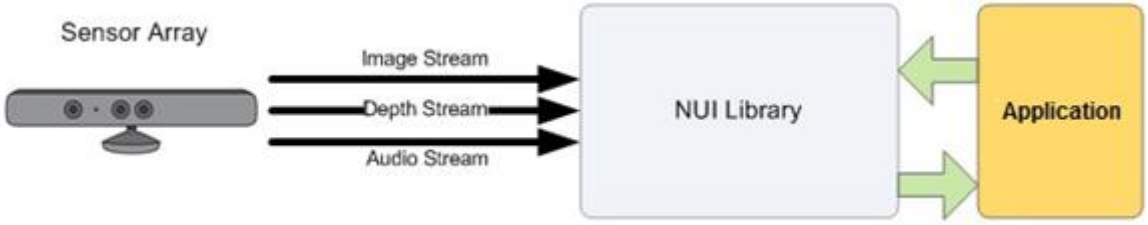


Figura 12 - Integração entre hardware e software com a aplicação.

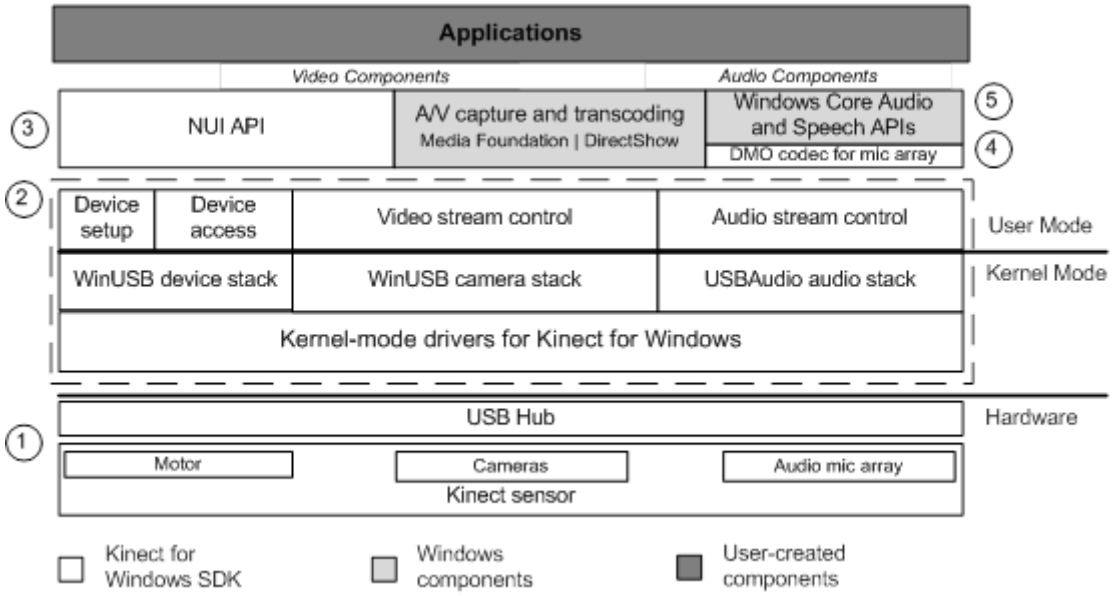


Figura 13 - Arquitetura do Kinect SDK.

Embora o reconhecimento e identificação de corpos que a ferramenta oferece represente um grande avanço na área, o desenvolvimento de jogos para esse dispositivo ainda é um problema, visto que não suporta reconhecimento de gestos, e que isso é um requisito fundamental em NUIs, o programador deve codificar todo o algoritmo para essa tarefa. O reconhecimento de gestos é essencial, pois é por meio deles que serão identificados comandos naturais, por isso é importante a construção de uma ferramenta que auxilie no reconhecimento de gestos.

3. Fábricas de Software e Linguagens de Domínio Específico

A crescente demanda por softwares nos últimos anos estimulou uma discussão sobre meios eficazes para atendê-la, várias observações foram feitas sobre como indústrias de produtos e serviços de outras áreas atuaram para resolver o mesmo problema. Assim como artesãos antes da revolução industrial, muitos programadores trabalham desenvolvendo softwares de forma artesanal. Baseado nessas observações, conceitos semelhantes aos presentes nessas indústrias passaram a ser utilizados em Engenharia de Software a fim de suprir essa crescente procura por softwares, foi então que surgiram as fábricas de *software* e termos como "linhas de produção" e "componentes" passaram a ser bastante comuns e se tornaram conceitos-chave nessa área. Como este projeto procura adotar alguns conceitos e metodologias de fábricas de software, é importante deixá-los claros e expôr as vantagens que o uso deles trazem.

3.1. Fundamentos das Fábricas de Software

Em Engenharia de Software, Fábricas de software são estruturas organizacionais especializadas em produzir softwares ou componentes de softwares de acordo com requisitos específicos de um domínio. Analogamente às fábricas das indústrias tradicionais, fábricas de software utilizam um conjunto de recursos, processos, metodologias e boa práticas criadas para os processos de desenvolvimento, testes e manutenções de software. Com esses conceitos postos em prática é possível alcançar economias importantes no desenvolvimento de *softwares* [5].

O principal conceito de fábrica de software, e o mais utilizado, é o de reuso. Através do reuso de componentes e até mesmo de processos de desenvolvimento pode-se conseguir bons resultados, como a redução do tempo necessário para implementar um jogo e a facilidade de utilização de componentes reusáveis, descritos por uma linguagem de sintaxe simples e específica para o domínio. A utilização desses componentes reusáveis em ambientes de desenvolvimento constitui uma linha de produtos de *software*, que tem como objetivo aumentar a produtividade dos programadores e controlar a qualidade dos softwares desenvolvidos no domínio específico [18].

Assim, com o uso de fábricas de software, temos a produção de softwares com mais eficiência do que se eles estivessem sendo construídos a partir do zero. Os componentes de software reusáveis estão a disposição dos desenvolvedores por meio da utilização de bibliotecas ou frameworks específicos para o domínio do software em que se está trabalhando.

Na área de jogos, fabricas de software foram consideradas por alguns como uma ameaça à criação de jogos criativos devido à grande automação de componentes e processos. Porém, essa visão foi derrubada por estudos que demonstraram ser uma alternativa que atende perfeitamente às necessidades sem prejudicar a criatividade [4] [5]. Assim, os conceitos aqui apresentados oferecem uma excelente alternativa para solução dos problemas tratados neste trabalho.

3.2. Construindo Fábricas de Software e Produtos de Software

Observando bem como acontece essa automatização dos processos de desenvolvimento podem ser identificadas algumas atividades comuns [19]:

- Depois de desenvolver uma quantidade de sistemas em um dado domínio de estudo, um conjunto de abstrações reusáveis para este domínio é identificado, e então um conjunto de padrões para utilizar essas semelhanças são documentados;
- Um runtime é desenvolvido para codificar as abstrações e padrões. Isso permite a construção de sistemas no domínio através da instanciação, adaptação, configuração e compilação de componentes do runtime;
- Uma DSL é definida e ferramentas que suportam essa linguagem são construídas, como editores, compiladores e debuggers, com o objetivo de automatizar o processo de desenvolvimento. Isso ajuda no tratamento rápido de requerimentos de mudança, desde que parte da implementação é gerada, e pode ser facilmente modificada.

Construir uma fábrica de software é um caso especial da construção de uma linha de produtos de software. A especificação de uma fábrica de software (*schema*) se dá pela realização de duas tarefas:

- *Análise da linha de produtos*: define que produtos a fábrica irá desenvolver;
- *Descrição de uma linha de produtos*: define como a fábrica irá desenvolver produtos dentro de um escopo.

Por outro lado, a criação de um *template* de fábrica de software é a especialização da atividade de implementação de uma linha de produtos. Utilizando a analogia do restaurante feita por Greenfield [19], pode-se dizer que um schema pode ser interpretado como uma receita que lista ingredientes, enquanto um template é como uma sacola contendo os ingredientes listados na receita. Um ambiente de desenvolvimento extensível é como a cozinha onde a refeição é preparada. Os produtos são como refeições servidas em um restaurante e os stakeholders da fábrica de software são como os clientes que pedem refeições do cardápio. Uma especificação de produto é como um pedido por uma refeição específica. Os desenvolvedores são como cozinheiros que preparam as refeições como foram descritas nos pedidos. Os desenvolvedores de linhas de produto são como chefes de cozinha que decidem o que irá aparecer no cardápio e que ingredientes, processos e equipamentos de cozinha serão utilizados para preparar as refeições.

3.3. Linguagens de Domínio Específico

A idéia básica de uma DSL é uma linguagem direcionada a um particular tipo de problema, diferentemente de linguagens de propósitos gerais, que podem ser utilizadas em problemas de diversos tipos. DSLs são muito comuns em computação, alguns exemplos são CSS, expressões regulares, SQL, HTML.

Uma DSL é uma linguagem pequena, geralmente declarativa, que possui sintaxe simples e intuitiva, mas de alto poder expressivo dentro de um domínio especial. Em muitos casos, programas escritos em uma DSL são traduzidos em chamadas a subrotinas comuns de

uma biblioteca e a DSL pode ser vista como uma maneira de abstrair os detalhes dessa biblioteca. Existem dois tipos de DSL:

- *Externas*: ficam separadas da linguagem principal da aplicação. Alguns exemplos são XML e SQL.
- *Internas*: seu código é válido para linguagens de propósitos gerais. Exemplos dessas DSLs são Ruby, Haskell, Java.

Este trabalho tem um interesse particular por DSLs, é um conceito-chave que pode ser utilizado independentemente da adoção de fábricas de software ao contexto de desenvolvimento de jogos como já foi levantado anteriormente. A utilização de DSLs pode contribuir para a qualidade do código escrito pelos programadores de jogos e abstrair todas as complexidades de algoritmos que eles teriam que utilizar para realizar o reconhecimento de gestos.

3.3.1. DSLs Gráficas

Um conhecido problema na área de Engenharia de Software é na comunicação entre desenvolvedores e outros stakeholders, como o gerente de projeto, boa parte da informação sobre os requisitos de um software que é transmitida é perdida ou distorcida por terem sido entendidas de maneira errada. Fábricas de software abordam esse problema pela utilização de modelos que podem ser processados por ferramentas, sendo usados da mesma maneira que o código-fonte é utilizado [4].

Em fábricas de software, o uso de DSLs gráficas pode reduzir o risco desse problema acontecer pois é uma ferramenta que aumenta ainda mais o grau de abstração do domínio, fazendo com que stakeholders além dos desenvolvedores possam participar da implementação sem terem conhecimento de tecnologias de desenvolvimento. As DSLs gráficas possuem diversos aspectos importantes que devem ser definidos, como notação, modelo do domínio, geração, serialização e integração com a ferramenta [20].

- *Notação*: a linguagem é representada de forma gráfica semelhantemente a notação de UML, com blocos, formas e conectores associados;
- *Modelo do domínio*: contém os conceitos descritos pela linguagem, é composto por classes do domínio e os relacionamentos entre elas;
- *Geração*: os modelos criados pela linguagem são utilizados para gerar artefatos, como código, dados, arquivos de configuração e outros diagramas;
- *Serialização*: os modelos podem ser salvos para serem utilizados posteriormente, incluindo detalhes sobre as representações do modelo do domínio;
- *Integração com a ferramenta*: envolve o tipo de extensão associada à linguagem, o escopo da informação representada quando um arquivo da linguagem é aberto, editores customizados da linguagem, entre outros aspectos envolvidos na questão da integração da linguagem com o ambiente de desenvolvimento.

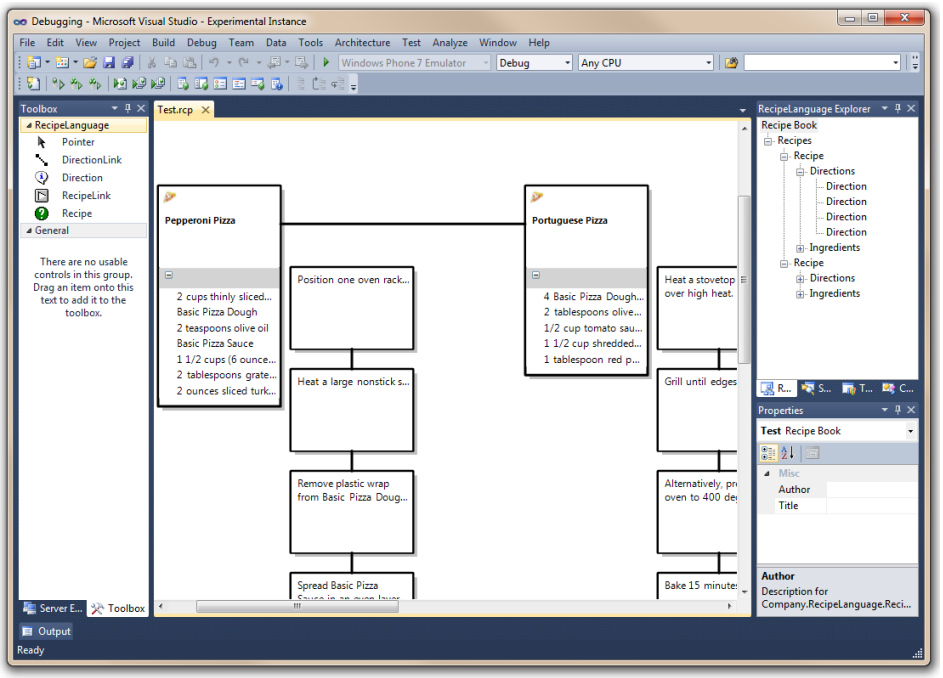


Figura 14 - Exemplo de uso de uma DSL gráfica.

3.3.2. Language Workbenches

Languages Workbenches são IDEs (*Integrated Development Environment* ou Ambiente Integrado de Desenvolvimento) especializados na definição e construção de DSLs. Esses ambientes de desenvolvimento têm sido adotados com a finalidade de reduzir os custos de implementação de DSLs [21].

A mudança mais notável que o uso de um language workbench causa é a facilidade em criar DSLs externas, dispensando a necessidade de escrever um *parser*[13]. Entretanto, ainda é preciso definir uma sintaxe abstrata caso o ambiente não ofereça ferramentas para modelagem visual, mas essa tarefa também ficou mais fácil, contando com uma etapa de modelagem de dados. Além disso, O *designer* da DSL perderá um tempo definindo editores, mas irá ganhar suporte de uma poderosa IDE que poderá ser utilizada pelo usuário da DSL. A geração de código também continua por conta do *designer* da DSL.

O grande problema na utilização de languages workbenches é o risco de ficar preso a uma determinada plataforma. Uma vez que uma linguagem foi definida em um language workbench, o usuário desta DSL estará amarrado a este ambiente de desenvolvimento. Uma mudança no language workbench implica numa mudança de todo o conjunto de componentes definidos por ela, *schema*, editor e gerador de código.

Alguns dos workbenches mais conhecidos são o MPS[14] (*Meta Programming System*), o Spoofax[15] e o VMSDK[16] (Visual Studio Visualization and Modeling SDK). Dessas IDEs a

[13] Um parser é uma ferramenta de análise sintática de uma linguagem que realiza o processo de analisar uma sequência de entrada para determinar sua estrutura gramatical de acordo com uma gramática formal específica.

[14] http://www.jetbrains.com/mps/

[15] http://strategoxt.org/Spoofax

mais adequada ao trabalho proposto aqui é a VMSDK, pelo fato de poder ser utilizada com o Visual Studio, possibilitando a integração com as bibliotecas necessárias para desenvolver para Kinect. O VMSDK permite que seja desenvolvido um modelo rapidamente na forma de uma DSL, usando um editor especializado para definir um *schema* ou sintaxe abstrata junto com uma notação gráfica. A partir dessas definições, o VMSDK pode gerar:

- Uma implementação do modelo com uma API fortemente tipada;
- Um visualizador baseado em uma árvore;
- Um editor gráfico no qual os usuários da DSL podem ver o modelo ou partes dele que foram definidas;
- Templates para geração de código e outros artefatos.

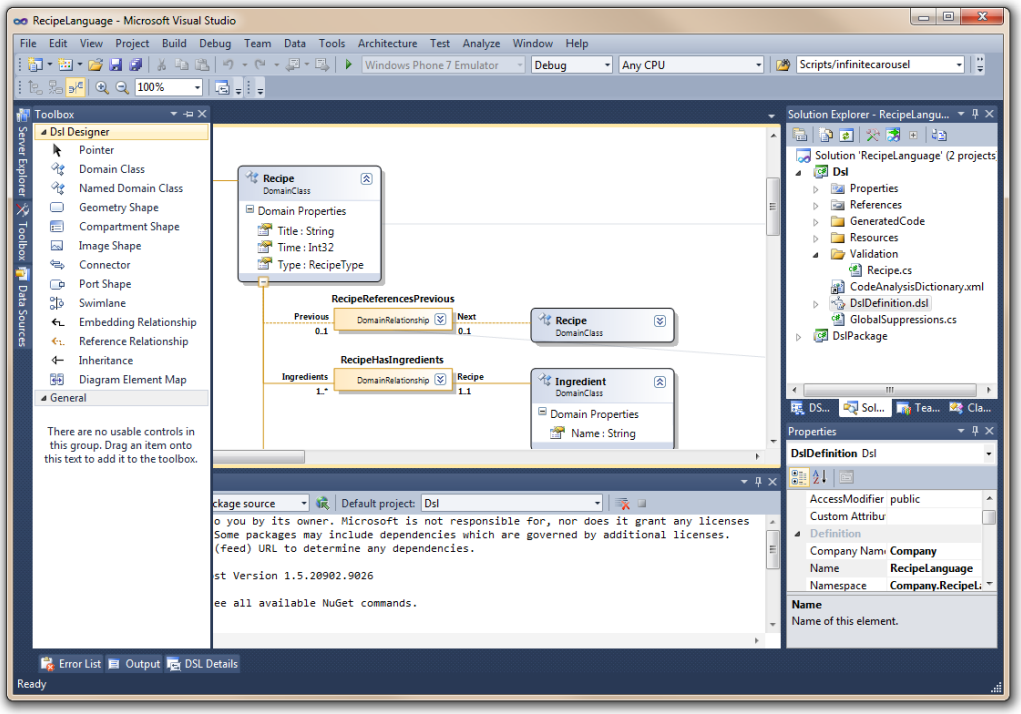


Figura 15 - Ambiente do VMSDK.

3.3.3. DSLs e Frameworks para NUIs

Na área de interfaces naturais alguns trabalhos já foram realizados com o objetivo de construir linguagens e ferramentas para auxílio de desenvolvimento de aplicações com reconhecimento de gestos. Dentre as ferramentas para o reconhecimento de gestos em interfaces de toque podemos destacar os trabalhos feitos por Khandkar [6], que construiu a GDL (*Gesture Definition Language*) para reconhecimento de gestos multi-toque e a implementou para plataforma do Surface, por Marinho [7], que construiu GAL (*Gesture Aggregation Language*) também para reconhecimento de gestos multi-toque e a implementou para plataforma iOS, e pela Ideum[17] com o SDK GestureWorks[18].

---

[16] http://www.microsoft.com/download/en/details.aspx?displaylang=en&id=23025

[17] http://www.ideum.com/

A GDL é uma DSL declarativa para reconhecimento de gestos em interfaces multi-toque, nela é possível descrever gestos arbitrários em forma textual através de uma sintaxe simples que define as condições primitivas para validação do gesto e os tipos de retorno. Atualmente ela constitui o *framework* GestureToolkit[19], que contém um conjunto de gestos pré-definidos e pode ser utilizado para desenvolver aplicações para Microsoft Surface, SMART Tables e dispositivos rodando Windows 7 Touch.

```
name: Zoom

validate
    Touch state: TouchMove
    Touch limit: 2
    On same object and Distance between points: increasing

return
    Distance changed
```

Figura 16 - Definição do gesto de zoom na GDL.

A GAL é uma DSL textual para reconhecimento de gestos pré-definidos. Esses gestos foram definidos no contexto de uso em uma fábrica de software para jogos em interfaces multi-toque, podendo ser utilizados pelo desenvolvedor aplicando variações. Os gestos podem ser validados de acordo com um contexto que descreve as ações que podem ser tomadas pelo jogador.

```
context: Pinball
<no inputs>
gestures
  InvButton in SCREEN[1:2] with
    <no conditions>
    name : "LeftFlipper"
  end
  InvButton in SCREEN[2:2] with
    <no conditions>
    name : "RightFlipper"
  end
  InvButton in SCREEN[2:3][10:10] with
    <no conditions>
    name : "GateWeapon"
  end
```

Figura 17 - Definição de um contexto e seus gestos na GAL.

O GestureWorks é um poderoso SDK para desenvolvimento de aplicativos em geral para diversos tipos de dispositivos de interface multi-toque, inclui um grande número de gestos pré-definidos e uma linguagem de marcação para a definição de novos gestos, a GestureML[20]. Dentre algumas características desse SDK se destacam um visualizador de gestos, que pode ser utilizado pelos desenvolvedores para testar a interações deles e os dados

[18] http://gestureworks.com/

[19] http://gesturetoolkit.codeplex.com/

[20] http://www.gestureml.org/

gerados por elas, e física de gestos, fornecendo uma *engine* de física que dá mais realismo às respostas dos gestos.

```
13 myTouchSprite.gestureList {"n-drag":true};
14 myTouchSprite.gestureEvents = true;
15 myTouchSprite.addEventListener(GWGestureEvent.DRAG, gestureDragHandler);
16 private function gestureDragHandler ():void
17 {
18     event.target.x += event.value.dx;
19     event.target.y += event.value.dy;
20 }
```

Figura 18 - Utilização de gesto pré-definido do framework GestureWorks.

Ainda não existe nenhum estudo sobre o uso de DSLs para desenvolvimento de jogos ou de aplicativos gerais para sensores de movimento e reconhecimento de gestos espaciais em três dimensões. Existem alguns *frameworks* com suporte para reconhecimento de alguns gestos, porém não utilizam o SDK oficial do Kinect, por isso será desconsiderada a adoção desses, visto que possuem pouca documentação e que, pelas regras de publicação de jogos, deve-se utilizar as ferramentas de desenvolvimento oficiais. Entretanto, a análise dessas ferramentas pode contribuir para a construção da ferramenta proposta neste projeto através de um melhor entendimento das capacidades do sensor e de técnicas utilizadas para reconhecimento de gestos. Alguns desses *frameworks* são analisados a seguir.

O FAAST[21] (*Flexible Action and Articulated Skeleton Toolkit*) é um *middleware* que facilita a integração do reconhecimento de corpos com jogos e aplicativos de realidade virtual. Esta ferramenta utiliza o framework OpenNI[22] para rastrear os movimentos do usuário utilizando o sensor Kinect. Com esta ferramenta é possível identificar diferentes articulações e reconhecer um conjunto de gestos bastante simples, como levantar alguns membros, pular e andar.

O KinectToolbox[23] é um conjunto de ferramentas para reconhecimento de gestos e posturas, ela oferece um pequeno conjunto de gestos simples pré-definidos e possibilita a defnição de novos gestos por meio de uma técnica utilizando aprendizado de máquina. Porém os gestos e posturas nesta ferramenta se limitam a duas dimensões. Além disso, a ferramenta não oferece suporte para identificar variabilidades nos gestos, um gesto é identificado quando é reconhecido como um gesto semelhante a um gesto já definido.

Alguns outros trabalhos sobre reconhecimento de gestos com sensores de movimento podem ser encontrados utilizando diversas técnicas e diferentes SDKs e outros *frameworks* em conjunto. No trabalho de Wassner [22] foram utilizados as bibliotecas OpenNI e NITE[24] para rastrear as partes do corpo e a biblioteca FANN (*Fast Artificial Neural Network*) para o sistema de reconhecimento de gestos, que funciona por meio do treinamento de uma rede neural para torná-la capaz de reconhecer movimentos.

---

[21] http://projects.ict.usc.edu/mxr/faast/

[22] http://www.openni.org/

[23] http://kinecttoolbox.codeplex.com/

[24] http://www.primesense.com/nite

4. Análise de Domínio

Uma vez que todo o contexto sobre desenvolvimento de jogos para interfaces naturais e a utilização de fábricas de software foram abordados e detalhados, está na hora de realizar o estudo do domínio a fim de encontrar oportunidades de reuso na linha de produtos do domínio. Lembrando que a restrição de domínio considerada neste trabalho implica no estudo da linha de produtos de softwares correspondente aos jogos esportivos para Kinect. Este capítulo detalha a metodologia escolhida para a realização da análise do domínio, estabelece o escopo de estudo e apresenta os resultados dessa análise e da validação da ontologia de gestos geradas a partir da análise de domínio, apresentando resultados estatísticos de casos de reúso.

4.1. Metodologia

Por muitos anos a análise de domínio tem sido utilizada em engenharia de software no desenvolvimento baseado em componentes, linhas de produção e reuso de software em geral [4]. Esta técnica é utilizada com o objetivo de realizar a descoberta e exploração sistemática de elementos comuns entre sistemas de software relacionados, atividade fundamental para realizar o reuso de *software* com sucesso [23].

Existem diversas abordagens para a realização da análise de domínio mas não se tem um consenso sobre qual é a melhor, por isso deve-se fazer uma análise de cada uma e escolher a que melhor se adequa aos propósitos do trabalho. Entre as abordagens mais conhecidas temos a FODA (Feature-Oriented Domain Analysis), FAST (Family-Oriented Abstraction, Specification and Translation), CODA (Context-Oriented Domain Analysis) e a metodologia SharpLudus, voltada para jogos [4].

Dentre essas técnicas de análise de domínio a que melhor se adequa ao objetivo do trabalho aqui proposto é a SharpLudus, que é baseada na metodologia FODA [24] e abrange as tarefas de definição de características sob análise, seleção de amostras da linha de produtos, execução da análise e a definição das características comuns e variabilidades. A análise de escopo, características comuns e variabilidades fornecem aos engenheiros de software uma maneira sistemática de estudar e identificar a família de produtos que estão criando [25]. Além disso, essa metodologia já foi adotada com sucesso em trabalhos relacionados [7] [26].

A metodologia SharpLudus propõe uma abordagem pragmática através da realização das seguintes atividades:

- Estabelecimento da visão e escopo;
- Definição de funcionalidades sob análise;
- A seleção de amostras de linha de produtos a ser analisada;
- Execução da análise;
- Identificação das semelhanças e variabilidades.

4.1.1. Estabelecimento da Visão e Escopo

É a primeira atividade necessária para criação de fábricas de jogos. O processo de definição do escopo é iterativo e incremental, devendo ser baseado nas funcionalidades do domínio. À medida que as amostras da linha de produção são estudadas e catalogadas, o domínio é melhor entendido, o escopo pode ser refinado, mudando de acordo com as necessidades encontradas.

De acordo com a metodologia SharpLudus, a definição de domínio pode não ser completamente estabelecida a partir fase, uma vez que, como qualquer área do conhecimento humano, o domínio estudado pode se modificar ao longo do tempo e as mudanças no escopo podem ser necessárias. Com o estudo de domínio pode ser identificado também DSLs para sub-domínios. Este trabalho tem o foco em um domínio mais restritos, que é o de gestos espaciais em jogos esportivos, assim é possível que o resultado do estudo seja a especificação de apenas uma DSL.

4.1.2. Funcionalidades do Domínio Sob Análise

Uma vez que o escopo do estudo do domínio está bem definido, é importante definir também quais aspectos dentro do domínio devem ser analisados. No processo SharpLudus esses aspectos assumem a forma de funcionalidades, uma vez que o foco desta metodologia é observar o domínio do ponto de vista dos *stakeholders* do projeto. No entanto, devido à natureza iterativa e incremental do processo, o estudo do domínio pode resultar na identificação de novas funcionalidades ou na remoção de, modificação ou divisão de algumas. Este trabalho tem um interesse particular em uma funcionalidade do domínio estudado, a foma como se dá a interação do usuário com os jogos, mas outros aspectos dos produtos poderão ser também considerados a fim de se obter uma melhor compreensão do impacto dessas funcionalidades em outros aspectos.

4.1.3. Seleção de Amostras de Linha de Produção

Esta tarefa trata da escolha de produtos existentes, em desenvolvimento ou ainda não desenvolvidos para serem estudados de acordo com os aspectos detectados na fase anterior. A quantidade de jogos a serem analisados é definida principalmente pelos recursos disponíveis da fábrica e pelo cronograma. Assim, é importante tomar decisões para que os jogos selecionados sejam os mais representativos possíveis. O ideal é que bons critérios sejam considerados a fim de selecionar a menor quantidade possível de amostras que representem fielmente o domínio. A tarefa de elicitação pode ser auxiliada por especialistas do domínio, usuários finais e fontes especializadas de informação.

Neste trabalho está sendo considerada a restrição de domínio em jogos esportivos, então se considerou adequado aplicar um processo de seleção baseado na filtragem das amostras iniciais. Este processo de filtragem utilizado está detalhado na subseção 4.2.3, que trata da seleção das amostras.

4.1.4. Execução da Análise do Domínio

Nesta fase todas informações coletadas durante o estudo devem ser agrupadas em um catálogo, facilitando o processo de extração de características comuns e variabilidades. É

importante a adoção de um padrão para se fazer o registro dessas informações, uma opção neste tipo de domínio é o uso de GERs (*Game Evaluation Records* ou Registro de Avaliações de Jogos). Um GER apresenta campos correspondendo às funcionalidades do domínio sob análise, que devem ser preenchidos com as informações extraídas de cada jogo. Dessa forma tem-se a identificação das características de forma ágil e prática.

4.1.5. Extração de Semelhanças e Variabilidades

Até então, tudo o que se tem são as análises individuais das amostras. Esta etapa é fundamental no processo de análise do domínio, pois visa o estudo dessas informações obtidas para criar um melhor entendimento do que é comum e do que é variável no domínio de estudo da fábrica. Dessa forma pretende-se alcançar um nível de abstração que represente todos os jogos pertencentes ao domínio. Para atingir tal objetivo o analista deve reunir todas informações acumuladas na análise das amostras e detalhar quais são os aspectos comuns e como as variações foram identificadas.

Com a restrição de domínio em jogos do gênero esportivo para este trabalho, adotou-se uma estratégia de criação de ontologias[25] para representar o domínio. Para a representação dessas ontologias foi elaborado um catálogo descrevendo os gestos comuns presentes nas amostras, as ações realizadas para executar esses gestos e as características variáveis de cada um, as relações entre esses elementos também é analisada com o objetivo de identificar dependências e hierarquias entre os elementos.

4.1.6. Validação do Domínio

Nesta etapa do processo da metodologia SharpLudus o modelo do domínio gerado deve ser considerado em relação às amostras que não foram escolhidas para testar sua efetividade, ou seja, o jogo escolhido deverá ter suas funcionalidades mapeadas no catálogo. Neste trabalho é feito um breve estudo estatístico da presença dos elementos do domínio, os gestos identificados, no conjunto de jogos para validação.

4.1.7. Análise Incremental e Critério de Parada

Como exposto em um trabalho similar [26], no processo de análise de domínio, experiência e conhecimento são acumulados até atingir-se um limiar. A partir desse ponto deve-se analisar se o domínio possui maturidade suficiente para gerar componentes reusáveis para serem utilizados em uma fábrica de software.

Como mencionado anteriormente, o processo de análise de domínio num contexto de fábrica de jogos é incremental. Processos incrementais, no entanto, deverão contar com o critério de parada bem definido, como a conclusão de um artefato ou realização de um marco, desse modo a metodologia poderá sair do laço e ir para a próxima fase.

O critério de parada para execução de análise de domínio em fábricas de jogos pode ser definido antecipadamente como uma consequência das restrições de projeto, baseada em

[25] Ontologia é uma representação formal de conhecimento como um conjunto de conceitos em determinado domínio e as relações entre eles.

tempo, orçamento, uma combinação de critérios ou numa quantidade de amóstras pré-definidas. No entanto, existe um número ideal de amostras que maximizam o resultado da análise, um número insuficiente de amostras tem um impacto negativo na efetividade da fábrica e exigirá mais iterações, um número grande demais implica na perda de recursos. Uma forma interessante para dizer empiricamente se a análise de domínio pode ser finalizada é a utilização de indicadores que mostrem a relação entre a quantidade de componentes mapeados e não mapeados. Após a análise do domínio, se novas funcionalidades e variações são identificadas, pode significar que a análise realizada não foi suficiente, porém, se as adições não forem relevantes ao entendimento do domínio, pode-se assumir que as semelhanças e variabilidades mais importantes foram cobertas e o esforço para descobrir as novas pode não valer a pena.

4.2. Escopo, Funcionalidades e Seleção de Amostras

Nesta seção a metodologia SharpLudus será aplicada ao domínio de estudo, o de jogos esportivos para Kinect. São executadas as três primeiras atividades da metodologia: estabelecimento de visão e escopo, definição das funcionalidades analisadas, seleção de amostras de linhas de produtos para serem analisadas.

4.2.1. Visão e Escopo

Para o estabelecimento do escopo neste trabalho serão tomadas algumas considerações que ajudam a filtrar as amostras de jogos resultando em um conjunto mais fiel de jogos que representam o domínio de estudo e atendam ao interesse no estudo de NUIs:

- Os produtos a serem analisados devem ser jogos eletrônicos do gênero esportivo;
- Os jogos estudados devem oferecer principalmente interação por meio de gestos espaciais em 2 ou 3 dimensões;
- A identificação dos gestos nos jogos sob análise deve atender aos princípios de NUIs, representando ações intuitivas para o usuário.

A restrição a gestos intuitivos tem o objetivo de fortalecer os princípios de NUIs já apresentados anteriormente para a construção de uma ferramenta que sirva como um exemplo de boas práticas de desenvolvimento na área. Como foi exposto no capítulo 2, muitos jogos estão desvirtuando os fundamentos de NUI com a utilização de comandos não intuitivos ao usuário e constituindo na verdade interfaces não naturais, embora utilizem dispositivos habilitados para esse tipo de interface.

4.2.2. Funcionalidades Sob Análise

O objetivo deste processo de análise é não só de identificar os gestos reusáveis como analisar a relevância que cada um tem para o domínio que está sendo estudado. Mesmo com as restrições aplicadas ao conjunto de jogos a serem estudados é possível que alguns apresentem características que não são muito pertinentes ao domínio, como movimentos de *fitness*, por exemplo.

A tabela 1 detalha um conjunto de características que devem ser analisadas em cada jogo a fim de entender que significado tem cada interação e que relação cada gesto tem com os objetos do mundo do jogo.

Característica	Descrição
Descrição	Breve descrição do jogo, expondo algumas características em geral e o seu contexto.
Entidades interativas	Identificação de elementos do mundo controlados pelo usuário ou que interagem com ele através do reconhecimento de gesto, detalhes dos comportamentos dessas entidades e seus relacionamentos com o mundo do jogo.
Fluxo da interação	Descrição de como acontece o fluxo de interação entre o jogador e os elementos do mundo do jogo.
Gestos	Detalhes dos gestos realizados pelo usuário durante a interação, descrevendo de que forma são realizados.

Tabela 1 - Funcionalidades do domínio sob análise.

4.2.3. Seleção das Amostras

Jogos para a plataforma Xbox podem ser comprados on-line através do Xbox MarketPlace[26]. Este site é o portal online de compras voltado para o Xbox e possui informações sobre todos os jogos lançados para o console, como reviews, fotos, vídeos, rankings, classificações, demos para download, bem como jogos completos, manuais, add-ons para jogos e avatares, entre diversas outras coisas. Para cada país há uma lista diferente de jogos, pois nem todos os jogos são lançados mundialmente, e alguns são lançados somente em um determinado país.

A lista de jogos consultada neste trabalho contempla os jogos que foram lançados nos E.U.A. As amostras selecionadas para esse estudo foram retiradas desta lista por meio de uma busca na loja on-line. Para a escolha das amostras para o domínio de estudo foram realizados uma série de filtragens e ordenação para tentar encontrar o conjunto de jogos mais adequado para análise.

Por meio da loja Marketplace foi realizada uma busca por jogos da plataforma. Em seguida foram aplicadas as seguintes ações sobre o resultado da busca:

1. Aplicou-se um filtro para selecionar apenas os jogos que tem compatibilidade com o Kinect;
2. O resultado anterior foi ordenado pelo número total de vendas (Apêndice A.1);
3. Foram filtrados os jogos com temáticas esportivas de acordo com a classificação da loja (Apêndice A.2);
4. Jogos relacionados a exercícios físicos ou fitness bem como outros jogos que não estavam realmente relacionados ao gênero foram eliminados.

Após essas ações tomadas, pôde-se obter um conjunto de jogos fielmente relacionados ao domínio de estudo selecionado. Embora o resultado tenha encontrado poucos jogos, 12 no

[26] http://marketplace.xbox.com/

total, a maioria dos jogos tem uma série de diferentes modalidades esportivas que podem enriquecer o estudo com uma maior quantidade de gestos identificados. Como o processo de análise de domínio da metodologia SharpLudus tem a característica de ser iterativo e incremental, os jogos selecionados foram divididos em 3 subconjuntos, cada subconjunto possui uma amostra de jogos para serem analisados em uma das interações. A validação do domínio pode ocorrer logo na primeira iteração como em qualquer outra iteração, caso o conjunto de gestos identificados até a iteração atual seja suficiente. Cada subconjunto de jogos e as iterações correspondentes ao estudo deles são mostrados no quadro a seguir:

Iteração	Amostra de jogos
1ª Iteração	• Kinect Sports • Deca Sports Freedom • Wipeout: In The Zone • Motionsports
2ª Iteração	• Kinect Sports: Season Two • Kinect Adventures! • Adrenalin Misfits • Wipeout 2
3ª Iteração	• EA Sports Active 2 • Big League Sports • Virtua Tennis 4 • Kung Fu

Tabela 2 - Conjunto de iterações da metodologia aplicada.

Nesse processo de análise foram usados vídeos, imagens, tutoriais, *reviews*[27] e *walkthroughs*[28] dos jogos listados (o apêndice B contém uma lista com informações sobre as fontes utilizadas) bem como a descrição do jogo na loja e em sites especializados para se obter informações sobre o seu funcionamento de uma forma geral. Além disso, alguns jogos completos e *demos*[29] foram extensivamente jogados pelo autor para auxiliar a identificação de gestos.

4.3. Análise do Domínio

Nesta seção descreve-se a execução das seguintes tarefas relacionadas à análise de domínio definidas pela metodologia SharpLudus para o domínio escolhido para estudo, são elas: a execução da análise de domínio, extração de semelhanças e variabilidade e validação do domínio.

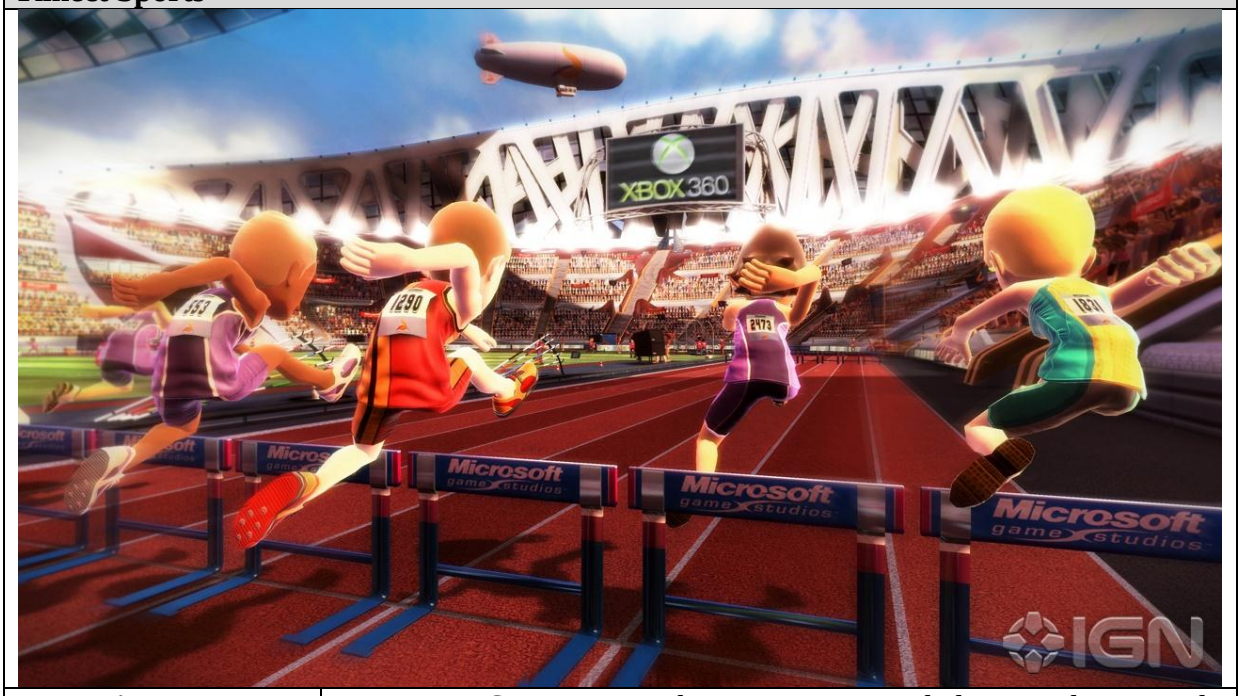
4.3.1. Execução da Análise

Cada jogo das amostras selecionadas foi analisado frente às funcionalidades descritas na subseção 4.2.2. O resultado desta análise encontra-se nas tabelas a seguir.

[27] Avaliação de uma publicação, produto ou serviço. Pode indicar pontos positivos e negativos do que está sendo avaliado.

[28] É um termo muito utilizado nos jogos eletrônicos para indicar a resolução de tarefas passo-a-passo.

[29] Termo utilizado para se tratar da amostra de um jogo que possui parcialmente suas funcionalidades.

Kinect Sports	
	
Descrição	No Kinect Sports o jogador assume o papel de um atleta e pode realizar diversos tipos de provas dentro do jogo de acordo com a modalidade esportiva. As modalidades esportivas presentes no jogo sáo: Atletismo, que subdivide em corrida, lançamento de dardo, lançamento de discos, corrida de obstáculos e salto em distância, boliche, boxe, futebol, vôlei de praia e tênis de mesa. O desafio do jogo está em superar os adversários nas provas das diversas modalidades esportivas e seus próprios recordes.

Entidades interativas	O Kinect Sports tem várias modalidades. As entidades interativas serão descritas para cada modalidade diferente:
	• *Atletismo*: Possui vários tipos de provas:
	  ○ *Corrida*: Um ou dois corredores comandados por meio dos gestos realizados pelo jogadores, além dos obstáculos que devem ser evitados.
	  ○ *Lançamento de dardo*: Assim como na corrida, um atleta deve correr e então projetar o seu corpo para frente e lançar o dardo.
	  ○ *Lançamento de discos*: O atleta deverá lançar um disco o mais longe possível.
	  ○ *Corrida de obstáculos*: O atleta deve correr até o fim da pista saltando os obstáculos presentes.
	  ○ *Salto em distância*: Um atleta deve correr até uma linha limite e então dar um salto.
	• *Boliche*: O jogador é comandado pelos movimentos do corpo. Outra entidade é a bola de boliche.
	• *Boxe*: Um ou dois boxeadores comandados pelos jogadores.
	• *Futebol*: Um ou dois jogadores que representam jogadores no campo de futebol, a bola que deve ser conduzida até o gol.
	• *Vôlei de praia*: Um ou dois jogadores que conduzem uma bola com passes de vôlei para a área dos adversários.

Kinect Sports	
	• *Tênis de mesa*: Um ou dois jogadores jogando uma bola de tênis para o lado do adversário.
Fluxo de interação	Nos menus a interação é baseada em botões virtuais, o usuário aponta sua mão na direção de um botão e então ele a opção é selecionada. Durante o jogo, depois de selecionada a modalidade, os usuários passam a interagir por meio de gestos com os movimentos dos seus corpos.

Gestos	Os gestos identificados detalhados de acordo com cada modalidade:
	• *Atletismo*
	  ○ *Corrida*: Aqui só é realizado o gesto de correr, reconhecido à medida que o jogador levanta os joelhos, quanto mais alto os joelhos forem levantados mais rápido o atleta irá correr.
	  ○ *Lançamento de dardo*: Utiliza o mesmo gesto que na corrida além do gesto de lançamento, que é reconhecido quando o jogador, direciona a mão de trás para frente do corpo.
	  ○ *Lançamento de discos*: Utiliza o mesmo gesto de lançamento que no lançamento de dardos.
	  ○ *Corrida de obstáculos*: Aqui é usado o mesmo gesto da corrida, porém o jogador deve pular quando houver obstáculos no caminho, fazendo com que o atleta dê um salto.
	  ○ *Salto em distância*: O jogador deve executar o gesto de corrida e quando alcançar a linha final deve pular para executar o salto. Quanto mais rápido correr e mais alto pular mais longo será o salto.
	• *Boliche*: Com um dos braços abaixados, o jogador deve movimentá-lo para trás e em seguida para frente para lançar a bola de boliche.
	• *Boxe*: O jogador deve desferir socos no ar para que o lutador acerte seu adversário, os socos podem ser no alto ou embaixo, dependendo da altura dos punhos do jogador.
	• *Futebol*: Está dividido em duas partes, ofensiva e defensiva. Na parte ofensiva o jogador deve direcionar o chute para executar um passe ou chutar para o gol. Na parte defensiva o jogador o jogador pode executar qualquer movimento para bloquear a passagem da bola.
	• *Vôlei de praia*: O jogador deve apontar os braços para cima para passar a bola, pular e mover um dos braços para atacar quando estiver próximo à rede, esses gestos são semelhantes ao de dar uma tapa, ou simplesmente pular com os braços levantados para bloquear o ataque de um adversário.
	• *Tênis de mesa*: Assim como num jogo de tênis, o jogador posiciona uma raquete com uma das mãos e faz um movimento de trás para frente do corpo para passar a bola para o outro lado, como se estivesse dando uma tapa.

Tabela 3 - Análise do Kinect Sports.

Deca Sports Freedom	
	
Descrição	Assim como no Kinect Sports, o Deca Sports Freedom possui uma série de diferentes modalidades de esportes disponíveis para jogar, 10 no total: tênis, tiro com arco, paintball, ski, vôlei de praia, kendô, patinação artística, queimada, snowboarding e boxe. O sucesso do jogo é obtido derrotando os adversários ou superando seus recordes
Entidades interativas	As entidades interativas presentes em cada modalidade de esporte disponível estão detalhadas a seguir: • *Tênis*: Semelhante ao tênis de mesa presente no Kinect Sports, deve-se controlar um tenista que utiliza a raquete para lançar a bola para o lado da quadra do adversário. • *Tiro com arco*: Um atirador que segura um arco carregado com uma flecha para ser disparada em direção a um alvo. • *Paintball*: O jogador controla um personagem que está em um campo de paintball e deve atirar nos adversários. • *Esqui*: O jogador controla um personagem que desce uma pista de esqui segurando um bastão em cada mão para auxiliar a guiar o trajeto. • *Vôlei de praia*: O personagem controlado é um jogador de vôlei que pode executar ações como passar a bola para um companheiro de time, atacar ou defender o ataque do adversário. • *Kendô*: O personagem controlado é um lutador que segura uma espada com as duas mãos desferindo golpes com o objetivo de acertar alguma parte do corpo do adversário. • *Patinação artística*: É controlado um personagem que patina no gelo e deve executar diversas acrobacias para pontuar no jogo. • *Queimada*: O personagem controlado é um dos jogadores da queimada, nesse jogo deve-se lançar a bola para o outro lado

Deca Sports Freedom	
	com o objetivo de atingir um dos jogadores da outra equipe, em seguida deve se defender do ataque do time adversário. • *Snowboarding*: O personagem controlado está sobre uma prancha de snowboarding e deve descer um trajeto executando algumas acrobacias e desviando a direção. • *Boxe*: Assim como está presente no Kinect Sports, o jogador controla um lutador que deve desferir socos com o objetivo de atingir o adversário e vencer a luta.
Fluxo de interação	Os menus do jogo contém botões virtuais que são selecionados ao apontar a palma da mão para eles durante determinado tempo. Após selecionar a modalidade de esporte que irá jogar, o jogador é levado direto para o jogo e passa a interagir de acordo com as ações necessárias por meio de gestos específicos.
Gestos	Segue a lista de gestos identificados em cada modalidade de esporte: • *Tênis*: Como no Kinect Sports, o jogador deve fazer um gesto semelhante a uma tapa para atingir a bola, podendo atingí-la em diferentes ângulos. • *Tiro com arco*: O jogador estende um braço horizontalmente indicando a direção da flecha que será disparada, o outro braço, flexionado, faz um movimento na mesma altura para trás indicando a intensidade da força e em seguida levanta esse braço para efetuar o disparo. • *Paintball*: Para atirar o jogador deve estar com o braço estendido apontando na direção que quer atirar e em seguida levantá-lo rapidamente, para caminhar dentro do campo o jogador pode esquivar-se para esquerda ou direita e mover o corpo para frente ou para trás. • *Esqui*: Para se movimentar o jogador deve manter suas mão à meia altura e esquivar-se de acordo com as inclinações da pista de esqui, ao saltar em um rampa o pode executar uma pose mantendo seus membros fixos em uma diferente pose por 2 segundos. • *Vôlei de praia*: Para passar a bola para um companheiro de time ou para o lado do time adversário deve-se dar um tapa enquanto a bola está no ar, no momento de defender um ataque deve-se posicionar as palmas das mãos juntas e para cima para efetuar o bloqueio. • *Kendô*: O jogador deve manter as mãos juntas com os braços levemente estendidos e executar movimentos de cima para baixo ou de um lado para o outro para atacar o adversário e também se defender de algum golpe. • *Patinação artística*: Para patinar é preciso que o jogador posicione um dos seus pés para trás e em seguida retorne ele para a posição onde estava e coloque o outro pé para trás, repetindo a ação para continuar patinando. Para executar alguma acrobacia basta o jogador realizar alguma pose, mantendo os membros fixos em uma diferente posição por 2 segundos.

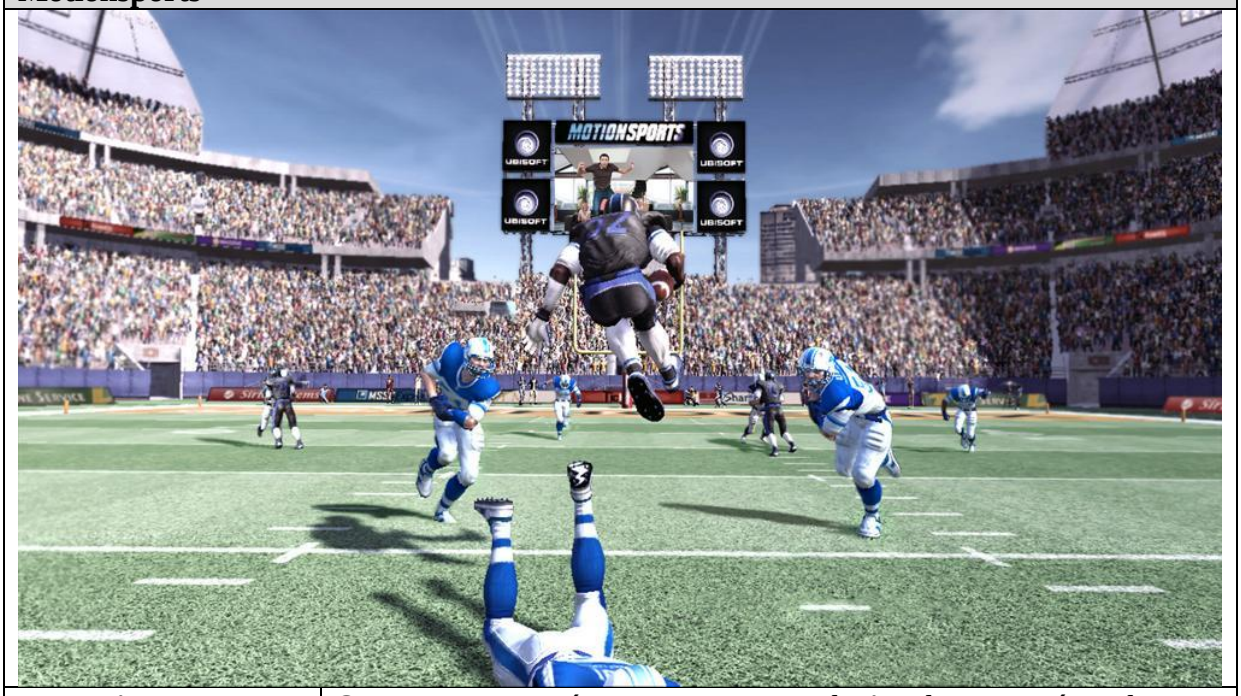
Deca Sports Freedom	
	• *Queimada*: Para atacar o jogador deve executar um lançamento quando a bola estiver nas mãos do seu personagem, movimentando a mão de trás para frente do seu corpo, para a defesa deve-se posicionar as palmas das mãos juntas na direção da bola. • *Snowboarding*: O jogador fica posicionado de lado com os braços estendidos na horizontal para descer a pista, para mudar a direção deve-se inclinar o corpo para o lado que deseja seguir. • *Boxe*: Assim como está presente no Kinect Sports, o jogador controla um lutador que deve desferir socos com o objetivo de atingir o adversário e vencer a luta.

Tabela 4 - Análise do Deca Sports Freedom.

Wipeout: In The Zone	
	
Descrição	Wipeout é um jogo baseado no programa de TV americano de mesmo nome. O funcionamento do jogo é semelhante ao que é visto no programa de TV, o jogador vê seu personagem de um ângulo externo e deve guiá-lo sobre as plataformas evitando os obstáculos. Assim como no programa de TV, o jogador deve correr de um ponto A para um ponto B se esquivando dos obstáculos à sua frente e pulando as lacunas entre plataformas.
Entidades interativas	O jogador interage apenas com personagem que o representa no mundo do jogo, ele deve guiá-lo correndo sobre plataformas que contém surpresas e obstáculos no meio do caminho até chegar ao ponto final.
Fluxo de interação	O fluxo de interação nesse jogo é bastante simples. No menu o jogador aponta para botões virtuais para selecioná-los e durante o jogo deve realizar os gestos necessários para cumprir as provas.
Gestos	Para percorrer as plataformas e superar os obstáculos o jogador deve

Wipeout: In The Zone	
	executar uma série de gestos como correr, pular, esquivar-se e agachar. Esses gestos correspondem a gestos semelhantes no mundo real.

Tabela 5 - Análise do Wipeout: In The Zone.

Motionsports	
	
Descrição	O Motionsports é mais uma compilação de uma série de joos esportivos, é possível escolher entre as modalidades de futebol americano, esqui, futebol, asa-delta, hipismo e boxe. No jogo os objetivos de cada modalidade são os mesmos que os do mundo real, buscando a maior pontuação para conseguir vencer.
Entidades interativas	A principal entidade interativa é o personagem controlado pelo próprio jogador. Em cada modalidade haverá objetivos com que o personagem deve interagir, como a bola no futebol e no futebol americano, segurar a barra da asa-delta, os bastões do esqui, ou a corda que direciona o cavalo no hipismo.
Fluxo de interação	Apresenta um menu com botões que virtuais que mudam de posição quando são apontados, o usuário pode selecionar um desses botões aponando a palma da mão em sua direção por 2 segundos. A partir desse menu é possível escolher uma das modalidades esportivas para jogar. Durante o jogo deve-se realizar gestos específicos para executar ações até alcançar o objetivo. Após a conclusão de cada jogo pode-se tirar uma foto, o jogador simplesmente se posiciona e aguarda o timer para que seja fotografado.
Gestos	Os gestos identificados de acordo com as modalidades de esportes foram: • *Futebol americano*: O personagem estará sempre correndo, não é necessário nenhum gesto para isso. O jogador deve desviar dos adversários se agachando, pulando ou estendendo um dos braços na horizontal na direção correspondente. O

Motionsports	
	jogador também pode executar lançamentos movimentando as mãos juntas de trás para frente do seu corpo. • *Esqui*: O jogador deve manter as mãos à meia altura para o personagem descer a pista e esquivar-se para mudar a direção para o lado correspondente. • *Futebol*: Para chutar a bola o jogador deve simplesmente realizar um chute na direção que quiser, para defender o jogador pode se movimentar livremente para evitar que a bola entre no gol. • *Asa-delta*: O jogador deve manter as mãos à meia altura e levantá-las, abaixá-las ou se inclinar para esquerda ou direita para indicar para que direção o personagem deve se mover. • *Hipismo*: Para fazer com que o cavalo corra deve-se manter as mãos levantadas à meia altura e movê-las rapidamente e repetidamente para cima e para baixo, para executar um salto o jogador deve se agachar e levantar em seguida, a direção deve ser indicada apontando o braço para o lado correspondente. • *Boxe*: Deve-se manter os punhos erguidos na altura do queixo, para aplicar os socos o jogador deve desferir os golpes movendo rapidamente uma de suas mãos para frente.

Tabela 6 - Análise do Motionsports.

Uma observação bastante clara é que a maioria dos jogos são uma compilação de modalidades esportivas, isso levou a identificação de mais gestos do que o esperado. Ficou evidente também que algumas modalidades de esporte são bastante populares e por isso facilitaram o reconhecimento de semelhanças.

Outra observação muito pertinente é o fato de a grande maioria dos gestos identificados serem altamente intuitivos, fazendo com que esses tipos de interação atendam muito bem aos requisitos de interfaces naturais.

4.3.2. Extração de Semelhanças e Variabilidades

O critério para extração de semelhanças foi identificar a utilização de um gesto mais de uma vez no mesmo jogo, em modalidades de esportes diferentes, ou em jogos diferentes. Os gestos identificados receberam um nome em inglês e serão mais detalhados a seguir. O apêndice C contém uma tabela que apresenta a quantidade de vezes que cada gesto identificado é utilizado em diferentes jogos, podendo aparecer mais de uma vez em um jogo se for utilizado em diferentes contexto. Uma análise estatística foi feita para analisar o grau de reusabilidade de cada gesto. A tabela 7 apresenta a quantidade de vezes que cada gesto foi utilizado em diferentes contextos nos jogos analisados. Alguns gestos foram identificados e não apresentaram semelhança com nenhum outro.

Jogos/Gestos	Kinect Sports	Deca Sports Freedom	Wipeout: In The Zone	Motionsports	Total
Run	4	0	1	0	5
Step	4	4	1	2	11

Jogos/Gestos	Kinect Sports	Deca Sports Freedom	Wipeout: In The Zone	Motionsports	Total
Jump	2	0	1	1	4
Throw	3	0	0	1	4
Kick	1	0	0	2	3
Punch	1	1	0	1	3
Slap	2	2	0	0	4
Selection	3	1	1	1	5
Slope	1	3	1	3	8
Crouch	1	0	1	2	4
Handlebars	0	1	0	2	3
Pose	0	2	0	1	3
Block	1	2	0	2	5
Outros	0	4	0	2	6

Tabela 7 - Quantidade de vezes que cada gesto identificado é utilizado pelos jogos do conjunto de análise.

Como pode-se notar, a maioria dos gestos possui grande grau de reuso e constituem um bom conjunto de componentes para se montar uma ontologia de gestos com possibilidade de ser utilizada em uma fábrica de jogos esportivos. Alguns gestos identificados são muito específicos e não fazem muito sentido com o domínio de estudo, por isso não foram incluídos na análise. A validade da ontologia formada por esse conjunto de gestos é analisada na subseção 4.3.3. A tabela 8 apresenta com detalhes os gestos identificados como reusáveis, dando um nome em portugês para cada e detalhando seu uso, a ação que deve ser realizada pelo jogador para que seja reconhecido e as suas variabilidades.

Nome do Gesto	Descrição	
Corrida (Run)	Uso	Usa-se esse gesto para fazer com que o personagem controlado pelo jogador execute uma corrida se deslocando pelo mundo.
	Ação	O jogador deve levantar um dos joelhos e em seguida abaixá-lo enquanto levanta o outro, repetindo o movimento para continuar a corrida.
	Variabilidade	A velocidade da corrida vai ser determinada pela altura que o jogador levanta o joelho. Quando mais alto mais rápida será a corrida.
Pulo (Jump)	Uso	Esse gesto é usado quando deseja que o personagem controlado pelo jogador execute um salto ou um pulo.
	Ação	O jogador pula levantando os joelhos e afastando os pés do chão.
	Variabilidade	O pulo pode ser executado em uma determinada altura, esta seria a altura mínima para validar o pulo.
Lançamento (Throw)	Uso	Esse gesto é utilizado quando deseja atirar algum objeto do jogo para longe.
	Ação	O jogador move o braço estendido que

Nome do Gesto	Descrição	
		segura o objeto de trás para frente do corpo, executando o lançamento do objeto.
	Variabilidade	O objeto será lançado em um ângulo determinado pela angulação do braço que executou o gesto, o jogador pode lançar com o braço esquerdo ou direito, ou utilizar as duas mão, o lançamento pode acontecer de acordo com a posição da mão do jogador, por cima ou por baixo, por exemplo.
Chute (Kick)	Uso	Utilizado para chutar algum objeto presente no mundo do jogo.
	Ação	O jogador move uma de suas pernas de trás para frente do corpo como num movimento de chute.
	Variabilidade	A direção do chute vai depender da direção do movimento da perna do jogador, o jogador pode chutar com a perna esquerda ou direita.
Soco (Punch)	Uso	Usa-se esse gesto quando deseja que o personagem execute um soco.
	Ação	O jogador, com cotovelos e punhos levantados na altura do peito, move o punho para frente para desferir um soco.
	Variabilidade	O jogador pode executar o gesto com a mão esquerda ou direita.
Tapa (Slap)	Uso	Usa-se esse gesto quando deseja que o personagem execute um movimento semelhante a uma tapa para acertar algum objeto no mundo do jogo.
	Ação	O jogador, com o braço levemente estendido, move a palma da mão de trás para frente do corpo.
	Variabilidade	O jogador pode executar o gesto com a mão esquerda ou direita e diferentes velocidades.
Seleção (Selection)	Uso	Usa-se para selecionar alguma opção à amostra para o jogador.
	Ação	O jogador deve apontar a palma da sua mão em direção ao alvo que deseja selecionar e manter essa direção por um determinado tempo para executar a seleção.
	Variabilidade	O gesto pode ser executado com a mão esquerda ou direita e o tempo necessário para validar o gesto pode variar.
Inclinar (Slope)	Uso	Esse gesto é utilizado quando deseja que o personagem do jogo desvie de alguma coisa, ou que tome determinada direção

Nome do Gesto	Descrição	
		caso esteja em movimento.
	Ação	O jogador deve inclinar o seu corpo em alguma direção.
	Variabilidade	O jogador pode inclinar para esquerda, para direita, para frente ou para trás.
Agachar (Crouch)	Uso	Esse gesto deve ser utilizado quando quer que o personagem do mundo do jogo se agache para desviar ou se esquivar de algo.
	Ação	O jogador deve se agachar abaixando os joelhos e parte do corpo para que o gesto seja identificado.
	Variabilidade	Esse gesto pode ser realizado por um determinado tempo para validar alguma ação.
Segurar barras (Handlebars)	Uso	Utiliza-se esse gesto quando quer que o personagem interaja com algum objeto como barras para se mover.
	Ação	O jogador mantém as mãos em meia altura para segurar a(s) barra(s).
	Variabilidade	O jogador pode mudar a direção realizando um gesto representando que está se inclinando para esquerda ou direita.
Pose (Pose)	Uso	Usa-se esse gesto quando quer identificar que o jogador está fazendo alguma pose.
	Ação	O jogador deve posicionar seus 4 membros na forma como deseja realizar a pose e manter a posição por determinado tempo.
	Variabilidade	A pose deve ser mantida por um determinado tempo.
Bloqueio (Block)	Uso	Esse gesto deve ser utilizado quando deseja que o personagem do jogo bloqueie um ataque ou um objeto.
	Ação	Deve-se posicionar as mãos juntas na frente do corpo em uma determinada altura.
	Variabilidade	A altura do bloqueio pode variar ou estar dentro de um intervalo.
Passo (Step)	Uso	Utiliza-se esse gesto para deslocar o personagem em determinada direção e sentido.
	Ação	O jogador deve mover seu corpo de acordo para onde quer que seu personagem se desloque.
	Variabilidade	O jogador pode se mover para frente, para trás, para esquerda ou para direita.

Tabela 8 - Gestos identificados nos jogos analisados.

Ao observar os gestos identificados nos jogos não se viu nenhum tipo de combinação entre dois ou mais gestos. O reconhecimento de alguns gestos se torna difícil em alguns momentos se o jogador não os realizar corretamente.

Outra observação importante é com relação às variações identificadas nos gestos, que geralmente se dá nas seguintes dimensões:

- Sentido;
- Velocidade com que o gesto é realizado;
- Distância de deslocamento do corpo;
- Tempo de duração;
- Regiões válidas;
- De que lado é o membro utilizado para realizar o gesto.

Além disso, outras informações devem ser obtidas quando alguns gestos são executados e podem ser utilizadas como parâmetros para realização ou validação de alguma ação pelo jogo, como:

- Identificação do esqueleto que realiza o gesto;
- Sentido e direção do gesto;
- Velocidade com que um gesto é realizado;
- Distância de deslocamento do corpo;
- Identificação do membro utilizado para executar o gesto.

Essas informações levantadas constituem um conjunto de configurações que devem ser especificados para utilização em algum gesto, para especificar como será realizada a validação do gesto e os tipos de retornos que a realização de cada gesto irá fornecer. É importante observar também que alguns gestos só fazem sentido se realizados em um determinado contexto, como, por exemplo, o bloqueio só faz sentido dentro de um contexto de defesa, seja defendendo algum golpe ou bloqueando um objeto atacado pelo adversário.

4.3.3. Validação do Domínio

Para a validação do domínio será feita uma análise de jogos semelhante a análise realizada anteriormente, dessa vez associando aos gestos já identificados da ontologia e considerando suas variabilidades. Este processo tem o objetivo de fornecer dados estatísticos acerca da possibilidade de resuso de componentes identificados como reusáveis. Caso os jogos analisados apresentem uma boa quantidade de gestos já identificados anteriormente a ontologia pode ser validada e utilizada para a modelagem do domínio. A tabela 9 apresenta um conjunto de jogos escolhidos aleatoriamente dentre os jogos restantes do conjunto de análise e a quantidade de vezes que cada gesto é utilizado por eles em diferentes contextos e variações.

Jogos/Gestos	EA Sports Active 2	Kinect Sports: Season Two	Kinect Adventures!	Big League Sports	Total
Run	2	2	0	3	7
Step	2	1	4	5	12

Jogos/Gestos	EA Sports Active 2	Kinect Sports: Season Two	Kinect Adventures!	Big League Sports	Total
Jump	3	1	2	2	8
Throw	1	5	0	4	10
Kick	0	1	0	3	4
Punch	1	0	0	0	1
Slap	1	1	1	2	5
Selection	3	2	2	4	11
Slope	1	1	1	1	4
Crouch	2	2	3	2	9
Handlebars	1	1	1	0	3
Pose	2	0	2	0	4
Block	0	1	0	4	5
Outros	0	1	0	4	5

Tabela 9 - Quantidade de vezes que cada gesto é utilizado pelos jogos do conjunto de validação.

Algumas foram levadas com relação à validação do domínio nos jogos. No caso do *EA Sports Active 2*, por exemplo, os gestos que não estavam presentes na ontologia de fato não constituiam gestos da prática de esportes, mas de fitness, lembrando que o domínio de estudo ficou restrito em jogos esportivos. Os jogos *Kinect Sports: Season Two* e *Big League Sports* apresentaram gestos compatíveis com o domínio ou variações que não estavam na ontologia, no caso do segundo jogo os gestos não identificados vieram da modalidade de basquetebol que não existia nos jogos analisados anteriormente. Uma nova iteração incluindo esses gestos e variações ou a escolha de um domínio mais restrito poderiam cobrir esses gestos. Entretanto, como explicado na subseção 4.3.4, o critério de parada já foi atingido e com isso foi encerrado o estudo do domínio para este trabalho.

Uma observação pertinente é que, apesar de o trabalho tratar de um domínio mais restrito de jogos, que é o gênero dos jogos esportivos, pode se observar entre muitos jogos de outros gêneros o uso de gestos semelhantes aos identificados aqui, como jogos do gênero de plataforma, ação, aventura, luta, *shooter*[30], entre outros.

4.3.4. Critério de Parada

O critério de parada do trabalho foi estabelecido de acordo com restrições temporais e métricas de aceitação, como a porcentagem de oportunidades de reuso encontradas. Por causa do curto tempo disponível para trabalho no projeto, a 1ª iteração realizada foi avaliada e, devido aos bons resultados alcançados pelas análises, pode-se considerar o conjunto de gestos como um bom modelo de representação do domínio. Observando a tabela 9 pode-se notar a identificação da possibilidade de utilização dos componentes de reuso em mais de 90% dos jogos analisados do conjunto de validação. Com isto é encerrada a análise de domínio para este trabalho. Porém é recomendado que trabalhos futuros continuem com a realização de iterações para análise de jogos além dos que foram validados neste trabalho a fim de se obter um modelo do domínio atualizado e fiel.

[30] É um gênero de jogo eletrônico em que se possui uma arma para atirar livremente.

5. Construção da Linguagem de Domínio Específico

Finalizado os processos estabelecidos pela metodologia, aqui é descrito o processo de criação da DSL focada no domínio de estudo. Para tanto é utilizado o modelo de domínio gerado no capítulo anterior. Algumas considerações devem ser tomadas na criação de uma DSL, este capítulo trata de detalhar como deve ser executado o processo de criação da DSL.

5.1. Modelagem de Uma DSL

Existem duas abordagens com relação a criação de uma DSL, a *top-down*, na qual a linguagem é definida a partir do modelo gerado pela análise de domínio, e *bottom-up*, que vê a linguagem do ponto de vista do estudo da tecnologia de baixo nível relacionada ao domínio. É importante que se considere as duas abordagens, ou a linguagem corre o risco de ser pouco compatível com a tecnologia que se relaciona ou perder parte de sua expressividade [21].

Até então, já foram realizadas diversas tarefas, entre identificar o domínio do problema, juntar o conhecimento relevante neste domínio e agregar esse conhecimento em algumas noções semânticas sobre as mesmas. As próximas atividades estão relacionadas à definição de uma DSL, construção de um framework que implemente as noções semânticas, implementar um compilador que traduz os programas na DSL para uma sequência de chamadas ao framework e criar validações semânticas para identificar erros de modelagem em tempo de desenvolvimento. Furtado [4] detalha as tarefas sugeridas por van Deursen [27] necessárias para o desenvolvimento de uma linguagem de domínio específico:

- *Análise*: (1) Identificar o domínio do problema; (2) Juntar todo conhecimento relevante neste domínio; (3) Agregar esse conhecimento em algumas noções semânticas e operações sobre as mesmas; (4) Definir uma DSL que descreve de forma concisa as aplicações no domínio.
- *Implementação*: (5) Construir um *framework* (biblioteca) que implemente as noções semânticas; (6) Desenvolver e implementar um compilador que traduz os programas na DSL para uma sequência de chamadas ao *framework*; Considerando languages workbenches e modelagem visual, Fowler sugere uma tarefa inicial adicional: (7) a criação de um editor visual para permitir aos desenvolvedores manipular graficamente a DSL. Considerando o contexto de fábrica de software, assim como no SharpLudus, neste trabalho deve-se (8) criar validações semânticas para identificar erros de modelagem em tempo de desenvolvimento.
- *Uso*: (9) Escrever programas na DSL para todas as aplicações desejadas e compilá-los.

As tarefas de 1 a 3 já foram executadas no capítulo anterior, este capítulo concentra-se, portanto, nas atividades de 4 a 8, a atividade 9 será realizada no próximo capítulo através de um estudo de caso.

Neste trabalho é proposta uma DSL visual, o motivo dessa escolha é devido ao curto tempo para implementar ferramentas como parser e a gramática da linguagem caso fosse definir uma DSL textual declarativa. Outro motivo para a criação de uma DSL visual é a maior facilidade que a ferramenta pode oferecer, não sendo necessário conhecimento da

gramática, basta apenas descrever os componentes e conectá-los por meio de uma interface gráfica. A ferramenta apresentada anteriormente, VMSDK para o Visual Studio, é utilizada para a definição da DSL.

5.2. Requisitos Identificados

O foco deste trabalho está em construir uma ferramenta que auxilie no desenvolvimento de jogos esportivos para uma interface natural de usuário, deve-se levar em consideração o atendimento desta aos princípios fundamentais de NUIs, além de outros detalhes, como:

- Escolha de um conjunto intuitivo de gestos;
- Configuração dos gestos escolhidos.

Assim a linguagem deve definir contextos de interação onde os gestos possíveis podem ser identificados e configurados de acordo com os parâmetros de cada um. Outros requisitos importantes identificados para a linguagem são:

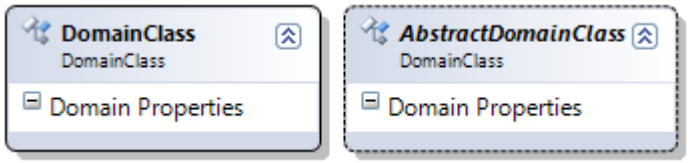
- Notação gráfica simples, intuitiva e expressiva;
- Boa performance em execução;
- Biblioteca de gestos extensível.

5.3. Especificação da Linguagem KGL

Nesta seção é descrita a linguagem criada, denominada KGL (*Kinetic Gesture Language*), oferecendo uma breve introdução aos conceitos envolvidos e detalhando alguns elementos mais importantes como os tipo de dado e os gestos e, finalmente, validando a linguagem frente aos requisitos.

5.3.1. Definição da KGL

Uma DSL visual é composta por diferentes elementos: um grafo de conceitos (ou classes), relacionamentos (papéis que compreendem, cardinalidade, etc.), atributos, entre outros. Uma abordagem para especificar uma DSL visual é utilizar uma linguagem de meta-modelagem. Neste trabalho será utilizada a linguagem de meta-modelagem da ferramenta VMSDK do Visual Studio.

Nome do conceito	Descrição	
Domain Class	Descição	Uma *domain class* representa um elemento básico de uma DSL visual. Ela se refere a um conceito do domínio, como um gesto. Ela também pode ser abstrata.
	Representação Gráfica	
Root class	Descrição	A *root class* é o principal conceito do modelo, como o contexto. É a base para relacionar todos os outros conceitos do domínio.
	Representação	Mesma representação que *Domain Class*.

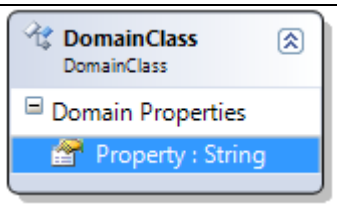
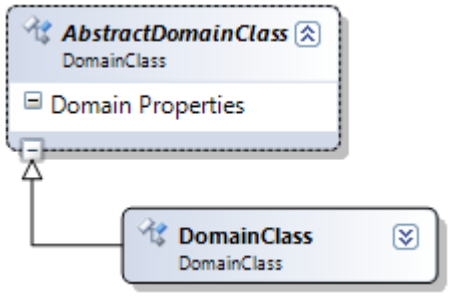
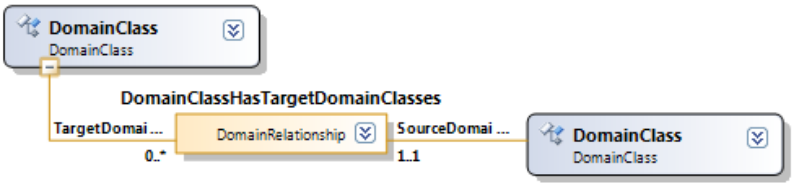
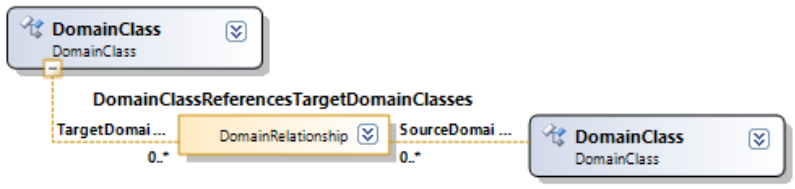
Nome do conceito	Descrição	
	Gráfica	
Domain Property	Descrição	Representa um atributo fortemente tipado de uma dada classe. Por exemplo, *name* é um atributo do tipo String para a classe Context.
	Representação Gráfica	
Domain Enumeration	Descrição	Esses são tipos especiais de valores. Por exemplo, Side é um *domain enumeration* que inclui os valores Left e Right.
	Representação Gráfica	Não possui representação gráfica.
Inheritance relationship	Descrição	Usado para expressas que um conceito é uma especialização de um outro.
	Representação Gráfica	
Embedding relationship	Descrição	Um embedding relationship representa uma relação forte entre dois conceitos. Por exemplo, um Context tem Gestures. Se um Context é deletado do modelo, todos os seus Gestures são deletados. Papéis e cardinalidades também se aplicam a esse tipo de relacionamento.
	Representação Gráfica	
Reference relationship	Descrição	Um reference relationship representa uma relação fraca entre dois conceitos.
	Representação Gráfica	

Tabela 10 - Linguagem de meta-modelagem do VMSDK.

Uma linguagem de meta-modelagem é usada para construir um meta-modelo, que descreve uma linguagem de modelagem (como a KGL), da mesma forma que um modelo descreve um sistema.

O meta-modelo da KGL é apresentado nas figuras a seguir, onde pode-se observar que a definição de uma ontologia, gerada no capítulo anterior, deixa a definição da DSL mais intuitiva e precisa.

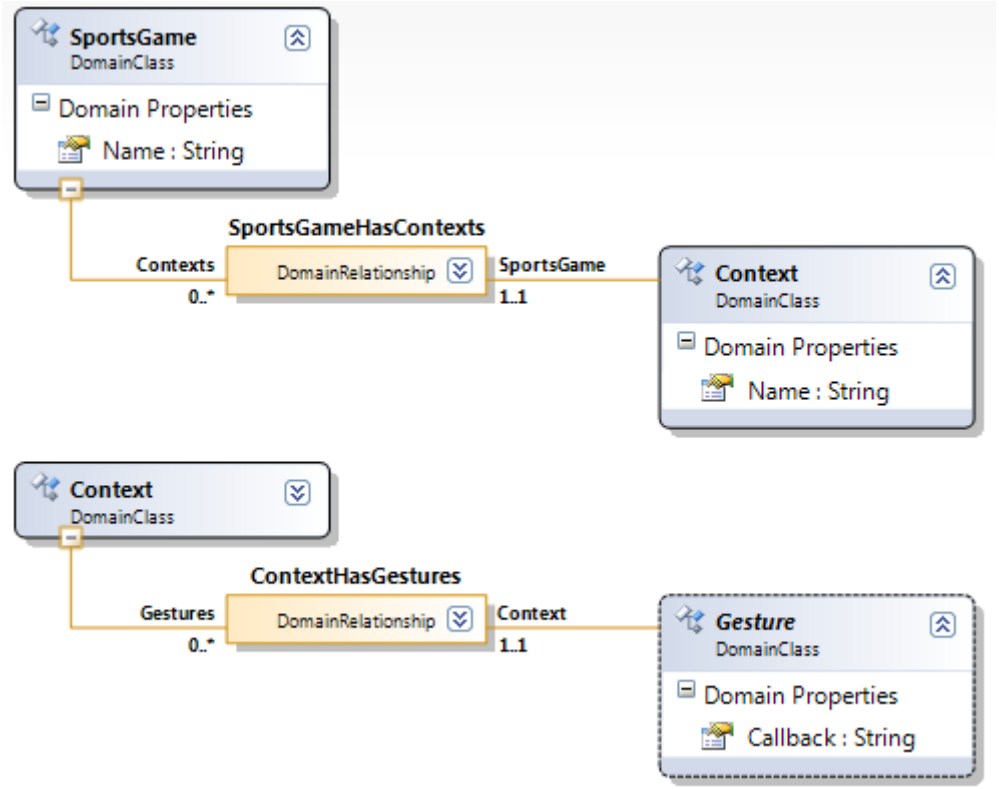


Figura 19 - Meta-modelo da KGL (1): Jogo, contextos e gestos.

A figura 19 apresenta a classe raiz (SportsGame), que define o nome do jogo e possui vários Contexts, que por sua vez possuem um nome associado e vários gestos relacionados, cada gesto possui um Callback, que é o nome da função a ser chamada ao reconhecer o gesto. Os tipos de gestos estão descritos no meta-modelo da KGL na figura 20.

Cada tipo de gesto possui diferentes propriedades que representam as variabilidades encontradas na análise dos jogos. A definição dos gestos e suas propriedades foram feitas baseadas na ontologia validada anteriormente. Com isso temos todas as classes do modelo do domínio definidas. Novos tipos de enumeração foram criados para representar variabilidades encontradas na ontologia:

- *Direction*: Indica a direção de um gesto: *Backward* (para trás), *Forward* (para frente), *ToTheLeft* (para esquerda), *ToTheRight* (para direita).
- *Side*: Indica de que lado fica a mão ou pé deve ser utilizado no gesto: *Left* (esquerdo); *Right* (direito).
- *Speed*: Indica com que velocidade deve ser executado o gesto: *Fast* (rápido), *Normal* (normal), *Slow* (devagar).

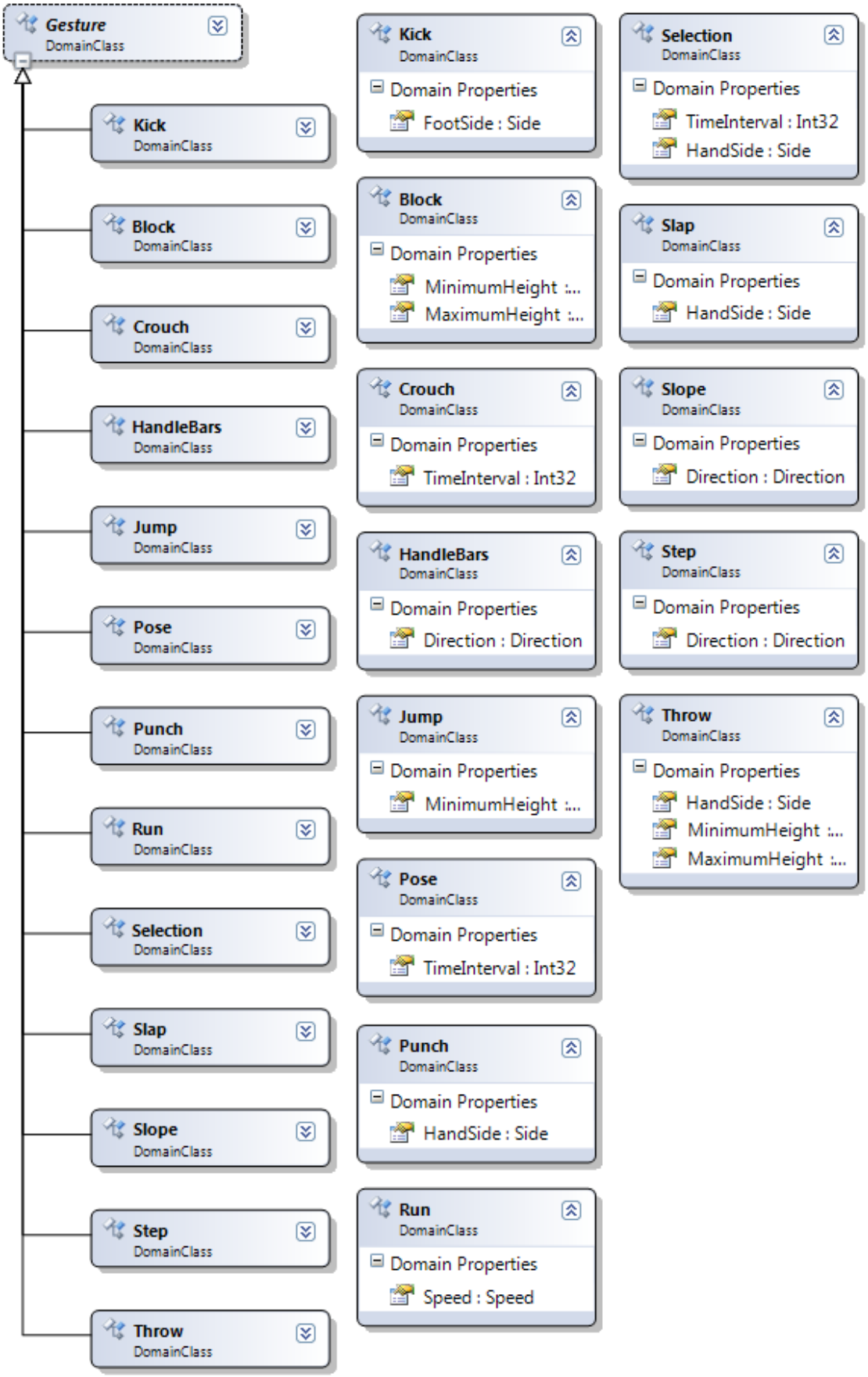


Figura 20 - Meta-modelo da KGL (2): Tipos de gestos e suas propriedades.

Algumas considerações devem ser feitas com relação aos valores válidos para as propriedades de algumas classes, esses valores estão determinados no próprio SDK do Kinect, portanto eles serão mantidos aqui. Regras de validação serão criadas para garantir que os valores entrados estejam dentro dos limites estabelecidos pelo SDK. Esses limites estão listados a seguir:

- *TimeInterval* >= 0;
- *MinimumHeight* e *MaximumHeight* devem possuir um valor entre -1 e 1.

As restrições de MinimumHeight e MaximumHeight tem a ver com os valores de referência que o Kinect utiliza para medir distâncias. A distância *z* da pessoa até o Kinect é medida em metros enquanto os objetos nos eixos x e y assumem valores entre -1 e 1. Ao utilizarmos esses valores para trabalhar em alguma aplicação devemos utilizar operações de vetores para reconstruí-los corretamente no tamanho da *viewport*.

5.3.2. KGL e Editor Visual

A sintaxe da linguagem define como os elementos da linguagem aparecem em uma forma concreta e usável pelo usuário. No caso de linguagens visuais, a sintaxe não é puramente textual; ela combina gráficos, textos e convenções pelas quais os usuários da linguagem podem interagir para configurar seu produto. Os elementos da sintaxe visual da KGL estão presentes na tabela 11.

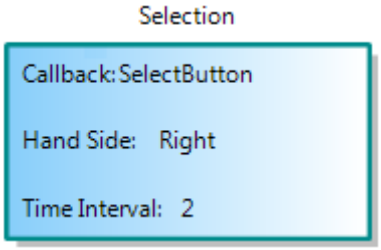
Nome	Representação Gráfica	Descrição
Context		Um Context é representado por uma caixa branca de contorno vermelho, o nome ao centro corresponde ao nome do Context definido.
Gesture		Um Gesture é representado por uma caixa azul com bordas escuras. Acima da caixa se localiza o nome do gesto, dentro da caixa tem-se o nome da função de callback caso o gesto seja reconhecido, e em seguida as propriedades específicas de cada gesto.
Connector		Representa a conexão estabelecida entre um Context com um Gesture

Tabela 11 - Elementos da sintaxe da KGL.

O Language Workbench composto pelo Visual Studio e VMSDK é uma ferramenta poderosa para desenvolvimento de DSLs, quase todas as funcionalidades e conceitos do ambiente são utilizados dentro da ferramenta de modelagem visual. É possível a utilização de componentes do ambiente, como toolbox, janela de propriedades das classes e um navegador que inclui todas as classes utilizadas no projeto de forma hierarquizada.

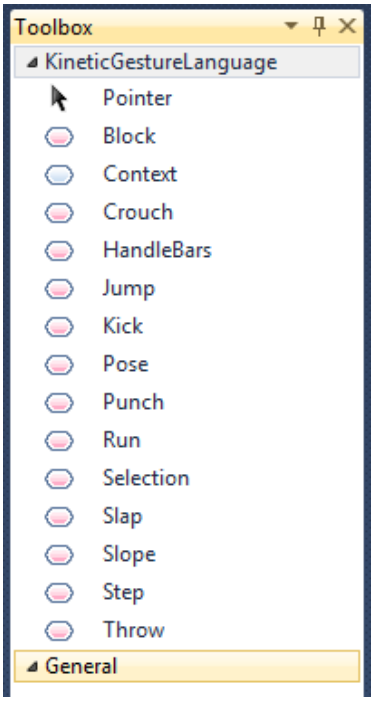


Figura 21 - Ferramentas do editor visual da KGL.

Os principais conceitos da linguagem KGL, Context e Gesture, podem ser manipulados pelo designer do jogo através da janela de propriedades da IDE (Figura 22), que é sensível ao elemento em foco, dependendo do elemento selecionado a janela exibirá editores de propriedades correspondentes à classe.

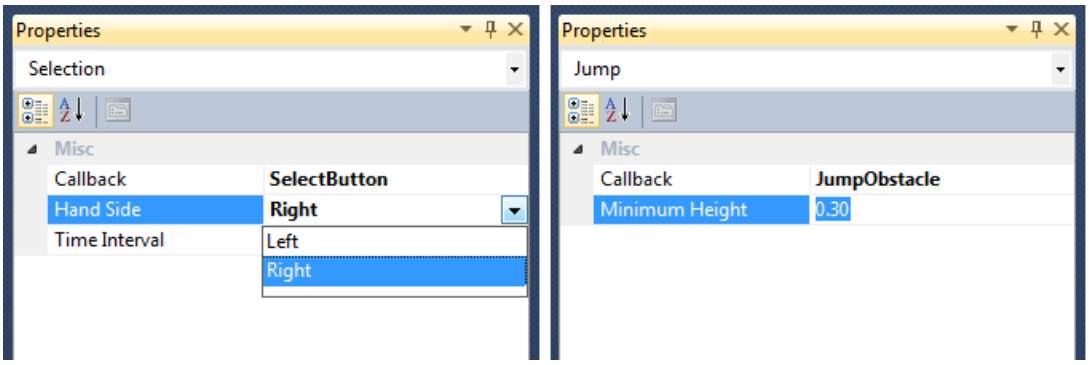


Figura 22 - Janela de propriedades da IDE.

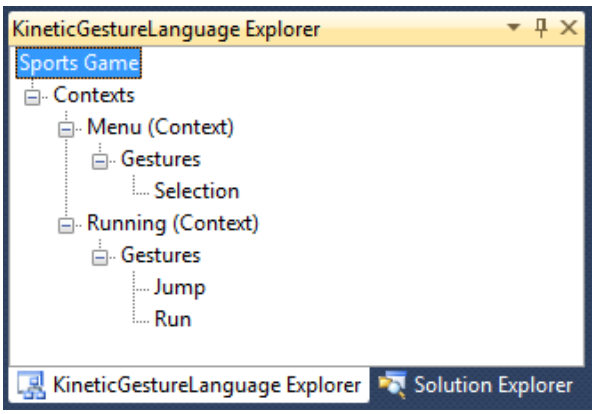


Figura 23 - KGL Explorer.

5.3.3. Validadores Semânticos da KGL

Além de auxiliar o designer de jogos com funcionalidades de edição visual, a experiência de modelagem com a KGL também garante que a lógica semântica da DSL seja respeitada pelo designer de jogos. Isso é feito por meio de validadores semânticos, que estão associados a um conceito do domínio e pode exibir erros de validação na lista de erros da IDE. Na figura 21, um exemplo de erro: o nome da função de callback é inválido.

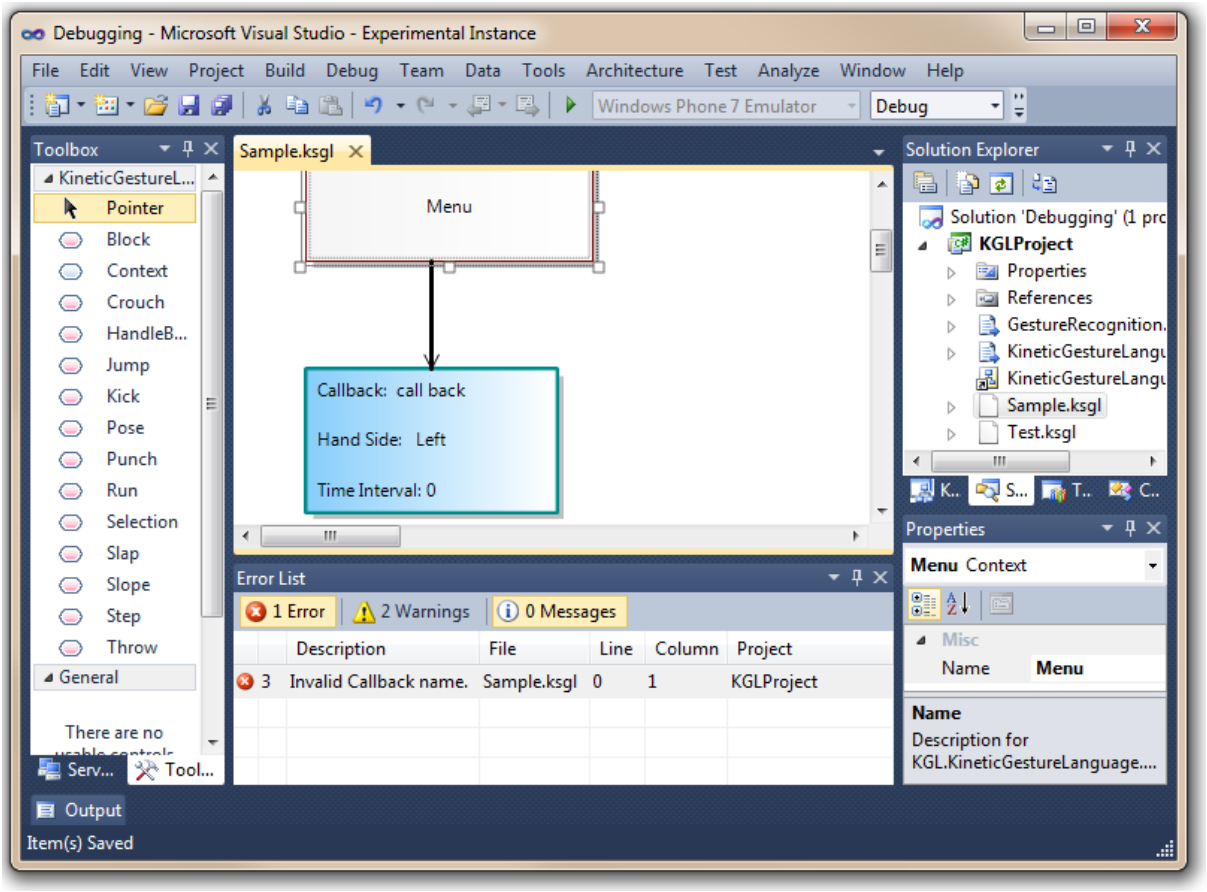


Figura 24 - Erros semânticos exibidos numa lista de erros da IDE.

No caso da ferramenta VMSDK para o Visual Studio, validadores semânticos podem ser criados através de programação. Uma vez que cada conceito de domínio gera uma classe parcial em C#, tudo que o designer da DSL precisa fazer é implementar um método na classe parcial do conceito desejado que verifique as condições de erro especiais e registre os erros. A figura 22 mostra a implementação de erros de validação relacionados aos gestos da linguagem.

```
bool isValid = !string.IsNullOrEmpty(g.Callback) && !g.Callback.Contains(' ');
if (!isValid)
{
    context.LogError("Invalid Callback name.", "InvalidCallBackName", this);
}
else
{
    if (g is Selection)
    {
        if (((Selection)g).TimeInterval > 5)
        {
            isValid = false;
            context.LogError("Invalid time interval.", "InvalidSelectionTimeInterval", this);
        }
    }
    else if (g is Block)
    {
        if (((Block)g).MinimumHeight < -1 || ((Block)g).MinimumHeight > 1 ||
            ((Block)g).MaximumHeight < -1 || ((Block)g).MaximumHeight > 1 ||
            ((Block)g).MinimumHeight > ((Block)g).MaximumHeight)
        {
            isValid = false;
            context.LogError("Invalid values for height.", "InvalidBlockHeightInterval", this);
        }
    }
    else if (g is Crouch)
    {
        if (((Crouch)g).TimeInterval > 10)
        {
            isValid = false;
            context.LogError("Invalid time interval.", "InvalidCrouchTimeInterval", this);
        }
    }
    else if (g is Jump)
    {
        if (((Jump)g).MinimumHeight < -1 || ((Jump)g).MinimumHeight > 1)
        {
            isValid = false;
            context.LogError("Invalid value for height.", "InvalidJumpHeightInterval", this);
        }
    }
    else if (g is Throw)
    {
        if (((Throw)g).MinimumHeight < -1 || ((Throw)g).MinimumHeight > 1 ||
            ((Throw)g).MaximumHeight < -1 || ((Throw)g).MaximumHeight > 1 ||
            ((Throw)g).MinimumHeight > ((Throw)g).MaximumHeight)
        {
            isValid = false;
            context.LogError("Invalid values for height.", "InvalidThrowHeightInterval", this);
        }
    }
    else if (g is Pose)
    {
        if (((Pose)g).TimeInterval > 10)
        {
            isValid = false;
            context.LogError("Invalid time interval.", "InvalidPoseTimeInterval", this);
        }
```

Figura 25 - Implementando um validador semântico para os gestos.

Alguns exemplos de regras semânticas da KGL que podem ser encontradas através dos validadores são:

- Os contextos e gestos devem ter nomes válidos (Sem espaços);
- Nos gestos que utilizam um intervalo de altura a altura mínima não pode ser maior que a altura máxima;
- Valores válidos para altura devem estar entre -1 e 1;
- Os intervalos de tempo não podem ser muito longos (no máximo 10 segundos);
- Os nomes das funções de callback devem ser válidos, para que se possa executá-los quando um gesto for validado.

A figura 26 apresenta todo o conjunto de funcionalidades que o uso do workbench Visual Studio em conjunto com VMSDK oferece ao designer do jogo.

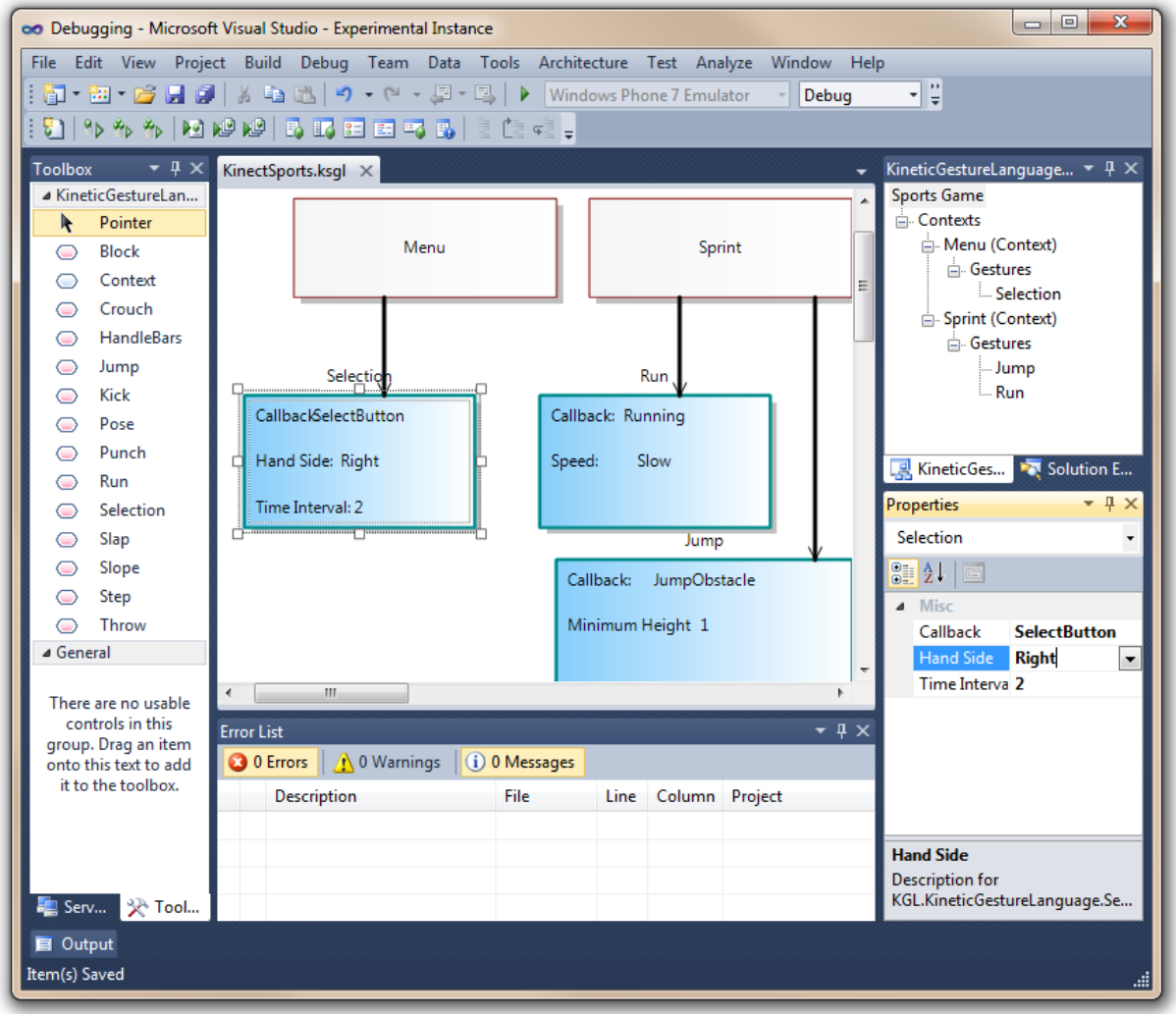


Figura 26 - Completa interface de modelagem da KGL.

5.3.4. Geração de Código e Framework

O código gerado pela KGL corresponde a um conjunto de classes que fazem parte do framework para reconhecimento dos gestos, essas classes implementam o reconhecimento de gestos de acordo com a biblioteca de gestos gerada e as configurações dos gestos. Essas classes podem ser acessadas e editadas manualmente e incluídas no projeto do jogo. A figura 27 mostra um diagrama de classes geradas a partir de um exemplo de jogo definido pela KGL.

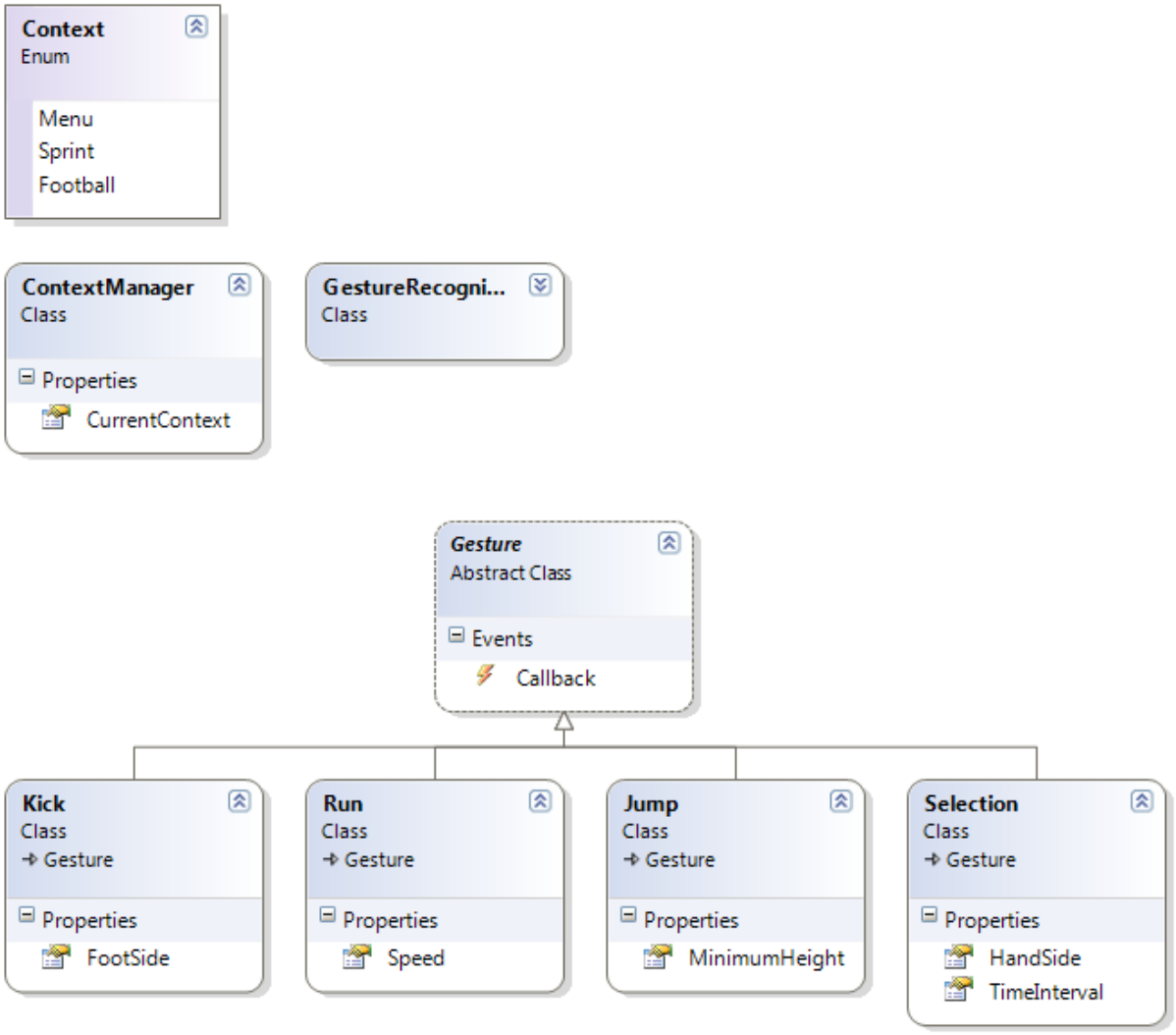


Figura 27 - Diagrama de classes de um exemplo de código gerado.

Esse conjunto de classes abstraem grande parte da complexidade do jogo, incluindo o reconhecimento dos gestos. A subseção a seguir discute a implementação de técnicas para reconhecimento de gestos fazendo uma breve análise sobre cada uma. Para uma primeira versão da KGL é proposto um subconjunto de gestos que devem ter seus reconhecedores implementados. A escolha dos gestos envolve questões de complexidade e de tempo. A tabela 12 apresenta um subconjunto de gestos escolhidos para serem implementados na primeira versão da KGL especificando entradas e saídas.

Gesto	Parâmetros	Saídas
Kick	Side FootSide	Vector3 Direction

Gesto	Parâmetros	Saídas
		Double Speed Int PlayerId
Run	Speed Speed	Int PlayerId
Jump	Double MinimumHeight	Int PlayerId
Selection	Int TimeInterval	Int PlayerId String SelectedButtonId

Tabela 12 - Gestos propostos para versão inicial da KGL.

5.3.5. Reconhecimento de Gestos

Esta subseção tem a intenção de fazer uma breve apresentação sobre algumas técnicas de reconhecimento de gestos. É importante fazer uma análise das opções pois cada uma é adequada para ser utilizada no reconhecimento de gestos de uma complexidade específica, cabe ao desenvolvedor decidir que técnica irá utilizar. Aqui serão abordadas duas técnicas: Algoritmica e *Exemplar Matching*.

Algoritmica

Define um gesto algoritmicamente, no caso de um aceno de mão, por exemplo:

- Mão acima do cotovelo;
- Mão na frente do ombro;
- Mão se movendo em uma mesma direção;
- Mão se movendo a uma certa velocidade;
- Quantidade de vezes que houve mudança de direção.

A utilização da técnica algoritmica tem vantagem por ser de fácil entendimento, fácil de implementar e de testar. No entanto, é difícil escolher os melhores valores para os parâmetros, não é escalável para variações do mesmo gesto e é muito complicado implementar essa técnica para gestos mais complexos e compensar os atrasos. Portanto é recomendado utilizá-la para reconhecimento de gestos mais simples, como Selection, Jump, Run.

Exemplar Matching

Define gestos pelo uso de animações gravadas anteriormente, deve-se escolher diferentes pessoas para executar o gesto, cada pessoa deve executar o gesto diversas vezes. Um *exemplar* é um exemplo ideal para comparação. Para verificar a correspondência é preciso se fazer a comparação entre *frames* dos esqueletos:

- Definir uma métrica de erros para os equeletos;
- Diferença angular para cada articulação no espaço local.

A busca pelos frames que melhor correspondem ao gesto pode ser realizada por meio de classificadores como *Dinamic Time Warping*[31] (DTW) e Modelo Oculto de Markov (HMM).

[31] http://en.wikipedia.org/wiki/Dynamic_time_warping

Esta técnica possui vantagens por poder detectar gestos mais complexos facilmente, compensar atrasos e ser escalável para variações do mesmo gesto. Por outro lado, requer muitos recursos para ser robusta (múltiplas gravações de múltiplas pessoas executando um gesto), exigindo mais poder de processamento e memória. É recomendada quando quer reconhecer gestos mais complexos e a redução do atraso é um fator crítico.

Para a implementação dos gestos propostos neste trabalho a técnica mais adequada é a algoritmica, que, em teoria, exige menos tempo para implementação e por ser fácil de entender e testar, além de se adequar bem aos simples gestos escolhidos.

5.3.6. Adequação aos Requisitos

Como visto nas análises realizadas no domínio anteriormente, a linguagem KGL descreve bem uma grande quantidade de gestos identificados no contexto de forma clara. A configuração dos gestos pode ser realizada facilmente por meio da interface gráfica para edição. Os gestos selecionados correspondem a gestos intuitivos e naturais, sendo muito comum a realização deles no dia-a-dia dos usuários. Assim, a DSL proposta aqui atende aos principais requisitos.

Com relação aos outros requisitos levantados anteriormente, pode-se dizer que possui extensibilidade para implementar novos gestos para o *framework* e a performance em execução está relacionada às técnicas de reconhecimento utilizadas para implementar os reconhecedores de gestos. Fica a cargo do designer dos novos gestos utilizar a técnica mais adequada.

6. Estudo de Caso

Este capítulo apresenta a criação de um jogo do gênero esportivo, assim como os jogos do domínio estudados aqui neste trabalho, o jogo é bastante simples. Se trata de um jogo de futebol onde o jogador pode treinar cobranças de pênalti. Inicialmente é apresentado um menu onde o jogador pode escolher de que lado vai bater, esquerdo ou direito, ele deve posicionar a mão direita sobre o botão correspondente durante 2 segundos. Em seguida ele é posto em campo, de frente para o gol para bater o pênalti. O jogador pode deslocar o ângulo da câmera para esquerda ou para direita, inclinando seu corpo para o lado correspondente. Quando estiver preparado, basta chutar a bola e ela irá seguir em determinada direção. No primeiro momento são treinados chutes com o pé esquerdo, no segundo momento são treinados chutes com o pé direito.

6.1. Implementação

Aqui serão levados em consideração os gestos necessários para jogar tal jogo. Podemos identificar três gestos presentes na KGL:

- *Selection*: utilizado no menu para selecionar de que lado irá bater o penalti, só é possível selecionar utilizando a mão direita;
- *Slope*: utilizado para deslocar o ângulo da câmera em relação ao jogador, ele pode deslocá-lo para esquerda ou para direita;
- *Kick*: usado para executar a ação de chutar a bola. O chute deve ser realizado com o pé esquerdo na primeira cobrança e com o pé direito na segunda cobrança.

Nota-se também o uso de três contextos aí: menu, cobrança1 e cobrança2. No contexto do menu utiliza-se somente o gesto de seleção, os outros gestos não fazem sentido nenhum aqui. Nos outros contextos, utiliza-se somente os gestos de inclinação e chute, uma vez que já foi selecionado o lado para bater o pênalti, na primeira cobrança o chute deve ser realizado com o pé esquerdo e, na segunda, com o pé direito.

A modelagem desses contextos é feita de maneira muito simples. A figura a seguir apresenta a modelagem dos contextos deste jogo.

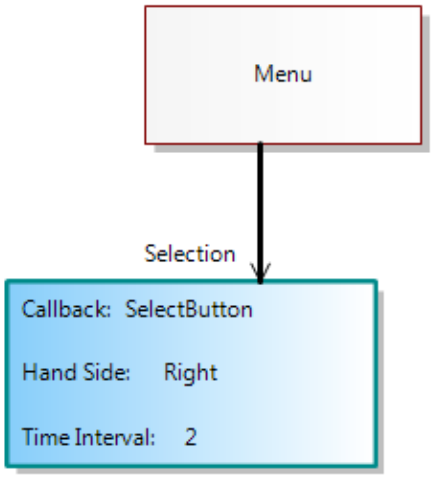


Figura 28 - Contexto de menu do jogo.

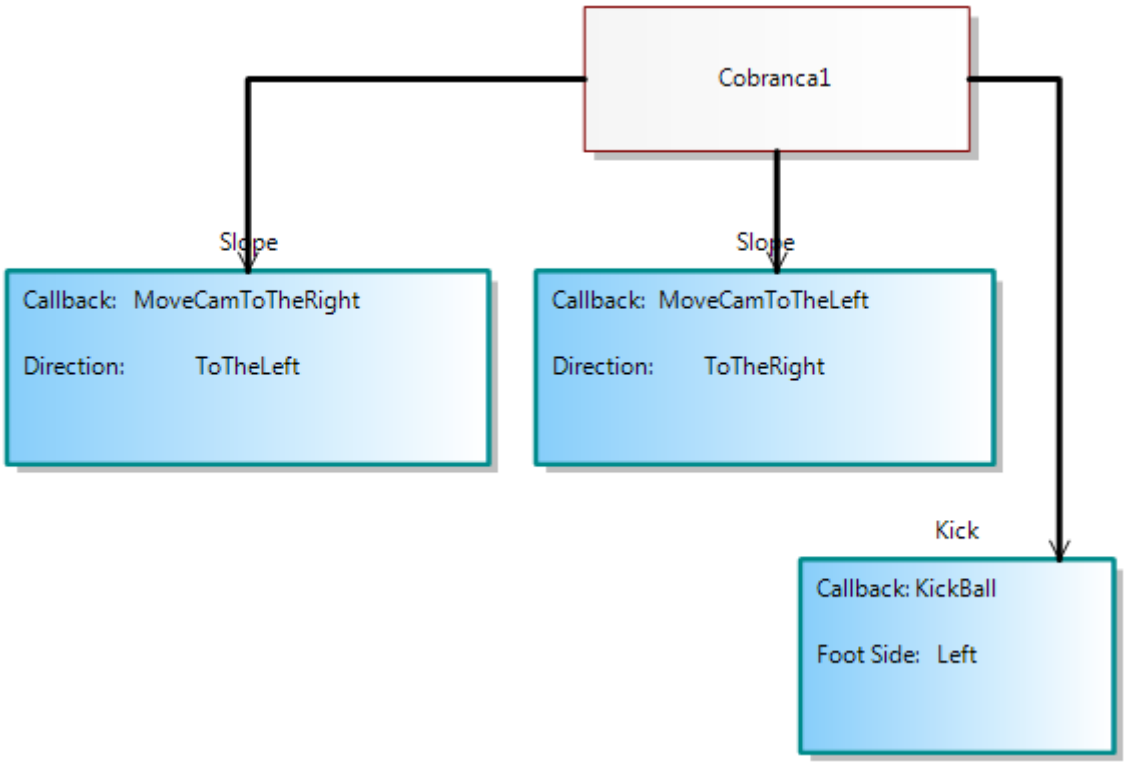


Figura 29 - Contexto da primeira cobrança de penalti.

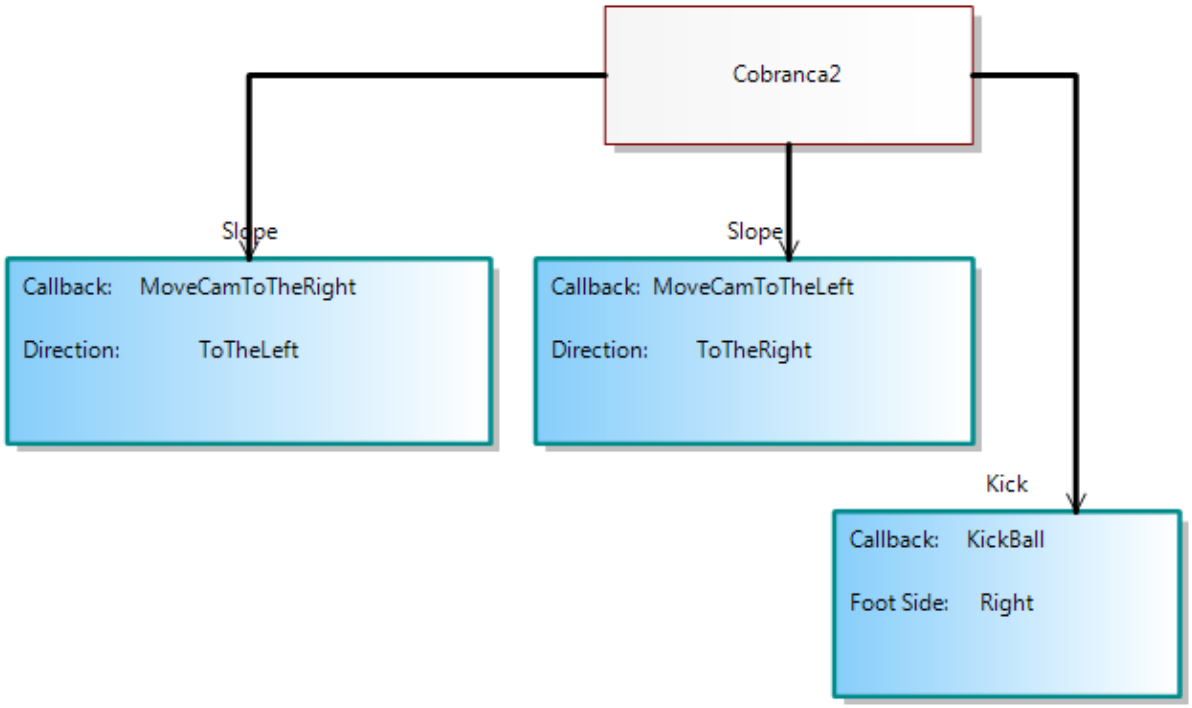


Figura 30 - Contexto da segunda cobrança de penalti.

Importante notar a partir daí que deve-se fazer o uso de callbacks para executar os comandos no jogo que atendem às ações relacionadas com os gestos. A figura 31 mostra o diagrama das classes geradas pela DSL e a tabela 14 mostra como devem ser mapeadas as infomações obtidas para as funções de callback realizarem a lógica associada.

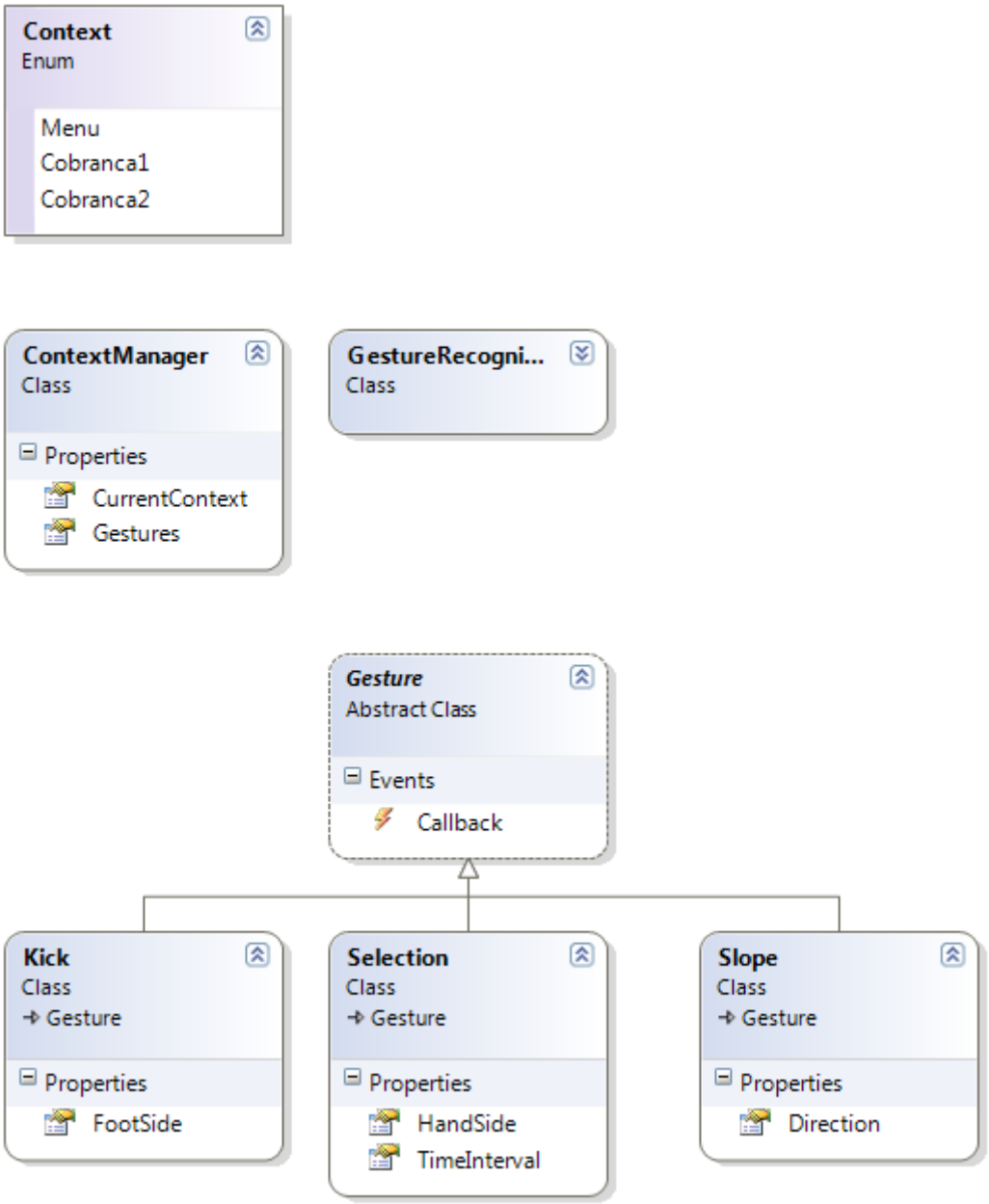


Figura 31 - Classes geradas pela KGL para uso do jogo.

Gesto	Parâmetros	Callback
Selection	Right (Side), 2 (TimeInterval)	SelectButton(Position selectPosition)
Slope	ToTheLeft (Direction)	MoveCamToTheRight()
Slope	ToTheRigt (Direction)	MoveCamToTheLeft()
Kick	Left (Side)	KickBall(Vector3 direction)
Kick	Right (Side)	KickBall(Vector3 direction)

Tabela 13 - Mapeamento de gestos para callbacks.

Assim, com os contextos e gestos do jogo já definido, resta a tarefa do desenvolvedor de programar a lógica do jogo contida nas funções de callback descritas acima.

6.2. Análise dos Resultados

Com a adoção da DSL desenvolvida foi possível uma grande redução na quantidade de código utilizado para implementar a interação, que passou a ser reconhecida através de uma biblioteca de gestos reusável. Outro aspecto muito importante a ser observado é abstração de toda a complexidade necessária para realizar o reconhecimento dos gestos. Como vimos anteriormente, algumas soluções exigem muito tempo para implementação e testes até que se obtenha um reconhecedor de gestos ideal. A adoção da KGL facilitou a definição de gestos utilizando uma notação gráfica de forma que os gestos podem ser definidos por outros stakeholders do projeto além dos desenvolvedores. Sem a adoção da KGL este trabalho estaria limitado às atividades dos desenvolvedores por envolver o entendimento da tecnologia, de algoritmos e de técnicas sofisticadas.

7. Conclusões

Este trabalho não se resume apenas na proposta do uso de uma ferramenta para desenvolvimento de jogos. Como foi visto ao longo do documento, foi identificada com sucesso uma linha de produto de softwares dentro do domínio, de jogos do gênero esportivo, de onde pôde-se extrair informações o bastante para se notar o grande potencial de reusabilidade que possui. Assim, vê-se uma boa oportunidade para se tratar desses jogos do ponto de vista das fábricas de software. Foi com esta visão que foi concebida a DSL proposta por este trabalho.

Outra discussão bastante relevante que foi realizada aqui foi o do uso de interfaces humano-computador ao longo do tempo, analisando sua evolução do ponto de vista do quanto que cada uma oferece em termos de meios intuitivos para interação. Nota-se um equívoco no uso do termo "interface natural" no uso de algumas aplicações para interfaces de toque e de gestos; de fato, esses dispositivos são dispositivos de interface natural, porém é necessário que as aplicações que rodam neles habilitem essa interface oferecendo um meio de interação tão intuitivo que a interface pareça invisível para o usuário. Assim, a discussão foi levada no sentido de defender os princípios das interfaces naturais.

Com relação ao uso de ferramentas no contexto de fábricas de software, foi possível realizar todo o trabalho utilizando somente um ambiente de desenvolvimento, o que se encaixa perfeitamente na definição de language workbenches sugerida por Fowler. O ambiente Visual Studio se mostrou uma ferramenta muito poderosa quando equipada com extensões adequadas. É importante notar que, apesar do tempo que se leva para configurar esse ambiente, os benefícios oferecidos por isso são alcançados de maneira muito rápida.

Finalmente, observou-se que a adoção das metodologias e boas práticas propostas pelas fábricas de software no domínio das interfaces naturais contribui bastante para o seu crescimento, visto que o estabelecimento de padrões por meio da Engenharia de Software pode ajudar a preservar os princípios das NUIs. Além disso, apesar da complexidade envolvida nas tecnologias para reconhecimento de gestos, nota-se que a utilização de fábricas de software reduz drasticamente o tempo necessário para implementação.

7.1. Contribuições

O objetivo inicial deste trabalho era a proposta de uma DSL para auxiliar o desenvolvimento de jogos para Kinect, uma plataforma que está ganhando mais mercado e atraingo muitas pessoas por dar uma visão de como serão as interfaces no futuro. Uma análise do domínio foi realizada com sucesso e pode-se tomar como exemplo para outros trabalhos semelhantes. Uma linguagem de domínio específico, chamada KGL (*Kinetic Gesture Language*) foi então criada a partir deste estudo com o objetivo de permitir a fácil utilização e configuração de um conjunto de gestos em diferentes contextos de jogos. Embora um domínio mais restrito tenha sido utilizado para o estudo, observou-se que muitos dos gestos identificados podem ser reusados em jogos de outros gêneros.

7.2. Trabalhos Futuros

Como visto anteriormente, a biblioteca de gestos implementada é pequena em relação ao conjunto de gestos identificados no domínio. Além disso, em breve haverá necessidade de se realizar novos estudos, executando novas iterações de análise do domínio, para enriquecer a ontologia de gestos. São nesses pontos que o projeto pode ser melhorado em relação a DSL. O estudo do reconhecimento de gestos tem sido muito discutido nesse meio, apresentando diversos meios e técnicas para auxílio. Algumas dessas técnicas envolve bom entendimento na área de Inteligência Artificial, portanto vê-se aí importantes aplicações novas surgindo para essa área. A utilização dessas técnicas contribuiria muito para tornar o reconhecimento de gestos mais efetivo.

Outro ponto interessante a se tocar em trabalhos futuros na área é a realização desses estudos em outros domínios, visto que é um tipo de tecnologia com grande potencial para ser utilizada em diversas atividades além de jogos.

7.3. Considerações Finais

Pelo fato de o trabalho tratar do uso de um dispositivo lançado recentemente houve uma série de dificuldades encontradas, desde problemas de compatibilidade do hardware até suporte oferecido para se trabalhar com ele. Essas dificuldades impactaram negativamente no trabalho causando um atraso que não estava previsto. Entretanto, vê-se um crescimento da comunidade desenvolvedora na área, novos dispositivos semelhantes ao Kinect serão lançados em breve e provavelmente novas ferramentas com melhor suporte.

Referências

[1] Guinness World Records, “Kinect Confirmed As Fastest-Selling Consumer Eletronics Device,” Guinness World Records, 30 03 2011. [Online]. Available: http://community.guinnessworldrecords.com/_Kinect-Confirmed-As-Fastest-Selling-Consumer-Electronics-Device/blog/3376939/7691.html. [Acesso em 10 11 2011].

[2] Gamer Investments, “July 2011 NPD U.S. Hardware and Software Sales,” 11 08 2011. [Online]. Available: http://gamerinvestments.com/video-game-stocks/index.php/2011/08/11/july-2011-npd-u-s-hardware-and-software-sales-data/. [Acesso em 15 09 2011].

[3] Gamer Investments, “October 2011 NPD U.S. Hardware and Software Sales,” 11 11 2011. [Online]. Available: http://gamerinvestments.com/video-game-stocks/index.php/2011/11/11/october-2011-npd-u-s-hardware-and-software-sales-data/. [Acesso em 18 11 2011].

[4] A. W. Furtado, SharpLudus: Improving Game Development Experience through Software Factories and Domain-Specific Languages, Recife: Centro de Informática, Universidade Federal de Pernambuco, 2006.

[5] A. W. Furtado e A. L. M. Santos, “Software Factories Relevance to Digital Games: A Practical Discussion,” em *SBGames*, Recife, 2006.

[6] S. H. Khandkar e F. Maurer, “A Domain Specific Language to Define Gestures for Multi-Touch Applications,” em *The 10th Workshop on Domain-Specific Modeling*, Reno/Tahoe, 2010.

[7] L. P. Marinho, GAL: Uma Linguagem de Domínio Específico Para Jogos Multi-Toque, Recife: Centro de Informática, Universidade Federal de Pernambuco, 2010.

[8] J. J. LaViola Jr. e M. R. L., “An Introduction to 3D Spatial Interaction with Video Game Motion Controllers,” em *37th International Conference and Exhibition on Computer Graphics and Interactive Techniques, SIGGRAPH*, Los Angeles, 2010.

[9] D. Natapov, S. J. Castellucci e I. S. MacKenzie, “ISO 9241-9 Evaluation of Video Game Controllers,” em *Graphics Interface*, Kelowna, 2009.

[10] A. de los Reys, “Predicting the Past,” em *Web Directions*, Sydney, 2008.

[11] D. Wigdor e D. Wixon, Brave NUI World: Designing Natural User Interfaces for Touch and Gesture, Morgan Kaufmann, 2011.

[12] D. A. Norman, “Natural User Interfaces Are Not Natural,” *interactions,* vol. 17, n. 3, pp. 6-10, 2010.

[13] L. Zaman, D. Natapov e R. J. Teather, “Touchscreens vs. Tradicional Controllers in Handheld Gamming,” em *International Academic Conference on the Future of Game*

Design and Technology, FuturePlay, Vancouver, 2010.

[14] G. Goth, "Brave NUI World," *Communications of the ACM,* vol. 54, n. 12, pp. 14-16, 2011.

[15] JDSU, "Optical 3D Gesture Recognition and JDSU," 2011. [Online]. Available: http://www.jdsu.com/en-us/Custom-Optics/applications/gesture-recognition/Pages/optical-3d-gesture-recognition-and-jdsu.aspx. [Acesso em 20 11 2011].

[16] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp e M. Finocchio, "Real-Time Human Pose Recognition in Parts from Single Depth Images," em *IEEE Computer Vision and Pattern Recognition, CVPR,* Colorado, 2011.

[17] M. Schramm, "Kinect: The company behind the tech explains how it works," 19 06 2010. [Online]. Available: http://www.joystiq.com/2010/06/19/kinect-how-it-works-from-the-company-behind-the-tech/. [Acesso em 20 11 2011].

[18] P. Clements, "Software Product Lines: A New Paradigm for the New Century," *The Journal of Defense Software Engineering,* pp. 20-22, 1999.

[19] J. Greenfield e K. Short, "Moving to Software Factories," 20 09 2004. [Online]. Available: http://blogs.msdn.com/b/askburton/archive/2004/09/20/232021.aspx. [Acesso em 04 11 2011].

[20] S. Cook, G. Jones, S. Kent e A. C. Wills, Domain-Specific Development with Visual Studio DSL Tools, Addison-Wesley Professional, 2007.

[21] M. Fowler, "Language Workbenches: The Killer-App for Domain Specific Languages?," 12 06 2005. [Online]. Available: http://martinfowler.com/articles/languageWorkbench.html. [Acesso em 15 10 2011].

[22] H. Wassner, "kinect + reseau de neurone = reconnaissance de gestes," 06 05 2011. [Online]. Available: http://professeurs.esiea.fr/wassner/?2011/05/06/325-kinect-reseau-de-neurone-reconnaissance-de-gestes. [Acesso em 08 12 2011].

[23] Prieto-Díaz, "Domain Analysis: An Introduction," *Software Engineering Notes,* vol. 15, n. 2, pp. 47-54, 1990.

[24] K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak e A. S. Peterson, "Feature-Oriented Domain Analysis (FODA) Feasibility Study," 1990.

[25] J. Coplien, D. Hoffman e D. Weiss, *Commonality and Variability in Software Engineering,* IEEE Software, 1998.

[26] L. S. d. S. T. Azevedo, ELEGY: Aplicando o processo de Fábrica de Jogos ao domínio de Role-Playing Games (RPGs), Recife: Centro de Informática, Universidade Federal de Pernambuco, 2009.

[27] A. van Deursen, P. Klint e J. Visser, “Domain-Specific Languages: An Annotated Bibliography,” 09 02 2000. [Online]. Available: http://homepages.cwi.nl/~arie/papers/dslbib/. [Acesso em 15 11 2011].

Apêndice A1 - Lista de jogos para Kinect

Nome	Lançamento	Gênero
Forza Motorsport 4	11/08/2011	Racing & Flying
Kinect Sports	04/11/2010	Sports
Dance Central	01/11/2010	Music/Rhythm
Dance Central 2	25/10/2011	Music, Dance
Kinect Fun Labs	06/06/2011	Adventure, Party, Puzzle
Sonic Free Riders	04/11/2010	Racing & Flying
Kinectimals	11/10/2011	Family
Kinect Joy Ride	04/11/2010	Racing
Fruit Ninja Kinect	08/11/2011	Action & Adventure, Family
The Biggest Loser: Ultimate Workout	31/05/2011	Family, Sports & Recreation
Your Shape: Fitness Evolved	04/11/2010	Sports
Rise of Nightmares	06/09/2011	Action & Adventure
Zumba Fitness	26/11/2010	Music, Sports & Recreation
DanceMasters	21/12/2010	Music
The Gunstringer	13/09/2011	Action & Adventure, Shooter
Kinect Sports: Season Two	25/10/2011	Sports
Body and Brain Connection	08/02/2010	Family, Educational, Puzzle & Trivia
Hole in The Wall	24/08/2011	Family
Fantastic Pets	08/04/2011	Family
Fighters Uncaged	04/11/2010	Action & Adventure, Fighting
Sesame Street: Once Upon a Monster	11/10/2011	Family, Adventure
Harry Potter and the Deathly Hallows – Part 1	16/11/2010	Action & Adventure, Family, Shooter
Build A Buddy (Kinect Fun Labs)	-	-
UFC Personal Trainer	28/06/2011	Fitness
Deca Sports Freedom	30/08/2011	Sports & Recreation
EA Sports Active 2	16/11/2010	Sports & Recreation
Game Party in Motion	-	Famil, Sports & Recreation
Avatar Kinect (Kinect Fun Labs)	-	-
Bobble Head (Kinect Fun Labs)	-	-
Motionsports	31/05/2011	Sports & Recreation
Child of Eden	14/06/2011	Music, Shooter
Kinect Adventures!	04/11/2010	Action
Carnival Games: MSMD	05/04/2011	Family
Raving Rabbids: A&K	-	Family
Chevrolet Volt and Kinect Joy Ride	-	Family
Leedmees	07/09/2011	Family
Just Dance 3	07/10/2011	Family, Music
Air Band	22/08/2011	Music
PowerUp Heroes	-	Fighting
Yoostar On MTV	15/11/2011	-
Yoostar 2	21/07/2011	Family
Grease Dance	01/11/2011	Music
Your Shape: Fitness Evolved 2012	28/11/2011	Family

Nome	Lançamento	Gênero
Disneyland Adventures	15/11/2011	Family
Kung Fu Panda 2	29/11/2011	Action & Adventure
Adrenalin Misfits	09/08/2011	Action & Adventure, Racing & Flying, Sports & Recreation
Deepak Chopra's Leela	08/11/2011	-
Twister Mania	16/11/2011	Family, Puzzle & Trivia
Michael Phelps: Push the Limit	15/11/2011	Family, Sports & Recreation
Michael Jackson The Experience	-	Music, Strategy & Simulation
Mutation Station (Kinect Fun Labs)	26/09/2011	-
Kinect Sparkler (Kinect Fun Labs)	-	-
Let's Cheer!	15/11/2011	Family, Music
Avatar Superstar	01/06/2011	Music
Harry Potter and the Deathly Hallows	-	Action & Adventure
Haunt	-	Action & Adventure
Nicktoons MLB	-	Family, Sports & Recreation
Hulk Hogan's Main Event	-	Fighting, Sports & Recreation
Wipeout 2	-	Platformer, Sports & Recreation
Battle Stuff (Kinect Fun Labs)	-	-
SpongeBob's Surf & Skate Roadtrip	08/11/2011	Sports & Recreation
Puss in Boots	-	Action & Adventure, Family
Jillian Michaels Fitness Ultimatum 2012	-	Sports & Recreation
Big League Sports	-	Sports & Recreation
Musical Feet (Kinect Fun Labs)	24/10/2011	-
Rapala for Kinect	-	Sports & Recreation
Adventure Camp	-	Family
Nickelodeon Dance	-	Family, Music
Black Eyed Peas Experience	-	Music
Virtua Tennis 4	-	Sports & Recreation
Kung Fu	24/11/2011	-
Wipeout: In The Zone	-	Family, Platformer
Just Dance Kids 2	-	Family, Music
Blackwater Kinect	18/10/2011	Shooter
Kinect Me (Kinect Fun Labs)	-	-
The Price Is Right	13/08/2011	Family
Family Game Night 4: The Game Show	25/10/2011	Family
Jillian Michaels's Fitness Adventure	04/11/2011	Other
Victorious: Time to Shine	-	Music
Kinect Googly Eyes	-	-
The Penguins of Madagascar	-	Action & Adventure
Who Wants To Be A Millionaire	-	Family, Puzzle & Trivia
Alvin and The Chipmunks: Chipwrecked	10/10/2011	Family
Dance Paradise	-	Music

Apêndice A2 - Jogos esportivos para Kinect

Nome	Lançamento	Gênero
Kinect Sports	04/11/2010	Sports
The Biggest Loser: Ultimate Workout	31/05/2011	Family, Sports & Recreation
Your Shape: Fitness Evolved	04/11/2010	Sports
Zumba Fitness	26/11/2010	Music, Sports & Recreation
Kinect Sports: Season Two	25/10/2011	Sports
Deca Sports Freedom	30/08/2011	Sports & Recreation
EA Sports Active 2	16/11/2010	Sports & Recreation
Game Party in Motion	-	Family, Sports & Recreation
Motionsports	31/05/2011	Sports & Recreation
Kinect Adventures!	04/11/2010	Action
Adrenalin Misfits	09/08/2011	Action & Adventure, Racing & Flying, Sports & Recreation
Michael Phelps: Push the Limit	15/11/2011	Family, Sports & Recreation
Nicktoons MLB	-	Family, Sports & Recreation
Hulk Hogan’s Main Event	-	Fighting, Sports & Recreation
Wipeout 2	-	Platformer, Sports & Recreation
SpongeBob’s Surf & Skate Roadtrip	08/11/2011	Sports & Recreation
Jillian Michaels Fitness Ultimatum 2012	-	Sports & Recreation
Big League Sports	-	Sports & Recreation
Rapala for Kinect	-	Sports & Recreation
Virtua Tennis 4	-	Sports & Recreation
Kung Fu	24/11/2011	-
Wipeout: In The Zone	-	Family, Platformer, Sports & Recreation

Apêndice B - Fontes de consulta sobre jogos

Nome	Endereço	Descrição
Xbox Marketplace	http://marketplace.xbox.com/en-US/Games?genre=3028&PageSize=90&Page=1	É o portal oficial de compras online para Xbox, contém listas de todos os jogos, classificações, gênero, reviews, imagens, vídeos e outras informações dos jogos, além de conteúdos disponíveis para download no xbox, como demos, complementos para jogos e jogos completos.
123kinect	http://123kinect.com/	Um dos maiores portais com informações sobre jogos para Kinect. Contém imagens, vídeos, notícias, *reviews* dos jogos lançados e informações sobre jogos que ainda serão lançados.
KinectAddict	http://www.youtube.com/user/kinectaddict	Canal do youtube especializado em Kinect, é possível encontrar vídeos de reviews, tutoriais e walkthrough de diversos jogos.
Zomb1eHeadShot	http://www.youtube.com/user/Zomb1eHeadShot	Outro canal do youtube muito popular e dedicado a apresentação de *reviews* e *gameplay* de jogos para Xbox 360 e Kinect.
XboxViewTV	http://www.youtube.com/user/XboxViewTV	Um dos canais do youtube relacionados a Xbox com mais acesso. Contém informações sobre jogos, como trailers, vídeos de lançamentos oficiais e o que vem sendo produzido pelos desenvolvedores de jogos para essa plataforma.
This Side of Insanity	http://www.youtube.com/user/lawrenceraybon	Canal do youtube que contém longos e detalhados *reviews* de diversos jogos para Kinect, apresentando formas de interação, dicas e informações.

Nome	Endereço	Descrição
Mahalo	http://www.youtube.com/user/MahaloVideoGames	Outro canal do youtube bastante acessado e com muitas informações sobre jogos de diversas plataformas.
EA Sports Active 2 Hot Site	http://www.ea.com/ea-sports-active-2/	Hot Site do jogo *EA Sports Active 2*. Contém informações sobre o jogo, vídeos e imagens.
IGN Entertainment	http://www.youtube.com/user/IGNentertainment	Canal do youtube de um dos sites mais populares com reviews de jogos.
Motion-Gaming	http://www.motion-gaming.fr/	Site francês dedicado à análise de jogos que utilizam sensores de movimento.

Apêndice C - Matrizes de identificação de gestos reusáveis

Kinect Sports							
Modalidades /Gestos	Atletismo	Boliche	Boxe	Futebol	Vôlei de praia	Tênis de mesa	Total
Run	4	0	0	0	0	0	4
Step	0	0	1	1	1	1	4
Jump	2	0	0	0	0	0	2
Throw	2	1	0	0	0	0	3
Kick	0	0	0	1	0	0	1
Punch	0	0	1	0	0	0	1
Slap	0	0	0	0	1	1	2
Selection	2	1	0	0	0	0	3
Slope	0	0	1	0	0	0	1
Block	0	0	0	0	1	0	1
Outros	0	0	0	0	0	0	0

Deca Sports Freedom						
Modalidades /Gestos	Tênis	Tiro com arco	Paintball	Esqui	Vôlei de praia	Total
Step	1	0	0	0	1	2
Slap	1	0	0	0	1	2
Slope	0	0	0	1	0	1
Handlebars	0	0	0	1	0	1
Pose	0	0	0	1	0	1
Block	0	0	0	0	1	1
Outros	0	1	1	0	0	2

Deca Sports Freedom					
Modalidades /Gestos	Kendô	Patinação artística	Snowboarding	Boxe	Total
Step	1	0	0	1	2
Punch	0	0	0	1	1
Slope	0	0	1	1	2
Pose	0	1	0	0	1
Block	0	0	0	1	1
Outros	2	0	0	0	2

Gestos	Wipeout: In The Zone	Total
Run	1	1
Step	1	1
Jump	1	1
Slope	1	1
Crouch	1	1
Outros	0	0

Motionsports							
Modalidades /Gestos	Futebol americano	Esqui	Futebol	Asa-delta	Hipismo	Boxe	Total
Step	0	0	1	0	0	1	2
Jump	1	0	0	0	0	0	1
Throw	1	0	0	0	0	0	1
Kick	1	0	1	0	0	0	2
Punch	0	0	0	0	0	1	1
Slap	0	0	0	0	0	0	0
Slope	0	1	0	1	0	1	3
Crouch	1	0	0	0	1	0	2
Handlebars	0	1	0	1	1	0	2
Pose	0	1	0	0	0	0	1
Block	0	0	0	0	1	1	2
Outros	1	0	0	0	1	0	2