3
Modelagem e Arquitetura

Neste capítulo são descritos detalhes da modelagem e da arquitetura do framework aqui desenvolvido. São tratadas as questões de design do jogo, do sistema como um todo e da integração com o simulador já pronto.

3.1
Implementação e Plataforma

Utilizando-se da base do Microbacias foi continuado o projeto com a nova extensão. Foi utilizado a IDE Visual Studio 2010 para a codificação e a linguagem C# para facilitar o trabalho, já que todo o projeto já vinha sendo construído em cima dessas IDE e linguagem.

Foi usada uma máquina com Windows 7 Professional Edition 64 bits, processador Intel Xeon X5650 2.67GHz 2.66GHz, 48 GB de memoria RAM e placa gráfica GTX 580.

3.2
Game Design

Como dito anteriormente, o Microbacias possibilita a criação de perfis, que são as configurações de terreno demarcadas e escolhidas pelo usuário em uma execução do programa. Entram nestas configurações, a área demarcada, os tipos de manejo e os tipos de produtos cultivados, que são os principais fatores a serem considerados.

Cada perfil agora se torna um fazendeiro que possui uma fazenda equivalente ao terreno criado pelo usuário e as microrregiões serão os cultivos desse fazendeiro. O perfil inicialmente aberto pelo usuário é o fazendeiro que o usuário terá controle e os demais fazendeiros são controlados por inteligência artificial, podendo isso ser trocado em qualquer momento do *game*, permitindo tanto que o usuário faça que todos os fazendeiros sejam controlados por inteligência artificial como todos podem ser controlados pelo usuário.

O *game* funciona em turnos, ou seja, existem *checkpoints* que separam bem as ações do usuário e dos agentes, cada turno representa um trimestre

no mundo real. Depois de ultrapassados os checkpoints as ações salvas já não podem ser desfeitas. Cada turno equivale a um ano em um ambiente real, mas este número pode ser alterado.

As ações principais dos fazendeiros, agentes e jogador humano são trocar de cultivo e vender a produção. Apesar de haver somente duas ações, o fazendeiro deve fazer uma análise complexa antes de tomar qualquer decisão. Para isso, cada fazendeiro controlado por inteligência artificial deve ser dotado de comportamento diferente para que o *game* sempre produza resultados e desafios diferentes e um mercado coerente com a realidade.

Cada fazendeiro possui um estoque onde pode guardar a sua produção, isto permite que alguns segurem seus produtos para vender no momento mais oportuno.

O mercado fica responsável pelo cálculo e manipulação dos preços a fim de fornecer aos fazendeiros formas de vender seus produtos, consultar preços e ter indicativos do comportamento futuro da curva de preços. Não há demanda explícita, portanto o mercado deve se basear apenas na oferta dos produtos para sua variação de preços, além de se basear nas configurações pré-estabelecidas para cada um dos produtos adicionados. O mercado sempre compra todos os produtos que os fazendeiros queiram vender.

Como os perfis, cada microrregião se torna um cultivo. Os cultivos devem ser baseados nos produtos que já estão na base do Microbacias, e devem utilizar, para sua produção, os cálculos de degradação, erosão e produção já fornecidos pelo Microbacias. Usando a mesma relação entre perfil e microrregião, cada fazendeiro possui os cultivos correspondentes às microrregiões daquele perfil, e cada fazendeiro só é capaz de produzir e vender a produção de seus cultivos, podendo apenas ter acesso às ações tomadas por outros.

Não há uma condição de vitória explícita, isso é deixado para a análise do próprio jogador que pode avaliar a quantidade de dinheiro obtida ao longo do tempo, ou analisar qual jogador conseguiu manter sua propriedade com menos degradação, ou o que seria ideal, uma ponderação dos dois, que pode ser definida pelo jogador. Isto porque no jogo não há fim, este fim é definido pelos próprios jogadores.

3.3 Projeto Modular do Sistema

A figura 3.1 ilustra a arquitetura do Microbacias e a conexão com o novo módulo de *game* que corresponde ao framework aqui implementado.

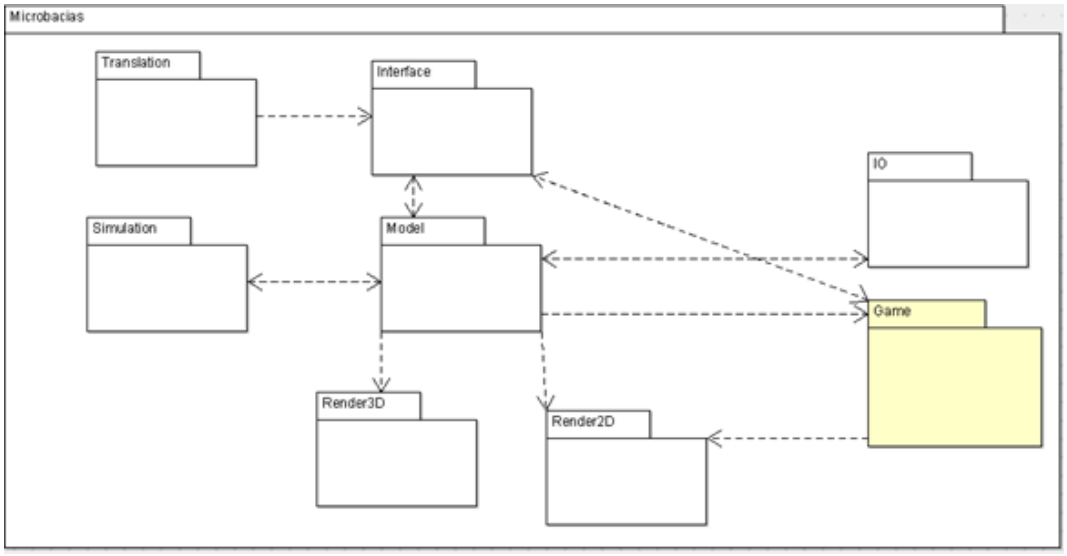

Figura 3.1: Diagrama de pacotes do Microbacias com pacote de *game* integrado

O pacote principal Model é que armazena os dados e configurações do programa, é nele que estão armazenados todos os dados da microbacia, os dados entrados pelo usuário e as configurações do programa.

O pacote IO fica responsável pela leitura dos dados da microbacia e das configurações do programa, e pela escrita de mudanças nas configurações. Além disso, é possível salvar as demarcações feitas pelo usuário e carregá-las por meio do perfil, tudo isto também passa pelo pacote IO.

A Interface fica responsável pela interação com o usuário, pegando dados do Model e exibindo e passando os dados do usuário para o Model. Translation provê serviços de tradução para a interface. Simulation é quem faz as simulações físicas de erosão e produção do terreno e comunica-se diretamente com o modelo para isto.

Render2D e Render3D trabalham como uma interface produzindo duas formas diferentes de visualização dos dados, uma em duas e outra em três dimensões. Enquanto a interface trabalha com entrada e saída de dados do usuário, os dois pacotes de "render" apenas exibem os dados de duas formas diferentes.

O pacote *Game*, que inclui o framework produzido, se comunica diretamente com Model solicitando dados para o *game*. E ainda é preciso comunicar-se com Render2D e Interface, avisando que o modo de *game* está ativado para que elas produzam saídas especiais para este modo e enviando dados para exibição. Nesta primeira versão, o pacote *game* não se comunica com o Render3D pois não há modo de *Game* com suporte a três dimensões.

A figura 3.2 ilustra, em outro nível, as interações entre os integrantes do

framework, entre si e com as entidades dos outros pacotes.

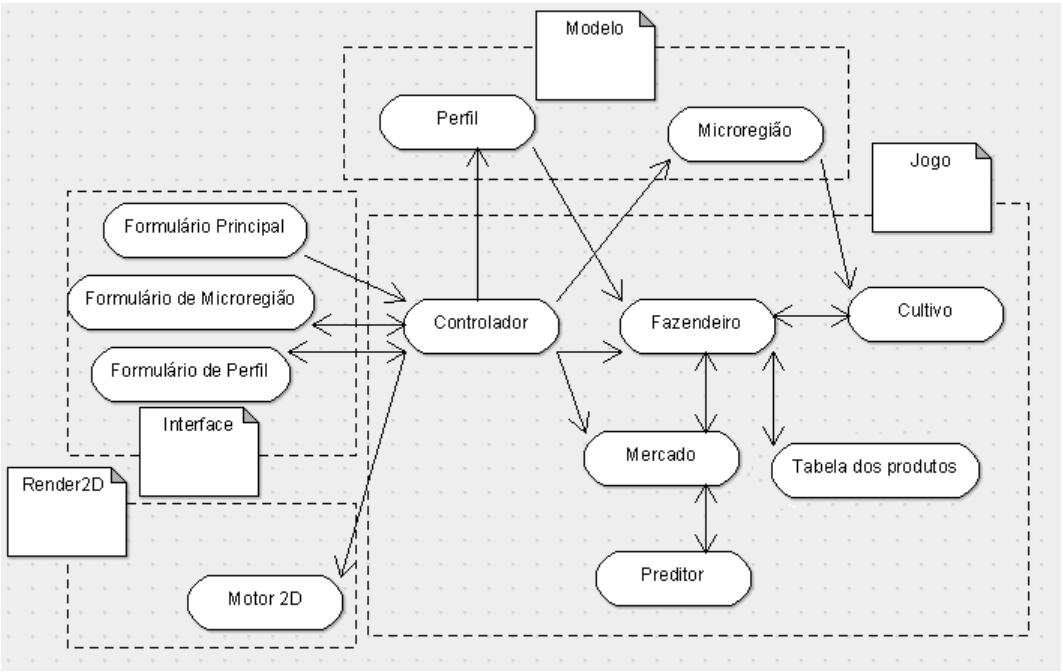

Figura 3.2: Diagrama com as entidades e suas comunicações entre si e com os outros pacotes

A entidade principal do pacote Jogo (*Game*) é o Controlador que comanda o fluxo de todo o jogo, nela estão contidos os fazendeiros, e por sua vez, nos fazendeiros temos os cultivos. O Controlador é avisado que o *game* começou por meio do formulário principal, e o jogo é jogado pelo jogador humano por meio do formulário de microrregião e do formulário de perfil, no qual é possível trocar de cultivo, vender produtos e alterar as configurações dos fazendeiros.

Foi adicionado, ao formulário principal do Microbacias, um botão para ativar o modo de *game*. Quando este modo está ativado, não é possível ativar nem opção 3D e nem a opção de simulação pura que vem com o simulador.

No início do jogo, o Controlador carrega os perfis e as microrregiões e cria os respectivos fazendeiros e cultivos, numa correspondência um a um. Durante o jogo o fazendeiro se comunica com os seus cultivos e com o mercado.

O Controlador ainda se comunica no inicio do *game* com Motor 2D para exibir corretamente os diferentes perfis que agora representam os fazendeiros.

Os fazendeiros se comunicam com o mercado vendendo os produtos e o mercado retorna a eles atualizações sobre preços e previsões, estas últimas obtidas por intermédio de um preditor.

Os fazendeiros ainda preenchem uma tabela de produtos que controla quais produtos são mais ou menos lucrativos e quais produtos são mais ou

menos degradantes.

3.4 Mercado

A princípio existiam duas maneiras distintas de se implementar o mercado do *game*. A primeira seria usar uma base de dados de preços provida previamente e que seria liberada aos jogadores à medida que o jogo fosse prosseguindo. Isso faria do mercado apenas uma interface do jogador com as tabelas previamente criadas.

A segunda solução seria criar um mercado dentro do próprio jogo que gerenciaria os preços dos produtos. Isso evitaria o problema de conseguir uma tabela de preços já pronta para cada produto cadastrado da microbacia, mas, por outro lado, teria que ser feita uma fórmula que funcionasse bem para cada produto e o mercado deveria ser um ambiente fechado, ou seja, os preços deveriam ser calculados apenas pelas vendas dos produtos.

A segunda opção foi escolhida, pois, em primeiro lugar, mostrou-se difícil achar uma tabela de preços ao longo de um longo período e padronizada para cada um dos produtos da microbacia. Em segundo lugar, uma tabela predefinida geraria sempre uma mesma curva de preços para todos os jogos criados no ambiente de uma microbacia, deixando o *game* sempre repetitivo e dando margem para que o jogador se especialize contra esta curva de preços em particular, acabando com o aprendizado proporcionado ao usuário.

Porém, a escolha desta estratégia criou o desafio de se elaborar uma fórmula que funcionasse bem para todos os produtos, e que proporcione a cada jogo uma experiência diferente, mesmo que tenha que se basear apenas nas vendas dos agentes.

A equação de cálculo do novo preço do mercado é a seguinte:

$$NovoPreco = (FatorAlfa - Vendas)/FatorBeta + ((0.5 - ruido) * FatorOmega) + UltimoPreco$$

Onde NovoPreço é o preço calculado para a rodada atual. UltimoPreço é o preço do produto na rodada anterior. Vendas é a quantidade vendida do produto na última rodada e Ruído é um número de ponto flutuante aleatório que varia de zero até um. Os fatores alfa, beta e ômega são números que variam de produto para produto.

O fator alfa funciona como uma demanda fixa para cada produto. Portanto variando-se essa demanda teremos uma variação na curva de preços,

que poderá ser crescente, decrescente ou manter-se estabilizada variando em torno de um eixo.

O valor de beta é o valor que transforma a unidade de vendas do produto em unidade de preços. Assim, quanto maior o beta, menor será a influência das vendas no valor total, e quanto menor o beta maior será a variação total.

Por último, o ômega pondera o quanto o ruído influenciará no resultado, assim quanto maior ômega, mais aleatória será a curva.

Durante algumas configurações que foram feitas para testes, observamos que os fatores alfa e beta dependem do número de fazendeiros presentes nos jogos. O que faz sentido pois se explicitamos o número de fazendeiros desses dois termos na equação e simplificamos, obtemos o número de vendas sobre o número de fazendeiros, ou seja, uma demanda fixa média por fazendeiro e não uma demanda fixa geral. Isto faz mais sentido, pois se mantivéssemos a mesma demanda para um jogo de 2 ou 20 fazendeiros, certamente algum destes dois jogos o mercado não se comportaria como esperado.

O mercado ainda foi projetado para prover algumas funções úteis para serem usadas nas estratégias dos fazendeiros. Eles têm acesso ao último preço de cada produto para que possam fazer suas vendas. Têm acesso também à média de preço de cada produto para saber como o último preço está em relação a todo histórico daquele produto. O mercado ainda provê acesso às vendas da rodada de cada produto, para que o fazendeiro saiba em que quantidade estão sendo vendidos cada produto.

Outra funcionalidade é um previsão que o mercado faz para cada produto baseado no histórico de preços. Esta é uma previsão bem simples e não muito certa, o que a torna mais interessante do ponto de vista do realismo do jogo. Esta previsão utiliza uma rede neural para prever o comportamento da curva de preços que pode ser vista como uma série temporal.

As redes neurais são sistemas que se assemelham ao cérebro humano, se constituem de múltiplas unidades simples que trabalham em paralelo sem um controle central. As conexões entre as unidades possuem pesos que são os elementos alterados pelo aprendizado, Russel (12).

A figura 3.3 mostra um exemplo de neurônio artificial, onde o neurônio recebe diversos sinais de entradas que são ponderados, somados e são capazes de ativar o sinal de saída ou não, que por sua vez transmitira outro sinal que poderá ser capturado por outros neurônios. A função de ativação representa uma das características importantes da rede, ela recebe uma soma ponderada dos sinais de entrada e define a saída. Podem ser usadas diferentes tipos de funções, como a linear ou a sigmoide por exemplo. Redes em que o sinal

de entrada de um neurônio é alimentado pelo sinal de saída de outro, são conhecidas por multicamadas, e quanto mais camadas, mais difícil de se configurar a rede, porém maior o seu poder.

Os algoritmos de rede neural passam por duas etapas, na primeira a rede é treinada com um conjunto de testes. A taxa de aprendizado define quão rápido a rede se adapta a uma nova entrada, taxas muito baixas demoram a aprender, enquanto taxas muito altas tendem a perder o conhecimento obtido facilmente. Após o treinamento a rede pode ser usada pelo usuário que pergunta à rede, e esta responde baseado no conhecimento obtido durante a etapa inicial. O conjunto de treinamento deve ser geral o suficiente para não haver discrepâncias com a realidade dos valores.

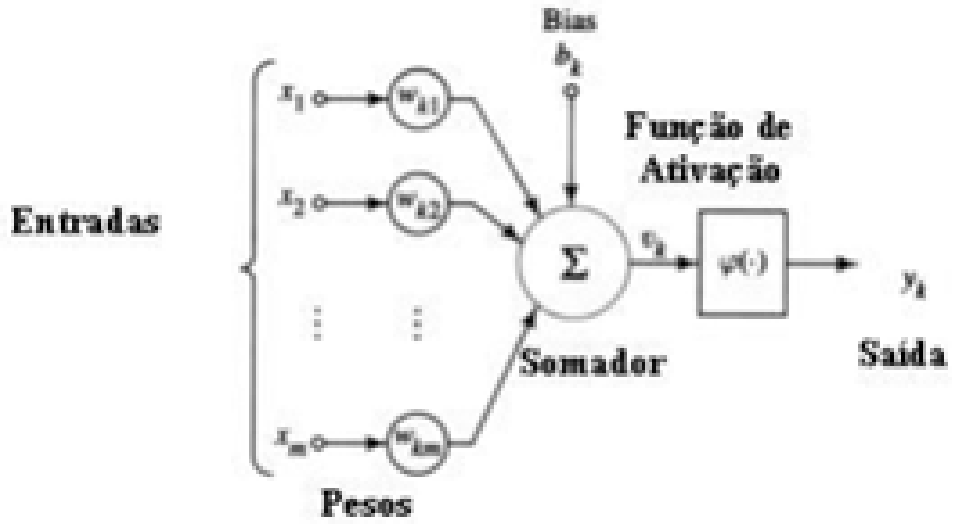

Figura 3.3: Exemplo de um neurônio artificial

Séries temporais são fenômenos cuja observação e quantificação gera uma sequência de dados ao longo do tempo, Souza (18). Elas podem ser contínuas ou discretas, contínuas quando os dados podem ser coletados a qualquer instante, e discreta quando os dados são coletados em intervalos geralmente equidistantes (8). O grande desafio para as séries temporais é realizar inferências em cima das propriedades da série a fim de se construir um modelo matemático simplificador da realidade que seja capaz de prever o comportamento futuro da série, Granger (8).

A previsão não constitui um fim, mas uma forma de apoiar a tomada de decisão em determinadas situações, Morettin (10). O horizonte de previsão como visto na figura 3.4, define até quantas previsões no futuro gostaríamos de fazer. Além do horizonte outro parâmetro usado em muitos métodos de previsão, é a janela de janela de previsão, que nada mais é do que um

subconjunto de valores sequenciais passados, usados para capturar padrões, Silva (16).

Figura 3.4: Exemplo de uma série temporal, exemplificando a janela de previsão

Após vários testes no sistema, foi configurada uma rede neural com janela de previsão 16, taxa de aprendizado igual a 0.8 e 200 iterações de treinamento a cada rodada. O aprendizado é feito online, ou seja, a medida que um novo preço entra na curva, a rede é retreinada com todos os valores já obtidos da curva de preço, para saber o próximo valor que virá na rodada seguinte.

3.5 Produção

A produção foi feita de modo que a cada turno do jogo o fazendeiro consiga produzir uma porcentagem do valor total de produção, dado pelos cálculos físicos já implementados do microbacias, seguindo a curva de produção, especificada na configuração do framework, para cada produto. Um exemplo de curva de produção é mostrado na figura 4.7. A interface mostrada nas figuras 3.5 e 3.6 ilustra esse processo.

A produção retornada pelo simulador depende dos tipos de produtos, tipo de manejo e da região escolhida. O simulador retorna dois valores, um que seria a perda de solo original e outro a perda de solo efetiva para a fazenda toda. Obtemos a porcentagem que a perda de solo efetiva representa sobre a original, de modo a saber se os cultivos são prejudiciais ou benéficos para a fazenda. É feito um cálculo para saber a contribuição de cada cultivo para a perda de solo.

O resultado da produção é estocado pelo fazendeiro, que pode vender ou não os produtos na rodada. Então, por exemplo, se o fazendeiro insistir em práticas não sustentáveis, no futuro pode não haver produção devido ao tratamento dado ao solo. Por outro lado, produzir de maneira sustentável pode

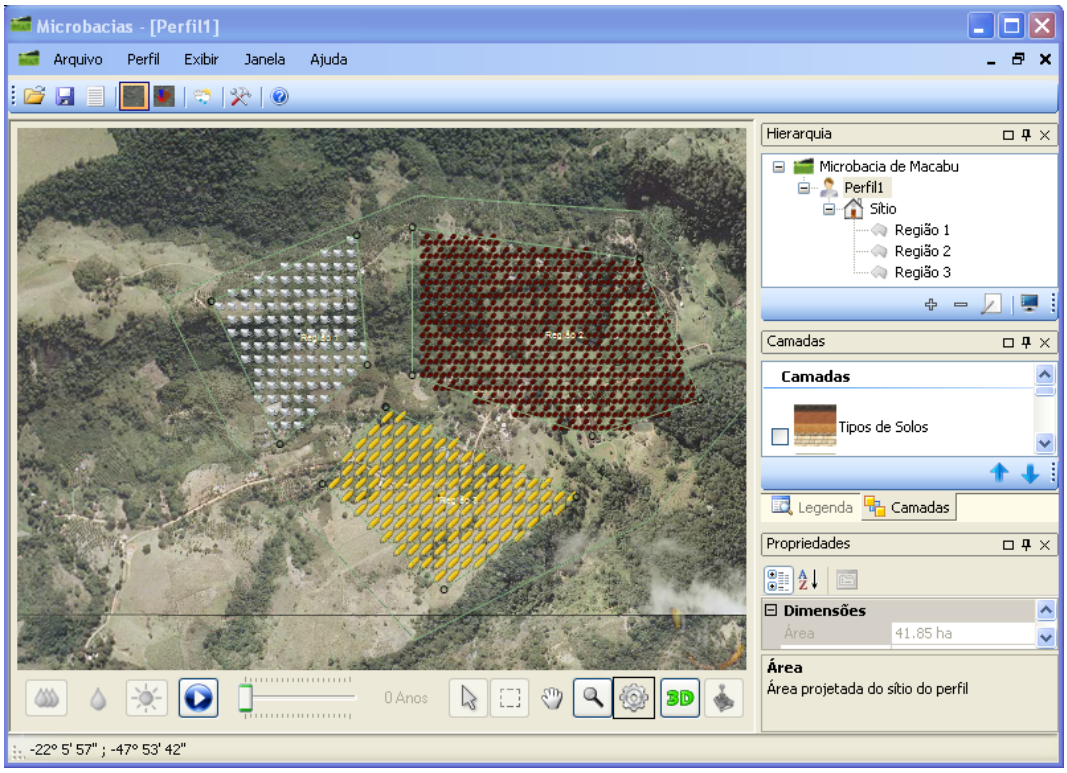

Figura 3.5: Fazendeiro com produção ideal

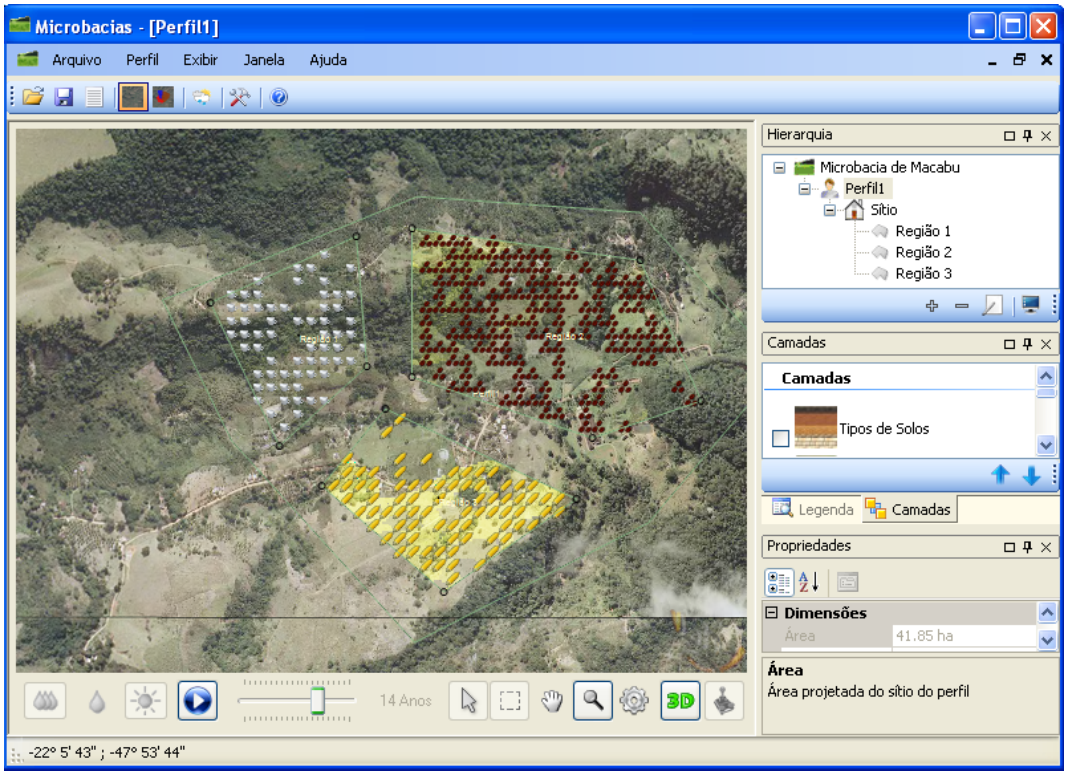

Figura 3.6: Fazendeiro com produção inferior ao total

gerar menor lucro a curto prazo, mas, garantir um longo período de produção. Esta é uma das lições que o jogador deverá aprender com o *game*.

Para compartilhar o conhecimento estre os agentes e fazer com que estes aprendam com o tempo sobre quão degradante ou quão lucrativo é um produto, foi utilizado um aprendizado por reforço bem simples e a experiência guardada na tabela de produtos. A figura 3.7 apresenta o fluxo dos dados.

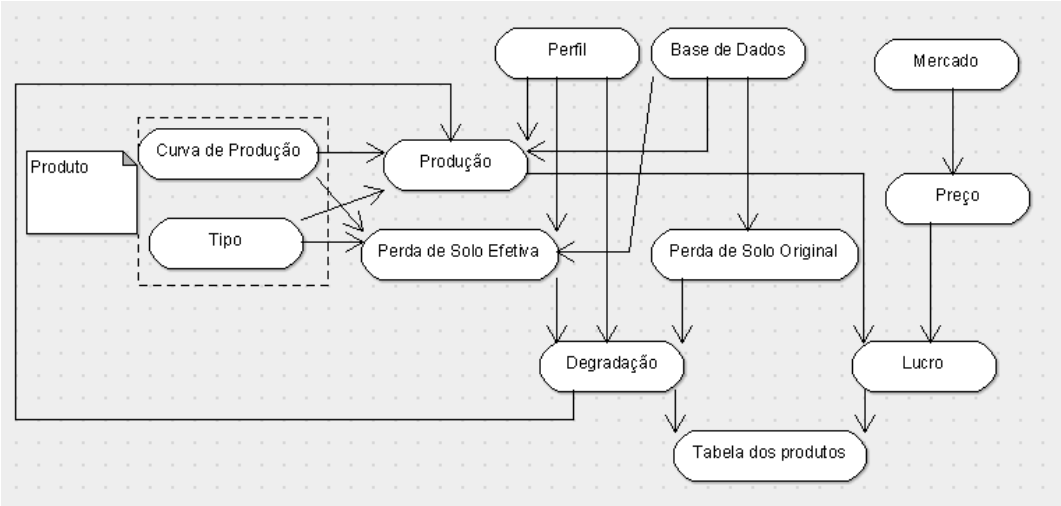

Figura 3.7: Esquema do fluxo de dados

Aprendizado por Reforço é um aprendizado que mapeia situações para ações de modo que maximize a recompensa, Sutton (20). No inicio o agente desconhece totalmente o ambiente, então ele escolhe uma ação ao acaso, para esta ação o ambiente lhe retorna uma resposta positiva ou negativa, e um novo estado. Baseado nessa resposta e na sua experiência que é dada pelas respostas anteriores, o agente escolhe outra ação. O objetivo do agente é maximizar a função de resposta do ambiente. A figura 3.8 mostra o funcionamento do Aprendizado por Reforço.

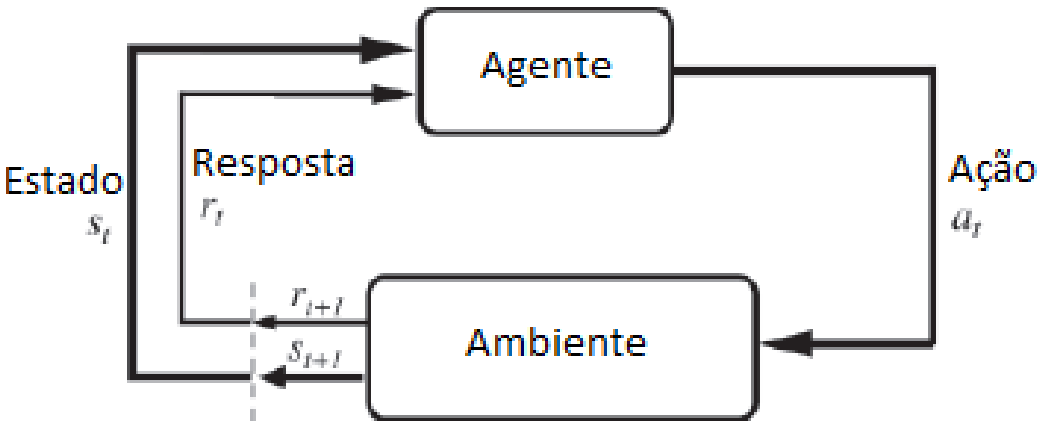

Figura 3.8: Ilustração do aprendizado por reforço

A cada rodada o fazendeiro pega o fator de degradação da sua fazenda e atribui para cada produto uma porcentagem desse valor. Essa porcentagem varia de acordo com o tamanho da fazenda. Dessa maneira cada produto tem um fator degradante para aquele fazendeiro, para aquela região e rodada, o mesmo é feito com as vendas, que nos retorna um valor para cada produto de lucro, para aquela rodada e para aquele fazendeiro. Então o fazendeiro usa estes valores que seriam a resposta do ambiente sobre degradação e lucratividade de cada um dos seus produtos, e manda gravar nas tabelas correspondentes. O programa então pondera o valor guardado na tabela com os novos valores fornecidos pelo fazendeiro. Este banco de conhecimento pode ser usado nos algoritmos de inteligência artificial dos agentes, para que eles tomem suas ações. Os estados são bem definidos como a configuração de uma dada fazenda.

Como todos os fazendeiros usam as tabelas, a medida que o jogo prossegue é possível observar que os produtos vão convergindo a uma ordem de lucratividade e degradação. Porém, caso o mercado mude ou algum valor de degradação do produto seja alterado, ele aos poucos vai mudando de posição na tabela. A parte de tomada de decisão é controlada pela configuração feita pelo framework, o que não garante a maximização da função de resposta. Outro ponto é que são duas variáveis a se considerar, degradação e lucratividade, e não necessariamente maximizar a função de resposta de uma, vai maximizar a outra. Caso a tomada de decisão consiga sempre melhorar as duas, ou pelo menos, não piorar os resultados, temos um algoritmo de aprendizado por reforço completo.

3.6 Modelagem dos agentes

Os agentes são os fazendeiros e estão intimamente ligados aos perfis como mostrado anteriormente.

São três as características definidas para diferenciar cada agente de inteligência artificial: Risco, Exploração e Inteligência.

Inteligência define qual estratégia o agente adotará. Para cada valor de inteligência deve existir uma estratégia correspondente. Inicialmente a inteligência varia de 0 a 4 com cinco estratégias distintas possíveis, sem qualquer relação de dificuldade ou capacidade entre elas. Estes valores podem mudar caso sejam implementadas mais estratégias.

Exploração está relacionada com a obtenção de novo conhecimento. Ou seja, se o agente apenas se mantém naquilo que é consolidado ou busca novas soluções ainda não testadas. Exploração está intimamente relacionada

ao cultivo de novos produtos na fazenda.

Risco é um fator que está relacionado às vendas no mercado, isto é, se um agente mantém um produto em seu estoque até obter um melhor preço ou se vende imediatamente mesmo que não tenha um preço tão bom.

Os valores de inteligência, risco e exploração são gerados aleatoriamente no início do jogo, mas podem ser alterados a qualquer momento pelo usuário. Há ainda uma *flag* dizendo quais agentes são controlados por inteligência artificial e quais são controlados pelo jogador. Os agentes controlados pelo jogador não utilizam em momento algum as características mostradas acima, mas elas estão presentes em todos os agentes, porque um agente pode ser marcado como não controlável no meio de um jogo.