UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

AVATARES INTELIGENTES COMO APOIO AO PROCESSO DE ENSINO E APRENDIZAGEM DE CONCEITOS DE TESTES DE SOFTWARE

TRABALHO DE GRADUÇÃO

Luis Henrique Carvalho Rosa

Santa Maria, RS, Brasil
2014

AVATARES INTELIGENTES COMO APOIO AO PROCESSO DE ENSINO E APRENDIZAGEM DE CONCEITOS DE TESTES DE SOFTWARE

por

Luis Henrique Carvalho Rosa

Trabalho de Graduação apresentado ao Curso de Ciência da Computação da Universidade Federal de Santa Maria (UFSM), como requisito parcial para a obtenção do grau de
Bacharel em Ciência da Computação

Orientadora: Profª. Drª. Giliane Bernardi

Trabalho de Graduação Nº 386

Santa Maria, RS, Brasil

2014

Universidade Federal de Santa Maria
Centro de Tecnologia
Curso de Ciência da Computação

A Comissão Examinadora, abaixo assinada,
aprova o Trabalho de Graduação

AVATARES INTELIGENTES COMO APOIO AO PROCESSO DE ENSINO E APRENDIZAGEM DE CONCEITOS DE TESTES DE SOFTWARE

elaborado por
Luis Henrique Carvalho Rosa

como requisito parcial para obtenção do grau de
Bacharel em Ciência da Computação

COMISÃO EXAMINADORA:

Dr. Giliane Bernardi
(Presidente/Orientadora)

Dr. Ana Trindade Wink

Dr. Lisandra Manzoni Fontoura

Santa Maria, 3 de dezembro de 2014.

“Education is our passport to the future, for tomorrow belongs only to the people preparing today.”

Malcolm X

SUMÁRIO

AGRADECIMENTOS

Gostaria primeiramente agrader a minha mãe, Cristina Carvalho e meu pai, Luis Rosa, por não medirem esfoços para que eu concluísse mais essa etapa da minha vida, sem vocês essa minha conquista não seria possível. Ainda agradeço-lhes e a meus irmãos, Victória Rosa e Germano Rosa, por todo o apoio, carinho, compreensão e amor que dedicaram a mim.

Agradeço também a minha namorada Nayra Silva, por todo seu apoio, carinho, amor e atenção, pois sem você todo esse caminho seria muito mais difícil e pesado.

Também agradeço imensamente minha grande orientadora e amiga, Professora Drª Giliane Bernardi, por todo o tempo, atenção e ajuda que dedicaste a mim. Agradeço também pela confiança que depositaste em mim, acreditando sempre no sucesso deste trabalho.

RESUMO

Trabalho de Conclusão de Curso
Curso de Graduação em Ciência da Computação
Universidade Federal de Santa Maria

Avatares inteligentes como apoio ao processo de ensino e aprendizagem de conceitos de testes de software

Autor: Luis Henrique Carvalho Rosa
Orientadora: Giliane Bernardi

A atividade de teste é um elemento crítico, considerando que os custos associados às falhas de software usualmente são elevados. O presente estudo tem por objetivo inserir avatares inteligentes em um ambiente virtual 3D para aprendizagem na área de Teste de Software, com a premissa de realizar uma interação com os estudantes no mundo, discutindo assuntos pertinentes a um estudante de cursos da área de computação. A metodologia seguida neste trabalho pode ser dividida em dois momentos: uma sequência de atividades para desenvolvimento da peça teatral e dos agentes envolvidos; e o desenvolvimento dos agentes inteligentes conversacionais (*chatterbots*) que irão interagir com os estudantes. O ambiente desenvolvido será apresentado para um grupo de estudantes, onde será realizada uma avaliação de usabilidade, para analisar a efetividade dos NPC's, dos *chatterbots* e da peça teatral. A proposta é realizar a avaliação inicial com um grupo de voluntários da pós-graduação do Mestrado Profissional em Tecnologias Educacionais em Rede de forma a verificar a usabilidade do mundo de testes, peça teatral e agente conversacional. Espera-se, com esta avaliação inicial, possam-se analisar potencialidades e fragilidades da proposta desenvolvida para, futuramente, avaliar novamente o mundo criado com alunos de disciplinas específicas da área de teste de software, visando, além da avaliação de usabilidade do ambiente, avaliar o potencial pedagógico da abordagem proposta.

Palavras-chave: mundos virtuais 3D, *chatterbot*, NPC.

ABSTRACT

Undergraduate Paper
Undergraduate Program in Computer Science
Federal University of Santa Maria

Intelligent avatars to support the teaching and learning process of software testing concepts

Author: Luis Henrique Carvalho Rosa
Advisor: Giliane Bernardi

The testing activity is a critical element, whereas the costs associated with software failures are usually high. This study aims to insert intelligent avatars in a 3D virtual environment for learning in Software Testing area, with the premise of performing an interaction with students in the world, discussing matters pertaining to courses a student computing area. The methodology used in this work can be divided into two phases: a sequence of activities for development of the play and the actors involved; and the development of conversational intelligent agents (chatterbot) that will interact with the students. The developed environment will be presented to a group of students, where a usability evaluation will be conducted to analyze the effectiveness of the NPC, the chatterbot and play. The proposal is to conduct the initial evaluation with a group of graduate volunteers Professional Masters in Educational Technology Network in order to verify the usability of the world tests, play and conversational agent. It is expected with this initial assessment, can be analyzed strengths and weaknesses of the proposal developed for the future, reassess the world created with students from specific disciplines of software testing area, in order, in addition to environmental usability evaluation, evaluate the pedagogical potential of the proposed approach.

Keywords: 3D virtual worlds, chatterbot, NPC.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

AVEA	Ambiente Virtual de Ensino e Aprendizagem
LSL	*Linden Scripting Language*
OSSL	*OpenSim Scripting Language*
TIC	Tecnologias da Informação e Comunicação
NPC	*Nonplayer Character*
JETS	Jogo da Equipe de Teste de Software
UUID	Universally Unique Identifier

1 INTRODUÇÃO

Teste de Software pode ser considerado uma das sub-áreas da Engenharia de Software, responsável pela verificação e validação no decorrer do desenvolvimento, abrangendo diversas atividades de Garantia de Qualidade de Software (Pressman, 2011). Ainda para o referido autor, a atividade de teste é um elemento crítico, considerando que os custos associados às falhas de software usualmente são elevados. Desta forma, o processo de teste deve ser realizado por pessoas qualificadas, em ambiente e ferramentas adequadas, de forma cuidadosa e planejada. De acordo com Myers et al. (2004), agregar valor por meio de testes significa aumentar a qualidade ou a confiabilidade do *software*.

Considerando as colocações acima, é possível perceber a importância desta etapa da Engenharia de Software, evidenciando a necessidade de profissionais experientes e com conhecimento acerca do tema. Em Silva et al. (2011), é apresentado um panorama do ensino de Engenharia de Software em cursos de graduação, com ênfase na discussão de testes de software. Os autores destacam que os conteúdos, em sua maioria, não são explorados de forma detalhada, bem como as práticas educacionais adotadas usualmente são expositivas, com poucas atividades práticas e estratégias alternativas para incentivar e motivar os estudantes.

De acordo com Sklar (2003), para um estudante manter o interesse e progredir é necessário que haja uma maior interação do mesmo com os conteúdos educacionais. Ainda, de acordo com Silva (2012), de forma geral na área de Engenharia de Software é possível observar uma crescente elevação na busca por estratégias alternativas, tais como o uso de jogos, simuladores, objetos de aprendizagens, entre outros, como forma de aumentar motivação e despertar maior interesse dos estudantes por conteúdos usualmente mais teóricos. Entre as propostas discutidas, surge a possibilidade de utilização de Mundos Virtuais 3D como uma maneira de manter o interesse dos estudantes, por meio da disponibilização de ambientes considerados mais realistas, podendo trazer para a sala de aula uma proximidade maior com o meio profissional com que vão se deparar mais futuramente.

Mundos Virtuais 3D são ambientes online gerados por computador, onde o estudante pode interagir de maneira comparável ao mundo real, seja com objetos do ambiente ou com outros estudantes (BAINBRIDGE, 2010). A proposta de Mundos Virtuais 3D visa ampliar e diversificar os métodos tradicionais de ensino e aprendizagem, por meio da integração entre os estudantes e o ambiente, para com isso serem protagonistas da construção do seu conhecimento (ROCHA et al., 2012).

No projeto desenvolvido por Silva (2012), um Mundo Virtual 3D foi desenvolvido, com o objetivo de simular uma empresa de testes de software. Neste mundo foi criado um jogo sério para trabalhar temas como estratégia de Teste de Software, planos e casos de teste, permitindo aos estudantes e professores trabalharem de maneira mais lúdica o conteúdo exposto na sala de aula, tornando a experiência de aprendizado mais prazerosa e proveitosa. Durante a avaliação deste projeto junto a estudantes de um curso da área de computação, em uma disciplina de Engenharia de *Software*, foi possível analisar que as interações e execução das atividades do jogo ocorriam de forma mais expressiva quando o professor estava presente esclarecendo certas questões do ambiente, das regras do jogo, bem como de conteúdos acerca do tema testes de *software*. Por mais que o ambiente fornecesse informações sobre a execução das atividades do jogo, bem como uma integração com o ambiente virtual de ensino e aprendizagem Moodle permitisse aos estudantes, por meio do próprio mundo, obter os materiais didáticos da disciplina, foi relatada nos questionários respondidos a sensação de isolamento e "solidão" quando o professor não estava presente. Como proposta de trabalhos futuros, foi destacada a possibilidade de criação de agentes de *software* (avatares inteligentes), que atuassem como companheiros, agentes conversacionais, potencializando a interação e colaboração entre os estudantes e o mundo.

Com o intuito de dar seguimento ao projeto de Silva (2012), o presente trabalho traz a proposta de estender o mundo virtual criado, inserindo agentes de software, de forma a analisar as perspectivas descritas em Silva (2012). Por intermédio desses agentes espera-se diversificar ainda mais as maneiras de

ensinar Testes de Software, para obter melhores resultados no processo de ensino e aprendizagem.

A implementação de tais agentes no âmbito deste trabalho ocorrerá de duas formas. Inicialmente, será desenvolvida uma peça teatral, onde agentes de software (avatares inteligentes) representarão os atores da mesma, e atuarão com base no texto teatral descrito em Rios e Moreira (2013) que busca discutir, de uma forma mais lúdica, a importância de realização de testes em um *software*. A proposta é que o professor possa utilizar este novo ambiente criado nas primeiras aulas de testes de software, estabelecendo desde o início uma conexão com a importância de se trabalhar tais conteúdos em sala de aula.

Neste caso, os avatares não seriam mais meras representações de usuários no mundo virtual, mas personagens com comportamento autônomo, ou seja, agentes inteligentes que executam alguma ação, depois de receberem uma sequência de percepções (TEICHRIEB, 1999).

A segunda forma com que agentes serão inseridos neste trabalho irá ocorrer por meio da criação de agentes conversacionais (*chatterbots*) que, munidos de uma base de conhecimentos sobre conceitos associados a testes de *software*, poderão interagir com os estudantes sobre o tema. O objetivo destes agentes é possibilitar que os estudantes possam, a qualquer momento, ou após assistirem a peça, esclarecer suas dúvidas mesmo sem a presença de um professor no ambiente.

1.1 OBJETIVOS

1.1.1 OBJETIVO GERAL

Inserir avatares inteligentes em um ambiente virtual 3D para aprendizagem na área de Teste de Software, com a premissa de realizar uma interação com os estudantes no mundo, discutindo assuntos pertinentes a um estudante de cursos da área de computação.

1.1.2 OBJETIVOS ESPECÍFICOS

Como objetivos específicos deste trabalho destacam-se:

- Desenvolver um teatro (instalação física) no ambiente *OpenSimulator*, inserido no mundo virtual desenvolvido em Silva (2012) para apoio ao ensino e aprendizagem de Teste de Software;
- Desenvolver os agentes de software que irão representar os atores na peça a ser encenada;
- Reproduzir a adaptação da peça teatral 'Testar ou não testar - Eis a questão', que retrata o ambiente do setor de Teste de uma empresa fictícia, conforme expressa no livro *Teste de Software*, de RIOS; MEDEIROS, (2013), que está descrita no Apêndice A;
- Desenvolver os agentes conversacionais, que irão interagir com os estudantes sobre conteúdos relacionados a testes de software;
- Desenvolver a base de conhecimentos para os agentes conversacionais;
- Analisar a potencialidade de utilização do ambiente desenvolvido no processo educacional, por meio da aplicação do mesmo em uma turma da disciplina de Engenharia de Software.

1.2 ESTRUTURA DO TEXTO

O presente trabalho está estruturado da seguinte forma: os Capítulos 2, 3 e 4 apresentam uma revisão bibliográfica sobre testes de software, mundos virtuais e as tecnologias envolvidas e avatares inteligentes em mundos.

No Capítulo 5 é apresentada e descrita proposta do ambiente, que discorre acerca da metodologia de desenvolvimento.

O capítulo 6 aborda o desenvolvimento do ambiente e tudo o que foi implementado, os resultados e discussões preliminares são apresentadas no capítulo 7 e por fim o capítulo 8 traz as conclusões deste trabalho.

2 TESTE DE SOFTWARE

A disciplina de Engenharia de Software é vista por Sommerville (2007) como de grande presença durante todo o processo de desenvolvimento de um sistema, desde o estágio de especificação até mesmo após sua implementação. Caracterizado com uma das fases do processo de produção de um software a aplicação de testes é uma maneira muito usual de averiguar se o processo de desenvolvimento foi eficiente e se o produto final cumpre com seu propósito (SOMMERVILLE, 2007). Os autores MYERS et. al. (2004) definem testes de software como a simulação de situação a fim de encontrar erros no sistema. Já a IEEE (2004) caracteriza teste de software como uma atividade direcionada a avaliação para implementação de melhorias no sistema.

Testes podem ser considerados uma ferramenta do processo de qualidade de software. A qualidade da aplicação varia de forma significativa de um software para outro. A norma ISO 9126 estipula como atributos qualitativos a funcionalidade, confiabilidade, usabilidade, eficiência, manutenibilidade e portabilidade (ABNT, 2014).

Outra definição de teste de software consiste em ações que, partindo de uma avaliação de um atributo ou capacidade do sistema consiga constatar se o mesmo responde da maneira esperada (RIOS E MOREIRA *apud* HETZEL, 2013).

Para Rios e Moreira (2013), teste de software trata-se de execução controladamente do sistema como o intuito de verificar seu comportamento baseado nas especificações, ou seja, um tipo de validação. Estima-se que a cada 100 instruções executadas o número de defeitos é de 1 a 3 em um programa liberado para fase de testes (RIOS E MEDEIROS *apud* BEIZER, 2013). Contudo, é fato que não se pode, na prática, testar todas as possibilidade e garantir que o sistema estará livre de bugs, então por que testar? Porque por meio dos testes poderá se descobrir os erros e executar melhorias no sistema o mais rápido possível.

A literatura aponta que uma tardia descoberta de um erro torna mais difícil sua correção e encarece de forma exponencial o projeto (RIOS E MEDEIROS *apud* BOEHM, 2013). Exemplos desses custos podem ser evidenciados no

orçamento para correção de erros, que é de aproximadamente 20% do orçamento de manutenção, que responde por 67% dos custos totais de um software (RIOS e MEDEIROS *apud* MARTIN e MCCLURE, 2013). Outra utilização relevante de teste de software se dá na manutenção de softwares, que em sua grande maioria são os mesmos realizados na fase de desenvolvimento.

É pontuado por Rios e Medeiros (2013) que testes com maior eficiência durante o desenvolvimento resultam em redução dos custos na manutenção, modificações em partes do programa podem resultar em novos problemas, até mesmo em partes não modificadas. A utilização de ferramentas simuladoras podem dar resultados que pessoas não encontrariam equipes com maior especialização obterão softwares de maior qualidade e com menor custo total.

Portanto, um software testado possui maior qualidade e confiabilidade (MYERS et. al., 2004). Devido ao crescimento das exigências da indústria de software, bem como o aumento na complexidade dos sistemas, cada vez mais se trabalha, dentro dos projetos, com o conceito de testadores especializados, profissionais voltados apenas para testes, visando minimizar os gastos financeiros e maximizar o desempenho no desenvolvimento (KANIJ; MERKEL; GRUNDY, 2011). Por conseguinte, é importante que os graduados em curso da área de Computação estejam preparados para o mercado.

2.1 Contexto educacional

Alguns pesquisadores apontam que nos cursos de graduação da área da computação o conhecimento sobre Engenharia de Software pode não estar sendo abordado de maneira correta. Pesquisas mostram que a maioria dos bacharéis da área de computação virá dominar essa área de conhecimento somente quando se fizer necessário aplicá-la (WANGENHEIM; SILVA, 2009).

Por contradições no currículo dos cursos da área de computação, conforme Stroustrup (2010), o formado encontra barreiras para atender as necessidades do mercado, pois apresenta deficiências nas áreas de teste e depuração de sistemas (ASTIGARRAGA et. al., 2010). Tais deficiências são indicativas de uma carência na diversificação das práticas educacionais (SILVA, 2012). O referido autor

discorre ainda sobre o engessamento do modelo tradicional de ensino de maneira a mostrar que ele pode ser pouco participativo, comunicativo, competitivo e que pode atrapalhar o desenvolvimento da visão sistêmica do estudante para resolução de problemas.

Como estratégias alternativas para solucionar esses problemas tem-se proposta de implementação de jogos, dinâmicas de grupo, atividade EAD (educação à distância) e atividades lúdicas no aprendizado por analogia (PRIKLADNICKI et. al., 2009).

Com o propósito de trabalhar essas estratégias de didáticas alternativas Silva (2012) propôs uma abordagem de aprendizagem baseada em jogos. Em seu trabalho foi desenvolvido o jogo sério JETS (Jogo da Equipe de Teste de Software), que simula o setor de teste de software de uma empresa, como maneira de motivar os estudantes. Este jogo foi implementado imerso em um mundo virtual de três dimensões. Por este trabalho de graduação tem como foco uma expansão do mundo criado por Silva (2012), sendo apresentado o trabalho da autora em detalhes na próxima seção.

2.2 Mundo Virtual de Testes de Software e o Jogo da Equipe de Teste de Software

O mundo virtual criado por Silva (2012) trata-se de um ambiente virtual educacional 3D voltado para trabalhar conceitos discutidos e trabalhados anteriormente em sala de aula, referentes a o desenvolvimento de casos e estratégias de Teste de Software. O mundo aplica a ideia de Aldrich (2009), que propõe a união de mundos virtuais, jogos e simulações educacionais formando, o que ele denomina, como Ambientes Virtuais Totalmente Imersivos. Neste mundo foi criada uma empresa fictícia que atua como testadora de software, onde tem-se profissionais como testadores, gerentes de testes e líderes de equipe, conforme a Figura 1.

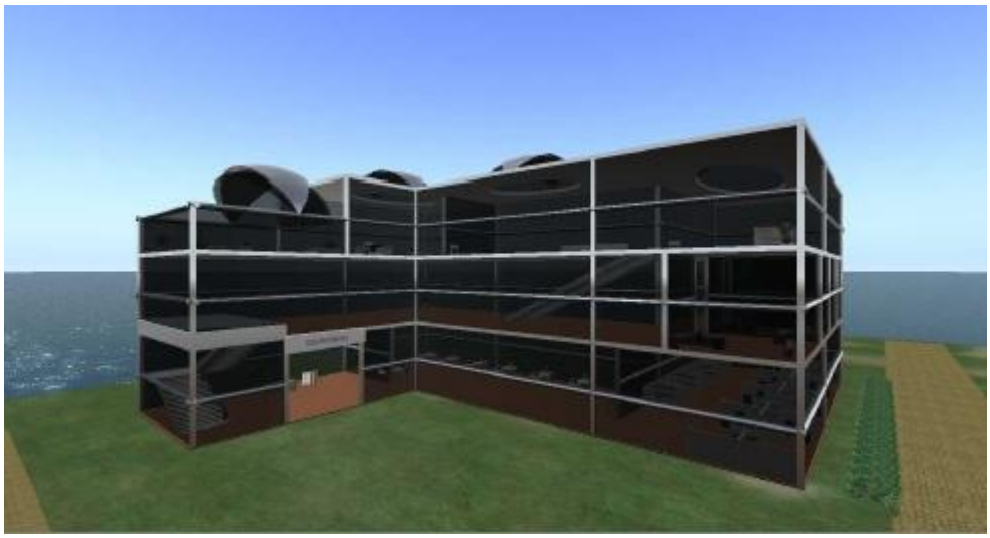

Figura 1. Prédio da Empresa de Teste

Dentro desta empresa, os estudantes podem jogar o JETS - Jogo da Equipe de Teste de Software, classificado por Silva (2012) de diversas maneiras:

> (...) como um Simulador, tendo em vista que simula atividades de uma Equipe de Teste de Software. Em relação aos gráficos, o jogo é classificado como 3D, pois foi desenvolvido em um mundo virtual 3D. Quanto ao número de jogadores ele é classificado como *Multiplayer*, pois pode ser jogado por um ou vários estudantes ao mesmo tempo, com a supervisão, ou não, do professor. Quanto ao tipo de interação, o JETS pode ser classificado como Múltiplos Jogadores Individuais Versus Jogo, pois cada jogador deve responder aos desafios do jogo. No entanto, o JETS também pode ser classificado como Jogo Cooperativo, pois os estudantes jogadores podem auxiliar seus colegas a atingirem os objetivos do jogo (SILVA, 2012).

O jogo foi desenvolvido na plataforma tridimensional de mundo virtuais *OpenSim* e integrado ao Ambiente Virtual de Ensino-Aprendizagem (AVEA) Moodle. Um dos diferenciais do trabalho de Silva (2012) consiste justamente nesta integração do AVEA Moodle com o *OpenSim*, o que permite que o professor poste conteúdos no mundo virtual por meio do próprio Moodle.

O objetivo ao jogar é progredir na empresa, passando por todos os cargos, até alcançar o posto de líder da equipe de teste. O professor é que determina todos os desafios e pontuações de cada fase, por meio de questionários no AVEA Moodle. Com isso o professor possui total controle sobre as atividades e pode analisar o desempenho de cada estudante e o tempo que cada um levou para

concluir a fase (SILVA, 2012).

Silva (2012) explica que a proposta do jogo é de apoiar o ensino e aprendizagem, pois os estudantes já devem possuir uma base teórica trabalhada em sala de aula previamente, para que possam realizar as atividades do jogo. Ainda, acrescenta que o jogo não deve ser um fator excludente de um professor. Considerando estas colocações, este trabalho focou-se em ampliar o mundo criado de modo a gerar discussões a respeito dos conceitos iniciais da área de Testes de Software, diversificando também esta etapa prévia teórica. Por se tratar de um ambiente desenvolvido a partir de uma plataforma de criação de mundos virtuais em três dimensões, o próximo capítulo discute sobre esta tecnologia, fundamental para o desenvolvimento deste trabalho.

3 MUNDOS VIRTUAIS 3D

Mundos virtuais 3D são ambientes imersivos gerados por computador, no qual o usuário pode realizar interações semelhantes ás do mundo real, podendo essas serem com os mais variados focos, como trabalho ou lazer. Essas interações do usuário ocorrem por meio de um personagem no mundo virtual, denominado avatar, com possibilidade de interação por intermédio de canal de voz, chat de texto e até mesmo expressões corporais e faciais (BRAINBRIDGE, 2010). Na combinação de gráficos 3D interativos, tecnologia de simulação, *Voice over Internet Protocol* e mídias digitais que habilitam ilimitadas possibilidades para a comunicação, colaboração e exploração, estão alicerçados os mundo virtuais (HODGE, COLLINS E GIORDANO, 2011). Por intermédio do seu conjunto de ferramentas, a experiência de imersão de um usuário se torna mais agradável e real, levando assim o usuário a interagir intuitivamente com o ambiente (MATTAR E VALENTE, 2007).

As qualidades singulares desses mundos proporcionam experiências sensoriais, imersivas, de simulação, de modelagem, colaborativas e atividades para o aprendizado experimental (MATTAR E VALENTE, 2007). Percebendo o potencial dos mundos, tanto a academia quanto empresas estão buscando se beneficiar desses ambientes, onde diversos usuários podem realizar interações, mesmo que estejam fisicamente separados (WANKEL E KINGSLEY, 2009).

Considerando o âmbito educacional, segundo Azevedo e Elia (2011), essas ferramentas computacionais proporcionam aos usuários novas formas de aprendizagem permitindo a construção de cenários e recursos que recriam/simulam com mais fidedignidade o mundo real. Ainda, para (OSÓRIO et al. 2004), o paradigma 3D oferece a possibilidade de representar a informação de um modo realístico, organizando-a de uma maneira espacial e tornando sua visualização mais intuitiva por ser mais natural ao ser humano.

Um dos elementos importantes em mundos virtuais é a representação do usuário no mundo. Essa representação é por intermédio de personagem, podendo facilmente ser personalizado conforme as preferências do usuário, podendo ser humano ou não. Esse personagem é denominado avatar, termo que possui sua

gênesis na mitologia hindu, que significa corpo temporário utilizado pelos deuses para circular pela terra (MACHADO, 2002). Avatares são qualificados como versões digitais do usuário, que possuem poder ilimitado sobre o mundo, até mesmo sobre as leis da física (DAMER, 1997).

Os mundos virtuais 3D são classificados de duas maneiras: de plataforma proprietária e *open source*. Nas proprietárias, fica a critério da empresa desenvolvedora as formas de distribuição, uso e autorizações para modificação; já as plataformas *open source*, por serem iniciativas livres, compartilham seu código de maneira a permitir que todos os usuários possam realizar modificações (CARMO, 2013). As principais plataformas de mundo virtuais 3D são *Second Life, Active Worlds, Open Wonderland* e *OpenSimulator,* sendo a primeira e a última de maior relevância para este trabalho.

A plataforma proprietária *Second Life* é uma das mais conhecidas e utilizadas. Sua popularidade se dá pela alta adesão de diversas instituições de ensino como Universidade do Vale do Rio dos Sinos (Brasil), Universidade de Trás-os-Montes e Alto Douro (Portugal), Boise State University (Estados Unidos da America) e University of St Andrews (Reino Unido). Desenvolvida em 2003 pela *Linden Lab*, o *Second Life* é um ambiente para o usuário ser, construir e vender o que quiser, explorar uma grande diversidade de mundos, jogar, ver apresentações e participar dessa que é a maior economia virtual do mundo (LINDEN LAB, 2013). O SL é baseado em uma economia própria, cuja moeda é o dólar *Linden*. Autores como Hodge, Collins e Giordano (2011) apontam um grande potencial no SL, contudo por se tratar de uma plataforma proprietária não possibilita sua alteração e criação, a menos que seja adquirido um terreno.

O *Second Life* trabalha com a linguagem *Liden Scripting Language* (LSL), linguagem própria do ambiente baseada em eventos, com sintaxe semelhante a linguagens orientadas a objeto. Essa linguagem é aplicada internamente no ambiente, estando presente em scripts de animação dos objetos e comunicação com ambientes externos ao mundo (SECOND LIFE, 2009).

Uma plataforma com grande popularidade é a iniciativa *open source* OpenSimulator. Por se tratar da plataforma utilizada em Silva (2012), será a ferramenta de criação de mundos virtuais 3D utilizada no desenvolvimento do

presente trabalho. Por sua compatibilidade com o SL, a mais popular, com maior suporte e por esse trabalho se tratar de um trabalho anterior desenvolvido neste ambiente, a próxima seção discorrerá com detalhamento sobre o mesmo.

3.1 OpenSimulator

O *OpenSimulator*, ou *Opensim*, como é popularmente conhecido, é uma iniciativa *open source*, desenvolvida em C#, criada para o desenvolvimento de ambientes virtuais 3D. Possui alta compatibilidade com os protocolos de comunicação do SL e o trabalha com *scripts* em LSL. Está disponível em versões para sistemas operacionais (SO) baseados em Unix e Windows, é distribuído sobre a Licença BSD (*Berkeley Software Distribution*) e é mantido por membros da comunidade (OPENSIMULATOR, 2014).

O *Opensim* tornou-se extremamente popular devido a sua agradável interface gráfica, funcionalidades para comunicação e seu sistema para modelagem de objetos, que é simples e intuitivo. Outra de ponto a seu favor é a sua compatibilidade com o ambiente virtual comercial, *Second Life*. Essa compatibilidade é muito atrativa, pois o desenvolvimento de artefatos e *scripts* para o *Second Life* é muito grande, possuindo também muita documentação disponível para pesquisas.

O modos de configuração do *OpenSim* são: *standalone* e *grid*. No modo *standalone*, que em tradução livre quer dizer ‘autônomo’, é possível criar diversas regiões, entretanto só poderá ser executado em uma máquina. Nesse modo, um processo cuida de todas as operações do simulador, esse processo é o *Opensim.exe*. Já o modo *grid*, tradução livre ‘rede’, trata-se do modo de configuração no qual a execução dos dados poderá ser realizada em varias instâncias. No modo *grid*, o responsável pela execução dos dados será o processo *Robust.exe*, isso acaba tornando o processo *OpenSim.exe* em um mero servidor de região. Por suas características esse modo de configuração é o utilizado para módulos multi-usuário. Além das possibilidades de instalação localmente e em servidores, o *OpenSim* possui mundos virtuais que podem ser atualizados gratuitamente, exemplo *OSGRID19*.

Para que o usuário possa acessar o mundo virtual é necessária a utilização

dos chamados visualizadores. Esses permitem ao usuário a visualização gráfica do mundo virtual. Como exemplos temos o *Imprudence*[1], *FireStorm*[2], *Singularity*[3] e *Radegast*[4].

Após conectar-se ao *OpenSim* ele apresenta uma diversidade de ferramentas para estabelecer comunicação, podendo elas serem o uso de áudio ou conversas via chat. Ainda á possível modelar objetos 3D de maneira simples e intuitiva, por meio da manipulação de primitivas disponíveis. Ele ainda suporta importações de objetos de repositórios como o *Google SketchUp* e *OpenSim Creations*[5], podendo inserir arquivos de áudio e imagens.

Para construir um mundo virtual ainda mais interativo e imersivo, o *OpenSim* possibilita a criação de animações tanto para os objetos quanto para os avatares, isso é possível com a utilização de *scripts*. Eles são desenvolvidos pelas linguagens de programação *LSL* (*Linden Scripting Language*), linguagem dos criadores do *Second Life*, a *OSSL* (*OpenSim Scripting Language*), uma extensão do *LSL*, C#, *Java Script* e *Visual Basic (.NET)*. Neste trabalho será utilizada a linguagem *LSL*, apresentada mais detalhadamente na próxima subseção.

3.1.1 Linguagem *LSL (Linden Scripting Language*)

A *LSL* é a linguagem desenvolvida para os usuários programarem os objetos tridimensionais em torno deles, para, com isso, torná-los interativos e interessantes para os usuários do mundo, estes *scripts* possibilitam que objetos do mundo real sejam criados com maior verossimilhança (HEATON, 2007).

A *LSL* é considerada uma linguagem orientada a eventos, por se tratar de uma linguagem de *scripts* possui semelhança com linguagens como *Python* e *Java Script*. A *LSL* trabalha com todos os tipos de dados mais utilizados, como *integer*, *float*, *string* e *vector*. A linguagem também trabalha com tipos de dados particulares com *key* (são as chaves de registros dos objetos) e *rotation* (utilizado

[1] http://wiki.kokuaviewer.org/wiki/Imprudence
[2] http://www.firestormviewer.org/
[3] http://www.singularityviewer.org/
[4] http://radegast.org/
[5] http://opensim-creations.com/

para rotação dos objetos). Os controladores de fluxo utilizados são ‘*do while*’, ‘*if else*’, ‘*for*’ e ‘*state*’. A linguagem oferece aproximadamente 40 eventos, ou funções próprias, como *touch_start* (reage ao toque do usuário) e *collision_start* (detecta a colisão entre objetos e usuários).

A linguagem *LSL* não possui um compilador muito otimizado, contudo, por compilar para um bytecode antes de executar, apresenta boa eficiência. Ela também possui algumas limitações, como o tamanho máximo de 16 kb para um script. Outro problema é o gerenciamento de *heap*, ela não possui uma implementação muito sofisticada, pois um *scripter* não aponta o quanto do *heap* está livre e nem o tamanho do maior bloco contíguo disponível (COX, 2009).

A linguagem apresenta uma *wiki* muito completa, onde pode-se encontrar uma variada lista de funções que a linguagem possuem como *llDetachFromAvatar* (retira o objeto do avatar), *llRegionSayTo* (envia uma mensagem para um avatar ou objeto especificado) e *llRemoveInventory* (remove do inventário um determinado item), entre outras que podem ser muito úteis. Além da possibilidade de animar primitivas e objetos, ela possibilita também animação de avatares e criação de agentes de software. Na próxima seção trabalharemos o conceito de mundos virtuais 3D na educação.

3.2 Mundos Virtuais 3D na Educação

Não podemos falar sobre mundos virtuais na educação sem uma introdução sobre Tecnologias de Informação e Comunicação (TIC). As TIC, quando usadas no âmbito educacional, consistem na utilização de ferramentas tecnológicas e de comunicação como amplificadores da interação e colaboração entre os estudantes e professores, para a construção do conhecimento. Com essa nova pedagogia aumenta a responsabilidade do professor ao produzir materiais de aprendizado que incluam essas tecnologias (WANKEL e KINGSLEY, 2009). Dentre as tecnologias utilizadas por essa nova prática pedagógica vem tendo destaque os mundos virtuais 3D (CHEN et al., 2010).

Por sua proposta de emulação do mundo real, pelo intermédio da computação gráfica, os mundos virtuais, diferente dos mundos com simples

interações por texto, trazem uma sensação de imersão mais satisfatória, o que traz mais motivação aos estudantes (SCHMITT E TAROUCO, 2008). Por suas qualidades, os mundos virtuais estão alcançando popularidade no desenvolvimento de ambientes colaborativos educacionais. Os mundos são novas formas para construção do conhecimento, pois permitem que os usuários construam ambientes que retratem com maior fidelidade o mundo real (AZEVEDO; ELIA, 2011). Segundo Osório et al. (2004), o paradigma 3D propicia que, por meio de representações de um modelo mais realístico, visualizações tornem-se mais intuitivas por sua maior familiaridade com o mundo real.

Teóricos afirmam que a formação do conhecimento acontece por meio da interação com outras pessoas e com objetos de conhecimento (SCHLEMMER; BACKES, 2008). Mundos virtuais 3D, por trazerem uma grande diversidade de recursos, apresentam-se como uma contribuição significativa para o processo de ensino e aprendizagem (DALSASSO, 2013).

Savin-Baden (2010) destaca que a adoção de mundos virtuais 3D na educação, pode ser benéfica devido a alguns fatores, tais como:

- Apoio à educação à distância;
- Aprendizado por imersão;
- Promoção da aprendizagem por dialógica;
- Horizontalidade das relações de poder no aprendizado;
- Estímulo à criatividade;
- Incentivo a exploração.

Algumas literaturas classificam mundos virtuais como uma boa ferramenta na educação à distância (RITZEMA E HARRIS, 2008; WANKEL E KINGSLEY, 2009; HODGE, COLLINS E GIORDANO, 2011). Para Savin-Baden (2010), as qualidades imersivas dos mundos virtuais, tornam a educação à distância mais eficaz, além de potencializar as interações e integração dos estudantes. Já Brito et al. (2013), discute que a utilização de mundos virtuais desenvolve o processo de ensino-aprendizagem abrangendo qualquer dimensão de aprendizagem não somente no ensino à distância, mas também no ensino presencial.

Muitos trabalhos ainda destacam as possibilidades dos mundos virtuais para a educação de nível superior (WANKEL E KINGSLEY, 2009; SAVIN-BADEN, 2010; VINCENTI E BRAMAN, 2011). Com mundos virtuais também é possível trabalhar com diversas abordagens de ensino como palestras, seminários, exibição de vídeos, simulações, performances virtuais, debates e interação com agentes de software (SAVIN-BADEN, 2010).

Na próxima seção será abordado o uso de agentes inteligentes imersos em mundos virtuais 3D, o foco deste trabalho.

4 AVATARES INTELIGENTES EM MUNDOS VIRTUAIS 3D

Este capítulo tem como objetivo discutir sobre o desenvolvimento e utilização de agentes inteligentes em mundos virtuais 3D. Para tanto, inicialmente serão destacados alguns conceitos associados a agentes de software e sua potencialidade em ambientes educacionais.

4.1 Agentes inteligentes em ambientes educacionais

Bogo *apud* Wooldridge (2003) conceitua agente como um programa que auxilia o usuário na realização de alguma tarefa ou atividade. Agentes inteligentes tratam-se de um sistema capaz de tomar decisões e interagir com o ambiente ou outros usuários com base em alguma fonte de dados.

Autores como Russel e Nörvig (1995) descrevem agentes como qualquer coisa capaz de perceber o seu ambiente e interagir pelo intermédio dessas percepções. Agentes inteligentes são, basicamente, componentes de software capazes de expressar características humanas como: pró-atividade, interação em linguagem humana e reatividade aos acontecimentos ao seu redor (TYUGU, 2011).

Segundo Bogo *apud* Wooldridge (2003), todo agente inteligente deve apresentar ao menos quatro características:

1. Autonomia, possuir controle sobre suas ações;
2. Reatividade, perceber alterações ao seu redor;
3. Sociabilidade, interagir com outros agentes utilizando algum tipo de linguagem de comunicação;
4. Pró-atividade, não só reagindo ao ambiente, mas tomando iniciativas quando conveniente.

Boff (2008) *apud* LÉVY (2008) teoriza que é a partir da interação com as coisas que as competências são desenvolvidas, que por meio da relação com a informação é possível adquirir conhecimento. O saber vive na troca de informações com outros, mediante iniciação e transmissão. Ainda, segundo Boff

(2008), o processo cognitivo consiste em três modos complementares: o saber, competência e o conhecimento. Toda a atividade gera um aprendizado; portanto, toda a relação humana implica num aprendizado (BOFF, 2008).

O uso de agentes inteligentes em ambientes de aprendizagem traz uma proposta muito mais ampla que simplesmente realizar algumas tarefas, uma vez que eles propõem a integração entre usuário e o ambiente, entre os usuários e entre usuário e o conhecimento disponível (BOFF, 2008).

Sklar (2006) aponta a existência de três tipos de agentes inteligentes: os *agentes pedagógicos*, os *peer learning agents (agente companheiro)* e os *agentes de demonstração*. O primeiro tipo, o *agente pedagógico*, é considerado uma espécie de tutor no processo de aprendizagem, atuando como um guia. No segundo tipo, *peer learning agents*, tem-se uma proposta de agente como parceiro no processo de aprendizagem. Segundo Sklar (2006), a proposta de *peer learning agents* é muito menos intrusiva que a de *agente pedagógico*. O terceiro tipo, *agente de demonstração*, incorpora uma área do conhecimento, exemplo disso são os simuladores interativos baseados em agentes (BOFF, 2008).

O *agente companheiro* busca simular o comportamento humano, atuando como um parceiro dentro do ambiente ou até mesmo como um adversário. Este agente, ao contrário do *agente pedagógico*, não atua como tutor ou guia, tendo sua atuação por meio da construção de um relacionamento com os estudantes (BOFF, 2008).

O agente companheiro interage com os estudantes com uma linguagem o mais natural possível, sendo que em implementações mais avançadas busca demostrar traços de personalidade e podem, inclusive, expressar certas emoções verbais e visuais como forma de maximizar as interações.

Em Sklar (2006) *apud* Sehaba e Estrailier (2008), é demonstrada a utilização de *agentes companheiros* para a recuperação de crianças com autismo, justamente por se tratarem de agentes com comportamento similar ao dos humanos.

Frozza (2011) traz uma proposta de integração de agente companheiro em seu ambiente virtual de aprendizagem, com o objetivo de ajudar o estudante na solução dos problemas e auxiliar a aprendizagem, realizando o papel de um

“colega virtual”. Neste trabalho, há a implementação de emoções no agente.

O trabalho de Testa (2006) dá uma demonstração do potencial dos agentes inteligentes, utilizando-os para monitorar e auxiliar estudantes que estão trabalhando em grupo, em um ambiente distribuído. Outro exemplo de agente inteligente é o assistente do *Microsoft Office* que utiliza um sistema de perguntas e respostas, no qual o usuário realiza uma pergunta e o agente busca a resposta ou recomenda tutoriais apropriados para o caso.

Guetl et al. (2010) ressalta que a utilização de agentes educacionais pode impulsionar o aprendizado. A partir de seu trabalho, os autores apresentam agentes que interagem com o estudante por meio de uma inteligência social, proporcionando uma potencialização da interação entre estudante e o meio virtual, ampliando assim sua curva de aprendizado.

Considerando os mundos virtuais 3D, os avatares podem ser controlados por humanos que decidirão sobre suas interações, ou os mesmos podem ser manipuláveis por software, atuando como agentes de software. Estes avatares que atuam de forma autônoma são chamados de *NPC’s* - *Nonplayer Character* (GUETL et al*.,* 2010). Na próxima seção será abordada a utilização de avatares inteligentes em ambientes virtuais de aprendizagem 3D.

4.2 Agentes inteligentes em mundos virtuais 3D

Apesar de toda a potencialidade dos ambientes virtuais 3D, ainda é possível identificar algumas fragilidades no que se refere à manutenção da interação e do envolvimento dos estudantes com o ambiente e com os outros avatares (BOFF, 2008; SILVA, 2012; BERNARDI et al*.*, 2013).

Conforme descrito nas seções anteriores, uma possibilidade para minimizar tais dificuldades é a utilização de avatares controlados por software, agentes inteligentes 3D, chamados de *bots,* ou comumente conhecidos como *NPC’s* (PRENDINGER et. al., 2011).

NPC's podem ser definidos como figuras humanoides controladas por software (OPENSIMULATOR, 2014). São, basicamente, agentes conversacionais

autônomos controlados por *scritps* (RICHARDS et. al., 2012), bastante utilizados em jogos, para tornar a experiência de imersão mais real.

Considerando mundos virtuais 3D como o *Second Life* e o *OpenSim*, os mesmos são implementados por meio de *scripts* em *LSL*, podendo ser programados para realizar tarefas pré-determinadas. Estas tarefas pré-determinadas são explicitadas por meio de *notecards*, que são pequenos cartões de notas, contendo as ações as quais eles deverão realizar. O *script LSL* gerará uma representação gráfica do *NPC* e após a tradução dos comandos contidos no *notecard*, formará um roteiro que o *NPC* deverá seguir (OPENSIMULATOR, 2014).

Os *NPC's* podem ser utilizados com diferentes finalidades, podendo ser utilizados para realização de tarefas, orientação de usuários reais (avatares humanos), interação com outros avatares de software ou humanos (VARVELLO et al., 2010), simulação do comportamento humano, integração com outros recursos e até a possibilidade de exprimir certa inteligência, por meio de incorporação de regras de inteligência artificial.

Ao adicionar regras de inteligência artificial nestes agentes, modifica-se o agente, que abandona sua condição de mero executor de instruções pré-programadas, pois suas ações não mais serão uma mera reprodução de um roteiro sem margem para construção de um novo fluxo de interação. A partir desse momento os agentes poderão imitar o comportamento humano (LEIBERMAN; SELKER, 2003). No caso deste trabalho, pretende-se trabalhar com agentes inteligentes do tipo companheiros conversacionais (*chatterbots*), que serão apresentados na próxima subseção.

4.2.1 Chatterbots

Chatterbot é um tipo de agente que simula conversas inteligentes com um ou mais usuários humanos (ORLANDO; GIOVANNI, 2008). Esses agentes podem ser usados para auxiliar os estudantes, ou fornecendo informações sobre o conteúdo pedagógico. Os *chatterbots* são muito populares sendo utilizados para diversos fins (COMARELLA; CAFÉ, 2008). Essa popularidade se deve a sua

capacidade de simular o comportamento humano de maneira tão convincente, que por vezes pode levar um humano a pensar que está realmente conversando com outra pessoa (PILASTRI; BREGA, 2009).

Esses *chatterbots* possuem uma base de conhecimento que é construída por meio de um arquivo AIML (Artificial Intelligence Markup Language), que contém diversas possíveis perguntas e suas respectivas respostas. Para responder a uma interação ele consulta sua base de conhecimento, realiza uma busca por intermédio de palavras-chaves e emite uma resposta de acordo com o que foi encontrado.

Dentre os exemplos de *chatterbots* de maior sucesso pode-se citar o A.L.I.C.E.[6], considerado um dos mais populares *chatterbots*, criado por Dr. Richard Wallace para simular conversas humanas na linguagem natural. Outro exemplo que podemos trazer é o ELIZA[7], que foi o primeiro software para simulação de diálogos, criado no MIT em 1966, por Joseph Weizenbaum. Ainda, pode-se citar o Cleverbot[8], *chatterbot* criado pelo veterano de inteligência artificial Rollo Carpenter com a finalidade de simular conversas humanas, que aprende a partir de conversas com outros humanos. Por fim, destaca-se a Profª. Elektra[9], um *chatterbot* educacional desenvolvido pela Universidade Federal do Rio Grande do Sul (UFRGS) para apoiar o ensino de redes de computadores e o robô ED[10], um *chatterbot* desenvolvido pela Conpet, em parceria com a Petrobras, com o objetivo de conscientizar a sociedade sobre questões ambientais e apresentar os projetos desenvolvidos pela Conpet.

Com base no projeto A.L.I.C.E., devido a sua grande popularidade e aceitação, foi criado o projeto Pandorabots[11], que consiste em um serviço de provedor de um *software* experimental de *bots* baseado no trabalho da comunidade de *software* livre (PANDORABOTS, 2014). Este projeto tem como

[6] http://alice.pandorabots.com/
[7] http://nlp-addiction.com/chatbot/eliza/
[8] http://www.cleverbot.com/
[9] http://penta3.ufrgs.br:2002/
[10] http://www.ed.conpet.gov.br/br/converse.php
[11] http://www.pandorabots.com/botmaster/pt/home

objetivo possibilitar que qualquer um crie seus próprios *chatterbots*, sendo este serviço utilizado no presente projeto.

5 PROPOSTA DO AMBIENTE

Após análise do trabalho desenvolvido por Silva (2012) e dos apontamentos para trabalhos futuros que a autora realizou, percebeu-se a necessidade de implementação de estratégias que envolvessem mais os estudantes nas atividades do mundo, motivando os mesmos e incentivando-os durante as interações com o ambiente. Em suas considerações, Silva (2012) relata que o uso de agentes do tipo *chatterbots* poderia potencializar tais interações, fazendo com que os mesmos atuassem no ambiente auxiliando os estudantes quando o professor não estivesse presente (foco da proposta, por se tratar de um ambiente de apoio ao ensino presencial, que seria utilizado além da sala de aula). O mesmo poderia atuar auxiliando na resolução dos desafios, no monitoramento das atividades e também aumentaria a satisfação e eficiência dos estudantes no contexto de *e-learning*.

Considerando estas colocações, desenvolveu-se a proposta desse trabalho. Pensando em inserir agentes no mundo desenvolvido, como forma de avaliar as expectativas descritas por Silva (2012), foram mescladas duas propostas que resultaram neste trabalho de conclusão de curso. A primeira proposta trata da utilização de agentes de *software* na encenação de uma peça teatral, com o objetivo de discutir a importância da área de testes de software. A escolha por uma peça teatral vai ao encontro das discussões feitas nos capítulos de referencial teórico e nas considerações de Silva (2012) de que estratégias pedagógicas diferenciadas podem motivar mais os estudantes no processo de ensino e aprendizagem. Conforme Koudela e Santana (2005), os educadores terão que se adequar a nova era e reciclarem sua metodologia de ensino, uma maneira disto é por meio da pedagogia do teatro.

A segunda traz a utilização de avatares conversacionais inteligentes que, a partir de uma base de conhecimento, poderão interagir com os estudantes de maneira a simular o comportamento humano, discutindo conceitos relacionados a testes de software.

5.1 Metodologia

A metodologia seguida neste trabalho foi ser dividida em dois momentos: uma sequência de atividades para desenvolvimento da peça teatral e dos agentes envolvidos; e o desenvolvimento dos agentes inteligentes conversacionais (*chatterbots*) que irão interagir com os estudantes.

Para o desenvolvimento da peça teatral, inicialmente foi feita uma análise do ambiente virtual já existente, de forma a redimensionar o espaço físico e projetar a construção a ser desenvolvida para servir de palco para o teatro proposto. Com isso chegou-se a definição de que o espaço físico do teatro estaria situado em uma extensão do terreno da ilha principal, onde se localiza a outra edificação do mundo. Para estipular uma arquitetura para o teatro, foram buscados modelos arquitetônicos que combinassem com a estrutura do prédio principal do mundo.

Na etapa seguinte iniciou-se um estudo para estabelecer a melhor maneira de implementar os atores, então chegou-se a proposta dos NPC's. Na fase de implementação dos NPC's foram utilizados scripts em LSL e suas ações foram programadas por intermédio de *notecards* (OPENSIMULATOR, 2014). A aparência dos NPC's foi construída a partir de scripts geradores de aparência, que capturam a aparência do avatar que os executa. Por último, foi determinado que para execução da peça seria necessário que um avatar humano, com um *click* em um botão criado na entrada do teatro, desse início a peça.

Com relação ao desenvolvimento dos agentes conversacionais (*chatterbots*), após a realização de uma revisão sistemática de literatura chegou-se a uma definição para os mesmos, como agente inteligente conversacional e companheiro. Para seguir com essa implementação foi necessário pesquisar maneiras que possibilitassem a construção dos avatares, então chegou-se ao Pandorabots.

Com a arquitetura e ferramenta definidas, iniciou-se a fase de desenvolvimento da base de conhecimento dos avatares. A mesma foi estruturada com base em uma arquitetura proposta por Benyon (2011), que sugere a criação de 3 (três) modelos: modelo de pessoa, domínio e interação. O *Modelo de Pessoa*

descreve o que o sistema sabe a respeito da pessoa (usuário - neste caso, avatar humano) que interage com o agente, como preferências e dificuldades que foram experienciadas no ambiente. O *Modelo de Domínio* descreve a representação que o agente tem do domínio, ele é necessário para que o sistema possa fazer interferências, possa se adaptar e avaliar essas adaptações. Por fim, tem-se o *Modelo de Interação,* que consiste em duas partes principais, o registro de diálogos, que consiste nos diálogos realizados entre o agente e o avatar humano, e a base de conhecimento (BENYON, 2011).

A próxima etapa é a definição e implementação das bases de conhecimentos identificadas. Criou-se uma base de conhecimento (a) sobre o **domínio**, ou seja, uma base de conhecimento em que o agente será capaz de atuar respondendo sobre o Mundo Virtual de Testes de Software, fornecendo informações sobre o que é o mundo, do que ele é formado e quais são os objetivos educacionais do mesmo; e (b) duas bases de **interação**, uma referente aos diálogos usuais de socialização, tais como cumprimentos, frases motivacionais, etc., e outra, a principal, contendo informações sobre os conteúdos específicos associados a conceitos de testes de software. Não está no escopo deste trabalho o desenvolvimento do Modelo de Pessoa.

Finalmente, com todas as etapas concluídas, o ambiente desenvolvido foi apresentado para um grupo de estudantes, onde realizou-se uma avaliação de usabilidade, para analisar a efetividade dos *NPC's*, dos *chatterbots* e da peça teatral.

6 DESENVOLVIMENTO

Desenvolver uma aplicação imersiva no ambiente virtual 3D Mundo de Testes (SILVA, 2012), para apoiar o ensino presencial do conteúdo de Teste de Software - com esse propósito iniciou-se a construção do presente trabalho. Na primeira etapa foi realizada uma revisão de literatura com o intuito de aprender com as experiências de outros trabalhos no desenvolvimento de aplicações com *NPC's* e *chatterbots* dentro de mundos virtuais 3D.

Com a definição do projeto, iniciou-se o desenvolvimento localmente, primeiro com a instalação da plataforma *OpenSimulator* e a posteriori o carregamento do arquivo .oar do Mundo de Testes de Silva (2012). Inicialmente o mundo contava somente com um terreno principal onde estavam as instalações do prédio da empresa de teste de software onde era executado o jogo JETS, apresentado na seção 2.2 na figura 1. Por se tratar de um novo trabalho, optou-se pela construção de um novo terreno em anexo ao terreno principal, onde construção do teatro foi realizada. A próxima seção destaca seu desenvolvimento.

6.1 Estrutura do Teatro

Com a construção do terreno, deu-se inicio a construção do projeto arquitetônico do teatro. Por meio de uma pesquisa sobre conceitos e arquitetura de teatros, modernos e históricos, optou-se por uma estrutura física que se adicionasse harmoniosamente ao mundo e que seguisse o modelo da estrutura já existente. Depois de selecionado uma arquitetura, a estrutura externa foi moldada a partir de primitivas do próprio ambiente.

Já para o ambiente interno foram utilizados alguns objetos que estão disponíveis no repositório *Zadaroo*[12]. A estrutura interna do teatro foi dividida em três partes, a primeira é uma pequena sala de espera, onde os visitantes podem ficar e interagir antes da peça, no local encontram-se sofás para que os visitantes possam sentar-se, enquanto aguardam o início da peça (Figura 2).

[12] http://zadaroo.com/

Figura 2. Entrada do Teatro.

O segundo espaço é destinado a plateia, onde se encontram distribuídas as poltronas para os espectadores acompanharem a peça (Figura 3).

Figura 3. Espaço destinado à plateia.

Por fim, o terceiro espaço compreende o palco, onde se encontra a

montagem do cenário para a peça, e um *backstage* onde ficam os atores antes do inicio da peça. A montagem do cenário se deu de acordo com as especificações do roteiro original, com um palco onde teriam três mesas e três cadeiras com computadores sobre elas. A única modificação realizada no cenário foi a inclusão de um fundo, por meio de uma primitiva com textura de uma imagem de escritório. Todos os objetos utilizados em cena foram encontrados nos repositórios *Zadaroo* e *GovGrid*[13], como nas Figuras 4 e 5.

Figura 4. Espaço do Cenário.

[13] http://govgrid.org/

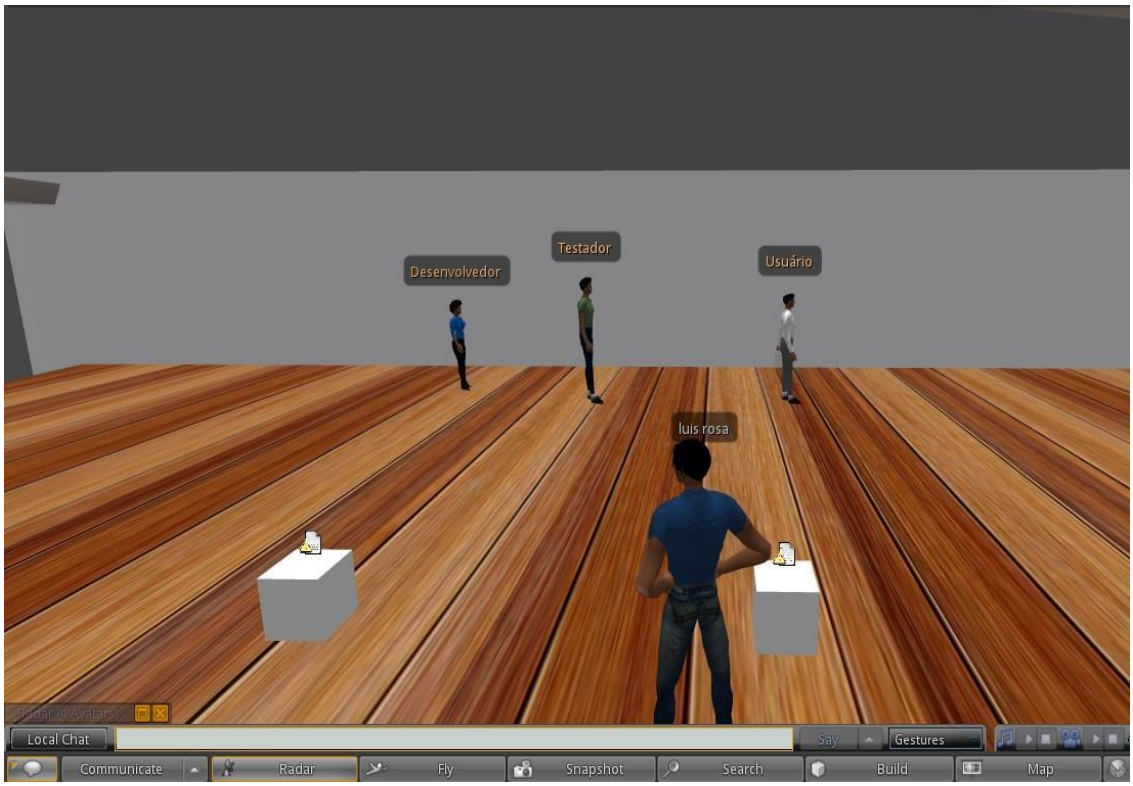

Figura 5. Espaço do Backstage.

Após a definição e construção do espaço físico, passou-se para a etapa de implementação do roteiro da peça, que será detalhada na próxima seção.

6.2 A Peça Teatral - Atores e Roteiro

Com a estrutura do teatro finalizada iniciou-se a adaptação da peça teatral apresentada em Rios e Medeiros (2013), bem como a implementação dos atores por meio de *NPC's*. A peça, intitulada "Testar ou não testar - eis a questão" tem como objetivo central de mostrar a situação de uma empresa na qual os teste eram realizados pelos desenvolvedores e como a figura do testador surge para impedir o fracasso do projeto. Ela é composta por três atores, que representam um usuário, um desenvolvedor e um testador. A peça adaptada pode ser visualizada no Apêndice A.

No processo de adaptação, além de alguns cortes necessários, pois a peça descrita era um tanto longa, foram realizadas alterações de modo a torná-la mais

inclusiva e contrapor estereótipos. Para a montagem da encenação foi projetado que todos os NPC's atores fossem negros e que o Desenvolvedor fosse uma personagem feminina enquanto o Testador e Usuário representariam figuras masculinas.

6.2.1 *NPC's* atores

Para a implementação dos atores da peça teatral foi realizada uma pesquisa detalhada em sites relacionados às plataformas do *Opensim* e do Second Life, tais como na Wiki do *Second Life*[14], na pagina do *OpenSimulator* (OPENSIMULATOR, 2014) e na *free-lsl-script*[15] (site com diversos scripts em LSL), em busca de modelos de implementação de *NPC's* em mundos virtuais. Com essa base, deu-se seguimento a configuração e implementação dos *NPC's*. Para que fosse possível a implementação dos mesmos, foram realizadas alterações no arquivo de configuração do *OpenSim*, o OpenSim.ini , habilitando algumas funções, como por exemplo de permissão para criação de *NPC* (Allow_osNpcCreate = true), bem como modificando níveis de alguma permissões. O arquivo com as modificações realizadas poderá ser visualizado no Apêndice B.

Com essas configurações foi possível prosseguir com os testes para a implementação da peça teatral. Como próximo passo, buscou-se exemplos de scripts em LSL que pudessem ser adequados à necessidade do projeto. Com os scripts apropriados, encontrados na página *free-lsl-scripts*, foram realizadas modificações de maneira a atender as necessidades para a construção do *NPC's* atores.

Para realizar a caracterização dos atores, utilizaram-se scripts geradores de aparência. Esses *scritps* funcionam de maneira simples, uma vez colocados em uma primitiva basta um click na mesma para disparar o evento que irá capturar a UUID, que é um código de registro dentro do *OpenSim* que é atribuído a tudo que está dentro do mundo virtual. Neste caso, será capturada a UUID do avatar humano que clicou na primitiva contendo o *script*, fazendo com que a aparência

[14] http://wiki.secondlife.com/wiki/LSL_Portal
[15] http://www.free-lsl-scripts.com/

deste avatar seja armazenada em uma *notecard*. Nesta nota estarão contidas todas as informações de aparência do NPC, como roupas e aparência física (cor dos olhos, estatura, cor do cabelo, etc.). Por meio deste script, foi gerada a aparência de cada um dos *NPC's* que representam um ator na peça.

Pensando em proporcionar um ambiente mais realístico e imersivo possível, passou-se para a codificação de scripts para acionar a geração dos *NPC's* de forma imperceptível para o espectador. A maioria dos scripts de geração de *NPC's* parte da premissa de que um avatar humano deve, explicitamente, acionar um comando de criação para cada avatar de software. Desta forma, buscou-se adaptar tais scripts de maneira que todos os atores fossem gerados sem que os espectadores percebessem. Para isso, foi construída uma primitiva do tipo cubo, um objeto que representa um botão, que possibilita ao professor ou outro visitante com um *click* iniciar a peça, criando todos os atores automaticamente e sem que necessite informar ou digitar qualquer comando. Dentro dessa primitiva foi colocado um *script* LSL que detecta o evento *click*, no objeto, e envia uma mensagem, por um canal de comunicação, para as três primitivas onde estarão os scripts que implementarão *os NPC's* e suas atuações.

Com as aparências já confeccionadas e com o dispositivo para iniciar a peça já implementada, passou-se a modificação dos scripts de geração dos *NPC's*. Para esse processo, foram criadas três primitivas cubo, que foram posicionadas atrás do plano de fundo do cenário (*backstage*). Com elas já posicionadas, foram inseridos, em cada uma, os scripts de geração dos *NPC's* e as *notecards* da aparência de cada *NPC*. Passou-se então a modificação destes scripts de geração dos *NPC's*, para que fossem executados quando o dispositivo para iniciar a peça fosse acionado. Esse, por meio dos canais de comunicação do *Opensim*, envia a mensagem "GO!" para as primitivas. Para que isso fosse possível foi pego o UUID de cada uma das primitivas, com isso o dispositivo para iniciar a peça consegue acioná-los. A mensagem é endereçada diretamente para a primitiva, que por sua vez possui no seu script a implementação de uma função de escuta, que ficará ouvindo no canal configurado. Essa função de escuta, *listen*, pega a mensagem endereçada a ela e a executa, no caso deste trabalho é a ação GO!, que irá rodar o script dentro da primitiva. O script completo da geração dos

NPC's pode ser visualizado no Apêndice C.

Configurada a inicialização da peça e a implementação do *NPC's*, foi iniciada a codificação do roteiro da peça.

6.2.2 Implementação do roteiro da peça

Nessa etapa, o roteiro adaptado foi configurado para que os *NPC's* os executassem. Para isso foi, necessária à criação de *notecards*, contendo ações a serem executadas pelos *NPC's.* A codificação dessas ações é relativamente simples, para que o *NPC,* por exemplo*,* execute uma ação de fala pelo *chat* de texto, basta adicionar na *notecard* o comando *"@say="* e o texto que deseja-se que ele fale. Graças aos *scripts* de geração dos *NPC's,* também foi possível adicionar outras ações nas suas *notecards*, como executar uma animação, andar, sentar em um determinado objeto, voar, executar um áudio, correr e etc. Utilizando estas possibilidades foi sendo construída a encenação da peça.

Por meio da sincronização das falas, no chat de texto, e as gesticulações, construídas a partir de animações próprias para *OpenSim* construiu-se a peça. Foram também adicionados *scripts* nas cadeiras para que os *NPC's* sentassem corretamente. A fala no *chat* de texto segue a sintaxe já descrita, e apenas mostra na tela o texto que foi configurado. Para que as animações rodem é necessário incluí-las dentro da primitiva. O comando que as executará é o *"@animate="*, que recebe o nome da animação e o seu tempo de execução. Os intervalos entre uma ação e outra são implementados com o comando *"@pause",* que recebe um tempo em segundos e deixa o *NPC* parado pelo tempo determinado.

Ainda, com a proposta de maximizar o sentimento de imersão e tornar a experiência da peça teatral mais realística implementaram-se áudios com cada uma das falas dos atores. Para a gravação dos áudios foi utilizado o programa de gravação *Audacity*[16]*,* indicado pela página do próprio *OpenSim.* Nesta tarefa, contou-se com a contribuição de duas vozes, uma feminina e a outra masculina, sendo que para a terceira voz foram realizadas alterações no tom da voz masculina. As duas primeiras vozes foram atribuídas ao Desenvolvedor e ao

[16] http://downloadsf.com/br/downloads/73187?gclid=CNOuhs27j8ICFbBm7AodohAAmQ

Testador, sendo a terceira voz atribuída ao Usuário. Todas as falas estão no formato *Som Wave*, extensão *".wav",* e seccionados em arquivos de no máximo 10 segundos, requisito exigidos pelo *OpenSim.*

Após a gravação e exportação dos áudios e *upload* deles para dentro do *OpenSim,* teve começo a configuração e sincronia dos áudios, textos do chat e animações. Na reprodução dos áudios, o *script* gerador do *NPC* trabalha com os *UUID* deles, os executando diretamente do inventário, que se trata de uma opção para armazenar e buscar facilmente diversos itens que desejas guardar, como Objetos, *scripts*, *notecards*, itens para personalização do avatar e muito mais. O acionamento ocorre pela função *llTriggerSound,* que recebe o *UUID* do áudio e o volume com que o áudio será executado, como pode ser visto na Figura 6.

Figura 6. Encenação da peça teatral.

Com a inserção e sincronização dos áudios, com as falas do chat e as animações, foi concluída a primeira parte deste trabalho, a implementação da peça teatral, e foi dado seguimento iniciando a implementação do Agente Companheiro Conversacional, o *chatterbot* com objetivo de interagir com estudantes dialogando sobre o ambiente em si e buscando esclarecer dúvidas

sobre conceitos associados a teste de software. A próxima seção destaca seu desenvolvimento.

6.3 O Agente Conversacional Zac

Com o propósito de aprofundar mais o tema trabalhado na peça foi construído um agente do tipo *chatterbot*, classificado como conversacional companheiro. Para a implementação deste agente foi utilizado um *script* para construção do agente e outro que dispara o evento de criação, sendo que cada *script* foi inserido em uma primitiva. O *script* de construção foi adaptado para que a partir do evento *click* (*touch_start*) capture-se o *UUID* do avatar que clicou, para estabelecer a conexão entre o usuário e o agente. No *script* que dispara a criação do agente foi implementada a função que envia o comando de criação para a prim onde está o *script* de construção, por meio da função *llRegionSayTo(),* que envia uma mensagem diretamente para o *UUID* da prim. Foram também modificados o *scripts* originais para que os agentes capturassem o nome do avatar que clicou na prim do *script* de construção. Esta aspatação tem como objetivo fazer com que o agente interaja com o estudante tratando-o pelo nome, de forma a buscar transmitir maior proximidade e empatia com o usuário, como mostra a Figura 7.

Figura 7. Agente Conversacional interagindo com avatar humano.

Para gerar a inteligência do agente, o *script* de construção também foi adaptado para estabelecer a conexão com a base de dados de conhecimento. Para esta etapa do desenvolvimento do projeto foi utilizado o servidor de *chatterbots pandorabots*[17], que trabalha com arquivos *AIML*. Assunto abordado na próxima seção.

6.3.1 *AIML*

A criação da base de conhecimento foi realizada por meio da criação de arquivos *AIML*, que se utiliza de uma estrutura de *tags* contendo o conjunto de perguntas e respostas que compõe tal base. O arquivo é iniciado com a *tag <aiml version=1.0>*; logo em seguida são implementadas as perguntas e respostas da base de conhecimento. Uma interação é implementada com as *tags <category>* para abrir e *</category>* para fechar, sendo que entre elas estará a pergunta e a respectiva resposta, conforme a figura 8.

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

```
<?xml version="1.0" encoding="UTF-8"?>

<aiml version="1.0">

<category>
<pattern>OLÁ</pattern>
<template>
<random>
<li>Oi!</li>
<li>Olá!</li>
<li>Oi, com você esta?</li>
<li>Olá, tudo bem com você?</li>
</random>
</template>
</category>

<category><pattern>ZAC</pattern>
<template><srai>OLÁ</srai></template></category>

<category>
<pattern>OI <bot name="name"/></pattern>
<template><srai>OLÁ</srai></template>
</category>

<category>
<pattern>OLÁ <bot name="name"/>.</pattern>
<template><srai>OLÁ</srai></template>
</category>

<category>
<pattern>ALO *</pattern>
<template><srai>OLÁ</srai></template>
</category>
```

Figura 8. Arquivo AIML exemplo da utilização do category.

Para perguntas é utilizada a *tag <pattern>* e *</pattern>*, sendo que entre estas *tag* é digitado o texto da pergunta. Para as respostas são utilizadas as *tags <template>* e *</template>*, e igualmente à estrutura da pergunta, é digitado o texto de resposta entre elas.

A *AIML* possibilita também que uma pergunta possua vária possíveis respostas, para isso é inserido logo após *template*, a *tag <random>* que seleciona randomicamente uma das respostas disponíveis, que estrão entre as *tags * e *
* como se pode ver na Figura 9.

```
<category>
<pattern>COMO VOCÊ ESTÁ</pattern>
<template>
<random>
<li>Eu estou bem! </li>
<li>Eu estou bem, e você?</li>
<li>Tudo bem comigo!</li>
<li>Estou bem, muito brigado por perguntar.</li>
</random>
</template>
</category>
```

Figura 9. Arquivo AIML exemplo de utilização do random.

Outra possibilidade do *AIML* é a de referenciar outras questões. Por meio da utilização das tags *<srai>* e *</srai>* responder a pergunta com uma resposta implementada para outra pergunta. Para isso teremos que inserir o a *tag* entre as *tags template* e dentro delas colocar a pergunta a qual queres fazer referência, conforme na Figura 10.

```
<category>
<pattern>OLÁ</pattern>
<template>
<random>
<li>Oi!</li>
<li>Olá!</li>
<li>Oi, com você esta?</li>
<li>Olá, tudo bem com você?</li>
</random>
</template>
</category>

<category><pattern>ZAC</pattern>
<template><srai>OLÁ</srai></template></category>

<category>
<pattern>OI <bot name="name"/></pattern>
<template><srai>OLÁ</srai></template>
</category>

<category>
<pattern>OLÁ <bot name="name"/>.</pattern>
<template><srai>OLÁ</srai></template>
</category>
```

Figura 10. Arquivo AIML exemplo de utilização do srai.

O passo seguinte foi a construção da base de conhecimento, que será descrita na sequência.

6.4 Base de conhecimento

Conforme descrito na seção 5, foram criadas duas base de interação e uma de domínio. Na base de interação do tipo *socialização* foram inseridas maneiras de interação corriqueiras, como “olá”, “como está você”, “eu estou bem” e etc. As respostas também passaram pelo mesmo processo, conforme a Figura 11.

Figura 11. Interação entre avatar humano e Agente, por meio da base de interação social.

Na base de conhecimento sobre o *conteúdo específico*, no caso de teste de software, para a construção do conhecimento do agente foram elencados alguns conceitos relevantes, considerando clássicos na literatura de Engenharia de *Software* (SOMMERVILLE, 2012; PRESSMAN, 2006; RIOS e MEDEIROS, 2013), como mostra a Figura 12.

Figura 12. Interação entre avatar humano e Agente, por meio da base de domínio.

A base de *domínio* foi abastecida com informações sobre o ambiente virtual e espaços físicos, tanto do teatro quanto sobre o prédio da empresa. Com a criação e combinação dessas três bases possibilitou aos agentes realiza interações autônomas com os avatares humanos, sem a interferência de outros avatares humanos.

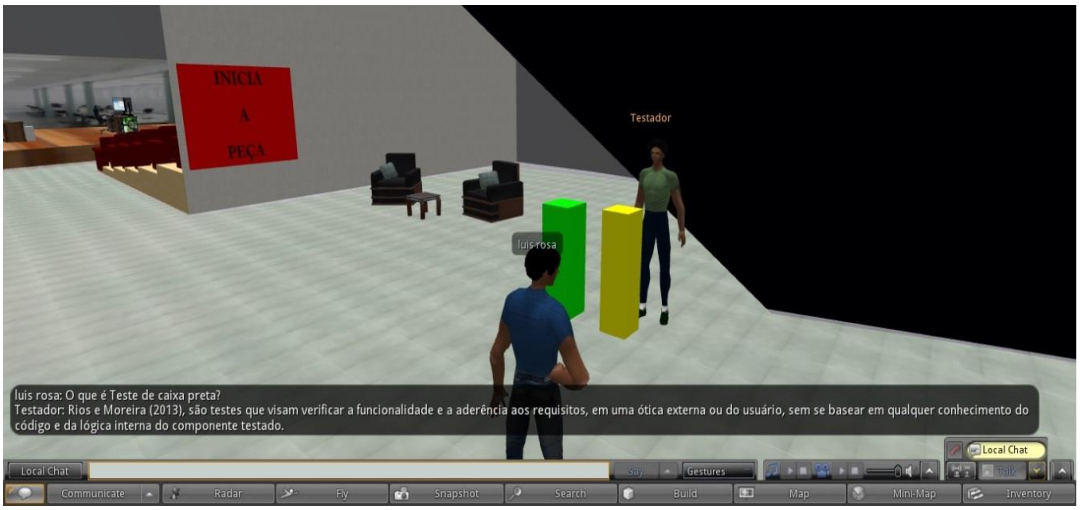

Figura 13. Interação entre avatar humano e Agente, por meio da base de integração sobre teste de software.

Com a criação e combinação dessas três bases possibilitou aos agentes realiza interações autônomas com os avatares humanos, sem a interferência de outros avatares humanos.

7 RESULTADOS E DISCUSSÃO

Com a finalização da modelagem e implementação da peça teatral e dos agentes inteligentes, passou-se para a etapa de avaliação. Baseando-se no método de avaliação de usabilidade, discutido em Barbosa e Silva (2010), foi planejada a avaliação da peça e do agente conversacional (*chatterbot*), no que se refere às interações com os usuários e à sua potencialidade no auxílio ao ensino presencial.

Para a avaliação foram seguidas as etapas sugeridas pelos autores:

- *preparação*, onde ocorre a definição dos participantes da avaliação e atividades que realizarão, preparação de material para observação e registros durante a atividade;
- *coleta de dados*, onde é observada e apanhada as opiniões dos participantes, sendo esta a etapa onde os participantes desenvolvem as atividades propostas, enquanto avaliadores coletam dados;
- *interpretação e consolidação dos dados*, que serve para reunir, contabilizar e sumarizar os dados coletados; e
- *relato dos resultados*, onde é realizado o relato do desempenho e as percepções dos participantes.

O método escolhido para avaliação foi o teste de usabilidade, por melhor se enquadrar ao objetivo da avaliação da peça e do agente, que consistiu em averiguar os pontos fortes e fracos, e o que poderia ser melhorado na interação do usuário com sistema.

Foram definidos como usuários-alvo desta avaliação 20 estudantes do curso de Mestrado em Tecnologias Educacionais em Redes da UFSM. Essa opção deu-se pelo fato dos estudantes deste curso poderem realizar uma análise funcional da peça e agente, por meio do olhar de quem produz aplicações para educação. Tais estudantes tem como sua formação básica, normalmente, cursos da área da educação ou computação, tendo como atuação profissional, em sua maioria, a docência. Ainda, outra justificativa para esta escolha, deve-se ao fato dos estudantes selecionados serem alunos de uma disciplina chamada Ambientes

e Comunidades Virtuais de Ensino e Aprendizagem e estarem, no período da avaliação, iniciando atividades junto ao OpenSimulator.

Para realização dos testes foram definidas duas datas para visitação, planejadas para serem executadas de forma semelhante, de forma a permitir uma avaliação posterior mais padronizada. Buscando simular uma sessão de teatro, foram selecionados dois horários para a exibição da peça, sendo um no final de semana e o outro no período regular de uma disciplina cursada por eles no mestrado. Planejou-se que a visitação seguiria um roteiro, onde primeiro os usuários seriam encaminhados para dentro do teatro para que esperassem o início da peça. Após a sua exibição, seria realizada uma rodada de discussão sobre a peça e sua implementação com o objetivo de captar algumas percepções sobre a mesma. Com o encerramento dessa etapa da atividade iniciaria a segunda parte, a integração com o agente. Nessa fase do teste seria pedido que os usuários se dirigissem até a parte externa do teatro e interagissem com o agente, que estaria a sua disposição.

Por fim seria solicitado que os usuários respondessem a um questionário (Apêndice D), com o objetivo de analisar as percepções de cada um sobre o mundo. Para o questionário foram elencadas algumas questões sobre o perfil dos participantes, tais como faixa etária, formação e experiência em mundos virtuais. Para a análise da peça e do espaço físico do teatro, foram construídas questões buscando coletar as percepções do usuário, buscando elencar potencialidades, melhorias e fraquezas, tanto na peça quanto no espaço físico. Por esse trabalho estar dividido em duas etapas também formuladas questões para a avaliação do agente, levando em conta sua capacidade de interação, conhecimento sobre o conteúdo e aplicabilidade. Terminada a preparação para a visitação, deu-se início a fase de testes.

Conforme já mencionado, a primeira visitação ao mundo desenvolvido foi agendada para um horário fora do horário acadêmico (das aulas do mestrado). Embora ocorrida com um baixo quórum, apenas três voluntários (dos 20 estudantes da disciplina), a atividade discorreu normalmente. Ao chegarem ao mundo os estudantes foram instruídos a entrarem no espaço do teatro e sentarem

em uma das poltronas disponíveis para aguardarem o início da exibição da peça, e que após a exibição seria dadas mais instruções sobre a atividade de interação com o agente. Com o término da exibição foi relatada uma dificuldade com os áudios, pois os mesmos se encontravam com volume baixo, o que os obrigou a acompanhar a peça predominantemente pelo *chat* de texto. Com exceção deste problema, a exibição da peça ocorreu normalmente. Após o término da peça, era planejada uma conversa com os estudantes para obter sua percerpção da peça, contudo devido ao baixo quórum eles foram imediatamente direcionados para o espaço externo do teatro, onde se encontrava o agente (*chatterbot*) e lá foram instruídos sobre como interagir com o mesmo. Dificuldades iniciais foram encontradas devido a maneira como deveriam interagir com o agente, contudo após elucidadas as dúvidas, tudo transcorreu bem.

O segundo encontro foi realizado durante o horário de aula, mas não de forma presencial. Os alunos puderam acessar o mundo virtual de onde desejassem. Devido ao fato da visitação ser em horário de aula, esta contou com a importante participação de doze estudantes, novamente instruídos a se dirigirem até o interior do teatro e aguardarem o início da exibição da peça. Com a chegada do horário combinado foram passadas algumas instruções a respeito da peça e das atividades que seriam realizadas. Após o encerramento da peça foi realizada uma interação com os usuários, a fim de captar algumas de suas percepções sobre a peça.

Cabe destacar que como resultado da primeira experiência realizada, neste intervalo de tempo os problemas com o áudio foram solucionados, funcionando adequadamente nesta segunda visitação. No entanto, durante a segunda exibição da peça, alguns problemas de sincronização foram relatados, em um primeiro momento não foi possível identificar o que causou a falha. Serão necessários mais testes para que se possa analisar de forma precisa o que pode ter acontecido.

Após esse momento no teatro passou-se a segunda parte da atividade, a interação com o agente. Neste momento houve problemas de comunicação, pois os estudantes foram instruídos a dirigirem-se até o exterior do teatro e interagirem com o avatar um de cada vez para que não houvesse sobreposição de falas.

Entretanto, a maioria tentou interagir com o agente ao mesmo tempo, causando confusão. Neste momento, foi reiniciada a atividade com todas as instruções sendo passadas de maneira correta e todas as dúvidas foram sanadas. Analisando as interações, alguns problemas com a base de conhecimento foram encontrados. Algumas questões, devido à maneira com foram formuladas, não foram respondidas corretamente, contudo grande parte dos questionamentos foram sanados.

Com o término da atividade, tanto na primeira visitação quanto na segunda, foi solicitado que os voluntários respondessem ao questionário com suas percepções sobre o teatro, peça e agente, de forma a complementar a avaliação. Com isto, foram cruzados os dados observados e relatados durante a atividade e as avaliações feitas respondidas pelos estudantes.

O perfil dos estudantes compreendem pessoas de ambos os sexos, com mais de 25 anos na grande maioria, oriundos das mais diversas áreas de formação e já com alguma experiência em ambientes virtuais 3D.

Com base nessas avaliações e percepções realizadas, pode se constatar que a peça apresenta grande potencial para ser utilizada com apoio ao ensino de teste de *software*, pois, segundo relatos, o conteúdo foi apresentado de forma compreensível e atrativo. Alguns relatos ainda apontaram que talvez interações com a plateia tornariam a peça ainda mais interativa, causando maior envolvimento por parte do usuário.

Quanto aos agentes, a avaliação e o questionário contribuíram significativamente para responder alguns questionamentos deste trabalho. Os relatos mostram que este tipo de interação pode ser facilmente utilizado como um apoio as aulas e como uma qualificada e interessante maneira de trabalhar os conteúdos. Na avaliação dos voluntários, o agente foi considerado uma boa estratégia de apoio ao ensino presencial, pois interagiu de forma bastante satisfatória e que, com poucas alterações, haveria grande similaridade à interação entre humanos. Com essa avaliação também foi possível mapear outras formas de questionamento para as resposta da base de teste de softwares, e com isso enriquecer mais a base de conhecimento dos agentes tornando-a mais robusta e

assim atendendo mais adequadamente às interações.

8 CONCLUSÕES

Para modificar o paradigma de aprendizado, iniciativas como uma peça teatral pode ser uma alternativa (KOUDELA E SANTANA, 2005). A peça teatral presente neste trabalho realiza um *link* entre o teatro, mundo real e o mundo virtual, proporcionando uma interação bastante realística, buscando uma maior sensação de imersão do usuário no mundo criado.

A construção da encenação da peça no *OpenSim* pode proporcionar uma nova experiência em mundos virtuais. Como a proposta de organizar sessões de exibição da peça, como no mundo real, pretende-se causar no visitante uma sensação de total imersão no mundo, trazendo assim uma interação capaz de promover a construção do conhecimento de maneira mais atrativa e diversificada.

A proposta da utilização de *NPC's* na peça busca trazer a experiência de interação com agentes de software. Os *NPC's* trazem a proposta de diversificar e intensificar a experiência em mundos virtuais. Eles podem imitar interações humanas como expressar raiva, alegria, satisfação e outras emoções. Com eles é possível, simplesmente modificando as *notecards*, construir uma nova peça.

Ainda como componente deste trabalho, os agentes conversacionais companheiros proporcionam aprendizado por meio da simulação de diálogo com outro avatar, impulsionando a construção do conhecimento por meio dessa troca de ideias. Com esses agentes é possível que o visitante possa aprender sozinho, sem a interação com um professor ou um guia humano. Essa simulação de comportamento humano pode solucionar o problema de abandono ou solidão, que é relatado em alguns trabalhos.

A proposta é realizar a avaliação inicial com um grupo de voluntários da pós-graduação do Mestrado Profissional em Tecnologias Educacionais em Rede de forma a verificar a usabilidade do mundo de testes, peça teatral e agente conversacional. Espera-se, com esta avaliação inicial, analisar potencialidades e fragilidades da proposta desenvolvida para, futuramente, avaliar novamente o mundo criado com alunos de disciplinas específicas da área de teste de software,

visando, além da avaliação de usabilidade do ambiente, avaliar o potencial pedagógico da abordagem proposta.

9 REFERÊNCIAS

ABNT (2014). Disponível em: <http://www.abntcatalogo.com.br/norma.aspx?ID=002815>. Acesso em 10 de outubro de 2014.

ACTIVE WORLDS (2014). Disponível em: <https://www.activeworlds.com/> . Acesso em 6 de setembro de 2014.

ASTIGARRAGA, T. et al. (2010). **The emerging role of software testing in curricula.** In: TRANSFORMING ENGINEERING EDUCATION: CREATING INTERDISCIPLINARY SKILLS FOR COMPLEX GLOBAL ENVIRONMENTS, 1., 2010, Dublin. Anais. Dublin: IEEE.

AZEVEDO, C. F. de; ELIA, M. da F. (2011). **Proposta de uma Aplicação de Mundos Virtuais na Educação usando o OpenSimulator com diferentes requisitos tecnológicos**. Anais do XXII SBIE - XVII WIE.

BAINBRIDGE, W. S. (editor) (2010). **Online Worlds: Convergence of the Real and the Virtual**. London: Springer.

BARBOSA, S. D. J.; SILVA, B. S. (2010) **Interação Humano-Computador**. Editora Campus-Elsevier, 2010.

BEHAR , P. A.; BERCHT, M.; LONGHI , M. T. (2009) **Integração do Humor do Aluno Ambiente Virtual de Aprendizagem RODA. Porto Alegre, RS, Brasil.**

BENYON, D. (2011). **Interface Humano-Computador.** Brasil : Pearson, 2ª ed.

BERNARDI, G., CORDENONSI, A. Z., MÜLLER, F. M., DA SILVA, T. G., BOS, A. S., & FLECK, R. (2013, October). **Aprendizagem Colaborativa em Mundos Virtuais 3D: analisando a colaboração sob a perspectiva do Modelo 3C de**

Colaboração. In: Proceedings of Brazilian Symposium on Collaborative Systems (p. 96). Sociedade Brasileira de Computação.

BOFF, E. (2008). **Colaboração em Ambientes Inteligentes de Aprendizagem mediada por um Agente Social Probálistico.** Tese (Doutorado em Ciência da Computação) - PPGC, UFRGS. Porto Alegre.

BOGO, L. H. (2003). C**riação de Comunidades Virtuais a partir de Agentes Inteligentes: Uma aplicação em E-learning**. Dissertação (Mestrado em Engenharia de Produção) – PPGEP, UFSC. Florianópolis.

BRITO, L. M. DE; GOMES, S. G. S.; MOTA, J. B. (2013) **Ambientes Virtuais De Aprendizagem Como Ferramentas De Apoio Em Cursos Presenciais E A Distância.** RENOTE, Revista Novas Tecnologias na Educação. V. 11 Nº 1, julho, 2013.

CARMO, F. M. do (2013). **Mundo virtual 3D em plataforma aberta como interface para ambientes de aprendizagem.** Dissertação de Mestrado na Escola Politécnica da Universidade de São Paulo. São Paulo.

CHEN, B. et al. VCUHK. (2010). **integrating the real into a 3D campus in networked virtual worlds.** In: INTERNATIONAL CONFERENCE ON CYBERWORLDS, 10., 2010, Singapura. Anais. Singapura: IEEE, 2010.

COMARELLA, R. L.; CAFÉ, L. M. A. (2008). **CHATTERBOT: conceito, características, tipologia e construção.** Informações & Sociedade. V. 18 Nº 2, maio/agosto 2008.

COX, R. J. e CROWTHER, P. S. (2009). **A Review of Linden Scripting Language and Its Role in Second Life. Lecture Notes in Computer Science** Volume 5322, pp 35-47. 2009.

DALSASSO, P. (2013). **Museu Virtual Da História Da Computação: Uma Abordagem De Apoio Ao Processo De Ensino-Aprendizagem De Introdução À Computação.** Trabalho de Graduação (Curso de Ciência da Computação) - UFSM. Santa Maria.

DAMER, B. (1997). **Avatars!; Exploring and Building Virtual Worlds on the Internet.** Peachpit Press Berkeley, CA, USA. 1997

FROZZA, R. ; KÜNZEL, M. F.;SILVA, A. K. da; LUX, B.; BAGATINI, D.. (2011). ***Dimi 3d - Companion Agent In A Learning Virtual Environment***. In: 2011 Workshop and School of Agent Systems, their Environment and Applications.

GUETL, C.; SOLIMAN, M. (2010). **Intelligent Pedagogical Agents in Immersive Virtual Learning Environments: A Review.**

HODGE, E.; COLLINS, S.; GIORDANO, T. (2011). **The virtual worldsh:** how to use Second Life and other 3D virtual environments. Sudbury: Jones and Bartlett.

KANIJ, T.; MERKEL, R.; GRUNDY, J. (2011) **A preliminary study on factors affecting software testing team performance.** In: INTERNATIONAL SYMPOSIUM ON EMPIRICAL SOFTWARE ENGINEERING AND MEASUREMENT, 5., 2011, Banff. Anais… Banff, 2011.

KOUDELA, I. D.; SANTANA, A. P. de (2005) **Abordagens metodológicas do teatro.** Ciências Humanas em Revista, São Luís, V. 3, n.2, dezembro 2005.

LINDEN LAB: MAKERS OF SHARED CREATIVE SPACES. (2014). Disponível em: <http://lindenlab.com/>. Acessado em 6 setembro.

MACHADO, A. (2002). Regimes de Imersão e Modos de Agenciamento. **INTERCOM – Sociedade Brasileira de Estudos Interdisciplinares da Comunicação XXV Congresso Brasileiro de Ciências da Comunicação**. Salvador/BA.

MATTAR, J.; VALENTE, C. (2007). **Second Life e Web 2.0 na educação: o potencial revolucionário das novas tecnologias.** São Paulo: Novatec.

MYERS, G. J. et al. (2004). **The art of software testing.** New Jersey: John Wiley & Sons, 2.ed.

NUNES, M. A. S. N.; BEZERRA, J. S.; Reinert, D.; Moraes, D.; Silva, É. P.; Pereira, A. J. S. (2010) **Computação Afetiva e sua influência na personalização de Ambientes Educacionais: gerando equipes compatíveis para uso em AVAs na EaD.**

NUNES, M. A. S. N. (2012). **Computação Afetiva personalizando interfaces, interações e recomendações de produtos, serviços e pessoas em ambientes computacionais.**

OPENSIMULATOR. (2014) Disponível em: <www.opensimulator.org>. Acessado em 6 de setembro.

ORLANDO, P; GIOVANNI, F. (2008) .**An integrated system, with natural language management, for the monitoring activities in e-learning environments.** In: INTERNATIONAL CONFERENCE ON COMPLEX, INTELLIGENT AND SOFTWARE INTENSIVE SYSTEMS, 2., 2008, Catalonia. Anais… Catalonia: IEEE, 2008.

OSÓRIO, F. S.; MUSSE, S. R.; SANTOS, C. T. dos; HEINEN, F; BRAUN, A; SILVA, André T. da. (2004). **Ambientes Virtuais Interativos e Inteligentes: Fundamentos, Implementação e Aplicações Práticas.** XXIV Congresso da SBC – JAI 2004 (Jornadas de Atualização em Informática). Tutorial. Salvador, Bahia.

PANDORABOTS. (2014) Disponível em:< http://www.pandorabots.com >. Acessado em 13 de julho.

PICARD, R. W. (1997). **Affective Computing.** MIT Press, Cambridge, MA, USA.

PILASTRI, A. L.; BREGA, J. R. F. (2009). **Chatterbot com Interatividade ao Avatar Encapsulado no Ambiente Virtual Second Life usando a base de conhecimento em AIML.** Workshop de Realidade Virtual E Aumentada (WRVA).

PRENDINGER, H.; ULLRICH, S.; NAKASONE, A.; ISHIZUKA, M. (2011). **MPML3D: Scripting Agents for the 3D Internet**.

PRESSMAN, R. S (2006). **Engenharia de software.** São Paulo: McGraw-Hill, 6.ed. 2006.

PRESSMAN, R. S. (2011). **Engenharia de software: uma abordagem profissional.** Porto Alegre: AMGH, 7.ed. 2011.

PRIKLADNICKI, R. et al. (2009). **Ensino de engenharia de software: desafios, estratégias de ensino e lições aprendidas.** In: FÓRUM DE EDUCAÇÃO EM ENGENHARIA DE SOFTWARE, 2., 2009, Fortaleza. Anais eletrônicos... Fortaleza: UFC, 2009.

RICHARDS, D.; SZILAS N. (2012). **Challenging Reality using Techniques from Interactive Drama to Support Social Simulations in Virtual Worlds.**

RIOS, E.; MOREIRA, T. (2013). **Teste de Software.** 3ª edição. Rio de Janeiro: Alta Books, 2013. 293 p.

RITZEMA, T.; HARRIS, B. (2008). **The use of Second Life for distance education.** Journal of Computing Sciences in Colleges, v.23, n.6, jun. 2008.

ROCHA, E. M. ; ELIAS, L. O.; OBARA, M. K. ; HASEGAWA, M. Y. ; NETO, L. A. ; ABE, F. H. N. (2012). **Da implantação à implementação de ambientes virtuais tridimensionais: caminhos em construção.**

SAVIN-BADEN, M. A. (2010). **practical guide to using Second Life in higher education.** New York: McGraw-Hill, 2010.

SECONDLIFE (2014). Dísponivel em: <www.secondlife.com>, Acessado em 6 de setembro.

SILVA, T. G. (2012). **Jogos Sérios em Mundos Virtuais: Abordagem para O Ensino-aprendizagem de Teste de Software.** Dissertação (Mestrado em Ciência da Computação) - PPGI, UFSM. Santa Maria.

SILVA, T. S.; MÜLLER, F. M.; BERNARDI, G. (2011). **Panorama do ensino de engenharia de software em cursos de graduação focado em teste de software: uma proposta de aprendizagem baseada em jogos.** Renote, v. 9, n. 2, 2011.

SKLAR, E.; RICHARDS, D. (2006). **The Use of Agents in Human Leaarning Systems**. New York.

SKLAR, E. (2003). **Agents for Education: When too Much Intellig ence is a Bad Thing**. New York.

SCHMITT, M. A. R. e TAROUCO, L. M. R. (2008). **Metaversos e laboratórios virtuais - possibilidades e dificuldades.** Anais da Renote - XII Ciclo de Palestras Novas Tecnologias na Educação. UFRGS. 2008.

SOMMERVILLE, I. (2007). **Engenharia de software.** São Paulo: Pearson Addison Wesley, 8.ed. 2007.

SOMMERVILLE, I. (2012). **Engenharia de software.** São Paulo: Pearson Addison Wesley, 9.ed. 2012.

STROUSTRUP, B. (2010). **What should we teach new software developers? Why?**. Communications of the ACM, v.53, n.1, jan. 2010.

TEICHRIEB, V. (1999). **Avatares como Guias Interativos para Auxílio na Navegação em Ambientes Virtuais Tridimensionais**. Dissertação (Mestrado em Ciência da Computação) – PGCC , UFPE. Recife.

TESTA, C. D. ; DE MARCHI, A. C. B. ; DA SILVA, F. B. (2006). **Uma Proposta de Uso de Agentes Inteligentes para auxiliar a Criação de Sub-comunidades de Aprendizagem na CVMuzar**.

TYUGU, E. (2011). **Artificial Intelligence in Cyber Defense.** In: Cyber Conflict (ICCC), 3rd International Conference on Cyber Conflict, p. 1-11.

VARVELLO, M.; VOELKER, G. M. (2010). **Second Life: a Social Network of Humans and Bots**.

VINCENTI, G.; BRAMAN, J. (2011). **Multi-user virtual environments for the classroom: practical approaches to teaching in virtual worlds.** Hershey: Information Science Reference, 2011.

W3C. (2010). **Emotion Markup Language (EmotionML) 1.0 W3C** -Working Draft 29 October 2009. Disponível em: http://www.w3.org/TR/2009/WD-emotionml-20091029/ .Acesso em 08 de agosto de 2014.)

WANGENHEIM, C. G.; SILVA, D. A. (2009). **Qual conhecimento de engenharia de software é importante para um profissional de software?** In: FÓRUM DE EDUCAÇÃO EM ENGENHARIA DE SOFTWARE, 2.,2009, Fortaleza. Anais eletrônicos... Fortaleza: UFC.
Disponível em: <http://fees.inf.puc-rio.br/FEESArtigos/ FEES09/ >. Acesso em: 2 setembro

WANKEL, C.; KINGSLEY, J. H. (2009). **Education in Virtual Worlds**. Bingley: Emerald.

10 APÊNDICES

APÊNDICE A - Roteiro adaptado da Peça

Testar ou não testar - Eis a questão

Trata-se de uma peça teatral com três personagens na qual as melhores práticas da atividade de testar uma aplicação ou software são discutidas. Entende-se que entre as melhores práticas estejam os usos de uma metodologia aderente ao processo de desenvolvimento, pessoal técnico capacitado e ferramentas de automação adequadas. A peça procura mostra a evolução de uma empresa onde os testes eram executados pelos próprios responsáveis pelo desenvolvimento das aplicações até um outro momento em que a figura de um analista de teste surge para impedir o fracasso de um projeto muito importante.

Para fins de simplificações, foram considerados três personagens representativos de todo o processo, aos quais foram dados os nomes de desenvolvedor (responsável pela liderança no projeto de desenvolvimento de qualidade da aplicação), testador, (o responsável pelo teste e garantia de qualidade da aplicação) e o usuário (quem faz o pedido para o desenvolvimento de uma aplicação).

Personagens

Desenvolvedor - Analista de Sistemas responsável pela liderança do processo de desenvolvimento de uma aplicação solicitada pelo usuário.

Testador - Analista de Testes responsável pela liderança do processo de testar uma determinada aplicação. Garante que o sistema faz o que deveria fazer e não o que não deveria fazer.

Usuário - Técnico da área de negócios responsável pelo pedido de uma nova aplicação à área de TI e pelo fornecimento de informações necessárias a seu desenvolvimento, basicamente formalizado no requerimento entregue ao desenvolvedor.

Cenário - No palco, estão colocadas três mesas, cada uma com um computador, significando cada uma das áreas envolvidas no trabalho de desenvolver, testar e usar a aplicação. Na mesa da extrema direita, está o usuário. Na mesa da extrema esquerda, está o desenvolvedor. Existe uma mesa no meio das duas, temporariamente vazia, mas que será usada pelo testador que aparecerá no decorrer da peça.

O inicio da peça

Tela: (1)Testar ou não testar, eis a questão. - Peça tecnológica de Emerson Rios (2) Para tornar a peça mais agradável criamos alguns estereótipos que não podem ser generalizados. Evidentemente que nem todos os desenvolvedores agem da forma mostrada nesta peça e nem todos os testadores são a salvação do projeto. Muitas vezes, ocorre até o inverso. Trata-se de uma liberdade artística que usamos para que a peça pudesse ser mostrada de forma didática. (3) Pedimos desculpas antecipadas pelos excessos, mas se tivéssemos feito tudo certo não haveria graça. (4) No início, a empresa não usava testadores para validar a aplicação. Veja como o usuário estabeleceu um prazo e o desenvolvedor, para agradar, aceitou.

Usuário: (com uma folha de papel na mão) Meu caro desenvolvedor, estou aqui com um documento com os meus requisitos detalhados descrevendo um sistema. Precisarei que o mesmo fique pronto em 2 meses. (Olhando melhor o papel) Aliás, acho que 1 mês e meio seria um prazo mais apropriado.

Desenvolvedor: (Analisando o papel) Tudo bem. Nós estamos acostumados a fazer este tipo de trabalho, temos algumas ferramentas de desenvolvimento, o que nos permite ter uma alta produtividade e eu posso garantir que não teremos nenhuma dificuldade em atender ao prazo solicitado. Existem algumas técnicas de desenvolvimento ágil de sistemas que serão úteis e darão rapidez ao nosso projeto.

Usuário: Desenvolvimento ágil? O que é isso?

Desenvolvedor: É a resposta às suas necessidades. Sistemas entregues em prazos menores. De qualquer forma, não cabe a você discutir o tipo de técnica que usaremos. Os prazos serão atendidos e isto é o que basta.

Usuário: (Surpreso) Mas você nem analisou em detalhes os meus requisitos!

Desenvolvedor: Não há necessidade de perdermos tempo, pois os prazos são curtos. Nós já conhecemos o seu negócio e sabemos exatamente o que você quer. Não é o primeiro sistema que fazemos e nem será o último. Relaxe e fique calmo, pois no final tudo terminará bem.

Desenvolvedor: O sistema está pronto!

Usuário: (olhando aterrorizado) Mas não foi nada disso que eu pedi! Você fez tudo errado! Você não leu direito o requerimento nem olhou a parte de trás da folha, onde está uma lista do que não deveria ser feito.

Usuário: Além de não cumprir o prazo ainda faz tudo errado. Não dá para confiar neste pessoal da área de TI. Eles fazem o que bem entendem e não verificam direito o que fizeram.

Desenvolvedor: Testar aplicações é muito chato. Vou dar uma rápida verificada para ver se está tudo funcionando, pois eu já testei a maior parte da outra vez.

Usuário: Vou avisar ao diretor que os prazos não vão ser cumpridos por causa daquele pessoal da área de TI. Olha que eu já tinha dado um prazo longo e colocado uma margem de segurança de vários dias. Coitado do diretor. Ele tinha alguns compromissos firmados.

Desenvolvedor: (Voltando lentamente para falar com o usuário) Agora, o sistema foi desenvolvido como você pediu. Desculpe o atraso e os problemas, mas agora está como foi definido nos requisitos que você fez. Inclusive com a parte de trás da folha.

Usuário: Testou tudo? Eu não posso perder o meu precioso tempo com erros idiotas.

Desenvolvedor: Bem, eu dei uma testada rápida. Você sabe, o tempo é curto e os prazos apertados. A pressão também é muito grande.

Usuário: Deixe-me dar uma olhada. Epa! Um erro idiota! Não falei! (Uma correria começa. O desenvolvedor fica indo da mesa do usuário à sua mesa e vice-versa. O usuário fica gritando "erro idiota" toda vez que ele volta para a sua mesa).

Usuário: Chega! Assim não dá! Eu deveria fazer apenas um teste de aceitação do sistema e não ficar descobrindo erros idiotas. Na empresa onde minha esposa/meu marido trabalha tem uns testadores de sistemas que fazem esse trabalho de eliminar os erros idiotas. Eles garantem a qualidade dos sistemas. Eles evitam que os usuários percam o seu precioso tempo.

Desenvolvedor: Aqui está o nosso testador. Ele vai testar o nosso sistema. Tudo vai funcionar como na empresa da/do sua/seu esposa/marido.

Usuário: (Já demonstrando cansaço) Todos os prazos estão se esgotando. O diretor está muito nervoso. Espero que pelo menos a qualidade melhore.

(O desenvolvedor vai para a sua mesa e o testador ocupa a mesa do centro. Trabalha um pouco e se dirige para a mesa do testador).

Desenvolvedor: Pode testar. Capricha no seu trabalho, pois o usuário está reclamando.

Testador: Já!? Eu ainda nem sei o que o sistema deve fazer!

Desenvolvedor: Você foi contratado porque o usuário pediu. O seu trabalho é testar. Vamos! Testa logo que estamos todos na espera pelo seu trabalho.

Testador: Ih! Tem um erro aqui!

Desenvolvedor: Você está usando uma versão antiga do programa. Use a última versão.

Testador: Ninguém falou nada comigo!

Desenvolvedor: E nem precisa falar nada! Chega! Você já trabalhou muito. Já deve estar tudo testado. Vamos mostrar ao usuário.

(O testador se levanta relutante. Os dois vão conversar com o usuário)

Testador: (Dirige-se à plateia) Eles não deixaram eu fazer o meu trabalho.

Usuário: Agora parece que está tudo bem. Eu não falei que vocês precisavam de alguém para testar as bobagens que vocês mesmos fazem? (Riso irônico) Vou avisar ao diretor. Ele vai ficar contente.

(Cada um senta na sua mesa. Passam-se alguns minutos. O usuário dá um grito).

Usuário: Um erro! Logo agora que já liberamos o acesso pela Internet. Vamos ter que tirar o site do ar! Os negócios serão interrompidos. Desenvolvedor, venha cá!

Desenvolvedor: (Levanta apressado da sua mesa) O que foi usuário? Está tudo testado. O sistema está fazendo tudo que você pediu.

Usuário: E passou também a fazer coisas que eu não pedi. Olhe aqui!

Desenvolvedor: A culpa é do testador!

Testador: Você não deixou eu fazer o meu trabalho.

Desenvolvedor: Você teve tempo de sobra para testar o sistema.Todos aqui estão de prova (aponta para a plateia).

Usuário: Chega! Corrija este erro! O sistema está fora do ar! O diretor vai me esganar.

(O desenvolvedor vai para a sua mesa e começa a trabalhar. O testador senta na sua mesa e começa a ler uma folha de papel. O usuário está muito nervoso).

Tela: Horas e talvez dias passam.

Desenvolvedor: (Dirigindo-se ao testador) Pronto! Pode testar.

(O testador ignora o que o desenvolvedor está falando e continua a trabalhar).

Desenvolvedor: Você vai testar o sistema ou vai continuar lendo estas bobagens? O tempo está passando. O usuário está nervoso.

Testador: Pode gritar à vontade. Pode até rolar pelo chão. Eu agora vou fazer o meu trabalho direito. Já preparei um plano de teste. Olha só o requisito que o usuário fez. Tem um erro de lógica.

Desenvolvedor: Oba! Agora podemos colocar a culpa no usuário. Enquanto você faz esta porcaria de plano, eu vou esfregar na cara do usuário este requisito errado.

Testador: (Dirigindo-se para os dois que ainda discutem) Acabei! Agora eu fiz o trabalho correto. Plano de Testes, Estratégia de Testes, Caso de Testes, Script de Testes...

Desenvolvedor: Chega! O sistema está ou não está testado?

Testador: Eu fiz também o seu trabalho. Fiz um teste unitário e um teste de integração, que na verdade caberia a você fazer. Encontrei alguns erros e já providenciei a correção com o pessoal da sua equipe. O nível de qualidade agora está bom. Pode liberar para a produção.

Desenvolvedor: O sistema acaba de receber o selo de qualidade pelo nosso pessoal de teste. Pode avisar ao diretor.

Usuário: Agora é tarde. O diretor já perdeu o emprego.

Desenvolvedor: Que pena! Será que foi por causa de uns errinhos bobos de sistema? Quem irá substituí-lo?

Usuário: Eu (olhando com sarcasmo para o desenvolvedor).

APÊNDICE B – Arquivo OpenSim.ini

```
;; This is the main configuration file for OpenSimulator.
;; If it's named OpenSim.ini then it will be loaded by OpenSimulator.
;; If it's named OpenSim.ini.example then you will need to copy it to
;; OpenSim.ini first (if that file does not already exist)
;;
;; If you are copying, then once you have copied OpenSim.ini.example to
;; OpenSim.ini you will need to pick an architecture in the [Architecture]
;; section at the end of this file.
;;
;; The settings in this file are in the form "<key> = <value>".  For example,
;; save_crashes = false in the [Startup] section below.
;;
;; All settings are initially commented out and the default value used, as
;; found in OpenSimDefaults.ini.  To change a setting, first uncomment it by
;; deleting the initial semicolon (;) and then change the value.  This will
;; override the value in OpenSimDefaults.ini
;;
;; If you want to find out what configuration OpenSimulator has finished with
;; once all the configuration files are loaded then type "config show" on the
;; region console command line.
;;
;;
;; NOTES FOR DEVELOPERS REGARDING THE FORMAT OF THIS FILE
;;
;; All leading white space is ignored, but preserved.
;;
;; Double semicolons denote a text comment
;;
;; ;# denotes a configuration directive description
```

```
;;    formatted as:
;;    {option} {depends on} {question to ask} {choices} default value
;;    Any text comments following the declaration, up to the next blank line.
;;    will be copied to the generated file (NOTE: generation is not yet
;;    implemented)
;;
;;    A * in the choices list will allow an empty entry.
;;    An empty question will set the default if the dependencies are
;;    satisfied.
;;
;; ;  denotes a commented out option.
;;    Any options added to OpenSim.ini.example should be initially commented
;;    out.


[Startup]
    ;# {ConsolePrompt} {} {ConsolePrompt} {} "Region (\R) "
    ;; Console prompt
    ;; Certain special characters can be used to customize the prompt
    ;; Currently, these are
    ;; \R - substitute region name
    ;; \\ - substitute \
    ; ConsolePrompt = "Region (\R) "

    ;# {save_crashes} {} {Save crashes to disk?} {true false} false
    ;; Set this to true if you want to log crashes to disk
    ;; this can be useful when submitting bug reports.
    ;; However, this will only log crashes within OpenSimulator that cause the
    ;; entire program to exit
    ;; It will not log crashes caused by virtual machine failures, which
    ;; includes mono and ODE failures.
```

```
    ;; You will need to capture these native stack traces by recording the
    ;; session log itself.
    ; save_crashes = false

    ;# {crash_dir} {save_crashes:true} {Directory to save crashes to?} {} crashes
    ;; Directory to save crashes to if above is enabled
    ;; (default is /opensimdir/crashes/*.txt or C:\opensim\crashes\*.txt)
    ; crash_dir = "crashes"

    ;# {PIDFile} {} {Path to PID file?} {}
    ;; Place to create a PID file
    PIDFile = "/tmp/my.pid"

    ;# {region_info_source} {} {Where to load region from?} {filesystem web}
filesystem
    ;; Determine where OpenSimulator looks for the files which tell it
    ;; which regions to server
    ;; Default is "filesystem"
    ; region_info_source = "filesystem"
    ; region_info_source = "web"

    ;# {regionload_regionsdir} {region_info_source} {Location of file?} {} Regions
    ;; Determines where the region XML files are stored if you are loading
    ;; these from the filesystem.
    ;; Defaults to bin/Regions in your OpenSimulator installation directory
    ; regionload_regionsdir="C:\somewhere\xmlfiles\"

    ;# {regionload_webserver_url} {region_info_source} {URL to load region from?}
{}
    ;; Determines the page from which regions xml is retrieved if you are
    ;; loading these from the web.
```

```
    ;; The XML here has the same format as it does on the filesystem
    ;; (including the <Root> tag), except that everything is also enclosed
    ;; in a <Regions> tag.
    ; regionload_webserver_url = "http://example.com/regions.xml";

    ;# {allow_regionless} {} {Allow simulator to start up with no regions configured.}
{true false} false
    ;; Allow the simulator to start up if there are no region configuration available
    ;; from the selected region_info_source.
    ; allow_regionless = false

    ;# {MaxPrimUndos} {} {Maximum number of undos avialable for position, rotation
and scale changes of each prim} {} 20
    ;; Increasing the number of undos available number will increase memory usage.
    MaxPrimUndos = 20

    ;# {NonPhysicalPrimMin} {} {Minimum size of nonphysical prims?} {} 0.001
    ;; Minimum size for non-physical prims.  Affects resizing of existing
    ;; prims.  This can be overriden in the region config file (as
    ;; NonPhysicalPrimMin!).
    ; NonPhysicalPrimMin = 0.001

    ;# {NonPhysicalPrimMax} {} {Maximum size of nonphysical prims?} {} 256
    ;; Maximum size for non-physical prims.  Affects resizing of existing
    ;; prims. This can be overriden in the region config file (as
    ;; NonPhysicalPrimMax!).
    ; NonPhysicalPrimMax = 256

    ;# {PhysicalPrimMin} {} {Minimum size of physical prims?} {} 10
    ;; Maximum size where a prim can be physical.  Affects resizing of
    ;; existing prims.  This can be overriden in the region config file.
```

```
    ; PhysicalPrimMin = 0.01

    ;# {PhysicalPrimMax} {} {Maximum size of physical prims?} {} 10
    ;; Maximum size where a prim can be physical.  Affects resizing of
    ;; existing prims.  This can be overriden in the region config file.
    ; PhysicalPrimMax = 10

    ;# {ClampPrimSize} {} {Clamp viewer rezzed prims to max sizes?} {true false}
false
    ;; If a viewer attempts to rez a prim larger than the non-physical or
    ;; physical prim max, clamp the dimensions to the appropriate maximum
    ;; This can be overriden in the region config file.
    ; ClampPrimSize = false

    ;# {LinksetPrims} {} {Max prims an object will hold?} {} 0
    ;; Maximum number of prims allowable in a linkset. Affects creating new
    ;; linksets. Ignored if less than or equal to zero.
    ;; This can be overriden in the region config file.
    ; LinksetPrims = 0

    ;# {AllowScriptCrossing} {} {Allow scripts to cross into this region} {true false}
true
    ;; Allow scripts to keep running when they cross region boundaries, rather
    ;; than being restarted.  State is reloaded on the destination region.
    ;; This only applies when crossing to a region running in a different
    ;; simulator.
    ;; For crossings where the regions are on the same simulator the script is
    ;; always kept running.
    ; AllowScriptCrossing = true
```

```
    ;# {TrustBinaries} {AllowScriptCrossing:true} {Accept compiled binary script
code? (DANGEROUS!)} {true false} false
    ;; Allow compiled script binary code to cross region boundaries.
    ;; If you set this to "true", any region that can teleport to you can
    ;; inject ARBITRARY BINARY CODE into your system. Use at your own risk.
    ;; YOU HAVE BEEN WARNED!!!
    ; TrustBinaries = false

    ;# {CombineContiguousRegions} {} {Create megaregions where possible? (Do
not use with existing content!)} {true false} false
    ;; Combine all contiguous regions into one large megaregion
    ;; Order your regions from South to North, West to East in your regions.ini
    ;; and then set this to true
    ;; Warning!  Don't use this with regions that have existing content!,
    ;; This will likely break them
    ; CombineContiguousRegions = false

    ;# {InworldRestartShutsDown} {} {Shutdown instance on region restart?} {true
false} false
    ;; If you have only one region in an instance, or to avoid the many bugs
    ;; that you can trigger in modules by restarting a region, set this to
    ;; true to make the entire instance exit instead of restarting the region.
    ;; This is meant to be used on systems where some external system like
    ;; Monit will restart any instance that exits, thereby making the shutdown
    ;; into a restart.
    ; InworldRestartShutsDown = false

    ;; Persistence of changed objects happens during regular sweeps. The
    ;; following control that behaviour to prevent frequently changing objects
    ;; from heavily loading the region data store.
    ;; If both of these values are set to zero then persistence of all changed
```

```
    ;; objects will happen on every sweep.

    ;# {MinimumTimeBeforePersistenceConsidered} {} {Time before un-changed
object may be persisted} {} 60
    ;; Objects will be considered for persistance in the next sweep when they
    ;; have not changed for this number of seconds.
    ; MinimumTimeBeforePersistenceConsidered = 60

    ;# {MaximumTimeBeforePersistenceConsidered} {} {Time before changed
objects may be persisted?} {} 600
    ;; Objects will always be considered for persistance in the next sweep
    ;; if the first change occurred this number of seconds ago.
    ; MaximumTimeBeforePersistenceConsidered = 600

    ;# {physical_prim} {} {Allow prims to be physical?} {true false} true
    ;; if you would like to allow prims to be physical and move by physics
    ;; with the physical checkbox in the client set this to true.
    ; physical_prim = true

    ;; Select a mesher here.
    ;;
    ;; Meshmerizer properly handles complex prims by using triangle meshes.
    ;; Note that only the ODE physics engine currently deals with meshed
    ;; prims in a satisfactory way.

    ;# {meshing} {} {Select mesher} {Meshmerizer ZeroMesher} Meshmerizer
    ;; ZeroMesher is faster but leaves the physics engine to model the mesh
    ;; using the basic shapes that it supports.
    ;; Usually this is only a box.
    ;; Default is Meshmerizer
    ; meshing = Meshmerizer
```

```
    ; meshing = ZeroMesher

    ;; Choose one of the physics engines below
    ;# {physics} {} {Select physics engine} {OpenDynamicsEngine BulletSim
basicphysics POS} OpenDynamicsEngine
    ;; OpenDynamicsEngine is by some distance the most developed physics
engine
    ;; BulletSim is experimental and in active development.
    ;; basicphysics effectively does not model physics at all, making all
    ;; objects phantom.
    ;; Default is OpenDynamicsEngine
    ; physics = OpenDynamicsEngine
    ; physics = BulletSim
    ; physics = basicphysics
    ; physics = POS

    ;# {DefaultScriptEngine} {} {Default script engine} {XEngine} XEngine
    ;; Default script engine to use. Currently, we only have XEngine
    ; DefaultScriptEngine = "XEngine"

    ;# {HttpProxy} {} {Proxy URL for llHTTPRequest and dynamic texture loading} {}
http://proxy.com:8080
    ;; Http proxy setting for llHTTPRequest and dynamic texture loading, if
    ;; required
    ; HttpProxy = "http://proxy.com:8080"

    ;# {HttpProxyExceptions} {HttpProxy} {Set of regular expressions defining URL
that should not be proxied} {}
    ;; If you're using HttpProxy, then you can set HttpProxyExceptions to a
    ;; list of regular expressions for URLs that you don't want to go through
    ;; the proxy.
```

```
    ;; For example, servers inside your firewall.
    ;; Separate patterns with a ';'
    ; HttpProxyExceptions = ".mydomain.com;localhost"

    ;# {emailmodule} {} {Provide llEmail and llGetNextEmail functionality? (requires SMTP server)} {true false} false
    ;; The email module requires some configuration. It needs an SMTP
    ;; server to send mail through.
    ; emailmodule = DefaultEmailModule

    ;# {SpawnPointRouting} {} {Set routing method for Telehub Spawnpoints} {closest random sequence} closest
    ;; SpawnPointRouting adjusts the landing for incoming avatars.
    ;; "closest" will place the avatar at the SpawnPoint located in the closest
    ;; available spot to the destination (typically map click/landmark).
    ;; "random" will place the avatar on a randomly selected spawnpoint;
    ;; "sequence" will place the avatar on the next sequential SpawnPoint
    ; SpawnPointRouting = closest

    ;# {TelehubAllowLandmark} {} {Allow users with landmarks to override telehub routing} {true false} false
    ;; TelehubAllowLandmark allows users with landmarks to override telehub
    ;; routing and land at the landmark coordinates when set to true
    ;; default is false
    ; TelehubAllowLandmark = false


[AccessControl]
    ;# {AllowedClients} {} {Bar (|) separated list of allowed clients} {}
    ;; Bar (|) separated list of viewers which may gain access to the regions.
    ;; One can use a substring of the viewer name to enable only certain
```

```
    ;; versions
    ;; Example: Agent uses the viewer "Imprudence 1.3.2.0"
    ;; - "Imprudence" has access
    ;; - "Imprudence 1.3" has access
    ;; - "Imprudence 1.3.1" has no access
    ; AllowedClients =

    ;# {BannedClients} {} {Bar (|) separated list of banned clients} {}
    ;; Bar (|) separated list of viewers which may not gain access to the regions.
    ;; One can use a Substring of the viewer name to disable only certain
    ;; versions
    ;; Example: Agent uses the viewer "Imprudence 1.3.2.0"
    ;; - "Imprudence" has no access
    ;; - "Imprudence 1.3" has no access
    ;; - "Imprudence 1.3.1" has access
    ; BannedClients =


[Map]
    ;# {GenerateMaptiles} {} {Generate map tiles?} {true false} true
    ;; Map tile options.
    ;; If true, then maptiles are generated using the MapImageModule below.
    ;; If false then the texture referenced by MaptileStaticUUID is used instead,
which can also be overriden
    ;; in individual region config file(s).  If you do not want to upload map tiles at all,
then you will need
    ;; both to set this to false and comment out the [Modules]
MapImageServiceModule setting in config-include/
    ; GenerateMaptiles = true
```

```
    ;# {MapImageModule} {} {The map image module to use} {MapImageModule
Warp3DImageModule} MapImageModule
    ;; The module to use in order to generate map images.
    ;; MapImageModule is the default.  Warp3DImageModule is an alternative
experimental module that can
    ;; generate better images.
    ;MapImageModule = "MapImageModule"

    ;# {MaptileRefresh} {GenerateMaptiles} {Maptile refresh period?} {} 0
    ;; If desired, a running region can update the map tiles periodically
    ;; to reflect building activity. This names no sense of you don't have
    ;; prims on maptiles. Value is in seconds.
    ; MaptileRefresh = 0

    ;# {MaptileStaticUUID} {} {Asset ID for static map texture} {} 00000000-0000-
0000-0000-000000000000
    ;; If not generating maptiles, use this static texture asset ID
    ; MaptileStaticUUID = "00000000-0000-0000-0000-000000000000"

    ;# {TextureOnMapTile} {} {Use terrain textures for map tiles?} {true false} true
    ;; Use terrain texture for maptiles if true, use shaded green if false
    ; TextureOnMapTile = true

    ;# {DrawPrimOnMapTile} {} {Draw prim shapes on map tiles?} {true false} false
    ;; Draw objects on maptile.  This step might take a long time if you've
    ;; got a large number of objects, so you can turn it off here if you'd like.
    ; DrawPrimOnMapTile = true


[Permissions]
```

```
    ;# {permissionmodules} {} {Permission modules to use (may specify multiple
modules, separated by comma} {} DefaultPermissionsModule
    ;; Permission modules to use, separated by comma.
    ;; Possible modules are DefaultPermissionsModule, PrimLimitsModule
    ; permissionmodules = DefaultPermissionsModule

    ;# {serverside_object_permissions}
{permissionmodules:DefaultPermissionsModule} {Activate permission handling by
the sim?} {true false} true
    ;; These are the parameters for the default permissions module
    ;;
    ;; If set to false, then, in theory, the server never carries out
    ;; permission checks (allowing anybody to copy
    ;; any item, etc.  This may not yet be implemented uniformally.
    ;; If set to true, then all permissions checks are carried out
    ; serverside_object_permissions = true

    ;# {allow_grid_gods} {} {Allow grid gods?} {true false} false
    ;; This allows users with a UserLevel of 200 or more to assume god
    ;; powers in the regions in this simulator.
    allow_grid_gods = true

    ;; This allows some control over permissions
    ;; please note that this still doesn't duplicate SL, and is not intended to
    ;# {region_owner_is_god} {} {Allow region owner gods} {true false} true
    ;; Allow region owners to assume god powers in their regions
    region_owner_is_god = true

    ;# {region_manager_is_god} {} {Allow region manager gods} {true false} false
    ;; Allow region managers to assume god powers in regions they manage
    ; region_manager_is_god = false
```

```
    ;# {parcel_owner_is_god} {} {Allow parcel owner gods} {true false} true
    ;; Allow parcel owners to assume god powers in their parcels
    ; parcel_owner_is_god = true

    ;# {simple_build_permissions} {} {Allow building in parcel by access list (no
groups)} {true false} false
    ;; More control over permissions
    ;; This is definitely not SL!
    ;; Provides a simple control for land owners to give build rights to
    ;; specific avatars in publicly accessible parcels that disallow object
    ;; creation in general.
    ;; Owners specific avatars by adding them to the Access List of the parcel
    ;; without having to use the Groups feature
    ; simple_build_permissions = false


[Estates]
    ; If these values are commented out then the user will be asked for estate details
when required (this is the normal case).
        ; If these values are uncommented then they will be used to create a default
estate as necessary.
    ; New regions will be automatically assigned to that default estate.

    ;# {DefaultEstateName} {} {Default name for estate?} {} My Estate
    ;; Name for the default estate
    DefaultEstateName = UFSM

    ;# {DefaultEstateOwnerName} {} {Default estate owner name?} {} FirstName
LastName
    ;; Name for default estate owner
```

DefaultEstateOwnerName = giliane bernardi

; ** Standalone Estate Settings **
; The following parameters will only be used on a standalone system to
; create an estate owner that does not already exist

;# {DefaultEstateOwnerUUID} {} {Default estate owner UUID?} {} 00000000-0000-0000-0000-000000000000
;; If DefaultEstateOwnerUUID is left at UUID.Zero (as below) then a random
;; UUID will be assigned. This is normally what you want
; DefaultEstateOwnerUUID = 00000000-0000-0000-0000-000000000000

;# {DefaultEstateOwnerEMail} {} {Default estate owner email?} {}
;; Email address for the default estate owner
; DefaultEstateOwnerEMail = owner@domain.com

;# {DefaultEstateOwnerPassword} {} {Default estate owner password} {}
;; Password for the default estate owner
; DefaultEstateOwnerPassword = password

[SMTP]
;; The SMTP server enabled the email module to send email to external
;; destinations.

;# {enabled} {[Startup]emailmodule:DefaultEmailModule} {Enable SMTP service?} {true false} false
;; Enable sending email via SMTP
; enabled = false

;# {internal_object_host} {[Startup]emailmodule:DefaultEmailModule enabled:true} {Host name to treat as internal (object to object) email?} {} lsl.opensim.local
; internal_object_host = lsl.opensim.local

;# {host_domain_header_from} {[Startup]emailmodule:DefaultEmailModule enabled:true} {From address to use in the sent email header?} {} 200.18.72.10
; host_domain_header_from = "200.18.72.10"

;# {email_pause_time} {[Startup]emailmodule:DefaultEmailModule enabled:true} {Period in seconds to delay after an email is sent.} {} 20
; email_pause_time = 20

;# {email_max_size} {[Startup]emailmodule:DefaultEmailModule enabled:true} {Maximum total size of email in bytes.} {} 4096
; email_max_size = 4096

;# {SMTP_SERVER_HOSTNAME} {[Startup]emailmodule:DefaultEmailModule enabled:true} {SMTP server name?} {} 200.18.72.10
; SMTP_SERVER_HOSTNAME = "200.18.72.10"

;# {SMTP_SERVER_PORT} {[Startup]emailmodule:DefaultEmailModule enabled:true} {SMTP server name?} {SMTP server port?} {} 25
; SMTP_SERVER_PORT = 25

;# {SMTP_SERVER_LOGIN} {[Startup]emailmodule:DefaultEmailModule enabled:true} {SMTP server user name?} {}
; SMTP_SERVER_LOGIN = ""

;# {SMTP_SERVER_PASSWORD} {[Startup]emailmodule:DefaultEmailModule enabled:true} {SMTP server password} {}

```
    ; SMTP_SERVER_PASSWORD = ""


[Network]
    ;# {ConsoleUser} {} {User name for console account} {}
    ;; Configure the remote console user here. This will not actually be used
    ;; unless you use -console=rest at startup.
    ; ConsoleUser = "Test"
    ;# {ConsolePass} {} {Password for console account} {}
    ; ConsolePass = "secret"
    ;# {console_port} {} {Port for console connections} {} 0
    ; console_port = 0

    ;# {http_listener_port} {} {TCP Port for this simulator to listen on? (This must be
unique to the simulator!)} {} 9000
    ;; Simulator HTTP port. This is not the region port, but the port the
    ;; entire simulator listens on. This port uses the TCP protocol, while
    ;; the region ports use UDP.
    http_listener_port = 9000

    ;# {ExternalHostNameForLSL} {} {Hostname to use for HTTP-IN URLs. This
should be reachable from the internet.} {}
    ;; Hostname to use in llRequestURL/llRequestSecureURL
    ;; if not defined - default machine name is being used
    ;; (on Windows this mean NETBIOS name - useably only inside local network)
    ; ExternalHostNameForLSL = "200.18.72.10"

    ;# {shard} {} {Name to use for X-Secondlife-Shard header? (press enter if
unsure)} {} OpenSim
    ;; What is reported as the "X-Secondlife-Shard"
    ;; Defaults to the user server url if not set
```

```
    ;; The old default is "OpenSim", set here for compatibility
    ;; The below is not commented for compatibility.
    shard = "OpenSim"

    ;# {user_agent} {} {User agent to report to web servers?} {} OpenSim LSL
(Mozilla Compatible)
    ;; What is reported as the "User-Agent" when using llHTTPRequest
    ;; Defaults to not sent if not set here. See the notes section in the wiki
    ;; at http://wiki.secondlife.com/wiki/LlHTTPRequest for comments on adding
    ;; " (Mozilla Compatible)" to the text where there are problems with a
    ;; web server
    ; user_agent = "OpenSim LSL (Mozilla Compatible)"


[XMLRPC]
    ;# {XmlRpcRouterModule} {} {Module used to route incoming llRemoteData
calls} {XmlRpcRouterModule XmlRpcGridRouterModule} XmlRpcRouterModule
    ;; If enabled and set to XmlRpcRouterModule, this will post an event,
    ;; "xmlrpc_uri(string)" to the script concurrently with the first
    ;; remote_data event. This will contain the fully qualified URI an
    ;; external site needs to use to send XMLRPC requests to that script
    ;;
    ;; If enabled and set to XmlRpcGridRouterModule, newly created channels
    ;; will be registered with an external service via a configured uri
    ;XmlRpcRouterModule = "XmlRpcRouterModule"

    ;# {XmlRpcPort} {} {Port for incoming llRemoteData xmlrpc calls} {} 20800
    ;XmlRpcPort = 20800

    ;# {XmlRpcHubURI} {XmlRpcRouterModule} {URI for external service used to
register xmlrpc channels created in the simulator. This depends on
```

```
XmlRpcRouterModule being set to XmlRpcGridRouterModule} {}
http://example.com
    ;; If XmlRpcRouterModule is set to XmlRpcGridRouterModule, the simulator
    ;; will use this address to register xmlrpc channels on the external
    ;; service
    ; XmlRpcHubURI = http://example.com


[ClientStack.LindenUDP]
    ;; See OpensSimDefaults.ini for the throttle options. You can copy the
    ;; relevant sections and override them here.
    ;; DO NOT MODIFY OpenSimDefaults.ini, as your changes would be lost
    ;; with the next update!

    ;# {DisableFacelights} {} {Stop facelights from working?} {true false} false
    ;; Quash and remove any light properties from attachments not on the
    ;; hands. This allows flashlights and lanterns to function, but kills
    ;; silly vanity "Facelights" dead. Sorry, head mounted miner's lamps
    ;; will also be affected.
    ;; This is especially important in artistic builds that depend on lights
    ;; on the build for their appearance, since facelights will cause the
    ;; building's lights to possibly not be rendered.
    ; DisableFacelights = "false"


[ClientStack.LindenCaps]
    ;; For the long list of capabilities, see OpenSimDefaults.ini
    ;; Here are the few ones you may want to change. Possible values
    ;; are:
    ;;   "" -- empty, capability disabled
    ;;   "localhost" -- capability enabled and served by the simulator
```

```
    ;;   "<url>" -- capability enabled and served by some other server
    ;;
    ; These are enabled by default to localhost. Change if you see fit.
    Cap_GetTexture = "localhost"
    Cap_GetMesh = "localhost"
    Cap_AvatarPickerSearch = "localhost"

    ; This is disabled by default. Change if you see fit. Note that
    ; serving this cap from the simulators may lead to poor performace.
    Cap_WebFetchInventoryDescendents = ""


[SimulatorFeatures]
    ;# {MapImageServerURI} {} {URL for the map server} {}
    ; Experimental new information sent in SimulatorFeatures cap for Kokua
    ; viewers
    ; meant to override the MapImage and search server url given at login, and
varying
    ; on a sim-basis.
    ; Viewers that don't understand it, will ignore it
    ;MapImageServerURI = "http://200.18.72.10:9000/"
    ;# {SearchServerURI} {} {URL of the search server} {}
    ;SearchServerURI = "http://200.18.72.10:9000/"


[Chat]
    ;# {whisper_distance} {} {Distance at which a whisper is heard, in meters?} {} 10
    ;; Distance in meters that whispers should travel.
    ; whisper_distance = 10

    ;# {say_distance} {} {Distance at which normal chat is heard, in meters?} {} 20
```

```
    ;; Distance in meters that ordinary chat should travel.
    ; say_distance = 20

    ;# {shout_distance} {} {Distance at which a shout is heard, in meters?} {} 100
    ;; Distance in meters that shouts should travel.
    ; shout_distance = 100


[EntityTransfer]
    ;# {DisableInterRegionTeleportCancellation} {} {Determine whether the cancel button is shown at all during teleports.} {false true} false
    ;; This option exists because cancelling at certain points can result in an unuseable session (frozen avatar, etc.)
    ;; Disabling cancellation can be okay in small closed grids where all teleports are highly likely to suceed.
    ;DisableInterRegionTeleportCancellation = false
MaxOutgoingTransferVersion = "SIMULATION/0.1"

[Messaging]
    ;# {OfflineMessageModule} {} {Module to use for offline message storage} {OfflineMessageModule "Offline Message Module V2" *}
    ;; Module to handle offline messaging. The core module requires an external
    ;; web service to do this. See OpenSim wiki.
    ; OfflineMessageModule = OfflineMessageModule
    ;; Or, alternatively, use this one, which works for both standalones and grids
    ; OfflineMessageModule = "Offline Message Module V2"

    ;# {OfflineMessageURL} {OfflineMessageModule:OfflineMessageModule Offline Message Module V2:Offline Message Module V2} {URL of offline messaging service} {}
```

;; URL of web service for offline message storage. Leave it commented if your service is local to the sim.

; OfflineMessageURL = http://yourserver/Offline.php or http://yourrobustserver:8003

;# {StorageProvider} {Offline Message Module V2:Offline Message Module V2} {DLL that provides the storage interface} {OpenSim.Data.MySQL.dll}

;; For standalones, this is the storage dll.

; StorageProvider = OpenSim.Data.MySQL.dll

;# {MuteListModule} {OfflineMessageModule:OfflineMessageModule} {} {} MuteListModule

;; Mute list handler (not yet implemented). MUST BE SET to allow offline

;; messages to work

; MuteListModule = MuteListModule

;# {MuteListURL} {OfflineMessageModule:OfflineMessageModule} {} {} http://yourserver/Mute.php

;; URL of the web service that serves mute lists. Not currently used, but

;; must be set to allow offline messaging to work.

; MuteListURL = http://yourserver/Mute.php

;; Control whether group invites and notices are stored for offline users.

;; Default is true.

;; This applies to both core groups module.

; ForwardOfflineGroupMessages = true

[ODEPhysicsSettings]

;# {mesh_sculpted_prim} {[Startup]physics:OpenDynamicsEngine} {Mesh sculpties so they collide as they look?} {true false} true

```
    ;; Do we want to mesh sculpted prim to collide like they look?
    ;; If you are seeing sculpt texture decode problems
    ;; (messages such as "Decoded image with unhandled number of components: 0
shortly followed by a physcs exception")
    ;; then you might want to try setting this to false.
    ; mesh_sculpted_prim = true

    ;# {use_NINJA_physics_joints} {[Startup]physics:OpenDynamicsEngine} {Use
jointed (NINJA) physics?} {true false} false
    ;; If you would like physics joints to be enabled through a special naming
    ;; convention in the client, set this to true.
    ;; (see NINJA Physics, http://opensimulator.org/wiki/NINJA_Physics)
    ; use_NINJA_physics_joints = false


[RemoteAdmin]
    ;; This is the remote admin module, which uses XMLRPC requests to
    ;; manage regions from a web interface.

    ;# {enabled} {} {Enable the remote admin interface?} {true false} false
    ; enabled = false

    ;# {port} {enabled:true} {Port to use for the remote admin interface? (0 = default)}
{} 0
    ;; Set this to a nonzero value to have remote admin use a different port
    ; port = 0

    ;# {access_password} {enabled:true} {Password for the remote admin interface}
{}
    ;; This password is required to make any XMLRPC call (should be set as
    ;; the "password" parameter)
```

```
    ; access_password = ""

    ;# {access_ip_addresses} {enabled:true} {List the IP addresses allowed to call
RemoteAdmin?} {}
    ;; List the IP addresses allowed to call RemoteAdmin
    ;; If access_ip_addresses isn't set, then all IP addresses can access
RemoteAdmin.
    ;; access_ip_addresses = 0.0.0.0, 0.0.0.0 ...
    ; access_ip_addresses =

    ;# {create_region_enable_voice} {enabled:true} {Enable voice for newly created
regions?} {true false} false
    ;; set this variable to true if you want the create_region XmlRpc
    ;; call to unconditionally enable voice on all parcels for a newly
    ;; created region
    create_region_enable_voice = true

    ;# {create_region_public} {enabled:true} {Make newly created regions public?}
{true false} false
    ;; set this variable to false if you want the create_region XmlRpc
    ;; call to create all regions as private per default (can be
    ;; overridden in the XmlRpc call)
    ; create_region_public = false

    ;# {enabled_methods} {enabled:true} {List of methods to allow, separated by |} {}
all
    ;; enable only those methods you deem to be appropriate using a | delimited
    ;; whitelist.
    ;; For example:
    ;; enabled_methods = admin_broadcast|admin_save_oar|admin_save_xml
    ;; if this parameter is not specified but enabled = true, all methods
```

```
    ;; will be available
    ; enabled_methods = all

    ;; specify the default appearance for an avatar created through the remote
    ;; admin interface
    ;; This will only take effect is the file specified by the
    ;; default_appearance setting below exists
    ; default_male = Default Male
    ; default_female = Default Female

    ;; Update appearance copies inventory items and wearables of default
    ;; avatars. if this value is false, just worn assets are copied to the
    ;; Clothes folder; if true, all Clothes and Bodyparts subfolders are copied.
    ;; The receiver will wear the same items the default avatar did wear.
    ; copy_folders = false

    ;; Path to default appearance XML file that specifies the look of the
    ;; default avatars
    ; default_appearance = default_appearance.xml


[Wind]
    ;# {enabled} {} {Enable wind module?} {true false} true
    ;; Enables the wind module.
    ; enabled = true

    ;# {wind_update_rate} {enabled:true} {Wind update rate in frames?} {} 150
    ;; How often should wind be updated, as a function of world frames.
    ;; Approximately 50 frames a second
    ; wind_update_rate = 150
```

```
    ;; The Default Wind Plugin to load
    ; wind_plugin = SimpleRandomWind

    ;; These settings are specific to the ConfigurableWind plugin
    ;; To use ConfigurableWind as the default, simply change wind_plugin
    ;; to ConfigurableWind and uncomment the following.
    ; avg_strength = 5.0
    ; avg_direction = 0.0
    ; var_strength = 5.0
    ; var_direction = 30.0
    ; rate_change = 1.0

    ;# {strength} {enabled:true wind_plugin:SimpleRandomWind} {Wind strength?} {}
1.0
    ;; This setting is specific to the SimpleRandomWind plugin
    ;; Adjusts wind strength. 0.0 = no wind, 1.0 = normal wind.
    ; strength = 1.0


[LightShare]
    ;# {enable_windlight} {} {Enable LightShare technology?} {true false} false
    ;; This enables the transmission of Windlight scenes to supporting clients,
    ;; such as the Meta7 viewer.
    ;; It has no ill effect on viewers which do not support server-side
    ;; windlight settings.
    ; enable_windlight = false


[DataSnapshot]
    ;# {index_sims} {} {Enable data snapshotting (search)?} {true false} false
    ;; The following set of configs pertains to search.
```

```
    ;; Set index_sims to true to enable search engines to index your
    ;; searchable data.
    ;; If false, no data will be exposed, DataSnapshot module will be off,
    ;; and you can ignore the rest of these search-related configs.
    ; index_sims = false

    ;# {data_exposure} {index_sims:true} {How much data should be exposed?}
{minimum all} minimum
    ;; The variable data_exposure controls what the regions expose:
    ;;    minimum: exposes only things explicitly marked for search
    ;;    all: exposes everything
    ; data_exposure = minimum

    ;# {gridname} {index_sims:true} {Enter the name fo your grid} {} OSGrid
    ;; If search is on, change this to your grid name; will be ignored for
    ;; standalones
    ; gridname = "OSGrid"

    ;# {default_snapshot_period} {index_sims:true} {Period between data
snapshots?} {} 1200
    ;; Period between data snapshots, in seconds. 20 minutes, for starters,
    ;; so that you see the initial changes fast.
    ;; Later, you may want to increase this to 3600 (1 hour) or more
    ; default_snapshot_period = 1200

    ;; This will be created in bin, if it doesn't exist already. It will hold
    ;; the data snapshots.
    ; snapshot_cache_directory = "DataSnapshot"

    ;# {data_services} {index_sims:true} {Data service URLs to register with?} {}
http://metaverseink.com/cgi-bin/register.py
```

```
    ; This semicolon-separated string serves to notify specific data services
    ; about the existence of this sim. Uncomment if you want to index your
    ; data with this and/or other search providers.
    ; data_services="http://metaverseink.com/cgi-bin/register.py"


[Economy]
    ;# {SellEnabled} {} {Enable selling for 0?} {true false} true
    ; The default economy module only implements just enough to allow free actions
(transfer of objects, etc).
    ; There is no intention to implement anything further in core OpenSimulator.
    ; This functionality has to be provided by third party modules.

    ;; Enables selling things for $0.  Default is true.
    ; SellEnabled = true

    ;# {PriceUpload} {} {Price for uploading?} {} 0
    ;; Money Unit fee to upload textures, animations etc.  Default is 0.
    ; PriceUpload = 0

    ;# {PriceGroupCreate} {} {Fee for group creation} {} 0
    ;; Money Unit fee to create groups.  Default is 0.
    ; PriceGroupCreate = 0


[XEngine]
    ;# {Enabled} {} {Enable the XEngine scripting engine?} {true false} true
    ;; Enable this engine in this OpenSim instance
    Enabled = true

    ;; How many threads to keep alive even if nothing is happening
```

```
    ; MinThreads = 2

    ;; How many threads to start at maximum load
    ; MaxThreads = 100

    ;; Time a thread must be idle (in seconds) before it dies
    ; IdleTimeout = 60

    ;# {Priority} {Enabled:true} {Priority for script engine threads?} {Lowest
BelowNormal Normal AboveNormal Highest} BelowNormal
    ;; Thread priority ("Lowest", "BelowNormal", "Normal", "AboveNormal",
    ;; "Highest")
    ; Priority = "BelowNormal"

    ;; Maximum number of events to queue for a script (excluding timers)
    ; MaxScriptEventQueue = 300

    ;; Stack size per script engine thread in bytes.
    ;; If you are experiencing StackOverflowExceptions you may want to increase
this (e.g. double it).
    ;; The trade-off may be increased memory usage by the script engine.
    ; ThreadStackSize = 262144

    ;; Set this to true (the default) to load each script into a separate
    ;; AppDomain.
    ;;
    ;; Setting this to false will load all script assemblies into the
    ;; current AppDomain, which will significantly improve script loading times.
    ;; It will also reduce initial per-script memory overhead.
    ;;
```

;; However, setting this to false will also prevent script DLLs from being unloaded from memory if the script is deleted.

;; This may cause an OutOfMemory problem over time when avatars with scripted attachments move in and out of the region.

;; Some Windows users have also reported script loading problems when AppDomainLoading = false

; AppDomainLoading = true

;; Controls whether scripts are stopped by aborting their threads externally (abort) or by co-operative checks from the compiled script (co-op)

;; co-op will be more stable but this option is currently experimental.

;; If moving from co-op to abort, existing script DLLs will need to be recompiled.

;; This currently can only be done manually, either by setting DeleteScriptsOnStartup = true for one run

;; or by deleting the script DLL* files in bin/ScriptEngines/<region-id>/

;; One can move from co-op back to abort without recompilation, but reverting back to co-op again will need script recompile

;; Current valid values are "abort" and "co-op"

; ScriptStopStrategy = abort

;# {DeleteScriptsOnStartup} {} {Delete previously compiled script DLLs on startup?} {true false} true

;; Controls whether previously compiled scripts DLLs are deleted on sim restart. If you set this to false

;; then startup will be considerably faster since scripts won't need to be recompiled. However, then it becomes your responsibility to delete the

;; compiled scripts if you're recompiling OpenSim from source code and internal interfaces used

;; by scripts have changed.

; DeleteScriptsOnStartup = true

```
    ;# {DefaultCompileLanguage} {Enabled:true} {Default script language?} {lsl vb
cs} lsl
    ;; Default language for scripts
    ; DefaultCompileLanguage = "lsl"

    ;# {AllowedCompilers} {Enabled:true} {Languages to allow (comma separated)?}
{} lsl
    ;; List of allowed languages (lsl,vb,cs)
    ;; AllowedCompilers=lsl,cs,js,vb.
    ;; *warning*, non lsl languages have access to static methods such as
    ;; System.IO.File.  Enable at your own risk.
    ; AllowedCompilers = "lsl"

    ;; Compile debug info (line numbers) into the script assemblies
    ; CompileWithDebugInformation = true

    ;; Allow the user of mod* functions.  This allows a script to pass messages
    ;; to a region module via the modSendCommand() function
    ;; Default is false
    ; AllowMODFunctions = false

    ;# {AllowOSFunctions} {Enabled:true} {Allow OSFunctions? (DANGEROUS!)}
{true false} false
    ;; Allow the use of os* functions (some are dangerous)
    AllowOSFunctions = true

    ;# {AllowLightShareFunctions} {Enabled:false [LightShare]enable_windlight:true}
{Allow LightShare functions?} {true false} false
    ; Allow the use of LightShare functions.
```

```
    ; The setting enable_windlight = true must also be enabled in the [LightShare]
section.
    ; AllowLightShareFunctions = false

    ;# {OSFunctionThreatLevel} {Enabled:true AllowOSFunctions:true} {OSFunction
threat level? (DANGEROUS!)} {None VeryLow Low Moderate High VeryHigh
Severe} VeryLow
    ;; Threat level to allow, one of None, VeryLow, Low, Moderate, High, VeryHigh,
Severe
    ;; See http://opensimulator.org/wiki/Threat_level for more information on these
levels.
    ;; We do not recommend that use set a general level above Low unless you
have a high level of trust
    ;; in all the users that can run scripts in your simulator.  It is safer to explicitly
    ;; allow certain types of user to run higher threat level OSSL functions, as
detailed later on.
    OSFunctionThreatLevel = Severe
; osNPC specific functions
; note: these are High threat level functions
        Allow_osNpcCreate = true
        Allow_osNpcMoveTo = true
        Allow_osNpcRemove = true
        Allow_osNpcSay = true
        ; and these let us animate the NPCs
        ; note: these two are VeryHigh threat level functions
        Allow_osAvatarPlayAnimation = true
        Allow_osAvatarStopAnimation = true
    ; OS Functions enable/disable
    ; For each function, you can add one line, as shown
    ; The default for all functions allows them if below threat level
```

```
    ; true allows the use of the function unconditionally
    ; Allow_osSetRegionWaterHeight = true

    ; false disables the function completely
    ; Allow_osSetRegionWaterHeight = false

    ; Comma separated list of UUIDS allows the function for that list of UUIDS
    ; Allow_osSetRegionWaterHeight = 888760cb-a3cf-43ac-8ea4-8732fd3ee2bb

    ; Comma separated list of owner classes that allow the function for a particular
class of owners. Choices are
    ; - PARCEL_GROUP_MEMBER:  allow if the object group is the same group as
the parcel
    ; - PARCEL_OWNER:         allow if the object owner is the parcel owner
    ; - ESTATE_MANAGER:       allow if the object owner is an estate manager
    ; - ESTATE_OWNER:         allow if the object owner is the estate owner
    ; Allow_osSetRegionWaterHeight = 888760cb-a3cf-43ac-8ea4-8732fd3ee2bb,
PARCEL_OWNER, ESTATE_OWNER>, ...

    ; You can also use script creators as the uuid
    ; Creators_osSetRegionWaterHeight = <uuid>, ...

    ; If both Allow_ and Creators_ are given, effective permissions
    ; are the union of the two.

    ;# {EventLimit} {} {Amount of time a script can spend in an event handler} {} 30
    ;; Time a script can spend in an event handler before it is interrupted
    ; EventLimit = 30

    ;# {KillTimedOutScripts} {} {Kill script in case of event time overruns?} {true false}
false
```

```
    ;; If a script overruns it's event limit, kill the script?
    ; KillTimedOutScripts = false

    ;# {ScriptDelayFactor} {} {Multiplier for scripting delays} {} 1.0
    ;; Sets the multiplier for the scripting delays
    ; ScriptDelayFactor = 1.0

    ;# {ScriptDistanceLimitFactor} {} {Multiplier for 10.0m distance limits?} {}
    ;; The factor the 10 m distances limits are multiplied by
    ; ScriptDistanceLimitFactor = 1.0

    ;# {NotecardLineReadCharsMax} {} {Maximum length of notecard line?} {} 255
    ;; Maximum length of notecard line read
    ;; Increasing this to large values potentially opens
    ;; up the system to malicious scripters
    ; NotecardLineReadCharsMax = 255

    ;# {SensorMaxRange} {} {Sensor range} {} 96.0
    ;; Sensor settings
    ; SensorMaxRange = 96.0
    ;# {SensorMaxResults} {} {Max sensor results returned?} {}
    ; SensorMaxResults = 16

    ;# {DisableUndergroundMovement} {} {Disable underground movement of prims}
{true false} true
    ;; Disable underground movement of prims (default true); set to
    ;; false to allow script controlled underground positioning of
    ;; prims
    ; DisableUndergroundMovement = true

    ;# {ScriptEnginesPath} {} {Path to script assemblies} {} ScriptEngines
```

```
;; Path to script engine assemblies
;; Default is ./bin/ScriptEngines
; ScriptEnginesPath = "ScriptEngines"

      ;; ========== OSSL FUNCTION BLOCK ==========
;; OS FUNCTIONS as of 8:23 AM February-01-12
;; Extracted from OpenSimulator DEV 0.7.3 OpenSim-003bd9f-r/17959
;;
;; ADJUST as needed or desired
;; ======================================
;;
;; *** Threat-Level=None
;Allow_osDrawEllipse = true
;Allow_osDrawFilledPolygon = true
;Allow_osDrawFilledRectangle = true
;Allow_osDrawImage = true
;Allow_osDrawLine = true
;Allow_osDrawPolygon = true
;Allow_osDrawRectangle = true
;Allow_osDrawText = true
Allow_osGetAgents = true
Allow_osGetAvatarList = true
;Allow_osGetCurrentSunHour = true
;Allow_osGetMapTexture = true
;Allow_osGetSunParam = true
;Allow_osGetTerrainHeight = true
Allow_osIsNpc = true
;Allow_osList2Double = true
Allow_osMovePen = true
Allow_osNpcGetOwner = true
;Allow_osParseJSON = true
```

;Allow_osParseJSONNew = true
;Allow_osSetFontName = true
;Allow_osSetFontSize = true
;Allow_osSetPenCap = true
;Allow_osSetPenColor = true
;Allow_osSetPenColour = true
;Allow_osSetPenSize = true
;Allow_osSetSunParam = true
Allow_osTeleportOwner = true
;Allow_osWindActiveModelPluginName = true
;;
;; *** Threat-Level=Nuisance
;Allow_osSetEstateSunSettings = false
;Allow_osSetRegionSunSettings = false
;;
;; *** Threat-Level=VeryLow
;Allow_osGetDrawStringSize = true
;Allow_osGetWindParam = true
Allow_osNpcStopMoveToTarget = true
;Allow_osSetDynamicTextureData = true
;Allow_osSetDynamicTextureDataBlend = true
;Allow_osSetDynamicTextureDataBlendFace = true
;Allow_osSetDynamicTextureURL = true
;Allow_osSetDynamicTextureURLBlend = true
;Allow_osSetDynamicTextureURLBlendFace = true
;Allow_osSetParcelMediaURL = false
;Allow_osSetParcelSIPAddress = false
;Allow_osSetPrimFloatOnWater = true
;Allow_osSetWindParam = false
;Allow_osTerrainFlush = false
;Allow_osUnixTimeToTimestamp = true

```
;;
;; *** Threat-Level=Low
Allow_osAvatarName2Key = true
;Allow_osFormatString = true
Allow_osKey2Name = true
;Allow_osLoadedCreationDate = false
;Allow_osLoadedCreationID = false
;Allow_osLoadedCreationTime = false
Allow_osMessageObject = true
;;
;; *** Threat-Level=Moderate
;Allow_osGetGridCustom = false
;Allow_osGetGridHomeURI = false
;Allow_osGetGridLoginURI = false
;Allow_osGetGridName = false
;Allow_osGetGridNick = false
Allow_osGetRegionStats = true
;Allow_osGetSimulatorMemory = false
;Allow_osSetSpeed = false
;;
;; *** Threat-Level=High
Allow_osOwnerSaveAppearance = true
;Allow_osCauseDamage = false
;Allow_osCauseHealing = false
Allow_osGetAgentIP = true
;Allow_osGetLinkPrimitiveParams = false
;Allow_osGetPrimitiveParams = false
;Allow_osGetRegionMapTexture = false
;Allow_osGetScriptEngineName = false
;Allow_osGetSimulatorVersion = false
Allow_osMakeNotecard = true
```

;Allow_osMatchString = false
Allow_osNpcCreate = true
Allow_osNpcGetPos = true
Allow_osNpcGetRot = true
Allow_osNpcLoadAppearance = true
Allow_osNpcMoveTo = true
Allow_osNpcMoveToTarget = true
Allow_osNpcPlayAnimation = true
Allow_osNpcRemove = true
Allow_osNpcSaveAppearance = true
Allow_osNpcSay = true
Allow_osNpcSetRot = true
Allow_osNpcSit = true
Allow_osNpcStand = true
Allow_osNpcStopAnimation = true
;Allow_osParcelJoin = false
;Allow_osParcelSubdivide = false
;Allow_osRegionRestart = false
;Allow_osSetParcelDetails = false
;Allow_osSetPrimitiveParams = false
;Allow_osSetProjectionParams = false
Allow_osSetRegionWaterHeight = true
;Allow_osSetStateEvents = false
;Allow_osSetTerrainHeight = false
;;
;; *** Threat-Level=VeryHigh
Allow_osAvatarPlayAnimation = true
Allow_osAvatarStopAnimation = true
Allow_osGetNotecard = true
Allow_osGetNotecardLine = true
Allow_osGetNumberOfNotecardLines = true

```
    ;Allow_osRegionNotice = false
    Allow_osAgentSaveAppearance = true    ;(missing from IOSSL_API.cs)
    ;Allow_osSetRot = false    ;(missing from IOSSL_API.cs)
    ;;
    ;; *** Threat-Level=Severe
    ;Allow_osConsoleCommand = false
    Allow_osKickAvatar = true
    Allow_osTeleportAgent = true


[MRM]
    ;; Enables the Mini Region Modules Script Engine.
    ; Enabled = false

    ;; Runs MRM in a Security Sandbox
    ;; WARNING: DISABLING IS A SECURITY RISK.
    ; Sandboxed = true

    ;; The level sandbox to use, adjust at your OWN RISK.
    ;; Valid values are:
    ;; *  FullTrust
    ;; *  SkipVerification
    ;; *  Execution
    ;; *  Nothing
    ;; *  LocalIntranet
    ;; *  Internet
    ;; *  Everything
    ; SandboxLevel = "Internet"

    ;; Only allow Region Owners to run MRMs
    ;; May represent a security risk if you disable this.
```

```
    ; OwnerOnly = true


[FreeSwitchVoice]
    ;; In order for this to work you need a functioning FreeSWITCH PBX set up.
    ;; Configuration details at http://opensimulator.org/wiki/Freeswitch_Module
    ; Enabled = false

    ;; You need to load a local service for a standalone, and a remote service
    ;; for a grid region. Use one of the lines below, as appropriate
    ;; If you're using Freeswitch on a standalone then you will also need to configure
the [FreeswitchService] section in config-include/StandaloneCommon.ini
    ; LocalServiceModule =
OpenSim.Services.FreeswitchService.dll:FreeswitchService
    ; LocalServiceModule =
OpenSim.Services.Connectors.dll:RemoteFreeswitchConnector

    ;; If using a remote connector, specify the server URL
    ; FreeswitchServiceURL = http://my.grid.server:8004/fsapi


[Groups]
    ;# {Enabled} {} {Enable groups?} {true false} false
    ;; Enables the groups module
    Enabled = true

    ;# {LevelGroupCreate} {Enabled:true} {User level for creating groups} {} 0
    ;; Minimum user level required to create groups
    ; LevelGroupCreate = 0
```

```
    ;# {Module} {Enabled:true} {Groups module to use? (Use GroupsModule to use
Flotsam/Simian)} {Default "Groups Module V2"} Default
    ;; The default module can use a PHP XmlRpc server from the Flotsam project at
    ;; http://code.google.com/p/flotsam/
    ;; or from the SimianGrid project at http://code.google.com/p/openmetaverse
    ; Module = Default
    ;; or... use Groups Module V2, which works for standalones and robust grids
    ; Module = "Groups Module V2"

    ;# {StorageProvider} {Module:Groups Module V2} {The DLL that provides the
storage for V2} {OpenSim.Data.MySQL.dll}
    ; StorageProvider = OpenSim.Data.MySQL.dll

    ;# {ServicesConnectorModule} {Module:GroupsModule Module:Groups Module
V2} {Service connector to use for groups} {XmlRpcGroupsServicesConnector
SimianGroupsServicesConnector "Groups Local Service Connector" "Groups
Remote Service Connector" "Groups HG Service Connector"}
XmlRpcGroupsServicesConnector
    ;; Service connectors to the Groups Service as used in the GroupsModule.
Select one as follows:
    ;; -- for Flotsam Groups use XmlRpcGroupsServicesConnector
    ;; -- for Simian Groups use SimianGroupsServicesConnector
    ;; -- for V2 Groups, standalone, non-HG use "Groups Local Service Connector"
    ;; -- for V2 Groups, grided sim, non-HG use "Groups Remote Service Connector"
    ;; -- for V2 Groups, HG, both standalone and grided sim, use "Groups HG
Service Connector"
    ;; Note that the quotes "" around the words are important!
    ; ServicesConnectorModule = XmlRpcGroupsServicesConnector

    ;# {LocalService} {ServicesConnectorModule:Groups HG Service Connector} {Is
the group service in this process or elsewhere?} {local remote} local
```

```
    ;; Used for V2 in HG only. If standalone, set this to local; if grided sim, set this to
remote
    ; LocalService = local

    ;# {GroupsServerURI} {Module:GroupsModule
(ServicesConnectorModule:Groups Remote Service Connector or
(ServicesConnectorModule:Groups HG Service Connector and
LocalService:remote))} {Groups Server URI} {}
    ;; URI for the groups services of this grid
    ;; e.g. http://yourxmlrpcserver.com/xmlrpc.php for Flotsam XmlRpc
    ;; or http://mygridserver.com:82/Grid/ for SimianGrid
    ;; or http:://mygridserver.com:8003 for robust, V2
    ;; Leave it commented for standalones, V2
    ; GroupsServerURI = ""

    ;# {HomeURI} {ServicesConnectorModule:Groups HG Service Connector}
{What's the home address of this world?} {}
    ;; Used for V2 in HG only. For example
    ;; http://mygridserver.com:9000 or http://mygridserver.com:8002
    ;; If you have this set under [Startup], no need to set it here, leave it commented
    ; HomeURI = ""

    ;# {MessagingEnabled} {Module:GroupsModule Module:Groups Module V2} {Is
groups messaging enabled?} {true false} true
    ; MessagingEnabled = true

    ;# {MessagingModule} {MessagingEnabled:true} {Module to use for groups
messaging} {GroupsMessagingModule "Groups Messaging Module V2"}
GroupsMessagingModule
    ; MessagingModule = GroupsMessagingModule
    ;; or use V2 for Groups V2
```

```
    ; MessagingModule = "Groups Messaging Module V2"

    ;# {NoticesEnabled} {Module:GroupsModule Module:Groups Module V2}
{Enable group notices?} {true false} true
    ;; Enable Group Notices
    ; NoticesEnabled = true

    ;# {MessageOnlineUsersOnly} {Module:GroupsModule Module} {Message online
users only?} {true false} false
    ; Experimental option to only message online users rather than all users
    ; Should make large groups with few online members messaging faster, as the
expense of more calls to presence service
    ; Applies Flotsam Group only. V2 has this always on, no other option
    ; MessageOnlineUsersOnly = false

    ;; This makes the Groups modules very chatty on the console.
    ; DebugEnabled = false

    ;; XmlRpc Security settings.  These must match those set on your backend
    ;; groups service if the service is using these keys
    ; XmlRpcServiceReadKey    = 1234
    ; XmlRpcServiceWriteKey   = 1234


[InterestManagement]
     ;# {UpdatePrioritizationScheme} {} {Update prioritization scheme?}
{BestAvatarResponsiveness Time Distance SimpleAngularDistance FrontBack}
BestAvatarResponsiveness
     ;; This section controls how state updates are prioritized for each client
     ;; Valid values are BestAvatarResponsiveness, Time, Distance,
     ;; SimpleAngularDistance, FrontBack
```

```
    ; UpdatePrioritizationScheme = BestAvatarResponsiveness


[MediaOnAPrim]
    ;# {Enabled} {} {Enable Media-on-a-Prim (MOAP)} {true false} true
    ;; Enable media on a prim facilities
    ; Enabled = true;


[NPC]
    ;# {Enabled} {} {Enable Non Player Character (NPC) facilities} {true false} false
    Enabled = true


[Terrain]
    ;# {InitialTerrain} {} {Initial terrain type} {pinhead-island flat} pinhead-island
    ; InitialTerrain = "pinhead-island"


[UserProfiles]
  ;# {ProfileURL} {} {Set url to UserProfilesService} {}
  ;; Set the value of the url to your UserProfilesService
  ;; If un-set / "" the module is disabled
  ;; ProfileServiceURL = http://200.18.72.10:8000


[Architecture]
    ;# {Include-Architecture} {} {Choose one of the following architectures} {config-
include/Standalone.ini config-include/StandaloneHypergrid.ini config-
include/Grid.ini config-include/GridHypergrid.ini config-include/SimianGrid.ini
config-include/HyperSimianGrid.ini} config-include/Standalone.ini
```

```
    ;; Uncomment one of the following includes as required.  For instance, to create a standalone OpenSim,
    ;; uncomment Include-Architecture = "config-include/Standalone.ini"
    ;;
    ;; Then you will need to copy and edit the corresponding *Common.example file in config-include/
    ;; that the referenced .ini file goes on to include.
    ;;
    ;; For instance, if you chose "config-include/Standalone.ini" then you will need to copy
    ;; "config-include/StandaloneCommon.ini.example" to "config-include/StandaloneCommon.ini" before
    ;; editing it to set the database and backend services that OpenSim will use.
    ;;
    ; Include-Architecture = "config-include/Standalone.ini"
    ; Include-Architecture = "config-include/StandaloneHypergrid.ini"
    Include-Architecture = "config-include/Grid.ini"
    ; Include-Architecture = "config-include/GridHypergrid.ini"
    ; Include-Architecture = "config-include/SimianGrid.ini"
    ; Include-Architecture = "config-include/HyperSimianGrid.ini"
```

APÊNDICE C – Script de geração dos NPC's

```
// :CATEGORY:NPC
// :NAME:All In One NPC Recorder and Player
// :AUTHOR:Ferd Frederix
// :KEYWORDS:
// :CREATED:2013-09-08 18:27:47
// :EDITED:2014-09-08
// :ID:27
// :NUM:1612
// :REV:1.9
// :WORLD:OpenSim
// :DESCRIPTION:
// All in one NPC recorder player.
// Supports both absolute and relative paths and many new commands
// Add animations named "Fly, Walk, Stand and Run"
// Click Prim to use.
// Should be worn as a HUD to record.
// Put it on the ground and click Sensor or Start NPC when done.
// :CODE:
// Rev 1.6 5-24-2014

// Rev 1.1 10-2-2014 @Sit did not work.  Minor tweaks to casting for lslEditor
// Rev 1.2 10-14-2014 @ sit had wrong type.
// Rev 1.3 relative movement fixed for @fly
// Rev 1.4 4-3-2014 allow anyone to use this, non owners and non group members can only start and stop.
// Rev 1.5 5-17-2014 set sensor to auto start on reboot of sim
// Rev 1.6 5-24-2014 move menu so you can get it by touching, removed many of the KeyValues to RAM for efficiency
// Rev 1.7 CHANGED_REGION_RESTART, not CHANGED_REGION_START (Opensim difference)
```

```
// Rev 1.8 tuned up Kill NPC, added more flexible upgrader
// Rev 1.9 Start NPC turns off Sensor
//*****************************************************************//

// Instructions on how to use this is at http://www.free-lsl-
scripts.com/opensim/posts/NPC/
// This is an OpenSim-only script.
// Author: Ferd Frederix


////////////////////////////////////////////////////////////////////////////////
//    Original code was Copyright (C) 2013 Wizardry and Steamworks - License:
GNU GPLv3    //
////////////////////////////////////////////////////////////////////////////////
//  Please see: http://www.gnu.org/licenses/gpl.html for legal details,          //
//  rights of fair usage, the disclaimer and warranty conditions.                //
////////////////////////////////////////////////////////////////////////////////
// The original NPC controller was from http://was.fm/opensim:npc
// Extensive additions and bug fixes by Fred Beckhusem, aka Ferd Frederix,
fred@mitsi.com
// llSensor had two params swapped
// @Wander would wander where it had rezzed, not where it was.
// There was no 'no_sensor' event in sit, so if a @sit failed, the NPC got stuck
// The animation and walks always stopped old, then started new.  It should be
start new, then stop old so the default stand would be suppressed.
// New code:
// Merged with new Route recorder and notecard writer
// If the NPC failed to reach a destination it never moved on. Added WAIT global to
tune this
// Exposed many tunable variables and ported the code to LSLEditor.
// Added floating point to times in notecard.
```

```
// Added @sound, @randsound, @whisper, @shout, and @cmd controls.
//
// notecards integers are not floats for better control
//
// Link Messages may be used to perform external control by injecting
@commands into the stream of actions
// Example:
// To chat something, such as with a chat robot
//  llMessageLinked(LINK_SET,0,"@npc_say=Hello","");

// This script assumes that NPCs and OSSl scripting is enabled in the OpenSim
configuration.
// In order to enable them, the following changes must be made in the OpenSim.ini
configuration file:
//
// ; Turn on OSSL
// AllowOSFunctions = true
// OSFunctionThreatLevel = Severe

//[NPC]
//    ;# {Enabled} {} {Enable Non Player Character (NPC) facilities} {true false}
//    Enabled = true
//
// and then the server has to be restarted.


// Commands: All commands begin with an @ sign.  All other lines are ignored
// @commands may have optional parameters.  The syntax is always:
//  @cmd=parm1|parm2
```

// NaN in the table below meand Not a Number. This means there is no parameter

//Command First Parameter Second Parameter Description
//@spawn name location (vector) Rezzes an NPC with name at a location.
//@walk destination (vector) NaN Makes the NPC walk to destination.
//@fly destination (vector) NaN Makes the NPC fly to destination.
//@land destination (vector) NaN Makes the NPC land at destination.
//@say string NaN Makes the NPC speak a phrase.
//@whisper string NaN Makes the NPC whisper a phrase.
//@shout string NaN Makes the NPC shout a phrase.
//@pause seconds (float) NaN Makes the NPC wait for a multiple of seconds.
//@wander radius (float) cycles (integer) Makes the NPC wander in radius, for cycles seconds.
//@delete NaN NaN Removes the NPC. Rerquires a link message of @npc_start to continue
//@npc_start NaN NaN Starts the NPC at the beginning
//@animate animation (string) time (float) Makes the NPC trigger the animation animation for time seconds.
//@goto label (string) NaN Jump to the label label in the script.
//@rotate degrees (float) NaN Rotate the NPC degrees around the Z axis.

```
//@sit        primitive name          NaN                 Sit on a primitive with a given
name.
//@stand      NaN                     NaN                 If sitting on a primitive, stand up.
//@sound      sound_name              NaN                 plays a sound from
inventory
//@randsound  NaN                     NaN                 Plays a random sound from
inventory
//@cmd        channel (integer)       string              Says string on channel, for
controlling external gadgets

//@stop       NaN                     NaN                 Halts the NPC script indefinitely.
Can be started with a link message
//@go         NaN                     NaN                 Continues on next notecard line,
for use in link messages
//key id3 = "19faa07d-4482-4d73-945e-3e3e898d6774";
key id3 = "4258b4ae-5bc0-4e0a-8e7c-a14fea5fe151";
////////////////////////////////////////////////////////
//                DEBUG                               //
////////////////////////////////////////////////////////
integer debug = FALSE;        // set to TRUE or FALSE for debug chat on various
actions
integer Editor = FALSE;       // set to to TRUE to working in  LSLEditor, FALSE for
in-world.
                              // you must also include the NPC commands found in the other
script since LSLEditor does not support OpenSim
integer iTitleText = FALSE;   // set to TRUE to see debug info in text above the
controller

////////////////////////////////////////////////////////
//                TUNABLE CONFIGURATION               //
////////////////////////////////////////////////////////
```

```
integer   allowUsers = TRUE;   // If true, any user can get a Start NPC and Stop NPC menu.  Only groups and owners canb get all commands if TRUE, or FALSE
float     MAXDIST = 1.0;       // how close a NPC has to get to a dest pos to continue to next state. Do not lower this too much, as it may miss the target
float     TIMER = 0.5;         // how often the system checks the distance traveled. Fastest you can go is 0.5 seconds
integer   WANDERRAND = TRUE;   // set to TRUE and they will pause during wanders a random number of seconds
float     WANDERTIME = 60.0;   // how long they stand after each @wander,if WANDERRAND is FALSE. If WANDERRAND is  TRUE, this is the max time
integer   WAIT = 10;           // wait for this number of seconds for the NPC to reach a destination (for safety). If it fails to reach a target, it will move on after this time.
float     RANGE = 96.0;        // 1 to 96.0 meters  - anyone this close to the controller will start NPCS if  Sensor button is clicked
float     REZTIME = 10.0;      // wait this lng for NPC to rez in, then start the process
string    STAND = "Stand";     // the name of the default Stand animation
string    WALK = "Walk";       // the name of the default Walk animation
string    FLY = "Fly";         // the name of the default Fly animation
string    RUN = "Run";         // the name of the default Run animation
string    LAND = "Land";       // the name of the default land animation ( for birds only)
float     OffsetZ = 0.5;       // appear 0.5 meter above ground, this is added to all destinations to keep them from sinking in.  For fun, make this large and watch them fall out of the sky


// DESCRIPTIONS FIELDS HAVE TO SURVIVE A RESET
//  These vars are stored  by saving them with KeyValueSet
// "pr" is a 0 if it is set for Owner Only, 1 for Group control
// "se" is "on" if Started
// "co" = "R" or "A" for relative or absolute addressing mode
```

```
// "key" = NPC key

// These Globals used to be stored in description.   Moved to RAM in V1.6
float RAMPause;          // @pause param
float RAMwd ;            // @wander distance
integer RAMwc;           // @wander count
float RAMrot;            //  @rotate
string RAMsit;           // @sit primname
string RAManimationName; // @animate animation (string) time (float)
float RAManimationTime;

// other globals section
integer iChannel;        // a listen channel, randomly assigned
integer iHandle;         // the handle to it



// NPC controls
vector newDest ;                // tmp storage for the walks
integer iWaitCounter ;          // wait for this number of seconds for the NPC to reach
a desrtination
string sNPCName;                // the name of the NPC that may be in world. So we
can remove it.
integer bNPC_STOP = FALSE;      // boolean to reuse a listener
integer bForget = FALSE;        // set to TRUE by link messages so we do not
remember them
float  fTimerVal ;              // how long we wait when wandering (calculated)

// OS_NPC_CREATOR_OWNED will create an 'owned' NPC that will only respond
to osNpc* commands issued from scripts that have the same owner as the one that
created the NPC.
```

```
// OS_NPC_NOT_OWNED will create an 'unowned' NPC that will respond to any script that has OSSL permissions to call osNpc* commands.
integer  NPCOptions = OS_NPC_CREATOR_OWNED;    // only yhe owner of this box can control this NPC.

integer walkstate = 0;  // helps us reshare the walk state for run, fly and land - a bit of a hack, but it saves RAM. Has to be done this way because some bits of NPCWalkOption are asserted as 0

integer NPCWalkOption;   // Some notes for what happens to NPCWalkOption:
// OS_NPC_FLY - Fly the avatar to the given position. The avatar will not land unless the OS_NPC_LAND_AT_TARGET option is also given.
// OS_NPC_NO_FLY - Do not fly to the target. The NPC will attempt to walk to the location. If it's up in the air then the avatar will keep bouncing hopeless until another move target is given or the move is stopped
//OS_NPC_LAND_AT_TARGET - If given and the avatar is flying, then it will land when it reaches the target. If OS_NPC_NO_FLY is given then this option has no effect.
// OS_NPC_RUNNING - if given, NPC avatar moves at running/fast flying speed, otherwise moves at walking/slow flying speed.

// menus
integer showMenu = FALSE;       // when we switch states, we need to bring up a menu
list lAtButtons = ["Menu","-", "More",         "@run","@walk","@fly", "@land","@wander","@sit",  "@stand","@animate","@rotate"];
list lMenu2 = ["<<", "@comment", "@stop",      "@say","@whisper","@shout", "@sound","@randsound", "-", "@cmd", "@pause", "@delete" ];
string sCommand;  // place to store a command for two-prompted ones
string sParam2;   // place to store a prompt for two-prompted ones
string priPub = "Owner Only";    // Private or Group
```

```
key kUserKey;        // the person who is controlling the avatar, not the Owner

// the command lists
list lCommands;  // commands are stored here
list lNPCScript; // Storage for the NPC script.
string npcAction; // Storage for the next action. @cmd=0|hello, this becomes
@cmd
string npcParams; // Storage for the param, @cmd=0|hello, this becomes 0|hello

// misc vars
string sNotecard; // commands are stored here temporarily for dumping
vector  vWanderPos; // a place to wander to
string lastANIM ;   // last animation run
// Sensor
integer avatarPresent = TRUE;   // Sensor sets this flag when people are within
Range.
integer Sensor;              // set to true if we are running a Sensor for avatars

// Coordinate control
vector vInitialPos ; // Vector that will be filled by the script with the initial starting
position in region coordinates.
vector vDestPos = ZERO_VECTOR; // Storage for destination position.
string relAbs = "Absolute";    // absolute vs relative positioning

////////////////////////////////////////////////////////////////////////////
//                          FUNCTIONS                          //
////////////////////////////////////////////////////////////////////////////


// DEBUG(string) will chat a string or display it as hovertext if debug == TRUE
DEBUG(string str)
```

```
{
    if (debug)
        llOwnerSay( str);                   // Send the owner debug info so you can chase
NPCS
    if (iTitleText)
    {
        llSleep(0.1);
        llSetText(str,<1.0,1.0,1.0>,1.0);    // show hovertext
    }
}


// common subroutines



// return TRUE if the avatar is owner when private is set, or TRUE if the avatar is in
th same group and GROUP is set.
integer checkPerms() {

    integer group = (integer) KeyValueGet("pr");
    if (! group)
        priPub = "Owner Only";
    else
        priPub = "Group";


    if (llDetectedKey(0) == llGetOwner()){
        kUserKey = llDetectedKey(0);
        return TRUE;
    }
```

```
    if ( group && llDetectedGroup(0)) {
        kUserKey = llDetectedKey(0);
        return TRUE;
    }
    kUserKey = llDetectedKey(0);
    return FALSE;
}



NPCStart(string anim)
{
    DEBUG(" Start Anim: " + anim);
    if (llGetInventoryType(anim) == INVENTORY_ANIMATION ) {

        if (lastANIM != anim) {
            osNpcPlayAnimation(NPCKey(), anim);

            if(llStringLength(lastANIM) && llGetInventoryType(lastANIM) ==
INVENTORY_ANIMATION) {
                osNpcStopAnimation(NPCKey(), lastANIM)      ;
            }

            lastANIM = anim;
        }
    } else {
        llSay(DEBUG_CHANNEL, "No animation named " + anim);
    }
}
```

```
TimerEvent(float timesent)
{
    //DEBUG("Setting  timer: " + (string) timesent);
    llSetTimerEvent(timesent);
}



ProcessLink(string str)
{
    // DEBUG("Processing exern cmd : " + str);
    bForget = TRUE;    // tell the NPCProcess state to forget this command after
processing it.
    lNPCScript = llListInsertList(lNPCScript,[str],0);    // add this command to the
beginning of the list of commands
    NPCStart(STAND);
}

// Kill a NPC by Name
Kill(string param)
{

    integer count;

    list avatars = osGetAvatarList(); // Returns a strided list of the UUID, position,
and name of each avatar in the region except the owner.

    integer i;
    integer j = llGetListLength(avatars);
    for (i=0 ; i <= j; i+=3){
```

```
        string desired = llList2String(avatars,i+2);
        desired = llStringTrim(desired,STRING_TRIM);    // should not be needed but
is needed

        if (desired == param){
            vector v = llList2Vector(avatars,i+1);
            key target = llList2Key(avatars,i);    // get the UUID of the avatar
            osNpcRemove(target);
            llOwnerSay("Removed " + param+ " at  location " + (string) v);
            count++;
        }
    }
    if (count)
        llOwnerSay("Removed " + (string) count + " NPC's");
    else
        llOwnerSay("Could not locate " + param);
}


// return a String for the position we are at. Strings used as the caller wants strings
string Pos()
{
    vector where = llGetPos(); // find the box position

    where.z +=    OffsetZ;  // use the ground position + an offset

    if (Editor)
        where  = <128,128,31 + llFrand(1)>; // center of sim for editing

   // if attached the height will be too high by 1/2 the agent size
```

```
    if (llGetAttached()) {
        vector size = llGetAgentSize(llGetOwner());
        float Z = size.z;
        where.z -= Z/2;
    }

    // DEBUG("Pos= " + (string) where);
    return (string) where;
}

Expire()
{
    llOwnerSay("Menu expired");
    iHandle = 0;
    TimerEvent(0.0);
}

// setup a menu with a timer for timeouts, called by all make*()
menu()
{
    llListenRemove(iHandle);
    iChannel = llCeil(llFrand(100000) + 20000);
    iHandle = llListen(iChannel,"","","");
    TimerEvent(120.0);
}


// make a text box
makeText(string Param)
{
    menu();
```

```
    llTextBox(kUserKey, Param, iChannel);
}

// top level menu
makeMainMenu()
{
    menu();
    list buttons = ["Appearance","Recording","Save","Help","-","Erase RAM",
priPub,relAbs,"-","Stop NPC","Sensor","Start NPC"];
    llDialog(kUserKey,(string) llGetListLength(lCommands) +
"Recordings",buttons,iChannel);
}


// Rev 1.4
// top level menu for non group/ non owners
makeUserMenu()
{
    if (!allowUsers)
            return;
    menu();
    list buttons = ["Start NPC","Stop NPC"];
    llDialog(kUserKey,"Choose",buttons,iChannel);
}



// programmable menu for @commands
makeMenu(list buttons)
{
    menu();
```

```
    llDialog(kUserKey,(string) llGetListLength(lCommands) +
"Recordings",buttons,iChannel);
}



// make one or two text boxes with prompts
Text(string cmd, string p1, string p2)
{
    sCommand = cmd;
     sParam2 = "";
    if (llStringLength(p2))
        sParam2 = p2;

    makeText(p1);
}

ProcessSensor(integer n)
{
    DEBUG("Sensor:" + (string) n);
    if (Sensor && n)
        avatarPresent = TRUE;   // someone is here and we need to tell the system to
run
    else if (Sensor && !n)
        avatarPresent = FALSE;  // someone is not here and we need to tell the
system to stop
    else
        avatarPresent = TRUE;   // someone is effectivly always here
}
```

```
vector CirclePoint(float radius) {
    float x = llFrand(radius *2) - radius;        // +/- radius, randomized
    float y = llFrand(radius *2) - radius;        // +/- radius, randomized
    return <x, y, 0>;        // so this should always happen
}

string KeyValueGet(string var) {
    list dVars = llParseString2List(llGetObjectDesc(), ["&"], []);
    do {
        list data = llParseString2List(llList2String(dVars, 0), ["="], []);
        string k = llList2String(data, 0);
        if(k != var) jump continue;
        //DEBUG("got " + var + " = " +  llList2String(data, 1));
        return llList2String(data, 1);
        @continue;
        dVars = llDeleteSubList(dVars, 0, 0);
    } while(llGetListLength(dVars));
    return "";
}

KeyValueSet(string var, string val) {

    //DEBUG("set " + var + " = " + val);
    list dVars = llParseString2List(llGetObjectDesc(), ["&"], []);
    if(llGetListLength(dVars) == 0)
    {
        llSetObjectDesc(var + "=" + val);
        return;
    }
    list result = [];
    do {
```

```
        list data = llParseString2List(llList2String(dVars, 0), ["="], []);
        string k = llList2String(data, 0);
        if(k == "") jump continue;
        if(k == var && val == "") jump continue;
        if(k == var) {
            result += k + "=" + val;
            val = "";
            jump continue;
        }
        string v = llList2String(data, 1);
        if(v == "") jump continue;
        result += k + "=" + v;
        @continue;
        dVars = llDeleteSubList(dVars, 0, 0);
    } while(llGetListLength(dVars));
    if(val != "") result += var + "=" + val;
    llSetObjectDesc(llDumpList2String(result, "&"));
}


// clear RAM
Clr() {

    lCommands = [];
    llOwnerSay("RAM Memory cleared. Notecards, if any, are not modified.");
    makeMainMenu();
}

integer checkNoteCards()
{
    // Check that they have saved an Appeaance and Path notecard
```

```
    integer num = llGetInventoryNumber(INVENTORY_NOTECARD);    // how many
notecards overall

    integer i;
    integer count;
    for (; i < num; i++){
        if (llGetInventoryName(INVENTORY_NOTECARD,i) == "Path")
            count++;
        if (llGetInventoryName(INVENTORY_NOTECARD,i) == "Appearance")
            count++;
    }
    // if we have both, run the NPC
    return count;
}
// saves a few bytes per call
key NPCKey()
{
    return KeyValueGet("key");
}

// Notes:
// No llResetScript() used so we can retain memory between rezzes and sim
restarts.  NPC info and stateful info is held in the Description
//


// This state is the first main menu
default
{
    state_entry()
    {
```

```
llSetText("",<1,1,1>,1.0);  // clr all hovertext- we may not be using it.

// delete all NPC*scripts except myself
integer i;
integer j = llGetInventoryNumber(INVENTORY_SCRIPT);
for (i = 0; i < j; i++) {
    string name = llGetInventoryName(INVENTORY_SCRIPT,i);
    string match = llGetSubString(name,0,2);
    if (match == "NPC" && llGetScriptName() != name)
    {
        llRemoveInventory(name);
        llOwnerSay("Upgraded");
    }

}


string rA = KeyValueGet("co"); // Get the remembered menu setting for Abs
Vs Relative
if (rA == "A")
    relAbs = "Absolute";
else  if (rA == "R")
    relAbs = "Relative";
else
    relAbs = "Absolute";


if (showMenu) {
    makeMainMenu();
    return;
```

```
    }

    // reenable NPC if sensor is on.
    if ("on" == KeyValueGet("se"))
    {
        ProcessSensor(1);       // fake 1 avatar to get it rezzed
        Sensor = TRUE;        // we need to scan for avatars
        state NPCGo;
    }
     // reenable NPC if On is on.
    if ("1" == KeyValueGet("On"))
    {
        ProcessSensor(1);       // fake 1 avatar to get it rezzed
        Sensor = FALSE;        // no need to scan for avatars
        state NPCGo;
    }
     llListen(-220, "", NULL_KEY, "");

}


// touch_start(integer n) {          // if touched, make a menu
  //   if (checkPerms())
    //     makeMainMenu();
    // else
      //   makeUserMenu();
// }

 // no changed event needed

 // menu listener
```

```
listen(integer iChannel, string name, key id, string message) {
    TimerEvent(0.0);    /// kill the menu expiration timer

    if (message == "Stop NPC")
    {
        KeyValueSet("On","0");   // On is Off
        if (llStringLength(sNPCName)){
            Kill(sNPCName);
            sNPCName = "";
        } else {
            bNPC_STOP = TRUE;
            makeText("Enter name of an NPC to stop");
        }
    }
    else if (message == "Erase RAM"){
        Clr();
    }
    else if (message == "Relative"){
        relAbs = "Absolute";
        KeyValueSet("co","A");   // remember coordinates = A
        Clr();
    }
    else if (message == "Absolute"){
        relAbs = "Relative";
        KeyValueSet("co","R");   // remember coordinates = R
        Clr();
    }
    else if (message == "Recording"){
        state Commands;        // show them the recording menu
    }
    else if (message == "Owner Only") {
```

```
            priPub = "Group";
            KeyValueSet("pr","1");

            llOwnerSay("Group members have control");
            makeMainMenu();
        }
        else if (message == "Group") {
            priPub = "Owner Only";
            KeyValueSet("pr","0");
            llOwnerSay("Only you have control");
            makeMainMenu();
        }
        else if (message == "Sensor") {
            KeyValueSet("On","1");   // On is Off
            integer count = checkNoteCards();

            if (count >= 2)  {
                KeyValueSet("se", "on");
                ProcessSensor(1);       // fake 1 avatar to get it rezzed
                Sensor = TRUE;        // we need to scan for avatars
                state NPCGo;
            }

            // lslEditor does not handle the above, so I hack it in
            if (Editor) {
                Sensor = TRUE;        // we need to scan for avatars
                state NPCGo;
            }

            llOwnerSay("You have not saved a recording and/or appearance, so you
cannot start a NPC");
```

```
            makeMainMenu();
        }
        else if (message == "Appearance")  {
            llRemoveInventory("Appearance");            // delete the notecard
            osAgentSaveAppearance(kUserKey, "Appearance");    // make the ntecard
            llOwnerSay("Your Appearance has been recorded in notecard
'Appearance'");
            makeMainMenu();
        }
        else if (message == "Save") {
            if (llGetListLength(lCommands) == 0) {
                llOwnerSay("Nothing recorded, you need to make some Recodings
first");
                makeMainMenu();
                return;
            }
            state Save;
        }
        else if (message == "Help"){
            llLoadURL(kUserKey,"Click to view help","http://www.free-lsl-
scripts.com/opensim/posts/NPC/");
            makeMainMenu();
        }

        else if (bNPC_STOP){
            bNPC_STOP = FALSE;
            Kill(message);
        }
        else  {
            KeyValueSet("On","1");   // On is On
            KeyValueSet("se", "off");
```

```
            integer count = checkNoteCards();
            if (Editor) state NPCGo;
            if (count == 2)
                state NPCGo;

            llOwnerSay("You have not saved a either recording or and appearance, so
we cannot start a NPC");

        }
    }
    timer(){
        Expire();
    }
}


// This state is used to save a Path notecard
state Save
{
    state_entry(){
        makeText("Stand where you want the NPC to appear, and enter the NPC
Name");
    }
    listen(integer iChannel, string name, key id, string message) {
        TimerEvent(0.0);    /// kill the menu expiration timer

        sNPCName = message; // in case we need to kill it.
        vector  vDest = (vector)  Pos();

        if (relAbs == "Relative")
        {
```

```
            vDest  -= llGetPos();    // just an offset for relative
        }
        sNotecard = "@spawn=" + message + "|" +  (string) vDest  + "\n";
        integer i;
        integer j = llGetListLength(lCommands);
        for (; i < j; i++){
            // get the command to save to the notecard
            string line = llList2String(lCommands,i);
            if (relAbs == "Absolute") {
                sNotecard += line;    // add the un-modified string to the notecard
            } else {
                // since we have to record absolute coords since we do not know where
the box goes until they press Save,
                // we process the absolute to relative conversion for walks here
                list parts = llParseString2List(line,["="],[]); //get the @command

                if (llList2String(parts,0) == "@walk") {
                    vector vec = (vector) llList2String(parts,1) - llGetPos();
                    sNotecard += "@walk=" + (string) vec + "\n";
                }
                else if (llList2String(parts,0) == "@fly") {
                    vector vec = (vector) llList2String(parts,1) - llGetPos();
                    sNotecard += "@fly=" + (string) vec + "\n";
                }
                else if (llList2String(parts,0) == "@run") {
                    vector vec = (vector) llList2String(parts,1) - llGetPos();
                    sNotecard += "@run=" + (string) vec + "\n";
                }
                else if (llList2String(parts,0) == "@land") {
                    vector vec = (vector) llList2String(parts,1) - llGetPos();
                    sNotecard += "@land=" + (string) vec + "\n";
```

```
            }
            else {
                sNotecard += line;    // add the un-modified string to the notecard
            }
          }
        }
        llRemoveInventory("Path");        // delete the old notecard
        osMakeNotecard("Path",sNotecard); // Makes the notecard.
        llOwnerSay("'Path' notecard has been written");
        state default;
    }
    timer(){
       Expire();
    }

}

// This state is used to record the path for the NPC
// Each command can take 0, 1, or 2 params
state Commands
{
    state_entry()    {
        makeMenu(lAtButtons);
    }

    on_rez(integer p)    {
       showMenu= TRUE;
        state default;
    }
    touch_start(integer n){
        if (checkPerms())
```

```
        makeMenu(lAtButtons);
    else
        makeUserMenu();

}

listen(integer iChannel, string name, key id, string message)
{
    TimerEvent(0.0);    /// kill the menu expiration timer

    if (message == "Menu"){
        showMenu= TRUE;
        state default;
    }
    else if (message == "More"){
        makeMenu(lMenu2);
    }
    else if (message == "<<") {
        makeMenu(lAtButtons);
    }
    else if (message == "@comment"){
        Text("# ","Enter a comment","");
    }
    else if (message == "@stop"){
        lCommands += "@stop"+  "\n";
        makeMenu(lAtButtons);
    }
    else if (message == "@run"){
        lCommands += "@run=" + Pos() + "\n";
        llOwnerSay("Recorded position: " + Pos());
        makeMenu(lAtButtons);
```

```
}
else if (message == "@fly"){
    lCommands += "@fly=" + Pos() + "\n";
    llOwnerSay("Recorded position: " + Pos());
    makeMenu(lAtButtons);
}
else if (message == "@land"){
    lCommands += "@land=" + Pos() + "\n";
    llOwnerSay("Recorded position: " + Pos());
    makeMenu(lAtButtons);
}
else if (message == "@walk") {
    lCommands += "@walk=" + Pos() + "\n";
    llOwnerSay("Recorded position: " + Pos());
    makeMenu(lAtButtons);
}
else if (message == "@stop"){
    lCommands += "@stop"+  "\n";
    makeMenu(lAtButtons);
}
else if (message == "@sound"){
    Text("@sound=","Enter a sound name or UUID to trigger","");
}
else if (message == "@randsound"){
    lCommands += "@randsound"+  "\n";
    makeMenu(lAtButtons);
}
else if (message == "@say") {
    Text("@say=","Enter what the NPC will say","");
}
else if (message == "@whisper"){
```

```
        Text("@whisper=","Enter what the NPC will whisper","");
    }
    else if (message == "@shout"){
        Text("@shout=","Enter what the NPC will shout","");
    }
    else if (message == "@wander") {
        Text("@wander=","Enter radius to wander","Enter number of wanders");
    }
    else if (message == "@pause") {
        Text("@pause=","Enter time to pause","");
    }
    else if (message == "@rotate") {
        Text("@rotate=","Enter degrees to rotate","");
    }
    else if (message == "@sit"){
        Text("@sit=","Enter name of object to sit on","");
    }
    else if (message == "@cmd"){
        Text("@cmd=","Enter cjhannel to speak on","Enter text to speak");
    }
    else if (message == "@stand"){
        lCommands += "@stand\n";
        llOwnerSay("Stand Recorded");
        makeMenu(lAtButtons);
    }
    else if (message == "@animate"){
        Text("@animate=","Enter animation name to play","Enter time to play the
animation");
    }
    else  if (! llStringLength(sParam2)) {
        lCommands +=  sCommand + message + "\n";
```

```
            llOwnerSay("Recorded");
            makeMenu(lAtButtons);
        }
        else if (llStringLength(sParam2)){
            sCommand = sCommand + message + "|";
            llOwnerSay("Recorded");
            makeText(sParam2);
            sParam2 = "";
        }
    }
    timer() {
       Expire();
}




}


// This state will create an NPC in world
state NPCGo {
    state_entry() {
        // DEBUG("NPCGo");
        ProcessSensor(1);       // assert that someone is home so we can get rezzed.
        osNpcRemove(NPCKey());
        TimerEvent(5);
    }
    timer() {
        lNPCScript = llParseString2List(osGetNotecard("Path"), ["\n"], []);
        if(llGetListLength(lNPCScript) == 0) {
            llSay(DEBUG_CHANNEL, "No Path notecard found.");
```

```
            TimerEvent(0.0);
            return;
        }
        state ProcessNPCLine;
    }
    changed(integer change) {
        if(change & CHANGED_REGION_RESTART)
            state ProcessNPCLine;
    }
    on_rez(integer num) {
        llResetScript();
    }
    state_exit() {
        TimerEvent(0.0);
    }
}


// This state loops over the notecard, processing each command
state ProcessNPCLine
{
    state_entry()
    {
        // DEBUG("ProcessNPCLine");
        @ignore;

        string next = llList2String(lNPCScript, 0);        // get the next command
        // DEBUG("Cmd:" + next);
        lNPCScript = llDeleteSubList(lNPCScript, 0, 0);      // delete it
        if (! bForget) {
```

```
            lNPCScript += next;                         // put it on the end unless we are
told to forget it from a Link Message
            bForget = FALSE;
        }
        if(llGetSubString(next, 0, 0) != "@") jump ignore;    // ignore non-@ commands
        list data  = llParseString2List(next, ["="], []);
        npcAction = llToLower(llStringTrim(llList2String(data, 0), STRING_TRIM));
        npcParams = llStringTrim(llList2String(data, 1), STRING_TRIM);

        @commands;

        if (! avatarPresent){
            state  nobodyHome;
        }
        if(npcAction == "@spawn") {
            // DEBUG("Spawning");
            integer lastIdx = llGetListLength(lNPCScript)-1;
            lNPCScript = llDeleteSubList(lNPCScript, lastIdx, lastIdx);        // remove
spawn commands, we do them only once
            list spawnData = llParseString2List(npcParams, ["|"], []);
            sNPCName =llList2String(spawnData, 0);    // V 1.6 name in RAM

            list spawnDest = llParseString2List(llList2String(spawnData, 1), ["<", ",",
">"], []);
            vInitialPos.x = llList2Float(spawnDest, 0);
            vInitialPos.y = llList2Float(spawnDest, 1);
            vInitialPos.z = llList2Float(spawnDest, 2);
            state spawn;
        }
        if(npcAction == "@stop") {
            state stop;
```

```
}
if(npcAction == "@goto") {
    // DEBUG("goto");
    integer lastIdx = llGetListLength(lNPCScript)-1;
    lNPCScript = llDeleteSubList(lNPCScript, lastIdx, lastIdx);
    // Wind commands till goto label.
    @wind;
    string next1 = llList2String(lNPCScript, 0);
    lNPCScript = llDeleteSubList(lNPCScript, 0, 0);
    lNPCScript += next1;
    if(next1 != npcParams) jump wind;
    // Wind the label too.
    next1 = llList2String(lNPCScript, 0);
    lNPCScript = llDeleteSubList(lNPCScript, 0, 0);
    lNPCScript += next1;
    // Get next command.
    list data1  = llParseString2List(next1, ["="], []);
    npcAction = llToLower(llStringTrim(llList2String(data1, 0), STRING_TRIM));
    npcParams = llStringTrim(llList2String(data1, 1), STRING_TRIM);
    // Reschedule.
    jump commands;
}

if(npcAction == "@sound") {
    // DEBUG("sound");
    llTriggerSound(npcParams,1.0);
    jump ignore;      // process next line
}
if(npcAction == "@randsound") {
    // DEBUG("random sound");
    integer N = llGetInventoryNumber(INVENTORY_SOUND);
```

```
    integer rand = llCeil(llFrand(N)) -1;    // pick a random sound
    string toPlay = llGetInventoryName(INVENTORY_SOUND,rand);
    llTriggerSound(toPlay,1.0);
    jump ignore;      // process next line
}

if(npcAction == "@walk") {
    list dest = llParseString2List(npcParams, ["<", ",", ">"], []);
    vDestPos.x = llList2Float(dest, 0);
    vDestPos.y = llList2Float(dest, 1);
    vDestPos.z = llList2Float(dest, 2);

    if (vDestPos == ZERO_VECTOR) {
        llSay(DEBUG_CHANNEL,"Bad (zeros) position for @walk");
        state ProcessNPCLine;
    }
    walkstate = 1;//  walking
    NPCWalkOption = OS_NPC_NO_FLY ;
    state walk;
}

if(npcAction == "@fly") {
    list dest = llParseString2List(npcParams, ["<", ",", ">"], []);
    vDestPos.x = llList2Float(dest, 0);
    vDestPos.y = llList2Float(dest, 1);
    vDestPos.z = llList2Float(dest, 2);

    if (vDestPos == ZERO_VECTOR) {
        llSay(DEBUG_CHANNEL,"Bad (zeros) position for @fly");
        state ProcessNPCLine;
    }
```

```
    walkstate = 2;//  flying
    NPCWalkOption = OS_NPC_FLY ;
    state walk;
}

if(npcAction == "@run") {
   list dest = llParseString2List(npcParams, ["<", ",", ">"], []);
    vDestPos.x = llList2Float(dest, 0);
    vDestPos.y = llList2Float(dest, 1);
    vDestPos.z = llList2Float(dest, 2);

    if (vDestPos == ZERO_VECTOR) {
        llSay(DEBUG_CHANNEL,"Bad (zeros) position for @walk");
        state ProcessNPCLine;
    }
    walkstate = 3;//  running
    NPCWalkOption = OS_NPC_NO_FLY | OS_NPC_RUNNING;
    state walk;
}

if(npcAction == "@land") {
    list dest = llParseString2List(npcParams, ["<", ",", ">"], []);
    vDestPos.x = llList2Float(dest, 0);
    vDestPos.y = llList2Float(dest, 1);
    vDestPos.z = llList2Float(dest, 2);

    if (vDestPos == ZERO_VECTOR) {
        llSay(DEBUG_CHANNEL,"Bad (zeros) position for @walk");
        state ProcessNPCLine;
    }
```

```
    walkstate = 4;//  landing


    NPCWalkOption= OS_NPC_FLY | OS_NPC_LAND_AT_TARGET ;
    state walk;
}



// chat commands
// speak in white text

if(npcAction == "@say") {
    // DEBUG("say");
    osNpcSay(NPCKey(),0, npcParams);
    jump ignore;    // process next line
}
if(npcAction == "@shout") {
    // DEBUG("shout");
    osNpcShout(NPCKey(),0, npcParams);
    jump ignore;    // process next line
}
if(npcAction == "@whisper") {
    // DEBUG("whisper");
    osNpcWhisper(NPCKey(),0, npcParams);
    jump ignore;    // process next line
}
// speak a command on a channel, so you can open doors and control stuff.
if(npcAction == "@cmd") {
    // DEBUG("cmd");
```

```
    list dataToSpeak = llParseString2List(npcParams, ["|"], []);
    integer iChannel = (integer) llList2String(dataToSpeak,0);
    string stringToSpeak = llList2String(dataToSpeak,1);
    llRegionSay(iChannel, stringToSpeak);     // V 1.2

    jump ignore;    // process next line
}
// stop everything
if(npcAction == "@pause") {
    // DEBUG("pause");
    RAMPause = (float) npcParams;
    state pause;
}
if(npcAction == "@wander") {
    // DEBUG("wander");
    list wanderData = llParseString2List(npcParams, ["|"], []);
    RAMwd = (float) llList2String(wanderData, 0);
    RAMwc = (integer) llList2String(wanderData, 1);

    vDestPos = osNpcGetPos(NPCKey());        // set the wander start
    DEBUG("Starting at " + (string) vDestPos);
    state wander;
}
if(npcAction == "@rotate") {
    // DEBUG("rotate");
    RAMrot = (float) npcParams;
    state rotate;
}
if(npcAction == "@sit") {
    // DEBUG("sit");
    RAMsit= npcParams;
```

```
            state sit;
        }
        if(npcAction == "@stand") {
            // DEBUG("stand");
            state stand;
        }
        if(npcAction == "@delete") {
            state delete;
        }
        if(npcAction == "@animate") {
            // DEBUG("animate");
            list animateData = llParseString2List(npcParams, ["|"], []);
            RAManimationName = llList2String(animateData, 0);
            RAManimationTime = (float) llList2String(animateData, 1);
            state animate;
        }
        llSay(DEBUG_CHANNEL, "ERROR: Unrecognized script line: " + npcAction +
"=" + npcParams);
        jump ignore;

    }
    changed(integer change) {
        if(change & CHANGED_REGION_RESTART)
            state NPCGo;
    }
    on_rez(integer num) {
        llResetScript();
    }
    link_message(integer sender, integer num, string str, key id) {
        ProcessLink(str);
        state ProcessNPCLine;
```

```
    }
    // if touched by owner while running code, make a menu
    touch_start(integer n) {

        if (checkPerms()) {
            TimerEvent(0); // stop the NPC from ticking
            showMenu = TRUE;
            state default;
        }  else {
            makeUserMenu();
        }

    }
}


state nobodyHome
{
    state_entry()    {
        // DEBUG("Removing NPC");
        osNpcRemove(NPCKey());
        llSensorRepeat("","",AGENT,RANGE,TWO_PI, 5);
    }
    sensor(integer n)    {
        llSensorRemove();
        state NPCGo;
    }
    // if touched by owner while running code, make a menu
    touch_start(integer n) {
        if (checkPerms()) {
            TimerEvent(0); // stop the NPC from ticking
```

```
            showMenu = TRUE;
            state default;
        } else {
            makeUserMenu();
        }

    }
}


state spawn
{
    state_entry() {
        // DEBUG("state spawn");
        list name = llParseString2List(sNPCName, [" "], []);
       // notecard is stored as offsets from this box with relative addressing.  Convert
to absolute
        if (relAbs == "Relative"){
            vInitialPos += llGetPos();

        }

        // DEBUG("rez:" + (string) vInitialPos);
        llSleep(llFrand(5));
        llRegionSayTo(id3, -220, "GO!");
        KeyValueSet("key", osNpcCreate(llList2String(name, 0), llList2String(name,
1), vInitialPos, "Appearance", NPCOptions));    // no
OS_NPC_SENSE_AS_AGENT allowed due to llSensor Use
        osNpcLoadAppearance(NPCKey(), "Appearance");
        TimerEvent(REZTIME);
        NPCStart(STAND);
```

```
    }
    link_message(integer sender, integer num, string str, key id) {
        ProcessLink(str);
        state ProcessNPCLine;
    }
    timer() {
        state ProcessNPCLine;
    }
    changed(integer change) {
        if(change & CHANGED_REGION_RESTART)
            state NPCGo;
    }
    on_rez(integer num) {
        llResetScript();
    }
    state_exit(){
        TimerEvent(0.0);
    }
    // if touched by owner while running code, make a menu
    touch_start(integer n) {
        if (checkPerms()) {
            TimerEvent(0); // stop the NPC from ticking
            showMenu = TRUE;
            state default;
        }else {
            makeUserMenu();
        }

    }
}
```

```
state rotate {
    state_entry() {
        // DEBUG("state rotate");
        osNpcSetRot(NPCKey(), llEuler2Rot(<0,0,RAMrot> * DEG_TO_RAD));
        TimerEvent(TIMER);
    }
    link_message(integer sender, integer num, string str, key id) {
        ProcessLink(str);
        state ProcessNPCLine;
    }
    timer() {
        state ProcessNPCLine;
    }
    changed(integer change) {
        if(change & CHANGED_REGION_RESTART)
            state NPCGo;
    }
    on_rez(integer num) {
        llResetScript();
    }
    state_exit() {
        TimerEvent(0.0);
    }
    // if touched by owner while running code, make a menu
    touch_start(integer n) {
        if (checkPerms()) {
            TimerEvent(0); // stop the NPC from ticking
            showMenu = TRUE;
            state default;
        }else {
```

```
            makeUserMenu();
        }
    }
}

state sit {
    state_entry() {
        // DEBUG ("state sit");
        llSensorRepeat(RAMsit, "", PASSIVE|ACTIVE,  96, TWO_PI, 1);
    }
    sensor(integer num) {
        llSensorRemove();
        osNpcSit(NPCKey(), llDetectedKey(0), OS_NPC_SIT_NOW); //V1.2
        TimerEvent(TIMER);
    }
    no_sensor(){
        state ProcessNPCLine;

    }
    timer() {
        state ProcessNPCLine;
    }
    link_message(integer sender, integer num, string str, key id) {
        ProcessLink(str);
        state ProcessNPCLine;
    }
    changed(integer change) {
        if(change & CHANGED_REGION_RESTART)
            state NPCGo;
    }
    on_rez(integer num) {
```

```
        llResetScript();
    }
    // if touched by owner while running code, make a menu
    touch_start(integer n) {
        if (checkPerms()) {
            TimerEvent(0); // stop the NPC from ticking
            showMenu = TRUE;
            state default;
        } else {
            makeUserMenu();
        }

    }


    state_exit(){
        TimerEvent(0.0);
    }


}

state stand {
    state_entry() {
        // DEBUG("state stand");
        osNpcStand(NPCKey());
        state ProcessNPCLine;
    }
    changed(integer change) {
        if(change & CHANGED_REGION_RESTART)
            state NPCGo;
```

```
    }
    // if touched by owner while running code, make a menu
    touch_start(integer n) {
        if (checkPerms()) {
            TimerEvent(0); // stop the NPC from ticking
            showMenu = TRUE;
            state default;
        } else {
            makeUserMenu();
        }

    }


    on_rez(integer num) {
        llResetScript();
    }

}

state animate {
    state_entry() {
        // DEBUG("state animate");
        NPCStart(RAManimationName);
        TimerEvent(RAManimationTime);
    }
    link_message(integer sender, integer num, string str, key id) {
        ProcessLink(str);
        state ProcessNPCLine;
    }
    timer() {
```

```
        NPCStart(STAND);
        state ProcessNPCLine;
    }
    changed(integer change) {
        if(change & CHANGED_REGION_RESTART)
            state NPCGo;
    }
    // if touched by owner while running code, make a menu
    touch_start(integer n) {
        if (checkPerms()) {
            TimerEvent(0); // stop the NPC from ticking
            showMenu = TRUE;
            state default;
        } else {
            makeUserMenu();
        }

    }


    on_rez(integer num) {
        llResetScript();
    }
    state_exit() {
        TimerEvent(0.0);
    }

}
state walk {
    state_entry() {
```

```
        DEBUG("NPCWalkOption = " + (string) NPCWalkOption);

        // walk, fly, run, land
       if (walkstate == 1) {
            NPCStart(WALK);
        } else if (walkstate == 2)  {
            llShout(299,"on");
            NPCStart(FLY);
        } else if (walkstate == 3) {
            NPCStart(RUN);
        } else if (walkstate == 4) {
            NPCStart(LAND);
        } else  {
            state ProcessNPCLine;
        }

        if (Sensor)  {
            // DEBUG("Sensor on");
            llSensor("","",AGENT,RANGE,TWO_PI);        // sensor survive state
switches.
        }

        newDest = vDestPos ;
        // notecard is stored as offsets from this box with relative addressing.  Convert
to absolute
        if (relAbs == "Relative"){
            newDest += llGetPos();
        }

        DEBUG("Moveto:" + (string) newDest);
        iWaitCounter = WAIT;            // wait 60 seconds to get to a destination.
```

```
    osNpcMoveToTarget(NPCKey(), newDest, NPCWalkOption);
    TimerEvent(TIMER);
}
link_message(integer sender, integer num, string str, key id) {
    ProcessLink(str);
    state ProcessNPCLine;
}
timer() {
    if (--iWaitCounter) {

        vector tDest = newDest;
        tDest.z = 0;
        vector hisDest = osNpcGetPos(NPCKey());
        hisDest.z = 0;

        if (llVecDist(hisDest, newDest) > MAXDIST)  {
            return;
        }
    }

     // walk, fly, run, land
    if (walkstate == 1) {
        NPCStart(STAND);
    } else if (walkstate == 2)  {
        // nothing
    } else if (walkstate == 3) {
        NPCStart(STAND);
    } else if (walkstate == 4) {
        llShout(299,"off");
        NPCStart(STAND);
    } else  {
```

```
        state ProcessNPCLine;
    }

    state ProcessNPCLine;
}
sensor(integer n) {
    ProcessSensor(n);
}
no_sensor(){
    ProcessSensor(0);
}
// if touched by owner while running code, make a menu
touch_start(integer n) {
    if (checkPerms()) {
        TimerEvent(0); // stop the NPC from ticking
        showMenu = TRUE;
        state default;
    } else {
        makeUserMenu();
    }

}


changed(integer change) {
    if(change & CHANGED_REGION_RESTART)
        state NPCGo;
}
on_rez(integer num) {
    llResetScript();
}
```

```
    state_exit() {
        TimerEvent(0.0);
    }

}

state wander
{
    state_entry() {
        DEBUG("state wander");
        if (Sensor)
        {
            // DEBUG("Sensor on");
            llSensor("","",AGENT,RANGE,TWO_PI);        // sensor survive state
switches.
        }

        vector point = CirclePoint(RAMwd);
        DEBUG("CirclePoint:" + (string) point);
        vWanderPos = vDestPos + point;
        DEBUG("vWanderPos:" + (string) vWanderPos);

        fTimerVal = WANDERTIME;    // default time to pause after each wander
        if (WANDERRAND)
            fTimerVal = llFrand(WANDERTIME);    // override, they want random times

        NPCStart(WALK);

        DEBUG("Wander to:" + (string) vWanderPos);

        osNpcMoveToTarget(NPCKey(), vWanderPos, NPCWalkOption);
```

```
        iWaitCounter = WAIT;            // wait 60 seconds to get to a destination.

        TimerEvent(TIMER);      // first time we wait for the short timer.
    }
    link_message(integer sender, integer num, string str, key id) {
        ProcessLink(str);
        NPCStart(STAND);
        state ProcessNPCLine;
    }
    timer() {

        if (--iWaitCounter)           // wait 60 seconds to get to a destination.
        {
            vector tDest = vWanderPos;
            tDest.z = 0;
            vector hisDest = osNpcGetPos(NPCKey());
            hisDest.z = 0;

            if (llVecDist(hisDest, tDest) > MAXDIST) return;
        }

        // see if wander counter == 0, if so, stop walking, go to stand and process next
line
        if(RAMwc == 0) {
            NPCStart(STAND);
            state ProcessNPCLine;
        }

        // one less time to wander around
        RAMwc--;
```

```
    NPCStart(STAND);
    state wanderhold;
}
sensor(integer n) {
    ProcessSensor(n);
}
no_sensor() {
    ProcessSensor(0);
}
// if touched by owner while running code, make a menu
touch_start(integer n) {
    if (checkPerms()) {
        TimerEvent(0); // stop the NPC from ticking
        showMenu = TRUE;
        state default;
    } else {
        makeUserMenu();
    }
}



changed(integer change) {
    if(change & CHANGED_REGION_RESTART)
        state NPCGo;
}
on_rez(integer num) {
    llResetScript();
}
state_exit() {
    TimerEvent(0.0);
}
```

```
}


state wanderhold
{
    state_entry(){
        // now that we have reached a wander spot, slow the timer down to the
desired value
        TimerEvent(fTimerVal);
        if (Sensor)
        {
            // DEBUG("Sensor on");
            llSensor("","",AGENT,RANGE,TWO_PI);        // sensor survive state
switches.
        }
    }
    timer()  {
        state wander;
    }
    sensor(integer n){
        ProcessSensor(n);
    }
    no_sensor(){
        ProcessSensor(0);
    }
    // if touched by owner while running code, make a menu
    touch_start(integer n) {
        if (checkPerms()) {
            TimerEvent(0); // stop the NPC from ticking
            showMenu = TRUE;
            state default;
```

```
        } else {
            makeUserMenu();
        }

    }


    changed(integer change) {
        if(change & CHANGED_REGION_RESTART)
            state NPCGo;
    }
    on_rez(integer num) {
        llResetScript();
    }
    state_exit() {
        TimerEvent(0.0);
    }


}



// @pause=10 will stand for 10 seconds
state pause {
    state_entry() {
        DEBUG("state pause");
        NPCStart(STAND);
        if (Sensor)
        {
            // DEBUG("Sensor on");
```

```
            llSensor("","",AGENT,RANGE,TWO_PI);        // sensor survive state
switches.
        }
        llSetTimerEvent(RAMPause);
    }
    link_message(integer sender, integer num, string str, key id){
        ProcessLink(str);
        state ProcessNPCLine;
    }
    timer() {
        NPCStart(STAND);
        state ProcessNPCLine;
    }
    sensor(integer n)
    {
        ProcessSensor(n);
    }
    no_sensor()
    {
        ProcessSensor(0);
    }
    // if touched by owner while running code, make a menu
    touch_start(integer n) {
        if (checkPerms()) {
            TimerEvent(0); // stop the NPC from ticking
            showMenu = TRUE;
            state default;
        } else
            makeUserMenu();

    }
```

```
    changed(integer change) {
        if(change & CHANGED_REGION_RESTART)
            state NPCGo;
    }
    on_rez(integer num) {
        llResetScript();
    }
    state_exit()
    {
        TimerEvent(0.0);
    }
}

// @stop makes the NPC stand there.  You have to linkmessage to get moving
again
state stop {
    state_entry() {
        NPCStart(STAND);
    }
    link_message(integer sender, integer num, string str, key id){
        if (str=="@go")
            state ProcessNPCLine;
        ProcessLink(str);
    }
    // if touched by owner while running code, make a menu
    touch_start(integer n) {
        if (checkPerms()) {
            TimerEvent(0); // stop the NPC from ticking
            showMenu = TRUE;
            state default;
        } else
```

```
            makeUserMenu();

    }

    changed(integer change) {
        if(change & CHANGED_REGION_RESTART)
            state NPCGo;
    }
    on_rez(integer num) {
        llResetScript();
    }
}



state delete {
    state_entry() {
        // DEBUG("state delete");
        KeyValueSet("On","0");   // On is Off
            if (llStringLength(sNPCName)){
                Kill(sNPCName);
                sNPCName = "";
            } else {
                bNPC_STOP = TRUE;
                makeText("Enter name of an NPC to stop");
            }
            llResetScript();
    }
    link_message(integer sender, integer num, string str, key id) {
        if(str == "@npc_start")
        {
```

```
            state NPCGo;
        }

    }

    // No on_rez or changed event needed, the only way out is a link message
}
```

APÊNDICE D – Questionário de avaliação da peça teatral e agentes inteligentes.

Avaliação da Peça teatral e Agentes Inteligentes

Avaliação de usabilidade

Perfil

1ª) Idade

Escolha sua faixa etária

menos de 25 anos
entre 25 e 29 anos
entre 30 e 34 anos
entre 34 e 38 anos
mais de 39 anos

2ª) Sexo

Feminino
Masculino
3ª) Qual a sua área de formação?

4ª) Você tem alguma experiência com OpenSim e Mundos Virtuais?

Sim
Não

5ª) Se sim, como foi?

Avaliação da Peça Teatral

6ª) Assistir uma peça teatral educacional foi uma experiência agradável.

Concordo plenamente
Concordo Parcialmente
Talvez
Discordo parcialmente
Discordo plenamente

7ª) A peça teatral correspondeu as suas expectativas.

Concordo plenamente
Concordo parcialmente
Talvez
Discordo parcialmente
Discordo plenamente

8ª) A mensagem educacional da peça teatral estava clara.

Concordo plenamente
Concordo parcialmente
Talvez
Discordo parcialmente
Discordo plenamente

9ª) A peça teatral estava bem montada.

Concordo plenamente
Concordo parcialmente
Talvez
Discordo parcialmente
Discordo plenamente

10ª) O espaço físico do teatro é agradável.

Concordo plenamente
Concordo parcialmente
Talvez
Discordo parcialmente
Discordo plenamente

11ª) A peça pode ser considerada uma boa estratégia de aprendizagem.

Concordo plenamente
Concordo parcialmente
Talvez
Discordo parcialmente
Discordo plenamente

12ª) A peça pode servir como ferramenta de auxílio ao ensino presencial.

Concordo plenamente
Concordo parcialmente
Talvez
Discordo parcialmente
Discordo plenamente

Caso deseje, deixe aqui seus comentários sobre a Peça Teatral.

Avaliação do Agente

13ª) O Agente Inteligente demonstrou bom conhecimento sobre Testes de Software

Concordo plenamente
Concordo parcialmente
Talvez
Discordo parcialmente
Discordo plenamente

14ª) O Agente Inteligente respondeu as perguntas de forma satisfatória.

Concordo plenamente
Concordo parcialmente
Talvez
Discordo parcialmente
Discordo plenamente

15ª) As interações do Agente se mostraram satisfatórias Concordo plenamente

Concordo parcialmente
Talvez
Discordo parcialmente
Discordo plenamente

16ª) O Agente Inteligente se mostrou uma boa estratégia de apoio a aprendizagem.

Concordo plenamente
Concordo parcialmente
Talvez
Discordo parcialmente
Discordo plenamente

17ª) O Agente pode servir como ferramenta de auxílio ao ensino presencial.

Concordo plenamente
Concordo parcialmente
Talvez
Discordo parcialmente
Discordo plenamente

Caso deseje, deixe aqui seus comentários sobre a experiência de interação com o agente.