

# Modelo de um Software Robô para Ações Automáticas no Jogo de MMORPG Tibia

Cleyton Aparecido Dim  
Universidade Federal do Pará  
Belém, Pará, Brasil  
cleytondim@ufpa.br

Marcelle Pereira Mota  
Universidade Federal do Pará  
Belém, Pará, Brasil  
mpmota@ufpa.br

Rafael Martins Feitosa  
Universidade Federal do Pará  
Belém, Pará, Brasil  
rafaelmf@ufpa.br

Jefferson Magalhães de Moraes  
Universidade Federal do Pará  
Belém, Pará, Brasil  
jeffersonmoraes@ufpa.br

## ABSTRACT

This paper presents the model of a robot software capable of doing some functions inside the online role-playing game Tibia automatically while avoiding the Battleye anti-cheating system and show countermeasures that can help Battleye and Tibia developers, as well as developers of similar games to prevent this type of bot from surpassing their anti-cheating system. Among these functions are the hunting, the healing, and the looting. Specifically, there is an explanation of the game's mechanics and how the robot surpasses the anti-cheating system by analyzing the game screen and simulating mouse and keyboard functions, without doing any memory injection or memory reading. The evaluation of the bot was done during six months on ten accounts with different variables such as the character's vocation, daily playing time, account status, and whether the character is hunting in the same place every day or variation occurs.

## KEYWORDS

Software Robô, MMORPG, Análise de Imagens

## 1 INTRODUÇÃO

Os jogos online atuais frequentemente sofrem com softwares que dão vantagens a alguns jogadores. Estes softwares podem operar para prover visão não autorizada de setores do jogo, transparência em paredes, mira automática, movimentos repetitivos ou mesmo robôs que jogam no lugar de uma pessoa enquanto esta fica longe do computador.

Existem esforços por parte dos desenvolvedores de jogos para mitigar o problema das trapaças, os quais envolvem um sistema anti-trapaça interno no jogo, softwares analíticos de terceiros ou uma combinação de ambos. Entre estes softwares analíticos de terceiros, temos como exemplo Valve Anti-cheat, GameGuard, e Battleye[6].

Considerando os vários tipos de jogos, jogos de RPG Online são possivelmente os mais desbalanceados entre jogadores que não usam software de trapaça e os que os usam, uma vez que estes jogos são contínuos e cumulativos nos dois mais importantes estados de evolução de um jogador: nível e ouro[8, 15].

Um destes jogos de role-playing online é chamado de Tibia. Este game sofreu por muitos anos com a infestação de jogadores usando softwares de trapaça; Mas em 2017 a CipSoft, desenvolvedora do

Tibia, decidiu unir forças com o Battleye para eliminar as trapaças no jogo. Esta estratégia funcionou até o surgimento de novos softwares robôs que podem escapar do radar do Battleye. Agora, em 2020, alguns robôs estão operando novamente, uma vez que muitas reclamações podem ser vistas nos fóruns oficiais do jogo, bem como pode ser experienciado no próprio jogo em observações nos locais vantajosos de caçadas.

Este artigo tem como objetivo apresentar um dos possíveis métodos atualmente utilizados para prevenir a detecção de softwares de trapaça no jogo Tibia por parte do Battleye. Também apresenta todo o processo de construção de um software robô para trapacear no Tibia, possibilitando de forma automática a caçada (luta contra os monstros no jogo), a cura (restauração da vida quando prestes a se esgotar), e o saque dos monstros mortos (obtenção dos itens e ouro que monstros mortos carregavam), para subir o nível do personagem e adquirir recursos financeiros. O software desenvolvido é completamente privado e não é comercializado. O principal propósito do artigo é puramente uma demonstração acadêmica para mostrar pontos fracos na detecção de trapaças no Tibia, para que estas informações auxiliem o Battleye e os desenvolvedores de jogos, especialmente do Tibia, a desenvolver formas mais efetivas de frear o uso de softwares ilegais usados para trapacear. Para reforçar este auxílio, são apresentadas possíveis contramedidas para detectar e combater o uso do software robô apresentado neste artigo.

O artigo está organizado da seguinte forma: A Seção 2 é dedicada aos trabalhos relacionados. Seção 3 destina-se a detalhar o jogo Tibia. A Seção 4 descreve os métodos antigamente usados nos softwares de trapaça antes da introdução do Sistema Battleye e como este sistema os detecta. Na Seção 5 são descritos métodos que evitam a detecção do Battleye e apresenta o processo de construção do software robô que utiliza este método. A seção 6 apresenta a avaliação do uso do software robô em diferentes contas e cenários. Na Seção 7 são mostradas contramedidas para auxiliar o sistema Battleye a bloquear o tipo de software robô apresentado neste artigo, bem como auxiliar os desenvolvedores de jogos a detectar ações incomuns, que normalmente não são realizadas por jogadores humanos. Por fim, a Seção 8 conclui o artigo com as considerações finais e discussões sobre trabalhos futuros.

## 2 TRABALHOS RELACIONADOS

Trapaças em jogos online para ganhar vantagens, superar outros jogadores ou realizar movimentos repetitivos que podem ser tediosos

mas que trazem ganhos no jogo tornaram-se bastante difundidas, e existem pesquisas recentes[7, 20] que estudam estes comportamentos. Por parte das companhias desenvolvedoras de jogos, há um grande interesse em barrar o uso de robôs em jogos online massivos devido aos efeitos na comunidade e no ambiente do jogo. Muitos trabalhos científicos tentam auxiliar neste processo de detecção e interrupção do uso de robôs nestes jogos.

Alguns trabalhos tem foco na análise do comportamento dos jogadores[9, 21], buscando ações que não são semelhantes a ações humanas. Outros focam na análise do tráfego de dados [4], comparando os dados gerados por jogadores normais e por robôs. Há um trabalho em particular[10] que descreve uma grande parte dos métodos anti-trapaça, mencionando os sistemas de vigilância em nível de kernel, como é o caso do Battleye.

Antes da parceria dos desenvolvedores do jogo Tibia com o Battleye, haviam diversas iniciativas de desenvolvimento comercial de softwares robôs para o jogo, das quais surgiram por exemplo TibiaNG e BlackD[16], que usavam mecanismos para leitura e escrita de memória e chamada de funções. Estes mecanismos são discutidos na literatura[3] mas com os métodos atuais são facilmente detectáveis. Atualmente, alguns softwares robôs como o OldBot[13] e WindBot[22] conseguem evitar o sistema Battleye fazendo uso de captura e análise de imagem. Mas como iniciativas comerciais, eles necessitam de uma camuflagem fazendo uso de máquinas virtuais para executar o jogo, enquanto o software robô é executado no sistema operacional principal. Esta abordagem tem sido combatida pelos desenvolvedores do Tibia através do bloqueio do acesso ao jogo através de máquinas virtuais, mas os desenvolvedores dos robôs estão sempre fazendo atualizações que permitem a execução do jogo em máquinas virtuais. A abordagem de software robô desenvolvida neste trabalho se refere a um software privado, sem a necessidade de camuflagem especial, visando demonstrar as possibilidades de evitar os sistemas anti-trapaças.

### 3 O JOGO TIBIA

Tibia é um típico jogo de interpretação de papéis online, multijogador em massa (MMORPG) [5]. Estes jogos são uma transição de jogos de interpretação de papéis (RPG) de mesa para o ambiente digital[17], onde milhares de jogadores se encontram simultaneamente para aprimorar seus personagens e executarem atividades juntos, com as ações de cada jogador podendo afetar a jogabilidade dos demais. A Figura 1 mostra o ambiente do jogo Tibia.

O ponto 1 é a representação do personagem e o ponto 2 é a tela principal do jogo, onde o mundo do Tibia, os jogadores e os monstros são representados. O ponto 3 é o minimapa do jogo, onde o jogador pode verificar em que ponto do mundo ele está, além de poder ser usado como uma das opções para se mover por este mundo. Os pontos 4 e 5 são respectivamente os equipamentos e o estado do jogador, com o estado sendo dividido entre pontos de vida (hit points) e pontos de mana (mana points). Há uma lista de batalha que mostra monstros e outros personagens no ponto 6, além exibição de itens e ouro no ponto 7. Por fim, o ponto 8 mostra as ações que podem ser executadas por um personagem, ativadas por teclas de atalho (hotkey).

Tibia é um jogo gratuito, mas com a maior parte do seu conteúdo sendo acessado somente com contas pagas, chamadas de Premium

Accounts. Estas contas acessam conteúdo exclusivo e seus personagens tem diversos benefícios, entre os quais destaca-se a evolução para uma classe superior.

### 3.1 Classes de Personagens

O jogo possui 4 classes de personagens principais, chamadas de vocações, cada uma com suas próprias peculiaridades: Knight, Paladin, Sorcerer e Druid.

O Knight é a classe guerreira, possui grande habilidade no combate corpo a corpo e usam com maestria escudos, machados, clavas e espadas. Possuem a maior capacidade de carregar itens e a maior quantidade de pontos de vida. Como desvantagem, eles possuem baixa habilidade mágica e baixa capacidade de combate à distância.

O Paladin é a classe arqueira, que executa ataques com arco e flechas, bestas, lanças e outras armas arremessáveis. Eles possuem uma razoável capacidade de carregar itens, bem como uma quantidade considerável de pontos de vida. Podem usar escudos em algumas situações e possuem habilidade mágica moderada.

O Sorcerer é a classe de magos de ataque, que possuem alto poder mágico destrutivo e grandes poderes de cura. Eles são os mestres da magia negra e podem causar grandes danos em grandes áreas, atingindo muitos inimigos simultaneamente. Como pontos negativos, esta classe possui baixa defesa, baixa capacidade de carregar itens e pouquíssimos pontos de vida.

Por fim, o Druid também é uma classe de magos, mas com foco maior nos poderes de cura, além de também possuírem um considerável poder mágico destrutivo de grandes áreas. Eles também sofrem com os pontos negativos de baixa defesa, baixa capacidade de carregar itens e pouquíssimos pontos de vida.

Quando o jogador possui uma premium account, cada uma destas vocações podem evoluir para Elite Knight, Royal Paladin, Master Sorcerer e Elder Druid, respectivamente. Estas evoluções fornecem magias novas e únicas, além de desfrutarem de uma regeneração mais rápida dos pontos de vida de mana.

### 3.2 Mecânica do Jogo

Tibia é um jogo de imagens bidimensionais (2D game) com uma jogabilidade simples e uma história complexa. O principal objetivo do jogador é aumentar seus níveis e completar os desafios do jogo.

Para aumentar o nível dos personagens, os jogadores devem caçar monstros para obter pontos de experiência. Eles deverão vagar pelo mundo do Tibia procurando por estes monstros, atacando-os até que sejam eliminados. Existem incontáveis monstros com diferentes dificuldades. Quanto mais poderoso for o monstro, mais pontos de experiência serão obtidos. Muitos destes monstros não podem ser derrotados por apenas um jogador, necessitando de um grupo que dividirá a experiência obtida. Assim como há diferentes classes de personagens, há diferentes tipos de monstros que atacam corpo a corpo, à distância ou com magia, cada qual sendo mais facilmente derrotado por uma vocação específica.

Ao batalhar com um monstro, jogadores devem atacar usando as ações das hotkeys após selecionar o monstro na lista de batalha ou clicar no monstro na tela principal. Excepcionalmente, Sorcerers e Druids podem usar hotkeys de magias de área e clicar em um ponto da tela para atingir uma quantidade maior de monstros.



Figura 1: Tela do Ambiente do Tibia

Durante uma batalha, o jogador deve curar seus pontos de vida, pois uma vez que estes pontos sejam reduzidos a zero, o personagem morrerá e perderá níveis e habilidades. Esta cura pode ser feita através de poções ou magias, ou o jogador pode se afastar do monstro, aguardar algum tempo em local seguro enquanto consome alguma comida no jogo. Nesta situação os pontos de vida se recuperarão gradualmente ao longo do tempo. Além disto, se o personagem estiver usando magias, os pontos de mana devem ser restaurados através de poções específicas, caso contrário, as magias não serão disparadas pois necessitam de pontos de mana, que são consumidos de acordo com a magia usada. Este processo de cura por poções ou magia, e a restauração de mana por poções pode ser realizado usando uma hotkey de açãopropriada para cada caso.

O sistema de hotkey é a simplificação de ações específicas realizadas através do teclado. O jogador atribui cada ação desejada a uma tecla, normalmente F1 a F12, ou uma combinação de CTRL ou SHIFT + F1 a F2, moldando assim as teclas de atalho da forma que lhe for mais prática.

Após a derrota de um monstro no jogo, ele deixará recompensas que podem ser itens ou ouro. O jogador pode saquear esta recompensa clicando no corpo do monstro derrotado, o que fará com que a recompensa desejada vá automaticamente para a bolsa do personagem se ele possuir a capacidade de carga necessária.

#### 4 TRAPAÇAS E O BATTLEYE

Antes da parceria entre a CipSoft e o Battleye, o jogo era dominado por trapaceiros. Em cada local que um jogador honesto fosse para fazer suas caçadas, encontraria um ou mais trapaceiros utilizando softwares robô, impossibilitando este jogador de fazer uma caçada efetiva, pois um local de caça disputado diminui a oferta de experiência pela falta de monstros, que são eliminados mais rapidamente

e levam algum tempo para renascer. Além disto, caso um único monstro esteja em disputa, apenas o personagem que infligir mais dano poderá saquear o monstro. Assim, além de obter menos pontos de experiência nas caçadas, os jogadores honestos obtém menos lucro em razão das disputas com jogadores utilizando softwares que automatizam as ações, sem poder dialogar com os mesmos e chegar a algum consenso sobre alguma divisão do local de caça.

Para tentar frear as ações dos trapaceiros, a CipSoft havia desenvolvido um sistema interno anti-trapaças<sup>[14]</sup> e muitos jogadores desonestos foram banidos, com suas contas deletadas, mas não o suficiente para ter um efeito positivo na comunidade de jogadores. A abordagem desta detecção de robôs utilizada é possivelmente similar ao proposto em uma pesquisa de mestrado<sup>[21]</sup>. Além disto, foi desenvolvida uma funcionalidade no jogo onde os jogadores podem reportar outros que estejam visivelmente utilizando softwares robôs para jogar. Estes reportes são então analisados caso a caso manualmente pela empresa.

Nesta época, existiam basicamente dois tipos de softwares de trapaça: os de injeção de código no software do jogo, e os de simulação de mouse e teclado.

Os softwares de injeção de código consistem na geração do código desejado em formato de bibliotecas (DLLs a serem usadas por métodos específicos do sistema) ou em formato de código binário puro (o qual deve ser inserido em espaços reservados de memória do jogo)<sup>[6]</sup>. Este código é responsável por executar as ações desejadas automaticamente ao invés do jogador real, e deve ser complementado pela leitura da memória do jogo, obtendo informações que guiarão a tomada de decisão.

Já os softwares baseados em simulação de teclado e mouse são mais simples de se construir, necessitando apenas efetuar a leitura da memória do jogo e externamente executar os cálculos necessários

para a tomada de decisão. Tomadas as decisões, o software então executa a ação por comandos de mouse e teclado. Embora eles tenham uma relativa restrição de funcionalidades em comparação com a injeção de código, eles são mais difíceis de detectar por não interferirem na operação original do jogo e reproduzem ações mais parecidas com as que um humano tomaria, uma vez que, com injeção de código, seria por exemplo enviado um comando de caminhar até certo ponto do mapa do jogo, sem utilização do mouse ou teclado, o que é impossível para um jogador sem utilizar meios ilícitos.

Com a chegada do Battleye no Tibia, tornou-se impraticável usar softwares com estes métodos de trapaça, uma vez que o Battleye detecta e desabilita sistemas que executam injeção de código ou leitura da memória do jogo. Adicionalmente, o Battleye é capaz de executar uma varredura dinâmica, completa e permanente em modo de usuário e modo de kernel, identificando programas ilícitos conhecidos ou blocos de código binário previamente conhecidos[1]. Isto significa que se um programa tentar injetar código ou ler a memória do jogo, esta tentativa será detectada. Além disto, se de alguma forma um programa puder automatizar ações sem algum fazer uso de algum destes artifícios, e este programa vier a ser público gratuito ou comercialmente, ele será marcado como um software de trapaça cedo ou tarde, e consequentemente será bloqueado. Desta maneira, apenas programas sem intrusão no software do jogo e que ou sejam privados ou tenham alguma forma de camuflagem podem ter sucesso em evitar o Battleye para trapacear.

Algumas pesquisas[12, 18] sugerem que o uso de virtualização oferecida por hardware recente poderia impedir o acesso direto a dados da memória do jogo, uma vez que estes dados estariam mais ocultos em uma sandbox sem conexão direta com o sistema operacional. Mas uma vez que o Battleye é capaz de identificar acesso aos dados restritos do jogo na memória do computador, o uso desta tecnologia não é necessário.

## 5 EVITANDO O BATTLEYE NO TIBIA

Se não é possível injetar código ou mesmo fazer leitura de dados básicos da memória do Tibia, como pode ser desenvolvido um software robô para automatizar ações de caça, cura e saque sem ser detectado pelo Battleye?

Um caminho para obter dados relevantes do jogo é por meio da captura de tela do computador e não diretamente do jogo em si. Com a captura de tela, pode-se obter dados com técnicas de análise de imagens, auxiliadas por processamento de imagem quando necessário. Todos os dados necessários para uma decisão humana no jogo estão contidos na tela capturada, que é exatamente o que um jogador humano estaria visualizando enquanto joga.

Com a imagem capturada, após uma análise apropriada é possível obter dados muito importantes para as tomadas de decisão, como a existência e posição de monstros na tela, a quantidade de pontos de vida e pontos de mana de um personagem, a posição de um personagem no mundo e o estado de batalha do personagem (se ele está ou não atacando um monstro no momento). Com estas informações, é possível executar as atividades básicas do jogo, pois com os pontos de vida e mana pode ser tomada a decisão de quando usar uma poção ou magia para se curar, com a existência de um monstro na tela e o estado de batalha pode ser tomada a decisão de atacar e de saquear o monstro após matá-lo, e com a posição do

personagem no mundo obtém-se a orientação para a movimentação do personagem.

Uma vez que as decisões sejam tomadas após a análise de imagens, as ações devem ser executadas sem realizar chamadas de funções internas do código do jogo. Ações humanas devem ser simuladas, as quais são baseadas no uso de teclado e mouse. Isto é obtido simulando o uso destes dispositivos através das funções do próprio sistema operacional. Assim, tanto para o Tibia quanto para o Battleye, haverá a impressão de que um jogador estará pressionando o teclado e movimentando o mouse. Com um software nestes moldes, a detecção de trapaça é evitada enquanto o software é mantido privado sem exposição pública. Caso se torne público, o software seria eventualmente inserido em uma lista negra e bloqueado pelo Battleye.

### 5.1 Construindo o Software Robô

O software robô foi desenvolvido em linguagem Java, usando a biblioteca Robot para capturar a tela do computador e simular o uso de mouse e teclado. A linguagem Java pode prover um nível adicional de segurança contra a detecção do Battleye, uma vez que é uma linguagem interpretada que não executa código binário diretamente no sistema operacional, mas através da Máquina Virtual Java. O software nem mesmo aparece com seu nome original na lista de processos do gerenciador de tarefas do Windows, como mostrado na Figura 2.

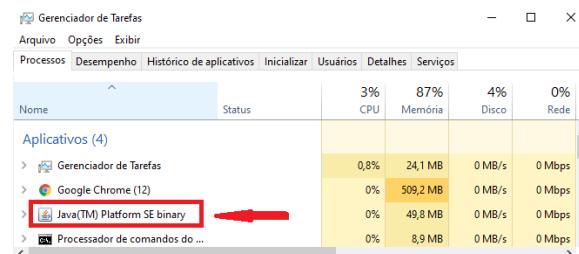


Figura 2: O Software no Gerenciar de Tarefas do Windows

As capturas de tela são realizadas a cada 100 milissegundos, pois decisões devem ser tomadas rapidamente, e todo o processamento para analisar as imagens pode levar outra centena de milissegundos. Sendo que o mais importante no jogo é não deixar o personagem morrer, uma thread foi dedicada para a cura do personagem, que pode ser realizada enquanto outras análises na imagem são feitas. A Figura 3 mostra o código Java para a captura de tela.

Duas informações a respeito do número de pontos de vida e de pontos de mana são exibidas na tela do jogo: barras coloridas e números. Uma análise dos números requer algoritmos de reconhecimento ótico de caracteres (OCR), o que poderia ser obtido

```
import java.awt.Robot;
Robot robot = new Robot();
BufferedImage image;
image = robot.createScreenCapture(
    new Rectangle(Toolkit.getDefaultToolkit().getScreenSize()));
```

Figura 3: Código Java - Biblioteca Robot para screenshots

utilizando por exemplo a biblioteca Tesseract[19] em trechos específicos da tela capturada. Porém, optou-se por uma abordagem mais simples, analisando o percentual de coloração das barras de estado. O tamanho da barra de cor é obtida e comparada com o que seria o estado da barra em tamanho completo, que seria a situação de pontos de vida totalmente cheios, e assim obtido o percentual atual. Caso este percentual esteja abaixo do limite estabelecido no software robô, uma ação de teclado correspondente à tecla de atalho associada à cura é executada. A Figura 4 apresenta a situação em que o robô verifica que o percentual de hit points está abaixo de 50%, de maneira que um evento de pressionamento da tecla “F1” será disparado, correspondente à ação de cura. Apresenta também o código java para esta análise.

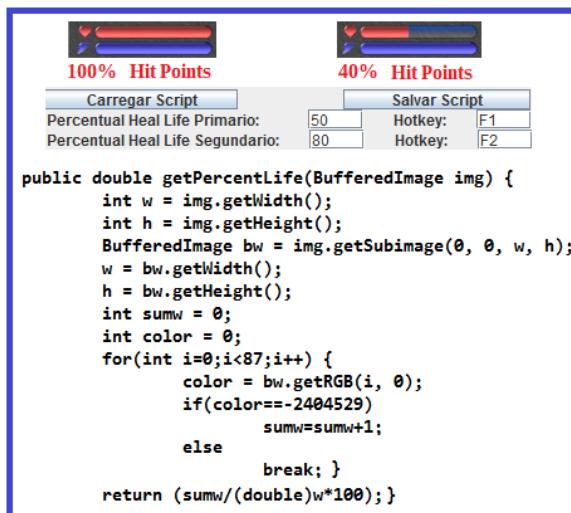


Figura 4: Hit points em 40% / Código Java

Para encontrar monstros, o personagem deve caminhar pelo mundo do Tibia. No software robô, marcadores são feitos para informar quais os pontos do minimapa devem ser clicados, além do tempo em segundos para clicar no próximo ponto do mapa. Este tempo é definido de acordo com o tempo esperado para se chegar ao destino. Os marcadores são feitos seguindo o caminho desejado uma vez, acionando a opção de inserir um ponto de caminhada (walkpoint). Uma pequena porção do minimapa será capturada, e esta parte capturada é buscada pixel a pixel em um momento posterior, quando se está verificando se o próximo ponto de caminhada está na tela. Desta forma, o próximo ponto deve estar visível a partir do ponto anterior, caso contrário, ele não será encontrado pois seus pixeis não estarão na tela capturada. Também é possível adicionar um ponto de emergência, que pode ser usado para evitar situações onde ações de outros jogadores fazem o personagem sair de sua rota original e não conseguir visualizar nenhum ponto de caminhada disponível. Se um ponto de emergência é encontrado, o computador é desligado e o personagem será desconectado, prevenindo possíveis mortes, pois fora da rota original o personagem ficará parado e suscetível a ser empurrado até áreas mais perigosas. A Figura 5 mostra o minimapa do jogo, as marcações feitas no software robô e a rota circular a ser percorrida pelo personagem.

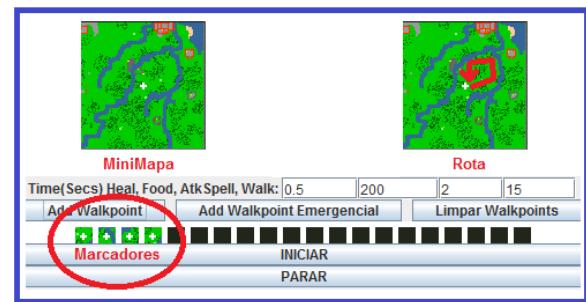


Figura 5: MiniMapa e a Rota por Marcadores

Uma vez que o personagem encerra um ciclo de ações e deve se mover até a próxima localização marcada, uma verificação de cores pixel a pixel é feita no minimapa para verificar se o bloco marcado está presente no atual minimapa (situação em que todos os pixeis na marcação correspondem a um setor do minimapa). Se estiver, um clique do mouse é simulado no ponto correspondente ao centro do marcador, caso contrário, o próximo marcador será analisado.

Enquanto está caminhando pelo mundo, o personagem pode encontrar monstros. Se encontrar, ele será exibido na lista de batalhas e pode ser atacado por meio de cliques ou teclas de atalho. O software robô verifica se em um determinado grupo de pixel existem pontos brancos. Caso existam, significa que há o nome de um monstro na lista, e então o estado de batalha do personagem é marcado como batalhando, e a tecla de atalho de ataque será acionada. A Figura 6 mostra a diferença entre a lista de batalha sem monstros e com monstros, bem como o código java para fazer a devida análise.

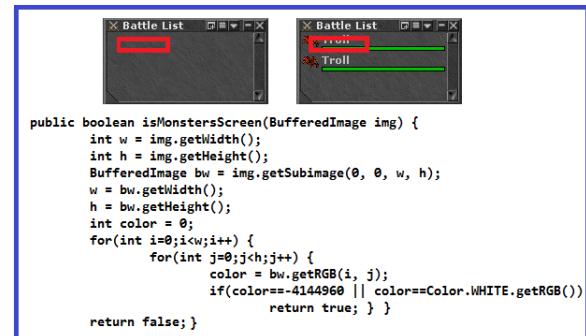


Figura 6: Battle List sem e com monstros / Código Java

Quando é atacado, o monstro é desenhado no jogo com um quadrado vermelho o cercando, tanto na lista de batalha quanto na tela principal do jogo. Este quadrado vermelho é um ponto chave para o software robô pois, ao combinar o estado de batalha do personagem com a existência de um quadrado vermelho na lista de batalha, é possível saber se o personagem atacou um monstro e já o matou (caso em que podem existir outros monstros na lista sem o quadrado vermelho), ou se atacou o monstro e um quadrado vermelho está presente na lista (situação em que a batalha ainda está acontecendo). Se o jogador já matou o monstro, então seu estado

de batalha volta a ser falso, e o personagem pode então saquear o corpo do monstro clicando sobre ele. A exata localização para o clique será a coordenada (x, y) da tela principal do jogo com a ultima ocorrência de um quadrado vermelho. Esta coordenada pode ser obtida com auxílio da biblioteca OpenCV no Java, pelos métodos de extração e matching de features do algoritmo Scale-Invariant Feature Transform (SIFT)[11] ou similares. Primeiramente é extraído o canal vermelho da imagem da tela do jogo, de modo que o quadrado vermelho desejado tenha cor mais próxima do branco. Em seguida esta imagem em escala de cinza é convertida para binária com limiar 0-250 para preto e 251-255 para branco. O resultado é a obtenção de alguns raros pontos brancos que teriam a mesma tonalidade de vermelho do quadrado, bem como feições aproximadas do quadrado vermelho original. Uma parte deste quadrado obtido é usado como referência de busca para matching de features no SIFT, que calcula características nesta imagem de referência e verifica se há características semelhantes em cada imagem obtida posteriormente durante uma caçada. Uma vez que são encontradas características iguais, obtém-se as coordenadas do ponto referente àquela característica, calculando-se onde deve ser enviado o comando de clique para saquear o monstro. A Figura 7 apresenta estas etapas.

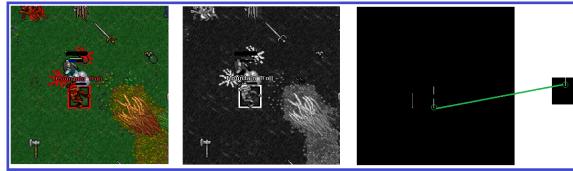


Figura 7: Etapas da obtenção de coordenadas de um monstro

Se o monstro tiver se movido após a ultima captura de tela e por acaso morrer antes da próxima captura de tela, em um intervalo que varia entre 100 a 200 milissegundos considerando o tempo de processamentos, o personagem poderá clicar na coordenada errada, uma vez que o corpo do monstro poderá não estar lá.

Até este ponto, o software robô é capaz então de caçar, saquear, curar e caminhar. Algumas funções extras foram adicionadas, como a checagem da existência de um anel e munições nos itens equipados, disparando uma tecla de atalho se algum item precisar ser reposto, bem como ações de se alimentar a cada intervalo de tempo especificado. Também há uma opção de atacar com magias, com definição do intervalo de tempo entre os ataques. Quando a opção de atacar com magias é usada, durante uma batalha a ação da magia é constantemente acionada, de acordo com o tempo especificado. O resultado final do software é mostrado na Figura 8.

## 6 AVALIAÇÃO

O software robô desenvolvido foi testado durante 6 meses, em 10 contas com personagens diferentes: 4 knights, 2 paladins, 2 sorcerers e 2 druids. A maior parte destas contas eram premium account, onde o jogador paga uma quantia mensal para desfrutar de todo o conteúdo do jogo. Todas as contas estiveram online em diferentes perfis: 24 horas por dia, 12 horas por dia e 6 horas por dia. Além disto, houve uma divisão na qual metade das contas caçaram no mesmo local todos os dias, e a outra metade variava o local de caçada a cada dia, sem repetir nenhum local durante a semana.

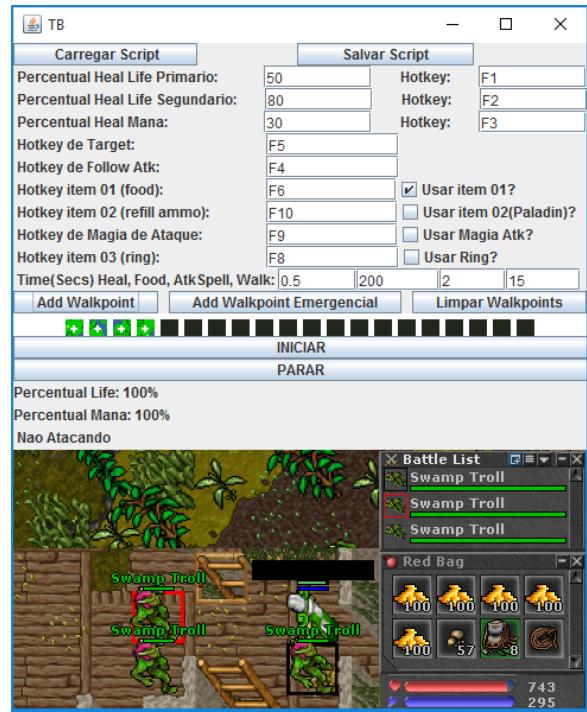


Figura 8: O Software Robô Desenvolvido

Apenas os knights ficaram online 24 horas diárias nos testes, devido à jogabilidade desta vocação que não requer munições ou muitos suprimentos de mana e vida, podendo permanecer no campo de batalha por longos períodos. As demais vocações, embora possam ser automatizadas, devem ser supervisionadas com o passar das horas, verificando se não estão sem suprimentos. Magos (sorcerers e druids) foram exclusivamente premium accounts, uma vez que ambos são desvantajosos sem conteúdo e magias adicionais. Os magos em contas gratuitas não possuem acesso à magias de ataque.

Entre as 10 contas, 3 foram banidas, duas das quais estiveram online 24 horas por dia, sendo banidas em menos de 2 meses. A outra conta banida esteve online por 12 horas diárias, sempre caçando no mesmo local, sendo banida em 3 meses. Informações sobre vocações, tempo online, estado da conta, estilo de caçada e verificação de banimento são mostradas na Tabela 1. É importante ressaltar que todos os banimentos por utilização de softwares de trapaça no Tibia são permanentes, com a conta sendo deletada, não havendo qualquer feedback sobre os motivos que levaram a empresa a concluir que os personagens estiveram usando softwares de trapaça.

As variáveis foram escolhidas em razão de sua ligação com a possibilidade de banimento. A primeira variável é a vocação do personagem, que dita todos os movimentos e ações no jogo. Cada vocação tem um modo diferente de ser jogado, e um estilo de jogo similar entre elas, como é o caso deste software robô, pode levar suspeitas no caso de uma análise minuciosa. Um exemplo de caso suspeito poderia ser o paladin, que deve manter distância dos monstros, mas com o uso deste software permanece próximo dos monstros o tempo todo, recebendo dano desnecessário. Nesta versão do software não há diferença na jogabilidade entre as vocações.

**Tabela 1: Tabela de Avaliação - Teste de Seis Meses**

Vocação	Tempo/Dia	Premium?	Mesmo Local	Ban.
Knight	24h	Sim	Não	Sim
Knight	24h	Não	Sim	Sim
Knight	12h	Não	Não	Não
Knight	6h	Sim	Sim	Não
Paladin	12h	Sim	Sim	Sim
Paladin	6h	Não	Sim	Não
Sorcerer	12h	Sim	Sim	Não
Sorcerer	6h	Sim	Não	Não
Druid	12h	Sim	Sim	Não
Druid	6h	Sim	Não	Não

A segunda variável é talvez a mais importante: O tempo diário de jogo. Tempos diferentes de jogo diário foram testados, para verificar se o sistema de análise interno da desenvolvedora do jogo realmente identifica comportamentos não humanos. A terceira variável se refere a se uma conta é premium ou não. Ela foi usada para verificar se banimentos poderiam ocorrer se um jogador paga a taxa mensal, o que poderia resultar em menos lucro para a companhia.

A ultima variável é relacionada à variação dos locais de caçada. Se um personagem caça repetidamente no mesmo local por várias semanas, isto poderia registrar nos sistemas de análise movimentos muito repetitivos, pois o software robô clicaria nos mesmos pontos do mapa todos os dias. Embora isto não necessariamente é definido como comportamento não humano, é improvável que uma pessoa seja capaz de clicar consistentemente nas mesmas coordenadas de pixel todos os dias. Além disto, ficar no mesmo local por várias horas todos os dias torna o personagem vulnerável a uma grande quantidade de reportes de outros jogadores, aumentando a chance de uma análise minuciosa do comportamento do personagem.

Como verificado nos testes, personagens utilizando o software robô por 24 horas todos os dias são possivelmente marcados pelo sistema interno de análise onde são verificados comportamentos não humanos. Como não é humanamente possível permanecer acordado por vários dias consecutivamente, e uma das regras do jogo é não compartilhar a conta com outras pessoas, estes banimentos foram esperados. Um terceiro banimento, com um personagem paladin jogando 12 horas todos os dias no mesmo local pode ter ocorrido através de uma análise manual após reportes de outros jogadores observando o personagem realizando movimentos repetitivos sem responder perguntas ou interagir com os jogadores. Portanto, aparentemente o software robô desenvolvido não é detectado caso sejam seguidas algumas precauções, como: não utilizar por mais horas diárias do que o humanamente possível, evitar áreas de caças muito populosas e rotacionar os locais de caça, sem repetir o mesmo lugar todos os dias. O fato de ser ou não uma premium account não parece impactar nas decisões de banimento da companhia.

## 7 PREVENINDO TRAPAÇAS COM ESTE SOFTWARE ROBÔ

Uma vez que o propósito deste artigo é demonstrar a possibilidade de um software robô evitar o atual sistema de detecção Battleye, e a

proliferação destes softwares não é desejável, os modos apropriados para barrar este software robô são aqui descritos.

Como o software é escrito em Java, o código não possui identificação explícita quando é executado. Desta maneira, o programa em si não pode ser colocado na lista negra do Battleye. Mas o software robô usa funções do sistema operacional para realizar capturas da tela do jogo e enviar comandos de teclado e mouse para o jogo. Como uma grande quantidade de capturas de tela deve ser obtida em um espaço de tempo curto para que os dados do jogo estejam tão atualizados quanto possível (10 capturas por segundo neste software robô) e não há uma razão plausível para que o jogador execute estas capturas por longos períodos, o sistema Battleye poderia analisar se a função de capturar tela do sistema operacional é chamada com uma frequência anormal, também verificando se as funções para simulação de teclado e mouse são chamadas, com esta sendo uma clara ocorrência de um jogador usando um sistema de trapaça. Deste modo, é possível fechar a aplicação Java que está realizando as chamadas destas funções e bloquear o acesso à conta do jogador de acordo com as regras do jogo Tibia.

Além disto, o sistema de análise interna de comportamentos não humanos poderia ser melhorado para considerar movimentos repetitivamente precisos como potencialmente não humanos, para então realizar uma análise posterior minuciosa. Entre estes movimentos precisos, pode-se destacar sempre clicar nos mesmos pixels do mapa, sempre se curar com um percentual específico dos pontos de vida, atacar os monstros com intervalos de tempo sempre semelhantes e movimentar o mouse em uma rota uniforme. A ocorrência destes fatos em grande parte do tempo de jogo de um personagem torna bastante provável a utilização de software de trapaça, incluindo o descrito neste trabalho.

## 8 CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

O software robô para ações automáticas no jogo Tibia foi desenvolvido com sucesso. O software pode efetivamente evitar o sistema de detecção Battleye através da captura e análise de imagens, associado com a simulação de ações por teclado e mouse, evitando banimento da conta sob circunstâncias específicas. Como prova deste fato, o software foi usado por 10 personagens, com pelo menos dois de cada vocação, por 6 meses em diferentes cenários, e 3 banimentos ocorreram. Estes banimentos foram possivelmente devido à configuração de uso na qual alguns personagens ficaram online 24 horas diárias por várias semanas, algo humanamente impossível. Um dos banimentos tinha um perfil onde estava sempre caçando no mesmo local, sendo vulnerável a reportes diários de outros jogadores e então sendo analisado manualmente pela desenvolvedora do jogo.

Embora desenvolvido especificamente para o jogo Tibia, o processo de desenvolvimento pode ser implementado em jogos 2D similares, como Zezenia[23] e Bloodstone[2], ou mesmo executar pequenas ações específicas em jogos 3D, uma vez que o maior fator para bloquear softwares de trapaça atualmente é a análise de interferência interna da aplicação de um jogo, seja acessando a memória ou injetando código, o que é facilmente evitável com o modelo descrito neste trabalho.

Este trabalho não tem a intenção de encorajar o uso de softwares de trapaça. Seu objetivo é demonstrar pontos cegos nos sistemas

de detecção de maneira que sejam encontrados meios para bloquear este tipo de software, o qual já está sendo utilizado, uma vez que o número de jogadores robôs tem aumentado novamente. Foi mostrada neste trabalho uma maneira de identificar este tipo específico de software robô. Todo o processo de desenvolvimento deste software pode contribuir com a comunidade acadêmica na prevenção ao uso de robôs em diversos jogos, incluindo Tibia. O trabalho mostra pontos de vulnerabilidades na detecção do Battleye que poderiam ser melhorados, bem como mostra comportamentos que deveriam ser melhor analisados pelos sistemas de checagem de comportamento não humano.

Como proposta para trabalho futuro, há a melhoria do sistema robô para ir além das três atividades principais atualmente executadas (caça, cura e saque), para também executar funções de comprar suprimentos, ir automaticamente até os locais de caça, verificar se os suprimentos estão em quantidade baixa, deixar o local de caça, guardar os itens obtidos, comprar suprimentos novamente e repetir o ciclo. Como mencionado nos resultados, apenas knights podem permanecer no campo de batalha por longos períodos sem a necessidade de repor suprimentos rapidamente. Para as demais vocações, o usuário deve observar a evolução do consumo de suprimentos de tempos em tempos e levar o personagem até a loja para comprar mais poções ou munição. Isto poderia ser conseguido com a análise gráfica das janelas de diálogo, canais de conversa, janela de compra e zona de depósito, as quais são áreas da janela do jogo ainda não exploradas no atual software robô.

Outro ponto de melhoria poderia ser a aplicação de técnicas de aprendizado de máquina para várias atividades, entre as quais podem ser mencionadas a melhoria no combate contra monstros, ações sendo mais similares às ações humanas no jogo, além de uma importante função: conversa automática com outros jogadores.

Na batalha contra os monstros, o atual sistema faz a luta ser desenvolvida em “modo de seguir”, onde o personagem sempre ficará próximo do monstro. Esta é uma abordagem apropriada para knights, mas para as demais vocações representa um gasto desnecessário de poções de cura. O mais apropriado para as demais vocações é sempre permanecer a uma distância segura dos monstros, e aprendizado de máquina poderia facilitar os movimentos apropriados com uma melhor análise da área de caça, a posição dos monstros, do jogador e de possíveis obstáculos. Além disto, aprendizado de máquina poderia melhorar o sistema de batalha mágica. Apenas uma magia de ataque é possível no software atual, e o usuário pode escolher a mais apropriada para o local de caça. Porém, podem existir diferentes monstros no local, cada um com fraquezas específicas ou algum grau de resistência a algumas magias, com alguns monstros podendo ser imunes a alguma delas. Desta forma, uma análise inteligente do dano recebido pelos monstros podem auxiliar na decisão do robô sobre qual magia utilizar para atacar, permitindo um leque maior de magias a serem utilizadas, ao invés de apenas uma como atualmente é feito.

Aprendizado de máquina poderia também fazer com que as ações do software robô sejam mais humanas, desde o movimento do mouse até os movimentos repetitivos quando atacando, saqueando, curando e andando pelo local. Estas ações são normalmente analisadas nos sistemas internos de detecção, especialmente quando muitos jogadores reportam o personagem, alertando a companhia do jogo sobre um certo jogador utilizando softwares de trapaça.

E para evitar que outros jogadores reportem o personagem utilizando o software robô, o sistema de conversa automática deve ser desenvolvido. Com técnicas apropriadas é possível simular uma conversa com o jogador que faz perguntas típicas antes de reportar um possível trapaceiro, por exemplo, “Você está aí?” e “Quanto tempo você ficará nesta caverna?”.

## REFERÊNCIAS

- [1] Battleye. 2020. *How Does It All Work?* Retrieved September 15, 2020 from <https://www.battleye.com/about/>
- [2] Bloodstone. 2020. *Bloodstone Online*. Retrieved September 15, 2020 from <https://www.bloodstoneonline.com>
- [3] Nick Cano. 2016. *Game Hacking: Developing Autonomous Bots for Online Games*. No Starch Press, San Francisco, USA.
- [4] Kuan-Ta Chen, Jhih-Wei Jiang, Polly Huang, Hao-Hua Chu, Chin-Laung Lei, and Wen-Chin Chen. 2008. identifying mmorg bots: a traffic analysis approach. *EURASIP Journal on Advances in Signal Processing* 797159 (oct 2008). <https://doi.org/10.1155/2009/797159>
- [5] CipSoft. 2020. *What is Tibia?* Retrieved September 15, 2020 from <https://www.tibia.com/abouttibia/?subtopic=whatistibia>
- [6] Tomas Curda and Petr Svenda. 2014. Analysis and detection of online game cheating software. Bachelor Thesis - Faculty of Informatics - Masaryk University, Brno, Czech Republic.
- [7] Sonia Fizek. 2018. Automated State of Play: Rethinking Anthropocentric Rules of the Game. *Digital Culture and Society - Rethinking AI* 4, 1 (2018), 201–214. <https://doi.org/10.25969/mediarep/13532>
- [8] Rahul Joshi. 2008. Cheating and virtual crimes in massively multiplayer online games. Technical Report, Royal Holloway - University of London.
- [9] Ah Reum Kang, Seong Hoon Jeong, Aziz Mohaisen and Huy Kang Kim. 2016. Multimodal game bot detection using user behavioral characteristics. *SpringerPlus* 5, 503 (apr 2016). <https://doi.org/10.1186/s40064-016-2122-8>
- [10] Samuli Lehtonen. 2020. Comparative Study of Anti-cheat Methods in Video Games. Master Thesis, University of Helsinki, Department of Computer Science, Helsinki.
- [11] David G. Lowe. 1999. Object recognition from local scale-invariant features. *Proceedings of the Seventh IEEE International Conference on Computer Vision* (1999), 1150–1157. <https://doi.org/10.1109/ICCV.1999.790410>
- [12] Kevin Kjelgren Mikkelsen. 2017. Information security as a countermeasure against cheating in video games. Master Thesis, Norwegian University of Science and Technology.
- [13] OldBot. 2020. *The OldBot*. Retrieved September 15, 2020 from <http://www.oldbot.com.br>
- [14] Stefano D. Paoli and Aphra Kerr. 2009. The Cheating Assemblage in MMORPGs: Toward a sociotechnical description of cheating. *Breaking New Ground: Innovation in Games, Play, Practice and Theory* (2009). [http://www.digra.org/dl/display\\_html?chid=09287.23190.pdf](http://www.digra.org/dl/display_html?chid=09287.23190.pdf) Proceedings of the 2009 Digital Games Research Association Conference.
- [15] Stefano D. Paoli and Aphra Kerr. 2010. The assemblage of cheating: How to study cheating as imbroglio in MMORPGs. *The Fibreculture Journal* 16 (jul 2010). <http://sixteen.fibreculturejournal.org/the-assemblage-of-cheating-how-to-study-cheating-as-imbroglio-in-mmorpgs/>
- [16] Stefano D. Paoli and Aphra Kerr. 2010. We will always be one step ahead of them: A case study on the economy of cheating in MMORPGs. *Journal of Virtual Worlds Research* 2, 4 (feb 2010).
- [17] Vanessa A. Pereira. 2013. O fascinante mundo do game Tibia: um estudo sobre sociabilidade jovem. *Teoria e Cultura* 8, 1 (jan/jun 2013), 93–114. <https://periodicos.ufjf.br/index.php/TheoriaeCultura/article/view/12181>
- [18] D. V. Silakov. 2012. Using Virtualization to Protect Application Address Space Inside Untrusted Environment. *Programming and Computer Software* 38, 1 (2012), 24–33. <https://doi.org/10.1134/S0361768812010069>
- [19] Ray Smith. 2007. An Overview of the Tesseract OCR Engine. *International Conference on Document Analysis and Recognition* (sep 2007), 629–633. <https://doi.org/10.1109/ICDAR.2007.4376991>
- [20] Li Wang, Liu Fan, and SungMin Bae. 2019. How to persuade an online gamer to give up cheating? Uniting elaboration likelihood model and signaling theory. *Computers in Human Behavior* 96 (jul 2019), 149–162. <https://doi.org/10.1016/j.chb.2019.02.024>
- [21] Vino Wilson and Jorge Basilio. 2020. Bot detection using Behavioral Analysis in MMORPG. Masters Thesis - National College of Ireland, Dublin, Ireland.
- [22] WindBot. 2020. *The WindBot*. Retrieved September 15, 2020 from <https://www.tibiawindbot.com/about-us.html>
- [23] Zezenia. 2020. *Play Zezenia*. Retrieved September 15, 2020 from <https://www.playzezenia.com/>