Universidade Federal de Juiz de Fora
Instituto de Ciências Exatas
Bacharelado em Ciência da Computação

Um Algoritmo Adaptativo On-Line para Jogos de Luta

Renan Motta Goulart

JUIZ DE FORA
NOVEMBRO, 2014

Um Algoritmo Adaptativo On-Line para Jogos de Luta

Renan Motta Goulart

Universidade Federal de Juiz de Fora

Instituto de Ciências Exatas

Departamento de Ciência da Computação

Bacharelado em Ciência da Computação

Orientador: Guilherme Albuquerque Pinto

JUIZ DE FORA

NOVEMBRO, 2014

Um Algoritmo Adaptativo On-Line para Jogos de Luta

Renan Motta Goulart

MONOGRAFIA SUBMETIDA AO CORPO DOCENTE DO INSTITUTO DE CIÊNCIAS EXATAS DA UNIVERSIDADE FEDERAL DE JUIZ DE FORA, COMO PARTE INTEGRANTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE BACHAREL EM CIÊNCIA DA COMPUTAÇÃO.

Aprovada por:

Guilherme Albuquerque Pinto
Doutor em Ciência da Computação - UNICAMP

Raul Fonseca Neto
Doutor em Eng. de Sistemas e Computação - UFRJ

Victor Stroele
Doutor em Eng. de Sistemas e Computação - UFRJ

JUIZ DE FORA

28 DE NOVEMBRO, 2014

Dedicado a minha mãe.

Resumo

Jogos eletrônicos são uma das principais aplicações de técnicas de Inteligência Artificial. A capacidade de aprendizado em um algoritmo adaptativo tornam essas técnicas interessantes principalmente para jogos onde dois jogadores se opõem e apresentam estilos característicos, com certos padrões de ação e reação, o que pode permitir a previsão dos movimentos. Para jogos de luta, nos quais os jogadores atuam simultaneamente, torna-se importante o aprendizado *on-line*, ou seja, durante a realização da luta.

Neste Trabalho de Conclusão de Curso foi desenvolvido um algoritmo adaptativo, seguindo a técnica de TDL - *Temporal Difference Learning*, que tenta prever a ação do adversário para escolher a melhor contra-medida, o melhor movimento, a se executar. O algoritmo foi implementado em Java, dentro do ambiente de simulação de jogos de luta *FightingICE*, da Universidade de Ritsumeikan, no Japão. Ele foi submetido para a competição "Figthing game AI Competition", que se realizou no congresso CIG-2014, *IEEE Conference on Computational Intelligence and Games*, na Alemanha, com excelentes resultados, ficando em terceiro lugar em uma das categorias.

Palavras-chave: Algoritmo Adaptativo, Inteligência Artificial, Jogos de Luta, *TDL - Temporal Difference Learning*.

Abstract

Electronic games are one of the main aplications of Artificial Inteligence techniques. The capability of learning in an adaptative algorithm makes these techniques interesting for games where two opposing players present characteristic styles, with some action and reaction patters, wich can lead to the prediction of movements used by these players. For fighting games, in wich both players fight simultaneasly, learning becomes important during the fight.

In this final paper it was developed an adaptative algorithm following the technique of TDL - *Temporal Difference Learning*, wich tries to predict the oponent's action to choose what is the best counter measure, the best movement, to execute. The algorithm was implemented in Java using the enviroment for simulation of fighting games FightingICE, from Ritsumeikan University, Japan. It was submitted to the competition *"Figthing game AI Competition"*, wich took place in the congress CIG-2014, IEEE Conference on Computational Intelligence and Games, in Germany, with exellent results, placing third in one of the categories.

Keywords: Adaptive Algorithm, Artificial intelligence, Fighting Games, TDL - Temporal Difference Learning.

Agradecimentos

Agradeço a minha mãe pelo apoio durante todos estes anos e por tudo que ela fez por mim.

Agradeço ao meu orientador pela ajuda e pela motivação que ele me deu para concluir a monografia.

'No, it's nothing as ambiguous as memory'

Serial Experiments Lain

Sumário

Lista de Figuras

Lista de Tabelas

1 Introdução

Jogos eletrônicos são uma das principais aplicações de técnicas de Inteligência Artificial. A capacidade de aprendizado em um algoritmo adaptativo tornam essas técnicas interessantes principalmente para jogos onde dois jogadores se opõem e apresentam estilos característicos, com certos padrões de ação e reação, o que pode permitir a previsão dos movimentos. Para jogos de luta, nos quais os jogadores atuam simultaneamente, torna-se importante o aprendizado *on-line*, ou seja, durante a realização da luta.

Jogos de luta tem voltado a se tornarem muito populares em anos recentes tendo avançado muito em suas mecânicas de jogabilidade e gráficos, porém houve poucas melhoras nas técnicas de inteligência artificial utilizadas, sendo a grande maioria delas não demonstrando nenhuma forma de aprendizado. As técnicas mais avançadas necessitam de váreas lutas para poder aprender algo(GRAEPEL et al, 2006). Nesta monografia foi realizado a implementação de uma inteligência artificial que consegue se adaptar ao seu oponente durante a luta. A implementação foi feita em java utilizando o ambiente FightingICE(LU et al, 2013), ela fora enviada para uma competição internacional de IAs de jogos de luta realizada no congresso CIG-2014 ficando em terceiro lugar em uma das duas categorias.

A inteligência artificial implementada teve como base um algoritmo desenvolvido com inspiração na técnica de inteligência computacional *Temporal Difference Learning*(SUTTON, 1988) em que se consegue realizar um aprendizado em tempo de execução levando em consideração os movimentos realizados pelo oponente. Uma das vantagens do algoritmo implementado é ser leve, podendo ser executado em tempo real sem exigir muito da máquina que o está executando e conseguir se adaptar a diferentes estilos de jogo, além de ser possível de modificar o que foi aprendido caso o oponente mude de comportamento.

1.1 Organização da Monografia

No Capítulo 2 será discutido o histórico de jogos como campo de testes para novos métodos e técnicas de inteligências artificiais juntamente com trabalhos mais recentes que tiveram jogos eletrônicos como foco de suas pesquisas. Após isto será apresentado a técnica de TLD que fora usada como inspiração para o desenvolvimento da inteligência artificial implementada. No Capítulo 3 será mostrado o ambiente FightingICE que foi utilizado para a implementação da inteligência artificial e também que fora usado na competição. No Capítulo 4 é apresentado o funcionamento do algoritmo desenvolvido juntamente com os resultados obtidos na competição e os trabalhos futuros que podem ser realizados para se melhorar o algoritmo. Por último se tem o Capítulo 5 resumindo o que se foi atingido com esta pesquisa tendo em vista a performasse na competição e o funcionamento da inteligência artificial implementada.

2 Aprendizado em Jogos

Desde os primórdios da inteligência artificial jogos são utilizados para se poder testar a eficácia de algoritmos devido a possibilitar uma fácil comparação entre eles e um ambiente controlado para testes. Os jogos mais tradicionais como Xadrez(SHANNON, 1950), Go(LEE et al, 2009) e Damas(FOGEL, 2001) são utilizados até hoje para se testar novas técnicas de IA e bons resultados já foram obtidos nestes testes.

Os primeiros algoritmos de inteligência artificial utilizados em jogos tiveram como base a utilização de árvores de decisão, em que se observa estados futuros a partir do estado atual para se escolher qual jogada pode resultar em um maior ganho ou mais chances de vitória. Um exemplo de algoritmo clássico desta categoria é o MinMax. Com o passar dos anos os algoritmos e técnicas utilizadas foram se aperfeiçoando de modo que hoje em dia é possível um computador derrotar até mesmo os melhores jogadores do mundo em alguns jogos clássicos como Xadrez, apesar de outros jogos como Go ainda proporcionarem um maior desafio para a área.

Com o advento de computadores mais baratos e pessoais a ideia da utilização deles para o entretenimento levou a criação e ao desenvolvimento de jogos digitais, o que proporcionou a criação de diversos novos tipos de jogos. Dentre eles os jogos de tempo real se popularizaram, estes são jogos em que todos os jogadores estão jogando ao mesmo tempo e não alternando entre turnos em que apenas um pode realizar jogadas. Apesar de estes avanços e novidades o uso de técnicas de inteligência artificial em jogos digitais se mostrou limitada a utilização de máquinas de estados finitos. Uma máquina de estados finitos é definida como tendo estados e transições ligando entre eles, uma transição é realizada dependendo do estado atual do jogo.

Este tipo de técnica é limitada devido a sua falta de capacidade em se adaptar e aprender a novos comportamentos levando isto a dois problemas que se mostram em vários jogos comerciais, inimigos que deveriam ser difíceis acabando sendo derrotados facilmente caso algum jogador jogue de um modo não previsto pelo programador e ao problema de que após se descobrir como derrotar uma máquina de estados finitos ao se deparar

com ela novamente basta utilizar a mesma estrategia que a vitória será obtida. Uma das principais razões para não se utilizar novas técnicas de inteligência artificial em jogos digitais está no fato de que a maioria dos jogos são lançados comercialmente e é visado que eles gerem lucro, sendo assim as empresas dedicadas a publicar os jogos ficam receosas a utilização de técnicas mais avançadas com medo de que elas apresentem comportamentos não previstos, fazendo com que as pessoas não gostem do jogo e consequentemente não o comprem.(SPRONCK et al, 2006)

Temos que estes problemas são intensificados mais ainda em jogos em que diferentemente de jogos tradicionais como Xadrez ou Damas em que a parte matemática do jogo é mais a importante. Nestes jogos temos que a personalidade do oponente se mostra mais relevante e que uma dada ação não pode ser considerada totalmente correta pois a vantagem obtida por usar ela depende de como o oponente estárá reagindo.

> "These games reward intuition, good guessing, and the ability to get into your opponents mind, rather than mastery of abstract game concepts. Hence, the static AI algorithms that are most prevalent tend to do even worse in these games than in others." (RICCIARDI AND THILL, 2008)

Até mesmo as Inteligências Artificiais estáticas mais complexas acabam tendo algum "buraco"em sua programação, não previsto pelo seu programador, que pode ser usado para a derrotar facilmente assim retirando parte da diversão do jogo. Como o propósito de videogames é divertir, quem está jogando, a inteligência artificial deve ser capaz de apresentar uma variedade de comportamentos diferentes, além disto ela não deve ser muito difícil de se derrotar, idealmente ela deve ser capaz de ajustar sua dificuldade a habilidade do jogador. Devido ao fato de caso se tenha uma inteligência artificial capaz de derrotar o jogador, é fácil diminuir a eficiência dela para que a dificuldade fique ajustada a habilidade dele. O estudo teve como foco apenas criar uma inteligência artificial que consiga jogar bem o jogo.

Jogos de luta oferecem dois fatores que os diferenciam de outros tipos de jogos estudados tradicionalmente no campo de inteligência artificial.

O primeiro fator diferencial é que os jogos de luta são jogados em tempo real e não em turnos. Deste modo, caso um algoritmo demore demais para escolher uma ação a

ser tomada pode ser que quando ela for executada o estado do jogo já esteja diferente ao ponto de que ela pode não ser mais vantajosa. Sendo assim, a velocidade de processamento do algoritmo se torna um fator importante.

O segundo fator está no fato de jogos de luta serem jogos nos quais para que uma ação seja classificada como vantajosa ou não, dependerá de qual será a próxima ação de seu oponente sendo que jogos de luta são jogos de informação imperfeita. Assim, para uma ação ser classificada como vantajosa ela depende da "personalidade" de seu oponente, diferentemente de jogos clássicos como Xadrez e GO em que o fator matemático do jogo é muito maior do que a individualidade dos jogadores.

Para a comparação de técnicas diferentes competições bem projetadas são o melhor modo de se testar Inteligências Artificiais para se determinar quais funcionam melhor devido a em jogos normalmente inteligências artificiais interessantes se saírem melhor que inteligências artificiais competitivas.(LUCAS, 2008)

> "There currently seems to be very little theoretical research that can help us here, and a great need for empirical investigation."(LUCAS, 2008)

> "In summary, much experimentation and perhaps even rather novel techniques may be necessary to get best performance from CI methods. This makes the area very interesting and challenging to research. here, and a great need for empirical investigation."(LUCAS, 2008)

Na academia nós temos alguns trabalhos que mostram técnicas mais avançadas para o uso de inteligência artificial em jogos digitais, dentre elas temos o uso de "dynamic scripting", "temporal difference learning"e computação evolucionista.

Foi feita uma comparação extensiva entre o algoritmo de *temporal difference learning* e computação evolucionista em diferentes jogos em (LUCAS, 2008). Foram realizados testes em jogos como o "mountain car problem", "Othelo"e um simulador de carros de corrida simplificado. Em todos os jogos se pode observar que o *temporal difference learning* conseguiu aprender mais rapidamente que o algoritmo evolucionista, porém após tempo suficiente o algoritmo evolucionista conseguiu resultados um pouco melhores que o *temporal difference learning*.

A técnica de "dynamic scripting"fora criada por (SPRONCK et al, 2003), ela pode ser descrita como um conjunto de regras que são usadas para se tomar a decisão de qual ação deve ser tomada. Cada regra tem um peso associado que indica qual a probabilidade dela ser escolhida, estes pesos são modificados durante uma partida. Esta técnica se inspirou no aprendizado por reforço porém com algumas mudanças para que se possa ser mais eficiente.

Na tese de mestrado de (PONSEN, 2004) o autor realiza a implementação de um algoritmo evolucionista e de "dynamic scripting"em uma IDE chamada "Stratagus"que fora feita para o teste de inteligência artificial em jogos de estratégia me tempo real(conhecidos como RTS). Um RTS normalmente tem como objetivo criar um exército para destruir a base de seu oponente ou o comandante dele, para se criar exércitos e construções é necessário extrair recursos naturais do ambiente. O algoritmo evolucionista se mostrou adaptar rapidamente as estratégias estáticas utilizadas contra ele, porém demorando várias partidas para descobrir como seu oponente jogava. O "dynamic scripting"mostrou que algumas mudanças nas regras utilizadas podem resultar em uma melhora significativa na eficiência da inteligência artificial, porém ele mostrou não conseguir se adaptar tão rapidamente quanto seria necessário para se vencer em certos casos em que se precisa utilizar uma regra que demore para ser ativada.

No artigo (SPRONCK et al, 2006) é apresentado uma técnica de inteligência artificial adaptativa chamada de "dynamic scripting". Sendo que ela funciona tendo como base, em sua memória, vários arquétipos de jogadores, cada um com seu estilo de jogar e estratégias de contra medida para cada um destes tipos de estratégia. Ao longo da partida a IA vai analisando seu oponente até que ela possa enquadrar ele em um dos arquétipos conhecidos por ela e então ela realiza sua contra medida conhecida para este arquétipo. Neste mesmo artigo é dito quatro requisitos funcionais e quatro requisitos computacionais que devem ser seguidos para se ter uma boa inteligência artificial online em um jogo. Estes requisitos foram obtidos após tanto uma análise cientifica de jogos digitais quanto com conversas e entrevistas com profissionais da área.

Requisitos Computacionais:

1. Velocidade. A inteligência artificial deve ser rápida, pois ela é computada durante

o jogo.

2. Efetividade. Ela deve mostrar bons resultados durante o processo de aprendizado.

3. Robustes. A inteligência artificial deve ser robusta devido a aleatoriedade inerente em todo jogo.

4. Eficiência. Ela deve ser capaz de conseguir aprender com as oportunidades limitadas de aprendizado presentes em uma partida.

Requisitos funcionais:

1. Claridade. Deve apresentar resultados que sejam facilmente interpretados.

2. Variedade. Apresentar comportamentos variados para torná=la mais divertida e imprevisível.

3. Consistência. Deve apresentar uma baixa variância na quantidade de oportunidades necessárias para um bom aprendizado.

4. Escalabilidade. Deve ser capaz de ter seu nível de dificuldade escalável para ficar próximo ao nível do oponente.

Na próxima seção será discutido com mais detalhe o funcionamento do algoritmo "temporal differençe learning", sendo ele o algoritmo encontrado na literatura que mais se assemelha com o algoritmo desenvolvido neste trabalho.

2.1 TDL–*Temporal Difference Learning*

"Temporal difference learning"é uma classe de procedimentos incrementais especializados em predição que utilizam experiências passadas combinadas com um sistema não conhecido completamente para predizer o seu futuro comportamento (SUTTON, 1988). Para problemas do mundo real métodos de Temporal Difference(diferença temporal) conseguem resultados melhores utilizando menos memória e menos processamento que métodos tradicionais.

Este método tem como objetivo aprender a predizer. Predição é uma das formas mais básicas de aprendizado, uma das vantagens do aprendizado de predição é que ela pode ser realizada sem um supervisor.

"Por exemplo, suponha um meteorologista tentando predizer em cada dia da semana se vai ou não chover no sábado. O método tradicional é para comparar cada predição com o resultado da chuva no sábado. Um método TD(temporal difference), por outro lado, compara a predição de cada dia com a feita no próximo dia(se foi feita uma predição com 50% de chance de chuva na segunda feira e 75% de chance de chuva na terça-feira então um método TD aumenta as predições para dias similares a segunda feira, enquanto que um método tradicional aumentaria ou diminuiria dependendo apenas do resultado de sábado"(TRADUÇÃO LIVRE de (SUTTON, 1988)).

Normalmente os pesos para cada valor recebido é o mesmo na hora de realizar a predição, porém pode ser que os valores mais recentes sejam mais importantes que os antigos, ou em que os antigos não possam mais ser usados de forma confiável para a predição, para isto existe uma "família"de TDs chamada de TD(lambda) em que lambda é um valor real entre 0 e 1 que serve para determinar o quanto é o decaimento da importância dos valores obtidos em tempos passados.

2.1.1 Vantagens de TDL em relação a outros métodos de aprendizado por reforço

Devido ao TDL utilizar a diferença entre os estados e não somente o resultado obtido a partir da predição ele pode conseguir uma predição mais rápida e mais precisa do que métodos de predição supervisionados. Um exemplo para isto seria um caso em que se tem um jogo, durante este jogo entramos em um estado novo em que estamos aprendendo mais sobre ele, sabemos que neste estado vamos para um outro estado mais estudado previamente em que existe uma maior chance de cair numa derrota do que numa vitória. Se ao desenrolar do jogo acontecer o improvável e resultar numa vitória, um método de aprendizado supervisionado colocaria este estado novo marcado com um estado vantajoso, enquanto isto o TDL seria capaz de perceber que mesmo ele tendo levado a um a vitória ele não é vantajoso devido a ter uma maior probabilidade de se levar a uma derrota.

Uma das outras vantagens de TDLs em relação a aprendizados supervisionados está em relação a casos em que se está querendo aprender mais sobre um estado dinâmico. Um estado dinâmico em um sistema é um estado que evolui com o passar do tempo.

3 O ambiente *FightingICE*

A implementação e os testes da IA desenvolvida foram feitos utilizando o ambiente FightingICE. Desenvolvido por pesquisadores do "Intelligent Computer Entertainment Lab" da universidade Ritsumeikan, Japão. Eles desenvolveram este ambiente com as seguintes motivações: encorajar estudantes a trabalharem com inteligência computacional, desenvolver algoritmos de IA que envolvam aplicações em tempo real, comparar e testar algoritmos e técnicas, criação de IA que sejam fortes e se pareçam com seres humanos. O ambiente possui uma visualização gráfica das lutas em tempo real, enquanto elas ocorrem e, além disso, ela também grava vídeos de "replay" das lutas realizadas. Também é possível que um humano lute no lugar de uma IA.

A plataforma foi criada para que as IAs sejam programadas utilizando a linguagem JAVA, proporcionar um desenvolvimento flexível de modo que se possa implementar vários algoritmos diferentes e que se possa aprender como o oponente joga em tempo real.

As IAs recebem a cada frame informações detalhadas de seu oponente e também enviam comandos para serem executados do mesmo modo que um humano faria se estivesse jogando, permitindo que as IAs tenham as mesmas informações que um humano teria se estivesse jogando. O jogo funciona a 60 frames por segundo, sendo que cada IA recebe a informação do estado em que o jogo estava a 15 frames atrás, este atraso existe para adicionar mais riscos para estratégias defensivas e para simular o tempo de reação que humanos tem quando estão jogando. Caso uma IA demore mais que um frame para enviar qual comando será executado e acabe demorando mais que 1 frame(16.666 milissegundos)o jogo assume que ela não quer realizar nenhuma ação neste frame, sendo assim é interessante que a IA seja capaz de processar em um tempo menor que 1 frame para não ficar parada sem fazer nada.

Existem quatro personagens distintos no FightingICE, sendo que um deles usado foi na competição 1C e os outros três na competição 3C. Por questões de balanceamento dos personagens não se espera realizar uma luta entre o personagem da categoria 1C com um personagem da categoria 3C. Um ataque tem no total três fases distintas sendo elas a

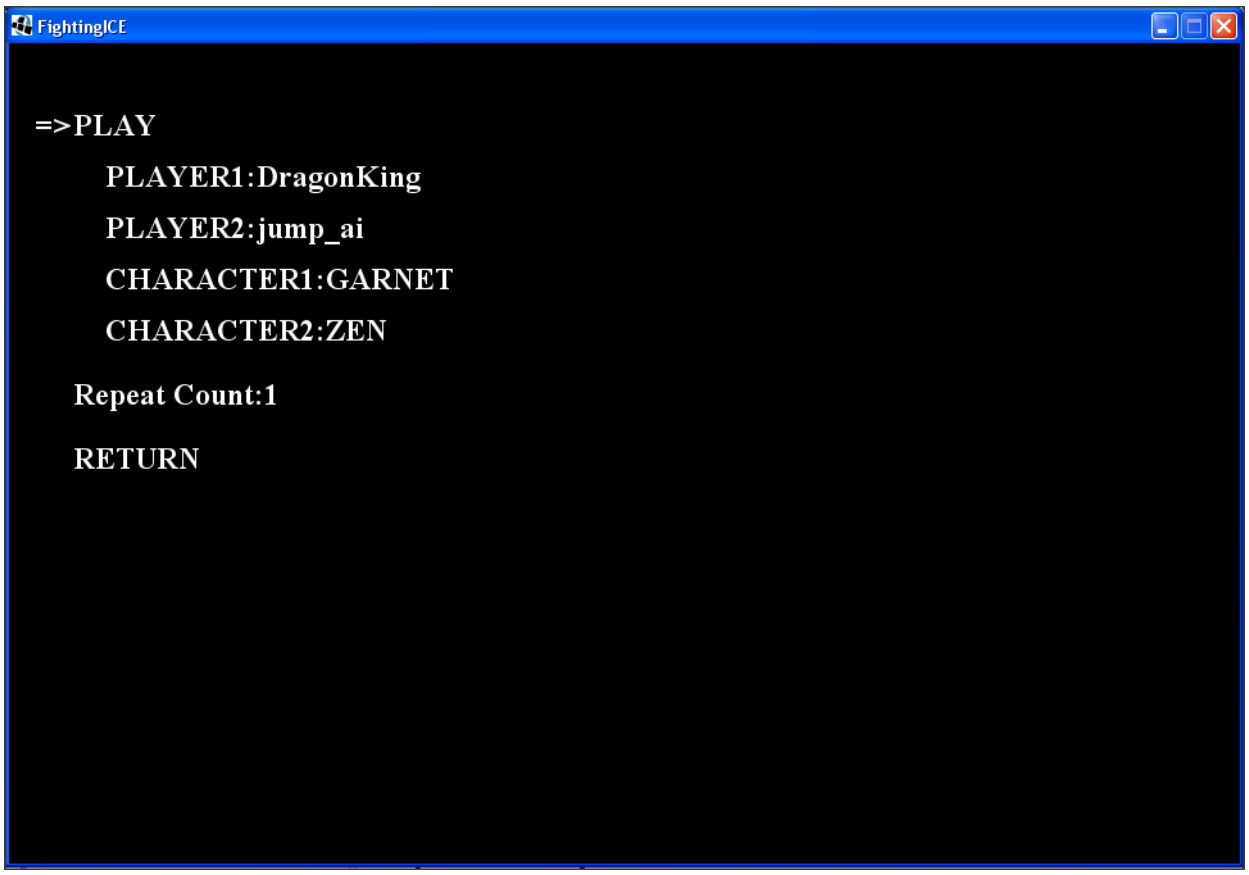

Figura 3.1: Tela de selecao de controladores e personagens.

de “startup”, “active”, “recovery” e há alguns frames durante a fase de “recovery” em que eles podem ser classificados também como “cancelable”. Estas fases estão demonstradas na figura 3.2 retirada de (LU et al, 2013).

A fase de “startup” é a primeira fase a se entrar após um comando ser confirmado, nesta fase o personagem começa a realizar seu ataque, porém não causa nenhum dano a seu oponente, nenhum outro comando pode ser confirmado durante esta fase. A fase de “active” é a fase em que se pode atingir o oponente com um ataque. A “hitbox” do ataque é criada. A fase de “recovery” é a fase em que o personagem não pode mais atingir seu oponente, a “hitbox” do ataque é destruída, e está voltando para o estado normal. Alguns frames neste estado podem ser classificados também como “cancelable”, durante estes frames caso seja enviado outro comando que possa cancelar o atual ele é confirmado.

A detecção de ataques funciona por meio do sistema de “hitbox”, em que o ambiente detecta se um ataque acertou ou não o oponente. Este ambiente fora usado na

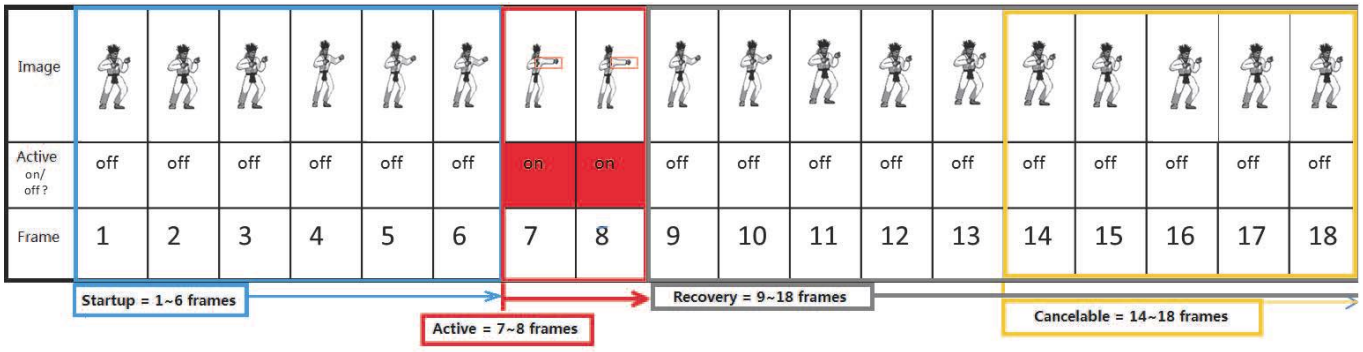

Figura 3.2: Fases de Ataque. Figura retirada de (LU et al, 2013)

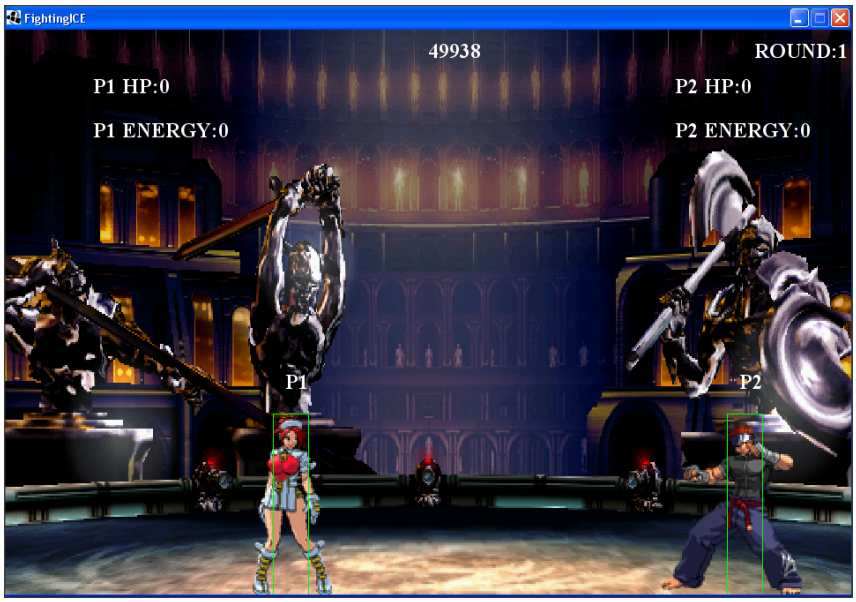

Figura 3.3: Imagem retirada de uma luta entre GARNET(personagem da esquerda) e ZEN(personagem da direita)

competição de jogo de luta na conferência CIG nos anos de 2013 e 2014.

3.1 Considerações sobre o *FightingICE* e Aprendizado em Jogos

O ambiente FightingICE se mostra um bom modo de realizar comparações entre diferentes algoritmos e estratégias de aprendizado offline e online, devido a ele simplificar elementos de programação em jogos que não são muito relevantes na pesquisa de inteligências

artificiais novas. Uma outra vantagem dele é já ter disponível alguns algoritmos de IA feitos em anos anteriores, disponibilizadas com seu código fonte e com arquivo pronto para serem executadas.

No próximo capítulo será descrito a inteligência artificial desenvolvida utilizando o FightingICE e será relatado como ela se comportou em um ambiente competitivo.

4 *DragonKing*: uma IA baseada em TDL

Foi desenvolvido uma IA utilizando o ambiente FightingICE para se aplicar os conceitos estudados. A IA desenvolvida fora enviada para a competição de IA de jogos de luta da "IEEE Conference on Computational Intelligence and Games". O nome da IA foi escolhido como DragonKing.

4.1 Funcionamento Básico

A IA desenvolvida tem como estratégia base escolher seu próximo ataque baseando-se nos últimos ataques de seu oponente, procurando encontrar uma relação entre qual ataque o oponente usa em quais distâncias relativas entre o personagem da IA e o personagem do oponente.

Para cada ataque que pode ser utilizado pelo oponente existe uma lista correspondente de ataques que podem ser utilizados contra este ataque de modo que tenham uma maior probabilidade de se obter uma vantagem sobre o oponente, seja o atingindo com um ataque ou se esquivando do ataque utilizado por ele. A lista de possíveis contra ataques tem eles ordenados do com mais probabilidade de atribuir vantagem para o com menor probabilidade, tais ataques foram encontrados ao se observar os dados referentes aos ataques dos personagens do jogo, que são disponibilizados pelos desenvolvedores da FightingICE, e ao se fazer testes comparando os ataques um a um para se ver qual obtém vantagem contra qual.

A predição de quais ataques o oponente utilizará é baseada na heurística de que jogadores tem um subconjunto de ataques favoritos dentre todos os ataques existentes e que eles tendem a utilizar estes ataques favoritos com mais frequência. Sendo assim ao se gravar quais foram os ataques utilizados por um jogador pode-se descobrir qual é o seu subconjunto de ataques favoritos e assim poder escolher com mais precisão uma melhor contra medida.

Outro fator que é importante na predição é em quais distâncias que o oponente

utilizou quais ataques. Esta informação possibilita reduzir ainda mais quais ataques se espera que o oponente utilize para um subconjunto dos ataques favoritos. Esta heurística se baseia no fato de que certos ataques não provem muito benefício ao serem utilizados em certas distâncias. Certos ataques podem prover benefícios iguais em um grande escopo de distâncias, porém devido ao modo de lutar do oponente eles podem ser utilizados apenas em certas distâncias.

Durante o período de desenvolvimento da AI foram-se utilizados como oponentes de teste a AI campeã de 2013 da competição de jogos de luta da CIG além das AIs disponibilizadas para teste desenvolvidas pelos organizadores da competição.

4.1.1 Análise do oponente

Sempre que o oponente utiliza um ataque, este é registrado para uso posterior durante a predição. Para cada ataque possível existe um contador que indica quantas vezes este ataque fora utilizado durante a partida. Além disto, também existe um vetor circular para cada ataque que armazena a posição das últimas vezes que o ataque fora utilizado. Após atualizar estes dois vetores é então calculado a média das últimas posições utilizadas pelo ataque e logo em seguida seu desvio padrão. Para cada ataque existe uma variável indicando qual é a menor posição relativa que se espera que este ataque seja utilizado pelo oponente e outra variável para registrar qual é a maior posição relativa que se espera que o oponente utilize o ataque. A menor posição relativa é calculada subtraindo da média o desvio padrão, e a maior posição é calculada adicionando o desvio padrão ao valor da média.

4.1.2 Escolha da contra medida

A predição dos ataques do oponente funciona primeiro analisando em qual distancia relativa o personagem da IA está em relação ao personagem do oponente. Após isso, ele consulta a lista de distâncias que se espera que o oponente utilize cada ataque para descobrir para quais ataques existe uma probabilidade de serem utilizados e de quanto é esta probabilidade.

A probabilidade de utilização atribuída para cada ataque é dada como a quan-

tidade de vezes que este ataque fora utilizado dividida pela quantidade de vezes que os ataques com probabilidade de serem utilizados nesta distância foram usados. Após cada ataque ter sua probabilidade de utilização atribuída é escolhido dentre eles um aleatoriamente para ser tomado como o próximo ataque que o oponente utilizará. A decisão de se escolher um ataque aleatoriamente deste grupo e não necessariamente o ataque com maior probabilidade está no fato de que sempre escolhendo o ataque com mais probabilidade faria com que a AI ficasse mais previsível e assim ferindo o requisito funcional de previsibilidade(SPRONCK et al, 2006).

Como os valores que indicam quais ataques o oponente utilizará e qual a probabilidade de utilização de cada um temos que a AI consegue se adaptar em tempo real ao modo de luta de seu oponente sendo que para dois oponentes distintos ela terá um comportamento específico para lidar com cada um deles, caso também o oponente mude de tática durante a luta é possível também a AI se adaptar a essa mudança durante a luta.

4.1.3 Pseudo Código

Algoritmo 1: Funcionamento do DragonKing.

Entrada: Estado atual do jogo
Saída: Qual movimento deve ser realizado

1 **início**
2 **Observa qual é o movimento atualmente usado pelo oponente**;
3 **Atualiza número de vezes que o movimento foi utilizado**;
4 **Calcula a média e o desvio padrão das distâncias em que o oponente estava quanto utilizou este movimento**;
5 **Atualiza as distâncias máximas e mínimas que se acredita que o oponente utiliza este movimento**;
6 **Distribui um valor de 0 a 1 para cada movimento que já foi utilizado pelo oponente na distância em que se está agora dela. Sendo este peso correspondente a porcentagem de vezes que o oponente utilizou o movimento em relação a soma da quantidade de vezes que ele utilizou todos os movimentos que se acredita serem usados nesta distância**;
7 **Escolhe um dos ataques com pesos atribuídos sendo a probabilidade dele ser escolhido igual ao seu peso**;
8 **Escolhe uma das contramedidas para o movimento escolhido e a executa**;
9 **fim**

4.2 Resultados da competição no CIG–2014

A competição de jogos de luta da CIG 2014 aconteceu em duas etapas, uma de pré torneio e uma do torneio em si. A etapa de pré torneio foi realizada em uma categoria na qual havia apenas um personagem disponível(*KFC*), após a execução dela os resultados foram disponibilizados junto com os *replays* das partidas para os competidores poderem ter um *feedback* de como suas AIs foram no pré torneio. Após a realização do pré torneio foi adicionado uma nova categoria, juntamente com uma melhora gráfica no ambiente FightingICE, em que nela competem três personagens diferentes(*Zen*, *Lud*, *Garnet*), esta segunda categoria foi intitulada 3C enquanto que a categoria anterior foi intitulada 1C pelos organizadores da competição.

A AI desenvolvida foi modificada levemente para poder ser enviada também na categoria 3C. As mudanças foram apenas em relação a alterações na lista dos ataques que existem e para ela saber contra qual personagem está lutando, tendo em vista que cada personagem tem suas peculiaridades. O personagem escolhido para ser o controlado pela AI foi a *Garnet*.

Durante o pré torneio houve 14 participantes no total, de 8 países diferentes (Brasil, Japão, Tailândia, Alemanha, Taiwan, Estados Unidos da América, Inglaterra, Israel). Dentre eles nós ficamos em quinto lugar, sendo que o campeão do ano passado ficou em sexto lugar.

Para cada luta é atribuída uma pontuação para cada AI envolvida segundo a seguinte fórmula:

$$\frac{100 \times \textit{Vida do Oponente}}{(\textit{Vida da AI} + \textit{Vida do Oponente})} \tag{4.1}$$

Após todas as lutas serem realizadas as pontuações de cada AI são somadas e quanto maior for a sua pontuação total, melhor será a sua colocação no ranking.

O pré torneio foi realizado em agosto de 2014 e seus resultados foram enviados para os competidores por meio de e-mail. os resultados de todas as AIs envolvidas no pré-torneio podem ser vistos na Tabela 4.1 a seguir..

Os códigos fonte das AIs envolvidas no pré torneio não foram disponibilizadas

AI	Midterm Ranking	Members	Affiliation
T	7	Phakhawat Sarakit	TAIST-Tokyo Tech
thunder	1	Eita Aoki	Nagoya University
PDSAI	11	Passan Julsaksrisakul	Chulalongkorn University
`Seal_Switch`	10	Komsorn Prammannee	Chulalongkorn University
Code Monkey AI	4	Kevin Majohrzak	TU Dortmund
DragonKing	6	Renan Motta Goulart, Guilherme Albuquerque Pinto	Universidade Federal de Juiz de Fora
JIN AI	9	Nattapon Werayawarangura	Chulalongkorn University
BOT AI	3	Eakasit Tangmunchittam	Chulalongkorn University
Agent Jhu	5	Jhu-Lin CHen, Tsung-Che Chiang	National Taiwan Normal University
AT Team	2	Thanarat jaruenying, Phuripat Prasittipap, Worawat Choensawat	Bangkok University
Throwing Loop	14	Thanat Damrongwatanapokin	Chulalongkorn University
evoSektor	8	Avi Levin, Aviad Haded	BenGurion University of the Negev
Little Fuzzy	13	Perry Monschau	University of Essex
Ninja Knightro	12	Joshua Haley and Jimmy Wong	University of Central Florida

Tabela 4.1: Rankings do Pré-Torneio.

porém fora disponibilizado os replays das lutas para que os competidores pudessem analisar melhor o comportamento de sua AI.

Durante o período entre o pré torneio e o torneio foi feito uma atualização na IDE para que fosse adicionado três novos personagens que compõe a categoria 3C. Sendo assim foram feitos dois torneios durante a competição não sendo obrigatório participar de ambos. Os rankings dos dois torneios realizados durante a CIG2014 estão nas tabelas 4.2 e 4.3 .

Junto com os rankings também foi divulgado um resultado detalhado das pontuações das lutas realizadas com *replays* das lutas. Além disto também foi divulgado arquivos ".jar"das IAs para que se os mesmos sejam utilizados na IDE. A divulgação do código fonte das IAs do torneio foi feita apenas para as equipes que quiseram disponibilizar o código de suas IA, alguns dos participantes que ficaram nas posições mais altas preferiram não divulgar seus códigos. O código fonte da IA desenvolvida neste trabalho pode ser encontrada no site da competição juntamente com o código fonte das AIs dos outros participantes que as deixaram serem divulgadas.

AI	Ranking
CodeMonkey	1
VS	2
T1c	3
LittleFuzzy	4
PnumaSON_AI	5
thunder_final	6
DragonKing1C	7
ATteam	8
SomJang	9
ThrowLooper	10

Tabela 4.2: Rankings do Torneio da categoria 1C.

AI	Character	Ranking
T3c	LUD	1
ATteam2	ZEN	2
DragonKing3C	GARNET	3
PnumaSON3C_AI	LUD	4
PasanAI2	GARNET	5
LittleFuzzy	ZEN	6
SejongFighter	GARNET	7
Seal_Switch	LUD	8

Tabela 4.3: Rankings do Torneio da categoria 3C.

4.3 Trabalhos Futuros

Após os resultados da competição saírem pode ser visto os pontos fracos e fortes da inteligência artificial e do algoritmo implementado. Se pode perceber que ela consegue se adaptar rapidamente e descobrir bem como os inimigos lutam, porém em alguns casos observou-se que ela mesmo sabendo como o inimigo lutava não conseguiu obter uma vantagem devido às ações utilizados como contra medida não acertarem o oponente ou não causarem tanto dano quanto estava sendo recebido.

O motivo para isto se deve a dois fatores, primeiro o fato de as contra medidas terem sido escolhidas realizando testes e observando suas estatísticas, porém não foi realizado testes exaustivos para se ter uma idéia perfeita do quão bom era uma contra medida devido a isto não fazer parte do foco da pesquisa que era criar um algoritmo adaptativo.

O outro fator se deve ao algoritmo levar em consideração quais ataques o oponente

vai utilizar separadamente e não levando em conta o conjunto como um todo, sendo assim pode ser que uma contra medida pode ser boa com um ataque específico mas ruim contra o resto do conjunto de ataques previstos, pode ser que haja uma contra medida que não seja a melhor contra nenhum ataque específico, porém seja a melhor ao se levar em consideração o conjunto como um todo. Por exemplo, suponhamos que o conjunto de ataques previstos que o oponente pode usar é composto por dois ataques "A"e "B"e a probabilidade de utilização de cada um é de 50%, suponhamos que temos uma contra medida "C"que tem 80% de chance de se obter vantagem contra o ataque "A"mas apenas 20% de chance contra o ataque "B", suponhamos que temos uma contra medida "D"para com 20% de probabilidade de vantagem para "A"e com 80% de probabilidade de vantagem para "B". Temos que para o algoritmo implementado nós obtemos um valor esperado de apenas 50% de chance de obter vantagem, mesmo utilizando contra medidas efetivas com 80% de chance de se obter vantagem contra estes dois ataques individualmente. Porém se tivermos uma contra medida "E"que tem 60% de probabilidade de se obter vantagem contra os ataques "A"e "B", ao utilizar ela pode ser uma estratégia mais eficiente, mesmo ela não sendo a melhor para cada um individualmente.

Se planeja futuramente para as próximas competições que utilizem a IDE FightingICE entrar nestas competições e melhorar a escolha das contra medidas para que se possa realizar a adaptação não somente em relação a se descobrir quais ataques o oponente utilizará mas também qual contra medida é mais eficaz levando em conta todo o seu conjunto de possíveis ações.

5 Considerações Finais

Nesta pesquisa foi implementado uma inteligência artificial em java para jogos de luta que consegue se adaptar aos seus oponentes durante a partida e sendo leve o bastante para poder realizar isto em tempo real, sessenta vezes por segundo. Temos que os resultados obtidos na competição, em que ela foi enviada, foram bons ficando em terceiro lugar em uma das categorias, derrotando muitas inteligências artificiais que não apresentam nenhuma ou quase nenhuma forma de adaptação, além de outras que tinham uma habilidade de adaptação e se mostrando ser capaz de se adaptar aos seus oponentes, de prever qual subconjunto de ataques eles utilizam e qual será o próximo ataque realizado mesmo em casos em que se resultou em uma derrota.

Se planeja em trabalhos futuros melhorar a principal fraqueza da inteligência artificial implementada que não é relacionado a predição de ataques mas sim saber quais são as contramedidas ótimas para cada oponente específico.

Referências Bibliográficas

FOGEL, D. **Blondie24: Playing at the Edge of AI**. The Morgan Kaufmann Series in Artificial Intelligence. Morgan Kaufmann, 2001.

GRAEPEL, T.; HERBRICH, R. ; GOLD, J. **Learning to fight**. In: Microsofth Research Ltd, 2004.

LEE, C.-S.; WANG, M-H nad CHASLOT, G.; J-B, H.; RIMMEL, A.; TEYTAND, O.; TSAI, S.-R.; HSU, S.-C. ; T-P, H. **The computational intelligence of mogo revealed in taiwan's computer go tournamentsfighting game artificial intelligence competition platform**. p. 73–89. IEEE Transactions on Computational Intelligence and AI in Games, 2009.

LU, F.; YAMAMOTO, K.; NOMURA, L.; MIZUNO, S.; LEE, Y. ; THAWONMAS, R. **Fighting game artificial intelligence competition platform**. Tokyo, Japan, 2013. IEEE Global Conference on Consumer Electronics.

LUCAS, S. Computational intelligence and games: Challenges and opportunities. **International Journal of Automation and Computing**, v.05, p. 45–57, 2008.

PONSEN, M. **Improving adaptative game ai with evolutionary learning**. Delft, Países Baixos, Dissertação de Mestrado - .

RICCIARDI, A.; THILL, P. Adaptative ai for fighting games. 2008.

SHANNON, C. **Programming a computer for playing chess**. In: Philosophical Magazine, volume 41, 1950.

SPRONCK, P.; SPRINKHUIZEN-KUYPER, I. ; POSTMA, E. **Online adaptation of game opponent ai in simulation and in practice**. In: Mehdi, Q.; Gough, N., editors, Proceedings of the 4th International Conference on Intelligent Games and Simulation (GAME-ON 2004), volume 4511, p. 93–100, 2003.

SPRONCK, P.; PONSEN, M.; SPRINKHUISEN-KUYPER, I. ; POSTMA, E. Adaptive game ai with dynamic scripting. **Springer Science + Business Media, LCC**, v.63, p. 217–248, 2006.

SUTTON, R. Learning to predict by the methods of temporal differences. **Machine Learning**, v.3, p. 9–44, 1988.

A – Código fonte da AI

A.1 Codigo Fonta da AI(DragonKing) no pré-torneio

```
1  import commandcenter.CommandCenter;
2
3  import structs.FrameData;
4  import structs.GameData;
5  import structs.Key;
6  import gameInterface.AIInterface;
7
8      //DEVELOPED BY : Renan Motta Goulart, Guilherme Albuquerque
           Pinto
9      // Universidade Federal de Juiz de Fora, 2014
10     //CONTACT INFO : raikoalihara@hotmail.com, renan.motta@ice.
           ufjf.br, guilherme.pinto@gmail.com
11
12 public class DragonKing implements AIInterface {
13
14     private boolean playerNumber;
15     private Key inputKey;
16     private FrameData frameData;
17     private CommandCenter cc;
18     /*
19     //probabilidade dos ataques
20         //throw
21       private int EP_Throw_A; --> 0
22       private int EP_Throw_B; --> 1
23     //
24     //High
25       private int EP_Stand_F_D_DFA; --> 2
26       private int EP_Stand_D_DF_FC; --> 3
27       private int EP_Stand_A; --> 4
28       private int EP_Stand_B; --> 5
29       private int EP_Stand_FA; --> 6
30       private int EP_Stand_D_DF_FA; --> 7
31       private int EP_Stand_D_DF_FB; --> 8
32       private int EP_Stand_D_DB_BB; --> 9
33     //
34     //Middle
35       private int EP_Stand_F_D_DFB; --> 10
36       private int EP_Stand_FB; --> 11
37       private int EP_Stand_D_DB_BA; --> 12
38     //
39     //Low
40       private int EP_Crouch_A; --> 13
41       private int EP_Crouch_FA; --> 14
```

```
42        private int EP_Crouch_B; --> 15
43        private int EP_Crouch_FB; --> 16
44      //
45      //total
46       */
47    //variaveis do aprendizado
48      private int i, j;//variavel usada nos for
49      //private int enemy_totalAttacks;//quantidade total de
           ataques utilizados pelo oponente ate agora
50      private int[] EPs = new int[17];//quantidade de vezes que ele
           utilizou os ataques
51      private int[] EAAverage = new int[17];
52      private int[] EAcounter = new int[17];
53      private int[] EAADistances = new int[20*17];
54      private int[] EAStdDeviation = new int[17];
55      private int[] EAmaxDistance = new int[17];//distancia maxima
           que o oponete utilizou os ataques
56      private int[] EAminDistance = new int[17];//distancia minima
           que o oponenten utilizou os ataques
57      private int[] Adistance = {150, 150, 250, 240, 220, 240, 240,
           300, 300, 300, 240, 240, 128, 220, 250, 240, 250};//new
           int[17];
58      private String[] calls = {"4 _ A", "4 _ B", "6 2 3 _ A", "2 3
           6 _ C", "A", "B", "6 _ A", "2 3 6 _ A", "2 3 6 _ B", "2 1
          4 _ B", "6 2 3 _ B", "6 _ B", "2 1 4 _ A", "2 _ A", "3 _ A"
          , "2 _ B", "3 _ B"};
59      private String[] calls2 = {"THROW_A", "THROW_B", "
          STAND_F_D_DFA", "EP_STAND_D_DF_FC", "STAND_A", "STAND_B", "
          STAND_FA", "STAND_D_DF_FA", "STAND_D_DF_FB", "STAND_D_DB_BB
          ", "STAND_F_D_DFB", "STAND_FB", "STAND_D_DB_BA", "CROUCH_A"
          , "CROUCH_FA", "CROUCH_B", "CROUCH_FB"};
60      private int probTotal;
61      private int ataqueEscolhido;//ataque que se acredita que o
          oponente vai usar agora
62      private int ataqueEscolhidoContador;//contadr para se desidir
           qual ataque o oponente deve utilizar agora
63      private int tempoFimAtaqueInimigo;//tempo que falata para o
          ataque acabar
64
65      private Boolean mostraBoo = true;
66
67      //private int distanciaAtaqueTeste;
68      //private int distanciaAtaqueTeste2;
69    //
70
71    @Override
72    public int initialize(GameData gameData, boolean playerNumber)
         {
73      this.playerNumber = playerNumber;
74      this.inputKey = new Key();
75      cc = new CommandCenter();
76      frameData = new FrameData();
77      for(i=0;i<17;i++)
```

```
78      {
79        EAmaxDistance[i] = 0;
80        EAminDistance[i] = 9999;
81      }
82      return 0;
83    }
84
85    @Override
86    public void getInformation(FrameData frameData) {
87      this.frameData = frameData;
88      cc.setFrameData(this.frameData, playerNumber);
89    }
90
91    @Override
92    public void processing() {
93      if (!frameData.emptyFlag && frameData.getRemainingTime() > 0)
           //linha obrigat ria segundo a documenta  o
94      {
95        if (cc.getskillFlag())//testa se um skill esta sendo "
             inputado"(n o sei como falar isso em portugu s)
96        {
97          inputKey = cc.getSkillKey();//caso sim entao continua a
               inputar ele
98        }
99        else
100       {
101         if(frameData.getRemainingTime() > 55000)
102         {
103            mostraBoo = true;
104         }
105         if(frameData.getRemainingTime() < 5000 && mostraBoo ==
               true)
106         {
107           System.out.println(" ");
108           System.out.println("1-> Average:"+EAAverage[1]+"
                 StdDeviation: "+EAStdDeviation[1]);
109           System.out.println("8-> Average:"+EAAverage[8]+"
                 StdDeviation: "+EAStdDeviation[8]);
110           System.out.println("9-> Average:"+EAAverage[9]+"
                 StdDeviation: "+EAStdDeviation[9]);
111           System.out.println("10-> Average:"+EAAverage[10]+"
                 StdDeviation: "+EAStdDeviation[10]);
112           System.out.println("12-> Average:"+EAAverage[12]+"
                 StdDeviation: "+EAStdDeviation[12]);
113           System.out.println("3-> Average:"+EAAverage[3]+"
                 StdDeviation: "+EAStdDeviation[3]);
114           mostraBoo = false;
115         }
116         inputKey.empty();//limpa a inputKey(reseta/reinicia ela)
117         cc.skillCancel();//limpa skillData(array que contem a
               sequ cia de botoes para serem apertados) e marca
               skillFlag como false
118
```

```
119         //Espera o inimigo finalizar o ataque dele, ainda da pra
                melhorar esta linha em relacao ao status atual do
                inimigo tambem...
120           if(tempoFimAtaqueInimigo > 0)
121           {
122             tempoFimAtaqueInimigo -= 1;
123             //System.out.println(tempoFimAtaqueInimigo);
124           }
125         //
126         //Descobre qual ataque o inimigo utiliza em relacao a
                distancia
127           if(tempoFimAtaqueInimigo == 0)
128           {
129             for(i=0;i<16;i++)
130             {
131               if(cc.getEnemyCharacter().getAction().name().equals
                      (calls2[i]))
132               {
133                 EPs[i] += 1;
134                 tempoFimAtaqueInimigo = cc.getEnemyCharacter().
                        getRemainingFrame();
135                 //EAAverage[i] = ( EAAverage[i]*(EPs[i]-1)*(9/10)
                        +cc.getDistanceX()*((EPs[i]-1)*(1/10) +1) )/EPs
                        [i];//ultimo ataque vale 10% na nova media
136                 EAADistances[20*i+EAcounter[i]] = cc.getDistanceX
                        ();
137                 EAAverage[i] = (EAAverage[i]*(EPs[i]-1) + cc.
                        getDistanceX() )/EPs[i];
138                 EAStdDeviation[i] = 0;
139                 for(j=0;j<20;j++)
140                 {
141                   EAStdDeviation[i] += (EAADistances[20*i+j]-
                          EAAverage[i])*(EAADistances[20*i+j]-EAAverage
                          [i]);
142                 }
143                 EAStdDeviation[i] = EAStdDeviation[i]/20;
144                 EAStdDeviation[i] =(int) Math.sqrt(EAStdDeviation
                        [i]);
145                 if(EAcounter[i] == 19)
146                   EAcounter[i] = 0;
147                 else
148                   EAcounter[i]+=1;
149                 //EAStdDeviation[i] = ( EAStdDeviation[i]*
                        EAStdDeviation[i]*(EPs[i]-1) + (cc.getDistanceX
                        () - EAAverage[i])*(cc.getDistanceX() -
                        EAAverage[i]) )/EPs[i];
150                 EAmaxDistance[i] = EAAverage[i]+EAStdDeviation[i
                        ];
151                 EAminDistance[i] = EAAverage[i]-EAStdDeviation[i
                        ];
152                 /*
153                 if(cc.getDistanceX() > EAmaxDistance[i])
154                   EAmaxDistance[i]  = cc.getDistanceX();
```

```
155                 if(cc.getDistanceX() < EAminDistance[i])
156                   EAminDistance[i]  = cc.getDistanceX();
157                 */
158               }
159             }
160           }
161         //
162
163         //Procura quais ataques o inimigo pode utilizar nesta
              distancia
164           probTotal = 0;
165           ataqueEscolhido = 0;
166           ataqueEscolhidoContador = 0;
167           for(i=0;i<17;i++)
168           {
169             if(cc.getDistanceX() <= EAmaxDistance[i] && cc.
                  getDistanceX() >= EAminDistance[i])
170               probTotal += EPs[i];
171           }
172           if(probTotal == 0)
173           {
174             ataqueEscolhido = (int) (Math.random()*16);//
                  utilizando uma heiristica bem simples de escolher
                  um ataque randomicamente. Vale a pena melhorar isto
                   em versoes futuras
175             //System.out.println("rand"+ataqueEscolhido);
176           }
177           else
178           {
179             //System.out.println("Aprendizado");
180             ataqueEscolhido = (int) (Math.random()*probTotal);//
                  TESTAR SE ISTO ESTA FUNCIONANDO MESMO, talvez tenha
                   um modo melhor de se realizar isto utilizando a
                  classe "Random"
181             //System.out.println(ataqueEscolhido);
182             for(i=0;i<17;i++)
183             {
184               if(cc.getDistanceX() <= EAmaxDistance[i] && cc.
                    getDistanceX() >= EAminDistance[i])
185               {
186                 ataqueEscolhidoContador += EPs[i];
187                 if(ataqueEscolhidoContador >= ataqueEscolhido)
188                 {
189                   ataqueEscolhido = i;
190                   i = 1000;
191                   //System.out.println(1);
192                 }
193                 /*
194                 if(ataqueEscolhidoContador >= probTotal)//esta
                      heuristica ajuda mais contra algoritmos que se
                      adaptam tambem. Para um algoritmo que nao se
                      adapta seria melhor escolher apenas o ataque
                      com mais probabilidade de ser utilizado
```

```
195                 {
196                   //ataqueEscolhido = i;
197                   i = 2000;//sai do loop
198                   System.out.println(2);
199                 }
200                 */
201               }
202             }
203           }
204           /*
205           if(i >= 2000)//nenhum ataque foi escolhido pois nao se
                 tem informacao do que o oponente tem mais chance de
                 fazer, acontece no inicio do jogo com frequencia
206           {
207             //ataqueEscolhido = -1;//ataque invalido, apenas para
                    testes
208             ataqueEscolhido = (int) Math.random()*16;//utilizando
                    uma heiristica bem simples de escolher um ataque
                   randomicamente. Vale a pena melhorar isto em
                   versoes futuras
209           }
210           */
211         //
212
213         //Decide qual ataque sera utilizado
214           //System.out.println(ataqueEscolhido);
215           if(cc.getDistanceX() >= 300 && cc.getMyEnergy() >= 400)
                 //utiliza o ataque especial
216           {
217             cc.commandCall(calls[3]);
218           }
219           else if(ataqueEscolhido == 0)//7f
220           {
221             if( cc.getDistanceX() <= Adistance[9])
222             {
223               cc.commandCall(calls[9]);
224             }
225             else if( cc.getDistanceX() <= Adistance[11])
226             {
227               cc.commandCall(calls[11]);
228             }
229             else
230               escolhaPadrao();
231           }
232           else if(ataqueEscolhido == 1)//13f
233           {
234             if( cc.getDistanceX() <= Adistance[0])
235             {
236               cc.commandCall(calls[0]);
237             }
238             else if( cc.getDistanceX() <= Adistance[5])
239             {
240               cc.commandCall(calls[5]);
```

```
241             }
242             else if( cc.getDistanceX() <= Adistance[2])
243             {
244               cc.commandCall(calls[2]);
245             }
246             else
247               escolhaPadrao();
248           }
249           else if(ataqueEscolhido == 2)
250           {
251             if( cc.getDistanceX() <= Adistance[9])
252             {
253               cc.commandCall(calls[9]);
254             }
255             else if( cc.getDistanceX() <= Adistance[2])
256             {
257               cc.commandCall("STAND_GUARD");//ataque mais rapido
                     do jogo
258             }
259             else
260               escolhaPadrao();
261           }
262           else if(ataqueEscolhido == 3)//especial
263           {
264             if(cc.getEnemyEnergy() > 300)
265               cc.commandCall(calls[9]);
266             else
267               escolhaPadrao();
268           }
269           else if(ataqueEscolhido == 4)
270           {
271             //cc.commandCall("2 3 6 _ B");
272             if( cc.getDistanceX() <= Adistance[9])
273             {
274               cc.commandCall(calls[9]);
275             }
276             else if( cc.getDistanceX() <= Adistance[16])
277             {
278               cc.commandCall(calls[16]);
279             }
280             else
281               escolhaPadrao();
282           }
283           else if(ataqueEscolhido == 5)//12f
284           {
285             if( cc.getDistanceX() <= Adistance[7])
286               cc.commandCall(calls[7]);
287             else if( cc.getDistanceX() <= Adistance[9])
288               cc.commandCall(calls[9]);
289             else if( cc.getDistanceX() <= Adistance[2])
290               cc.commandCall(calls[2]);
291             else
292               escolhaPadrao();
```

```
293             }
294             else if(ataqueEscolhido == 6)
295             {
296               if( cc.getDistanceX() <= Adistance[7])
297               {
298                 cc.commandCall(calls[7]);
299               }
300               else if( cc.getDistanceX() <= Adistance[9])
301               {
302                 cc.commandCall(calls[9]);
303               }
304               else
305                 escolhaPadrao();
306             }
307             else if(ataqueEscolhido == 7)
308             {
309               if( cc.getDistanceX() <= Adistance[8])
310               {
311                 cc.commandCall(calls[8]);
312               }
313               else if( cc.getDistanceX() <= Adistance[9])
314               {
315                 cc.commandCall(calls[9]);
316               }
317               else
318                 escolhaPadrao();
319             }
320             else if(ataqueEscolhido == 8)//13f
321             {
322               if( cc.getDistanceX() <= Adistance[9])
323               {
324                 cc.commandCall(calls[9]);
325               }
326               else
327                 escolhaPadrao();
328               /*
329               if( cc.getDistanceX() <= Adistance[10])
330               {
331                 cc.commandCall(calls[10]);
332               }
333               else if( cc.getDistanceX() <= Adistance[13])
334               {
335                 cc.commandCall(calls[13]);
336               }
337               else
338                 escolhaPadrao();
339               */
340             }
341             else if(ataqueEscolhido == 9)
342             {
343               //if( cc.getDistanceX() <= Adistance[14])
344               //{
345                 cc.commandCall(calls[9]);
```

```
346             //}
347             //else
348               //escolhaPadrao();
349             /*
350             if( cc.getDistanceX() <= Adistance[13])
351             {
352               cc.commandCall(calls[13]);
353             }
354             else if( cc.getDistanceX() <= Adistance[14])
355             {
356               cc.commandCall(calls[14]);
357             }
358             else
359               escolhaPadrao();
360             */
361             //cc.commandCall(calls[9]);//fireball
362           }
363           else if(ataqueEscolhido == 10)
364           {
365             if( cc.getDistanceX() <= Adistance[9])
366             {
367               cc.commandCall(calls[9]);
368             }
369             else if( cc.getDistanceX() <= Adistance[13])
370             {
371               cc.commandCall(calls[13]);
372             }
373             else if( cc.getDistanceX() <= Adistance[14])
374             {
375               cc.commandCall(calls[14]);
376             }
377             else if( cc.getDistanceX() <= Adistance[8])
378             {
379               cc.commandCall(calls[8]);
380             }
381             else
382               escolhaPadrao();
383             //cc.commandCall("2 3 6 _ B");
384           }
385           else if(ataqueEscolhido == 11)
386           {
387             if( cc.getDistanceX() <= Adistance[15])
388             {
389               cc.commandCall(calls[15]);
390             }
391             else if( cc.getDistanceX() <= Adistance[10])
392             {
393               cc.commandCall(calls[10]);
394             }
395             else if( cc.getDistanceX() <= Adistance[16])
396             {
397               cc.commandCall(calls[16]);
398             }
```

```
399             else
400               escolhaPadrao();
401           }
402           else if(ataqueEscolhido == 12)
403           {
404             if( cc.getDistanceX() <= Adistance[0])
405             {
406               cc.commandCall(calls[0]);
407             }
408             else if( cc.getDistanceX() <= Adistance[8])
409             {
410               cc.commandCall(calls[8]);
411             }
412             else
413               escolhaPadrao();
414             /*
415             if( cc.getDistanceX() <= Adistance[0])
416             {
417               cc.commandCall(calls[0]);
418             }
419             else if( cc.getDistanceX() <= Adistance[16])
420             {
421               cc.commandCall(calls[16]);
422             }
423             else if( cc.getDistanceX() <= Adistance[9])
424             {
425               cc.commandCall(calls[9]);
426             }
427             else
428               escolhaPadrao();
429             */
430           }
431           else if(ataqueEscolhido == 13)
432           {
433             if( cc.getDistanceX() <= Adistance[0])
434             {
435               cc.commandCall(calls[0]);
436             }
437             else if( cc.getDistanceX() <= Adistance[7])
438             {
439               cc.commandCall(calls[7]);
440             }
441             else
442               escolhaPadrao();
443           }
444           else if(ataqueEscolhido == 14)//refazer os testes
445           {
446             if( cc.getDistanceX() <= Adistance[7])
447             {
448               cc.commandCall(calls[7]);
449             }
450             else if( cc.getDistanceX() <= Adistance[2])
451             {
```

```
452               cc.commandCall(calls[2]);
453             }
454             else
455               escolhaPadrao();
456           }
457           else if(ataqueEscolhido == 15)
458           {
459             if( cc.getDistanceX() <= Adistance[13])
460             {
461               cc.commandCall(calls[13]);
462             }
463             else if( cc.getDistanceX() <= Adistance[9])
464             {
465               cc.commandCall(calls[9]);
466             }
467             else
468               escolhaPadrao();
469           }
470           else if(ataqueEscolhido == 16)
471           {
472             if( cc.getDistanceX() <= Adistance[9])
473             {
474               cc.commandCall(calls[9]);
475             }
476             else
477               escolhaPadrao();
478           }
479         //
480         //Descobrindo a distancia dos ataques
481           //if(cc.getEnemyCharacter().getAttack() != null)
482           //{
483             //distanciaAtaqueTeste = cc.getEnemyCharacter().
                  getAttack().getHitAreaSetting().getR();
484             //distanciaAtaqueTeste2 = cc.getEnemyCharacter().
                  getAttack().getHitAreaSetting().getL();
485             //System.out.println(distanciaAtaqueTeste+"   "+
                  distanciaAtaqueTeste2);
486           //}
487         //
488       }
489
490     }
491
492   }
493
494
495   private void escolhaPadrao()
496   {
497     if(cc.getMyEnergy()>= 300)
498       cc.commandCall(calls[3]);
499     else if (cc.getMyEnergy()> 100)
500       cc.commandCall(calls[9]);
501     else
```

```
502         cc.commandCall(calls[(int) (Math.random()*16)]);
503     }
504 
505     @Override
506     public Key input() {
507       return inputKey;
508     }
509 
510     @Override
511     public void close() {
512 
513     }
514 
515 }
```

A.2 Código fonte da AI(DragonKing1C) enviada para o torneio na categoria 1C

```
1  import commandcenter.CommandCenter;
2  import structs.FrameData;
3  import structs.GameData;
4  import structs.Key;
5  import gameInterface.AIInterface;
6  
7      //DEVELOPED BY : Renan Motta Goulart, Guilherme Albuquerque
           Pinto
8      // Universidade Federal de Juiz de Fora, 2014
9      //CONTACT INFO : raikoalihara@hotmail.com, renan.motta@ice.
           ufjf.br, guilherme.pinto@gmail.com
10 
11 public class DragonKing1C implements AIInterface {
12 
13   private boolean playerNumber;
14   private Key inputKey;
15   private FrameData frameData;
16   private CommandCenter cc;
17   /*
18   //probabilidade dos ataques
19     //throw
20       private int EP_Throw_A; --> 0
21       private int EP_Throw_B; --> 1
22     //
23     //High
24       private int EP_Stand_F_D_DFA; --> 2
25       private int EP_Stand_D_DF_FC; --> 3
26       private int EP_Stand_A; --> 4
27       private int EP_Stand_B; --> 5
```

```
28        private int EP_Stand_FA; --> 6
29        private int EP_Stand_D_DF_FA; --> 7
30        private int EP_Stand_D_DF_FB; --> 8
31        private int EP_Stand_D_DB_BB; --> 9
32      //
33      //Middle
34        private int EP_Stand_F_D_DFB; --> 10
35        private int EP_Stand_FB; --> 11
36        private int EP_Stand_D_DB_BA; --> 12
37      //
38      //Low
39        private int EP_Crouch_A; --> 13
40        private int EP_Crouch_FA; --> 14
41        private int EP_Crouch_B; --> 15
42        private int EP_Crouch_FB; --> 16
43      //
44      //total
45       */
46    //variaveis do aprendizado
47      private int i, j;//variavel usada nos for
48      //private int enemy_totalAttacks;//quantidade total de
           ataques utilizados pelo oponente ate agora
49      private int qtdAttacks = 17+14;//ground+air
50      private int[] EPs = new int[qtdAttacks];//quantidade de vezes
            que ele utilizou os ataques
51      private int[] EAAverage = new int[qtdAttacks];
52      private int[] EAcounter = new int[qtdAttacks];
53      private int memorySize = 10;
54      private int[] EAADistances = new int[memorySize*qtdAttacks];
55      private int[] EAStdDeviation = new int[qtdAttacks];
56      private int[] EAmaxDistance = new int[qtdAttacks];//distancia
            maxima que o oponete utilizou os ataques
57      private int[] EAminDistance = new int[qtdAttacks];//distancia
            minima que o oponenten utilizou os ataques
58      private int[] Adistance = {150, 150, 250, 240, 220, 240, 240,
            250, 300, 300, 240, 240, 128, 220, 250, 240, 250, 300,
           300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300,
            300};//new int[17];
59      private String[] calls = {"4 _ A", "4 _ B", "6 2 3 _ A", "2 3
            6 _ C", "A", "B", "6 _ A", "2 3 6 _ A", "2 3 6 _ B", "2 1
           4 _ B", "6 2 3 _ B", "6 _ B", "2 1 4 _ A", "2 _ A", "3 _ A"
           , "2 _ B", "3 _ B", "A", "B", "2 _ A", "2 _ B", "6 _ A", "6
            _ B", "8 _ A", "8 _ B", "2 3 6 _ A", "2 3 6 _ B", "6 2 3 _
            A", "6 2 3 _ B", "2 1 4 _ A", "2 1 4 _ B"};
60      private String[] calls2 = {"THROW_A", "THROW_B", "
           STAND_F_D_DFA", "EP_STAND_D_DF_FC", "STAND_A", "STAND_B", "
           STAND_FA", "STAND_D_DF_FA", "STAND_D_DF_FB", "STAND_D_DB_BB
           ", "STAND_F_D_DFB", "STAND_FB", "STAND_D_DB_BA", "CROUCH_A"
           , "CROUCH_FA", "CROUCH_B", "CROUCH_FB", "AIR_A", "AIR_B", "
           AIR_DA", "AIR_DB", "AIR_FA", "AIR_FB", "AIR_UA", "AIR_UB",
           "AIR_D_DF_FA", "AIR_D_DF_FB", "AIR_F_D_DFA", "AIR_F_D_DFB",
            "AIR_D_DB_BA", "AIR_D_DBarrumar, talvez ataque aerio_BB"};
61      private int probTotal;
```

```
62      private int ataqueEscolhido;//ataque que se acredita que o
            oponente vai usar agora
63      private int ataqueEscolhidoContador;//contadr para se desidir
             qual ataque o oponente deve utilizar agora
64      private int tempoFimAtaqueInimigo;//tempo que falata para o
            ataque acabar
65
66      private Boolean mostraBoo = true;
67
68      //private int distanciaAtaqueTeste;
69      //private int distanciaAtaqueTeste2;
70    //
71
72    //variaveis do combo
73      private boolean comboLock = false;
74      private float enemyHP = 0;
75      private boolean specialDodgeLock = false;
76    //
77
78    @Override
79    public void close() {
80      // TODO Auto-generated method stub
81
82    }
83
84    @Override
85    public String getCharacter() {
86      return CHARACTER_KFM;
87    }
88
89    @Override
90    public void getInformation(FrameData arg0) {
91      this.frameData = arg0;
92      cc.setFrameData(this.frameData, playerNumber);
93
94    }
95
96    @Override
97    public int initialize(GameData gameData, boolean playerNumber)
         {
98      this.playerNumber = playerNumber;
99      this.inputKey = new Key();
100     cc = new CommandCenter();
101     frameData = new FrameData();
102     for(i=0;i<17;i++)
103     {
104       EAmaxDistance[i] = 0;
105       EAminDistance[i] = 9999;
106     }
107     return 0;
108   }
109
110   @Override
```

```
111   public Key input() {
112     return inputKey;
113   }
114
115   @Override
116   public void processing() {
117     if (!frameData.emptyFlag && frameData.getRemainingTime() > 0)
          //linha obrigat ria segundo a documenta  o
118     {
119       if (cc.getskillFlag())//testa se um skill esta sendo "
            inputado"(n  o sei como falar isso em portugu s)
120       {
121         inputKey = cc.getSkillKey();//caso sim entao continua a
              inputar ele
122       }
123       else
124       {
125         if(frameData.getRemainingTime() > 55000)
126         {
127            mostraBoo = true;
128         }
129         if(frameData.getRemainingTime() < 5000 && mostraBoo ==
              true)
130         {
131           System.out.println(" ");
132           System.out.println("1-> Average:"+EAAverage[1]+"
                StdDeviation: "+EAStdDeviation[1]);
133           System.out.println("8-> Average:"+EAAverage[8]+"
                StdDeviation: "+EAStdDeviation[8]);
134           System.out.println("9-> Average:"+EAAverage[9]+"
                StdDeviation: "+EAStdDeviation[9]);
135           System.out.println("10-> Average:"+EAAverage[10]+"
                StdDeviation: "+EAStdDeviation[10]);
136           System.out.println("12-> Average:"+EAAverage[12]+"
                StdDeviation: "+EAStdDeviation[12]);
137           System.out.println("3-> Average:"+EAAverage[3]+"
                StdDeviation: "+EAStdDeviation[3]);
138           mostraBoo = false;
139         }
140         inputKey.empty();//limpa a inputKey(reseta/reinicia ela)
141         cc.skillCancel();//limpa skillData(array que contem a
              sequ cia de botoes para serem apertados) e marca
              skillFlag como false
142
143         //Espera o inimigo finalizar o ataque dele, ainda da pra
              melhorar esta linha em relacao ao status atual do
              inimigo tambem...
144           if(tempoFimAtaqueInimigo > 0)
145           {
146             tempoFimAtaqueInimigo -= 1;
147             //System.out.println(tempoFimAtaqueInimigo);
148           }
149         //
```

```
150         //Descobre qual ataque o inimigo utiliza em relacao a
                distancia
151           if(tempoFimAtaqueInimigo == 0)
152           {
153             for(i=0;i<qtdAttacks;i++)
154             {
155               if(cc.getEnemyCharacter().getAction().name().equals
                      (calls2[i]))
156               {
157                 EPs[i] += 1;
158                 tempoFimAtaqueInimigo = cc.getEnemyCharacter().
                      getRemainingFrame();
159                 EAADistances[memorySize*i+EAcounter[i]] = cc.
                      getDistanceX();
160                 EAAverage[i] = 0;
161                 if(EPs[i] >= memorySize-1)
162                 {
163                   for(j=0;j<memorySize;j++)
164                     EAAverage[i]+=EAADistances[memorySize*i+j];
165                 }
166                 else
167                 {
168                   for(j=0;j<EPs[i];j++)
169                     EAAverage[i]+=EAADistances[memorySize*i+j];
170                 }
171                 EAAverage[i] = EAAverage[i]/memorySize;
172                 EAStdDeviation[i] = 0;
173                 for(j=0;j<memorySize;j++)
174                 {
175                   EAStdDeviation[i] += (EAADistances[memorySize*i
                        +j]-EAAverage[i])*(EAADistances[memorySize*i+
                        j]-EAAverage[i]);
176                 }
177                 EAStdDeviation[i] = EAStdDeviation[i]/memorySize;
178                 EAStdDeviation[i] =(int) Math.sqrt(EAStdDeviation
                      [i]);
179                 if(EAcounter[i] == memorySize-1)
180                   EAcounter[i] = 0;
181                 else
182                   EAcounter[i]+=1;
183                 EAmaxDistance[i] = EAAverage[i]+EAStdDeviation[i
                      ];
184                 EAminDistance[i] = EAAverage[i]-EAStdDeviation[i
                      ];
185               }
186             }
187           }
188         //
189
190         //Procura quais ataques o inimigo pode utilizar nesta
                distancia
191           probTotal = 0;
192           ataqueEscolhido = 0;
```

```
193             ataqueEscolhidoContador = 0;
194             for(i=0;i<qtdAttacks;i++)
195             {
196               if(cc.getDistanceX() <= EAmaxDistance[i] && cc.
                    getDistanceX() >= EAminDistance[i])
197                 probTotal += EPs[i];
198             }
199             if(probTotal == 0)
200             {
201               ataqueEscolhido = (int) (Math.random()*(qtdAttacks-1)
                    );//utilizando uma heiristica bem simples de
                    escolher um ataque randomicamente. Vale a pena
                    melhorar isto em versoes futuras
202             }
203             else
204             {
205               ataqueEscolhido = (int) (Math.random()*probTotal);//
                    TESTAR SE ISTO ESTA FUNCIONANDO MESMO, talvez tenha
                     um modo melhor de se realizar isto utilizando a
                    classe "Random"
206               for(i=0;i<qtdAttacks;i++)
207               {
208                 if(cc.getDistanceX() <= EAmaxDistance[i] && cc.
                      getDistanceX() >= EAminDistance[i])
209                 {
210                   ataqueEscolhidoContador += EPs[i];
211                   if(ataqueEscolhidoContador >= ataqueEscolhido)
212                   {
213                     ataqueEscolhido = i;
214                     i = 1000;
215                   }
216                 }
217               }
218             }
219           //
220
221           //Decide qual ataque sera utilizado
222
223             if(cc.getEnemyHP()<enemyHP)
224             {
225               if(cc.getMyCharacter().getAction().name().equals(
                    calls2[0]) || cc.getMyCharacter().getAction().name
                    ().equals(calls2[1]))//acertei o throw
226                 comboLock = true;
227             }
228             if(cc.getMyCharacter().getAction().name().equals(calls2
                  [2]))//combo realizado
229             {
230               comboLock = false;
231             }
232             if(cc.getEnemyCharacter().getAction().name().equals(
                  calls2[3]))//inimigo usando especial
233             {
```

```
234             cc.commandCall(calls[9]);//escapar com rasteira
235           }
236           if(cc.getMyCharacter().getState().equals("DOWN") && (cc
                .getEnemyCharacter().getAction().name().equals(calls2
                [3]) || cc.getEnemyCharacter().getAction().name().
                equals(calls2[7])))
237           {
238             specialDodgeLock = true;
239           }
240           if(cc.getMyCharacter().getAction().name().equals(calls2
                [9]))
241           {
242             specialDodgeLock = false;
243           }
244           if(comboLock == true)
245           {
246             cc.commandCall(calls[2]);//soco subindo
247           }
248           else if(specialDodgeLock == true)
249           {
250             cc.commandCall(calls[9]);//soco subindo
251           }
252           else if(cc.getEnemyCharacter().getAction().name().
                equals(calls2[7]))//inimigo usando bola de fogo
253           {
254             cc.commandCall(calls[9]);//escapar com rasteira
255           }
256           else if(cc.getDistanceX() >= 300 && cc.getMyEnergy() >=
                300)//utiliza o ataque especial
257           {
258             cc.commandCall(calls[3]);
259           }
260           else if(ataqueEscolhido == 0)//7f
261           {
262             if( cc.getDistanceX() <= Adistance[9])
263             {
264               cc.commandCall(calls[9]);
265             }
266             else if( cc.getDistanceX() <= Adistance[11])
267             {
268               cc.commandCall(calls[11]);
269             }
270             else
271               escolhaPadrao();
272           }
273           else if(ataqueEscolhido == 1)//13f
274           {
275             if( cc.getDistanceX() <= Adistance[11])
276             {
277               cc.commandCall(calls[11]);
278             }
279             else if( cc.getDistanceX() <= Adistance[2])
280             {
```

```
281             cc.commandCall(calls[2]);
282           }
283           else
284             escolhaPadrao();
285         }
286         else if(ataqueEscolhido == 2)
287         {
288           if( cc.getDistanceX() <= Adistance[28])
289           {
290             if(cc.getMyCharacter().getState().name().equals("
                  STAND") || cc.getMyCharacter().getState().name().
                  equals("CROUCH"))
291               cc.commandCall("JUMP");
292             else
293               cc.commandCall(calls[28]);
294           }
295           if( cc.getDistanceX() <= Adistance[8])
296           {
297             cc.commandCall(calls[8]);
298           }
299           else if( cc.getDistanceX() <= Adistance[2])
300           {
301             cc.commandCall("STAND_GUARD");//ataque mais rapido
                  do jogo
302           }
303           else
304             escolhaPadrao();
305         }
306         else if(ataqueEscolhido == 3)//especial
307         {
308           if(cc.getEnemyEnergy() > 300)
309             cc.commandCall(calls[9]);
310           else
311             escolhaPadrao();
312         }
313         else if(ataqueEscolhido == 4)
314         {
315           if( cc.getDistanceX() <= Adistance[9])
316           {
317             cc.commandCall(calls[9]);
318           }
319           else
320             escolhaPadrao();
321         }
322         else if(ataqueEscolhido == 5)//12f
323         {
324           if( cc.getDistanceX() <= Adistance[9])
325             cc.commandCall(calls[9]);
326           else
327             escolhaPadrao();
328         }
329         else if(ataqueEscolhido == 6)
330         {
```

```
331             if( cc.getDistanceX() <= Adistance[9])
332             {
333               cc.commandCall(calls[9]);
334             }
335             else
336               escolhaPadrao();
337           }
338           else if(ataqueEscolhido == 7)
339           {
340             if( cc.getDistanceX() <= Adistance[8])
341             {
342               cc.commandCall(calls[8]);
343             }
344             else if( cc.getDistanceX() <= Adistance[9])
345             {
346               cc.commandCall(calls[9]);
347             }
348             else
349               escolhaPadrao();
350           }
351           else if(ataqueEscolhido == 8)//13f
352           {
353             if( cc.getDistanceX() <= Adistance[9])
354             {
355               cc.commandCall(calls[9]);
356             }
357             else
358               escolhaPadrao();
359           }
360           else if(ataqueEscolhido == 9)
361           {
362             if( cc.getDistanceX() <= Adistance[1])
363             {
364               cc.commandCall(calls[1]);
365             }
366             else
367               cc.commandCall(calls[9]);
368           }
369           else if(ataqueEscolhido == 10)
370           {
371             if( cc.getDistanceX() <= Adistance[28])
372             {
373               if(cc.getMyCharacter().getState().name().equals("
                     STAND") || cc.getMyCharacter().getState().name().
                     equals("CROUCH"))
374                 cc.commandCall("JUMP");
375               else
376                 cc.commandCall(calls[28]);
377             }
378             else if( cc.getDistanceX() <= Adistance[15])
379             {
380               cc.commandCall(calls[15]);
381             }
```

```
382             else if( cc.getDistanceX() <= Adistance[8])
383             {
384               cc.commandCall(calls[8]);
385             }
386             else
387               escolhaPadrao();
388           }
389           else if(ataqueEscolhido == 11)
390           {
391             if( cc.getDistanceX() <= Adistance[28])
392             {
393               if(cc.getMyCharacter().getState().name().equals("
                    STAND") || cc.getMyCharacter().getState().name().
                    equals("CROUCH"))
394                 cc.commandCall("JUMP");
395               else
396                 cc.commandCall(calls[28]);
397             }
398             else if( cc.getDistanceX() <= Adistance[15])
399             {
400               cc.commandCall(calls[15]);
401             }
402             else if( cc.getDistanceX() <= Adistance[10])
403             {
404               cc.commandCall(calls[10]);
405             }
406             else if( cc.getDistanceX() <= Adistance[16])
407             {
408               cc.commandCall(calls[16]);
409             }
410             else
411               escolhaPadrao();
412           }
413           else if(ataqueEscolhido == 12)//soco invertido em pe
414           {
415             if( cc.getDistanceX() <= Adistance[10])
416             {
417               cc.commandCall(calls[10]);
418             }
419             else if( cc.getDistanceX() <= Adistance[8])
420             {
421               cc.commandCall(calls[8]);
422             }
423             else
424               escolhaPadrao();
425           }
426           else if(ataqueEscolhido == 13)
427           {
428             if( cc.getDistanceX() <= Adistance[0])
429             {
430               cc.commandCall(calls[0]);
431             }
432             else if( cc.getDistanceX() <= Adistance[7])
```

```
433             {
434               cc.commandCall(calls[7]);
435             }
436             else
437               escolhaPadrao();
438           }
439           else if(ataqueEscolhido == 14)//ARRUMAR, TALVEZ ATAQUE
                AERIO
440           {
441             if( cc.getDistanceX() <= Adistance[0])//iniciar combo
                  ?
442             {
443               cc.commandCall(calls[0]);
444             }
445             if( cc.getDistanceX() <= Adistance[7])
446             {
447               cc.commandCall(calls[7]);
448             }
449             else if( cc.getDistanceX() <= Adistance[2])
450             {
451               cc.commandCall(calls[2]);
452             }
453             else
454               escolhaPadrao();
455           }
456           else if(ataqueEscolhido == 15)
457           {
458             if( cc.getDistanceX() <= Adistance[9])
459             {
460               cc.commandCall(calls[9]);
461             }
462             else if( cc.getDistanceX() <= Adistance[13])
463             {
464               cc.commandCall(calls[13]);
465             }
466             else
467               escolhaPadrao();
468           }
469           else if(ataqueEscolhido == 16)
470           {
471             if( cc.getDistanceX() <= Adistance[9])
472             {
473               cc.commandCall(calls[9]);
474             }
475             else
476               escolhaPadrao();
477           }
478           else if(ataqueEscolhido == 17)
479           {
480             if( cc.getDistanceX() <= Adistance[2])
481             {
482               cc.commandCall(calls[2]);
483             }
```

```
484             else
485               escolhaPadrao();
486           }
487           else if(ataqueEscolhido == 18)//testar
488           {
489
490             if( cc.getDistanceX() <= Adistance[20])
491             {
492               if(cc.getMyCharacter().getState().name().equals("
                     STAND") || cc.getMyCharacter().getState().name().
                     equals("CROUCH"))
493                 cc.commandCall("JUMP");
494               else
495                 cc.commandCall(calls[20]);
496             }
497             if( cc.getDistanceX() <= Adistance[8])
498             {
499               cc.commandCall(calls[8]);
500             }
501             else
502               escolhaPadrao();
503           }
504           else if(ataqueEscolhido == 19)//testar
505           {
506
507             if( cc.getDistanceX() <= Adistance[20])
508             {
509               if(cc.getMyCharacter().getState().name().equals("
                     STAND") || cc.getMyCharacter().getState().name().
                     equals("CROUCH"))
510                 cc.commandCall("JUMP");
511               else
512                 cc.commandCall(calls[20]);
513             }
514             if( cc.getDistanceX() <= Adistance[8])
515             {
516               cc.commandCall(calls[8]);
517             }
518             else
519               escolhaPadrao();
520           }
521           else if(ataqueEscolhido == 20)//soco anti aerio
522           {
523             if( cc.getDistanceX() <= Adistance[8])
524             {
525               cc.commandCall(calls[8]);
526             }
527             else
528               escolhaPadrao();
529           }
530           else if(ataqueEscolhido == 21)//soco anti aerio
531           {
532             if( cc.getDistanceX() <= Adistance[2])
```

```
533             {
534               cc.commandCall(calls[2]);
535             }
536             else
537               escolhaPadrao();
538           }
539           else if(ataqueEscolhido == 22)//soco anti aerio
540           {
541             if( cc.getDistanceX() <= Adistance[2])
542             {
543               cc.commandCall(calls[2]);
544             }
545             else
546               escolhaPadrao();
547           }
548           else if(ataqueEscolhido == 23)//soco anti aerio
549           {
550             if( cc.getDistanceX() <= Adistance[20])
551             {
552               if(cc.getMyCharacter().getState().name().equals("
                     STAND") || cc.getMyCharacter().getState().name().
                     equals("CROUCH"))
553                 cc.commandCall("JUMP");
554               else
555                 cc.commandCall(calls[20]);
556             }
557             else
558               escolhaPadrao();
559           }
560           else if(ataqueEscolhido == 24)//soco anti aerio
561           {
562             if( cc.getDistanceX() <= Adistance[23])
563             {
564               if(cc.getMyCharacter().getState().name().equals("
                     STAND") || cc.getMyCharacter().getState().name().
                     equals("CROUCH"))
565                 cc.commandCall("JUMP");
566               else
567                 cc.commandCall(calls[23]);
568             }
569             else
570               escolhaPadrao();
571           }
572           else if(ataqueEscolhido == 25)//n funciona direito
573           {
574             if( cc.getDistanceX() <= Adistance[8])
575             {
576               cc.commandCall(calls[8]);
577             }
578             else if( cc.getDistanceX() <= Adistance[2])
579             {
580               cc.commandCall(calls[2]);
581             }
```

```
582             else
583               escolhaPadrao();
584           }
585           else if(ataqueEscolhido == 26)//n funciona direito
586           {
587             if( cc.getDistanceX() <= Adistance[20])
588             {
589               if(cc.getMyCharacter().getState().name().equals("
                     STAND") || cc.getMyCharacter().getState().name().
                     equals("CROUCH"))
590                 cc.commandCall("JUMP");
591               else
592                 cc.commandCall(calls[20]);
593             }
594             else if( cc.getDistanceX() <= Adistance[2])
595             {
596               cc.commandCall(calls[2]);
597             }
598             else
599               escolhaPadrao();
600           }
601           else if(ataqueEscolhido == 27)//n funciona direito
602           {
603             if( cc.getDistanceX() <= Adistance[21])
604             {
605               if(cc.getMyCharacter().getState().name().equals("
                     STAND") || cc.getMyCharacter().getState().name().
                     equals("CROUCH"))
606                 cc.commandCall("JUMP");
607               else
608                 cc.commandCall(calls[21]);
609             }
610             else if( cc.getDistanceX() <= Adistance[8])
611             {
612               cc.commandCall(calls[8]);
613             }
614             else
615               escolhaPadrao();
616           }
617           else if(ataqueEscolhido == 28)//n funciona direito
618           {
619             if( cc.getDistanceX() <= Adistance[8])
620             {
621               cc.commandCall(calls[8]);
622             }
623             else
624               escolhaPadrao();
625           }
626           else if(ataqueEscolhido == 29)//n funciona direito
627           {
628             if( cc.getDistanceX() <= Adistance[2])
629             {
630               cc.commandCall(calls[2]);
```

```
631             }
632             else if( cc.getDistanceX() <= Adistance[8])
633             {
634               cc.commandCall(calls[8]);
635             }
636             else
637               escolhaPadrao();
638           }
639           else if(ataqueEscolhido == 30)//n funciona direito
640           {
641             if( cc.getDistanceX() <= Adistance[2])
642             {
643               cc.commandCall(calls[2]);
644             }
645             else if( cc.getDistanceX() <= Adistance[8])
646             {
647               cc.commandCall(calls[8]);
648             }
649             else
650               escolhaPadrao();
651           }
652           enemyHP = cc.getEnemyHP();
653
654         //
655       }
656
657     }
658
659   }
660
661   private void escolhaPadrao()
662   {
663     if(cc.getMyEnergy()>= 300)
664       cc.commandCall(calls[3]);
665     else if (cc.getMyEnergy()> 100)
666       cc.commandCall(calls[9]);
667     else
668       cc.commandCall(calls[(int) (Math.random()*16)]);
669   }
670 }
```

A.3 Código da AI(DragonKing3C) enviada para o Torneio na Categoria 3C

```
1 import commandcenter.CommandCenter;
2 import structs.FrameData;
3 import structs.GameData;
```

```
4  import structs.Key;
5  import gameInterface.AIInterface;
6
7      //DEVELOPED BY : Renan Motta Goulart, Guilherme Albuquerque
           Pinto
8      // Universidade Federal de Juiz de Fora, 2014
9      //CONTACT INFO : raikoalihara@hotmail.com, renan.motta@ice.
           ufjf.br, guilherme.pinto@gmail.com
10
11 public class DragonKing3C implements AIInterface {
12
13   private boolean playerNumber;
14   private Key inputKey;
15   private FrameData frameData;
16   private GameData gameData;
17   private CommandCenter cc;
18   private String enemyChar;
19   /*
20   //probabilidade dos ataques
21     //throw
22       private int EP_Throw_A; --> 0
23       private int EP_Throw_B; --> 1
24     //
25     //High
26       private int EP_Stand_F_D_DFA; --> 2
27       private int EP_Stand_D_DF_FC; --> 3
28       private int EP_Stand_A; --> 4
29       private int EP_Stand_B; --> 5
30       private int EP_Stand_FA; --> 6
31       private int EP_Stand_D_DF_FA; --> 7
32       private int EP_Stand_D_DF_FB; --> 8
33       private int EP_Stand_D_DB_BB; --> 9
34     //
35     //Middle
36       private int EP_Stand_F_D_DFB; --> 10
37       private int EP_Stand_FB; --> 11
38       private int EP_Stand_D_DB_BA; --> 12
39     //
40     //Low
41       private int EP_Crouch_A; --> 13
42       private int EP_Crouch_FA; --> 14
43       private int EP_Crouch_B; --> 15
44       private int EP_Crouch_FB; --> 16
45     //
46     //total
47      */
48   //variaveis do aprendizado
49     private int i, j;//variavel usada nos for
50     //private int enemy_totalAttacks;//quantidade total de
           ataques utilizados pelo oponente ate agora
51     private int qtdAttacks = 17+14;//ground+air
52     private int[] EPs = new int[qtdAttacks];//quantidade de vezes
           que ele utilizou os ataques
```

```
53      private int[] EAAverage = new int[qtdAttacks];
54      private int[] EAcounter = new int[qtdAttacks];
55      private int memorySize = 10;
56      private int[] EAADistances = new int[memorySize*qtdAttacks];
57      private int[] EAStdDeviation = new int[qtdAttacks];
58      private int[] EAmaxDistance = new int[qtdAttacks];//distancia
            maxima que o oponete utilizou os ataques
59      private int[] EAminDistance = new int[qtdAttacks];//distancia
            minima que o oponenten utilizou os ataques
60      private int[] Adistance = {150, 150, 250, 240, 220, 240, 240,
            250, 300, 300, 240, 240, 128, 220, 250, 240, 250, 300,
           300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300,
            300};//new int[17];
61      private int[] AdistanceG = {250, 250, 250, 250, 250, 250,
           250, 250, 250, 250, 650, 650, 900, 250, 250, 250, 300, 250,
            250, 250, 250, 250, 250, 250, 250, 250, 250, 250, 250,
           250, 250};
62      private String[] calls = {"4 _ A", "4 _ B", "6 2 3 _ A", "2 3
            6 _ C", "A", "B", "6 _ A", "2 3 6 _ A", "2 3 6 _ B", "2 1
           4 _ B", "6 2 3 _ B", "6 _ B", "2 1 4 _ A", "2 _ A", "3 _ A"
           , "2 _ B", "3 _ B", "A", "B", "2 _ A", "2 _ B", "6 _ A", "6
            _ B", "8 _ A", "8 _ B", "2 3 6 _ A", "2 3 6 _ B", "6 2 3 _
            A", "6 2 3 _ B", "2 1 4 _ A", "2 1 4 _ B"};
63      private String[] callsG = {"4 _ A", "4 _ B", "A", "B", "2 _ A
           ", "2 _ B", "6 _ A", "6 _ B", "3 _ A", "3 _ B", "2 3 6 _ A"
           , "2 3 6 _ B", "2 3 6 _ C", "6 2 3 _ A", "6 2 3 _ B", "2 1
           4 _ A", "2 1 4 _ B", "A", "B", "2 _ A", "2 _ B", "6 _ A", "
           6 _ B", "8 _ A", "8 _ B", "2 3 6 _ A", "2 3 6 _ B", "6 2 3
           _ A", "6 2 3 _ B", "2 1 4 _ A", "2 1 4 _ B"};
64      private String[] callsZ = {"THROW_A", "THROW_B", "STAND_A", "
           STAND_B", "CROUCH_A", "CROUCH_B", "STAND_FA", "STAND_FB", "
           CROUCH_FA", "CROUCH_FB", "STAND_D_DF_FA", "STAND_D_DF_FB",
           "STAND_D_DF_FC", "STAND_F_D_DFA", "STAND_F_D_DFB", "
           STAND_D_DB_BA", "STAND_D_DB_BB",  "AIR_A", "AIR_B", "AIR_DA
           ", "AIR_DB", "AIR_FA", "AIR_FB", "AIR_UA", "AIR_UB", "
           AIR_D_DF_FA", "AIR_D_DF_FB", "AIR_F_D_DFA", "AIR_F_D_DFB",
           "AIR_D_DB_BA", "AIR_D_DB_BB"};
65      private String[] calls2 = {"THROW_A", "THROW_B", "
           STAND_F_D_DFA", "EP_STAND_D_DF_FC", "STAND_A", "STAND_B", "
           STAND_FA", "STAND_D_DF_FA", "STAND_D_DF_FB", "STAND_D_DB_BB
           ", "STAND_F_D_DFB", "STAND_FB", "STAND_D_DB_BA", "CROUCH_A"
           , "CROUCH_FA", "CROUCH_B", "CROUCH_FB", "AIR_A", "AIR_B", "
           AIR_DA", "AIR_DB", "AIR_FA", "AIR_FB", "AIR_UA", "AIR_UB",
           "AIR_D_DF_FA", "AIR_D_DF_FB", "AIR_F_D_DFA", "AIR_F_D_DFB",
            "AIR_D_DB_BA", "AIR_D_DB_BB"};
66      private int probTotal;
67      private int ataqueEscolhido;//ataque que se acredita que o
            oponente vai usar agora
68      private int ataqueEscolhidoContador;//contadr para se desidir
            qual ataque o oponente deve utilizar agora
69      private int tempoFimAtaqueInimigo;//tempo que falata para o
            ataque acabar
70
```

```
71      private Boolean mostraBoo = true;
72
73    //
74
75    @Override
76    public void close() {
77      // TODO Auto-generated method stub
78
79    }
80
81    @Override
82    public String getCharacter() {
83      return CHARACTER_GARNET;
84    }
85
86    @Override
87    public void getInformation(FrameData arg0) {
88      this.frameData = arg0;
89      cc.setFrameData(this.frameData, playerNumber);
90    }
91
92    @Override
93    public int initialize(GameData gameData, boolean playerNumber)
         {
94      this.playerNumber = playerNumber;
95      this.gameData = gameData;
96      this.inputKey = new Key();
97      cc = new CommandCenter();
98      frameData = new FrameData();
99      for(i=0;i<17;i++)
100     {
101       EAmaxDistance[i] = 0;
102       EAminDistance[i] = 9999;
103     }
104     return 0;
105   }
106
107   @Override
108   public Key input() {
109     return inputKey;
110   }
111
112   @Override
113   public void processing() {
114     if (!frameData.emptyFlag && frameData.getRemainingTime() > 0)
           //linha obrigat ria segundo a documenta  o
115     {
116       if(playerNumber == true)
117         enemyChar = gameData.getPlayerTwoCharacterName();
118       else
119         enemyChar = gameData.getPlayerOneCharacterName();
120       if (cc.getskillFlag())//testa se um skill esta sendo "
             inputado"(n  o sei como falar isso em portugu s)
```

```
121         {
122           inputKey = cc.getSkillKey();//caso sim entao continua a
                 inputar ele
123         }
124         else
125         {
126           if(frameData.getRemainingTime() > 55000)
127           {
128              mostraBoo = true;
129           }
130           if(frameData.getRemainingTime() < 5000 && mostraBoo ==
                 true)
131           {
132             System.out.println(" ");
133             System.out.println("1-> Average:"+EAAverage[1]+"
                   StdDeviation: "+EAStdDeviation[1]);
134             System.out.println("8-> Average:"+EAAverage[8]+"
                   StdDeviation: "+EAStdDeviation[8]);
135             System.out.println("9-> Average:"+EAAverage[9]+"
                   StdDeviation: "+EAStdDeviation[9]);
136             System.out.println("10-> Average:"+EAAverage[10]+"
                   StdDeviation: "+EAStdDeviation[10]);
137             System.out.println("12-> Average:"+EAAverage[12]+"
                   StdDeviation: "+EAStdDeviation[12]);
138             System.out.println("3-> Average:"+EAAverage[3]+"
                   StdDeviation: "+EAStdDeviation[3]);
139             mostraBoo = false;
140           }
141           inputKey.empty();//limpa a inputKey(reseta/reinicia ela)
142           cc.skillCancel();//limpa skillData(array que contem a
                 sequ cia de botoes para serem apertados) e marca
                 skillFlag como false
143
144           //Espera o inimigo finalizar o ataque dele, ainda da pra
                 melhorar esta linha em relacao ao status atual do
                 inimigo tambem...
145             if(tempoFimAtaqueInimigo > 0)
146             {
147               tempoFimAtaqueInimigo -= 1;
148             }
149           //
150           //Descobre qual ataque o inimigo utiliza em relacao a
                 distancia
151             if(tempoFimAtaqueInimigo == 0)
152             {
153               for(i=0;i<qtdAttacks;i++)
154               {
155                 if(cc.getEnemyCharacter().getAction().name().equals
                       (callsZ[i]))
156                 {
157                   EPs[i] += 1;
158                   tempoFimAtaqueInimigo = cc.getEnemyCharacter().
                         getRemainingFrame();
```

```
159                 EAADistances[memorySize*i+EAcounter[i]] = cc.
                       getDistanceX();
160                 EAAverage[i] = 0;
161                 if(EPs[i] >= memorySize-1)
162                 {
163                   for(j=0;j<memorySize;j++)
164                     EAAverage[i]+=EAADistances[memorySize*i+j];
165                 }
166                 else
167                 {
168                   for(j=0;j<EPs[i];j++)
169                     EAAverage[i]+=EAADistances[memorySize*i+j];
170                 }
171                 EAAverage[i] = EAAverage[i]/memorySize;
172                 EAStdDeviation[i] = 0;
173                 for(j=0;j<memorySize;j++)
174                 {
175                   EAStdDeviation[i] += (EAADistances[memorySize*i
                         +j]-EAAverage[i])*(EAADistances[memorySize*i+
                         j]-EAAverage[i]);
176                 }
177                 EAStdDeviation[i] = EAStdDeviation[i]/memorySize;
178                 EAStdDeviation[i] =(int) Math.sqrt(EAStdDeviation
                       [i]);
179                 if(EAcounter[i] == memorySize-1)
180                   EAcounter[i] = 0;
181                 else
182                   EAcounter[i]+=1;
183                 EAmaxDistance[i] = EAAverage[i]+EAStdDeviation[i
                       ];
184                 EAminDistance[i] = EAAverage[i]-EAStdDeviation[i
                       ];
185               }
186             }
187           }
188         //
189
190         //Procura quais ataques o inimigo pode utilizar nesta
               distancia
191           probTotal = 0;
192           ataqueEscolhido = 0;
193           ataqueEscolhidoContador = 0;
194           for(i=0;i<qtdAttacks;i++)
195           {
196             if(cc.getDistanceX() <= EAmaxDistance[i] && cc.
                   getDistanceX() >= EAminDistance[i])
197               probTotal += EPs[i];
198           }
199           if(probTotal == 0)
200           {
201             ataqueEscolhido = (int) (Math.random()*(qtdAttacks-1)
                   );//utilizando uma heiristica bem simples de
                   escolher um ataque randomicamente. Vale a pena
```

```
                melhorar isto em versoes futuras
202           }
203           else
204           {
205             ataqueEscolhido = (int) (Math.random()*probTotal);//
                   TESTAR SE ISTO ESTA FUNCIONANDO MESMO, talvez tenha
                    um modo melhor de se realizar isto utilizando a
                   classe "Random"
206             for(i=0;i<qtdAttacks;i++)
207             {
208               if(cc.getDistanceX() <= EAmaxDistance[i] && cc.
                     getDistanceX() >= EAminDistance[i])
209               {
210                 ataqueEscolhidoContador += EPs[i];
211                 if(ataqueEscolhidoContador >= ataqueEscolhido)
212                 {
213                   ataqueEscolhido = i;
214                   i = 1000;
215                 }
216               }
217             }
218           }
219         //
220
221         //Decide qual ataque sera utilizado
222
223           if(enemyChar == "GARNET")
224           {
225             if(cc.getMyEnergy() > 300)
226             {
227               if(cc.getMyCharacter().getState().name().equals("
                     STAND") || cc.getMyCharacter().getState().name().
                     equals("CROUCH") && ataqueEscolhido<17)
228               {
229                 cc.commandCall(callsG[12]);
230               }
231             }
232             else if(ataqueEscolhido == 0)
233             {
234               if(cc.getDistanceX() <= AdistanceG[7])
235                 cc.commandCall(callsG[7]);
236               else
237                 escolhaPadrao();
238             }
239             else if(ataqueEscolhido == 1)
240             {
241               if(cc.getDistanceX() <= AdistanceG[7])
242                 cc.commandCall(callsG[7]);
243               else
244                 escolhaPadrao();
245             }
246             else if(ataqueEscolhido == 2)
247             {
```

```
248                 if(cc.getDistanceX() <= AdistanceG[7])
249                   cc.commandCall(callsG[7]);
250                 else
251                   escolhaPadrao();
252               }
253               else if(ataqueEscolhido == 3)
254               {
255                 if(cc.getDistanceX() <= AdistanceG[13] && cc.
                      getMyEnergy() > 30)
256                   cc.commandCall(callsG[13]);
257                 if(cc.getDistanceX() <= AdistanceG[1])
258                   cc.commandCall(callsG[1]);
259                 else
260                   escolhaPadrao();
261               }
262               else if(ataqueEscolhido == 4)
263               {
264                 if(cc.getDistanceX() <= AdistanceG[13])
265                   cc.commandCall(callsG[13]);
266                 if(cc.getDistanceX() <= AdistanceG[7])
267                   cc.commandCall(callsG[7]);
268                 else
269                   escolhaPadrao();
270               }
271               else if(ataqueEscolhido == 5)
272               {
273                 if(cc.getDistanceX() <= AdistanceG[13] && cc.
                      getMyEnergy() > 30)
274                   cc.commandCall(callsG[13]);
275                 if(cc.getDistanceX() <= AdistanceG[1])
276                   cc.commandCall(callsG[1]);
277                 else
278                   escolhaPadrao();
279               }
280               else if(ataqueEscolhido == 6)
281               {
282                 if(cc.getDistanceX() <= AdistanceG[5])
283                   cc.commandCall(callsG[5]);
284                 else
285                   escolhaPadrao();
286               }
287               else if(ataqueEscolhido == 7)
288               {
289                 if(cc.getDistanceX() <= AdistanceG[16] && cc.
                      getMyEnergy() > 50)
290                   cc.commandCall(callsG[16]);
291                 if(cc.getDistanceX() <= AdistanceG[7])
292                   cc.commandCall(callsG[7]);
293                 else
294                   escolhaPadrao();
295               }
296               else if(ataqueEscolhido == 8)
297               {
```

```
298                 if(cc.getDistanceX() <= AdistanceG[7])
299                   cc.commandCall(callsG[7]);
300                 else
301                   escolhaPadrao();
302               }
303               else if(ataqueEscolhido == 9)
304               {
305                 if(cc.getDistanceX() <= AdistanceG[7])
306                   cc.commandCall(callsG[7]);
307                 else
308                   escolhaPadrao();
309               }
310               else if(ataqueEscolhido == 10)
311               {
312                 if(cc.getMyCharacter().getState().name().equals("
                       STAND") || cc.getMyCharacter().getState().name().
                       equals("CROUCH"))
313                   cc.commandCall("JUMP");
314                 else
315                   cc.commandCall(callsG[22]);
316               }
317               else if(ataqueEscolhido == 11)
318               {
319                 if(cc.getDistanceX() <= AdistanceG[16] && cc.
                       getMyEnergy() > 50)
320                   cc.commandCall(callsG[16]);
321                 if(cc.getDistanceX() <= AdistanceG[4])
322                   cc.commandCall(callsG[4]);
323                 else
324                   escolhaPadrao();
325               }
326               else if(ataqueEscolhido == 12)
327               {
328                 cc.commandCall("GUARD");
329               }
330               else if(ataqueEscolhido == 13)
331               {
332                 if(cc.getDistanceX() <= AdistanceG[2])
333                   cc.commandCall(callsG[2]);
334                 else
335                   escolhaPadrao();
336               }
337               else if(ataqueEscolhido == 14)
338               {
339                 if(cc.getDistanceX() <= AdistanceG[9])
340                   cc.commandCall(callsG[9]);
341                 else
342                   escolhaPadrao();
343               }
344               else if(ataqueEscolhido == 15)
345               {
346                 if(cc.getDistanceX() <= AdistanceG[9])
347                   cc.commandCall(callsG[9]);
```

```
348                 else
349                   escolhaPadrao();
350               }
351               else if(ataqueEscolhido == 16)
352               {
353                 if(cc.getDistanceX() <= AdistanceG[7])
354                   cc.commandCall(callsG[7]);
355                 else
356                   escolhaPadrao();
357               }
358               else if(ataqueEscolhido == 17)
359               {
360                 if(cc.getDistanceX() <= AdistanceG[11])
361                   cc.commandCall(callsG[11]);
362                 else
363                   escolhaPadrao();
364               }
365               else if(ataqueEscolhido == 18)
366               {
367                 if(cc.getDistanceX() <= AdistanceG[11])
368                   cc.commandCall(callsG[11]);
369                 else
370                   escolhaPadrao();
371               }
372               else if(ataqueEscolhido == 19)
373               {
374                 if(cc.getDistanceX() <= AdistanceG[4])
375                   cc.commandCall(callsG[4]);
376                 else
377                   escolhaPadrao();
378               }
379               else if(ataqueEscolhido == 20)
380               {
381                 if(cc.getDistanceX() <= AdistanceG[4])
382                   cc.commandCall(callsG[4]);
383                 else
384                   escolhaPadrao();
385               }
386               else if(ataqueEscolhido == 21)
387               {
388                 if(cc.getDistanceX() <= AdistanceG[4])
389                   cc.commandCall(callsG[4]);
390                 else
391                   escolhaPadrao();
392               }
393               else if(ataqueEscolhido == 22)
394               {
395                 if(cc.getDistanceX() <= AdistanceG[9])
396                   cc.commandCall(callsG[9]);
397                 else
398                   escolhaPadrao();
399               }
400               else if(ataqueEscolhido == 23)
```

```
401             {
402               if(cc.getDistanceX() <= AdistanceG[4])
403                 cc.commandCall(callsG[4]);
404               else
405                 escolhaPadrao();
406             }
407             else if(ataqueEscolhido == 24)
408             {
409               if(cc.getDistanceX() <= AdistanceG[4])
410                 cc.commandCall(callsG[4]);
411               else
412                 escolhaPadrao();
413             }
414             else if(ataqueEscolhido == 25)
415             {
416               if(cc.getDistanceX() <= AdistanceG[9])
417                 cc.commandCall(callsG[9]);
418               else
419                 escolhaPadrao();
420             }
421             else if(ataqueEscolhido == 26)
422             {
423                 cc.commandCall("DASH");
424             }
425             else if(ataqueEscolhido == 27)
426             {
427               if(cc.getDistanceX() <= AdistanceG[4])
428                 cc.commandCall(callsG[4]);
429               else
430                 escolhaPadrao();
431             }
432             else if(ataqueEscolhido == 28)
433             {
434               if(cc.getDistanceX() <= AdistanceG[4])
435                 cc.commandCall(callsG[4]);
436               else
437                 escolhaPadrao();
438             }
439             else if(ataqueEscolhido == 29)
440             {
441               if(cc.getDistanceX() <= AdistanceG[4])
442                 cc.commandCall(callsG[4]);
443               else
444                 escolhaPadrao();
445             }
446             else if(ataqueEscolhido == 30)
447             {
448               if(cc.getDistanceX() <= AdistanceG[4])
449                 cc.commandCall(callsG[4]);
450               else
451                 escolhaPadrao();
452             }
453
```

```
454
455             }
456             else if(enemyChar == "ZEN")
457             {
458               if(cc.getMyEnergy() > 300)
459               {
460                 if(cc.getMyCharacter().getState().name().equals("
                       STAND") || cc.getMyCharacter().getState().name().
                       equals("CROUCH") && ataqueEscolhido<17)
461                 {
462                   cc.commandCall(callsG[12]);
463                 }
464               }
465               else if(ataqueEscolhido == 0)
466               {
467                 if(cc.getDistanceX() <= AdistanceG[7])
468                   cc.commandCall(callsG[7]);
469                 if(cc.getDistanceX() <= AdistanceG[16] && cc.
                       getMyEnergy() > 50)
470                   cc.commandCall(callsG[16]);
471                 else
472                   escolhaPadrao();
473               }
474               else if(ataqueEscolhido == 1)
475               {
476                 if(cc.getDistanceX() <= AdistanceG[5])
477                   cc.commandCall(callsG[5]);
478                 else
479                   escolhaPadrao();
480               }
481               else if(ataqueEscolhido == 2)
482               {
483                 if(cc.getDistanceX() <= AdistanceG[7])
484                   cc.commandCall(callsG[7]);
485                 else
486                   escolhaPadrao();
487               }
488               else if(ataqueEscolhido == 3)
489               {
490                 if(cc.getDistanceX() <= AdistanceG[7])
491                   cc.commandCall(callsG[7]);
492                 else
493                   escolhaPadrao();
494               }
495               else if(ataqueEscolhido == 4)
496               {
497                 if(cc.getDistanceX() <= AdistanceG[13])
498                   cc.commandCall(callsG[13]);
499                 if(cc.getDistanceX() <= AdistanceG[7])
500                   cc.commandCall(callsG[7]);
501                 else
502                   escolhaPadrao();
503               }
```

```
504             else if(ataqueEscolhido == 5)
505             {
506               if(cc.getDistanceX() <= AdistanceG[13] && cc.
                     getMyEnergy() > 30)
507                 cc.commandCall(callsG[13]);
508               else if(cc.getMyCharacter().getState().name().
                     equals("STAND") || cc.getMyCharacter().getState()
                     .name().equals("CROUCH"))
509                 cc.commandCall("JUMP");
510               else
511                 cc.commandCall(callsG[22]);
512             }
513             else if(ataqueEscolhido == 6)
514             {
515               if(cc.getDistanceX() <= AdistanceG[7])
516                 cc.commandCall(callsG[7]);
517               else
518                 escolhaPadrao();
519             }
520             else if(ataqueEscolhido == 7)
521             {
522               if(cc.getDistanceX() <= AdistanceG[16] && cc.
                     getMyEnergy() > 50)
523                 cc.commandCall(callsG[16]);
524               if(cc.getDistanceX() <= AdistanceG[4])
525                 cc.commandCall(callsG[4]);
526               else
527                 escolhaPadrao();
528             }
529             else if(ataqueEscolhido == 8)
530             {
531               if(cc.getDistanceX() <= AdistanceG[16] && cc.
                     getMyEnergy() > 50)
532                 cc.commandCall(callsG[16]);
533               if(cc.getDistanceX() <= AdistanceG[4])
534                 cc.commandCall(callsG[4]);
535               else
536                 escolhaPadrao();
537             }
538             else if(ataqueEscolhido == 9)
539             {
540               if(cc.getMyCharacter().getState().name().equals("
                     STAND") || cc.getMyCharacter().getState().name().
                     equals("CROUCH"))
541                 cc.commandCall("JUMP");
542               else
543                 cc.commandCall(callsG[22]);
544             }
545             else if(ataqueEscolhido == 10)
546             {
547               if(cc.getMyCharacter().getState().name().equals("
                     STAND") || cc.getMyCharacter().getState().name().
                     equals("CROUCH"))
```

```
548                 cc.commandCall("JUMP");
549               else
550                 cc.commandCall(callsG[22]);
551             }
552             else if(ataqueEscolhido == 11)
553             {
554               if(cc.getDistanceX() <= AdistanceG[16] && cc.
                     getMyEnergy() > 50)
555                 cc.commandCall(callsG[16]);
556               if(cc.getDistanceX() <= AdistanceG[4])
557                 cc.commandCall(callsG[4]);
558               if(cc.getDistanceX() <= AdistanceG[7])
559                 cc.commandCall(callsG[7]);
560               else
561                 escolhaPadrao();
562             }
563             else if(ataqueEscolhido == 12)
564             {
565               cc.commandCall("GUARD");
566             }
567             else if(ataqueEscolhido == 13)
568             {
569               if(cc.getDistanceX() <= AdistanceG[4])
570                 cc.commandCall(callsG[4]);
571               else
572                 escolhaPadrao();
573             }
574             else if(ataqueEscolhido == 14)
575             {
576               if(cc.getDistanceX() <= AdistanceG[13] && cc.
                     getMyEnergy() > 50)
577                 cc.commandCall(callsG[13]);
578               else if(cc.getMyCharacter().getState().name().
                     equals("STAND") || cc.getMyCharacter().getState()
                     .name().equals("CROUCH"))
579                 cc.commandCall("JUMP");
580               else
581                 cc.commandCall(callsG[22]);
582             }
583             else if(ataqueEscolhido == 15)
584             {
585               if(cc.getDistanceX() <= AdistanceG[16] && cc.
                     getMyEnergy() > 50)
586                 cc.commandCall(callsG[16]);
587               if(cc.getDistanceX() <= AdistanceG[9])
588                 cc.commandCall(callsG[9]);
589               else
590                 escolhaPadrao();
591             }
592             else if(ataqueEscolhido == 16)
593             {
594               if(cc.getDistanceX() <= AdistanceG[16] && cc.
                     getMyEnergy() > 50)
```

```
595                 cc.commandCall(callsG[16]);
596               else
597                 cc.commandCall("GUARD");
598             }
599             else if(ataqueEscolhido == 17)
600             {
601               if(cc.getDistanceX() <= AdistanceG[9])
602                 cc.commandCall(callsG[9]);
603               else
604                 escolhaPadrao();
605             }
606             else if(ataqueEscolhido == 18)
607             {
608               if(cc.getDistanceX() <= AdistanceG[9])
609                 cc.commandCall(callsG[9]);
610               else
611                 escolhaPadrao();
612             }
613             else if(ataqueEscolhido == 19)
614             {
615               if(cc.getDistanceX() <= AdistanceG[9])
616                 cc.commandCall(callsG[9]);
617               else
618                 escolhaPadrao();
619             }
620             else if(ataqueEscolhido == 20)
621             {
622               if(cc.getDistanceX() <= AdistanceG[11])
623                 cc.commandCall(callsG[11]);
624               else
625                 escolhaPadrao();
626             }
627             else if(ataqueEscolhido == 21)
628             {
629               if(cc.getDistanceX() <= AdistanceG[11])
630                 cc.commandCall(callsG[11]);
631               else
632                 escolhaPadrao();
633             }
634             else if(ataqueEscolhido == 22)
635             {
636               if(cc.getDistanceX() <= AdistanceG[11])
637                 cc.commandCall(callsG[11]);
638               else
639                 escolhaPadrao();
640             }
641             else if(ataqueEscolhido == 23)
642             {
643               if(cc.getDistanceX() <= AdistanceG[11])
644                 cc.commandCall(callsG[11]);
645               else
646                 escolhaPadrao();
647             }
```

```
648             else if(ataqueEscolhido == 24)
649             {
650               if(cc.getDistanceX() <= AdistanceG[9])
651                 cc.commandCall(callsG[9]);
652               else
653                 escolhaPadrao();
654             }
655             else if(ataqueEscolhido == 25)
656             {
657               if(cc.getDistanceX() <= AdistanceG[11])
658                 cc.commandCall(callsG[11]);
659               else
660                 escolhaPadrao();
661             }
662             else if(ataqueEscolhido == 26)
663             {
664               if(cc.getDistanceX() <= AdistanceG[11])
665                 cc.commandCall(callsG[11]);
666               else
667                 escolhaPadrao();
668             }
669             else if(ataqueEscolhido == 27)
670             {
671               if(cc.getDistanceX() <= AdistanceG[11])
672                 cc.commandCall(callsG[11]);
673               else
674                 escolhaPadrao();
675             }
676             else if(ataqueEscolhido == 28)
677             {
678               if(cc.getDistanceX() <= AdistanceG[11])
679                 cc.commandCall(callsG[11]);
680               else
681                 escolhaPadrao();
682             }
683             else if(ataqueEscolhido == 29)
684             {
685               if(cc.getDistanceX() <= AdistanceG[8])
686                 cc.commandCall(callsG[8]);
687               else
688                 escolhaPadrao();
689             }
690             else if(ataqueEscolhido == 30)
691             {
692               if(cc.getDistanceX() <= AdistanceG[8])
693                 cc.commandCall(callsG[8]);
694               else
695                 escolhaPadrao();
696             }
697           }
698           else if(enemyChar == "LUD")
699           {
700             if(cc.getMyEnergy() > 300)
```

```
701             {
702               if(cc.getMyCharacter().getState().name().equals("
                     STAND") || cc.getMyCharacter().getState().name().
                     equals("CROUCH") && ataqueEscolhido<17)
703               {
704                 cc.commandCall(callsG[12]);
705               }
706             }
707             else if(ataqueEscolhido == 0)
708             {
709               if(cc.getDistanceX() <= AdistanceG[7])
710                 cc.commandCall(callsG[7]);
711               else
712                 escolhaPadrao();
713             }
714             else if(ataqueEscolhido == 1)
715             {
716               if(cc.getDistanceX() <= AdistanceG[7])
717                 cc.commandCall(callsG[7]);
718               else
719                 escolhaPadrao();
720             }
721             else if(ataqueEscolhido == 2)
722             {
723               if(cc.getDistanceX() <= AdistanceG[7])
724                 cc.commandCall(callsG[7]);
725               else
726                 escolhaPadrao();
727             }
728             else if(ataqueEscolhido == 3)
729             {
730               if(cc.getDistanceX() <= AdistanceG[7])
731                 cc.commandCall(callsG[7]);
732               else
733                 escolhaPadrao();
734             }
735             else if(ataqueEscolhido == 4)
736             {
737               if(cc.getDistanceX() <= AdistanceG[7])
738                 cc.commandCall(callsG[7]);
739               else
740                 escolhaPadrao();
741             }
742             else if(ataqueEscolhido == 5)
743             {
744               if(cc.getDistanceX() <= AdistanceG[13] && cc.
                     getMyEnergy() > 30)
745                 cc.commandCall(callsG[13]);
746               if(cc.getDistanceX() <= AdistanceG[1])
747                 cc.commandCall(callsG[1]);
748               else
749                 escolhaPadrao();
750             }
```

```
751             else if(ataqueEscolhido == 6)
752             {
753               if(cc.getDistanceX() <= 250)
754                 cc.commandCall("6");
755               if(cc.getDistanceX() <= 250)
756                 cc.commandCall(callsG[7]);
757               else
758                 escolhaPadrao();
759             }
760             else if(ataqueEscolhido == 7)
761             {
762               if(cc.getDistanceX() <= 250)
763                 cc.commandCall("6");
764               if(cc.getDistanceX() <= 250)
765                 cc.commandCall(callsG[7]);
766               else
767                 escolhaPadrao();
768             }
769             else if(ataqueEscolhido == 8)
770             {
771               if(cc.getDistanceX() <= AdistanceG[7])
772                 cc.commandCall(callsG[7]);
773               else
774                 escolhaPadrao();
775             }
776             else if(ataqueEscolhido == 9)
777             {
778               if(cc.getDistanceX() <= AdistanceG[7])
779                 cc.commandCall(callsG[7]);
780               else
781                 escolhaPadrao();
782             }
783             else if(ataqueEscolhido == 10)
784             {
785               if(cc.getMyCharacter().getState().name().equals("
                     STAND") || cc.getMyCharacter().getState().name().
                     equals("CROUCH"))
786                 cc.commandCall("JUMP");
787               else
788                 cc.commandCall(callsG[22]);
789             }
790             else if(ataqueEscolhido == 11)
791             {
792               if(cc.getDistanceX() <= AdistanceG[7])
793                 cc.commandCall(callsG[7]);
794               else
795                 escolhaPadrao();
796             }
797             else if(ataqueEscolhido == 12)
798             {
799               if(cc.getDistanceX() <= AdistanceG[7])
800                 cc.commandCall(callsG[7]);
801             }
```

```
802             else if(ataqueEscolhido == 13)
803             {
804               if(cc.getDistanceX() <= AdistanceG[13])
805                 cc.commandCall(callsG[13]);
806               else
807                 escolhaPadrao();
808             }
809             else if(ataqueEscolhido == 14)
810             {
811               if(cc.getDistanceX() <= AdistanceG[13])
812                 cc.commandCall(callsG[13]);
813               else
814                 escolhaPadrao();
815             }
816             else if(ataqueEscolhido == 15)
817             {
818               if(cc.getDistanceX() <= AdistanceG[9])
819                 cc.commandCall("6");
820               else
821                 escolhaPadrao();
822             }
823             else if(ataqueEscolhido == 16)
824             {
825               if(cc.getDistanceX() <= AdistanceG[3])
826                 cc.commandCall(callsG[3]);
827               else
828                 escolhaPadrao();
829             }
830             else if(ataqueEscolhido == 17)
831             {
832               if(cc.getDistanceX() <= AdistanceG[11])
833                 cc.commandCall(callsG[11]);
834               else
835                 escolhaPadrao();
836             }
837             else if(ataqueEscolhido == 18)
838             {
839               if(cc.getDistanceX() <= AdistanceG[11])
840                 cc.commandCall(callsG[11]);
841               else
842                 escolhaPadrao();
843             }
844             else if(ataqueEscolhido == 19)
845             {
846               if(cc.getDistanceX() <= AdistanceG[9])
847                 cc.commandCall(callsG[9]);
848               else
849                 escolhaPadrao();
850             }
851             else if(ataqueEscolhido == 20)
852             {
853               if(cc.getDistanceX() <= AdistanceG[8])
854                 cc.commandCall(callsG[8]);
```

```
855                 else
856                   escolhaPadrao();
857               }
858               else if(ataqueEscolhido == 21)
859               {
860                 if(cc.getDistanceX() <= AdistanceG[11])
861                   cc.commandCall(callsG[11]);
862                 else
863                   escolhaPadrao();
864               }
865               else if(ataqueEscolhido == 22)
866               {
867                 if(cc.getDistanceX() <= AdistanceG[9])
868                   cc.commandCall(callsG[9]);
869                 else
870                   escolhaPadrao();
871               }
872               else if(ataqueEscolhido == 23)
873               {
874                 if(cc.getDistanceX() <= AdistanceG[9])
875                   cc.commandCall(callsG[9]);
876                 else
877                   escolhaPadrao();
878               }
879               else if(ataqueEscolhido == 24)
880               {
881                 if(cc.getDistanceX() <= AdistanceG[9])
882                   cc.commandCall(callsG[9]);
883                 else
884                   escolhaPadrao();
885               }
886               else if(ataqueEscolhido == 25)
887               {
888                 if(cc.getDistanceX() <= AdistanceG[8])
889                   cc.commandCall(callsG[8]);
890                 else
891                   escolhaPadrao();
892               }
893               else if(ataqueEscolhido == 26)
894               {
895                 if(cc.getDistanceX() <= AdistanceG[9])
896                   cc.commandCall(callsG[9]);
897                 else
898                   escolhaPadrao();
899               }
900               else if(ataqueEscolhido == 27)
901               {
902                 if(cc.getDistanceX() <= AdistanceG[11])
903                   cc.commandCall(callsG[11]);
904                 else
905                   escolhaPadrao();
906               }
907               else if(ataqueEscolhido == 28)
```

```
908             {
909               if(cc.getDistanceX() <= AdistanceG[9])
910                 cc.commandCall(callsG[9]);
911               else
912                 escolhaPadrao();
913             }
914             else if(ataqueEscolhido == 29)
915             {
916               if(cc.getDistanceX() <= AdistanceG[8])
917                 cc.commandCall(callsG[8]);
918               else
919                 escolhaPadrao();
920             }
921             else if(ataqueEscolhido == 30)
922             {
923               if(cc.getDistanceX() <= AdistanceG[9])
924                 cc.commandCall(callsG[9]);
925               else
926                 escolhaPadrao();
927             }
928           }
929           else
930             escolhaPadrao();
931
932       }
933
934     }
935
936   }
937
938   private void escolhaPadrao()
939   {
940     if(cc.getMyEnergy()>= 300)
941       cc.commandCall(callsG[12]);
942     else if (cc.getMyEnergy()> 50)
943       cc.commandCall(callsG[16]);
944     else
945       cc.commandCall(callsG[(int) (Math.random()*16)]);
946   }
947 }
```