UNIVERSIDADE FEDERAL FLUMINENSE
INSTITUTO DE COMPUTAÇÃO
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

FELIPI EUZÉBIO ARENA BOARIM
LEONARDO BEZERRA NEPOMUCENO DOS SANTOS

Jogo Educativo sobre Controle de Epidemias

EPIDEMICS

Niterói
2011

FELIPI EUZÉBIO ARENA BOARIM
LEONARDO BEZERRA NEPOMUCENO DOS SANTOS

Jogo Educativo sobre Controle de Epidemias
EPIDEMICS

Monografia apresentada ao Departamento de Ciência da Computação da Universidade Federal Fluminense como parte dos requisitos para obtenção do Grau de Bacharel em Ciência da Computação.

Orientador : Esteban Walter Gonzalez Clua

Niterói
2011

FELIPI EUZÉBIO ARENA BOARIM
LEONARDO BEZERRA NEPOMUCENO DOS SANTOS

Jogo Educativo sobre Controle de Epidemias
EPIDEMICS

Monografia apresentada ao Departamento de Ciência da Computação da Universidade Federal Fluminense como parte dos requisitos para obtenção do Grau de Bacharel em Ciência da Computação

Aprovados em Julho de 2011

BANCA EXAMINADORA

_____

Prof. ESTEBAN WALTER GONZALEZ CLUA
Orientador
UFF

_____

Prof. Luciana Tricai Cavalini
UFF

_____

Prof. José Raphael Bokehi
UFF

Niterói
2011

RESUMO

Este trabalho trata do desenvolvimento de um jogo de estratégia com foco na área de saúde, mais especificamente epidemiologia, chamado *Epidemics*.

O *Epidemics* permite ao jogador incorporar um avatar e tentar resolver casos desafiadores de epidemias em um ambiente hospitalar.

O jogador possui acesso a vários recursos, e precisa fazer uso sábio de suas ferramentas e habilidades para solucionar os casos e, consequentemente, evoluir no hospital. O avatar começa no nível de aprendiz, devendo enfrentar vários casos complexos até atingir o nível máximo de Doutor para finalmente ganhar o jogo.

À medida que o *epidemics* se desenvolve, o avatar também aumenta seus privilégios no hospital, podendo incrementar determinadas funcionalidades do hospital e até administrar controles epidemiológicos na cidade.

Palavras Chave: controle de epidemias, saúde, estratégia, jogo.

ABSTRACT

This work is about developing a strategy game focusing the healthcare area, specifically epidemiology, known as *Epidemics*.

Epidemics allows the player to play as an avatar and try to solve challenging medical cases of *epidemics* in a hospital environment.

The player has access to many resources and must use his tools and skills wisely to solve cases, increasing his level in the hospital. The avatar starts at apprentice level, and must face many complex cases until it reaches the maximum level of doctor and, consequently, win the game.

As the game develops itself, the avatar also increases his privileges in the hospital, and may upgrade several functionalities, and even administrate epidemiologic control methods in the city.

Keywords: epidemiology, healthcare, hospital, game

LISTA DE ACRÔNIMOS

Npc:	*Non-Player Character*
Hud:	*Head up Display*
Gui:	*Graphical User Interface*
Pc:	*Portable Computer*
SMC:	*Simple Manage C*
PDC:	*Professional Developers Conference*
UTI:	Unidade de Tratamento Intensivo
HDN:	História Natural da Doença

SUMÁRIO

LISTA DE FIGURAS

1 INTRODUÇÃO

1.1 Motivação

Em muitos aspectos, os jogos eletrônicos possibilitam um melhor ambiente de aprendizado. Os jogos permitem um ajuste de nível de dificuldade conforme as habilidades do jogador, provêem aos jogadores um *feedback* claro e imediato, e dá aos mesmos escolhas e controle sobre suas ações. Também despertam a fantasia e a curiosidade, além de oportunidades para colaborar, competir, ou socializar-se com os outros jogadores.

Projetos e estudos sobre o uso de jogos para aprendizado estão dando legitimidade ao assunto e apontam a idéia como tendo grande potencial para atingir a geração atual de "nativos digitais", ou seja, todos aqueles já acostumados com vídeo games, e-mail, *Chat*, telefones celulares e outras tecnologias interativas (Ref. 13).

A partir desta idéia, os departamentos de epidemiologia e ciência da computação da Universidade Federal Fluminense criaram uma parceria para a criação de um jogo de computador focado na área de epidemiologia.

O departamento de epidemologia criou primeiramente um documento, semelhante a um *game design*, no qual são especificados diversos aspectos essenciais em um jogo sobre epidemiologia, e esse documento foi a base para o desenvolvimento do *Epidemics*, sendo essencial para o processo de programação do jogo.

1.2 Objetivo

O objetivo deste projeto é levantar requisitos, projetar e desenvolver um jogo com as características e técnicas de desenvolvimento de um jogo para computador focado na área de saúde, mais especificamente, epidemiologia.

O propósito do *Epidemics* é criar uma atmosfera interativa e estimulante para o estudo do comportamento de algumas epidemias, simulando casos no qual o jogador

deve aprender e utilizar o conhecimento específico da área para controlar epidemias em um ambiente hospitalar.

1.3 Organização

Este trabalho foi estruturado da seguinte forma: No capítulo 2 são apresentados trabalhos relacionados e sobre a *Engine* e as ferramentas utilizadas no desenvolvimento. No capítulo 3 está descrito o *game design*. No capítulo 4 está descrito detalhadamente o desenvolvimento do jogo, tal como problemas encontrados e soluções propostas. No capítulo 5, são apresentados os resultados obtidos em diversos casos de uso. Por fim, no capítulo 6, é apresentada a conclusão obtida ao final deste trabalho.

2 TRABALHOS RELACIONADOS

2.1 Outros Jogos do Gênero

Existem alguns trabalhos realizados na área médica com intuito de educar e informar através de métodos interativos, porém a maioria deles não tem conteúdo muito aprofundado, e dificilmente iriam acrescentar um conhecimento utilizável em cenários da vida real. Em sua maioria são voltados para Web, onde são jogados através do navegador.

Destacam-se alguns jogos, como é o caso do jogo educativo *Re-Mission*, desenvolvido pela *Hope Lab*. *Re-Mission* foi criado com intuito de ajudar crianças com Câncer a entender de uma forma interativa como funcionam os inúmeros tratamentos para sua condição. O fator chave foi criar o jogo de acordo com o gosto atual das crianças, em um ambiente cheio de ação e movimentado, deixando-as com um bom astral enquanto jogam (Ref. 10).

Os resultados do jogo mostram sua qualidade e a quantidade de conteúdo que ele passa para os jogadores através de diálogos entre as diversas missões. Eles são absorvidos quase que imperceptivelmente. Pode ser usado como um excelente modelo de padrão de qualidade para qualquer futuro jogo a ser lançado nessa área, mostrando que é possível obter um resultado satisfatório de conhecimento e aprendizado através de métodos menos convencionais (Ref. 11 e 12).

2.2 Ferramentas para Desenvolvimento de Jogos

A seguir serão brevemente descritas as principais ferramentas que foram utilizadas para o desenvolvimento do *Epidemics*.

2.2.1 *Game Engine*

Para o desenvolvimento do *Epidemics*, foi utilizado o *game engine* Unity3D, desenvolvido pela companhia *Unity Technologies*.

Foi utilizada a versão 3.3.0, havendo duas distribuições principais: Unity3D e Unity3D Pro, sendo a primeira gratuita.

Esta ferramenta possibilita a democratização do desenvolvimento de jogos digitais e permite a criação rápida e intuitiva de aplicações de qualidade. Uma das principais características do Unity3D é a versatilidade, pois nele pode-se criar aplicativos para PC, Mac, WEB, Wii, Xbox 360, Playstation 3 e portáteis (Ref. 5).

Uma aplicação desenvolvida em Unity3D é dividida entre *game objects*, que são os objetos que compõem uma cena, e componentes, que são responsáveis por aplicar diversas propriedades nos *game objects*, tais como iluminação, física e movimento. Dentre os componentes, existem os scripts, que permitem ao desenvolvedor atribuir qualquer comportamento ao objeto.

Na versão 3.3.0 são suportadas as linguagens de programação *Javascript, C# e Boo* (um dialeto do *Python*). Outra qualidade neste *game engine* para ser destacada é a quantidade de formatos de arquivos suportados. O Unity3D pode importar arquivos como imagens e modelos em vários formatos e torná-los editáveis dentro da própria interface.

Além disso, a *engine* possui uma vasta coleção de *assets* nativos, como *shaders*, algoritmos de *lightmapping*, scripts básicos de locomoção de personagens, movimentação de câmera, agrupamento de *game objects* e também possui integração com a tecnologia *PhysX*, da NVIDIA®, que é um poderoso *engine* de física.

2.2.2 C#

Durante o desenvolvimento do framework .NET, as bibliotecas de classe eram originalmente escritas usando um compilador chamado SMC. Em janeiro de 1999, Anders Hejlsberg formou um time para desenvolver uma nova linguagem chamada *Cool*, que siginificava *C-Like Object Oriented Language* (Ref. 6).

No mesmo período o projeto .NET foi publicamente anunciado no PDC, em julho de 2000. A linguagem então seria renomeada para C#, e as bibliotecas .NET seriam importadas para o C# (Ref. 6).

O principal desenvolvedor da linguagem foi Anders Hejlsberg, que fora envolvido na criação do Turbo Pascal, Embarcadero Delphi e Visual J++.

A linguagem C# é definida como uma linguagem de programação orientada a objetos e simples. Com tipificação segura, ela deriva seus recursos de diversas linguagens como C, C++ e Java. Foi desenvolvida para oferecer a simplicidade do

Visual Basic e o poder da C++ como uma linguagem orientada a objetos, o C# facilita aos desenvolvedores criar, depurar e distribuir aplicações corporativas.

Pode ser considerada uma linguagem de aprendizado intuitivo para quem já tem conhecimentos nas linguagens da qual ela foi derivada, e sua simplicidade junto com sua versatilidade são fundamentais para o desenvolvimento de *games.*

2.2.3 Autodesk Maya

Maya é um programa de modelagem, animação e efeitos especiais desenvolvido pela Alias e é frequentemente utilizado na indústria de cinema e de televisão, tal como no desenvolvimento de jogos de computador e de consoles (Ref. 4).

Uma de suas melhores funcionalidades é a animação de modelos 3D, onde há as uma interface intuitiva, visando agilizar o processo de animação, que é, muitas da vezes, o processo mais desgastante na criação da arte visual.

Muitos desenvolvedores e artistas optam por utilizar o *Maya* em suas animações, independentemente de confeccionar seus modelos em outros aplicativos, exatamente por essa praticidade.

2.2.4 Photoshop

Em 1987, Thomas Knoll, um estudante de PhD na Universidade de Michigan começou a escrever um programa para exibir imagens em escalas de cinza em uma tela monocromática. Esse programa, chamado originalmente de *Display*, chamou a atenção de seu irmão John Knoll, um empregado da companhia *Industrial Light & Magic*, na Califórnia.

John sugeriu que seu irmão Thomas transformasse esse programa em um editor de imagem completo. Thomas, então, decidiu afastar-se de seus estudos por seis meses para se dedicar exclusivamente ao desenvolvimento da nova ferramenta, que viria a se chamar ImagePro.

Ao final deste mesmo ano, Thomas renomeou seu programa para *Photoshop* e conseguiu um acordo com uma empresa desenvolvedora de *scanners* com o objetivo de

distribuir cópias de seu software com um scanner portátil para fotografias, chamado *slide scanner*.

Nesse mesmo período, John viajou ao Vale do Silício e demonstrou sua criação aos engenheiros da gigante *Apple* e o diretor artístico da *Adobe*, Russel Brown. Ambas as demonstrações foram bem sucedidas, então a *Adobe* decidiu comprar a licença para distribuir o *Photoshop* em setembro de 1988.

Em fevereiro de 1990, foi lançado o *Adobe* Photoshop v1.0 e, até o presente, passou por dezesseis novas versões até chegar na atual *CS5.1*, lançada em maio de 2011.

O *Photoshop* possui poderosas funcionalidades e é líder de mercado em aplicações de edição de imagem em duas dimensões. Com uma imensa difusão na indústria de jogos, tornou-se uma ferramenta fundamental para o desenvolvimento das texturas que compõem os materiais dos modelos 3D. Neste projeto foi utilizada esta ferramenta para a criação e manipulação das inúmeras texturas e figuras necessárias.

3 *GAME DESIGN*

Tal como um software inicia seu processo de desenvolvimento através de documentos de engenharia de software, um jogo inicia-se mediante a elaboração do documento de *game design*, que irá detalhar todas as características que serão implementadas (Red. 9).

3.1 Descrição Detalhada do *game design* do *Epidemics*

O Jogo se inicia no nível de Aprendiz. O jogador tem um avatar que se encontra na recepção de um hospital. Há cadeiras para os pacientes, e conforme sua chegada eles são organizados em uma fila. Entretanto, a ordem de atendimento não é necessariamente por ordem de chegada.

No inicio do jogo os pacientes são gerados em estados de saúde aleatórios, e conforme o decorrer do jogo e com o nível de contaminação da cidade, pode haver um padrão nas infecções dos pacientes indicando que a cidade está contaminada por alguma epidemia.

A lógica do jogo gera um paciente e seleciona as características que podem ser informadas para o jogador: sexo, idade, endereço de residência, sendo então atribuída uma doença a ele. Caso o paciente tenha sido contaminado devido a outro paciente infectado que voltou para sua rua, ele já será gerado com a mesma doença, mostrando que existe uma possível epidemia se alastrando em seu endereço.

As informações sobre sexo e idade são transmitidas automaticamente para o jogador assim que o paciente chega. A informação geral sobre o estado clínico será o diagnóstico dado quando o paciente é enviado para uma sala de exames, e nela permanece até o exame terminar. Exibe-se então uma prancheta contendo os sintomas do paciente para a visualização do jogador. Assim, fica a cargo deste descobrir qual a doença que afeta o paciente e quais medidas tomar para curá-la ou amenizá-la.

Os pacientes são tratados de acordo com seus sintomas, onde cada tratamento abrange uma lista de sintomas relacionados, tal como tratar dores musculares pode curar fadiga. Porém, nem todos os sintomas podem ser curados por tratamentos convencionais. Alguns sintomas provenientes de doenças chamadas persistentes só serão curados quando a doença em si é tratada (como tuberculose).

Se o tratamento for efetivo, o jogador irá gradativamente perceber melhoras no paciente, assim como na lista de diagnóstico o sintoma irá desaparecer.

Para tratamento de sintomas o paciente deve ser enviado para a sala de enfermaria, onde o jogador poderá selecionar dentre uma lista de tratamentos aquele que julgar o mais adequado. Cada tratamento tem um tempo determinado de efetividade, onde a idade do paciente é relevante, ou seja, para pacientes mais novos o tempo de efetividade é menor, assim ele será curado dos sintomas mais rapidamente.

A doença no *epidemics* pode ser classificada em dois tipos: não contagiosa e contagiosa. A primeira não apresenta risco de contaminação na cidade, ou seja, um paciente doente enviado para casa não disseminará sua condição, apenas penalizará o jogador pelo erro de diagnóstico. A segunda, por outro lado, além de penalizar o jogador, também infecta os habitantes da cidade.

O horário de funcionamento do hospital é de oito às dezenove horas. Neste período, o jogador pode intervir em qualquer momento após a chegada do paciente, porém algumas horas antes do fechamento do hospital, todos os pacientes que ainda estiverem na sala de espera irão embora, e caso alguns deles esteja infectado, poderá contaminar os habitantes da cidade. Quando o jogador libera pacientes doentes ou não trata algum paciente, sofre penalidades em sua reputação. Caso a reputação do jogador caia em demasia ele poderá ser demitido.

Existe também uma sala para pacientes muito debilitados, sendo que nessa sala os danos causados pela doença incidente são minimizados. Essa sala é importante para casos onde o paciente apresenta sintomas persistentes, onde após a doença específica ser tratada, haverá um período de cura, sendo que os sintomas irão persistir de modo que o jogador não terá influência nessa situação. Ou seja, para esperar a melhora do paciente sem riscos, o jogador deverá enviá-lo para a sala de tratamento intensivo.

Se o jogador for capaz de manter uma regularidade de casos tratados com sucesso, ele acumulará pontos de experiência e sua reputação irá aumentar, o que lhe garantirá diversas vantagens no jogo.

Para o término do jogo há duas possibilidades: a cidade ficar com um índice muito alto de contaminação ou o doutor ser demitido do hospital em função de elevada incidência de erros nos tratamentos.

3.2 Personagens

No *Epidemics*, existem quatro tipos de personagens : o jogador principal, representado por um avatar *in-game*, o paciente, as enfermeiras e o doutor.

A interação entre os personagens é constante durante o jogo, pois é necessário para o mesmo consultar dados dos outros personagens e obter informações relevantes para solucionar os casos.

3.2.1 O Personagem Principal

O personagem principal é representado pelo avatar de um estudante de medicina (Fig.1). Este avatar pode andar livremente pelo ambiente hospitalar, exceto por algumas salas de tratamento. O jogador terá opção de escolher o nome do personagem no início do jogo e este nome será apresentado sempre que for consultada a tela com os atributos do personagem.

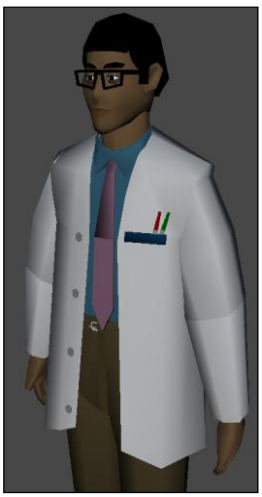


Figura 3 : Personagem Principal.

3.2.1.1 Atributos do Personagem Principal

O personagem principal possui quatro atributos e um indicador de nível (*level)*. O jogador inicia o jogo no nível de aprendiz, e adquire experiência a cada tratamento efetuado com sucesso. À medida que sua experiência aumenta seu nível também é incrementado e, consequentemente, mais pontos serão disponíveis para serem gastos nos atributos.

O personagem principal inicia o jogo com cinco pontos em cada atributo e cinco pontos para distribuir entre os mesmos (Fig.2). Os atributos no *Epidemics* exercem forte influência nas decisões do jogador, podendo facilitar ou dificultar uma determinada

jogada. Não há número máximo de pontos em cada atributo, mas o nível máximo do personagem é dez (doutor). Após distribuídos os pontos, deve-se clicar no botão salvar e em seguida voltar, caso o jogador se arrependa da distribuição feita, podendo assim fazê-la novamente. Entretanto, depois de salvar a fase, não há mais como redistribuir.

Segue uma breve descrição de cada atributo:

Conhecimento: Afeta a quantidade de tempo necessária para diagnosticar um paciente enviado a sala de examinação, diminuindo o tempo necessário para gerar diagnósticos. A idéia é que quanto mais conhecimento o personagem tem mais rápido ele consegue identificar os sintomas do seu paciente.

Gerenciamento: São disponibilizadas algumas "ferramentas" no hospital, que podem ser melhoradas conforme a capacidade de gerenciamento do jogador progride, sendo que essas melhoras apenas serão disponibilizadas para compra se o personagem tiver uma quantidade mínima de gerenciamento requerida.

Resistência: A resistência do personagem é a capacidade de agüentar trabalhar por mais tempo. Ou seja, quando o jogador alcança a capacidade máxima de trabalho que ele pode agüentar em um dia o jogo seguirá para o próximo dia. Durante a noite todos os pacientes infectados submetidos a algum tratamento continuarão melhorando ou piorando de situação clínica. Quanto maior a resistência, maior o escopo de tempo que o jogador possui para tomar decisões.

Sorte: influência minimamente em fatores randômicos do jogo. Todos os fatores recebem um coeficiente pequeno de acréscimo de acordo com a quantidade de sorte que o jogador possui.

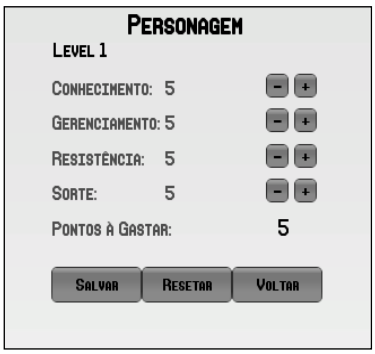


Figura 4 : Tela de Atributos do Personagem

3.3 O Dr. Gregório

O Dr. Gregório será um personagem auxiliar, representado por um avatar de médico, que é utilizado como um fornecedor de ajuda ao jogador. Para pessoas iniciantes, ele será muito útil, pois dará dicas e informações relevantes para a progressão do *Epidemics*.

3.4 Enfermeiras

No *Epidemics* há dois avatares de enfermeiras paradas à frente das salas de tratamento. Cada enfermeira terá ações específicas às salas correspondentes. A enfermeira da sala de enfermaria será responsável por administrar qualquer tratamento existente no jogo. O personagem precisa interagir diretamente com ela e selecionar o paciente desejado para aplicar a medicação correta. Caso o tratamento sugerido não faça efeito no paciente, o jogador perderá tempo e o doente não terá melhorado de condição.

A outra enfermeira, responsável pela UTI, será apenas um avatar de consulta, pois esta sala terá o propósito de evitar que a saúde dos pacientes com doenças persistentes se deteriore.

3.5 Pacientes

Os pacientes são personagens essenciais no jogo. Através da interação com eles o jogador poderá tomar as decisões de tratamento e visualizar informações necessárias para tomá-las. A tela inicial de diálogo com o paciente permitirá ao jogador visualizar a tela de ações do paciente. É através desta que o jogador poderá ser enviado para as outras salas.

3.5.1 Movimentação dos Pacientes

O movimento dos pacientes é determinado por um algoritmo de *pathfinding* descrito na seção 4.11 deste trabalho, de modo que, através deste, os seus movimentos

estejam limitados a caminhos fixos que partem da sala de recepção e vão até alguma das salas de tratamento, passando pelo *hall* principal.

3.5.2 Atributos dos Pacientes

Os atributos dos pacientes são os indicadores de saúde e de sintomas, sendo este segundo não visível ao jogador a priori. Os sintomas podem ser visualizados depois que o paciente recebe um diagnóstico na sala de examinação.

A deterioração da saúde do paciente é diretamente proporcional aos sintomas que o afetam e a sua idade. Pacientes mais idosos pioram sua condição mais rápido que pacientes jovens. Se um tratamento é efetuado com sucesso em um paciente, este ficará livre dos sintomas que o tratamento abrange. Porém, se a doença for do tipo que apresenta sintomas persistentes, ele não irá ser curado do sintoma imediatamente. Cabe ao jogador descobrir pelo conjunto de sintomas que esta afetando o paciente qual doença o está infectando.

3.6 Jogabilidade

O jogador se encontra em um hospital onde pode se locomover livremente. Pacientes chegam e se alojam em uma sala de espera.

3.6.1 Objetivo do jogo

Cabe ao personagem se comunicar com os pacientes e descobrir através do diagnóstico de seus sintomas quais a doenças que eles possuem e seus riscos para a sociedade, para que assim consiga tratar de forma eficiente a sua crescente fila de pacientes.

Cabe também ao jogador gerar os recursos do hospital e fazer melhoria nas suas instalações. Essas melhorias trazem mais benefícios aos pacientes que as utilizam, sejam na velocidade do tratamento, ou no retardamento da evolução das doenças.

3.6.2 Interações

O *epidemics* possui interação direta entre alguns objetos e o personagem principal. Para realizá-la, é necessário aproximar-se de algum componente passível de interação (computador, enfermeiras, pacientes, etc.) e pressionar o comando correspondente. Uma GUI (*Graphical User Interface)* de menu aparecerá no centro da tela com as opções disponíveis daquela interação.

Durante a interação, o jogador não consegue realizar outras ações, como a movimentação, até que essa termine. Todos os outros componentes do jogo continuam em funcionamento.

3.6.3 Condição de Vitória

Para vencer, o jogador deverá combater os casos com uma taxa mínima de sucesso, isso significa manter sua reputação positiva, visto que ela varia de acordo com as decisões tomadas pelo jogador, e impedir que a população total da cidade esteja contaminada durante um período determinado. O jogador deve manter uma reputação acima de um mínimo estipulado, caso contrário perderá o jogo.

3.7 Público Alvo

O *Epidemics* se destina a dois públicos diferentes, mas que interagem entre si :

- Pessoas que estão aprendendo epidemiologia, ou seja, alunos em formação, ou pessoas já formadas que estão se tornando especialistas que podem usar como ferramenta de aprendizado.
- Pessoas leigas que têm interesse pelo assunto. O jogo pode ajudar algumas pessoas a se livrarem de temores infundados em relação às doenças epidêmicas.

3.8 Desafios

O desafio apresentando ao jogador é o tratamento de diversas doenças. Serão geradas doenças com sintomas específicos, de acordo com a realidade (doenças

existentes), porém com alguns sintomas agravantes que constam em alguns quadros registrados da doença. Esses sintomas serão apresentados ao paciente de forma aleatória, ou seja, o paciente pode ou não apresentar alguns, todos ou até mesmo nenhum destes sintomas agravantes.

Esses sintomas causam um desafio não óbvio ao jogador, pois muitas das doenças, apesar de terem uma cadeia de sintomas constantes, podem apresentar outros de acordo com cada paciente, fazendo com que o jogador não faça suposições.

Há também doenças complexas que possuem sintomas persistentes, onde tratamentos convencionais não surtem efeito. Para elas será necessário um tratamento específico que também leva um tempo para ser realizado. Mesmo após este tratamento ser finalizado, não há garantia que o paciente esteja livre da doença, podendo os sintomas ainda persistirem por algum tempo até que o paciente esteja finalmente curado. O doutor deverá usar as ferramentas do hospital para estabilizar a condição do paciente até que isso ocorra.

O desafio ao jogador é gerar as ferramentas do hospital de modo a conseguir tratar todos os seus pacientes em um tempo estipulado antes que o mesmo perca toda a sua saúde e ainda assim não enviá-los doentes para casa ou com a saúde muito debilitada (abaixo de um nível estipulado).

3.9 Interface

Na tela principal do jogo serão disponibilizados ao jogador duas *huds*, uma com a quantidade de experiência para o jogador avançar ao próximo nível e outra para a quantidade de reputação conquistada pelo jogador. Também sempre estará ativo um relógio que informa ao jogador a hora virtual do jogo, assim como dia, mês e ano. Esse tempo é fundamental para o jogador conseguir administrar as ferramentas do hospital de maneira eficiente e para que as doenças possam progredir e afetar a saúde dos pacientes.

Em algumas situações, haverá barras de progresso, que correspondem ao tempo que uma determinada ação necessita para ser finalizada.

3.10 Ambiente

O jogo está situado num hospital, que por sua vez se localiza numa cidade.

3.10.1 Hospital

O hospital (Fig.3) é o local de trabalho do jogador. É um cenário sempre bastante movimentado, com alta rotatividade de pacientes. Há cinco ambientes internos importantes que devem ser ressaltados.

Figura 3: Cenário completo do hospital

3.10.1.1 Sala de Exames

A sala de exames é responsável por gerar diagnósticos, sendo o tempo de examinação relativo ao conhecimento do doutor. É conveniente que o jogador trate os pacientes apenas após saber de seus sintomas, mas ele não é limitado a isso. Ele pode tentar tratar o paciente com tratamentos aleatórios, porém poderá ocupar desnecessariamente vagas em outras instalações, e o tratamento pode não ocorrer rápido o suficiente.

3.10.1.2 Sala de UTI (Unidade de tratamento intensivo)

A internação na UTI não melhora a condição do paciente por conta própria se ele não recebeu um tratamento específico anteriormente. Existem doenças no jogo que não tem cura, ou que possuem sintomas persistentes mesmo depois de tratadas. Para estes casos a sala de UTI é fundamental, pois manterá o paciente o mais estável possível

durante esse período de ineficácia no tratamento. Essa sala é importante para não deixar que a condição do paciente deteriore.

3.10.1.3 Sala de Enfermaria

A sala de enfermaria é a sala onde os pacientes recebem seus tratamentos. Cada tratamento tem um tempo específico de eficácia, sendo que durante esse período o paciente continua sofrendo com todos os sintomas e somente quando o tratamento é finalizado ele estará livre dos sintomas, caso estes sejam tratáveis.

Há uma caixa de ajuda explicando quais sintomas cada tratamento engloba, sendo eles agrupados de forma coerente com a parte do corpo o sintoma atinge. Por exemplo, tratamento ocular engloba qualquer sintoma que afete a visão do paciente.

Há também tratamentos específicos para sintomas derivados de doenças mais complexas. Tuberculose tem sintomas em que o tratamento comum respiratório não surtirá efeito mesmo depois de finalizado. Cabe ao jogador notar isso. Existem alguns tratamentos para doenças específicas e cada tratamento leva um tempo diferente.

3.10.1.4 Computador

O jogador tem a sua disposição um computador, sendo através dele possível ver as atuais condições de suas ferramentas no hospital e utilizar seus recursos para realizar melhorias nessas ferramentas até uma capacidade máxima.

Também é através dele que o jogador recebe informações das doenças e de seus sintomas principais e secundários. Como um computador real, ele deve ser consultado constantemente para que o jogador possa descobrir com mais facilidade através dos sintomas qual doença está afetando o paciente. Esta ferramenta será uma das mais educativas do jogo, pois através dela o jogador conseguirá coletar uma extensa quantidade de informações reais.

3.10.1.5 Mapa

O mapa encontra-se no *Hall* do hospital perto de um agente do governo. No mapa é possível visualizar as informações de contaminação da cidade, ou seja, quantos

moradores então contaminados por alguma doença em cada rua. A doença específica atingindo a rua não é informada, cabe ao jogador utilizar os conhecimentos adquiridos durante o jogo para aplicar as medidas corretas. As ações possíveis por rua são campanhas realizadas, sendo estas campanhas colocadas como mecanismo de prevenção, ou seja, não curam a população infectada, mas apenas previnem o surgimento de novos casos.

A população que permaneceu infectada será direcionada ao hospital como novos pacientes.

3.10.2 Cidade

A cidade é organizada por um conjunto de ruas, sendo que dessas ruas surgem os pacientes. A informação de onde mora o paciente consta no seu diagnóstico, e caso o paciente receba alta ainda estando doente ele poderá criar um foco de epidemia em sua rua. Esse foco poderá crescer se não combatido pelo jogador, onde pode até mesmo se alastrar para outras ruas ou até mesmo por toda a cidade.

4 Desenvolvimento

4.1 Introdução

Neste capítulo serão discutidos os aspectos relacionados ao desenvolvimento do *Epidemics*. Serão descritos os principais *Game Objects*, como funcionam, sua importância no contexto do jogo, sua conexão com outros objetos, as dificuldades de implementação e soluções propostas para resolver esses problemas. Também serão descritos os scripts e componentes desses objetos, bem como a forma de como eles se "comunicam" para produzir um resultado satisfatório.

Os *game objects* citados neste capítulo são: fila de bancos, fila de UTI, fila de enfermeiras, fila da sala de examinação, Jogador, Hospital, Iluminações, Paciente, Gerador de Doenças, Cidade, Interação de Infermeira, Interação de UTI e Computador.

4.2 Fila de bancos

Esse *Game Object* é responsável pelo controle de assentos para os pacientes e é através dele que o jogo verifica se há a possibilidade de entrar um novo paciente no hospital. A posição da fila é (0,0,0) no espaço da cena e ele não é renderizado, ou seja, não aparece para o jogador ou para o desenvolvedor.

A fila possui um valor de assentos definido em 8, sendo esse valor constante durante todo o jogo e representando a quantidade máxima de pacientes que o hospital pode comportar em um instante.

Para representar cada posição individual da recepção, foram criados 8 objetos não renderizáveis e amarrados à fila de bancos. Cada objeto, chamado de *bench* (banco), possui um script chamado *Bench Numeration* (numeração de banco).

Cada *bench* corresponde a uma posição na sala de recepção, propositalmente coincidindo com as posições dos assentos. Cada paciente que entra no hospital recebe uma posição (se disponível) que corresponde a uma cadeira, como mostra a Figura 4 .

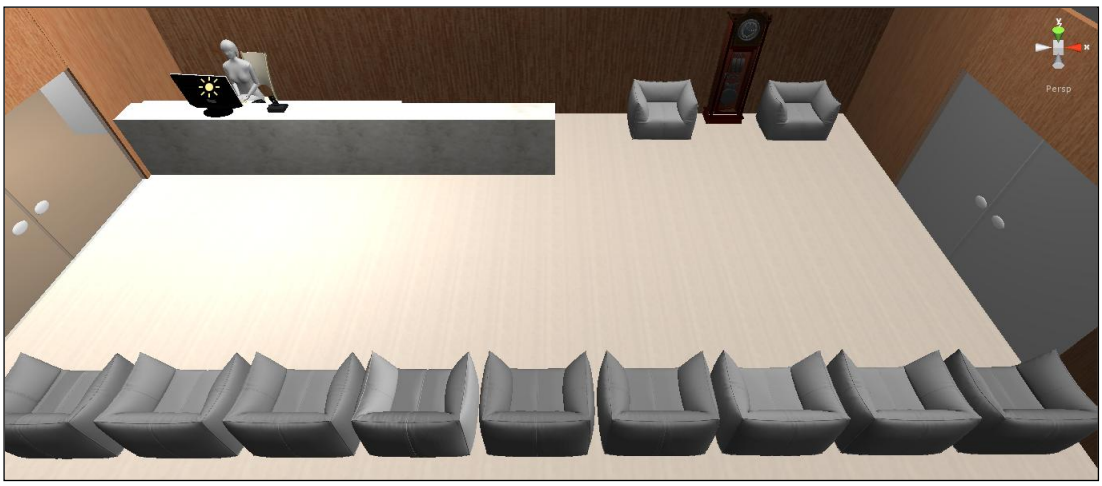


Figura 4 : Posição dos bancos na sala de recepção

No *Bench Numeration*, cada banco possui uma variável inteira responsável pela posição dele no vetor de bancos e um método que retorna o valor desta variável.

O principal e único componente da fila de bancos é um script chamado *Bench Occupation* (ocupação de bancos), que implementa o controle das posições através de um vetor de variáveis lógicas e coordena os *benchs.*

O vetor de variáveis lógicas é coordenado através do método *getFreeBench*, que varre esse vetor verificando qual variável possui o valor *True* e retorna o objeto *bench* correspondente. Caso o vetor esteja cheio, retorna a porta de saída. Isso significa que o componente responsável por instanciar os pacientes no cenário precisa consultar a fila de bancos antes de criá-los.

Além do método de procurar lugares disponíveis, a fila de bancos também possui outros dois métodos: *benchAvaliable e freeBench*.

O primeiro varre o vetor de bancos sequencialmente e retorna *True* se existe alguma posição disponível. Esse método de consulta é usado em combinação com o *getFreeBench*, sendo que primeiramente verifica-se se existe um lugar disponível para depois obter o banco correspondente para enviar o paciente.

O segundo método é responsável por atribuir um valor lógico a um banco, caso um paciente entre ou saia da sala de recepção. O valor dessa posição no vetor então deve mudar para verdadeiro ou falso.

4.2 Fila de UTI

A fila de UTI é semelhante à fila de bancos nos aspectos de renderização e posição na cena, sendo no entanto mais complexo que o anterior. Esse *game object* é responsável pelo controle da fila de pacientes da sala de UTI.

A sala de UTI é um dos objetos que podem ser evoluídos durante o jogo. Isso significa que no início ela possui uma quantidade de posições definida em um, e pode ser incrementada até três. Analogamente à fila de bancos, a fila de UTI possui objetos associados que representam as posições disponíveis.

Cada objeto é chamado de *Tube*, que também não são renderizados e são amarrados à fila de UTI, pois são responsáveis pelas vagas disponíveis nos aparelhos.

Cada *Tube* possui um script semelhante ao *BenchNumeration*, chamado *TubeNumeration*, que armazena uma variável inteira que representa a posição desse tubo.

A fila de UTI possui apenas um componente: um script chamado *TubeOccupation*. Esse script possui um vetor lógico semelhante ao *BenchOccupation*, e outro vetor de *GameObjects* que armazena os objetos *Patients*.

Além dos vetores, esse script também declara uma varável do tipo *GameObject*, que instanciará o *Computer*, pois é através deste que o jogador pode melhorar a sala da UTI. Essa instanciação é necessária para acessar um método do computador que retorna um inteiro representando o nível da UTI.

O script de *TubeOccupation* possui um método *Start*, nativo do Unity3D. Esse método é sempre executado apenas uma vez no início do jogo. A instanciação do computador é feita nesse ponto.

Esse script também possui um método *GetFreeTube*, que retorna um objeto *Tube* correspondente e outros responsáveios por atribuir um paciente a uma posição e retirar o mesmo desta.

4.3 Fila da sala de Examinação

Esse componente tem o mesmo funcionamento da Fila de UTI na parte referente a preenchimento e desocupação da sala. O diferencial deste componente é a forma de como o atributo do doutor afeta o tempo de examinação do paciente e como é gerado o diagnóstico do mesmo.

Os atributos de cada doença são definidos no paciente no momento em que ele é criado, conforme será detalhado na seção 4.9 deste trabalho, e uma variável lógica de ativação do diagnóstico começa com valor falso. No momento em que um paciente adentra na sala de examinação, é iniciada imediatamente uma contagem regressiva para a ativação da variável de diagnóstico. Caso o paciente sofra outra intervenção durante sua examinação o diagnóstico fica indisponível.

Enquanto o paciente está na sala de examinação, é informado ao jogador na respectiva *GUI* que o processo de examinação foi iniciado. Quando o processo é finalizado, o diagnóstico fica disponível para consulta sempre que o jogador quiser acessar esta informação.

O tempo de examinação é idêntico para todos os pacientes. O fator que altera esse tempo é o nível da sala, que pode ser incrementado no script *computer*, em troca de recursos do jogador e a quantidade de pontos no atributo "conhecimento" do mesmo.

A equação 1 exemplifica cálculo de tempo de examinação:

$$\text{Tempo Total} = \frac{\text{(Tempo Inicial de Examinação)}}{\text{(Conhecimento/10)+Nível da Sala)}}$$

Equação 1: Cálculo do tempo total de examinação

Ou seja, só há uma mudança razoável no tempo a partir do conhecimento maior que dez, a sala de examinação começa no nível zero, logo ela não influencia na equação até que o jogador já tenha gastado algum recurso para tal. Esta abordatem permite um desenvolvimento gradual, impedindo o jogador de atingir um nível máximo e parar de investir nesse atributo, dessa forma o conhecimento necessitará de investimento durante boa parte do jogo.

4.4 Jogador

4.4.1 Interação do Jogador

Para entendimento deste componente, é necessária uma breve explicação sobre o conceito de *raycast.*

Raycast é uma técnica utilizada utilizada para resolver vários problemas em computação gráfica. Consiste em traçar uma linha virtual partindo de um observador para detectar a primeira colisão com um objeto. Neste trabalho, esta técnica foi utilizada para determinar a natureza dos objetos intercedidos pelo raio e tratá-los.

O *PlayerInteraction* é o script que controla a ativação de interações do jogador com outros objetos. O *player* tem sempre um *Raycast* ativo na sua frente, com um determinado alcance. Os testes realizados são os seguintes, caso o botão de ação seja pressionado pelo jogador:

- Existe algum objeto dentro do alcance do *raycast* do *player*?
- Esse objeto está na camada física de "objetos interativos"?
- Esse objeto possui um script de interação?
- Ative os scripts de interação e desative qualquer outro script de interação que estiver ativo (programação defensiva).
- Desative a movimentação do jogador enquanto a interação estiver ocorrendo.

4.5 Computador

O Computador do hospital é um componente estático e é através deste que o jogador pode consultar informações de doenças, evoluir o equipamento do hospital e comprar suprimentos.

Ele possui a função de exibir uma interface através de uma GUI que foi desenvolvida inteiramente em um único script chamado *Computer*.

Esse script também possui a função de administrar os níveis de todas as salas que possuem a possibilidade de serem evoluídas e o contador dos suprimentos do hospital, que serão utilizados principalmente para as campanhas de controle de epidemias.

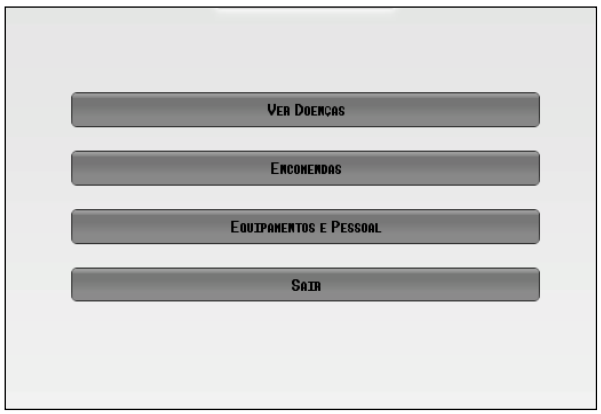


Figura 5 : Interface Principal do Computador

Como mostra a figura 5, a primeira tela do computador apresenta quatro botões. O botão “Ver Doenças” exibe uma nova tela com uma GUI simulando um computador real, que será a principal fonte de consulta para o jogador. Esta GUI se ramifica em três GUIs menores, exemplificadas na figura 6.

Figura 6 : Tela de Consulta de Doenças

A seção “Foto da Doença” possui uma imagem específica para cada doença. A seção de sintomas exibe uma lista dos sintomas primários e secundários, como detalhado na seção 4.7 deste trabalho. Por último, a seção “Dados da Doença” exibe o nome da doença, a taxa de transmissibilidade, a possibilidade de ser curável ou não, e o agente patogênico (Ref. 7).

O segundo botão, “Encomendas”, possibilita ao jogador adquirir suprimentos necessários para as campanhas de prevenção na cidade.Por último, o botão “Equipamentos e Pessoal” (Fig.7) é responsável pela melhoria nas salas, como descrito anteriormente.

Figura 7 : Tela de Equipamentos e Pessoal

Para incrementar o nível de determinada sala, o jogador precisa utilizar os recursos do hospital, que serão mais detalhados na seção 4.7. Cada nível possui um custo associado e esse valor aumenta exponencialmente à medida que o hospital vai sendo incrementado.

4.7 *Iluminação*

No *Unity3D* existem três tipos de componentes de iluminação nativos: *directional light*, *pointlight* e *spotlight*. A primeira é chamada de luz direcional, ou infinita, pois nela os raios de luz incidentes são paralelos, por isso ela é utilizada para simular os raios solares. No *Epidemics* esse tipo de iluminação não foi utilizado, pois o objetivo é simular um ambiente hospitalar fechado, portanto há apenas o tratamento da iluminação interna.

Pointlight, ou luz pontual, é a fonte de luz que emana de um ponto singular no espaço para todas as direções. Ela pode ser utilizada para representar fontes pequenas de

luz como lâmpada ou disparos de arma de fogo. No *epidemics*, a luz pontual está presente em todas as salas, pois o objetivo é simular um objeto iluminador sem precisar exibi-lo na cena.

A terceira e última, *spotlight*, é semelhante à *directional light*, porém finita, pois possui uma direção definida e afeta uma área em formato de cone na sua frente. Ela é utilizada para simular tochas, lanternas ou qualquer outro objeto com fonte de luz direta e gradual.

Neste trabalho, foi utilizada a *spotlight* apenas nas janelas horizontais do escritório do doutor. O objetivo é simular uma fonte de luz menos incisiva que o sol, porém com força suficiente para iluminar parte do ambiente.

No cenário do *Epidemics*, o objeto Iluminação foi criado por questões de organização para agrupar todas as luzes existentes no hospital.

4.8 Gerador de Doenças

4.8.1 Introdução

Atualmente, o *Epidemics* possui seis doenças implementadas: botulismo, dengue, influenza, mononucleose infecciosa, rubéola e tuberculose. Cada uma dessas doenças possui um *game object* associado e ela e este é persistente durante todo o jogo. O esquema com as doenças e sintomas abordados neste trabalho está descrito na figura 8.

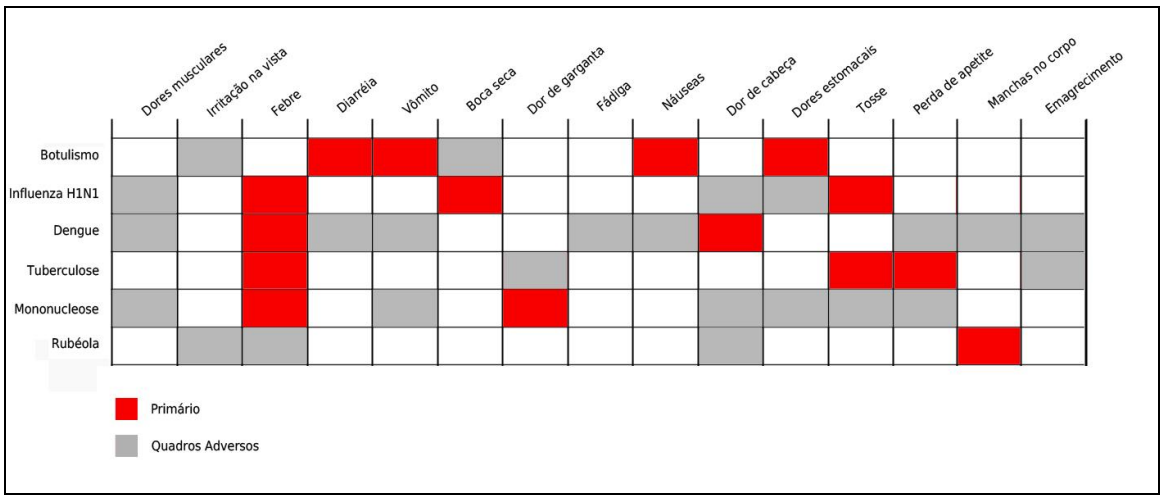


Figura 8 : Esquema de Doenças e Sintomas do *Epidemics*

A função do gerador de doenças é atribuir as propriedades corretas a cada objeto *Disease* da lista de doenças.

Os sintomas foram retirados dos livros citados nas referências bibliográficas (Ref. 1) e (Ref. 2).

4.8.2 Problema

O desenvolvimento da lógica da geração de doenças foi o primeiro desafio na criação do *Epidemics*. O paradigma original seria instanciar objetos de epidemias e doenças dinamicamente durante o *gameplay*. Desse modo, se houvesse algum paciente contaminado com dengue haveria um *game object* Dengue na cena, que faria parte da epidemia da mesma. Essa estratégia tornou-se ineficiente em diversos pontos.

Primeiramente, poderia haver casos nos quais doenças seriam geradas sem necessariamente compor uma epidemia. Porém, a maior dificuldade seria nas variáveis de cada doença. Em cada *game object* do tipo *Disease*, existem variáveis como quantidade de ruas contaminadas, que não poderiam ser perdidas caso não houvesse pacientes com essa doença.

4.8.3 Solução

Para solucionar esse problema, foi adaptada uma solução na qual se perdia a visão de epidemias e doenças, e se trabalharia apenas com doenças e sintomas. Para esse fim, foi desenvolvido um script com uma classe estática, chamado *Diseases Generator*.

O script de geração de doenças possui um vetor de objetos *Disease* e instancia todas as doenças do jogo. Para cada doença instanciada, ele cria e preenche com valores pré-definidos e invariáveis dois vetores de sintomas: o primeiro possui os sintomas principais, ou seja, aqueles que se apresentam na maioria dos casos, e o segundo possui os sintomas com menor probabilidade de aparecer no paciente.

No início do jogo, o script atribui os sintomas e o nome a cada *game object* do tipo *Disease* que está amarrado ao Gerador de doenças. Portanto, neste momento serão criadas todas as doenças com suas propriedades correspondentes.

4.8.4 Doença

As doenças são os objetos que compõem o gerador de doenças. Como descrito anteriormente, são persistentes e não são renderizadas no cenário.

Cada doença possui um script *disease* que possui um nome, uma lista de ruas infectadas, o tipo de transmissão, um indicador informando se é tratável e, assim como o gerador, possui vetores de sintomas primários e secundários.

Esse script realiza apenas a contaminação na cidade, pois a geração de doenças no paciente é feita pelo próprio *Patient*. A lista de ruas infectadas armazena todas as ruas que receberam pacientes ainda doentes.

O método mais importante desse script é o *disseminateInfection*, que se comunica diretamente com a cidade para disseminar esta doença em todos os endereços da lista. Outro fator importante é que, antes de gerar um novo paciente, o jogo consulta as doenças que possuem endereços nesta lista, e dá prioridade a eles.

Essa lógica caracteriza indiretamente uma infecção, pois pacientes doentes enviados para casa disseminarão sua doença na sua vizinhança a cada dia, e o número de doentes nesta rua aumentará exponencialmente. Sabendo disso, o jogo assegura-se que os próximos doentes a chegar ao hospital possuam a mesma doença e endereço dos infectados da cidade.

Outro aspecto importante sobre as doenças é que ela pode ser classificada de duas formas: persistente ou não-persistente. As doenças persistentes são imunes aos tratamentos convencionais e devem ser tratadas especificamente, ou seja, o jogador deve saber exatamente qual a doença que está infectando o paciente e tomar as ações necessárias para tratá-la. Estes casos são mais complexos e menos prováveis de ocorrer.

Porém, mesmo quando realizado o tratamento a doença persistente não é eliminada, o jogador deve apenas conter os sintomas e esperar a degradação dos mesmos. Somente então o paciente será considerado curado e começará a melhorar sua saúde.

Já as doenças normais serão tratadas pelos sintomas convencionalmente, utilizando a UTI ou a enfermaria.

4.9 Hospital

O Hospital corresponde ao cenário do jogo propriamente dito. Nele, são agrupadas todas as salas para maior organização no desenvolvimento, como mostra a figura 9. Ele possui uma sala de recepção, um *hall* principal, duas salas de tratamento, uma sala de examinação e o escritório do doutor.

Figura 9 : Estrutura hierárquica de objetos do Hospital

No fluxo do jogo, todos os pacientes entram pela porta dupla de vidro à direita da recepção e se direcionam a um dos assentos pelo critério explicado na seção 4.15 deste trabalho.

Na recepção, o jogador deve iniciar as interações com os pacientes e direcioná-los às outras salas. O *hall*, apesar de pouco movimentado, possui uma importante função no fluxo de movimentação dos pacientes. Antes de chegarem a suas respectivas salas de tratamento, eles passam obrigatoriamente pelo salão, portanto ele diminui o "gargalo" de movimentação na saída da recepção e na entrada das outras salas.

O *hall* também contem as enfermeiras e um mapa, elementos que serão responsáveis pela interação entre o jogador e a cidade.

4.10 *Cidade*

Este componente é responsável pela interação entre o jogador e a cidade. O objetivo original deste trabalho era desenvolver uma cidade inteira à parte do cenário padrão do hospital. Nesta cidade, o jogador poderia dirigir uma ambulância e atender os doentes em suas residências. Apesar de ser uma idéia interessante e aumentar drasticamente a complexidade do *Epidemics*, por motivos de cronograma, ela não pôde ser implementada.

Por outro lado, não seria viável descartar o componente cidade de um jogo sobre epidemias. Então, foi proposta uma solução que abordava uma implementação de uma cidade simplificada, na qual seria possível apenas visualizar a população contaminada e aplicar campanhas de prevenção nas ruas.

Seguindo esta abordagem, foi introduzido ao cenário do hospital um mapa anexado a uma parede (Fig.10), sendo através deste que o jogador poderá interagir com a cidade.

Para tal, foram desenvolvidos três scripts que representarão a cidade virtual: *Street*, *Town* e *TownGUI.*

Figura 10 : Mapa da Cidade

O primeiro é apenas uma classe que possui indicadores de nome, total de habitantes, habitantes infectados e uma lista de campanhas efetivas.

O script *town* é mais complexo, possuindo indicadores do total de habitantes e total de habitantes doentes, além de um vetor de ruas e uma lista de doenças incidentes.

Além das variáveis, esse script possui métodos essenciais para o estabelecimento da lógica proposta. Quando um paciente doente é enviado para casa, sua rua é adicionada na lista de ruas que sua doença possui. Todos os dias às oito horas da manhã,

as doenças infectam mais uma parcela da população através da chamada do método *disseminateInfection* contido neste script.

Este método é responsável por acrescentar a respectiva doença na lista de doenças (este procedimento é importante para a geração dos pacientes infectados na cidade), varrer os endereços contidos na doença e disseminar a infecção. Porém, antes de disseminar, é feita uma verificação em cada rua acerca de campanhas que possam impedir o aumento da população doente.

O terceiro e último componente deste *game object* é o script *TownGUI*, que é responsável por toda a informação exibida na tela e pela interação com o personagem principal. A tela principal deste componente está exemplificada na figura 11.

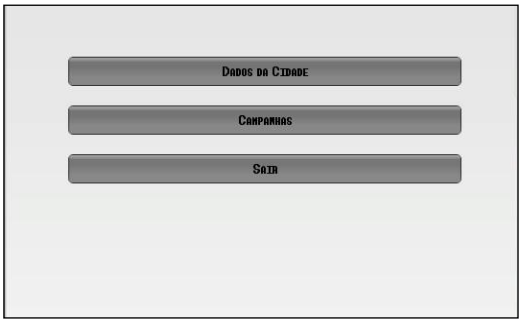


Figura 11 : Tela Principal do Script *TownGUI*

O *TownGUI* também possui uma tela responsável pela aplicação das campanhas na cidade.

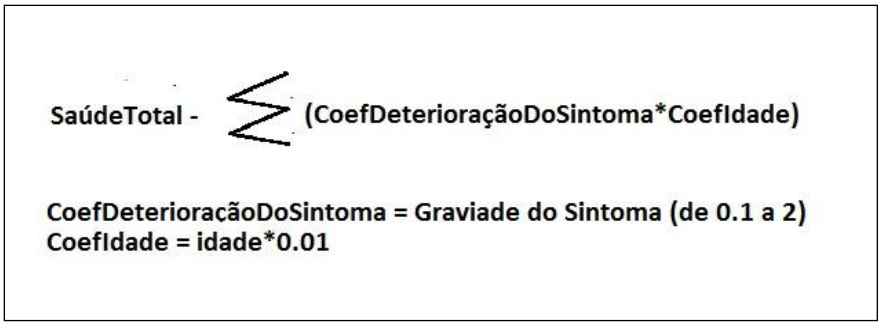
4.11 Paciente

4.11.1 Saúde do Paciente

Este componente controla os atributos e sintomas da doença designada ao paciente no momento de sua criação. Os sintomas são controlados por variáveis lógicas, que informam se o paciente possui ou não o sintoma. Esses sintomas apresentam um coeficiente de dano que faz parte de uma equação para calcular qual o dano total que o paciente irá receber por cada parcela de tempo. Essa equação também é influenciada pela idade do paciente para causar um efeito realista.

Esses coeficientes são atribuídos com pesos maiores para sintomas mais preocupantes, como febre, e com pesos menores para sintomas que afetam minimamente o organismo, como espirros.

A equação é dada pela soma dos produtos destes coeficientes, modificado de um percentual de degradação calculado através da idade, como mostra a equação 2.



Equação 2 : Cálculo da saúde do paciente

Também é importante citar que o paciente aumenta sua saúde espontaneamente em função do tempo, simulando que, na ausência de sintomas, o organismo retorna gradativamente ao seu estado normal, acarretando uma melhora na saúde do paciente. Assim, existe um valor de aumento de saúde espontâneo para cada paciente dependendo da sua idade. Para que a saúde do paciente tenha uma variação positiva esse valor deve ser maior que o resultado da equação de degradação referente aos sintomas do paciente. Isso simula que mesmo com sintomas que afetam minimamente uma pessoa ela ainda assim consegue recuperar sua saúde.

O paciente ficará doente durante um determinado período de tempo se seus sintomas não forem curados, isso acontece devido a cada doença possuir um HDN (história natural da doença) específico, que é o tempo que a doença persiste se nenhum tratamento é efetuado no paciente. Caso todos os sintomas forem tratados, o paciente estará curado da doença e não apresentará risco de contaminação na cidade.

Conforme os tratamentos são finalizados os sintomas não persistentes relativos serão atualizados e curados. Um exemplo do componente *PatientHealthStatus*, contendo os sintomas, doença e outras variáveis de controle é exibido na figura 12.

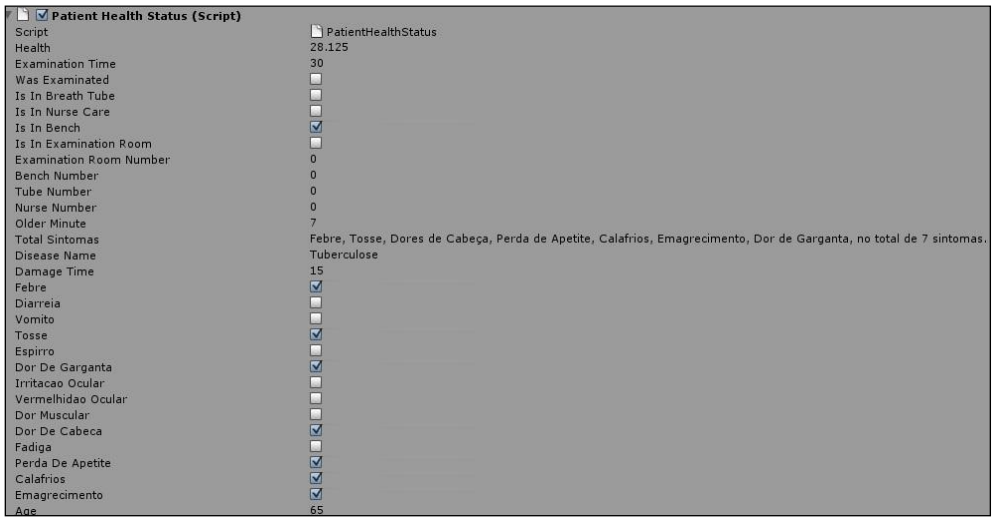


Figura 12 : Inspetor do *PatientHealthStatus*

4.11.2 Tratamento do Paciente

Controla os tratamentos que estão sendo realizados no paciente. Esse *script* calcula quanto tempo levará para um determinado tratamento surtir efeito no paciente. Esse calculo utiliza a idade do paciente com o nível da instalação sendo utilizada. É possível realizar tratamentos para sintomas que o paciente não apresenta para dificultar as situações e dar a chance do jogador poder tomar decisões erradas.

Esse *script* também controla qual a posição ele está ocupando na determinada sala que ele está. Quando mudado de sala ou recebendo alta ele libera essa posição que ele estava ocupando na sala, sedendo a vaga para outros pacientes.

4.11.3 Movimentação do Paciente

Exerce o controle das rotinas de movimentação do paciente pelo hospital. Esse script funciona como uma Inteligência Artificial que implementa uma solução baeada em *pathfinding*. Ele controla os estados do paciente.

Ele também controla a frequência de procura de novos *waypoints*, que são os caminhos que o paciente irá seguir até o seu destino dentro do hospital. Esses waypoints, por motivos de otimização, não são conectados diretamente da origem até o destino, mas atualizados de acordo com essa frequência, tornando os caminhos mais precisos de acordo com a posição do personagem num determinado momento.

O paciente tem dois estados: “caminhando” e “em espera”. Este script envia mensagens para o *pathfinding* a todo instante para saber qual o estado atual do paciente. Se ele está caminhando é controlado a frequência de procura de novos caminhos até o destino, e quando o destino é alcançado o estado do paciente é alterado e ele fica em espera, até receber uma novo comando de movimentação.

Também possui atributos básicos, como velocidade de movimento e rotação.

4.11.4 Interação de Paciente

Controla todas as interfaces de interações do jogador com o paciente (a forma como a interação é ativada foi explicada na seção 4.4.1) e as chamadas de método de movimentação na Inteligência Artificial.

Ao interagir com o paciente abre-se uma primeira tela (Fig. 13) com três opções:

- Conversa: É retornada uma resposta do paciente, como se o paciente estivesse se diagnosticando, sendo este diagnóstico pouco confiável. Para cada sintoma do paciente há 25% a mais do fator de sorte do paciente afirmar corretamente o sintoma. Esta lista pode conter sintomas omitidos pelo paciente, aumentando assim a situação real.

Figura 13 : Tela inicial de interação com o Paciente

- Ação: Controla as ações disponíveis para serem tomadas pelo jogador referentes ao paciente. A interface das ações é exibida na figura 15. Essas ações são:

  a. Prontuário: permite ao doutor visualizar o prontuário do paciente (Fig. 14). Esse diagnóstico requere que o paciente já tenho sido examinado na sala de examinação e tem 100% de precisão. Após feito o exame o prontuário do paciente fica disponível a qualquer momento que o jogador quiser acessar, através do menu de ações. Essa ferramenta também é atualizada conforme o paciente é curado dos seus sintomas, sendo altamente importante para o tratamento eficiente do paciente. Nela são listadas as informações pessoais do paciente, como idade, nome, sexo e endereço. As informações pessoais do

paciente só podem ser vistas dessa forma. Os sintomas são listados de forma organizada seguido do número total de sintomas que o paciente apresenta. Esses sintomas não são discernidos por principal ou secundário, causando certa dificuldade para o jogador descobrir qual a doença que realmente afeta o paciente, contando que a única forma totalmente precisa de se saber qual a doença do paciente é pela lista obrigatória de sintomas, sendo estes misturados com sintomas não obrigatórios, que podem ocorrer em alguns casos.

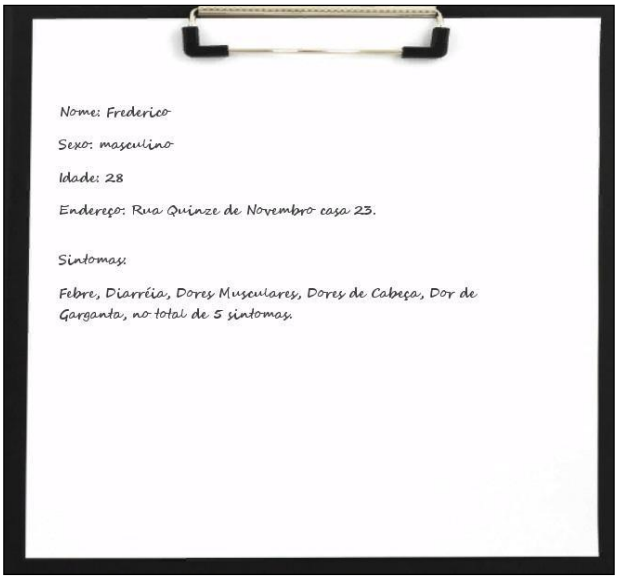


Figura 14 : Prontuário

b. Enviar para Enfermaria, UTI ou sala de examinação: Chama um método do *script* de movimentação para enviar o paciente para o local correspondente, e atribui seu estado como atualmente em uma sala ocupando uma posição da mesma. Os *scripts* de envio às salas funcionam com o mesmo mecanismo.

c. Dar alta: Chama o *script* de movimentação para enviar o paciente para a porta de saída do hospital.

Figura 15 : Tela de ações do Paciente

4.12 *ExitDoor*

A porta de saída funciona como o calculador de pontuação. Quando um paciente que recebeu alta passa por este objeto auxiliar, dois testes são efetuados:

1- Quanto é a saúde do paciente?

Este teste afeta diretamente a quantidade de bonificações que o jogador irá receber em sua reputação e experiência. Quanto maior a saúde mais bonificações.

2- O paciente ainda está doente?

Não há nenhuma penalidade caso um paciente receba alta em uma boa condição de saúde. Porém se ainda estiver doente e a doença for transmissível ocorrem testes nos *scripts* de cidade referentes a probabilidade de contaminação da doença. Esse paciente pode se tornar um foco de epidemia na rua onde mora.

4.13 Fluxo do Tempo

Componente que gerencia a passagem do tempo virtual do jogo. O cronômetro do jogo é divido em horas e minutos, e o calendário em Dia, Mês e Ano. Este componente possui um script chamado *RealTime,* que é responsável pela manutenção do tempo virtual do jogo.

Este script inicia-se invocando uma função nativa do *Unity* chamada *InvokeRepeating*. Essa função executa um método várias vezes em um intervalo de

tempo estipulado na chamada. Neste caso, é chamado o método *TimeFlow* com um intervalo determinado pela variável *realTime*. Este método controla o tempo virtual fazendo a incrementação dos minutos do jogo e as outras variáveis são atualizadas de acordo com a passagem dos minutos como na realidade. O data do jogo só é mostrada na passagem do dia.

O dia do jogo não possui vinte e quatro horas, mas inicia-se às oito da manhã, indo até um horário máximo calculado a partir da quantidade do atributo resistência que o jogador acrescentar. Assim os dias passam a ser maiores, dependendo deste atributo. O dia padrão, sem nenhuma resistência extra acrescentada termina as dezoito horas.

Uma hora antes (tempo virtual) de atingir o limite máximo de tempo do dia é executada uma função que dá alta a todos os paciente ainda na sala de espera. Nenhum paciente pode estar na sala de espera uma hora antes do dia passar, por isso o jogador deve sempre estar atento em gerenciar este recurso. Esses pacientes que recebem alta, se ainda não receberam nenhum tratamento, podem causar uma grande penalidade ao jogador.

Duas horas antes da passagem de dia nenhum novo paciente é gerado, para não ocorrer a situação em que um paciente é gerado muito próximo do momento em que todos recebem alta.

Quando o cronômetro atinge o limite máximo do dia é iniciada uma rotina de troca de dias, onde todas as variáveis referentes ao tempo são atualizadas. Quando a passagem do dia ocorre, a posição do jogador é retornada para um valor específico. Um cálculo contando o tempo do fim do dia anterior até o começo do seguinte é feito, e então todas as condições que dependiam do tempo para acontecer (como o dano causado pelos sintomas) são acrescentados deste valor. Dessa forma pacientes que pernoitam no hospital continuam sendo afetados por suas doenças, fazendo com que o medico fique atento ao local que os pacientes mais debilitados passem a noite, sem ser postos em risco.

A rotina de contaminação da cidade também é ativada na troca de dias, sendo que todos os pacientes que foram enviados para casa tem probabilidade de espalhar a doença na rua em que moram, e posteriormente, quando esta estiver totalmente contaminada, para as ruas vizinhas.

A interface do cronômetro é simples e se assemelha a um relógio digital. Este ativa todo o tempo para facilitar a gerência das ferramentas do hospital.

4.14 Interação de Enfermeira e Interação de UTI

Estes componentes pertencem a categoria de *NPCs*. Eles informam quais pacientes estão atualmente utilizando suas respectivas salas, e caso o jogador queira, permite alterar o tratamento do paciente selecionado.

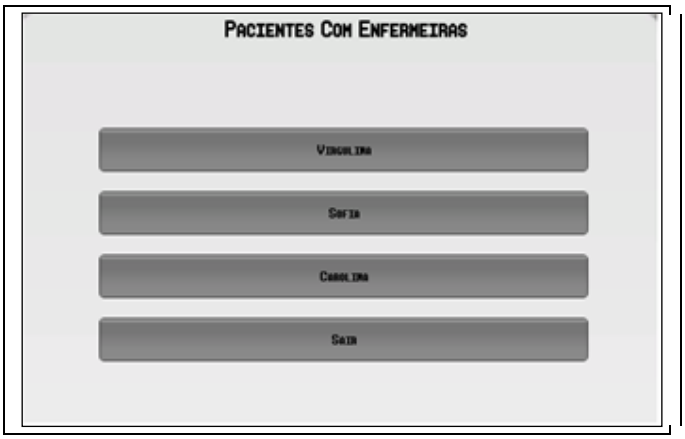


Figura 16 : Tela inicial do Interação de Enfermeiras / Interação de UTI

Esse script recupera a lista de pacientes alojados em suas salas e as listam em forma de botões com o nome dos pacientes (Fig. 16). No caso do *NurseInteraction*, se o jogador clica em um paciente uma outra interface abre com duas opções:

Ações do Paciente: é a listagem já explicada na seção 4.11.4. São as ações de movimentação e prontuário do paciente.

Ações da Enfermaria: É a listagem de todos os tratamentos possíveis para serem realizados pela enfermaria. Cada tratamento tem uma duração, que é calculada através de uma equação envolvendo o nível da enfermaria no momento, a idade do paciente e qual tratamento está sendo efetuado. Assim que um tratamento é selecionado é iniciada uma contagem até a duração máxima. Para melhor visualização e gerenciamento do jogador, é mostrada através de uma barra de progresso (Fig. 17). Quando a barra de progresso fica completa, significa que o tratamento é finalizado, e todos os sintomas abrangidos pelo tratamento são eliminados em caso de doenças não persistentes.

Se a doença for persistente cabe ao jogador descobrir exatamente qual doença é aquela e iniciar um novo tratamento. Esse tratamento também leva algum tempo sendo calculado da mesma forma que um tratamento comum.

No caso da UTI, quando selecionado o nome de um paciente da lista não haverá mais de uma opção para tipo de ações a tomar, apenas aparecerá diretamente a interface de ações do paciente.

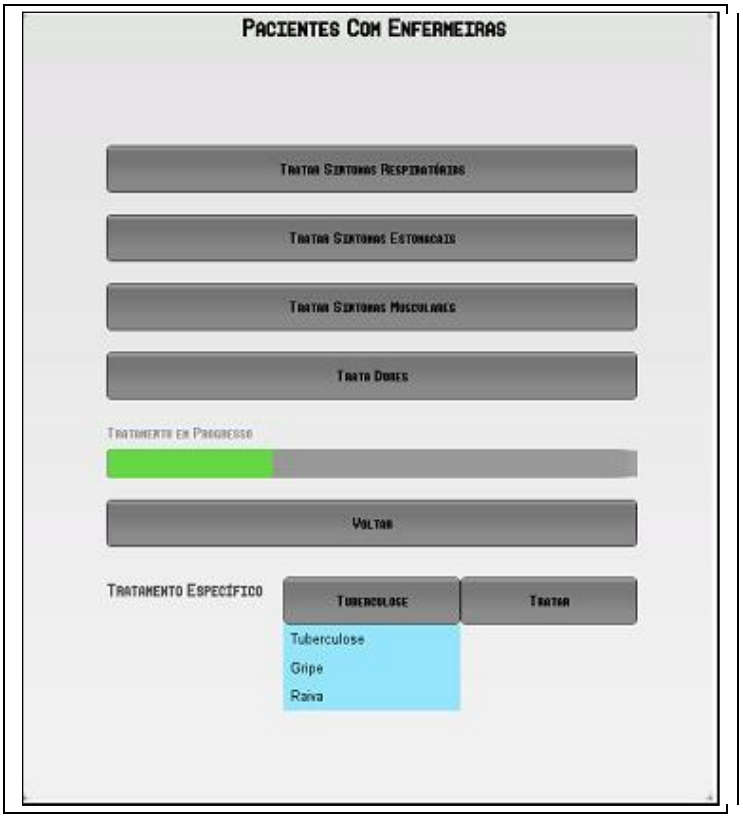


Figura 17: Tela de Pacientes com Enfermeiras

4.15 *Pathfinding*

Este componente foi originalmente criado por *Aron Granberg* e adaptado às necessidades do jogo. Ele é totalmente editável e apresentou desempenho aceitável para os fins deste projeto. Os nós são criados automaticamente dependendo do cenário e é possível criá-los para cenários planos e irregulares (Ref. 3).

Para editá-lo é preciso criar um *grid* em três dimensões que representa a área total do cenário que utilizará o *pahtfinding* (Fig 18). Esse tamanho é influenciado pelo número de nós que o usuário ache necessário para o seu uso. Quanto menor o número de nós menos preciso é o caminho, por outro lado quanto maior o número de nós mais custoso se torna o cálculo dos caminhos. Como o *Epidemics* utiliza um cenário estático e não muito complexo não foi necessário comprometer o desempenho para obter

caminhos mais precisos. O cenário também precisa ser dividido em duas camadas, uma camada para *colliders* caminháveis e outra para *colliders* não-caminháveis.

O *grid* com todos os seus respectivos nós pode ser calculado através do componente de pathfinding. *Colliders* definidos como não-caminháveis aparecem como retângulos vermelho no plano do cenário, e os caminháveis não apresentam retângulos e aparecem como linhas coloridas, onde cada cor representa uma área diferente separada pelos nós não-caminháveis.

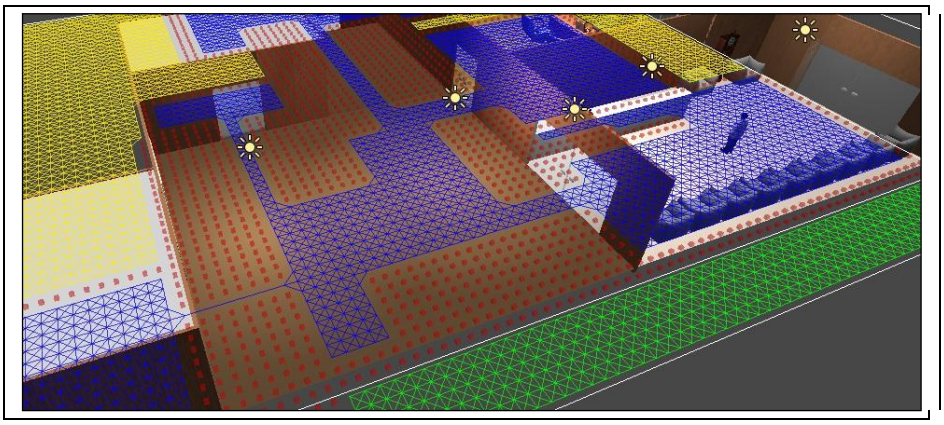


Figura 18: Exemplo de *Grid* calculado pelo *pathfinding*

Sempre que algum objeto da cena precisa se locomover através de *pathfinding* é necessário chamar um método que requer duas coordenadas, uma representando o ponto de origem do objeto que irá se movimentar, e outra representando o seu destino. Caso esses pontos estejam em áreas de cores diferentes, significando que são partes não conectadas por nós, ou em pontos inalcançáveis é retornado vazio.

Se o nó for alcançável a heurística de melhor caminho é efetuada através dos nós da origem e do destino e é retornado um conjunto de *waypoints* representando o melhor caminho que o objeto deve seguir (Ref. 8). Esse conjunto de *waypoints* é atualizado várias vezes enquanto o objeto está em movimento, através de uma variável de freqüência no *script* da inteligência artificial da locomoção, o *PatientRoomMovement.*

5 CONCLUSÃO

A proposta deste trabalho foi a de desenvolver um jogo de computador focado na área de epidemiologia em parceria com o departamento de epidemiologia da Universidade Federal Fluminense.

O principal objetivo deste jogo é o de proporcionar o aprendizado e fornecer conhecimento ao jogador de forma não convencional, divertida e interativa.

Dentro do escopo deste trabalho, pode-se concluir que o objetivo foi atingido e, além disso, abre margem para diversas melhorias do projeto. Na atual versão do *Epidemics*, é possível experimentar diversão e aprender sobre epidemias em algumas horas de jogo.

Entretanto, por limitações de cronograma, equipe e a ausência de artistas especializados, o jogo não pode atingir um nível comparável aos encontrados no mercado.

A fim de indicar os tópicos que podem ser relevantes de modo a dar continuidade ao desenvolvimento do sistema, serão apresentados no item a seguir alguns aspectos já mapeados para a melhoria do projeto em questão.

5.1 Trabalhos Futuros

Alguns elementos que podem ser trabalhados de modo a melhorar este projeto são:

- Arte Gráfica: No trabalho atual não foi incluída uma arte própria, apenas animações. Isso significa que o ambiente criado não foi desenhado especificamente para este jogo. Seria importante uma equipe de artistas para incrementar a qualidade visual dos cenários e cativar ainda mais o jogador.

- Implementar *Save Games* e *Load Games*: Atualmente, o *epidemics* não possui sistema de *save* e *load.* O jogador não poderá guardar o progresso do seu jogo depois de algum tempo.

- *Cinematics*: Outro aspecto importante em jogos de computador são os *cinematics*. Exemplos destes são vídeos de introdução. Visionou-se neste trabalho que poderiam ser utilizados vídeos explicativos durante o jogo, para transparecer mais o conteúdo e aumentar o nível de interatividade.

- Sons: Outro aspecto essencial para jogos que não foi desenvolvido neste trabalho foi a aplicação de uma trilha sonora própria. Acreditou-se que adquirir sons externos, mesmos livres para distribuição, descaracterizaria completamente o objetivo do trabalho. Propõe-se, então, a criação de sons específicos para o *Epidemics*.

REFERÊNCIAS BIBLIOGRÁFICAS

(1)Ministério da Saúde, Departamento de Vigilância Epidemiológica, Guia de Vigilância Epidemiológica, 7ª Edição, 2009.

(2)Ministério da Saúde, Doenças Infecciosas e Parasitárias Guia de Bolso, 8ª Edição, 2009.

(3)arongranberg.com, http://www.arongranberg.com/unity/a-pathfinding/, Janeiro 2011.

(4) Jae-jin Choi, Maya Character Animation. Sybex, 2006.

(5) unity3d.com, http://unity3d.com/support/documentation/, Janeiro 2011.

(6) Andrew Stellman e Jennifer Greene, Use a Cabeça! C#. O'Reilly Media, Inc. Alta Books, 2009.

(7) tutoriaisphotoshop.net, http://www.tutoriaisphotoshop.net/2011/06/criando-um-botao-web-com-layer-styles, março 2011.

(8) David M Bourg e Glenn Seemann, AI for Game Developers. O'Reilly Media, Inc, Julho 2004.

(9) Richard Rouse III, *game design*: Theory and Practice, 2nd Edition. Wordware Publishing. 2010.

(10) re-mission.net, http://www.re-mission.net/site/game/index.php. Abril 2011.

(11) pt.yupis.org, http://pt.yupis.org/jogos-medicina/. Março 2011.

(12) medicalgames.net, http://www.medicalgames.net/. Abril 2011.

(13) James Paul GeeWhat, Video Games Have to Teach Us About Learning and Literacy. Second Edition, Palgrave Macmillan, 2007.

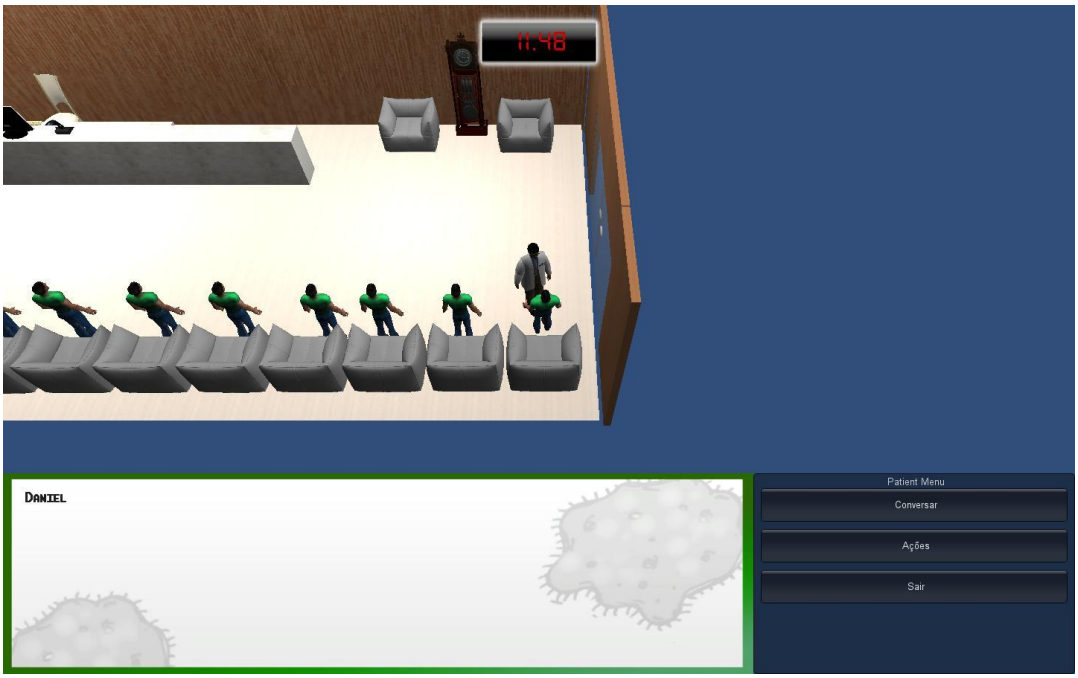
APÊNDICE

Apêndice A – Casos de Uso

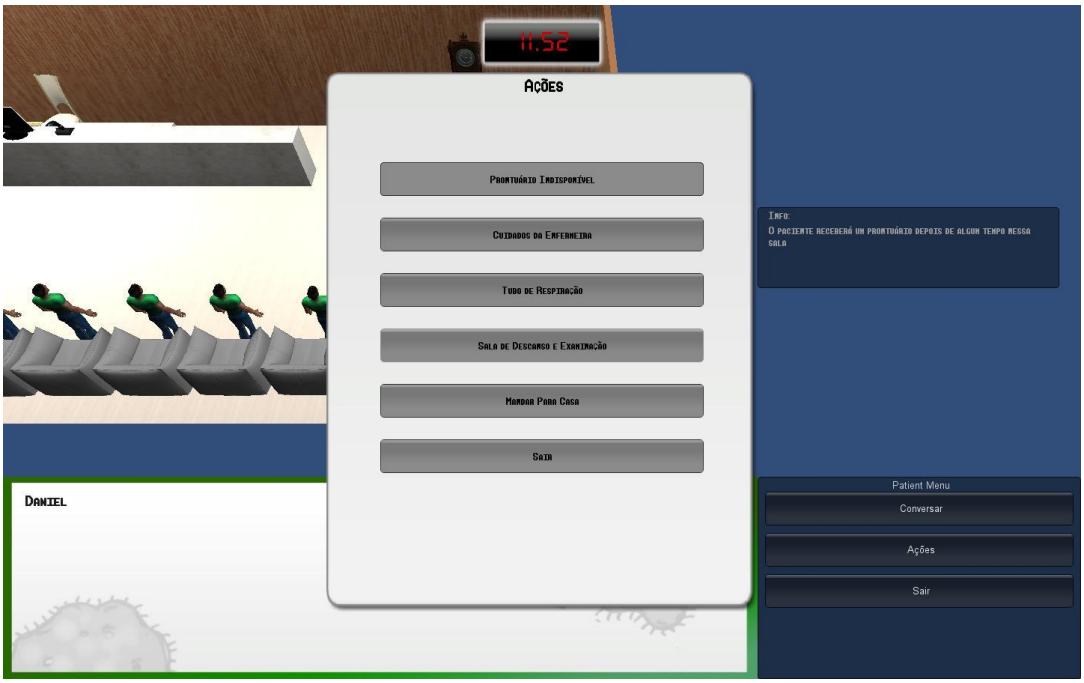
Nesta seção serão apresentados alguns casos de uso, que representam situações básicas no *Epidemics*.

- Visualizar o prontuário de um paciente.

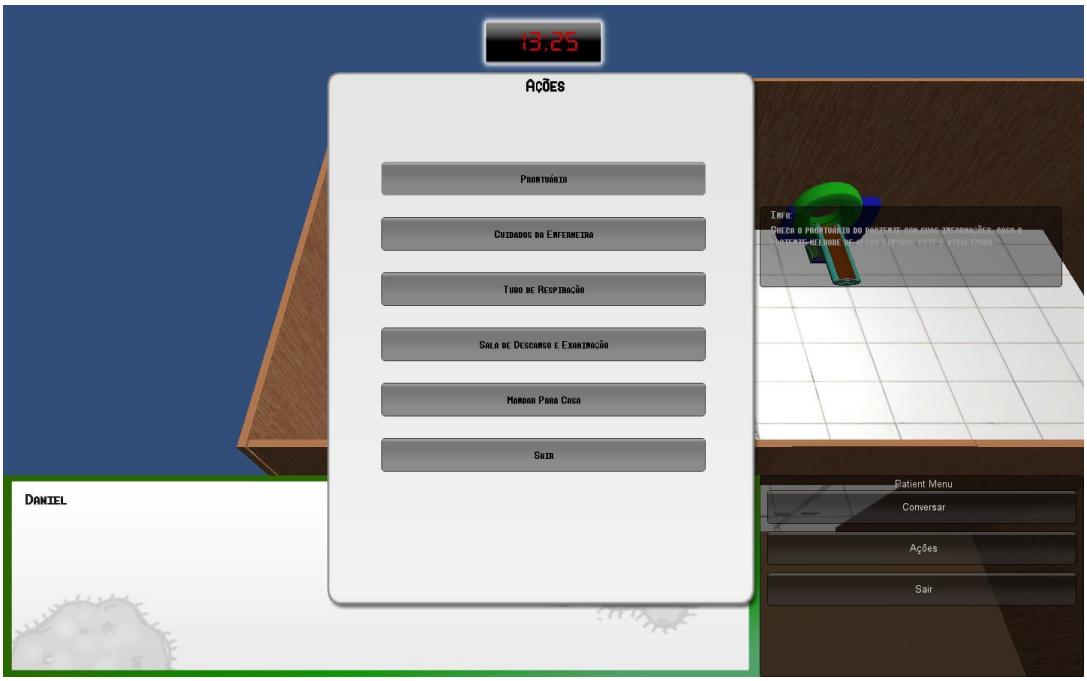
1. O jogo gera um novo paciente.
2. O paciente se move até uma posição disponível na sala de recepção.
3. O jogador pressiona o botão de interação dentro do alcance do paciente.
4. A janela de interação do jogador com o paciente abre.



5. O jogador seleciona a opção “ações”.



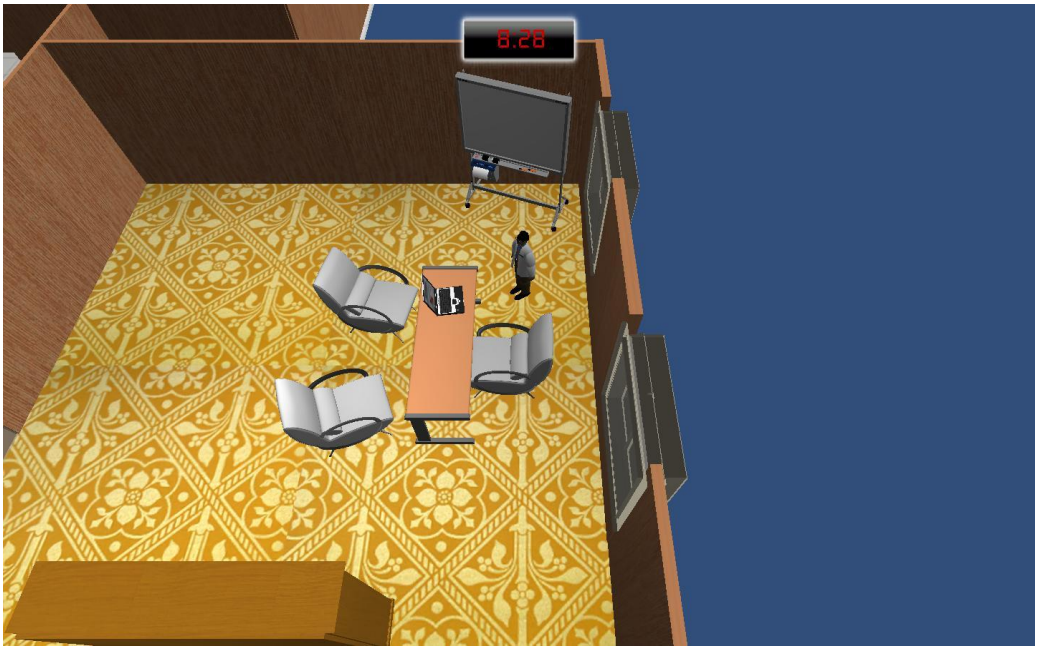
6. O jogador envia o paciente para a sala de examinação.
7. O jogo inicia a contagem para geração do prontuário.
8. O jogador interage com o paciente novamente.



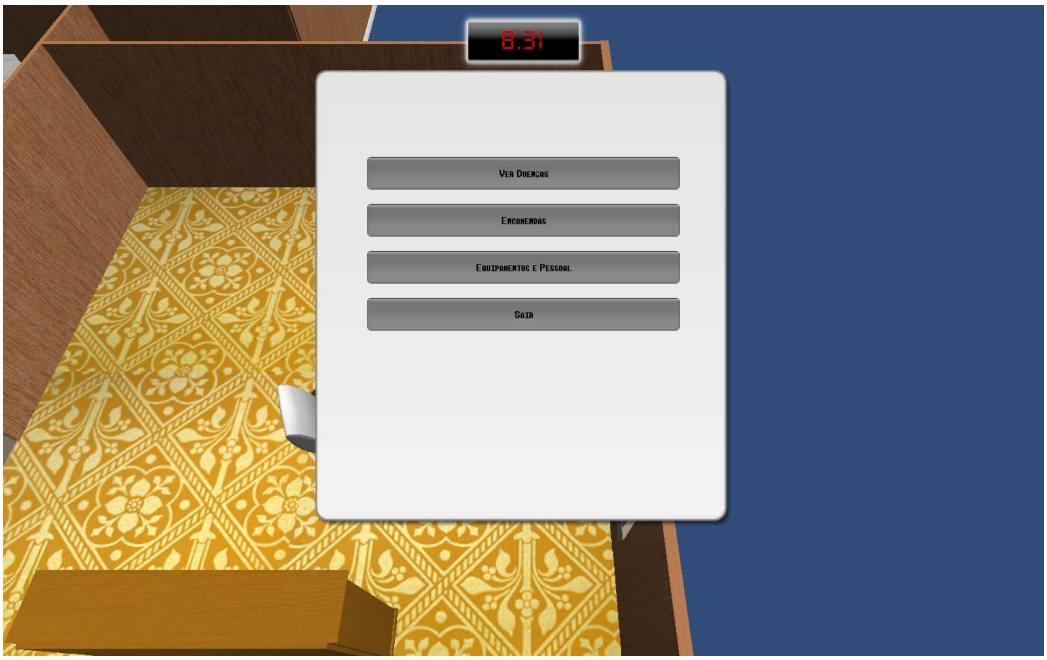
9. O jogador seleciona a opção “Ações”.
10. O jogador seleciona a opção “Prontuário”.

- Evoluir um nível da sala de Enfermaria.

1. O jogador se posiciona ao alcance da interação com o computador e pressiona o botão de interação.



2. O menu de interação com o computador é exibido na tela.



3. O jogador seleciona a opção “Equipamentos e Pessoal”.

4. O jogador seleciona o nível dois de Enfermaria.

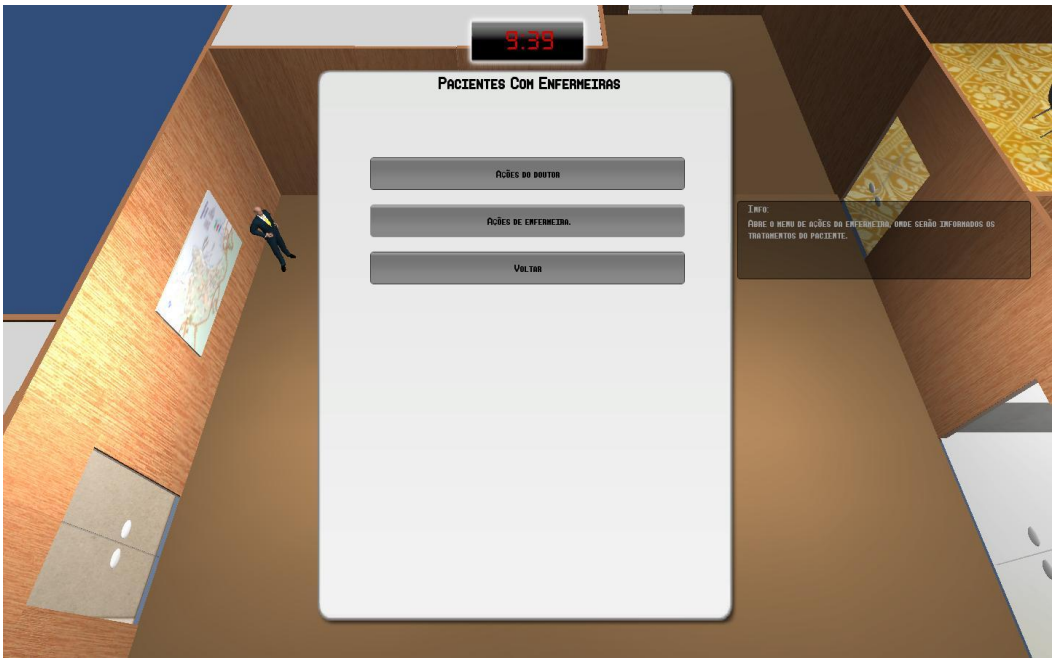
7. O jogador seleciona sair.

- Enviar um paciente para tratamento respiratório.

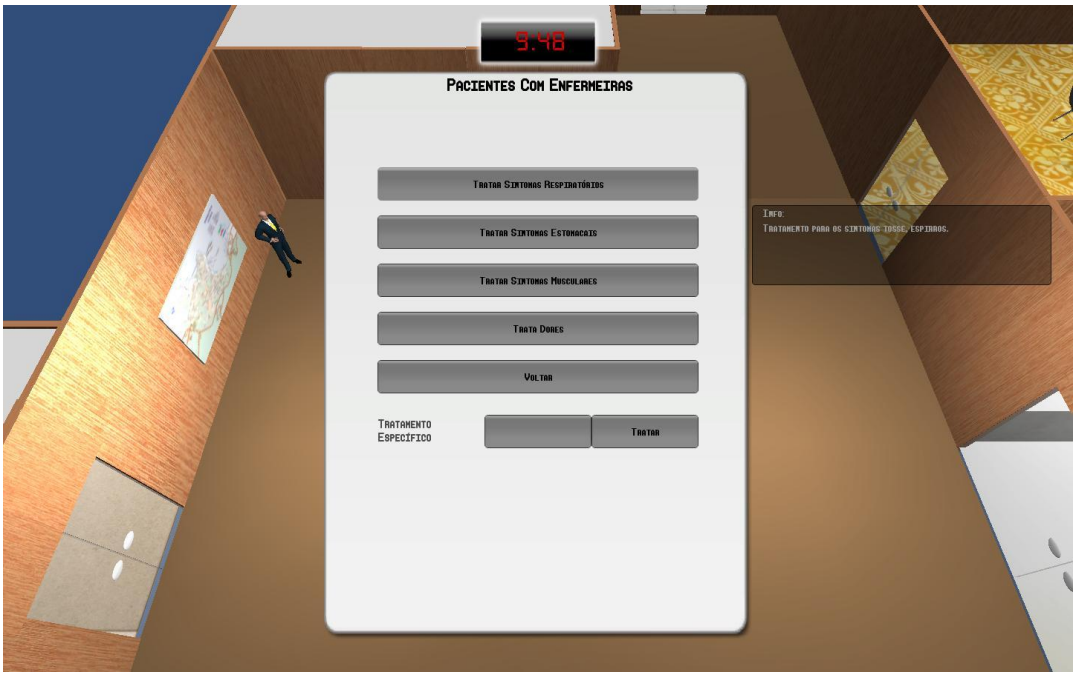
1. O jogo gera um novo paciente.
2. O paciente se move até uma posição disponível na sala de recepção.
3. O jogador pressiona o botão de interação dentro do alcance do paciente.
4. A janela de interação do jogador com o paciente abre.
5. O jogador seleciona a opção “ações”.

6. O jogador envia o paciente para a sala de enfermaria.

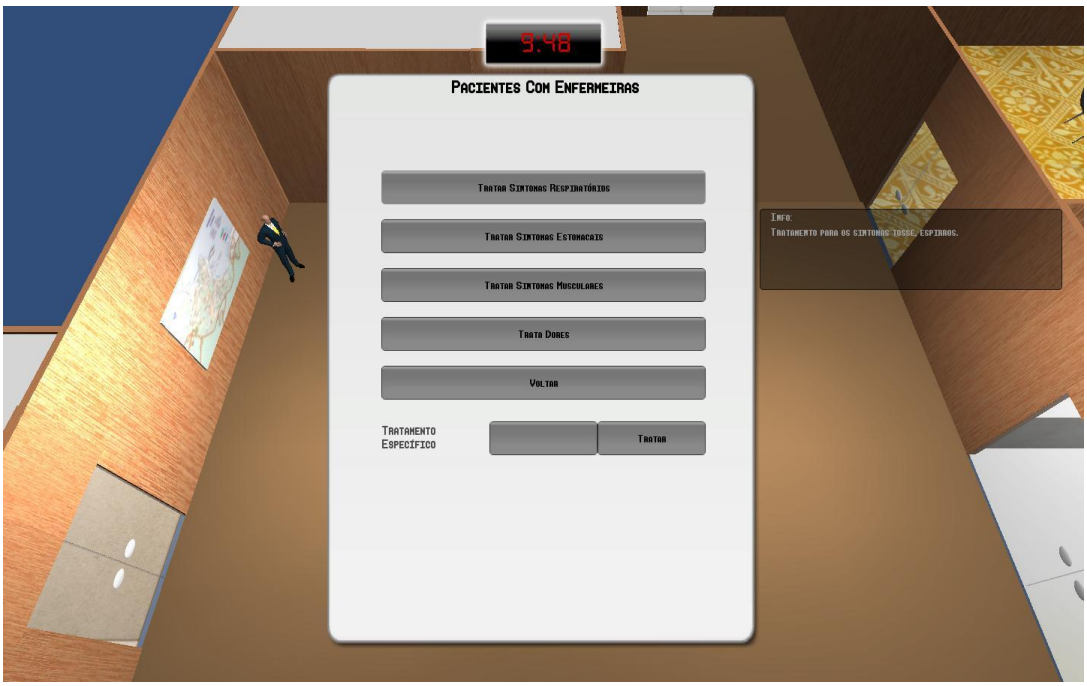
7. O jogador interage com o NPC da enfermaria.



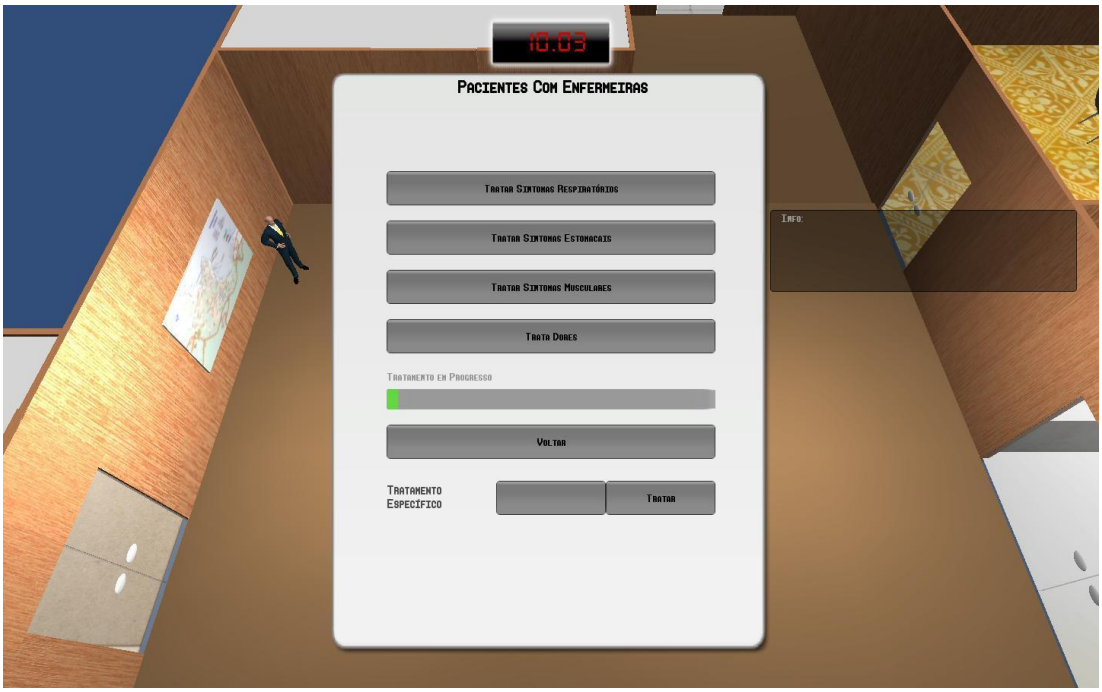
8. O jogador seleciona o nome de um dos pacientes.



9. O jogador seleciona o tratamento de enfermeira.



10. O jogador seleciona tratamento respiratório.
11. A barra de progresso simboliza a duração do tratamento.



12. O jogador fecha a tela de interação.