PROJETO CITTÀ COSMOPOLITA:
simuladores de redes de cidades

Processo 551846/2011-9

Edital MCT/CNPq N º 75/2010 - RHAE - Pesquisador na Empresa

Vigência: 28/11/2011 a 27/05/2014

Relatório Final

PROJETO CITTÀ COSMOPOLITA
Relatório Final

Resumo:

O projeto *Città Cosmopolita- Simuladores de Redes* de *Cidades* visa à pesquisa e ao desenvolvimento dos módulos para otimização do jogo computacional educacional e simulador de cidades Città, que vem sendo desenvolvido pelo LELIC (Laboratório de Estudos em Linguagem, Interação e Cognição da Faculdade de Educação da UFRGS) em parceria com a empresa CONEXUM. Os novos módulos concebidos para o Città contemplam a interação entre as cidades virtuais construídas pelos estudantes e entre estes, a comunicação com a assistente inteligente Maga Vitta baseada na análise do estado afetivo (emoção no texto e na voz). O interesse da CONEXUM está no desenvolvimento de novos sistemas com interfaces facilitadas por assistentes. Ao longo do projeto as escolas atendidas pelo LELIC através do Projeto CIVITAS terão um benefício direto com o uso do jogo em suas atividades pedagógicas.

Sumário

Introdução 3
Desenvolvimento das Metas 5
Ensaios:
- **CI(TTÀ)VITAS** 12
- **Multiplayer no Città** 16
- **Computação Afetiva e Agentes Conversacionais** 22
- **Construindo Cidades Virtuais com o Civitas Città** 27

Relatórios individuais:
- **Maribel Susane Selli** 39
- **Mércio Luciano Teixeira Soares** 57
- **Laurent Le Avner Quevedo** 81
- **Andressa Cruz Nepomuceno** 111
- **Juarez Dubois da Costa** 124
- **Pablo Felipe Leonhart** 153
- **Yuri Dias Meira** 161
- **Andréa Aparecida Konzen** 172
- **Alexsandro Rosa dos Anjos** 175
- **Vinícius Brum Bertoldo** 176

Publicações em eventos:
- **Editor de Cidades Civitas Città** 191
- **LELIC – Laboratório de Estudos em Linguagem, Interação e Cognição** 202
- **Città Cosmopolita – simuladores de cidades em redes** 203

TecDoc 204

Introdução

O Projeto ***Città Cosmopolita*** - ***Simuladores de Redes de Cidades***, foi aprovado em novembro de 2011 pelo edital *MCT/SETEC/CNPq N º 75/2010 - RHAE - Pesquisador na Empresa*, no intuito de desenvolver novos módulos para o jogo educacional Città, desenvolvido para o projeto CIVITAS – Cidades Virtuais: tecnologias de aprendizagem e simulação, do Laboratório de Estudos em Linguagem, Interação e Cognição (LELIC[1]), da Faculdade de Educação/UFRGS, pela empresa Conexum Sistemas Computacionais Inteligentes Ltda[2]. Città é um jogo digital que vem sendo projetado pelo LELIC desde 2002 e tem tido protótipos desde então. Mas a primeira versão que esteve habilitada a ser aplicada em sala de aula foi em 2009, escrita em java e utilizando uma biblioteca específica de renderização 3D por OpenGL. A partir daí a versão Java foi sendo mantida até o início deste projeto.

Analisando as possibilidades de desenvolvimento com a equipe no início do projeto, em novembro de 2011, chegamos à conclusão que seria urgente a troca da plataforma de desenvolvimento de Java para Blender. Também percebemos que esta alteração iria qualificar não só o desenvolvimento computacional, como também a percepção pelo lado dos estudantes usuários do Città.

Desta forma, os desenvolvedores do Città, que estavam em nosso projeto anterior (Assistiva[3]), **Vinícius Brum Bertoldo** e **Mércio Luciano Teixeira Soares** iniciaram os estudos de viabilidade para a transposição de plataforma de desenvolvimento. Luciano aprovou a facilidade de importação de objetos 3D usados na versão Java, e posteriormente se manifestou positivamente pela construção dos objetos apenas no Blender. Atualmente é a sua ferramenta preferida para modelagem 3D.

Vinícius rapidamente criou o universo básico do Città, através dos scripts python para criação do mapa e inserção dos objetos. Suas primeiras dificuldades foram com relação à manipulação dos objetos como mouse e posteriormente com os blocos lógicos, que permitem a definição de sensores, controladores e atuadores para o jogo.

Enquanto isso, **Alexsandro Rosa dos Anjos** dava suporte nos sistemas e computadores, configurava o servidor e zelava pela atualização e backup.

Dentro da área pedagógica, **Maribel Susane Selli** , que também já vinha acompanhando o Projeto Assistiva, agora como bolsista cuidou de fazer o planejamento da inserção do jogo primeiro com os professores atuantes nas escolas do Projeto CIVITAS, para posteriormente ser aplicado aos alunos.

Em maio de 2012, Alexandro sai do projeto ao ser contratado por um empresa de software, e assume em seu lugar **Juarez Dubois da Costa**, analista de sistemas que prosseguiu o trabalho, acrescentando maior apoio à equipe de desenvolvimento. Juarez auxiliou no acesso a banco de dados e posteriormente na implementação dos primeiros testes multijogador (veja o ensaio *Multiplayer no Città* neste relatório).

A primeira oficina do Città com professores é feita em agosto de 2012, ainda na versão Java. Como previsto, as limitações da versão Java eram de difícil superação, projetando um futuro promissor para a versão Blender. Apesar da dificuldade de manutenção do código Java, as correções foram realizadas, disponibilizadas e aplicadas com os alunos. Mesmo após o uso

[1] http://www.lelic.ufrgs.br
[2] http://www.conexum.com.br
[3] http://assistiva.conexum.com.br

pelos alunos, atualizações continuaram sendo feitas, de forma a qualificar o uso do Città nas escolas. Todas as intervenções realizadas estão registradas no relatório, ensaios e artigos da bolsista Maribel.

Em novembro de 2012, alegando dificuldade de conciliar trabalho e estudo, Vinícius deixa a equipe e em seu lugar inicia **Laurent Le Avner Quevedo**, graduando de jogos digitais. Neste mês também entra na equipe **Andressa Cruz Nepomuceno**, até então bolsista da Conexum, mas com sua conclusão no curso técnico em informática, solicitou-se a indicação para bolsa ATP-B, sendo aceita pelo comitê do CNPq. Andressa começou a atuar então com o desenvolvimento de chat de texto e estudos da possibilidade de comunicação também por voz no Città.

Os novos bolsistas rapidamente inteiram-se com a plataforma Blender e com a linguagem python, prosseguindo o desenvolvimento da primeira versão neste ambiente, a qual foi apresentada em sessão de uma hora no Fórum Internacional do Software Livre, em 3 de julho de 2013[4]. Para esta seção, a equipe usou o próprio Blender para montar uma apresentação, o que permitiu a navegação dentro de uma cidade 3D entre cada tópico exposto. Nesta primeira versão do Città Blender, já estava funcional a modelagem do terreno, a construção de casas. As funcionalidades de chat de texto e multijogador estavam em protótipos independentes. Apenas o protótipo multijogador não pode ser mostrado por não ter rede local disponibilizada para o evento na sala.

Ainda em julho, foi ministrada uma oficina de jogos em Blender na Universidade do Estado da Bahia (UNEB), em Salvador. O convite partiu do GEOTEC[5], que trabalha em parceria com a Conexum e o LELIC.

Em setembro de 2013, Juarez sai do projeto devido à contratação por empresa de software. Dada a dificuldade de encontrar-se profissionais disponíveis para o projeto, solicitou-se o desmembramento em 3 bolsas, que foram destinadas a **Andréa Aparecida Konzen**, professora da UNISC, colaboradora desde o Projeto Assistiva, **Pablo Felipe Leonhart** e **Yuri Dias Meira**, graduandos de Computação.

Andréa foi incumbida de supervisionar os bolsistas e analisar a funcionalidade do assistente inteligente Maga Vitta (desenvolvido no Projeto Assistiva) para a versão Blender, que teria auxílio de Yuri para esta tarefa.

Pablo entrou para auxiliar na funcionalidade dos diferentes bancos de dados usados na versão monojogador (SQLite) e multijogador (MySQL). Como a equipe estava preocupada em poder resgatar mapas desenhados nas diferentes versões, foi necessário o desenvolvimento de um script em PHP para fazer a conversão entre as diferentes bases de dados.

Em novembro de 2013, o coordenador Daniel Nehme Müller participou do 4º Encontro Técnico do RHAE Pesquisador na Empresa, onde apresentou um poster[6] durante o evento. O Encontro é o momento de troca de experiências entre os diversos projetos RHAE financiados pelo CNPq.

Em 2014 foram ministrados novos cursos de Blender em fevereiro, novamente para o GEOTEC, na UNEB em Salvador; em abril, para o grupo CIUP, o Centro de Informática da Universidade Pedagógica de Moçambique, em Maputo; e em maio, no Fórum Internacional do Software Livre, em Porto Alegre. Os grupos GEOTEC e CIUP que convidaram para os cursos são parceiros na pesquisa científica e na execução de projetos com LELIC e com a Conexum.

---

[4] A gravação em vídeo desta sessão foi disponibilizada pelo evento no link: *http://hemingway.softwarelivre.org/fisl14/high/40a/sala40a-high-201307031611.ogg*

[5] http://www.uneb.br/geotec

[6] https://sourceforge.net/p/civitas/code/ci/master/tree/relatorios/2013/poster_cosmopolita.jpg

Desenvolvimento das Metas

1. aplicar o Città nas escolas para o intercâmbio pessoal entre os estudantes acerca das estórias desenvolvidas individualmente e coletivamente sobre as cidades, através de um ambiente gráfico

Devido ao término do convênio das escolas com o LELIC/UFRGS no segundo semestre de 2013, a úlltima versão do Città de 2014 não pode ser aplicada. Apesar disso, as funcionalidades solicitadas foram implementadas, conforme pode ser constatado no código disponibilizado no SourceForge[7] e nos relatórios individuais dos bolsistas apresentados neste relatório.

2. desenvolver novos módulos do jogo Città, envolvendo:

2.1. Conexão dos simuladores em rede

A primeira forma de conexão testada foi através de banco de dados, para troca de objetos e de texto. Constatou-se que deveria haver duas versões distintas, monojogador e multijogador, uma vez que esta última necessita de um módulo servidor adicional para ser instalado na rede da escola. As funcionalidades de chat e assistente inteligente podem operar somente na versão multijogador, por necessitarem do módulo servidor. Maiores detalhes podem ser encontrados nos relatórios de Andressa, Juarez e Pablo.

2.2. Criação de interfaces de conexão entre os estudantes

O chat de texto foi a funcionalidade criada para conexão entre os estudantes. Estudos de voz também foram feitos, porém foram encontradas limitações no uso das bibliotecas python de áudio a partir do Blender.

2.3. Melhorias de usabilidade em termos de ambiente gráfico, alertas do agente pedagógico e testes de uso

Todos os testes executados mostraram uma tendência de melhoria sensível a favor da versão Blender, tanto na qualidade das imagens quanto na usabilidade. Também foi desenvolvido o avatar da Maga Vitta para os alertas do assistente (ver relatórios de Mércio Luciano Teixeira Soares e Yuri Dias Meira e o ensaio *Computação Afetiva e Agentes Conversacionais*).

3. Capacitar professores do Projeto CIVITAS no âmbito das pesquisas realizadas no projeto Cosmopolita

Saídas de campo às escolas foram realizadas durante todo o projeto. Maiores detalhes nos relatórios de Maribel Susane Selli , Andréa Aparecida Konzen, Mércio Luciano Teixeira Soares, Andressa Cruz Nepomuceno e Laurent Le Avner Quevedo.

[7] https://sourceforge.net/projects/civitas/

4. Tornar público os achados do projeto, através de publicações semestrais, seminários, cursos e palestras

Os trabalhos desenvolvidos foram e ainda vêm sendo sistematizados e divulgados na forma de artigos, palestras, cursos e distribuição de software, como descritos a seguir.

- ***Trabalhos apresentados em eventos:***

MÜLLER, DANIEL NEHME ; QUEVEDO, LAURENT LE AVNER ; NEPOMUCENO, ANDRESSA CRUZ ; SOARES, LUCIANO ; DA COSTA, JUAREZ DUBOIS ; SELLI, MARIBEL SUZANE ; AXT, MARGARETE. Editor de Cidades Civitas Città**.** In: Worshops do II Congresso Brasileiro de Informática na Educação, 2013.

MÜLLER, DANIEL NEHME ; DA SILVA, ANDRÉA KONZEN ; SELLI, MARIBEL SUZANE ; TIMBANE, SANSÃO ; BRAITBACK, OTTO ; EPSTEIN, DANIEL ; REMIÃO, JOELMA ; AXT, Margarete . LELIC – Laboratório de Estudos em Linguagem, Interação e Cognição**.** In: Worshops do II Congresso Brasileiro de Informática na Educação, 2013. (Resumo sobre o grupo de pesquisa).

QUEVEDO, L. L. A. ; NEPOMUCENO, A. C. ; SOARES, M. L. T. ; COSTA, J. D. ; SELLI, M. S. ; MÜLLER, D. N. Città Cosmopolita – simuladores de cidades em redes**.** 2013. (Palestra no 14o Forum do Software Livre – Trilha Academia Livre).

- ***Capítulos de livro:***

O Projeto Cosmopolita vem publicando desde 2012 coletâneas de suas pesquisas, iniciadas com uma bolsa dentro do Programa Nacional de Pós-Doutorado (PNPD), juntamente com o GEOTEC/UNEB. Em 2013 e 2014 foram editadas as seguintes obras:

OBRA PUBLICADA EM 2013

Organização para publicação de uma segunda Coletânea com as ações desenvolvidas em 2012, intitulada ***CULTURA DIGITAL E ESPAÇO ESCOLAR: diálogos sobre jogos, imaginário e crianças***, Daniel Nehme Müller, Margarete Axt e Tânia Maria Hetkowski (orgs). Salvador: Editora EDUNEB, 2013 (Edital: 031/2012, Nº pedido 7156/2012), composta por duas sessões de artigos, conforme a seguir:

Sessão I - Pressupostos que embasam a discussão

- Desvendando o espaço nos jogos digitais: a dimensão lúdica e o imaginário (Walter Von Czkus Garrido/ Fabiana dos Santos Nascimento/ Inaiá Brandão Pereira/ Tânia Regina Dias Silva Pereira)
- Entre avatares do virtual e personagens da literatura: a estética da *persona* infantil (Magale de Camargo Machado/ Margarete Axt)
- Marcas criativas do Design de um jogo educacional: O Caso do Jogo/Simulador Kimera - Cidades Imaginárias (André Luiz da Silva/ Edson Machado Filho/ Gustavo de Andrade/ Josemeire Machado Dias)
- Assistente Inteligente *Maga Vitta ((*Daniel Nehme Müller/ Otto Lopes Braitback de Oliveira)
- Aprendizagem e as tecnologias: desvelando um percurso (André Rezende/ Juliana Moura)

Sessão II - Práticas e Vivência no espaço da escola

- Cidades Narradas: dispositivo dialógico para uma estética das relações no campo da pesquisa-formação (Joelma Adriana Abrão Remião/ Paula Marques da Silva/ Margarete Axt)
- Os espaços da cidade de cidade de Salvador: explorando práticas e processos tecnológicos da Educação Cartográfica (Tânia Regina Dias Silva Pereira, Fabiana dos Santos Nascimento, Inaiá Brandão Pereira e Josemeire Machado Dias)
- Civitas e Città compondo os cenários da escola (Maribel Susane Selli/ Felisberto Felix Singo/ Daniel Nehme Müller)
- Formação de Professores produzindo inclusão digital e aprendizagem numa perspectiva (est)ética (Maribel Susane Selli/ Margarete Axt)
- Geotecnologia e Educação Cartográfica na formação de professores de professores dos anos iniciais do ensino. (Rafaela Santos de Souza e Tânia Maria Hetkowski)
- Civitas Brasil – Moçambique: Percursos de uma parceria académica (Cacilda Nhanisse/ Félix Singo)

OBRA A SER PUBLICADA – 2014

Organização da terceira Coletânea com as ações desenvolvidas em 2013, sob o título de **JOGOS-SIMULADORES: relatos de experiência sobre o desenvolvimento do Cittá e do Kimera**, Daniel Nehme Müller, Tânia Maria Hetkowski, Margarete Axt e André Luiz da Silva (orgs). Salvador: EDUNEB, 2014 (no prelo), composta por duas sessões de artigos, conforme a seguir:

Parte 1 - LÓCUS LELIC-CONEXUM

- Interface para Jogos Educacionais: o impacto da usabilidade no simulador de cidades Città (Daniel Nehme Müller Laurent Le Avner Quevedo Luciano Soares)
- Realidade Aumentada como ferramenta educacional (Vítor Azevedo Andriotti Daniel Nehme Müller)
- Desenvolvimento de Módulo Multijogador no Editor de Cidades Città (Juarez Dubois da Costa Daniel Nehme Müller)
- Nas itinerâncias de um foguete-ilha-tecnológica, a dialogia e polifonia de uma sala de aula (Paula Marques da Silva, Joelma Adriana Abrão Remião e Margarete Axt)

Parte 2 - LÓCUS GEOTEC-UNEB

- K-Engine: Desenvolvimento do Motor do Jogo-simulador Kimera-Cidades Imaginárias (Diego Potapuczuk e Tânia Maria Hetkowski)
- Refletindo sobre Criação de Narrativas e Bandas Sonoras para Jogos Digitais: O Mundo possível do jogo Kimera – cidades imaginárias (Gustavo Erick de Andrade e Eliaquim Aciole)
- O Uso do método PDCA na Criação do Design de Personagens para a Produção do Jogo/Simulador Kimera – Cidades Imaginárias (André Luiz Souza da Silva, Gabriele Falcão Duque e Manoella Christina Souza Oliveira)
- Design Gráfico: Criação de Elementos 3D para a composição do Jogo-simulador Kimera (Josemeire Dias e Edson Machado)

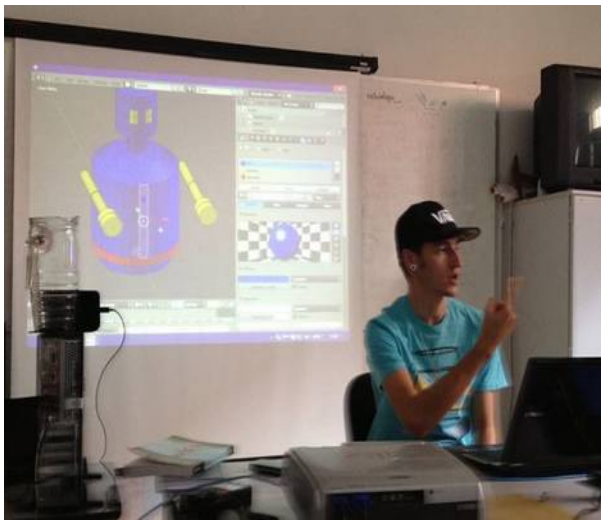
- Validação do Beta do Jogo-simulador Kimera: a cidade imaginária dos meninos e meninas da Escola municipal Álvaro da Franca Rocha (Fabiana Nascimento dos Santos, Inaiá Brandão Pereira e Walter Garrido)

- ***Outros trabalhos derivados:***
  - **Página do Projeto Cosmopolita**: http://cosmopolita.conexum.com.br
  - **JOGO-SIMULADOR KIMERA (GEOTEC/UNEB): Coordenação Geral:** Tânia Maria Hetkowski (UNEB). **Parceiros:** Daniel Nehme Müller – CONEXUM, Margarete Axt – UFRGS e Lynn Alves – UNEB. **Equipe Pedagógica:** Fabiana Nascimento, Inaiá Brandão, Walter Garrido e Tânia Regina Pereira. **Design e Transmídia:** Josemeire Dias, André Betonnasi, Edson Machado, Gabriele Duque, Gabriel Torres (1ª fase) e Fabiana Carvalhal (1ª fase). **Programação:** André Rezende, Diego Potapczuk, Saulo Leal e Humberto Santiago. **Sound Design:** Eliaquim Aciole e Acácia Monteiro. **Marketing:** Ricardo Garcia. **Roteiro:** Gustavo de Andrade

- ***Cursos:***

Oficina 1: treinamento no uso do Città versão Java para professores municipais de Estrela (RS), em 7 de agosto de 2012, na escola Leo Joas.

Oficina 2: **Blender (blender3d),** ocorrida na UNEB, em Salvador, entre **18 e 20 de julho de 2013,** ministrada pelos bolsistas Laurent Quevedo e Luciano Soares**, com a participação de 23 (vinte e três) pesquisadores do GEOTEC, UNEB, Instituto Federal Baiano, Instituto Federal Bahia e SENAI.** Possibilitou aos componentes do grupo GEOTEC conhecerem um programa de código aberto que possibilita explorar animação, texturização, composição, renderização, edição de vídeos e criação de aplicações interativas em 3D, importantes técnicas para a criação de jogos e de simuladores. Este programa foi, após a oficina, usado pelos professores do curso de design nas suas aulas junto com seus alunos de graduação. As fotos a seguir foram tiradas no curso.



Oficina 3: **Blender (blender3d),** novamente na UNEB, agora para um público maior, de 28 a 31 de janeiro de 2014, ministrada pelos bolsistas Laurent Quevedo, Luciano Soares e Andressa Nepomuceno.

Oficina 4: **Blender (blender3d),** a mesma oferecida na UNEB, mas agora para alunos de graduação e pós-graduação da Universidade Pedagógica de Moçambique, em Maputo. Ocorreu entre 11 e 15 de abril de 2014, ministrada pelo coordenador Daniel Müller. A foto a seguir foi tirada em uma das aulas do curso.

Oficina 5: Oficina de Desenvolvimento de Jogos em Blender, 15ª edição do Fórum Internacional do Software Livre, em 8 de maio de 2014, ministrada pelos bolsistas Laurent Quevedo e Andressa Nepomuceno.

- ***Código fonte em software livre:***

Desde novembro de 2008 o código-fonte do Città vem sendo disponibilizado e atualizado no repositório de software livre SourceForge: http://sourceforge.net/projects/civitas/

5. Consolidar ferramentas de software para desenvolvimento de futuras aplicações baseadas nas pesquisas

A plataforma Blender ficou consolidada para a equipe como a ferramenta mais adequada a todas as necessidades do desenvolvimento do Città. Isso porque o Blender contempla:

- desenho dos modelos gráficos;
- modelagem de animações (previsto para serem incluídas em futuras versões);
- máquina de jogo própria, em constante atualização pela comunidade desenvolvedora;

- facilidade para a criação de sensores de jogo, através dos blocos lógicos;
- simplicidade da programação script, através do python, para desenvolvimento dos recursos de jogabilidade;
- geração de um arquivo executável, ou pode ser executado através do aplicativo Blenderplayer;
- flexibilidade do software livre, que permite livre distribuição e alteração de seu código-fonte.

Por outro lado, encontramos desafios, quanto ao uso da ferramenta, a solucionar no futuro:

- limitações de uso de bibliotecas python dentro do Blender;
- falta de ferramenta de depuração de código python;
- necessidade de configurar um grande número de elementos internos ao Blender;
- aparente impossibilidade de abrir navegador web internamente no game.

Embora existam estes desafios, eles são vistos como solucionáveis num futuro próximo. Apesar deles, tanto a velocidade de programação quanto a usabilidade do jogo ficaram extremamente qualificadas com o uso do Blender. A equipe acredita que tenha encontrado a ferramenta ideal para o Città, após tantos anos de projeto, passando por diversas tecnologias que não satisfaziam todas as necessidades sonhadas para o jogo.

6. Agente inteligente Maga Vitta como agente pedagógico emocional

Foi possível ainda desenvolver um estudo exploratório sobre a faces emocionais da Maga Vitta, através do trabalho de Andréa Konzen (ver ensaio **Computação Afetiva e Agentes Conversacionais**) e da colaboração do doutorando PPGIE/UFRGS Daniel Epstein com a nossa equipe (mais detalhes no ensaio **Construindo Cidades Virtuais com o Civitas Città**).

7. Conquistas para a empresa Conexum

Para a empresa Conexum foram muitas conquistas. Além da contribuição social na formação de pesquisadores, profissionais, professores e crianças, propiciou a implementação do primeiro jogo educacional de construção de cidades em três dimensões (3D) existente no Brasil.

Com o estudo exploratório de suas características de uso de multijogador em rede abre um leque de possibilidades para a constituição de ambientes de produtividade em rede, permitindo a interação, interconexão para realização de tarefas cotidianas, treinamentos, planejamentos, etc.

A possibilidade de inserção de modelos no jogo, abre caminho para modelagem de cidades na Arquitetura. A facilidade de alteração do terreno permite que sejam inseridos modelos reais de terreno, com finalidades educacionais, análise geográfica, entre outras.

A tecnologia de monitoramento do ambiente de jogo, e sua análise por assistente inteligente apresenta novos caminhos para o acompanhamento de atividades no computador. Desta forma, os recursos desenvolvidos podem ser aplicados na área educacional, de treinamento, no auxílio em tratamentos de saúde, acessibilidade e tantas outras atividades que necessitam recursos de assistência e monitoramento.

Ensaios

CI(TTÀ)VITAS

Pela bolsista Maribel Selli

Mestre em Educação

Doutoranda em Informática na Educação

Inicia-se há doze anos a idealização de um projeto que continha em seu escopo uma proposta de formação continuada em serviço de professores. Apontava para outras possibilidades de ensino e aprendizagem com a inserção de tecnologias digitais, incorporadas ao contexto da sala de aula e a prática docente, coexistindo com outras propostas até então desenvolvidas pela escola.

O projeto Civitas se constitui em dois planos: pela via da pesquisa e extensão. Na extensão se faz na formação continuada de professores em serviço de escolas públicas e pela via da pesquisa a formação de mestrandos e doutorandos dos Programas de Pós-Graduação em Educação e Informática na Educação da UFRGS vinculados ao LELIC.

Como desdobramentos do projeto, no decorrer desse período, firmamos convênios entre a UFRGS e alguns municípios do Vale do Rio Pardo: Venâncio Aires; do Vale do Taquari: Mato Leitão, Cruzeiro do Sul e Estrela e da Região Centro – Serra: Sobradinho.

Inicialmente, a formação era realizada com professores da terceira série, atualmente o quarto ano do Ensino Fundamental. Com o passar do tempo surgiram outras demandas e foi ampliada para professores da Educação Infantil e do primeiro ao quinto ano. O intercessor nesse caso passa a ser além da cidade, o personagem e seus contextos.

Buscamos referência em alguns princípios que norteiam a proposta. Referiremos dois que farão acontecer o projeto. O primeiro é a garantia de que as reuniões com os pesquisadores/formadores aconteçam incorporadas a carga horária dos participantes do grupo. O segundo, a adesão, entendendo que, ao assumirmos uma outra maneira de pensar a sala de aula é preciso o desejo em fazê-lo.

Ao aceitar o convite, frequentemente, instala-se no professor a dúvida de como desenvolver o trabalho com as crianças. Para nós, quando se questiona e inquieta-se com o seu fazer o professor já iniciou seu percurso.

> A ideia de "início" pode ser traduzida em termos de ruptura, de uma parada, de mudança de rumo. Construir uma diferença na própria prática não apenas não era reconhecida como possível como era desviada (enquanto tendência apenas) do que poderia parecer-lhes repetição da mesma prática. Os professores, ao serem desafiados a observar as atividades em curso (acontecendo independentemente do projeto), e nelas procurar algo inusitado, já estariam experimentando um micro deslocamento da atenção. (MARTINS, 2009, p. 46)

Entre tantas outras concepções de formação que trabalham com uma lógica do como fazer através de sugestões de atividades, com exemplos e uma maneira pré-determinada apenas a ser cumprida trabalhamos com a construção que se faz no processo, pela problematização e invenção de problemas que norteiam os acontecimentos na sala de aula sendo estes oriundos do professor ou das próprias crianças.

Distante da pretensão de que o Civitas é algo jamais proposto enquanto formação. Podemos observar, nas histórias e registros que temos colecionado até então, que o trabalho

acontece sempre de uma outra maneira. Isso possivelmente acontece, em nosso entendimento, porque os participantes e contextos não são os mesmos e cada grupo constrói seu trabalho do seu próprio jeito.

Para, além disso, partimos do saber de cada um e cada uma e de suas histórias de vida. Na composição do trabalho, percebemos, tanto pelas crianças como pelos professores e, na maioria das vezes, extensivos as suas famílias e comunidade escolar que a metodologia que norteia nosso trabalho pode constituir-se como uma possibilidade potente de repensarmos o fazer pedagógico.

O Civitas se propõe, partindo da pesquisa, do trabalho em grupos e da construção coletiva a desafiar as crianças ocupando não somente do lugar de quem só vem para a escola para aprender, mas também para compartilhar seus saberes, com autonomia, na construção do seu conhecimento, pela interação dialógica.

Ao referir-se a este conceito Axt (2006) busca sustentação no dialogismo bakhtiniano que tem implicação nas relações são pautadas pelo diálogo, quando um interlocutor acolhe o enunciado do outro numa relação ética e cuidadosa.

A interação dialógica faz pelos enunciados produzidos pelos interlocutores, em nosso caso, a professora e as crianças.

> [...] corresponde sempre uma atitude responsiva ativa prévia, dando conta de um contexto pragmático, interacional e enunciativo-responsivo, enquanto pressuposto de linguagem e de sujeitos de linguagem em relação. (AXT, 2006, p. 257)

Ao pensarmos a interação como possibilidade de construção assume-se também, enquanto grupo de formadores, um lugar de quem, para além de compartilhar conhecimentos teóricos e práticos, também aprende.

Evidencia-se aí a concepção de que as histórias de vida, as diferentes formações, vivências e experimentações compõem a produção de sentidos e aprendizagem enquanto acontecimento Axt e Elias (2003).

> É trabalhar ao mesmo tempo sobre planos de composição (est) ética, planos de referência científica e planos de construção conceitual, numa relação de reciprocidade, acolhendo a diferença na construção conjunta de saberes/conhecimento. (AXT, 2004, p. 225)

Entender a potência do coletivo na construção do conhecimento é um dos principais eixos que nos levam a acreditar que pode ser possível de que uma metodologia que privilegie a coexistência de práticas, atores, cenários, experimentações num constante ensinar e aprender.

Tendo como pressupostos a invenção, a imaginação, a interação dialógica, a fabulação, a cooperação, a imaginação, o trabalho coletivo, um outro elemento passa a compor os cenários de aprendizagem.

A inserção de tecnologias digitais, entre elas o computador – instalado num espaço dentro da própria sala de aula -, a máquina digital, a filmadora, o gravador mp3 são ferramentas que, junto ao Città desafiam os professores e suas crianças a lançar mão desses recursos nos mais variados momentos da construção do conhecimento.

Entre tantos outros fatores que geram uma autocrítica do trabalho docente porque por mais que se estude e pesquise não raras às vezes nos deparamos com imprevistos que nos desestabilizam. Surge não com o propósito de ser a solução para tudo, mas como mais um elemento que poderia compor com os demais recursos didáticos já utilizados no dia a dia da turma, um intercessor, que seria mais uma alternativa nesse complexo processo de apropriação de conceitos.

Inicialmente, quando trabalhávamos com a 3ª série/4º ano, o intercessor era a cidade, seja ela representada por uma instalação utilizando materiais como papelão, papel, caixas de papel de variados modelos e tamanhos, retalhos, sucatas, seja no simulador de cidades Città.

Agora, com a expansão do projeto para outros anos, desde a Educação Infantil até o quinto ano do Ensino Fundamental, outros intercessores são incorporados ao Civitas como a invenção de personagens e contextos.

Retornaremos ao Città – software livre simulador de cidades com fins educativos que teve sua primeira versão desenvolvida em 2003 utilizando como suporte um editor gráfico em formato VLMR. Em 2009 tivemos uma segunda versão do Città em *Java*.

Atualmente, pelo Projeto Città Cosmopolita 2011-2014 resultante de uma parceria entre a Empresa CONEXUM - Sistemas Computacionais Inteligentes Ltda. e a UFRGS, em linguagem *Blender*, sendo financiado pelo CNPq. Esta última versão está sendo disponibilizada às escolas que trabalham com a metodologia proposta pelo Civitas.

Os princípios do Città pelas vias do Civitas são, entre outros que possam surgir, a construção de cidades ambientalmente sustentáveis, economicamente viáveis e socialmente possíveis que se faz no respeito e reconhecimento entre os moradores já instalados ou a outros que desejarem habitar a cidade.

Com isso vivenciam princípios indispensáveis ao convívio humano como a solidariedade, a consciência ecológica, o respeito e a valorização da vida, a preservação do ambiente, a cooperação e o respeito pelo outro no acontecer da vida em sociedade representada nos mais diversos espaços de fabulação, criação, invenção, problematização do cotidiano e aprendizagem obtidos nos mais variados momentos compartilhados, na construção da cidade.

É premente destacarmos que essa experimentação se dá pela amorosidade e reconhecimento do outro como legítimo e indispensável na relação consigo, com o próprio outro e com o mundo. Como nos diz Bakhtin (2003) eu-para-mim, eu-para-o-outro e o outro-para mim. Ao seguirmos esses princípios nos é possibilitada a administração de conflitos e divergências de forma construtiva. Os problemas que possam surgir na idealização da cidade imaginada se constituem assim como um exercício de cidadania num compromisso ético e alteritário assumido em nossa relação com o outro.

Para atingirmos esse propósito o simulador desafia as crianças, pela invenção de problemas relacionados tanto ao cotidiano como em situações inusitadas, a solucionar por meio de propostas apresentadas ao grupo e, por consenso, definida a mais adequada alternativa para o momento.

Destacamos que os mentores da ideia não são nem supervalorizados nem recebem destaque. O mesmo critério para as propostas que não foram escolhidas que não são desmerecidas ou desprezadas. Todas as contribuições, sem distinção, são valorizadas e acolhidas.

Retomamos nesse momento, a proposta de formação continuada de professores em serviço, o eixo que conduz e viabiliza o projeto. Apropriando-se da metodologia que se faz pelo pressuposto dialógico, propomos ao grupo que participa dos encontros, entre outras tecnologias, digitais ou não, a utilização do simulador Città como um possível disparador de aprendizagens. Ao lançarmos esse propósito nos desafiamos, enquanto grupo, a construir práticas interdisciplinares em nossas aulas que viabilizem o diálogo, também, entre as diferentes áreas do conhecimento. Tudo isso mediado pela escuta, os afetos, os processos criativos que vão, no percurso, dando origem a cidade.

Diante disso poderíamos construir novos caminhos e possibilidades de aprendizagem aproximando-o do dia a dia da sala de aula e da vida das crianças. Os acontecimentos da cidade imaginada se tornam os vividos no cotidiano e, pela fabulação, são compartilhados com os colegas reproduzidos na vida dos moradores, personagens que convivem nesse espaço. Percebemos aí o quanto pode ser promissora essa prática tanto no que diz respeito a formação dos professores, como sua reflexão sobre a prática e, por conseguinte, como implicação tanto na vida como na apropriação do conhecimento da turma.

Entre as reverberações possíveis que emergem dessas experimentações compartilhamos princípios vitais ao convívio em sociedade que também ecoam na sala de aula como um exercício de cidadania e produção de autonomia.

Sendo um dos possíveis diálogos que, tanto a formação de professores pelas vias do Ci (ttà) vitas quanto à pesquisa podem produzir pelo pressuposto dialógico que os sustenta - enquanto orientação metodológica - pode fazer com a muitas vezes fragilizada relação professor que ensina/ aluno que aprende proporcionando que ambos assumam um outro lugar, o de atores e autores do aprender.

Compartilhamos aqui um dos percursos, entre tantos outros, que podem ser escolhidos pelos profissionais que desejam buscar outras formas de ensinar e aprender.

Nossos registros, no decorrer desses anos têm traduzido alguns dos encantamentos que se produzem, tanto entre os pesquisadores, os professores em formação, como nas crianças. Ainda, em não raras às vezes nas famílias que também acabam por se envolver no projeto fazendo parte da vida escolar de seus filhos. Isso acaba se transformando numa alternativa alegre de trazer os pais para dentro da escola, um propósito constantemente perseguido por esta instituição.

Acreditamos que os primeiros doze anos do Ci (ttà) vitas serão apenas o registro dessa primeira década de sua existência e que ainda vamos continuar produzindo muitas histórias em nossas escolas e na vida dos integrantes do projeto.

Assim seguiremos compartilhando o sonho de tornar nossas escolas públicas um pouco mais desafiadoras e instigantes para nossas crianças abrindo espaços para que o conhecimento em suas diferentes manifestações e formas de expressão possa emergis nos mais variados espaços tanto da sala de aula como da escola e seu entorno.

Multiplayer no Città

Pelo bolsista Juarez Dubois da Costa
Bacharel em Análise de Sistemas

Introdução

Este artigo trata da implementação de uma versão multijogador do simulador de cidades Città onde tentamos utilizar algumas técnicas de programação baseadas no conceito de multijogador.

O Città foi desenvolvido para ser utilizado em sala de aula com os estudantes do quarto ano do ensino fundamental, congregando estudos de geografia, cartografia, gerenciamento de recursos naturais, espaciais e outros relativos às cidades criadas virtualmente no simulador de cidades.

Os jogos multijogador permitem aos usuários experimentar a troca de informações através do ambiente virtual do jogo, seja ele por métodos de comunicação verbal de palavras ou por métodos de troca de informações dos objetos de jogo, como a alteração do mapa, objetos virtuais e outros tipo de dados disponibilizados através do jogo e de suas possibilidades de construção de objetos virtuais.

Durante o desenvolvimento do software Città havia a necessidade de criação de uma versão de múltiplos jogadores para que pudesse haver uma maior integração dos estudantes para a criação de uma cidade virtual onde pudessem ser aplicados os conceitos abordados em sala de aula em um único mapa.

Como o Città estava sendo desenvolvido sobre a plataforma do Blender havia a necessidade de se utilizar a linguagem de programação Python ou recursos incorporados ao próprio Blender.

A partir desta perspectiva de arquitetura de sistema de computação foram pesquisadas as possibilidades de se implementar vários jogadores ao mesmo tempo no Blender. Apenas alguns tipos de implementação estavam disponíveis para a possibilidade de múltiplos jogadores, dentre elas escolhi utilizar duas perspectivas de implementação: um plugin, ou add-on, do Blender que estava sendo desenvolvido para multiplayer e outra perspectiva mais baseada em termos de programação mais ortodoxa com a utilização de sistemas gerenciadores de bancos de dados em redes.

Após algumas tentativas de implementação utilizando o add-on de multiplayer do Blender foi descartada esta possibilidade, pois o add-on possui uma implementação de jogo no qual seria mais conveniente ser utilizado em jogos de tiro em primeira pessoa (FPS ou First Player Shooter), ou simuladores de corridas, etc. O add-on é mais adequado ao FPS porque torna seja necessário representar o outro jogador na cena de jogo - o que no simulador de cidades Città não ocorre. A grande limitação do add-on era a necessidade de representação dos demais jogadores na cena de jogo. Mas o nosso simulador de cidades não possui um jogador representado por uma entidade virtual em três dimensões, mas sim um mapa onde cada jogador real insere seus objetos sobre o mapa para caracterizar uma cidade. Em cada interação dos jogadores reais haveria uma substituição do objeto virtual, pois a cada objeto novo inserido no mapa o objeto virtual seria substituído por outro a ser colocado no mapa pelo jogador real.

Após repensar a forma de jogo multijogador foi sendo visualizada uma nova perspectiva para a utilização de múltiplos jogadores no Città.

A nova perspectiva foi baseada na possibilidade de os jogadores reais terem os dados do seu mapa em comum salvos em um banco de dados. Este modelo de jogo faria com que cada jogador tivesse que ler e gravar os dados do jogo em uma mesma tabela de um banco de dados em comum. Assim sendo o que o jogador inserisse no mapa seria imediatamente visível

aos demais jogadores conectados ao jogo. Necessitando para isto algumas modificações na lógica do jogo com relação à versão desenvolvida para um único jogador.

As diferenças de implementação do jogo de múltiplos usuários para a versão de um único usuário são o modo de inserção de objetos, o carregamento dos dados do mapa, a remoção de objetos e a criação de uma nova tabela de dados para um jogo novo.

Na inserção de objetos a diferença está no fato de no modo de um único jogador o objeto é inserido sem ter que ser gravado de imediato no banco de dados, pois o mapa todo é salvo ao sair do jogo, enquanto que no modo de múltiplos usuários o objeto deve ser gravado no banco de dados a cada nova inserção no mapa, para que o objeto fique imediatamente visível a todos os usuários.

Conceitos

Multiplayer – Para este nosso trabalho o conceito de multiplayer, ou multiusuário, ou ainda multijogador, está relacionado ao fato de termos vários jogadores participando de uma mesma partida de um jogo. Atualmente os jogos multiplayer têm se tornado bastante populares, principalmente os jogos chamados de "jogo eletrônico online multijogador em massa" onde muitos jogadores compartilham de um mesmo jogo para realizar determinadas tarefas ou um objetivo em comum.

Città – Software simulador de cidades virtuais desenvolvido pelos bolsistas do Conselho Nacional de Pesquisa e desenvolvimento tecnológico (CNPq) coordenados pelo professor Daniel Nehme Müller.

Blender - O Blender é uma ferramenta de modelagem e edição de objetos em 3 dimensões e também é utilizado para a criação de animações gráficas. Sua licença livre permite que seja utilizado para o desenvolvimento de modelos, animações e jogos sem fins lucrativos como no caso do Città. Um recurso recentemente adicionado foi o motor de jogos, o qual estamos utilizando para o desenvolvimento do simulador de cidades Città. O Blender foi utilizado para a criação de alguns filmes como Big Buck Bunny, Yo Frankie! e Sintel conforme informações do site Wikipedia e nos sites dos próprios filmes.

Técnicas aplicadas

Nesta seção são descritas as técnicas que foram utilizadas no desenvolvimento do jogo durante as fases que necessitaram de modificações com relação à versão monousuário.

Remoção de objetos

Na remoção de objetos, os dados do objeto devem ser prontamente removidos da tabela do banco de dados para que os demais jogadores saibam que o objeto foi removido e a área no mapa foi disponibilizada enquanto que na versão de um único jogador o mapa completo é salvo ao sair do mapa do jogo com todas as alterações que foram realizadas durante a partida.

Carregamento de objetos

Com relação ao carregamento dos objetos do mapa teve que ser feita uma nova lógica de recuperação das informações. No modo de jogo de um único jogador o carregamento era feito de uma única vez no início do jogo enquanto na versão de múltiplos jogadores deve haver uma leitura quase que constante dos dados do mapa para que a visualização das alterações no mapa seja quase instantânea.

Criação de um novo jogo

A criação de novos jogos foi outro aspecto que teve que ser alterado para a versão de múltiplos jogadores. Na versão de um único jogador foi utilizado um sistema de bancos de dados local no sistema de arquivos onde é criado um novo arquivo de banco de dados e uma nova tabela de mapa no arquivo recém criado. Na versão de múltiplos jogadores é criada uma nova tabela no banco de dados de rede no servidor remoto.

Modelo de persistência dos dados

Outro fator importante foi o desenvolvimento do modelo de persistência dos dados no sistema gerenciador de bancos de dados. A versão da linguagem Python que estava sendo utilizada no Blender estava impossibilitando de se utilizar uma biblioteca de conexão com o banco de dados pois ao salvar os dados dos objetos do mapa estes vinham com representações de zeros ao invés de serem inseridos os dados reais dos objetos. As tentativas de inserção manuais eram executadas com todos os dados inseridos normalmente. Porém as bibliotecas que o Blender estava utilizando faziam com que os dados fossem passados de forma errônea, originando dados em zero.

Arquitetura baseada em webservices

A próxima proposta de modelo de arquitetura do sistema foi baseada em webservices. Neste modelo os dados são passados para um webservice localizados em um servidor web, servidor de internet, o qual faz a inserção ou consulta dos dados no banco de dados. O mecanismo de busca de dados foi realizado da mesma maneira com a exceção de que o webservice monta um arquivo XML com os dados dos objetos a serem inseridos no mapa do jogo.

A perspectiva de webservices foi implementada com sucesso pois os dados são inseridos com os valores originais sem que haja alterações por parte das bibliotecas da linguagem Python existentes no Blender.

Um dos problemas relativos à utilização de webservices é a programação dos arquivos de script dentro do Blender, pois necessitariam de muitas linhas de código e a verificação de que o Blender possua as suas bibliotecas em Python necessárias à correta utilização dos webservices e dos dados repassados via XML. Mas após alguns testes com o Blender foi constatado de que o mesmo possuía todas as qualidades necessárias à implementação de utilização dos webservices.

Após a implementação dos scripts de carga de dados, de inserção de dados e criação de tabelas via webservices foi necessário alterar alguns aspectos do Città como fazer com que o jogo identifique o modo de jogo selecionado pelo jogador real, se é monousuário ou multiusuário para fazer a execução do script correto conforme o tipo de jogo selecionado. Além destes foi necessário implementar um sensor para detecção de mudanças no mapa de jogo, fazendo com que seja consultado de forma periódica o mapa do jogo, verificando se houveram mudanças no mapa.

Discussão

As alterações realizadas para proporcionar a funcionalidade de multiusuários foram ralizadas com sucesso. Os testes de acesso a dados utilizando a arquitetura de webservices foram satisfatórios com resultados bastante expressivos e um tempo baixo de resposta, inferior a 1 segundo na maior parte dos resultados dos testes.

O motor de jogos do Blender (BGE) utiliza da linguagem Python para efetuar o desenvolvimento de códigos de programação com a finalidade de proporcionar funcionalidade

aos objetos 3D na cena de jogo, lógica de interação entre os objetos, conexão com bancos de dados dentre outras funcionalidades.

Um dos problemas constatados durante o desenvolvimento foi o fato de que ao utilizarmos a biblioteca de software do banco de dados MySQL através do Blender os dados foram transmitidos nulos. Efetuamos uma alteração nas tabelas do banco de dados para que não aceitassem valores nulos e na próxima interação de testes os valores inseridos foram zerados.

Durante os testes manuais efetuados dentro da linguagem Python original instalada fora do Blender os dados eram registrados com sucesso.

Chegamos à conclusão de que a linguagem Python utilizada no Blender possui modificações que interferem na utilização desta biblioteca de software do banco de dados MySQL, fazendo com que tivéssemos que alterar a arquitetura de persistência de dados para utilizar webservices.

Efetuamos a alteração da linguagem Python do Blender instalando a linguagem Python sobre o Python do Blender mas esta alternativa se mostrou inviável devido ao fato de que o Blender apresentou diversos erros que impossibilitaram que o mesmo funcionasse corretamente.

Sendo assim necessitamos de melhorias na linguagem Python utilizada pelo Blender para que possa utilizar outras bibliotecas de software da mesma forma que é utilizado na linguagem Python padrão.

Conclusão – passos seguintes

Após desenvolvermos este trabalho de múltiplos jogadores no Città constatamos que existe uma necessidade de desenvolvimento na linguagem Python utilizada no Blender para que possa aceitar as bibliotecas de software externas.

No módulo (“add-on”) de multiplayer verificamos a necessidade de alterações para que possa ser utilizado em jogos onde não é requerido um avatar.

Os próximos passos residem em efetuar testes com as novas versões do Blender a fim de verificar se a linguagem Python já aceita a inclusão de bibliotecas de software da mesma forma que a linguagem Python externa ao Blender aceita estas bibliotecas de software para utilizarmos sistemas de bancos de dados sem a necessidade de utilização de webservices.

Bibliografia

Big Buck Bunny Movie - Filme <http://www.bigbuckbunny.org/index.php/about/>
Blender – Site Oficial <http://www.blender.org>
Blender Game Engine - Multiplayer <http://code.google.com/p/bge-networking/>
Città – Simulador de cidades <http://www.sourceforge.net/p/civitas>
Wikipedia Brasil – Jogos multiplayer massivos online <http://pt.wikipedia.org/wiki/Massively_multiplayer_online_game>
Sintel – Filme <http://www.sintel.org/>
Yo Frankie! - Filme <http://www.yofrankie.org/>
Wikipedia Brasil – Blender <http://pt.wikipedia.org/wiki/Blender>

Anexo 1 - Testes de tempo de resposta realizados com Webservices

Testes de tempo de resposta de rede baseados em webservices realizados no software Città.

Os testes a seguir foram realizados para averiguar o tempo de resposta do software Città para averiguar o tempo de resposta dos dados através da rede com a arquitetura de persistência baseada em webservices, seja ela na Internet ou na rede local por interfaces de rede cabeadas e sem fio (wireless).

Seguem os testes:

a. **Teste 1 - realizado dia 21/07/13 às 13:11:**

Realizamos um teste de acesso aos webservices de controle de dados do software Città em dois computadores que estavam conectados à Internet para averigarmos o tempo de resposta através da Internet. Utilizamos o software CittaPackedWS.blend, disponível em <http://sourceforge.net/p/civitas/code/ci/master/tree/citta/citta-blender-mp/CittaPackedWS.blend>, e efetuamos testes de acesso a dados e salvamento de dados.

Na primeira parte do teste foi realizado um teste de carregamento de 10 objetos do tipo casa por quatro vezes na tabela de mapa "mapa_teste2". Os tempos de resposta foram inferiores a um segundo.

Na segunda etapa do teste foram realizados testes de salvamento, onde foram gravados no banco de dados 10 objetos do tipo casa em três passos diferentes. São eles:

Na tabela "*mapa_teste3*" o host A gravou 5 objetos simultaneamente com a gravação de 5 objetos do tipo casa pelo host B. O tempo de resposta para a conclusão da gravação foi inferior a 1 segundo.

Na tabela "*mapa_teste4*" o host A gravou 10 objetos do tipo casa e o host B leu os 10 objetos. Em ambos os casos o tempo de resposta foi menor do que 1 segundo.

Na tabela "*mapa_teste5*" o host B gravou 10 objetos do tipo casa e o host A leu os 10 objetos. A gravação dos dados pelo host B foi superior a 1 segundo e inferior a 2 segundos. Já a leitura dos dados pelo host A foi inferior a 1 segundo.

Dados do teste:

Servidor de rede onde as tabelas estavam armazenadas: Servidor Dell Power Edge T110 II, CPU Xeon E3 1270 V2, 16 Gigabytes de memória, 4 Terabytes de HD em RAID-1 formando 2 Terabytes utilizáveis. Conectado à Internet através de cabo de rede do tipo Gigabit Ethernet. Executando Debian 6 GNU/Linux de 64 bits;

Host A: Computador Core i5 2400, 8 Gigabytes de memória, 1 Terabyte de HD, interface de vídeo Geforce GTS450 com 1 Gigabyte de memória de vídeo GDDR5 dedicada. Conectado à Internet através de cabo de rede do tipo Gigabit Ethernet. Executando o Blender de 64 bits versão 2.67 sobre o sistema Slackware64 14.0 de 64 bits;

Host B: Noteboook Dell Inspiron 14R, CPU Core i5 3210m, 6 Gigabytes de memória, 1 Terabyte de HD, interface de vídeo Geforce GT 630m com 1 Gigabyte de memória de vídeo

DDR3 dedicada. Conectado à Internet através de conexão wireless 802.11g de 54 Mbps. Executando o Blender 2.67 sobre o sistema Ubuntu 13.04 de 64 bits.

b. Teste 2 - realizado dia 22/07/2013 às 14:54:

Realizamos um teste de acesso aos webservices de controle de dados do software Città em cinco computadores que estavam conectados à rede local para averigarmos o tempo de resposta. Utilizamos o software CittaPackedWS.blend, disponível em <http://sourceforge.net/p/civitas/code/ci/master/tree/citta/citta-blender-mp/CittaPackedWS.blend>, e efetuamos testes de acesso a dados.

Neste teste foi realizado um teste de carregamento simultâneo de 10 objetos do tipo casa por quatro vezes na tabela de mapa "*mapa_teste6*" e "*mapa_teste7*". Os tempos de resposta foram inferiores a um segundo.

Dados do teste:

Servidor de rede onde as tabelas estavam armazenadas: Servidor Dell Power Edge T110 II, CPU Xeon E3 1270 V2, 16 Gigabytes de memória, 4 Terabytes de HD em RAID-1 formando 2 Terabytes utilizáveis. Conectado à Internet através de cabo de rede do tipo Gigabit Ethernet. Executando Debian 6 GNU/Linux de 64 bits;

Host 1: Netbook Acer Apire One, CPU Intel Atom N570, 2 Gigabytes de memória, 500 Gigabytes de HD. Conectado à rede através de Wireless 802.11g de 54 Mbps. Executando Blender Player sobre o sistema Windows 7 Home de 64 bits;

Host 2: Computador Core i5-2310, 4 Gigabytes de memória, 1 Terabyte de HD. Conectado na rede através de cabo de rede padrão Gigabit Ethernet. Executando Blender 2.68 de 64 bits sobre o sistema Windows 7 Pro de 64 bits;

Host 3: Computador Core i5-750, 8 Gigabytes de memória, 2 Terabytes de HD. Interface de vídeo Geforce 8400 GS de 256 Mb de memória dedicada. Conectado na rede através de cabo de rede padrão Gigabit Ethernet. Executando Blender 2.67b de 64 bits sobre o sistema Windows 7 Pro de 64 bits;

Host 4: Computador Lenovo Core 2 Quad, 4 Gigabytes de memória, 500Gigabytes de HD. Interface de vídeo Geforce 8400 GS de 256 Mb de memória dedicada. Conectado na rede através de cabo de rede padrão Gigabit Ethernet. Executando Blender 2.68 de 32 bits sobre o sistema Windows 7 Profissional de 32 bits;

Host 5: Computador Core 2 Duo E4500, 2 Gigabytes de memória, 320 Gigabytes de HD. Conectado na rede através de cabo de rede padrão Gigabit Ethernet. Executando Blender 2.68 de 64 bits sobre o sistema Windows 7 Pro de 64 bits.

Com exceção da gravação dos 10 objetos do tipo casa realizadas pelo host B no teste 1, que demorou entre 1 e 2 segundos, os demais testes foram bastante satisfatórios atingindo um tempo de resposta inferior a um segundo (1s).

Referências:

Città – Simulador de cidades <http://www.sourceforge.net/p/civitas>
Blender – Site Oficial <http://www.blender.org>

Computação Afetiva e Agentes Conversacionais

Pela bolsista Andrea Aparecida Konzen
Mestre em Ciência da Computação
Doutoranda em Informática na Educação

1. Introdução

O uso do computador em sala de aula tem proporcionado maior diversificação na maneira de transmitir e receber conhecimento, promovendo a motivação do aluno no seu aprendizado (CUTMORE, 2000).

Os agentes pedagógicos animados têm sido empregados em ambientes de aprendizagem por terem um forte apelo de motivação, justamente por terem meios de interagir de maneira mais antropomórfica através de gestos e expressões faciais emocionais (JAQUES, 2004; McQUINGGAN, 2007). Um tipo específico de agente animado são os agentes conversacionais, que são mais interativos por possuírem a capacidade de se comunicar em linguagem natural com os usuários, envolvendo personalidade e a expressão de emoções. Os trabalhos que existem aplicam estratégias afetivas isoladas das estratégias cognitivas nos ambientes inteligentes de aprendizagem, ou seja, existem ambientes que aplicam somente estratégias afetivas (BURLESON, 2006), outros que aplicam somente estratégias cognitivas (YUNG, 2009) e, ainda existem aqueles que aplicam os dois tipos de estratégias, porém, de forma desconexa ou considerando poucas informações cognitivas do aluno (CONATI, 2002; JAQUES, 2004).

A partir disso, a proposta deste trabalho é a definição e o desenvolvimento de um modelo da mente de um agente pedagógico[8] emocional com o objetivo de aplicar estratégias de competência e afetivas, baseando-se nos estados emocionais inferidos dos alunos, a partir da conversação entre eles e deles com o sistema. Assim, esse modelo da mente do agente pedagógico será aplicado no agente MagaVitta para fins de validação experimental.

2. Estados Afetivos e Traços de Personalidade

A Computação Afetiva é a área da computação que está relacionada com os aspectos que influenciam as emoções (PICARD, 1997). Este campo de pesquisa, que pode ser visto como uma intersecção das áreas Inteligência Artificial, Interação Homem-Máquina e Ciências Cognitivas, tem dois focos principais de pesquisa: um em que são estudados mecanismos para reconhecimento de emoções em máquina através da interação homem-computador e, outro, em que é feita a síntese de emoções através da simulação de emoções em máquina.

A afetividade é todo o domínio das emoções propriamente ditas, dos sentimentos das emoções, das experiências sensíveis e, principalmente, da capacidade em se poder entrar em contato com sensações. Os vários estados que englobam a afetividade são conhecidos como estados afetivos.

Um tipo específico de estado afetivo é a emoção. A emoção não possui uma definição específica, sendo assim diferenciada dos outros estados afetivos através de algumas de suas características, como resposta breve, resultado de uma avaliação de um evento e outros (JAQUES, 2004). Elas são disparadas e diferenciadas por uma avaliação subjetiva de um

[8] No texto, refere-se ao agente proposto tanto como "agente pedagógico animado", assim como "agente conversacional", visto que esse pode ser visto como um agente pedagógico animado devido à sua funcionalidade no ambiente, assim como agente conversacional devido às técnicas de Inteligência Artificial relacionadas à conversação empregadas neste e suas características.

evento, situação ou objeto, chamada de *appraisal* (SCHERER, 1999). São exemplos de emoções a raiva, a tristeza, a alegria, o medo, o desespero e a vergonha.

Outro tipo de estado afetivo considerado mais global e estável é os traços de personalidade, que podem ser definidos como tendências de comportamento típicas de uma pessoa, elementos da personalidade que caracterizam o temperamento (SCHERER, 1999).

3. Agentes Conversacionais

Um agente conversacional, também conhecido como *chatterbot*, tem como objetivo simular um ser humano através da conversação com pessoas, respondendo perguntas de maneira que se tenha a impressão de estar conversando com outra pessoa e não com um programa de computador. Após o envio de perguntas em linguagem natural, o programa consulta uma base de conhecimento e em seguida fornece uma resposta verbal que tenta imitar o comportamento humano. Quando parte de um ambiente inteligente de aprendizagem, o agente conversacional deve possuir *expertise* em um domínio de conhecimento e compreender as necessidades do usuário para melhor direcioná-lo nos conceitos ou nas tarefas mais complexas.

Na mesma linha dos agentes conversacionais, tem-se o ECA (*Embodied Conversational Agent*), um agente autônomo inteligente, que possui uma representação gráfica em um mundo virtual e é capaz de manter alguma forma de conversação com o usuário (PELACHAUD, 2001). Isto é, o agente "fala" com o usuário e tem uma forma gráfica personificada no mundo virtual. Além das representações verbais, esses agentes possuem a capacidade de reconhecer e reproduzir uma comunicação não-verbal.

4. Projeto Civitas

O Civitas (Cidades Virtuais: Tecnologias de Aprendizagem e Simulação) é um projeto desenvolvido no Laboratório de Estudos em Linguagem, Interação e Cognição (LELIC) da Faculdade de Educação da UFRGS. Possui como objetivo principal criar e desenvolver um ambiente para construção de cidades virtuais por simulação baseado em interação multiusuário e compartilhamento de conteúdos digitais. Este projeto é focado em jogos educacionais colaborativos, destinando-se a alunos de 3ª e 4ª séries do ensino fundamental.

Este projeto é composto por várias ferramentas, como: o editor de relevos (Mappa); o construtor de cidades, que é considerado o jogo de civilizações (Città), principal módulo; o editor de objetos (Edittore) e; o agente inteligente (Maga Vitta).

O jogo Città é um jogo colaborativo, onde diversos alunos, orientadas por seus professores, interagem numa cidade virtual, que simula as reações do ambiente às ações de cada usuário conectado, com o objetivo de lhe proporcionar noções básicas de cidadania, ecologia e meio-ambiente (Müller, 2008).

Para auxiliar no controle ecológico da cidade virtual, foi especificado um ECA pedagógico, chamado de Maga Vitta. A idéia é, na medida em que, o aluno utiliza o jogo, de acordo com suas ações no ambiente, o agente Maga Vitta faz suas intervenções necessárias. O agente faz o controle de quatro aspectos: natureza (poluição do ar, água e terra), água (contaminação, desperdício e abastecimento de água), energia (a iluminação e uso de aparelhos elétricos) e população (monitoramento da quantidade de habitantes e sua relação com a infra-estrutura disponível). O aspecto natureza está em fase de elaboração, sendo que os demais aspectos serão desenvolvidos posteriormente (AXT, 2008).

5. Discussão

A partir do exposto, propõe-se um modelo da mente de um agente pedagógico emocional com o propósito de aplicar estratégias de competência e afetivas. As estratégias de competência são aquelas que se interessam no conhecimento do aluno, que possuem a capacidade de compreender determinadas situações e reagir adequadamente a estas. Já as estratégias afetivas são aquelas que levam em consideração os estados afetivos do aluno com o objetivo de determinar formas de motivar o aluno, tornando-as parte do processo de aprendizagem, sendo possível representá-las através de comportamentos verbais e corporais do aluno. A aplicação dessas estratégias será baseada nos estados emocionais inferidos dos alunos (crianças), a partir da conversação entre eles e deles com o sistema. Essa inferência terá como base a teoria dos atos de fala (SEARLE, 1969; BERGER, 2005) e o trabalho proposto por (LEONHARDT, 2009) que trata da comunicação afetiva nos alunos. Como também, as pesquisas do grupo de pesquisa do LELIC, que são baseadas em estudo da linguagem, terão importante papel no desenvolvimento deste trabalho de doutorado, pois será possível entender melhor os processos de comunicação dos alunos, para posteriormente serem implantados no agente pedagógico animado.

Na área de agentes conversacionais busca-se tornar a comunicação entre os agentes e os usuários, bem como as ações de comunicação e o modelo do agente, o mais próximo possível da comunicação humana. Dessa maneira, para que o agente pedagógico animado tenha capacidade de compreender o aluno, é necessário construir um modelo da mente desse, considerando suas emoções, personalidade e características que possam ser consideradas importantes, tais como, estilo cognitivo. Para o reconhecimento de emoções, pretende-se utilizar o trabalho proposto por (BERGER, 2005), que abrange a montagem de uma linguagem para agentes conversacionais respeitando os atos de fala e integrando elementos essenciais em uma conversação, bem como, os trabalhos já existentes no LELIC e estudos em campo, através de filmagem das ações dos alunos durante a utilização do jogo.

O agente pedagógico animado será inserido no jogo Città, sendo denominado de Maga Vitta. Este agente será aplicado como estudo de caso do que se pretende desenvolver, salientando que, como as estratégias cognitivas e afetivas e, a inferência de emoções são fortemente baseadas no contexto e no domínio do Projeto Civitas, todos os resultados serão específicos ao agente MagaVitta. Este agente (AXT, 2008) já possui uma arquitetura padrão de agente autônomo inteligente, composta por quatro principais módulos: interpretador de ações (modelo da mente do agente), modelador do agente conversacional (captura o modelo e decide qual ação realizar gerando um conjunto de características que resultarão na fala e nas expressões faciais, que fazem parte do modelo do agente), organizador dos planos de ação (planeja o conjunto de ações e interações determinadas para o agente conversacional, gerando a ordem de sincronização) e o gerador de expressões faciais (mostra os resultados). Alguns desses módulos estão sendo desenvolvidos pelo grupo de pesquisa, sendo que o modelo da mente do agente, que determina as estratégias afetivas e cognitivas integrando informação sobre conhecimento do aluno e sobre seus estados emocionais, assim como a inferência de emoções baseada em atos de fala, será desenvolvido nesta tese de doutorado.

Como procedimento metodológico da pesquisa, inicialmente será realizada a pesquisa bibliográfica envolvendo os estados afetivos, os traços de personalidade e as teorias que estão relacionadas com a importância das condutas sociais no processo de aprendizagem. A partir disso, será feito um *corpus* de gravações de vídeo com os alunos interagindo com o ambiente Città para posterior análise dos estados mentais, cognitivos e afetivos, dos mesmos. Através da análise deste material serão definidas as estratégias cognitivas e afetivas. Com esses dados o modelo da mente de um agente pedagógico emocional será definido e desenvolvido, posteriormente, este modelo será aplicado como estudo de caso no agente Maga Vitta. Além

disso, testes com o uso do agente serão realizados com alunos do Ensino Fundamental, a fim de validar a aplicabilidade do que foi desenvolvido.

A principal contribuição deste trabalho é a integração de estratégias de competência e afetivas que irão compor o modelo da mente de um agente pedagógico emocional. Outro diferencial deste trabalho envolve a coleta de estratégias durante a interação dos alunos no ambiente, utilizando materiais diversificados, como vídeos e escuta desses alunos e, o estudo de trabalhos já existentes no LELIC. A partir de toda essa informação, o agente poderá fazer análises e gerar desafios para os alunos baseando-se em dados reais levantados e não somente em estudos teóricos como vem acontecendo nos trabalhos de pesquisa em agentes conversacionais emocionais.

Como contribuição adicional pode ser citada a integração do projeto de doutorado com o projeto Civitas, que possui como uma das etapas, o desenvolvimento do agente para o ambiente do jogo Città, utilizando os processos metodológicos que permitem atuar na sala de aula. Uma segunda contribuição é o desenvolvimento da área de Inteligência Artificial, que não está sendo trabalhada no projeto e é considerada uma necessidade pelos integrantes do grupo. A aplicação de técnicas de Inteligência Artificial irá tornar o jogo mais complexo, contribuindo para o desenvolvimento do conhecimento dos alunos. E, por fim, irá integrar a escola com a área de pesquisa.

6. Referências Bibliográficas

AXT, M. et al. Maga Vitta: conversational ecological agent in an interactive collective construction environment for basic education. In VICARI, Rosa; JAQUES, Patricia A. (Orgs.). **Agents-based Tutoring Systems for Affective and Cognitive Modelling**. Ed. Idea, 2008, chap. IV.

BERCHT, Magda. **Em Direção a Agentes Pedagógicos com Dimensões Afetivas**. Tese (Doutorado em Ciência da Computação). Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2001.

BERGER, Alexandra; PESTY, Sylvie. **Towards a Conversational Language for Artificial Agents in Mixed Community.** In Fourth International Central and Eastern European Conference on Multi-Agent Systems CEEMAS'05, Springer Verlag, Budapest, 2005, pages 31-40.

BURLESON, Winslow. **Affective Learning Companions:** Strategies for Empathetic Agents with Real-Time Multimodal Affective Sensing to Foster Meta-Cognitive and Meta-Affective Approaches to Learning, Motivation, and Perseverance. MIT PhD Thesis, 2006.

CONATI, Cristina. **Probabilistic Assessment of User's Emotions in Educational Games.** Applied Artificial Intelligence, Philadelphia, v. 16, n. 7-8, 2002.

CUTMORE, T. R. H. et al. **Cognitive Ad Gender Factors Influencing Navigation in Virtual Environment.** International Journal of Human-Computer Studies, 53, 2000.

GOLEMAN, Daniel. **Emotional Intelligence**, New York: Bantam Books, 1995.

GRATCH, Jonathan.; MARSELLA, Stacy. **Tears and Fears: Modelling Emotions and Emotional Behaviors in Synthetic Agents.** In: International Conference on Autonomous Agents and Multiagents Systems. **Proceedings.** Montreal: ACM Press, 2001.

IZARD, Carrol E. Emotion-cognition relationships and human development. In: IZARD, C.; KAGAN, J.; ZAJONC, R.B. (Ed.). **Emotions, cognition, and behavior.** New York: Cambridge University Press, 1984. p. 17-37.

JAQUES, Patricia A. **Using an Animated Pedagogical Agent to Interact Affectively with the Student.** 2004. Tese (Doutorado em Ciência da Computação). Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2004.

JOHN, Oliver P. The "Big Five" factor taxonomy: Dimensions of personality in the natural language and in questionnaires. In: PERVIN, Lawrence. A. (Ed.). **Handbook of personality: Theory and research**. New York: Guilford, 1999.

LEONAHRDT, M.; VICARI, R.; PESTY, S. **Enhancing Affective Communication in Embodied Conversational Agents.** In COLIBRI - Colóquio em Informática, 2009, Bento Gonçalves. [Colibri] - Anais do Colóquio em Informática: Brasil / INRIA, Cooperações, Avanços e Desafios, 2009.

LESTER, J. et al. **Cosmo: A life-like animated pedagogical agent with deictic believability.** In: IJCAI97 Workshop on Animated Interface Agents: Making them Intelligent. **Proceedings.** Nagoya, 1997.

McQUINGGAN, Scott W.; LESTER, James C. **Modeling and evaluating empathy in embodied companion agents.** In International Journal of Human-Computer Studies/International Journal of Man-Machine Studies. 65 (4): 348-360, 2007.

MÜLLER, D. N. ET AL. **Cidades virtuais como ambiente educacional colaborativo**. In: Digital Games 2008. **Proceedings.** Porto/Portugal, 2008.

ORTONY, A.; CLORE, G. L.; COLLINS, A. **The cognitive structure of emotions**. Cambridge: Cambridge University Press, 1988.

PELACHAUD, Catherine. Contextually Embodied Agents. In MAGNENAT-THALMANN, N. THALMANN, D. (eds.). **Deformable Avatars**. Kluwer Publishers, 2001.

PICARD, Rosalind. **Affective Computing.** Cambridge: MIT Press, 1997.

SCHERER, Klaus. Appraisal Theory. In DALGLEISH, T.; POWER, M. (ed.). **Handbook of Cognition and Emotion.** New York: John Wiley & Sons, 1999.

SEARLE, John. **Speech Acts.** Cambridge, Cambridge University Press, 1969, 203 p.

VILHJALMSSON, H. et al. **The behavior markup language: recent developments and challenges.** In 7tg International Conference on Intelligent Virtual Agents: Electronic Notes in Artificial Intelligence. **Proceedings.** Berlin: Springer, 2007. p. 90-111.

VYGOTSKY, Lev S. **Thought and Language. Cambridge**. MA: MIT Press, 1962.

WEIZENBAUM, Joseph. **ELIZA - A Computer Program for the Study of Natural Language Communication between Man and Machine**. Communication of the ACM. 25th Anniversary Issue. 26 (1): 23-28. Cambridge, MA. USA, 1983.

YUNG, Hsin. I. **Effects of animated agent with instructional strategies in facilitating student achievement of different educational objectives in multimedia learning.** In Journal of Educational Multimedia and Hypermedia, 18(4), 453-466. Chesapeake, VA: AACE, 2009.

Construindo Cidades Virtuais com o Civitas Città

Pelas bolsistas Andréa Konzen e Maribel Selli
Colaboração do doutorando Daniel Epstein
(Informática na Educação/UFRGS)
e do coordenador Daniel Müller

1. Introdução

Ao pensarmos as diferentes práticas que coexistem na escola podemos perceber que, em muitas situações, parecemos precisar sempre algo a mais, irmos além do que já viemos fazendo e propondo como práticas de ensino para que se efetivem os processos de ensino e construção do conhecimento.

O Projeto Civitas propõe, através da formação de professores em serviço, movimentos que produzam deslocamentos, sem que as práticas até então desenvolvidas sejam abandonadas. O professor assume outro fazer a partir do momento em que decide participar do grupo que, na interação com os outros sujeitos que o compõem, busca outras práticas, vislumbrando diferentes possibilidades de realizar o que já vinha fazendo.

A partir do Civitas instaura-se na escola um cenário que proporciona a aprendizagem pela pesquisa, pela descoberta, pela invenção de problemas, desafiando as crianças a serem atores e autores de seu conhecimento pela produção de sentidos e autoria.

Para Axt e Martins (2008) essa proposta de formação situa-se num campo de correlação de forças estabelecidas entre as macros e micropolíticas considerando-se que não existe nem algo mais nem menos eficaz, mas o diferente. Busca-se assim a coexistência de práticas em que o professor pode continuar desenvolvendo o que vinha fazendo até então, mas ao sentir-se convidado a experimentar outras práticas fica imerso nesse outro olhar sobre possibilidades de ensinar percebendo todos os sujeitos que fazem parte desse processo como importantes e imprescindíveis.

Os professores e as crianças planejam e constroem juntos os trajetos a serem percorridos para que a construção do conhecimento aconteça descobrindo novos modos de ser e estar na escola.

Professores e crianças experimentam outros fazeres e formas de aprender que se tornam mais significativas e próximas ao seu contexto. Isso tem acontecido porque as crianças sentem-se parte de seu aprender, pois percebem que também têm muito a contribuir e ensinar sobre o que sabem e viveram até então sentindo-se reconhecidos e importantes nas relações dialógicas que se constroem no cotidiano da escola.

Para Bakhtin (2003) o diálogo é constituído por uma incompletude. Assim, não existe um fim nos enunciados que se produzem num diálogo. Isso ocorre, pois as enunciações que se produzem buscam outros enunciados, numa relação de inacabamento. Sempre haverão novos sentidos sendo produzidos o que constitui uma cadeia de enunciados que podem retornar ao diálogo num processo dialógico ou em outros momentos de enunciação. Os sentidos produzidos ficam reverberando e, com isso, retornam ao diálogo sendo expressos de outras formas produzindo assim novos enunciados.

Entendemos que podem ocorrer mudanças, inovações e outras formas de se fazer educação, partindo das dúvidas, incertezas e inquietações que o professor enfrenta em seu cotidiano. O Civitas, partindo do pressuposto dialógico em que todos os envolvidos no processo

são sujeitos importantes para que, tanto as relações dialógicas como a construção do conhecimento aconteçam, parte dessa propositura.

A reflexão, que se dá a partir da presença/importância do outro enfatizando sua importância/necessidade de sua presença e proximidade com para que cada um e cada uma experimentem momentos de interação dialógica produzindo sentidos e aprendizagem pela composição (est) ética que se dá nessa relação eu/outro.

1.1. Projeto CIVITAS

O Projeto Civitas surgiu em 2002 com uma proposta inicial do desenvolvimento de um software livre para a simulação de cidades como mais uma tecnologia que pudesse compor o cenário da sala de aula (Axt et al, 2002; Axt, 2003), o Città que será abordado de forma mais específica no decorrer deste artigo. Para tal se pensou, a partir de uma proposta de formação, problematizar a prática cotidiana em conjunto com os professores através de uma proposta metodológica embasada no pressuposto dialógico partindo da invenção de problemas, pesquisa, invenção, criação, imaginação e fabulação.

Temos como referencial teórico as contribuições de Mikhail Bakhtin, Gilles Deleuze, Félix Guattari e Henri Bergson.

Como a proposta é partir sempre da realidade e das necessidades dos sujeitos que compõem os grupos surgiram outras demandas. O projeto foi ampliado a profissionais desde a Educação Infantil até o Quinto Ano dos Anos Iniciais e também aos gestores de escolas e da SMED.

Temos percebido significativas mudanças na prática das professoras envolvidas, pois a alegria e os acontecimentos que permeiam o dia a dia de sua sala de aula as contagiam e também seus alunos. Os processos vivenciamos transbordam os muros da escola envolvendo as famílias e a comunidade. Os familiares das crianças envolvem-se com o aprendizado dos seus filhos participando de sua vida escolar e das atividades que acontecem na escola.

A proposta metodológica que surge a partir do Civitas tem sido permeada pelo inusitado, pela alegria e envolvimento das professoras, crianças, famílias e comunidade escolar, pois os intercessores, sejam eles a cidade, sejam os personagens conceituais ultrapassam os limites da sala de aula e da escola e passam a fazer parte da vida desses sujeitos e de seu cotidiano.

Em se tratando mais especificamente do quarto ano, o editor de cidades Città causou um grande impacto positivo na sala de aula, pois chega como mais um recurso pedagógico e tecnológico para coexistir com outras tecnologias e recursos já utilizados anteriormente.

Ao passar a existir na vida das crianças o editor de cidades Città opera como um disparador de ideias, possibilidades e acontecimentos. Elas cooperam, buscam soluções para a construção da cidade em grupos, discutem com os outros grupos as melhores decisões. Num processo em que a criação, fabulação, invenção de problemas e imaginação são presentes dão forma aos seus projetos e criam as cidades que os ajudam a estudar e pensar a cidade que vivem buscando alternativas para solucionar os problemas vivenciados.

2. O jogo de simulação Città

O desenvolvimento do jogo de simulação Città é a versão digital do simulacro da cidade que ocorre na sala de aula com alunos do quarto ano do Ensino Fundamental atendidos pelo Projeto Civitas. As crianças reinventam cidades imaginárias onde eles são cidadãos e tornam-se responsáveis por todos os setores da sociedade.

Além da construção das cidades com papelão, isopor e argila, elas agora também estão no formato digital. A cidade física construída pelas crianças fica ainda sobre uma mesa no canto da

sala, mas a turma pode construir muitas outras cidades, muito maiores, sem ocupar espaço físico, sem ser desgastada pelo manuseio, e ainda perpetuar-se no tempo, podendo ser constantemente aprimorada ou apenas visitada pelas gerações futuras. Isso porque agora no as cidades também existem em meio digital.

Através do Città, a criança constrói prédios e vias, modela terrenos, troca cores e apaga os objetos que não servem mais. Também está sendo testado um agente inteligente para questionar a criança sobre a sustentabilidade da cidade (ver seção AA).

O Città está em constante evolução, onde o Projeto Città Cosmopolita[9] é o resultado de um ciclo de dez anos de estudos e ações contínuas executadas no Projeto Civitas. Por esse motivo a releitura e reelaboração do Città é uma prática comum nas equipes destes projetos. Nas seções a seguir tem-se mais uma versão otimizada a partir do diálogo entre o mundo computacional e o pedagógico.

2.1. Módulo de Interface

Até 2011 o Città era desenvolvido em Java, e a interface do Città havia sido totalmente remodelada e seus objetos tridimensionais foram redesenhados frente à versão anterior. Com base nos relatos de experiências com as crianças das escolas nas quais o Città foi apresentado, percebeu-se que havia necessidade de melhorar a interface gráfica primeiramente para facilitar a compreensão.

Ao implementar a nova interface, notou-se as limitações que a falta de conhecimento sobre a *engine* gráfica e bibliotecas não recentemente atualizadas implicariam em futuras atualizações mais complexas como animações. Foi considerada a utilização de uma *engine* completamente diferente, optando-se pelo Blender.

2.2. Nova plataforma de desenvolvimento: Blender

A equipe do Città já trabalhava com o Blender para modelagem dos objetos em 3D e decidiu-se usá-lo como plataforma de desenvolvimento, apoiada pela linguagem Python, integrada no Blender. Desta forma seria possível excluir boa parte dos problemas de compatibilidade, tanto na criação dos objetos quanto na execução do Città, e dispor de uma ferramenta ainda mais potente, estável e que poderíamos ter mais controle sobre o próprio desenvolvimento do software, sem o uso de diversos programas para adaptação.

Desta forma, o Città está mais estável e rápido na versão Blender, sendo definido na programação a quantidade de frames que rodará o jogo. E o jogo passará a rodar em Windows, Linux e Mac sem nenhum problema, além de ser possível seu uso em navegadores web.

Pontos decisivos para escolhermos o Blender:

- Programa em software livre que roda em Windows, Linux e Mac, além de terem sido desenvolvidos plugins para rodar um arquivo blender em navegadores web, como o Burster[10], e o Blenderplayer para Android, permitindo também a execução em dispositivos móveis.
- Agilidade no processo do jogo, pois temos todas as ferramentas em um programa.
- Possibilidade de construção da abordagem multijogador, a exemplo do jogo WSAG[11], e a

[9] O projeto *Città Cosmopolita* é financiado com o apoio do CNPq – Conselho Nacional de Desenvolvimento Científico e Tecnológico.

[10] http://geta3d.com
[11] http://www.oldjim.ch/WSAG

constituição de redes de cidades.

Dentro dessa nova concepção do Città no Blender, estão sendo previstos novos recursos como a leitura de dados de relevo reais para modelagem do terreno da cidade. Isso significa que uma cidade poderá ser reconstruída com os dados reais em relação ao seu relevo. Outro recurso previso com o multijogador e com a interface web é a geolocalização. Através dela será possível o mapeamento das diferentes culturas, e da localização dos grupos de alunos, suas escolas e cidades (imaginadas ou reais).

2.3. Banco de Dados

O banco de dados livre SQLite[12] foi escolhido por ser de fácil distribuição dentro com os pacotes Java do Città. Durante seu desenvolvimento, foi discutido um método mais otimizado de gravação de mapas, que também facilitaria a implementação de um histórico de eventos também mais otimizado. Este novo método consiste em um arquivo de banco de dados para cada mapa, contendo um registro do estado atual do mapa e um registro de eventos.

O registro do mapa serve para uma rápida reconstrução do mundo de jogo, independente de quantas alterações foram feitas até agora naquele mapa. O registro de eventos grava a "história" daquele mapa, indicando a data de jogo, o tipo do evento (construção/destruição/"biografia da construção") e um texto, descrevendo o evento em detalhes.

Foi definido também o critério de salvamento dos mapas. Os mapas são salvos assim que o jogador decide sair do mapa atual, não há necessidade de navegar menus ou apertar outros botões.

2.4. Reconhecimento e síntese de fala

O módulo de voz consiste no reconhecimento de voz da criança usuária do Città e a síntese do agente inteligente Maga Vitta.

O reconhecimento de voz foi continuado com o aperfeiçoamento do modelo linguístico e teste de palavras para reconhecimento, sempre dentro do modelo dos sistemas de software livre *HTK*[13] (treinamento estocástico) e *Julius*[14] (reconhecimento de voz). Neste sentido, foram feitas interações com bolsistas do Projeto Comunica[15], do Instituto de Informática da UFRGS, e com membros do Projeto FalaBrasil[16] da UFPA, no sentido de otimizar o reconhecimento.

A integração do módulo de reconhecimento de voz através do Julius ocorreu parcialmente através da inserção de botões de controle de gravação de voz ("Começar gravação" e "Parar gravação") na interface do chat com a Maga. No entanto, o módulo de reconhecimento de voz não está funcional em conjunto com os outros módulos. Foi cogitado o uso de uma ferramenta semelhante ao *grep* (GNU Project) para buscar as palavras-chave capturadas pelo Julius e repassá-las ao assistente do jogo.

Quanto à síntese de fala, utilizamos o sistema de software livre *MaryTTS*[17], também com apoio da equipe do Projeto Comunica, que desenvolveu um modelo acústico com voz feminina,

---

[12] http://www.sqlite.org

[13] http://htk.eng.cam.ac.uk

[14] http://julius.sourceforge.jp/en_index.php

[15] http://comunica.conexum.com.br

[16] http://www.laps.ufpa.br/falabrasil

[17] http://mary.dfki.de

adequada à Maga Vitta. MaryTTS é uma solução completa e configurável para síntese, incluindo a construção do modelo acústico para voz, o léxico de fonemas e a parametrização de itens como a prosódia (entonação) e velocidade de leitura (duração). Ele utiliza um padrão de serviço internet (WebService) para receber um texto e enviar um arquivo de áudio. Ao iniciar o Città é necessário ativar o servidor (*maryserver*) e, quando necessário, apenas solicitar o arquivo de áudio para determinada frase. Para tanto, foi adicionada uma chamada de sistema para o servidor, ao início do Città, e uma classe Java (*MaryClientUser*) para a síntese, quando esta for solicitada.

Os modelos acústicos de síntese para o Português que vêm com o MaryTTS são primitivos, proporcionando uma voz visivelmente mecânica e de difícil audição. Neste sentido, foi necessária a gravação de mais de 600 frases (locução feminina) e geração de um léxico de 700 palavras com sua anotação fonética para treinamento do modelo acústico. Este modelo proporcionou uma síntese com boa audição, embora ainda possa melhorar, através do ajuste de parâmetros de saída do MaryTTS.

3. Agente pedagógico Maga Vitta

Sistemas tutores inteligentes são definidos como sistemas de computador cujo objetivo é prover uma interação com o usuário de forma adaptativa (Psotka e Mutter 1988). São comumente utilizados para fornecer instruções ou respostas às ações realizadas pelo usuário. Eles devem ser capazes de se adaptarem as interações do usuário tanto em relação ao conteúdo e conhecimento quanto em relação a questões emocionais (D'Mello et. al. 2007) e estilos de aprendizagem (Yannibelli et. al. 2006). Dentre as diversas formas com que o sistema pode ser apresentado é destacada a de agente pedagógico. O uso de agentes pedagógicos estimula os estudantes, facilitando o aprendizado e incentivando o estudo (Craig et. al. 2002).

Um agente pedagógico é um sistema que opera de forma continua em um ambiente de aprendizagem, fornecendo suporte aos estudantes. Dentre os benefícios de se utilizar um agente pedagógico está o aumento na motivação dos usuários, um aperfeiçoamento em questões de interação e comunicação e um melhor entendimento de problemas e resolução de problemas (Gulz 2004). Conforme Kim e Baylor (Kim e Baylor 2006) agentes pedagógicos podem ser desenvolvidos para atuar de forma similar aos papeis de atuação de uma pessoa, sendo um tutor (Graesser 2001), professor (Baylor e Kim 2006), motivador ou colega (Kim e Baylor 2006).

A teoria sociocultural considera a aprendizagem como sendo um processo regulatório, mediado por interações sociais, objetos e texto (Jonassen et. al. 2005). Nesse âmbito, um agente pedagógico pode ser entendido como uma ferramenta capaz de mediar o aprendizado (Pinho et. al. 2013). Além disso, a intervenção do agente pedagógico deve ser realizada de forma a auxiliar os estudantes a desenvolverem esse processo regulatório no devido contexto, promovendo assim a autonomia deles e favorecendo o pensamento autônomo (Wood et. al. 1976). Uma vez que os erros são percebidos pela teoria sociocultural como parte natural do processo de aprendizagem, as intervenções do agente devem ser no sentido de permitir ao usuário compreender o problema e achar uma solução própria, favorecendo a reflexão, motivação a aprendizagem.

O agente pedagógico Maga Vitta (MV) é um agente que assiste aos usuários do jogo educacional e simulador de cidades Città. Ele tem por objetivo auxiliar estes usuários a desenvolverem cidades que sejam ecologicamente sustentáveis e que possuam os recursos necessários para seu desenvolvimento. O enfoque do agente está em garantir que os níveis de diversos recursos necessários à cidade estejam dentro do valor considerado ideal. Estes recursos são:

- Quantidade de alimento que a cidade produz e armazena;
- Quantidade de água potável da cidade;
- Qualidade do atendimento médico aos habitantes da cidade;

- Quantidade de energia produzida e utilizada;
- Se há tratamento de lixo e se esta quantidade de tratamento é suficiente;
- Se o esgoto da cidade está sendo adequadamente tratado;
- Se há espaço para a cidade continuar a se desenvolver.

A cada ação do usuário, um novo objeto pode ser adicionado ou removido da cidade. Diferentes objetos e estruturas causam diferentes reações, podendo estes objetos ser benéficos a alguns recursos e maléficos a outros. O correto equilíbrio de construções é objetivado de forma a permitir que a cidade se desenvolva sem que haja problemas para a população ou o meio ambiente. Quando certos recursos da cidade não são suficientes para o bom funcionamento da mesma, o agente Maga Vitta apresenta uma intervenção, buscando que o usuário possa identificar qual recurso está sendo negligenciado e corrigir o problema.

O agente MV foi concebido para operar de forma a permitir que o usuário construa e desenvolva sua cidade de forma individual ou coletiva. É possível que múltiplos usuários em diferentes computadores desenvolvam uma mesma cidade e se comuniquem entre si. Esta perspectiva educativa fomenta a interação entre os estudantes e o processo de aprendizagem sociocultural (Vygotsky 1978, 1986). Outro aspecto que foi considerado no desenvolvimento do agente é que as escolas onde é proposto utilizar o simulador (e, consequentemente, o MV) nem sempre possuem computadores com avançado padrão tecnológico. É comum encontrar em escolas públicas computadores tecnologicamente defasados.

Pensando nessas características, o agente Maga Vitta está desenvolvido em dois módulos: módulo cliente e módulo servidor. A divisão do agente em módulos possibilita dividir o custo de processamento das informações entre os computadores disponíveis com diferentes recursos. Mesmo em escolas com computadores mais antigos, o acesso à internet ou um servidor central mais avançado permite que o agente opere na sua total funcionalidade através do uso do módulo servidor.

3.1. Desenvolvimento do agente Maga Vitta

Para o correto funcionamento do agente, é necessário que este tenha conhecimento de todas as ações que ocorreram no simulador, sejam elas de adição ou exclusão de objetos. O principal suporte do agente será o banco de dados no qual o simulador registrará todas as ações realizadas na cidade. A cada ação registrada, o agente buscará saber o que foi feito e as informações sobre as estruturas modificas.

Esta interação será realizada pelo módulo cliente do agente. O módulo cliente está localizado na mesma máquina onde o simulador está sendo executado. Ele não possui um grande custo de processamento associado, tendo em vista que a sua função principal é apenas identificar que houve uma alteração no banco de dados e buscar as informações referentes a esta alteração. Posteriormente, ele também servirá para interação com o usuário, enviando informações ao banco de dados que serão interpretadas pelo simulador como uma intervenção do agente MV. Novamente, o simples acesso ao banco de dados permite que o agente esteja presente na mesma máquina onde se está rodando o simulador sem representar um custo computacional relevante.

Uma vez que as informações tenham sido identificadas no banco de dados, elas são enviadas através de mensagens ao módulo servidor. Caso a escola onde se está sendo utilizado o simulador possua acesso à internet, o módulo servidor será um computador dedicado ao agente, tendo alto poder computacional e capaz de processar diversas requisições simultaneamente. Caso não seja possível enviar informações pela internet, o servidor será local, podendo até mesmo ser a própria máquina onde o simulador está rodando. Isso impediria que múltiplos usuários trabalhassem sobre a mesma cidade, mas permitiria o uso individual do simulador e de todas as

suas características.

O principal objetivo do módulo servidor será analisar os dados recebidos e identificar se a cidade apresenta algum déficit nos seus recursos. Ele possui um banco de dados próprio, permitindo que identifique de forma única cada simulação e tenha controle de todas as modificações realizadas em cada cidade. Ao receber a mensagem do módulo cliente com as informações mais recentes do simulador, o módulo servidor identifica se há problemas na cidade e, caso haja, envia uma mensagem de volta ao módulo cliente informando que tipo de intervenção o agente deve realizar.

As intervenções realizadas pelo agente Maga Vitta tendem apenas a indicar aos usuários que algum aspecto da cidade não está funcionando da forma planejada. Estas intervenções não devem indicar diretamente ao usuário o que deve ser alterado, mas devem apenas incentivar que ele reflita sobre o problema e encontre a solução (Pinho et. al. 2013). Através das intervenções é esperado que o usuário corrija o problema de acordo com a sua experiência e conhecimento, buscando refletir sobre o que está sendo realizado.

A interação com o usuário atualmente está ocorrendo de forma textual. Mensagens são trocadas entre o usuário e o agente, sendo previsto que esta comunicação possa ser incrementada em um projeto futuro com o uso de reconhecimento de voz e áudio, a fim de tornar mais natural e acessível os recursos do agente. Diversas mensagens podem ser enviadas para o mesmo tipo de problema, de forma a evitar que o agente se torne repetitivo e desinteressante aos usuários. Também é previsto que mensagens diferentes sejam enviadas dependendo da gravidade do problema encontrado.

O agente afetivo

Segundo Bercht (2001), a ideia de que razão e emoção são coisas distintas tem sido discutida desde Sócrates nos anos de 470 a 399 a.c e perdurou até o final do século XIX. Porém, no ultimo século, novos estudos de diversos pesquisadores (Piaget, Damásio e Le Doux, Vygotsky) acabaram por reescrever essa suposição. A afetividade passou a ser vista como parte do processo cognitivo.

Segundo Longhi (2008), embora muitos pensadores, nas diversas épocas, insistissem nas relações existentes entre razão e emoção, apenas nas duas últimas décadas, importantes pesquisas em Neurociência, Psicologia e Ciências Cognitivas têm demonstrado como a afetividade está interligada de forma complexa com a cognição, constituindo papel fundamental em funções como tomada de decisão, memorização, e criatividade.

Para Picard (1997), a Computação Afetiva é a “computação que está relacionada com, que surge de ou deliberadamente influencia emoções”. Assim, se quisermos computadores genuinamente inteligentes, adaptáveis às nossas necessidades e que interajam de forma natural conosco, então estes computadores precisarão de habilidades para reconhecer e expressar emoções, ter emoções ou até mesmo possuir inteligência emocional.

Isso é especialmente útil para sistemas educacionais e seus agentes pedagógicos. Ao dar características afetivas a esses agentes, espera-se que tanto a interação com estudantes quanto a aprendizagem dos mesmos seja melhorada, pois um agente que além de ensinar também expresse emoções cria um vínculo mais íntimo com seu interlocutor. Quando esse vínculo é explorado, fica mais fácil motivar o estudante a se esforçar ou focar nos exercícios propostos (Longhi 2008).

O agente Maga Vita pode ser considerado um companheiro virtual no ambiente do jogo. Esse agente foi inicialmente proposto em Axt (2008). Sua definição é de um agente autônomo ECA (*Embodied Conversational Agent*) dotado de capacidades afetivas e que auxilia usuários durante o processo de construção e/ou simulação de um ambiente, neste caso, uma cidade virtual.

As emoções mostradas são baseadas em uma avaliação individual ou na avaliação de um significado cognitivo de um evento conforme descrito dentro do modelo OCC (Axt 2008) proposto por Ortony (1988). Neste modelo são analisadas as ações do usuário ao dispor os elementos da cidade propondo sugestões, questionamentos e informando sobre o processo de construção da cidade.

Seu padrão de comportamento foi alinhado para estar constantemente preocupado com o meio ambiente. Assim ele memoriza as ações que possam refletir no meio ambiente e adverte sobre possíveis tragédias ambientais (Longui 2004). O agente não oferece soluções. Ao invés disso, ele mostrará as futuras implicações políticas e éticas ou interpretará as possíveis representações dos habitantes insatisfeitos com o planejamento urbano tomado na construção da cidade.

Em sua arquitetura, o agente acaba mesclando noções de dois modelos cognitivos: o tradicional modelo OCC (Ortony 1988) que contém o *appraisal* das principais emoções inferidas pelo agente, e o modelo de Rosemann (1990), que considera a emoção surpresa que não é considerada pelo modelo OCC. Dessa forma, as emoções do agente são: Alegria, Tristeza, Preocupação, Surpresa, Raiva, Nojo e Medo.

O modelo conversacional do agente é baseado no modelo proposto por Cassel (2001) conhecido como FMBT (*Functions, Modalities, Behaviors, Time*), que incorpora as principais características presentes numa conversação.

Segundo Cassel (2001), esse modelo caracteriza-se por possuir: (1) *Functions*, funções proposicionais e funções interacionais que correspondem, respectivamente, ao conteúdo da conversa como a fala ou a entonação da voz e as interjeições que regulam o processo de conversação, que podem ser definidas como os comportamentos sonoros, como *hã?*, *óóó!*, *hein?*, *iiih*, ou os não sonoros, como movimento de ombros, mãos e face que podem indicar dúvida ou alegria, por exemplo; (2) *Modalities*. Modalidades de comunicação, que constroem as funções proposicionais e interacionais, podendo ser *verbais* como palavras ou cantos e *não-verbais* como gestos faciais como piscar os olhos, por exemplo, ou gestos comportamentais como gesticular as mãos ou olhar atentamente alguém; (3) *Behaviors.* Comportamentos de uma conversação onde, por exemplo, baixar a cabeça pode ser um gesto de concordância ou uma saudação; (4) *Time*. Tempo e sincronização envolvendo uma conversação onde, por exemplo, uma fala excessiva ou um sinal sendo realizado diversas vezes pode, eventualmente, indicar ansiedade, nervosismo ou tédio, bem como a demora em fornecer uma resposta, pode ser interpretada como não cooperação, desatenção ou desistência.

A arquitetura do agente afetivo divide-se em quatro módulos principais, conforme mostrados na Figura 1: Interpretador de Ações, Modelador do ECA, Organizador dos Planos de Ações do ECA e Gerador de Expressões Faciais.

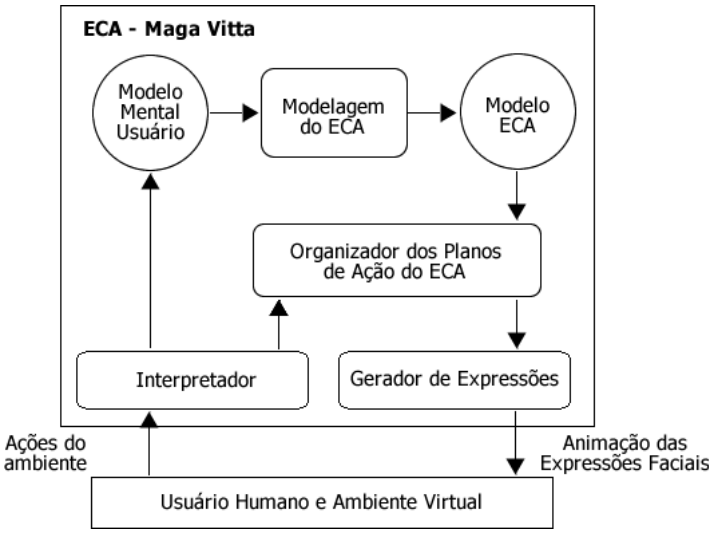


Figura 1 Arquitetura do agente Maga Vitta.

Para que o agente entenda o usuário, o agente constrói um Modelo do Usuário. Este modelo é obtido através da interpretação das ações do aluno durante a construção da cidade virtual. Esse modelo é guardado no perfil do usuário junto com outras características observadas durante a manipulação do ambiente virtual (Longui 2004).

O Modelador ECA do agente possui a estratégia de sempre capturar o modelo do usuário para poder gerar uma coleção de recursos de fala e de movimentos faciais. O Organizador de Planos de Ação usa o modelo ECA gerado pelo Modelador ECA para organizar a coleção de ações do agente.

Referências

Axt, Margarete. (2003) **CIVITAS, a cidade viva: ou do espaço de invenção do educador na escola.** In: Neusa Guareschi. (Org.). Estratégias de invenção do presente: a Psicologia Social no contemporâneo, p. 67-82, EDIPUCRS, Porto Alegre, 1 ed., v. único.

Axt, Margarete; Lopes, E. C. ; Freimuller, M. (2002) **Civitas: o que as crianças escrevem quando pensam Cidades Virtuais.** In: XI EAIC - Encontro Anual de Iniciação Científica, 2002, Maringá - Paraná. Anais do XI EAIC - Encontro Anual de Iniciação Científica.

Axt, Margarete; Martins, Márcio André Rodrigues. (2005) **CIVITAS: abrindo espaços de invenção na escola.** In: MOLL, Jaqueline (org.). **Múltiplos alfabetismos: diálogos com a escola pública na formação de professores.** Porto Alegre: Editora da UFRGS, p. 17-39.

AXT, Margarete; MARTINS, Márcio André Rodrigues. (2008) **Coexistir na diferença: de quando a formação em serviço pensa modos de habitar a sala de aula.** In: TRINDADE, Iole Maria Faviero (org.). **Múltiplas alfabetizações e alfabetismos**. Porto Alegre: Editora da UFRGS.

Axt, M. and Longhi, M. T. and Silveira, P. D. and Guimarães, L. N. (2008) "**MagaVitta: Conversational Ecological Agent in a Interactive Collective Construction Environment for Basic Education", In:** Agent-Based Tutoring Systems by Cognitive and Affective Modeling, Edited by Viccari, R. M. and Jaques, P. A. and Verdin, R. **Porto Alegre.**

Axt, M.; Ferreira Filho, R. C. M.; Balle, E. E. Rodrigues, S. S.; Müller, D. N. **Cidades virtuais: tecnologias para aprendizagem e simulação.** In: IV Seminário Jogos eletrônica educação e comunicação: construindo novas trilhas, 2008, Salvador.

Bakhtin, M. M. **Para uma filosofia do ato**. Tradução de Carlos Alberto Faraco e Cristovão Tezza da edição americana *Toward a philosophy of the act.* Austin: University of Texas Press, 1993. (tradução destinada exclusivamente para uso didático e acadêmico).

_____. **Problemas da Poética em Dostoievski**. 3ª ed. São Paulo: Forense Universitária, 2002.

_____. **Estética da Criação Verbal.** 2ª ed. São Paulo: Martins Fontes, 2003. 421 p.

_____. **Marxismo e Filosofia da Linguagem**. 12ª ed. São Paulo: HUCITEC, 2006.

Baylor, A. L.; Kim, Y. (2006) **Simulating instructional roles through pedagogical agents.** In: International Journal of Artificial Intelligence in Education, 15.

Bercht, M. (2001) **Direção a Agentes Pedagógicos com Dimensões Afetivas.** Porto Alegre, Brasil. Academic Press.

Cassel, J. (2001) **Embodied Conversational Agents: Representation and Intelligence in User Interface.** In: AI Magazine, Vol. 22, pages 67-83. Cambridge, USA.

Craig, S. D.; Gholson, B.; Driscoll, D. M. (2002) **Animated Pedagogical Agents in Multimedia Educational Environments**: Effects of Agent Properties, Picture Features, and Redundancy. Journal of Educational Psychology.

D'Mello, S. K.; Picard, R.W.; Graesser, A. C. (2007) **Towards an affect-sensitive autotutor.** IEEE Intelligent Systems, Special issue on Intelligent Educational Systems, 22(4).

Graesser, A. C.; Person, N.; Harter, D. (2001) **Teaching tactics and dialog in autotutor.** International Journal of Artificial Intelligence in Education.

Gulz, A. **Benefits of virtual characters in computer-based learning environments**: Claims and evidence. In: International Journal of Artificial Intelligence in Education, 14(3), 2004.

Jonassen, D. H.; Lee, C. B.; Yang, C.; Laffey, J. (2005) **The Collaboration Principle in Multimedia Learning.** The Cambridge Handbook of Multimedia Learning. Cambridge: Cambridge University Press.

Kim, Y.; Baylor, A. L. (2006) **Pedagogical Agents as Learning Companions**: The Role of Agent Competency and Type of Interaction. ETR&D, Vol. 54, No. 3, p, 223-243.

Longui, M. T. and Nedel, L. P. and Viccari, R. M. and Axt, M. (2004) "**Especificação e Interpretação de Gestos Faciais em um Agente Inteligente e Comunicativo"**. In: SBC Symposium on virtual reality. São Paulo, Brasil.

Longui, M. T. and Behar, P. A. and Bercht, M. (2008) "**O Desafio de Reconhecer a Dimensão Afetiva em Ambientes Virtuais de Aprendizagem"**. In: SBIE 2008 - XIX Simpósio Brasileiro de Informática na Educação. Fortaleza, Brasil.

Müller, D. N. ; Oliveira, O. L. B. de ; Remião, J. A. A. ; Silveira, P. D. ; Martins, M. A. R. ; Axt, M. (2009) **Virtual Cities as a Collaborative Educational Environment.** In: Arthur Tatnall; Anthony Jones. (Org.). Education and Technology for a Better World, pp. 112-120, Springer, Berlin, v. 302.

Ortony, A. and Gerald, L. C. and Allan C. (1988) **The Cognitive Structure of Emotions.** Cambridge, USA.

Picard, R. (1997) **Affective Computing.** Cambridge, USA. Publishing Press.

Pinho, I.C.; Epstein, D.; Reategui, E. B.; Polonia, E.; Correa, Y. **The Use of Text Mining to Build a Pedagogical Agent Capable of Mediating Synchronous Online Discussions in the Context of Foreign Language Learning.** In: Annual Frontiers in Education Conference (FIE), 2013, Oklahoma City, Oklahoma - USA. 2013.

Psotka, J; Massey, L. D.; Mutter, S. A. (1988) Intelligent Tutoring Systems: Lessons Learned. Lawrence Erlbaum Associates.

Roseman, I. J. and Spindel, M. S. and Jose, P. E. (1990) **Appraisals of emotion-eliciting events: Testing a theory of discrete emotions.** In Journal of Personality and Social Psychology. Vol. 59.

Wood, D.; Bruner, J. S.; Ross, G. **The Role of Tutoring in Problem Solving.** Journal of Child Psychology and Psychiatry, v. 17, n. 1, p. 89-100, 1976.

Vygotsky, L. S. **Mind in Society: the Development of Higher Psychological Processes.** Cambridge, MA.: Harvard University Press, 1978

Vygotsky, L. S. **Thought and Language.** Cambridge, MA: MIT Press, 1986.

Yannibelli, V.; Godoy, D.; Amandi, A. **A genetic algorithm approach to recognise students' learning styles.** Interactive Learning Environments, 14 (1): pp. 55-78. Taylor & Francis. 2006.

Relatórios individuais

RELATÓRIO DO PROJETO CITTÀ COSMOPOLITA

Projeto processo CNPq: *551846/2011-9*

Relatório de Atividades do Bolsista: *Maribel Susane Selli - Pedagoga - Doutoranda em Informática da Educação da UFRGS*

Bolsista processo CNPq: *350710/2011-2 - SET F*

Coordenador: *Daniel Nehme Müller*

Período do Relatório: *novembro/2012 a janeiro/2014*

Plano de Trabalho:

Concepção pedagógica, apoio no desenvolvimento dos módulos do projeto, formação das professoras do projeto e intercâmbio entre a escola e o desenvolvimento computacional.

CRONOGRAMA DE ATIVIDADES 2012

Meta: Aplicar o Città nas escolas

- Instalação do Città nos netbooks adquiridos pela SMED com recursos do plano de investimento na educação e FNDE (julho);
- Realização de oficina do Città com as professoras das escolas que receberam os netbooks a fim de prepara-las para trabalhar com o editor de cidades (agosto);
- Problematização da prática cotidiana das professoras dos quartos anos que trabalham com o simulador para descobrir as necessidades das crianças e aperfeiçoar/adequar o simulador (agosto a novembro);
- Visita às turmas da EMEF LEO JOAS para saber do andamento e envolvimento do trabalho das crianças com a construção da cidade virtual (setembro e outubro);
- Intermediação entre a professora e sua prática sempre que o trabalho não conseguia ir adiante através de sugestões/questionamentos/solidariedade/comprometimento com o projeto (maio a outubro);
- Visita a EMEF Conego Wolkmer para acompanhar o trabalho das professoras e ouvir as crianças (novembro);

Meta: Capacitar professores

- Formação dos professores em Estrela, encontros mensais com cada um dos seis grupos participantes (março a novembro);
- Encontro com gestores (fevereiro a dezembro);
- Problematização da prática cotidiana das professoras dos quartos anos que trabalham com o simulador para descobrir as necessidades das crianças e aperfeiçoar/adequar o simulador (agosto a novembro);

- Formação mensal com os grupos de quartos anos para pensarmos juntos os caminhos das cidades idealizadas/construídas pelas crianças estimulando-os a fazer durar a construção da cidade e com isso a produção textual, os registros no diário de bordo, as interações no Forchat (março a novembro);

- Oficina de photo shop, power point e movie maker com os colegas pesquisadores da UNEB no município de Estrela (novembro).

Meta: Tornar público os achados do projeto

- Apresentação de artigos científicos no XVII SIEDUCA – Seminário Internacional de Educação, na ULBRA em Cachoeira do Sul (julho);

- Participação na Feira do Livro de Estrela onde as crianças expuseram suas criações e produções (outubro);

Análise de 2012

Conceber a atuação de um/uma pedagogo/a no desenvolvimento de um software livre simulador de cidades virtuais pode inicialmente causar certo estranhamento. No entanto, ao adentrarmos na proposta do Projeto Civitas ao qual o Città está vinculado isso passa a ser mais do que uma necessidade, mas uma possibilidade de que possamos ser o mais fiel possível ao contexto da sala de aula e necessidades dos alunos ao nos propormos desenvolver um simulador de cidades como mais uma ferramenta que contribua na aprendizagem de nossos alunos.

Com o propósito contribuir tanto na busca de informações que alimentassem a imaginação e a possibilidade de tornar o quanto mais próximo dos interesses das crianças os recursos do Città trazendo uma gama de informações que contribuísse com o trabalho dos programadores e ao mesmo tempo com as expectativas dos alunos que utilizavam o simulador em suas aulas buscou-se estabelecer um elo entre a escola e o laboratório de desenvolvimento do Città. O espaço foi estreitado pela proposta que assumi, enquanto pedagoga pela minha presença/escuta nos diferentes espaços sejam eles presenciais, sejam nas interações nos ambientes virtuais do Forchat onde mantemos interações. A escuta sensível concepção de BARBIER (2004) foi importante aliada, pois nos permite apreender informações e registros importantes para a continuidade do trabalho.

> [...] A escuta sensível apoia-se na empatia. O pesquisador deve saber sentir o universo afetivo, imaginário e cognitivo do outro para 'compreender o interior' as atitudes e os comportamentos, o sistema de ideias, de valores, de símbolos e de mitos (ou a 'existencialidade interna', na minha linguagem). (SELLI, 2011, p. 28 apud BARBIER, 2004, p. 95) [grifos do autor].

A partir dessa presença/escuta, seja nos encontros de formação de professores, seja nos momentos de contato com os alunos nas visitas de acompanhamento às escolas foi possível capturar diferentes e significativas intervenções sobre as necessidades, expectativas, urgências e sugestões das crianças ao explorar o simulador e suas possibilidades.

Partindo da concepção dialógica na formação de professores, tem-se como referência metodológica dessa escuta/intervenção na sala de aula e na formação de professores a interação dialógica que para AXT (2006) se faz pelos enunciados que se produzem entre os sujeitos.

> A interação dialógica ampara-se no dialogismo bakhtiniano, em que toda relação dialogal implica necessariamente a alteridade e em que a toda expressão enunciativa (mesmo quando de efeito retardado em relação a uma expressão anterior) corresponde sempre uma atitude responsiva ativa prévia, dando conta de um contexto pragmático, interacional e enunciativo-responsivo, enquanto pressuposto de linguagem e de sujeitos de linguagem em relação. (SELLI, 2011, p. 25 apud AXT, 2006, p. 257)

Ao pensar a nessa interação como possibilidade de construção de relações que se produzam pela escuta do outro considerando a importância de sua presença-gente no grupo, com suas memórias, histórias, trajetórias, anseios, projetos, sonhos, medos e desejos concebe-se a interação dialógica como uma das formas de interação possíveis entre os sujeitos.

Para nós, enquanto pesquisadores do Città Cosmopolita pela via do Projeto Civitas, aqui se pode aproximar um pouco do que consiste essa proposta de construir cidades virtuais sejam em diferentes suportes concretos como papelão, papel, caixas, retalhos, sucatas, seja no simulador de cidades Città.

Refiro-me a isso, pois entendo desenhar-se um cenário infértil para a grandiosidade desse projeto caso pensássemos em hierarquizar presenças, posições de formação acadêmica ou qualquer outra grandeza que fôssemos nos apegar para estabelecer essas relações, por isso somos um grupo coeso que se complementa com seus saberes e funções. Estabelecemos essa mesma relação e temos o mesmo entendimento em nossas relações com professores em formação e as crianças que trabalhamos.

O princípio do Città, pelas vias do Civitas é exatamente este: construir cidades ambientalmente sustentáveis, economicamente viáveis e socialmente possíveis podendo existir respeito e reconhecimento entre seus moradores ou a quem desejar lá se estabelecer. Com isso podendo vivenciar princípios indispensáveis ao convívio humano como a solidariedade, a consciência ecológica, a valorização da vida e preservação do ambiente, a cooperação, no convívio em sociedade e nos espaços de fabulação, criação, invenção, problematização do cotidiano e aprendizagem.

Propõe-se esta experimentação num espírito de amorosidade e reconhecimento do outro como legítimo na relação consigo e com o mundo e, para, além disso, seguindo princípios que nos possibilitem administrar os conflitos e divergências que possam vir a acontecer durante o processo de planejamento, construção e habitação de nossas cidades tendo presente, principalmente o compromisso ético que assumimos na relação com o outro em nossas vidas.

Ações do Città Cosmopolita

- **Instalação do Città nos nets adquiridos pela SMED com recursos do plano de investimento na educação e FNDE;**
- **Visita às turmas da EMEF LEO JOAS para saber do andamento e envolvimento do trabalho das crianças com a construção da cidade virtual;**

- **Intermediação entre a professora e sua prática sempre que o trabalho não conseguia ir adiante através de sugestões/questionamentos/solidariedade/comprometimento com o projeto;**

- **Problematização da prática cotidiana das professoras dos quartos anos que trabalham com o simulador para descobrir as necessidades dos alunos e aperfeiçoar/adequar o simulador;**

- **Participação na Feira do Livro de Estrela onde as crianças expuseram suas criações e produções;**

- **Participação na oficina do Città com as professoras das escolas que receberam os netbooks;**

- **Formação mensal com os grupos de quartos anos para pensarmos juntos os caminhos das cidades idealizadas/construídas pelas crianças estimulando-os a fazer durar a construção da cidade e com isso a produção textual, os registros no diário de bordo, as interações no Forchat;**

- **Visita a EMEF Conego Wolkmer para acompanhar o trabalho das professoras e ouvir as crianças.**

Outras ações desenvolvidas em 2012:

- Realização de grupo de estudos semanal sobre a teoria bakhtiniana, um dos suportes teórico-metodológicos dessa pesquisa;

- Coordenação do Civitas em Estrela com a participação das colegas de pesquisa Camila Camargo Prates e Candice Kipper Klemm;

- Planejamento dos encontros mensais com o grupo da Educação Infantil (professoras e coordenadoras) com 21 participantes, o grupo dos Primeiros Anos com 11 participantes, o grupo dos quartos anos com 14 participantes e o grupo das coordenadoras com 11 participantes;

- Formação dos professores em Estrela, encontros mensais com cada um dos seis grupos participantes;

- Apresentação de artigos científicos no XVII SIEDUCA – Seminário Internacional de Educação, na ULBRA em Cachoeira do Sul;

- Oficinas com o grupo de pesquisadores do Cittá Cosmopolita e professores dos quartos anos em Estrela;

- Produção de artigo para publicação em livro a ser editado pela UNEB, parceira da CONEXUM/UFRGS sobre cultura digital;

- Criação de uma sala de discussão no Forchat[18] para interação dos alunos dos quartos. Nesse espaço trocam informações e dúvidas sobre suas cidades imaginárias, os acontecimentos que ocorrem nesses espaços e os planos que possuem para as mesmas;
- Mediação da sala CIVITAS ESTRELA 2012, problematizando, provocando, instigando o pensamento a produzir sentidos e aprendizados, na interação com os alunos e seus professores;
- Apoio pedagógico e organizacional nas formações promovidas pelo CONEXUM/UNEB/LELIC/UFGRS nos meses de julho, outubro e novembro de 2012;
- Apoio ao funcionamento das atividades do LELIC realizando atividades técnico-administrativas para andamento do projeto sempre que necessário ou solicitado;
- Acompanhamento das pesquisas das colegas moçambicanas que realizam seu trabalho com o grupo de professores do Civitas auxiliando-as nas revisões e fundamentação teórica de seus projetos para a qualificação de suas dissertações.

Trabalhos apresentados em eventos:

SELLI, M. S. ; AXT, M. Docência Dialógica na Formação de Professores. In: **SIEDUCA**, 2012, Cachoeira do Sul. O que o Ensino precisa saber sobre a Aprendizagem, 2012. Anais SIEduca 2012ISSN 2177-9058

PRATES, C. C. ; SELLI, M. S.; AXT, M. Dispositivos metodológicos como proposta para exercício de reflexão sobre a prática em formação de professores. In: **SIEDUCA**, 2012, Cachoeira do Sul. O que o Ensino precisa saber sobre a Aprendizagem, 2012. Anais SIEduca 2012ISSN 2177-9058

SELLI, M. S. ; AXT, M. . Fracasso escolar, mitos e possibilidades. In: **SIEDUCA**, 2012, Cachoeira do Sul. O que o Ensino precisa saber sobre a Aprendizagem, 2012. Anais SIEduca 2012ISSN 2177-9058

Capítulos de Livros:

SELLI, M. S.; AXT, M. **Formação de professores produzindo inclusão digital e aprendizagem numa perspectiva (est)ética.** Organizadores: Daniel Müller, Margarete Axt e Tania Hetkowski. Editora EDUNEB. Ano de Publicação: 2013.

Referências:

[18]O ForChat funde, num mesmo espaço, simultaneamente, as funções de Chat, Fórum e Mura. Sua estrutura visa impulsionar e potencializar os processos de produção de autoria, envolvendo a emergência de novos sentidos, bem como uma construção conceitual, continuamente realimentada a partir de novas contribuições, tornando visíveis novas e sutis diferenciações, novas e mais ricas composições. http://lab.lelic.ufrgs.br/forchat/

AXT, Margarete et. al. Interação Dialógica: uma proposta teórico-metodológica em ambientes virtuais de aprendizagem. **RENOTE**: revista novas tecnologias na educação, Porto Alegre, v. 4, p. 1, 2006.

AXT, Margarete; MARTINS, M.A.R. Coexistir na Diferença: de quando a formação em serviço pensa modos de habitar a sala de aula. In: TRINDADE, Iole Maria Faviero (Org.). **Múltiplas Alfabetizações e Alfabetismos**. Porto Alegre: Editora da UFRGS, 2008. P. 133-157.

BAKHTIN, Mikhail Mikhailovich. **Estética da Criação Verbal**. Tradução do russo Paulo Bezerra. 2. ed. São Paulo: Martins Fontes, 2003.

SELLI, Maribel. S. **FORMAÇÃO DOCENTE: reverberações possíveis na prática pedagógica**. Porto Alegre, 2011. 121 f. Dissertação (Mestrado em Educação) – Programa de Pós-Graduação em Educação, Faculdade de Educação, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2001.

CRONOGRAMA DE ATIVIDADES 2013

Meta: Aplicar o Città nas escolas

- Visita às turmas de quarto ano de uma Escola de Estrela pelos desenvolvedores para que pudéssemos conhecer mais de perto a forma de pensar e conviver daquelas crianças (junho e julho);
- Visitas às EMEIs do município de Estrela para acompanhamento dos trabalhos (abril);
- Visitas à turma para acompanhamento das produções e planejamento coletivo dos próximos acontecimentos (setembro a dezembro);
- Apresentação do projeto e das produções para o grupo de desenvolvedores (dezembro);
- Testes com o editor de cidades Città (dezembro).

Meta: Capacitar professores

- Reuniões mensais de formação com professores do Município de Estrela – RS nos meses de abril a julho quando se encerra o convênio firmado entre a Prefeitura daquele município e a UFRGS (abril a julho);
- Reuniões com os gestores do município (abril a julho);
- Reunião de avaliação do Projeto Civitas no município considerando o término do convênio em julho de 2013 ocasião em que o mesmo não foi renovado (julho);
- Apresentação da proposta do Projeto Civitas na nova Escola que passa a integrar a proposta (agosto);
- Reuniões de formação com encontros quinzenais (setembro a novembro);

- Acompanhamento do trabalho de uma professora que aceitou o desafio de trabalhar com o projeto (setembro a dezembro);
- Visitas à turma para acompanhamento das produções e planejamento coletivo dos próximos acontecimentos (setembro a dezembro);

Meta: Tornar público os achados do projeto

- Apresentação de artigo científico no XVIII SIEDUCA – Seminário Internacional de Educação, na ULBRA em Cachoeira do Sul (julho);
- Mostra de Software no II CBIE - Congresso Brasileiro de Informática na Educação, Campinas, 2013 (novembro);
- Apresentação de Painel *dos Grupos de Pesquisa em Informática na Educação.* LELIC – Laboratório de Estudos em Linguagem, Interação e Cognição. II CBIE - Congresso Brasileiro de Informática na Educação, Campinas, 2013 (novembro);
- Oficinas com o grupo de pesquisadores do Città Cosmopolita e da UNEB com professores do Quarto Ano em Estrela (junho).
- Seminário de avaliação e sistematização na escola onde foi desenvolvido o projeto com apresentação do que foi produzido para as demais turmas da escola (dezembro);

Análise de 2013

A atuação pedagógica no desenvolvimento do Città Cosmopolita possivelmente trouxe algumas contribuições significativas para o desenvolvimento desse software livre. Estar constantemente em contato com as professoras e crianças ouvindo o que tem acontecido e sido produzido na escola, possivelmente, tornou nos mais próximos da realidade da escola e da sala de aula. Com isso, foi possível à equipe de programadores que trabalhou no projeto colher sugestões do cotidiano tomando conhecimento das sugestões e necessidades das crianças.

A participação de um/uma pedagogo/pedagoga nesse projeto tem uma razão significativa: o fato do Città compor um projeto de formação de professores trabalhando com a imaginação, a problematização do cotidiano e a invenção de problemas.

O Projeto Civitas tem esses princípios que são também extensivos ao editor de cidades Città. A fim de buscar sua inserção no cotidiano da sala de aula propomos às professoras em formação a inserção do simulador em suas aulas. Isso se dá de uma forma muito tranquila, pois o simulador passa a ser um intercessor, um disparador de aprendizagens.

Delineando um pouco mais essa proposta poderia dizer que o simulador provoca o pensamento das crianças. Com a tarefa de criar uma cidade virtual as crianças precisam seguir alguns princípios, sejam eles: a escuta do outro, a cooperação, o respeito pelas ideias do outro e o trabalho coletivo permeados pelos processos criativos que estão sempre presentes desafiando as crianças nessa tarefa.

Nesse cenário um profissional da educação diretamente em contato com as crianças e as professoras que trabalham com o projeto possibilita a busca de registros que tornam o simulador mais significativo e atrativo para as crianças.

Nos diferentes momentos de formação as professoras traziam seus questionamentos e dificuldades do cotidiano. A fim de conseguirmos transformar questionamentos em produção tudo o que era registrado a partir das reuniões era levado para as reuniões com os programadores.

Nesse processo de afetos construímos novos caminhos e possibilidades que tornassem o Città cada vez mais próximo do dia a dia da sala de aula, da vida e dos problemas e acontecimentos da cidade dialogando com o processo de aprender. Isso possivelmente aconteceu pelos infinitos momentos de escuta que permearam os encontros com as professoras com as crianças, nas visitas a escola e sala de aula para acompanhamento dos trabalhos, momentos esses de extrema importância e riqueza para essa construção. Entende-se essa postura como de grande significado para um pesquisador.

> [...]O pesquisador deve saber sentir o universo afetivo, imaginário e cognitivo do outro para 'compreender o interior' as atitudes e os comportamentos, o sistema de ideias, de valores, de símbolos e de mitos [...]. (BARBIER, 2004, p. 95) [grifos do autor]

Ao considerarmos a importância dessa presença que, para além de estar junto, acolhe o que é enunciado no grupo, numa escuta ética e responsiva. A ética e a responsividade provém do compromisso e cuidado que temos com o outro, com o que ele diz e com o que isso pode significar.

> [...] A arquitetônica da responsividade está profundamente implicada, tanto numa estética da criação do sentido na linguagem, quanto ainda numa ética do sentido para o outro, eu diria uma (est) ética dialógica e amorosa, fundada nos fluxos afectivos das relações interacionais. Considero que Bakhtin institui, assim, brilhantemente, a um só e mesmo tempo: uma estética da expressividade, cujo suposto será o de investir um sentido "respond/sável" que implique uma "resposta", instaurando então uma relação de "responsabilidade" entre o si e o outro; e uma ética da escuta, cujo suposto será o de investir em um "cuidado" com o sentido, garantindo-lhe "respons/dibilidade" na ordem das intensidades, fazendo sentido para o corpo, para a vida, para o outro. (AXT, 2006, p. 261) [grifos da autora]

Assim se produzem os enunciados nos diferentes momentos de formação onde ideias são trocadas, conhecimentos construídos, projetos idealizados. Esses enunciados também provém das práticas vivenciadas na escola pelas professoras e pelas crianças.

Ao colocar-se nesse lugar de quem escuta/acolhe/respeita as diferentes interlocuções que se produzem nesses momentos entre professores podemos efetivamente construir, no coletivo, caminhos que enriqueçam o processo de ensinar e aprender, dialogicamente.

Partindo do pressuposto dialógico tanto na proposta de formação de professores quanto nas relações que se estabelecem entre pesquisadores, professores e crianças, sem grau de importância entre eles, mas considerando-os todos parte imprescindível a toda essa construção percebemos o quanto pode ser promissora essa concepção tanto no que diz respeito a formação quanto às relações entre seus diferentes atores.

Pensando nessas relações percebemos a importância das interações dialógicas são permeadas pela escuta e respeito pelos diferentes e seus pensamentos e posturas. Na diferença e em sua infinitude de possibilidades surgem possibilidades que viram produções que produzem aprendizados.

Para Axt (2006) a interação dialógica se faz pelos enunciados que são produzidos pelos sujeitos da linguagem. Ao pensarmos a interação como possibilidade de construção e de relações nos colocamos também enquanto grupo de desenvolvedores/pesquisadores, pois todos estamos aprendendo juntos com essas diferentes situações que nos deparamos.

Assim como na relação com as escolas temos formações diferentes, mas é nisso que possivelmente reside a riqueza do nosso trabalho e equipe que busca coletivamente tornar o Città cada vez mais desafiador e atrativo para as crianças através dos princípios de construir cidades ambientalmente sustentáveis, economicamente viáveis e socialmente possíveis podendo existir respeito e reconhecimento entre seus moradores ou a quem desejar lá quiser fixar residência ou construir sua morada.

Com isso compartilhamos princípios indispensáveis ao convívio humano como a solidariedade, a consciência ecológica, a valorização da vida e preservação do ambiente, a cooperação, no convívio em sociedade e nos espaços de fabulação, criação, invenção, problematização do cotidiano e aprendizagem, numa relação de reconhecimento do outro, amorosa e ética.

Ações do Città Cosmopolita em 2013

- Visitas às EMEIs do município de Estrela para acompanhamento dos trabalhos;
- Visita às turmas de quarto ano de uma Escola de Estrela pelos desenvolvedores para que pudéssemos conhecer mais de perto a forma de pensar e conviver daquelas crianças;
- Reuniões mensais de formação com professores do Município de Estrela – RS nos meses de abril a julho quando se encerra o convênio firmado entre a Prefeitura daquele município e a UFRGS;
- Apresentação da proposta do Projeto Civitas e acompanhamento do trabalho de uma professora que aceitou o desafio de trabalhar com o projeto;
- Visitas à turma para acompanhamento das produções e planejamento coletivo dos próximos acontecimentos;
- Seminário na escola onde foi desenvolvido o projeto com apresentação do que foi produzido para as demais turmas da escola;
- Apresentação do projeto e das produções para o grupo de desenvolvedores.

Outras ações desenvolvidas em 2013:

- Realização de grupo de estudos semanal sobre a teoria bakhtiniana, um dos suportes teórico-metodológicos dessa pesquisa;
- Coordenação do Civitas em Estrela no primeiro semestre com a participação das colegas de pesquisa Andréa Aparecida Konzen da Silva, Cacilda Rafael Nhanisse e Sansão Albino Timbane;
- Planejamento dos encontros mensais com o grupo da Educação Infantil (professoras) com 21 participantes, grupo dos Quarto Ano com 12 participantes e o grupo da Educação Infantil com dez participantes;

- Formação dos professores em Estrela, encontros mensais com cada um dos dois grupos participantes;
- Apresentação de artigo científico no XVIII SIEDUCA – Seminário Internacional de Educação, na ULBRA em Cachoeira do Sul;
- Mostra de Software no **II CBIE - Congresso Brasileiro de Informática na Educação,** Campinas, 2013;
- Apresentação de Painel *dos Grupos de Pesquisa em Informática na Educação.* LELIC – Laboratório de Estudos em Linguagem, Interação e Cognição. **II CBIE - Congresso Brasileiro de Informática na Educação,** Campinas, 2013;
- Oficinas com o grupo de pesquisadores do Città Cosmopolita e da UNEB com professores do Quarto Ano em Estrela;
- Apoio pedagógico e organizacional nas formações promovidas pelo CONEXUM/UNEB/LELIC/UFGRS no mês de julho de 2013;
- Apoio ao funcionamento das atividades do LELIC realizando algumas atividades técnico-administrativas para andamento do projeto sempre que necessário ou solicitado;
- Auxílio, sempre que solicitada, nas pesquisas de colegas do LELIC para leitura e revisão de fundamentação teórica em suas dissertações e teses;
- Proposta de formação continuada numa Escola Estadual de Zona Rural em que desenvolvemos o Projeto Civitas com uma turma de 17 crianças com intensa produção;
- Testagem da versão do Città em Blender com as crianças da escola referida anteriormente.

Trabalhos apresentados em eventos:

SELLI, M. S.; MÜLLER, D. N.; AXT, M.. Coexistência de tecnologias na prática docente pelos percursos e interfaces do Projeto Civitas. In: XVIII **SIEDUCA**, 2013, Cachoeira do Sul., 2013. Anais SIEduca 2013 - ISSN 2177-9058.

Workshops:

MÜLLER, D. N.; QUEVEDO, L.L.A.;, NEPOMUCENO, A. C.; SOARES, M.L.; COSTA, J. D. C.; SELLI, M.S.; AXT, M.; *Mostra de Práticas de Informática na Educação - Trilha 1: Mostra de Software.* Editor de Cidades Civitas Città. Anais do **II CBIE - Congresso Brasileiro de Informática na Educação** - Workshops (WCBIE 2013). Campinas, 2013, p. 192 – 202.

MÜLLER, D. N.; SILVA A. K. da S., SELLI, M. S.; TIMBANE, S.; BRAITBACK, O.; EPSTEIN, D.; Joelma REMIÃO, J. A. A.; AXT, M. *Painel dos Grupos de Pesquisa em Informática na Educação.* LELIC – Laboratório de Estudos em Linguagem, Interação e Cognição. **II CBIE - Congresso Brasileiro de Informática na Educação** (CBIE 2013) Workshops (WCBIE 2013). Campinas, 2013, p. 378.

MÜLLER, D.N.; COSTA, J.D.; NEPOMUCENO, A.C.; QUEVEDO, L.A.; SELLI, M.S.; SOARES, L.M.; Città Cosmopolita. Workshop de Software Livre. **FISL 2013**, Porto Alegre, 2013.

SEMINÁRIO DE APRESENTAÇÃO DOS TRABALHOS PARA O GRUPO DE DESENVOLVEDORES PELAS CRIANÇAS

TESTE DA VERSÃO BLENDER DO CITTÀ

12 09 2013

12 09 2013

12 09 2013

Considerações:

A partir do relato apresentado pela professora, tanto no evento realizado para todos os professores e alunos do sexto ano à oitava série na escola, quanto para os desenvolvedores do Città trazem aspectos significativos e importantes de serem referidos e analisados.

Dentre as coisas ditas pela professora uma me chamou muito a atenção, o fato dela dizer que " tinha encontrado uma nova forma de fazer educação que exigia mais dela, mas apresentava resultados que não tinham valor que pudesse ser mensurado. Os alunos se envolveram muito em todas as atividades propostas, se dedicaram muito, participavam de cada atividade com uma dedicação e vontade que não eram vistos anteriormente. O projeto do Andrekit como as crianças chamavam trouxe vida para nossa sala de aula. Conseguimos, juntos, resolver alguns conflitos que insistiam em pertencer aos nossos dias e todos tiveram oportunidade de demonstrar o que sabiam fazer sentindo que eram importantes na turma e que todos temos muito valor."

O projeto entra na sala de aula em função de uma necessidade apresentada pela professora de trabalhar, a partir da pesquisa sócio antropológica o tema Alimentação e Higiene. Desejava fazê-lo de uma forma diferente, mais desafiadora para as crianças. Foi aí que chegou o amigo, que queria fazer parte da turma e, a partir da imaginação das crianças assume a forma de um boneco amarelo de pernas compridas.

O Andrekit, como foi chamado, começou a passear pelas casas das crianças com um caderno de memórias onde deveriam ser registrados os acontecimentos de sua visita à família daquela criança. A partir dos relatos a professora tinha informações importantes sobre os hábitos de alimentação e higiene de sua turma e, a partir dos registros, podia problematizar o que percebia ser necessário.

Com o tempo as crianças começaram a se questionar como era o lugar que Andrekit morava e foi aí que surgiu a cidade de Segresópolis, pois ele precisava ter onde morar. Os conteúdos previstos para o estudo daquele período vieram junto com esses temas. A partir da chegada do novo amigo construíram cardápios saudáveis, discutiram o número de calorias necessárias para uma alimentação equilibrada, calcularam frações, números decimais e escala a partir das dietas, receitas e construção da cidade.

Como em toda cidade vários problemas aconteceram, como a proporção dos carros em relação às construções. Chegou-se à conclusão, embora a escola se localize numa localidade da zona rural que o problema da mobilidade urbana é muito sério e afeta todos os lugares. Para tanto precisaram decidir o que fazer com aquele excesso de carros que não tinham lugar para circular. Foi aí que decidiram criar uma exposição, num lugar afastado da cidade, para que esses carros ficassem expostos para a visitação dos moradores.

Enumerar os temas discutidos e a forma que eles se entrelaçaram com a vida do Andrekit, a construção da cidade, produção de textos, criação do jogo de trilha, histórias em quadrinhos, livrinhos de história dentre tantas construções acontecidas nesse período, mesmo que tentasse, seria quase impossível, dada à riqueza desses incontáveis momentos.

Depois de explorarem muito todos esses estudos e produções eis que surge o momento tão esperado: a instalação do Città nos computadores do laboratório da escola para que as crianças pudesses explorá-lo e sugerir algumas inovações. Embora algumas das crianças tenham contato com essa tecnologia em suas casas poder explorar o editor foi um momento único. Por muito tempo ficaram em silêncio, imóveis, pois nem todos fizeram uso das máquinas ao mesmo tempo, somente observando como manusear cada possibilidade que existia, encantados com a ideia de poder construir, também no computador, uma cidade em que poderiam aprender ainda mais coisas.

Como o ano letivo da escola estava terminando não tivemos oportunidade de retornar para novas atividades, mas as crianças pediram que não parasse nesse ano o trabalho e assim ficou combinado com a professora que eles tiveram nesse ano que passa a coordenar as atividades na escola a partir do próximo ano e de alguns colegas professores que já tinham demonstrado interesse em se envolver no projeto.

Permeando os momentos de construção, os seminários, a experimentação no editor Città foi possível perceber o quanto a invenção, fabulação, respeito, cooperação, respeito e cuidado pelo outro, a problematização do cotidiano, a invenção e resolução de problemas e uma cadeia infinita de perguntas que os desafiava a buscar respostas foi presente durante os diferentes momentos que envolveram essa professora e suas crianças.

As crianças, embora tenham explorado com maior intensidade outras tecnologias além das digitais viveram momentos que possivelmente os acompanharão para sempre, pois o que aprenderam não desaparecerá de suas memórias. A filmadora, a máquina digital, o gravador de voz, foram presença nos diferentes dias em que as construções foram planejadas e executadas.

Quando chega o momento de trabalhar com o editor Città não havia mais problemas de cada um esperar sua vez, de prestar atenção do que o outro estava fazendo, de entender que o trabalho se faz no coletivo, pois já tinham vivenciado isso anteriormente, quando da construção de sua cidade com outras tecnologias. Ao nos reportarmos aos relatos do ano passado e até de visitas que realizamos no outro município onde trabalhamos no primeiro semestre verificamos que isso não aconteceu, pois as crianças não tinham paciência nem tempo de esperar a sua vez, seguidamente haviam conflitos entre elas e também algumas que simplesmente se negavam a continuar trabalhando porque sua ideia não tinha sido acatada pelo restante do grupo.

A utilização dos computadores, mesmo em número reduzido, pois contamos com alguns imprevistos como a falta de energia elétrica em determinado momento da experimentação de forma nenhuma afetou as crianças que, calmamente, participaram de tudo,

esperando ansiosos, mas muito atentos a sua vez de jogar e descobrir os recursos que o editor disponibilizava.

A partilha, a noção de coletivo, nesse mundo em que vivemos onde a ganância e a competição nos movem e dificilmente nos percebemos parando para escutar e acolher o outro percebemos o quanto uma proposta com esses princípios pode contribuir para a formação das crianças e, consequentemente, sua vida em comunidade e sociedade, pois o editor simula a vida acontecendo na cidade.

Ultrapassando a proposta de construção da cidade, seja no editor, seja a cidade criada pelas crianças com outras tecnologias é perceptível a possibilidade de podermos trazer tanto a vida real para ser pensada e reorganizada dentro desses contextos como o contrário, a vida imaginária passar a coexistir com a vida real na busca da construção de outras relações e formas de habitar o mundo.

Em todo o processo que atravessa sejam os momentos da produção em sala de aula com o estudo dos temas referentes ao ano escolar das crianças, sejam os momentos de contato com o editor ou a construção da cidade concretizada através da união de algumas caixas, tampas, potes, cordões, palitos, cola, têmpera, isopor, mais cola, mais têmpera e muita criatividade surgem as cidades que problematizam o dia a dia das crianças transformando práticas de transmissão de conteúdos em construção de conhecimento, embora elas coexistam, sem problema algum, sempre que a professora entender necessário.

Alguns ou em vários momentos pode ser difícil de acreditar que ainda é possível isso acontecer, mas nesse ano, mais uma vez, foi possível constatar que, quando o desejo pelo diferente nos permite ousar a alegria do processo educativo retorna aos espaços escolares. Mais uma vez pude compartilhar muitas alegrias com essa professora e suas crianças que experimentaram outras formas de se relacionar e aprender e isso ficou evidenciado em cada depoimento e demonstração desse atores.

Mais uma vez experimentamos a presença de diferentes tecnologias coexistindo na sala de aula e, para além disso, o quanto isso é possível quando nos propomos a viver outros desafios. Assim encerramos mais essa etapa do projeto Città Cosmopolita, vinculado ao Projeto Civitas com alguns indicadores de que essa construção está sendo extremamente rica e promissora para nossas crianças e suas professores. Além disso destaco o quanto nós, pesquisadores, estamos aprendendo e agregando aos nossos conhecimentos com todas essas possibilidades compartilhadas.

Referências:

AXT, Margarete et. al. Interação Dialógica: uma proposta teórico-metodológica em ambientes virtuais de aprendizagem. **RENOTE**: revista novas tecnologias na educação, Porto Alegre, v. 4, p. 1, 2006.

AXT, Margarete; MARTINS, M.A.R. Coexistir na Diferença: de quando a formação em serviço pensa modos de habitar a sala de aula. In: TRINDADE, Iole Maria Faviero (Org.). **Múltiplas Alfabetizações e Alfabetismos**. Porto Alegre: Editora da UFRGS, 2008. P. 133-157.

BAKHTIN, Mikhail Mikhailovich. **Estética da Criação Verbal**. Tradução do russo Paulo Bezerra. 2. ed. São Paulo: Martins Fontes, 2003.

SELLI, Maribel. S. **FORMAÇÃO DOCENTE: reverberações possíveis na prática pedagógica**. Porto Alegre, 2011. 121 f. Dissertação (Mestrado em Educação) – Programa de Pós-Graduação em Educação, Faculdade de Educação, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2001.

ATIVIDADES E ANÁLISE 2014

O primeiro semestre de dois mil e catorze foi marcado por significativos avanços. Desde o início do ano letivo começamos trabalhar com as professoras que demonstraram interesse no projeto Civitas.

Nossas interações eram permeadas de dúvidas de como fazer considerando sendo, no meu entendimento enquanto pesquisadora, uma necessidade urgente que muitos profissionais da educação possuem de terem uma indicação clara e objetiva de como fazer o trabalho.

Não nos deteremos aqui em discutir essa questão por não se tratar do objetivo deste relatório e sim, dos registros vivenciados pelas nossas experimentações – enquanto pedagoga que compõe a equipe de desenvolvedores do Città – realizadas numa escola pública estadual onde acontece a formação e as testagens da última versão do simulador de cidades Città.

Embora muito se tenha visto produções acadêmicas que tratam do uso de tecnologias digitais em sala de aula raramente se percebe – o que seria o propósito real dessa inserção segundo autores e, nessa mesma direção, qual seria a postura do professor diante dessa nova forma de ensinar/aprender.

Em Lévy (1999) encontramos significativas contribuições que nos auxiliam nessa reflexão. Contradizendo o que muito já se ouviu frente a chegada das tecnologias na sala de aula, o professor jamais será substituído pela máquina, mas precisa sim, repensar seu papel diante da turma de crianças e adolescentes que tem a sua frente. Diz-nos o autor,

> [...] a principal função do professor não pode mais ser uma difusão dos conhecimentos, que agora é feita de forma mais eficaz por outros meios. [...] deve deslocar-se no sentido de **incentivar a aprendizagem e o pensamento.** O professor torna-se um **animador da inteligência coletiva dos grupos** que estão ao seu encargo. Sua atividade será centrada no acompanhamento e na gestão das aprendizagens: o incitamento à troca dos saberes, a mediação relacional e simbólica, a pilotagem personalizada dos percursos de aprendizagem [...]. (LÉVY1999, p. 171) [grifo nosso]

Partirmos dessa prerrogativa nos faz pensar as múltiplas possibilidades que podem se constituir ao partirmos de uma prática pedagógica que possibilita espaços de criação e invenção na sala de aula. Por esses sobrevoos situa-se o Civitas e também, como uma das ferramentas de aprendizagem possíveis, o Città – simulador de cidades em redes.

O uso de mais esse recurso se dá a partir dos mesmos princípios do projeto ao qual está vinculado. Desafiamo-nos, ao vivenciarmos essa proposta, a construir, com nossas crianças espaços de invenção, trabalho coletivo, imaginação, construção, respeito pelo outro numa relação alteritária Bakhtin (2003), problematização do cotidiano tanto da prática de sala

de aula como dos acontecimentos da cidade que são entrelaçados pelo inusitado vão compondo os cenários que autores e atores protagonizam o ensinar e aprender.

Passaremos à descrição das atividades que aconteceram no período letivo de fevereiro a maio, data de encerramento do desenvolvimento do projeto. Cabe destacar que continuamos, enquanto equipe de desenvolvimento, em constante diálogo buscando atender as demandas que possam ainda surgir no decorrer da utilização do simulador de cidades Città.

Organizamos os acontecimentos desse período em três momentos: I se refere aos encontros dos programadores; II diz dos encontros de formação dos professores e o III os diferentes momentos de interação das crianças com o Città. Passaremos agora a referenciar cada um desses momentos:

I – a realização de encontros com o grupo do desenvolvimento acontecia semanalmente, para verificarmos os avanços, pois estávamos na reta final desse primeiro propósito – desenvolver uma versão do simulador de cidades Città na versão Blender. Nesses encontros, os programadores, o designer e a pedagoga – nesse caso – a função que assumi no grupo e a qual me refiro nesse relatório discutiam os ajustes finais necessários a partir das testagens realizadas, num primeiro momento pela própria equipe e, posteriormente, conforme será apresentado no momento III, com as crianças da escola que estamos trabalhando nesse ano;

II – os encontros de formação de professores iniciaram em março com duas professoras do segundo ciclo – a escola é estadual, de zona rural, e por isso denominada escola do campo tendo essa organização curricular. Com o passar do tempo outras passaram a integrar o grupo. Nossos encontros presenciais são semanais para analisarmos o andamento do trabalho e nossos momentos de estudo e as discussões teóricas acontecem numa ferramenta chamada Forchat – um ambiente que possibilita a interação síncrona e assíncrona - que também foi desenvolvida pelos pesquisadores do LELIC – Laboratório de Estudos em Linguagem Interação e Cognição – tendo vínculo com o Programa de Pós Graduação em Educação - PPGEDU e o Programa de Pós Graduação em Informática na Educação – PPGIE – da Universidade Federal do Rio Grande do Sul.

III – as inserções na sala de aula tanto nos momentos de escuta das crianças e professoras quanto nos momentos da realização dos testes do Città com o propósito de analisar seu funcionamento e também, com maior relevância, ouvir o que as crianças tinham a dizer/sugerir sobre esse simulador de cidades em rede.

Entendemos importante ressaltar ainda um acontecimento que ainda não havíamos presenciado nesses anos todos de Projeto Civitas. Uma das turmas, juntamente com a professora, superou as expectativas que anteriormente ultrapassavam os muros da escola envolvendo as famílias nas diferentes atividades inerentes a sua realização. Dessa vez as crianças foram mais longe. Ao discutir a vida e as formas de sustento possíveis concluíram que as crianças também possuem um grande potencial para produzir e comercializar – mesmo no entorno de suas casas o que não ofereceria risco a elas – a produção de algo que seria acolhido pelo mercado. Assim, depois de muitas discussões, ideias, análises, decidiram fazer uma fábrica de chocolates.

Aproveitam esses espaços de produção para aprender diferentes assuntos que ultrapassam o currículo, pois pensam desde a pesquisa de preço e qualidade do produto a ser utilizado como matéria prima, passando pela fabricação/condicionamento e os cuidados para tal, a comercialização e os resultados. Numa das reuniões estabeleceram coletivamente uma meta: atingir dois mil reais com as vendas, valor líquido, até o mês de novembro do corrente. Ainda não decidiram o que farão com o dinheiro, pois são muitas as possibilidades, desde uma viagem de estudos, lazer, até o que ainda não foi possível de ser imaginado.

Nessa turma, a cidade, também de forma distinta de todas as outras, mesmo que no Civitas nada se repete, começou pela fábrica de chocolates e com ela a chegada dos

moradores que vinham trabalhar na fábrica que é organizada através da criação de uma cooperativa.

Assim, continuamos colecionando encantamentos pelas vias do Civitas/Città acreditando numa proposta que oportunize as nossas crianças diferentes momentos de aprendizagem de uma maneira que os instigue e provoque o desejo pela aprendizagem.

Referências:

LÉVY, Pierre. A inteligência coletiva; para uma antropologia do ciberespaço. Lisboa: Instituto Piaget,1999.

RELATÓRIO DO PROJETO CITTÀ COSMOPOLITA

Projeto processo CNPq: *551846/2011-9*

Relatório de Atividades do Bolsista: *Mércio Luciano Teixeira Soares*

Bolsista processo CNPq: *350687/2011-0*

Coordenador: Daniel Nehme Müller

Período do Relatório: *novembro/2012 a maio/2014*

Plano de Trabalho:

Programação visual do módulo de interface e assistente inteligente.

---

1. Introdução

Neste documento estão relatadas todas as atividades realizadas por mim, de Novembro de 2011 até a atual data (Julho de 2014), para o desenvolvimento do jogo Città, parte do projeto Cosmopolita.

Inicialmente fiquei encarregado de desenvolver novos objetos e artes para o jogo, mas ao longo do projeto e com renovação da bolsa de pesquisa acabamos desenvolvendo novas ferramentas e consequentemente outra versão do Città feita em Blender.

1.1 Apresentação do Città

O Assistiva é um projeto com foco na implementação pela Conexum do assistente inteligente Maga Vitta, voltado à assistência do jogo educacional Città no Projeto Civitas, desenvolvido pelo Laboratório de Estudos em Linguagem, Interação e Cognição (LELIC), da Faculdade de Educação da UFRGS.

O jogo Città é voltado a estudantes entre 8 e 10 anos de idade para a construção de cidades virtuais. O assistente tem a finalidade de interagir com os usuários provocando desafios e trocando informações na forma de diálogo.

Além da interação por texto, também estamos desenvolvendo o diálogo por voz, que possibilitará uma nova dinâmica na comunicação do assistente com os estudantes.

1.2 Objetivos do Projeto

O objetivo principal é que o usuário aprenda através da construção e da organização de uma cidade. Para que o mesmo passe por diversas situações que proporcionem este aprendizado ele terá que completar tarefas fornecidas pelos educadores ou pelo próprio jogo. Nessas tarefas os alunos terão que satisfazer a população e ter uma sustentabilidade que respeite o meio ambiente. Além disso, objetivos em comum entre os alunos fará com que troquem experiências e frequentemente haja colaboração e cooperação entre todos, incluindo o educador.

1.3 Proposta Inovadora

Existem inúmeros programas de computador com propostas semelhantes. Porém, a grande maioria é produzida por grandes corporações de software estrangeiras, com objetivos puramente comerciais. Estes jogos frequentemente estimulam a competitividade e, às vezes, induzem o participante a exercer um capitalismo desenfreado para “superar” e “vencer” seus oponentes.

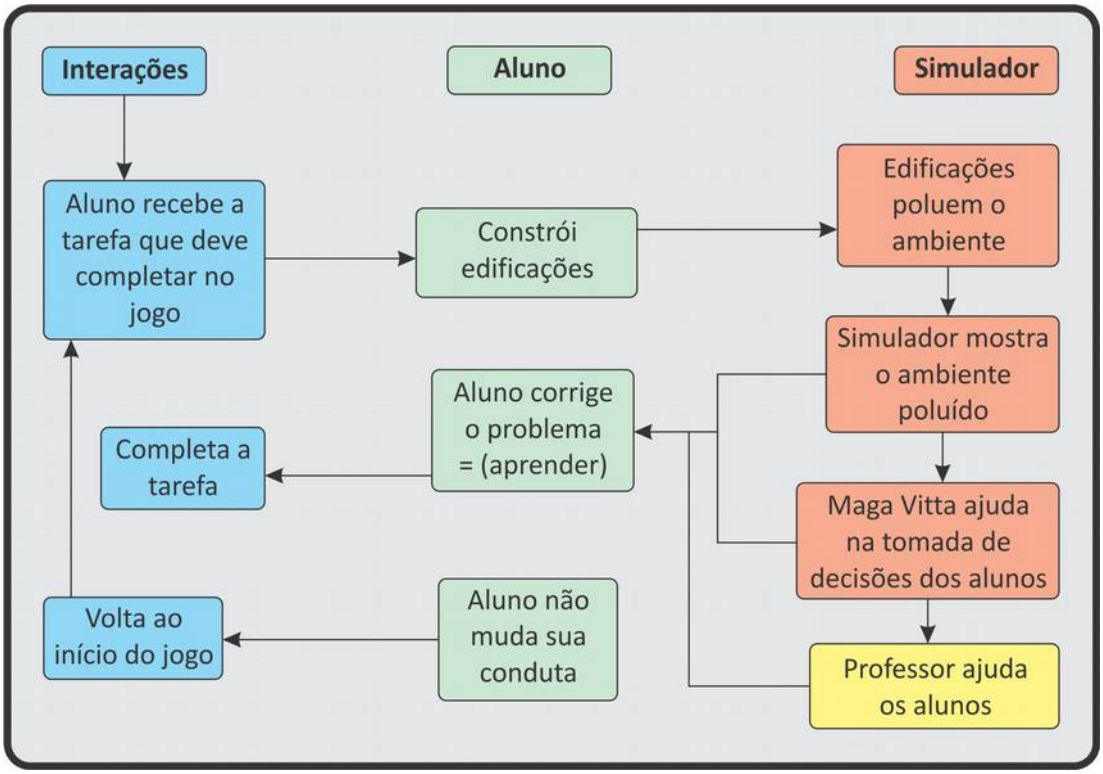
É fácil constatar que este tipo de produto pode ser indicado para divertir seus usuários por algumas horas, mas não para formar um senso de cidadania e ecologia que dure a vida inteira. Eis a principal diferença entre o Città e seus assemelhados disponíveis comercialmente: o nosso programa é capaz de entreter com uma proposta lúdica enquanto mostra para o usuário noções sobre a sociedade em que ele vive e o meio ambiente que ele, como cidadão, deve desde criança aprender a proteger.

Outro recurso inovador é a agente Maga Vitta. Que se comunicará por voz com o jogador. Através de um microfone o jogador poderá também falar com ela.

1.4 Sobre o Gameplay

O aluno receberá do instrutor ou de missões do próprio jogo, tarefas a serem completadas. Por exemplo, uma missão do jogo:

Construir 20 indústrias, pois há a necessidade crescimento da região. Logo o aluno terá que construir casas para os habitantes trabalharem nas fábricas. Habitantes precisam de alimentos, saúde, energia elétrica, transporte, educação, etc. Logo mais habitantes são necessários para preencher estes campos de trabalho, e todo este sistema polui e afeta o meio ambiente. O aluno então precisa gerenciar tais recursos para poder completar o objetivo. A figura a seguir descreve as interações entre aluno, professor e simulador.



Um pequeno esquema das interações entre aluno, Città, Maga Vitta e os professores.

1.5 Città 2.0 – Versão em Blender 3D

Tínhamos como base a primeira versão do Città desenvolvido ainda em Java para iniciar e acelerar o processo de criação da versão Blender. Mas embora eu tenha participado da versão anterior, principalmente nas atualizações de interface e objetos 3D. Não conseguimos aproveitar praticamente nada. Resolvemos começar o jogo do zero. Pegamos os pontos fortes do que já tínhamos definido anteriormente e reescrevemos tudo o que ficou pendente ou o que não agradava na versão Java. O código começou a ser escrito todo em Python, pois já vem integrado com o Blender. No início foi bastante duro, mas a cada pequeno avanço era uma vitória.

Um novo código merecia um novo visual, então colocamos em prática todos os testes que foram possíveis em relação à modelagem, animação, interface e tudo o que iria ser trabalhado. Pois a falta de testes mais específicos na versão anterior acabou nos atrapalhando a longo prazo no projeto. Os objetos e construções em 3D foram novamente retrabalhados, os menus, o mapa e novamente a Maga Vitta, que agora irá ter sua versão finalmente em 3 dimensões com algumas animações enquanto se comunica com o aluno.

1.6 Sobre meu trabalho - Apresentação

Oficialmente eu sou responsável por toda parte gráfica do jogo. Mas todas as decisões são tomadas em grupo, nas reuniões do projeto que acontecem semanalmente.

Os desafios desta versão são um pouco maior, porque temos uma ferramenta bastante poderosa, mas que tem que ser usada com muito cuidado, pois não adiantaria nada o jogo ser em 3D com gráficos de última geração e não rodar nas máquinas das escolas. Então após inúmeros testes com a versão anterior e ainda mais testes com partes fracionadas da versão em Blender, está sendo possível realizar o que eu particularmente idealizava: um jogo com uma tecnologia atual (gráficos com qualidade) que realmente funcione nos computadores básicos das escolas que o Città está inserido.

Outro fator importante para o desenvolvimento do projeto foi termos abandonado o estilo de modelagem mais voltado para realidade e começarmos a abordar o estilo *cartoon*, chamado no Blender de *toon shader*. Não esquecemos que o jogo, embora seja didático e com uma grande responsabilidade por trás no auxílio ao aprendizado das crianças, foi feito também para ser divertido. Com esse estilo de interface podemos abusar de cores e texturas, para encher os olhos e agradar por ter uma cara muita animada e amigável. Ainda temos vantagem, pois futuramente gostaríamos de implementar um editor de construções, onde a criança literalmente criaria construções a sua maneira.

Todo o jogo será ambientado em desenho mesmo. Como se o usuário estivesse desenhando a cidade e não jogando no computador. Os sons de quando se constrói algo será como de um lápis rabiscando o papel e um pequeno lápis será responsável por essa curta animação durante o jogo. A borracha será usada para quando algum objeto for destruído também tendo sua animação particular, entre outros detalhes que ainda estão sendo discutidos e planejados em cima desse universo de possibilidades que estamos trabalhando.

2. O Desenvolvimento

Assim que finalizamos a versão do Città criada em Java, pensamos em aproveitar melhor essa nova fase do projeto. Já estávamos trabalhando com o Blender (programa que tem licença de código aberto, criado pela Blender Foundation – http://www.blender.org) para modelagem dos objetos em 3D e partimos para a linguagem Python (já integrada nas versões

do Blender). Pois assim poderíamos excluir boa parte de problemas que tivemos de compatibilidade com programas e teríamos ainda uma ferramenta mais potente, estável e que poderíamos ter mais controle sobre o próprio desenvolvimento do software, sem o uso de diversos programas para adaptação. Logo após algumas reuniões e testes para decisão final de implementação do Blender como principal ferramenta de desenvolvimento, começamos a estruturação dos pontos de partida para organização do projeto.

2.1 Organização

Comecei separando tudo o que havia sido desenvolvido na versão em Java. Peguei os objetos 3D que eu mesmo havia criado na versão anterior para servirem como base de testes para a nova programação que seria desenvolvida. Assim ficaria com mais tempo para desenvolver os novos que planejávamos para o menu inicial e outras telas.

2.2 Shaders

Resolvi trabalhar com OpenGL Shading Language (GLSL), uma linguagem de *shading* de alto nível baseada na linguagem de programação C. Foi criada pela OpenGL ARB para dar aos desenvolvedores controle mais direto do *pipeline* de gráficos sem ter de usar a linguagem de *assembly* ou linguagens especificas de hardware. A GLSL é extremamente poderosa se utilizada da maneira correta, mas em contrapartida exige mais do hardware em que irá rodar.

Comecei então um longo processo de estudo sobre materiais e texturas para GLSL e também sobre ela para adequação ao projeto, pois os computadores precisariam de um mínimo de configuração e suporte a OpenGL.

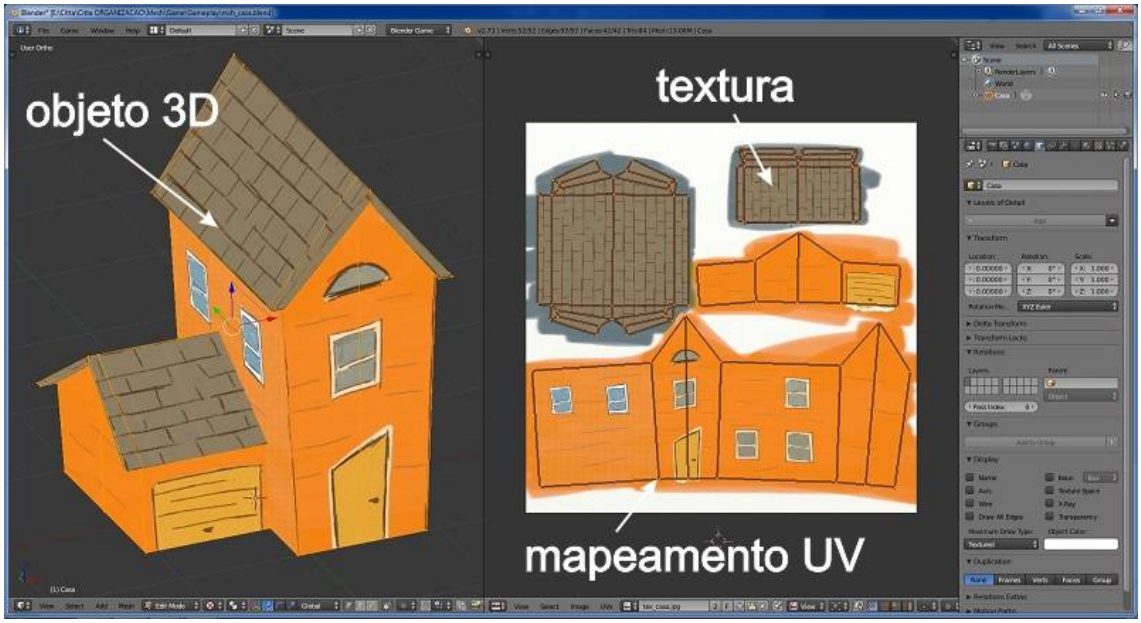
E infelizmente, foi aí em que boa parte das implementações tiveram que ser readequadas, pois os computadores em que estávamos testando o Città nas escolas não suportavam os novos modelos que eram produzidos em GLSL.

São os *shaders* que calculam e comandam todos os gráficos do jogo (cores, formas, texturas, sombras, reflexos, etc.) em tempo real, desta forma, não funcionando o sistema de GLSL tivemos que nos contentar em trabalhar com o *multitexture* que o próprio Blender oferece sem necessidade de pacotes externos.

(Sem GLSL | Com GLSL)

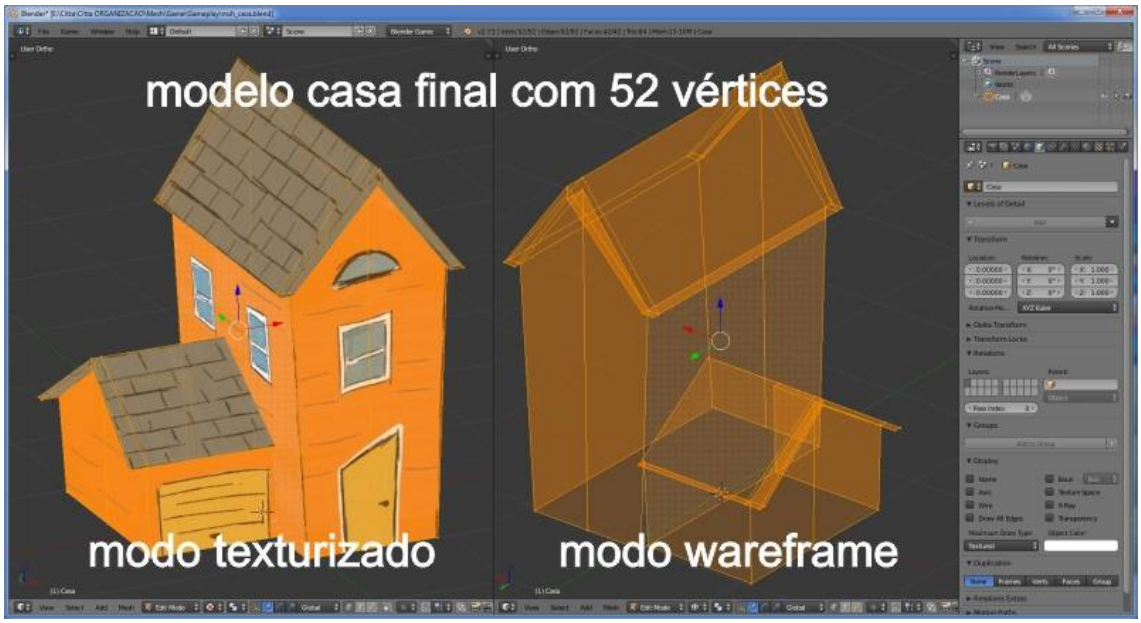
Aqui um exemplo bastante claro comparando o mesmo modelo 3D com as duas texturas

As texturas dos objetos são mapeadas em planos 2D dentro do próprio Blender, depois são exportadas como um arquivo de imagem para serem "pintadas" em um programa externo. Nesse caso eu utilizei o GIMP (*GNU Image Manipulation Program* – http://www.gimp.org – também escrito em Python e com código livre).



Objeto casa sendo modelado/texturizado no Blender. Note que a esquerda o objeto 3D e a direita a textura planificada com o mapeamento UV dos vértices sobrepostos.

Outro ponto muito importante para o bom andamento do jogo sem *legs* ou travamento é a quantidade de vértices que cada modelo/objeto teria. Logicamente, quanto menos vértices, mais leve é o modelo. Sendo assim, o maior desafio era criar modelos com poucos vértices que não perdesse sua qualidade visual.



Objeto casa finalizado, sendo exibido no modo textura (texture) e modo arame (wareframe)

2.3 Menu Inicial do Città

Normalmente a tela de apresentação do jogo tem uma imagem de fundo relacionada ao jogo, claro, e em seguida surgem os botões e outras imagens formando o menu inicial do jogo. Como era, inclusive, na versão Java do Città.

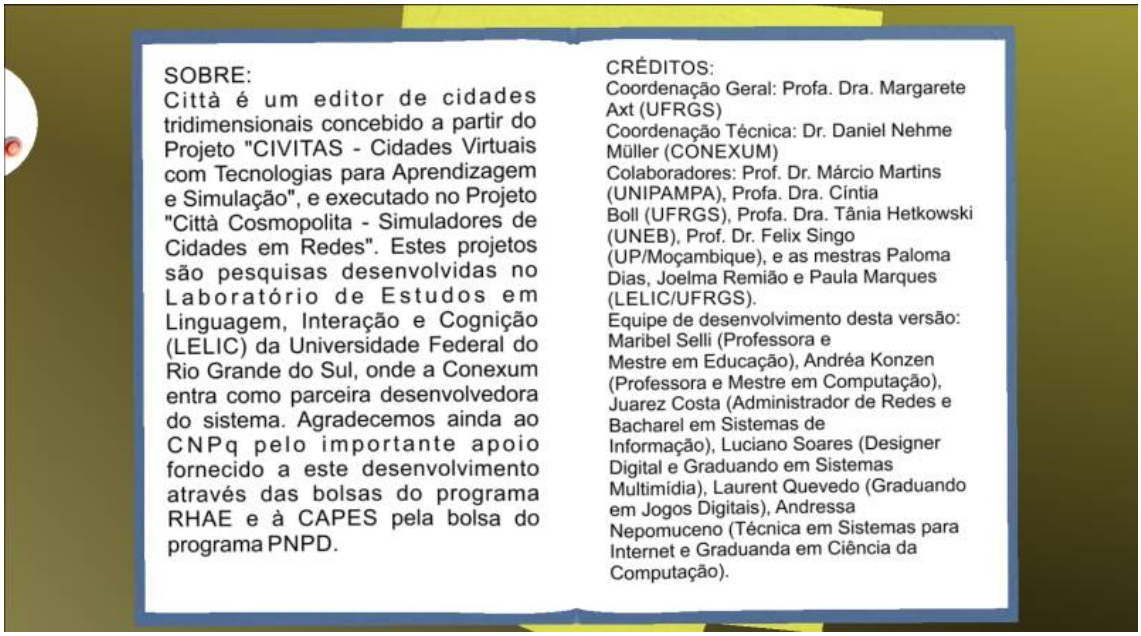
Bem, desta vez não. Achei mais interessante ter uma abordagem diferente já no inicio do jogo, pois não é um jogo qualquer. Acredito que a interação com o usuário, no caso as crianças, é talvez o mais importante fator para que se tenha um aproveitamento superior no aprendizado. E já que a proposta é criar um jogo para isso, porque não aproveitar também a experiência para aproximá-lo ao máximo da realidade dessa criança?

Baseando-me nesta ideia, bolei um menu bastante interativo onde não se tem botões ativos inicialmente e sim, objetos cotidianos para criança na faixa etária do projeto Assistiva. O menu inicial irá mostrar uma mesa com diversos objetos bastante conhecidos e comum a todos que vão para a escola: mochila, cadernos, livros, estojos, lápis, borracha etc. Ou seja, todo o material escolar básico. E o desafio começa aqui: cada objeto terá uma função se for acionado com um clique do mouse. Por exemplo:

Ao passar o mouse por cima do caderno, por exemplo, aparecerá uma pequena descrição do que ele faz. No caso, “Ir para o Città”, se for clicado o jogo será iniciado. Uma animação de câmera irá focar no caderno para a escolha dos modos de jogo e seleção de mapas, após, o jogo iniciará, dando a sensação de que o aluno está brincando de desenhar a cidade. Assim, cada objeto terá sua legenda e animação respectiva ao ser clicado.

Menu inicial do Città com todos objetos finalizados

Objeto CADERNO selecionado – Seleção de novo mapa



Objeto LIVRO selecionado – Sobre e Créditos

Objeto BOTTONS selecionado – Parceiros no desenvolvimento do Projeto

Objeto GAMEBOY selecionado – Pequeno tutorial explicando as funcionalidades do jogo

Objeto MOCHILA selecionado – Sair do jogo

Sequência de telas do GAMEBOY

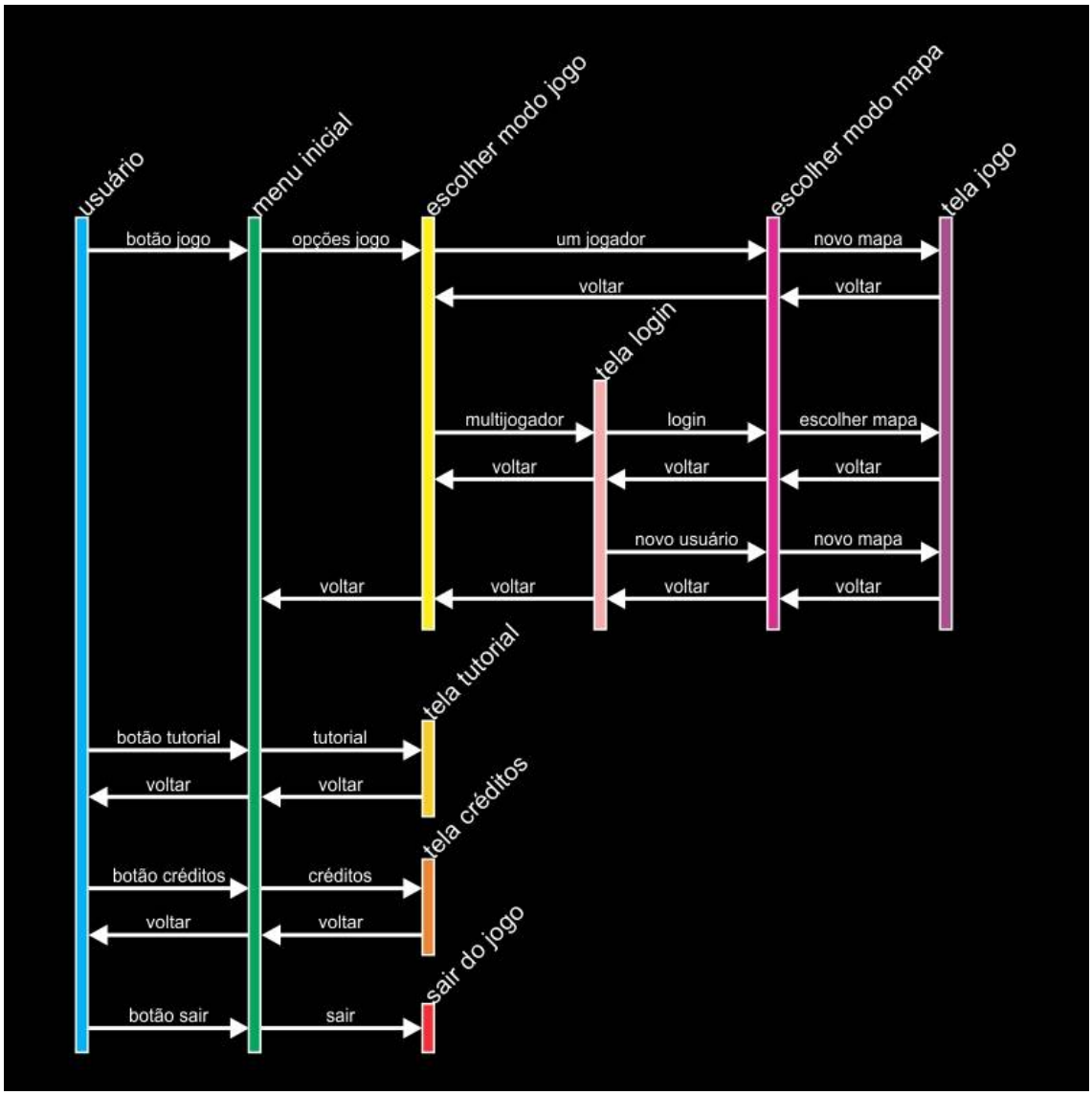
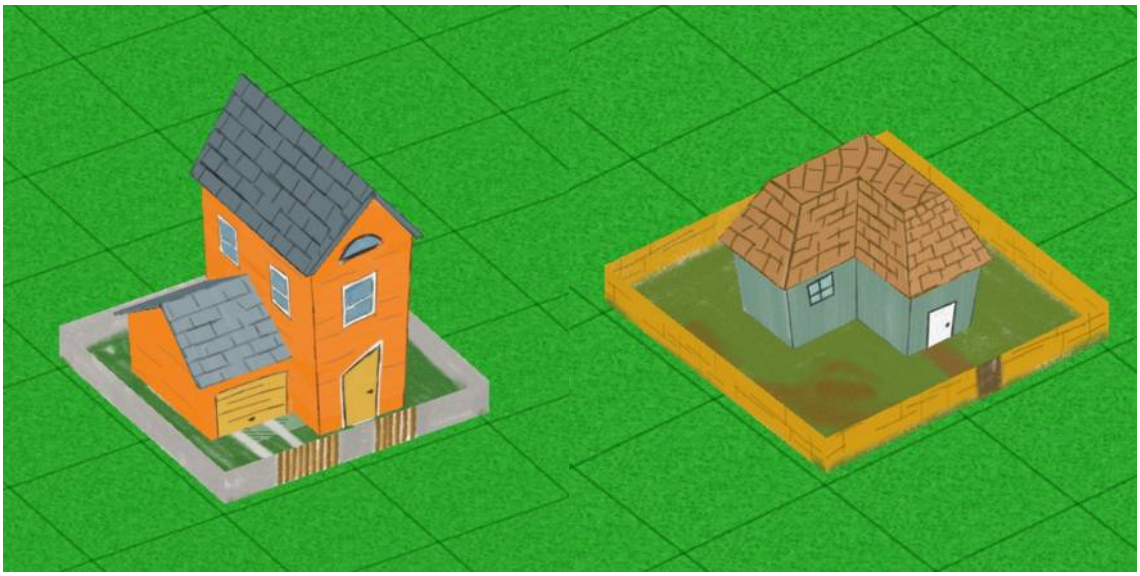


Diagrama de sequência do Menu Inicial

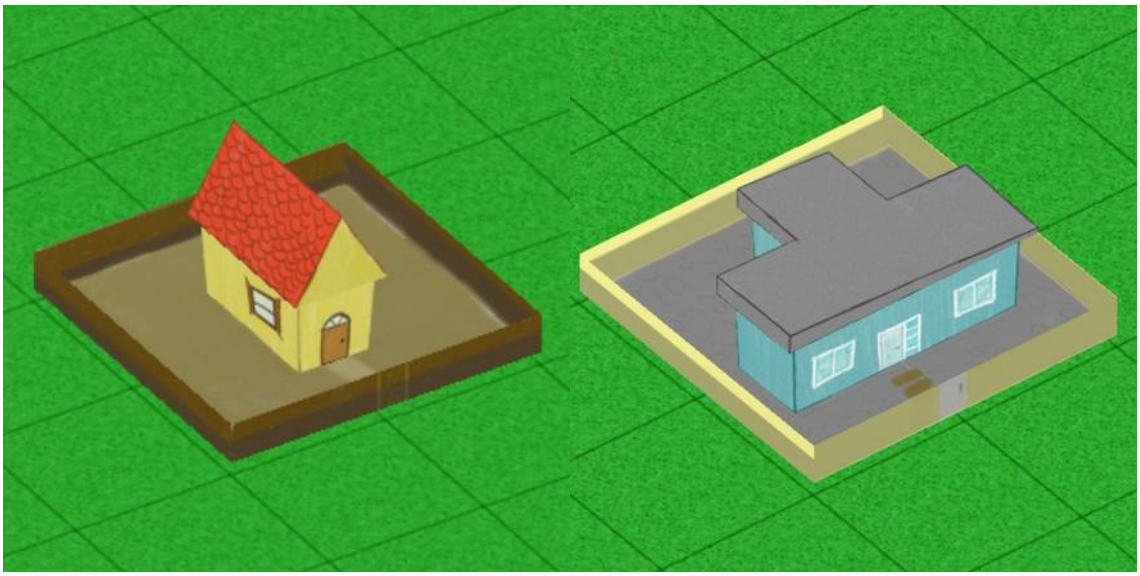
2.4 Objetos 3D

Na construção dos objetos 3D tive que tomar cuidado com as dimensões e escala dos objetos dentro do mundo 3D. Pois como parti para o estilo cartoon, por um lado não queria me prender a realidade tão precisa, por outro não podia extrapolar nas dimensões dos modelos 3D. Segui alguns pequenos padrões que achei necessários no momento em que fazia as modelagens, mas somente testando todos os objetos juntos e rodando o jogo é que tive noção exata de como a disposição era melhor. A estrada, por exemplo, de início parecia muito pequena com relação aos demais modelos, ficando a impressão que as construções eram imensas.

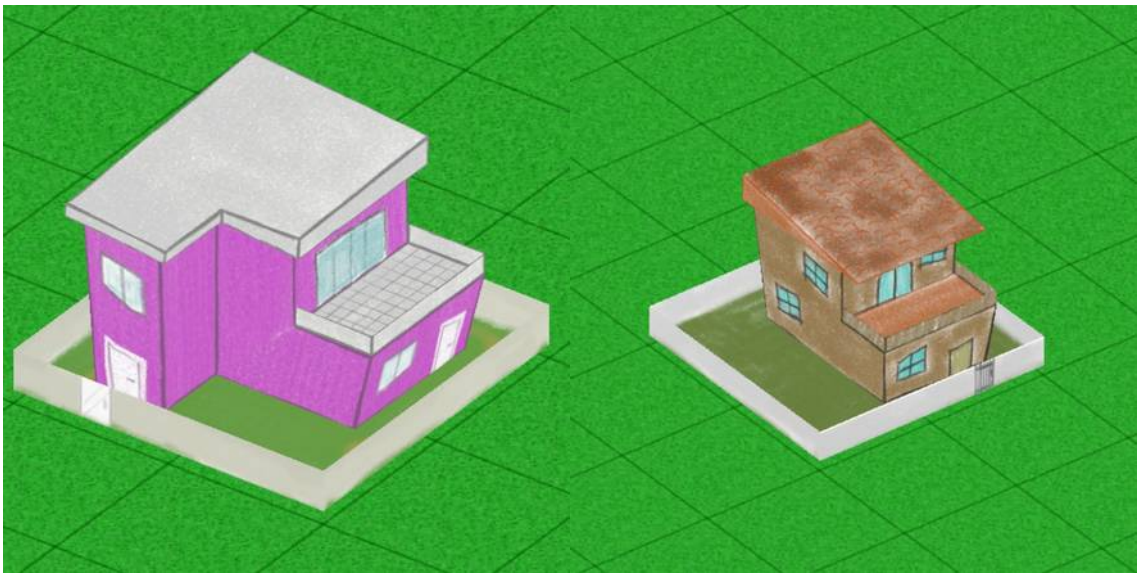
O terreno também recebeu atenção, primeiramente era para ser “liso”, depois com os testes de jogo, tanto do ponto de vista prático quanto estético escolhi uma textura de grama com as marcações dos *tiles*, facilitando a visualização e localização no espaço disposto para as construções. Também ajudou bastante para o comparativo de tamanho dos objetos e ainda a relação de espaço entre eles.



Objetos 3D – Casa #1 e Casa #2

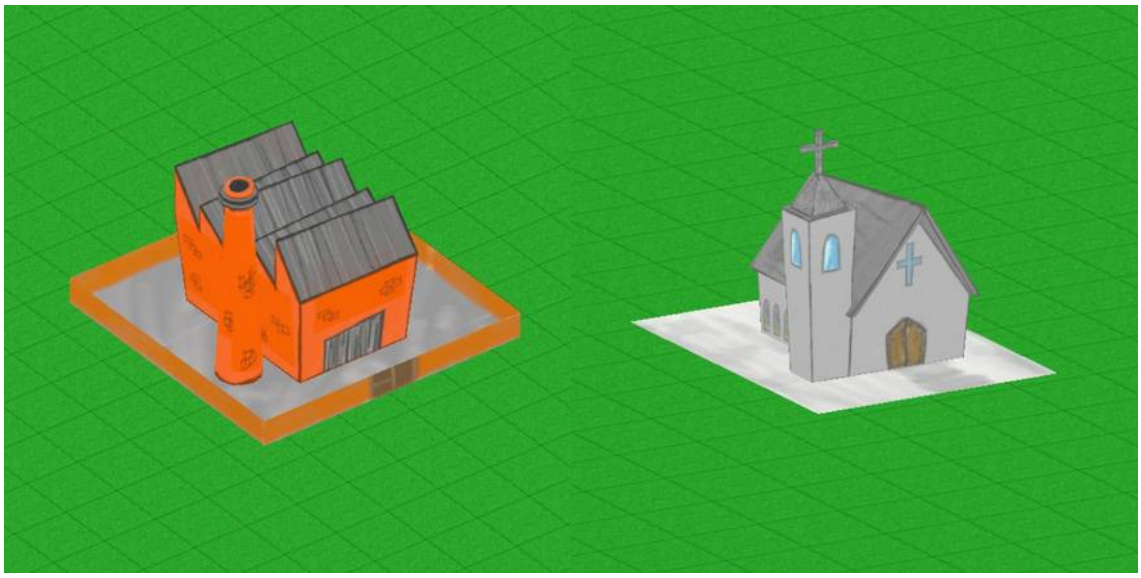


Objetos 3D – Casa #3 e Casa #4

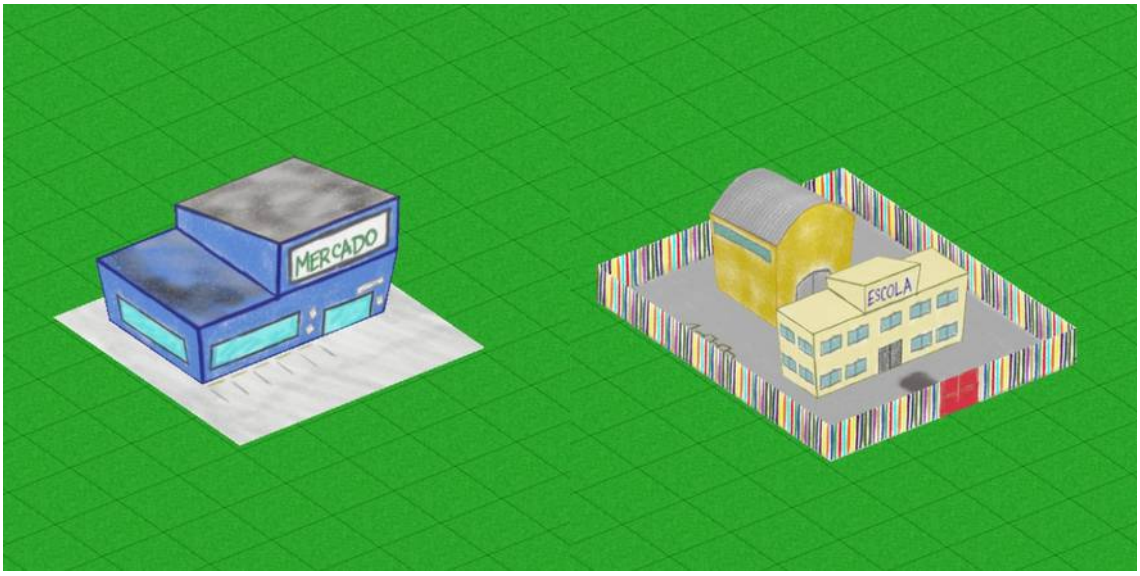


Objetos 3D – Casa #5 e Casa #6

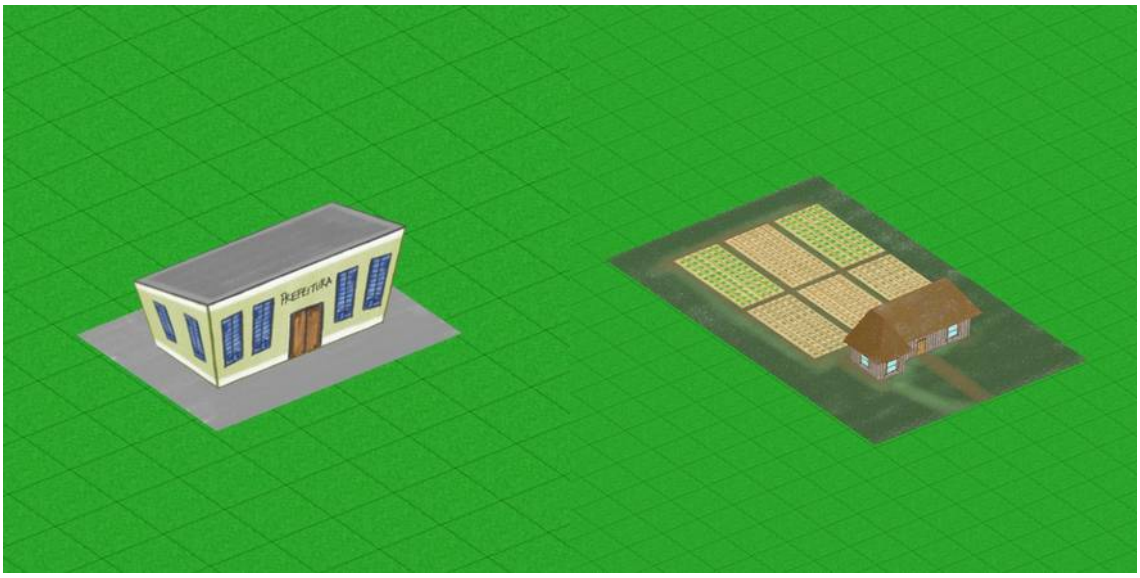
Objetos 3D – Usina Eólica e Hospital



Objetos 3D – Fábrica e Igreja



Objetos 3D – Mercado e Escola

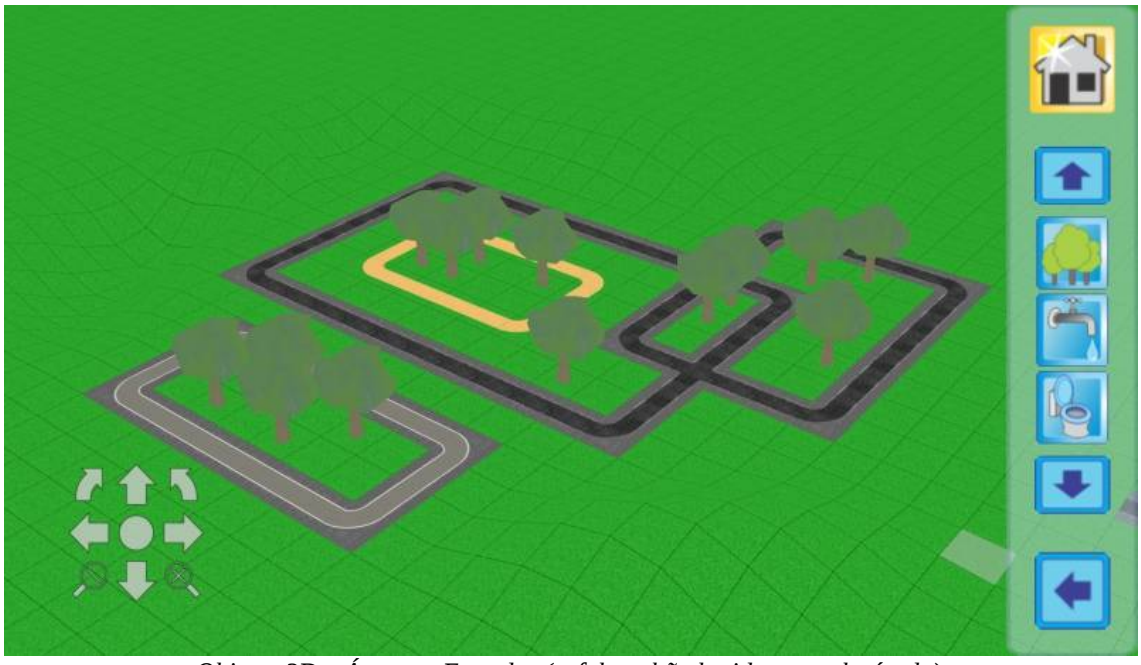


Objetos 3D – Prefeitura e Fazenda

Objetos 3D – Estação de Tratamento de Esgoto e Estação de Tratamento de Água

Objetos 3D – Prédio #1 e Prédio #2

Objetos 3D – Prédio #3 e Prédio #4



Objetos 3D – Árvore e Estradas (asfalto, chão batido e paralepípedo)

2.5 Menu do Jogo (HUD)

O menu com as ferramentas que utilizamos dentro do jogo, chamado menu HUD, já havia sido bastante alterado na sua ultima versão desde o inicio do projeto. Tentando parecer mais dinâmico e intuitivo, a ideia foi mostrar para o usuário apenas o que é necessário. Alguns estudos que fizemos, por exemplo: grande parte das crianças não entendia o que o objeto que tinha como símbolo uma escavadeira significava e quando falávamos que era para apagar os objetos (ou seja, era uma escavadeira para demolição de terrenos e edificações) elas nos diziam “mas então tem que ser uma borracha para apagar as coisas”. Esse é um dos exemplos que corrigimos. Todos os botões são mostrados em tons de cinza e ao passar o mouse sobre cada botão ele fica colorido, quando é selecionado o fundo fica dourado enquanto o objeto estiver sendo utilizado. Assim fica bastante rápido de visualizar e identificar qual botão/objeto está sendo ativado. Existe uma barra de rolagem para cada submenu que é exibido, podendo ter controle sobre os botões sem poluir a tela de jogo como acontecia com as versões anteriores.

Os botões também foram desenhados simbolizando seus respectivos modelos, mas foram desenhados em 2D para deixar menos repetitivo e também explorar outro conceito de arte voltada ao estilo *cartoon*.

Menu de Criação – onde ficam os objetos do jogo que poderão ser construídos

Menu de Terreno – onde pode-se alterar propriedades do terreno como elevações, declíneos e inserções de água, areia, grama, etc.

Menu de Estradas – onde são criadas ruas (asfalto, chão batido e paralepípedo)

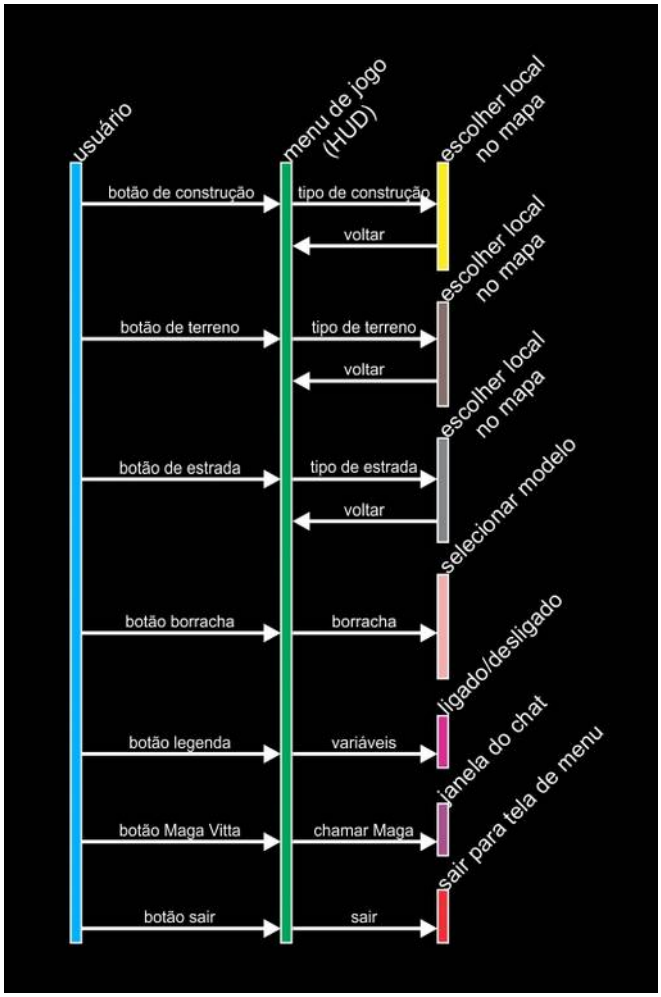


Diagrama de sequência do Menu de Jogo (HUD)

3. Maga Vitta

A Maga Vitta ganhou uma atenção especial, pois precisávamos de uma figura amigável, que fosse interessante para crianças (por isso o estilo de desenho *mangá*), mas que não fosse infantil demais, porque ela é o assistente que deve auxiliar as crianças. Então foi a primeira a passar pelo processo de mudança ganhando um aspecto jovem e moderno com diferentes cores de cabelo de acordo com a reação das respostas dos alunos e/ou como está o andamento do jogo (níveis de poluição e ordem ou desordem da cidade).

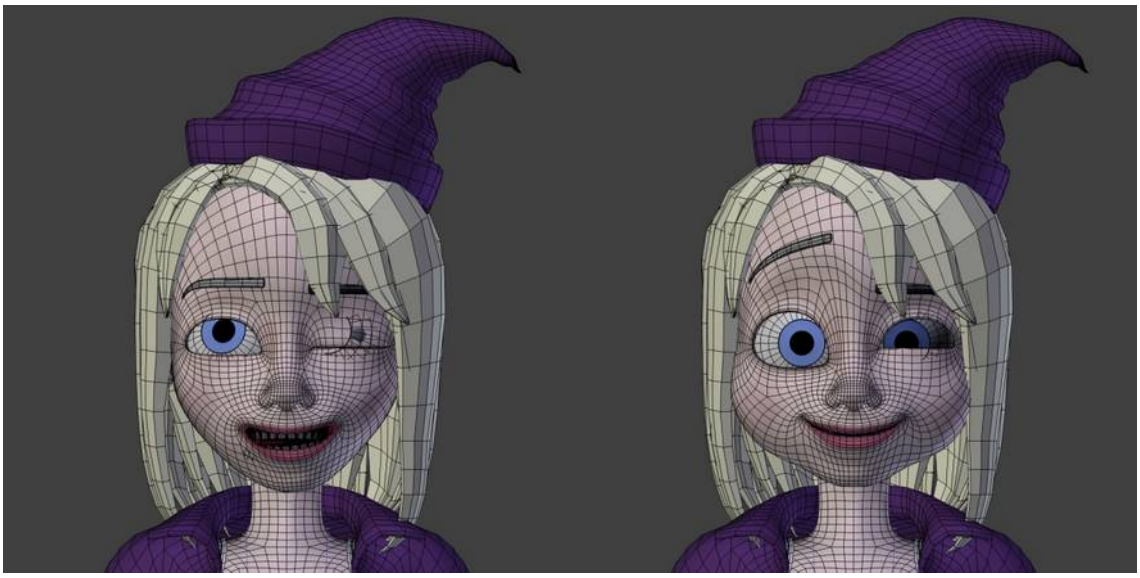
Arte Conceitual da Maga Vitta – suas versões de humor (da mais calma para mais nervosa)

Atualmente a Maga foi modelada em 3D também no Blender 3D. A ideia é fazer o assistente ter pequenas reações e movimentos para dar mais vida ao personagem. Outro item bastante importante é que a Maga aparecerá quando for de extrema necessidade com uma janela de aviso alertando o aluno do que está acontecendo de errado na cidade. Mas também poderá ser ativada sempre que o aluno sentir que precisa de algum tipo de ajuda.

Outra grande importância aqui é ter o modelo da Maga com qualidade e ao mesmo tempo com a quantidade menor possível de polígonos. Como temos máquinas muito limitadas para rodar o jogo, as imagens que aparecerão são pré-renderizadas, isto é, são imagens que simularão algum movimento e expressões da personagem.

No caso das imagens abaixo modelei em *high poly* (quantidade grande de polígonos) para depois de pronto adaptar ao que precisamos que é *low poly*. Como não foi possível trabalhar com materiais em GLSL, assim fica possível deixá-la com maior qualidade gráfica, visto que são imagens geradas fora do jogo, assim excluindo a necessidade de preocupação com processamento de polígonos.

A importância das expressões principalmente faciais da Maga me necessitou boas horas de estudo sobre o assunto, além dos estudos sobre técnicas de modelagem orgânica específica para personagens e animação para tal fim. Abaixo alguns testes e a versão final da Maga Vitta.



Maga Vitta – testes de expressões com o modelo final em 3D

Maga Vitta – versão final em 3D

4. Pesquisa de Campo

Fomos a uma Escola Estadual de Zona Rural para testar o Città com alunos do 5° ano do ensino fundamental e avaliar o entendimento de alguns itens bastante importantes para a continuidade pro projeto:

- do menu inicial;
- do menu HUD;
- dos objetos e seus elementos;
- compreensão geral de como o jogo funciona;

A compreensão geral do jogo teve um resultado bastante positivo, primeiro pela novidade e curiosidade por parte dos alunos. Mas também pela rápida assimilação dos elementos do Città. O menu inicial foi compreendido, e agradou muito pelo impacto visual logo de início e pelas animações e transições de tela. O menu HUD (menu de jogo) foi bastante utilizado e teve seu papel cumprido, os alunos compreenderam seus elementos e botões e como utilizar da maneira correta. Isso aconteceu muito rápido e nos incentivou a continuarmos por esse caminho no que diz respeito ao menu de jogo. Os objetos 3D chamaram muita atenção por suas cores e formas desenhadas, ficando bem evidente o impacto visual e aceitação comparando a versão anterior com essa.

Em geral, o jogo foi muito bem aceito, arrancando muitos sorrisos a cada objeto criado no mapa do Città, e também muitos detalhes pertinentes que nos permitiram melhorar o que ainda não estava bem definido ou que precisava ser realmente alterado.

5. Curso 3DS Max

Os bolsistas do projeto receberam um mini-curso de 3DS Max em Novembro de 2012 ministrado pelo designer e professor do IFBA Edson Machado, promovido pela parceria com o grupo Geotec e a UNEB.

Realizamos modelagens simples com durante 3 dias e aprendemos algumas técnicas de mapeamento, aplicação de texturas e render.

As aulas ocorreram dentro do campus da UFRGS na sala do Laboratório de Estudos em Linguagem, Interação e Cognição (LELIC).

6. Apresentação do Città no FISL14

Participamos da 14ª edição do Fórum Internacional do Software, apresentando o Projeto Cosmopolita em seu todo. Desde o conceito da pesquisa até a aplicação nas escolas. E aproveitamos os próprios modelos e objetos do Città para montar uma apresentação dinâmica, passeando pela cidade ao longo da palestra. Os *slides* foram montados em *outdoors* dentro da cidade montada exclusivamente para apresentação.

Fiquei encarregado de produzir os modelos, a montagem da cidade e as animações da apresentação. Também refinei as imagens que me foram fornecidas para texturizar os *outdoors* com o material que foi apresentado.

Na minha parte do discurso, basicamente falei sobre as características visuais do jogo e suas funcionalidades. E das escolhas e questionamentos que fizemos para chegar ao design geral do Città.

7. Primeira Viagem à Salvador

Em julho de 2013 juntamente com os bolsistas Laurent e Vitor, viajamos para Salvador interagir com o Grupo de Geotecnologias, Educação e Contemporaneidade (GEOTEC). Lá nós apresentamos o Città e fomos apresentados ao Kimera (cosmopolita sendo desenvolvido por eles). Assim, trocamos experiencias e informações além de ministrarmos um pequeno curso de um dia, sobre modelagem para jogos no Blender e realidade aumentada.

De minha parte, fiquei encarregado da parte de modelagem de objetos 3D e animação.

8. Artigo Interface para Jogos Educacionais

Em julho de 2013, escrevi, juntamente com Laurent Quevedo e Daniel Müller, o artigo ***Interface para Jogos Educacionais: o impacto da usabilidade no simulador de cidades Città***, que seria disponibilizado no livro (nome) pela UNEB.

O artigo visa a descrição do processo evolutivo da interface do Città, e a compara com suas versões anteriores, se baseando em estudos feitos na área de IHC (Interface Homem-Computador).

A minha parte do trabalho era descrever todo o desenvolvimento do processo visual do jogo. Abordando desde os conceitos para a concepção do trabalho até os detalhes do processo de criação, chegando ao ponto do projeto em que nos encontrávamos na época.

9. Segunda Viagem à Salvador

Em janeiro de 2014, junto com os bolsistas Laurent e Andressa, fomos à Salvador novamente para dar um novo curso de desenvolvimento de jogos me Blender, pois da outra vez tivemos muito pouco tempo útil para isso. Tivemos quatro dias de curso durante manhã e tarde e finalizamos com o fechamento de um jogo de naves, como todos os elementos base de um jogo: modelos 3D, cenários, barras de vida, menus, etc.

Poster de divulgação do curso

10. Referências

Manual Oficial do Blender

http://wiki.blender.org/index.php/Doc:2.6/Manual/

Tutoriais de sites dedicados ao Blender

http://www.blenderguru.com/tutorial-category/modeling-tutorials/

http://cgcookie.com/blender/category/tutorials/modeling/

Cursos Online de Blender que comprei para os estudos

http://ead.allanbrito.com/cursos/blender/

Brito, Allan (2011 – Editora Novatec 1º Ed.). Blender 3D – Jogos e Animações Interativas

http://www.novateceditora.com.br/livros/blender_3d/

RELATÓRIO DO PROJETO CITTÀ COSMOPOLITA

Projeto processo CNPq: *551846/2011-9*

Relatório de Atividades do Bolsista: *Laurent Le Avner Quevedo*

Bolsista processo CNPq: *350725/2012-8*

Coordenador: Daniel Nehme Müller

Período do Relatório: *novembro/2012 a maio/2014*

Plano de Trabalho:

Remodelagem do editor Città na plataforma Blender, bem como suas conexões com bancos de dados, sincronização de usuários em rede, participação de reuniões, visitas de campo nas escolas, formatação de documentação técnica e elaboração de relatórios.

---

1. Introdução

Neste documento estão relatadas todas as atividades realizadas por mim, de novembro de 2012 até maio de 2014, para o desenvolvimento do jogo Città, parte do projeto Cosmopolita.

Fiquei encarregado de desenvolver o programa em Blender, recriando tudo aquilo que havia sido feito em sua última versão em Java, e melhorar algumas dessas funções que este deveria realizar, pois a ferramenta utilizada anteriormente dificultava a implementação de determinadas funcionalidades.

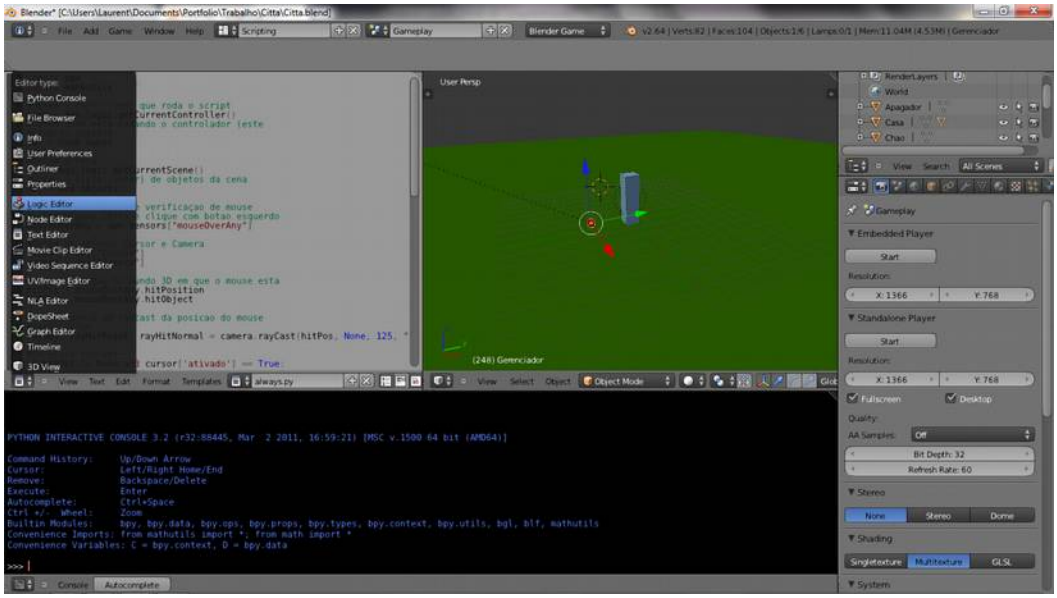
1.1. Visão do Projeto

Desenvolvimento de um software gráfico interativo (jogo) para apoiar pedagogicamente na construção de conhecimento das crianças de 4º. e 5º. ano do ensino fundamental, no que se refere ao desenvolvimento de capacidades que lhes permitam pensar-construir-organizar uma cidade virtual a partir da realidade infantil e seus conhecimentos incorporados através dos conteúdos oferecidos no aprendizado dos sistemas que regulam uma cidade e suas relações.

2. Desenvolvimento

2.1. Primeiros Passos

Durante minha primeira semana no projeto, procurei me familiarizar com o Blender, para então estudar a Blender Game Engine (BGE), o motor utilizado para a criação do jogo.



A interface do programa funciona por meio de janelas customizáveis, de forma que o usuário pode definir padrões de UI e alternar entre eles.

Como o próprio Blender foi desenvolvido usando Python, os scripts também devem ser criados nessa mesma linguagem de programação.

2.1.1. Importando o primeiro objeto 3D externo

Tendo noção de como utilizar as funções básicas do programa, tive como primeira tarefa importar um modelo 3D (uma casa), em formato Wavefront (.obj), para a cena de jogo. Para isso, apenas usei o comando “Import Wavefront” presente na aba File. Algumas propriedades do material do objeto, porém, tiveram de ser ajustadas para sua visualização em tempo de execução: aumento do valor de Specular Hardness, para tornar suas cores mais nítidas, e remoção da transparência Z do material, pois fazia com que algumas faces ficassem sobrepostas em determinados ângulos de visão.

2.1.2. Estudando os Blocos de Lógica

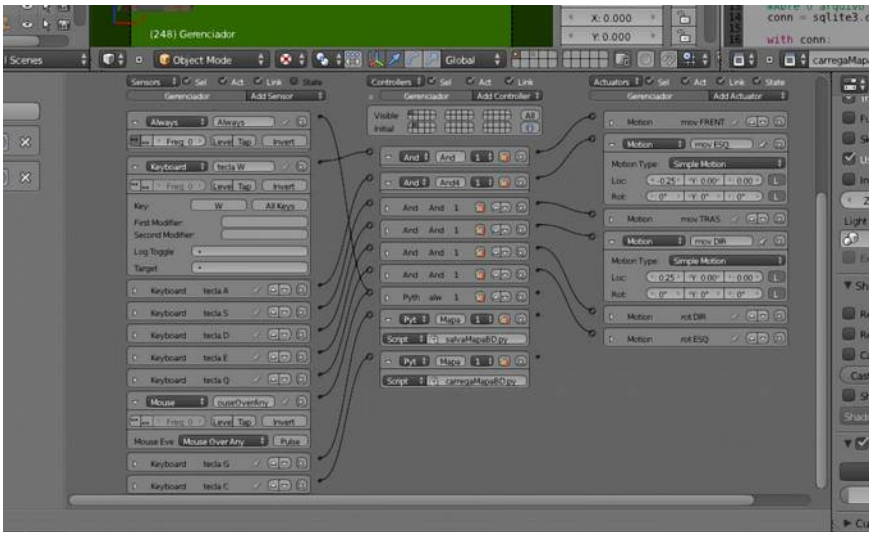
Feito isso, estudei sobre os Blocos de Lógica (Logic Bricks) da BGE, para então entender como funcionava a execução de scripts em Python no programa.

Para estes, temos Sensores (Sensors), Controladores (Controllers) e Atuadores (Actuators). É possível fazer ligações entre Sensores e Controladores e Controladores e Atuadores.

Sensores correspondem a um comando de entrada do usuário ou algum evento que desencadeará uma ação. Podem enviar pulsos positivos ou negativos para o controlador em que estiverem conectados.

Controladores funcionam como portas lógicas, analisando os pulsos recebidos por sensores e então mandando uma resposta para os atuadores em que estiver ligado caso sua condição seja verdadeira.

Atuadores realizam de fato uma ação se receberem um pulso de um controlador.

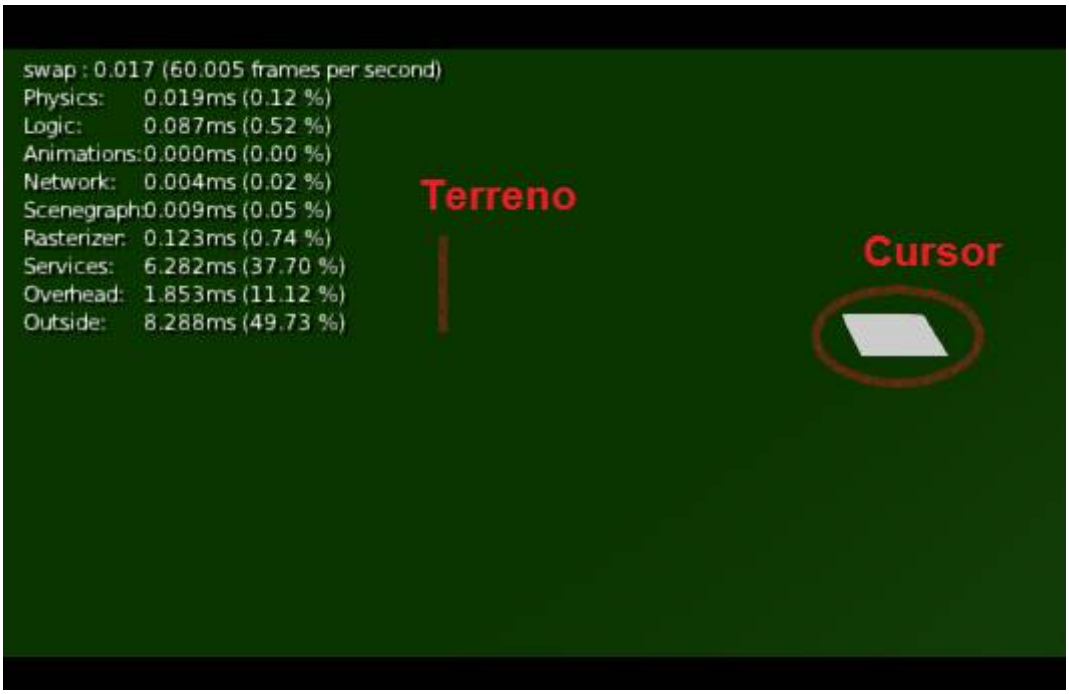


2.1.3. Criando o primeiro algoritmo de construção de objetos

Os scripts em Python são executados a partir de um controlador, conforme recebe o pulso de um sensor ligado a ele.

Sabendo como funcionam os logic bricks e como executar scripts a partir deles, trabalhei na criação da casa em tempo real por comandos do usuário.

Algumas funcionalidades do projeto, como uma cena básica com câmera e chão, e um cursor projetado no terreno a partir da posição do mouse no mundo 3D, haviam sido implementadas pelo bolsista anterior. Meu trabalho foi apenas de entender o que ele tinha feito, até mesmo para poder melhorar essas funções.



Primeiramente, foi criado um sensor num objeto chamado Gerenciador para a tecla "C", que foi conectado a um controlador que executava o script:

```
1.  import bge

2.

3.  cont = bge.logic.getCurrentController()

4.  own = cont.owner

5.  sce = bge.logic.getCurrentScene()
```

A primeira linha importa o módulo "bge", responsável pelos métodos específicos da engine.

A variável "cont" referencia o controlador que estiver rodando o script, e com isso pode-se obter o objeto que possui esse controlador ("own = cont.owner").

"sce" referencia a cena em que o objeto rodando esse script pertence, e "teclaC" recebe o sensor de nome "tecla C" deste objeto.

A linha 7 checa se houve pulso positivo partindo do sensor "teclaC", o que, nesse caso, corresponde ao pressionar da tecla C do teclado. Se for verdade, "sce.addObject" deve adicionar o objeto de nome "Casa", na posição do cursor, e obter a referência desse objeto. A casa também deve estar em uma camada não visível da cena para se utilizar essa última função.

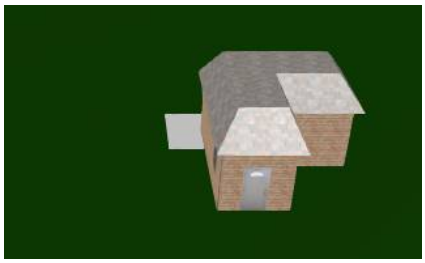
Então o programa, de fato, criava um objeto Casa na posição do cursor, porém, com uma rotação de 90º em torno de seu eixo X. Depois de alguns testes com esse método, percebi que ele não somente passa a posição do objeto indicado como referência para o objeto a ser criado, mas também sua rotação e escala. Sendo assim, somente acrescentei as seguintes linhas:

```
1.  objCasa.applyRotation([1.570796326794897,0.0,0.0], True)

2.  objCasa.scaling = [0.5,0.5,0.5]
```

O método “applyRotation()” rotaciona o objeto, em radianos, nos eixos XYZ. No caso da linha 1, apenas o eixo X é rotacionado, pois os valores para Y e Z estão em 0.0.

Scaling define a escala do objeto, também nos eixos X, Y e Z respectivamente.

Com isso, a casa era criada no lugar do cursor que percorre o chão, mas este ainda não se movia em quadrados (tiles) como deveria ser.

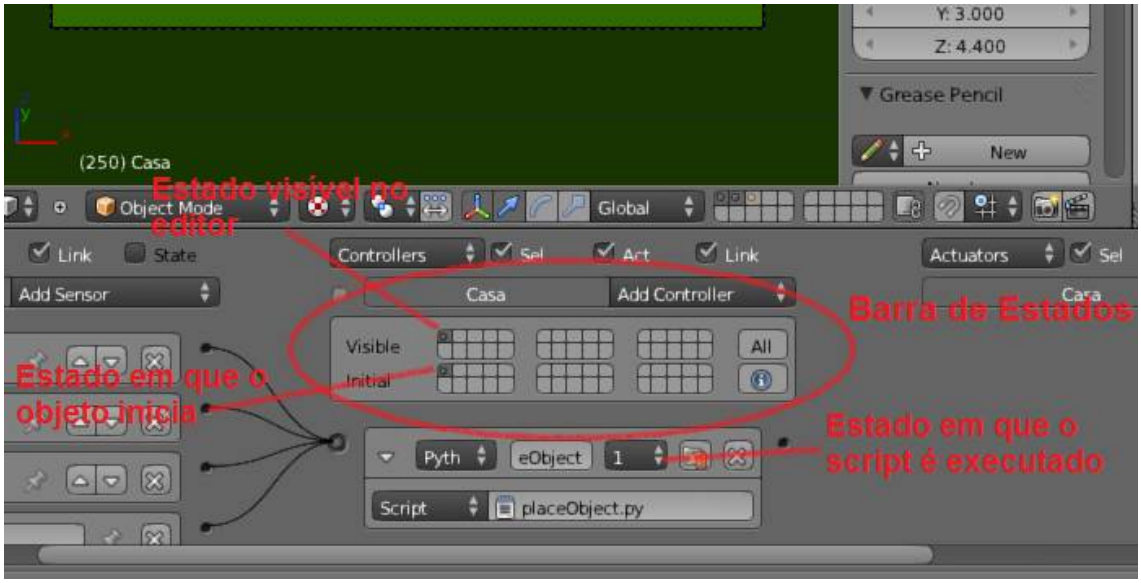


Primeiro, padronizei as dimensões do terreno. Considerando que ele deve ter o tamanho de 50x50 tiles, fiz suas dimensões em X e Y se tornarem 50.0 e as dimensões do cursor se tornarem 1.0. Próximo passo foi arredondar a posição do cursor sempre após o raycast que o posiciona no terreno, para que ele sempre esteja no lugar equivalente à de um tile. Então o script responsável pelo posicionamento do cursor no mapa ficou assim:

```
import bge
import mathutils
#Obtem o controlador que roda o script
cont = bge.logic.getCurrentController()
#Obtem quem esta rodando o controlador (este proprio objeto)
own = cont.owner
#Obtem a cena
sce = bge.logic.getCurrentScene()
#Obtem a lista (vetor) de objetos da cena
obj = sce.objects
#Obtem os sensores de verificaçao de mouse sobre alguma coisa e clique com botao esquerdo
mouseOverAny = own.sensors["mouseOverAny"]
leftButton = own.sensors["leftButton"]
#Obtem os objetos Cursor e Camera
cursor = obj["Cursor"]
camera = obj["Camera"]
#Pega a posiçao no mundo 3D em que o mouse esta
hitPos = mouseOverAny.hitPosition
hitObj = mouseOverAny.hitObject
#Camera envia um raycast da posicao do mouse para o Chao
rayObj, rayHitPoint, rayHitNormal = camera.rayCast(hitPos, None, 50, "chao", 0, 1)
#Se houve raycast
if rayHitPoint != None:
    #Passa as informacoes das coords x e y para
    #variaveis
    x = rayHitPoint[0]
    y = rayHitPoint[1]
    #Converte essas posicoes para pixels
    x = int(x+50.0)
    y = int(y+50.0)
    #Define a posicao do cursor conforme a posicao do raycast do mouse, convertido para tiles
    cursor.localPosition = mathutils.Vector((x-50+0.5, y-50+0.5, 0.025))
```

A tarefa então passou a ser fazer o objeto seguir o cursor antes de ser posto na cena, para que o usuário pudesse visualizar o espaço que este ocuparia no mapa.

Utilizei então o recurso de states (estados) dos objetos do Blender, que permite que os controladores sejam executados somente em um determinado estado. Com isso, eu poderia fazer com que a casa inicialmente seguisse o cursor (state 1), e então mudasse para um estado em que pararia de segui-lo com um clique do mouse (state 2). Foi o que fiz.



O script “placeObject.py”, responsável por fazer o objeto se mover ao local do cursor e detectar o clique esquerdo e direito do mouse, simplesmente iguala a “worldPosition”, vetor que guarda os valores de X, Y e Z do objeto em relação ao mundo, da casa a do cursor, e muda seu estado para 2 se houver clique esquerdo do mouse. Também a destrói se houver clique com o botão direito.

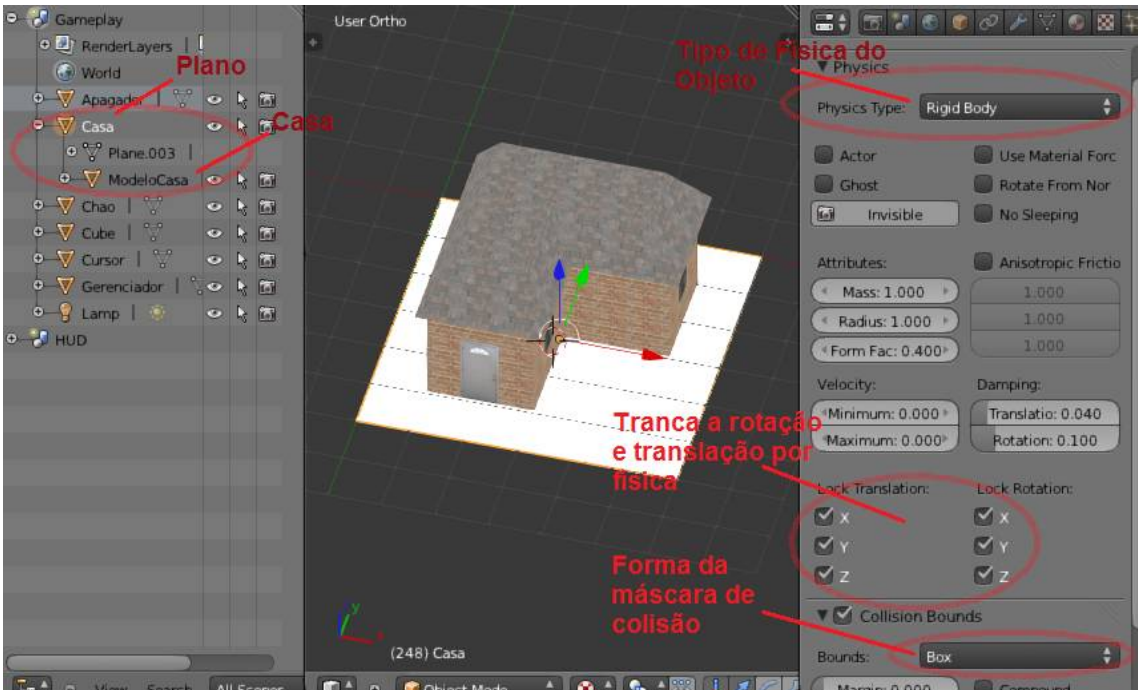
2.1.4. Configurando propriedades físicas e colisões

Com isso feito, o objeto a ser posto na cena só precisava detectar se não está por cima da área de outro objeto.

Primeiro, procurei no Blender uma opção para alterar a máscara de colisão do objeto, mas o somente permite a alteração de sua forma (cubo, esfera, etc...) mas não seu tamanho. A caixa de colisão é fixada nas extremidades do objeto.

Para contornar essa limitação, determinei que cada construção terá um plano “grudado” em sua parte inferior, que, pelo seu tamanho, representará o espaço que o objeto ocupa e será usado para detectar a colisão.

Objetos com física “static” não acionam o sensor de colisão, então optei por mudar para “rigid body” e desativar a alteração na rotação e na posição por conta da física realística.



Agora faltava somente criar o sensor de colisão, liga-lo ao script “placeObject.py” e alterar o código deste para permitir que o objeto seja posto no lugar especificado somente se não estiver ocupando a área de outro.

Então a casa colidia com qualquer objeto 3D disposto na cena, porém, ele só deveria colidir com outras construções e objetos específicos.

O sensor de colisão possui a opção de mandar pulso positivo apenas se o objeto com quem este estiver colidindo tiver a propriedade especificada.

Definindo o filtro como “objeto”, agora é só criar uma propriedade qualquer com esse nome nos modelos que devem colidir entre si. Então restou alterar o script “placeObject.py”:

```
import bge
import mathutils
#Script para fazer a casa seguir o cursor para a visualizacao de onde o objeto sera criado.
#Obtem o proprio objeto
own = bge.logic.getCurrentController().owner
#Obtem o objeto cursor
cursor = bge.logic.getCurrentScene().objects["Cursor"]
#Obtem o objeto gerenciador
gerenciador = bge.logic.getCurrentScene().objects["Gerenciador"]
#Faz com que esse objeto siga o cursor
own.worldPosition = cursor.worldPosition
#Obtem o sensor de clique esquerdo do mouse
cliqueEsquerdo = own.sensors["CliqueEsquerdo"]
cliqueDireito = own.sensors["CliqueDireito"]
#Sensor para testar colisao
colisao = own.sensors["Colisao"]
#Verifica se houve clique com o botao esquerdo e se o objeto nao esta em colisao com outro
if cliqueEsquerdo.positive and not colisao.positive:
    #Mapeia as coordenadas do cursor para as
    #coordenadas da matriz de jogo
    x = int(cursor.worldPosition[0]+50-0.5)
    y = int(cursor.worldPosition[1]+50-0.5)
    print("x: ", x, "y: ", y)
    #Muda o estado, para desativar a interacao
    own.state = 2
    gerenciador['constroi'] = False
    cursor.color = mathutils.Vector((0.75,0.75,0.75,1))
#Verifica clique direito para apagar o objeto
if cliqueDireito.positive:
    gerenciador['constroi'] = False
    own.endObject()
    cursor.color = mathutils.Vector((0.75,0.75,0.75,1))
```

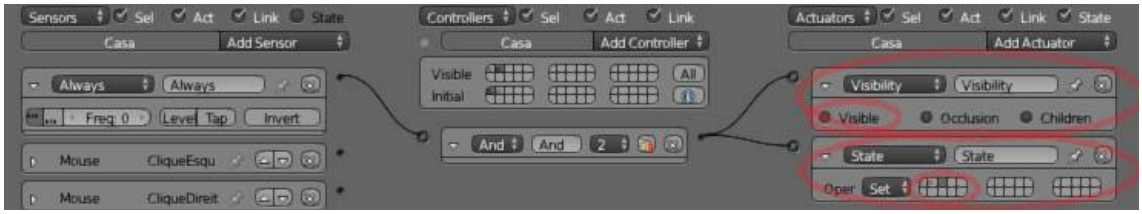
Foi então que surgiu a ideia de usar esse mesmo plano que detecta a colisão para mostrar ao usuário se a casa pode ser posta no lugar destinado, indicando a permissão conforme a cor. Para isso, apenas acrescentei as seguintes linhas ao “placeObject.py”:

```
if colisao.positive:
    own.color = mathutils.Vector((0.75,0.25,0.25,1))
else:
    own.color = mathutils.Vector((0.25,0.75,0.25,1))
```

Na linha 1, verifica se houve colisão. Então se for verdade, muda a cor do plano para vermelho. Caso contrário, a cor deve se mostrar verde.

2.1.5. Finalizando a primeira funcionalidade

Feito isso, constatei que não havia necessidade de o plano ser mostrado mesmo após a casa ser fixada na cena, logo adicionei dois atuadores ao objeto “Casa”, um para alterar a visibilidade do plano e outro para mudar mais uma vez seu estado.



Para finalizar a criação básica do objeto, faltava apenas possibilitar sua rotação quando este estivesse para ser adicionado ao mapa (não era possível no Città Java), e impedir que o objeto passe da área do terreno. Para isto, o objeto teve de ter o número de tiles em X e Y guardados, pois a rotação em 90º deve realizar uma troca entre esses dois valores, e isso afeta a centralização da construção no cursor.

Criei duas propriedades para a casa: “tilesX” e “tilesY”, que guardam valores inteiros. Defini a tecla “R” a ação de rotação. Acrescentei as seguintes linhas de código ao “placeObject.py”.

```
#Faz entao com que esse objeto siga o cursor e arredonda para o tile da esquerda superior
#se o numero de tiles do objeto for par
own.worldPosition[0] = cursor.worldPosition[0] - (0.5-((own['tilesX']%2)/2))
own.worldPosition[1] = cursor.worldPosition[1] + (0.5-((own['tilesY']%2)/2))
own.worldPosition[2] = cursor.worldPosition[2] + 0.001
#Sensor da tecla R
teclaR = own.sensors["TeclaR"]
#Verifica se a tecla R foi pressionada, entao rotaciona o objeto e faz uma troca de valores
#pros tiles em x e em y
if teclaR.positive:
	own.applyRotation([0.0,1.570796326794897,0.0], True)
	 valor = own['tilesX']
	own['tilesX'] = own['tilesY']
	own['tilesY'] = valor
	own['rotacao'] += 90
	if (own['rotacao'] >= 360):
		own['rotacao'] = 0
	print("Rot: ", own['rotacao'])
#Limita o objeto de passar da área do terreno em X
if own.worldPosition[0] - own['tilesX']/2 < -50:
	own.worldPosition[0] = -50 + (own['tilesX']/2)
elif own.worldPosition[0] + own['tilesX']/2 > 50:
	own.worldPosition[0] = 50 - (own['tilesX']/2)
#Limita o objeto de passar da área do terreno em Y
if own.worldPosition[1] - own['tilesY']/2 < -50:
	own.worldPosition[1] = -50 + (own['tilesY']/2)
elif own.worldPosition[1] + own['tilesY']/2 > 50:
	own.worldPosition[1] = 50 - (own['tilesY']/2)
```

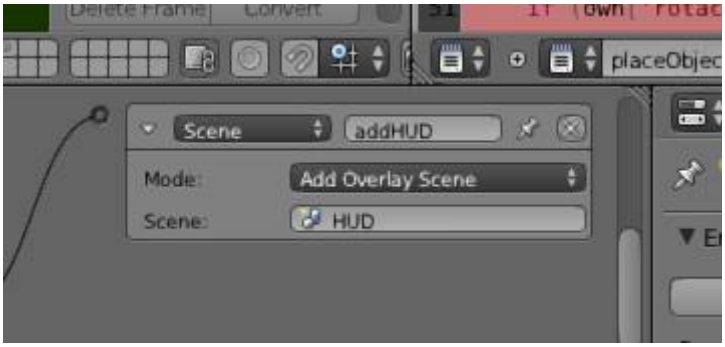
2.2. Primeira interface de jogo

Após definir a criação da casa no mundo, decidi trabalhar na interface de jogo (GUI).

2.2.1. Sobreposição de cenas

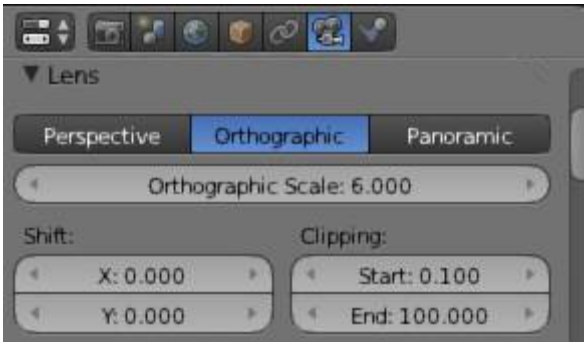
Primeiramente, pensei em pôr os objetos da GUI em uma camada diferente da mesma cena, mas depois de algumas pesquisas, descobri que é possível rodar duas cenas em paralelo na mesma janela.

Esse recurso no Blender se chama “Overlay Scene”, que sobrepõe uma cena em outra. Os blocos de lógica já possui um atuador para controle de cena, e é possível, por meio deste, adicionar uma nova cena sobreposta.



Tendo uma cena somente para o mundo de jogo e outra para a interface (que nomeei como “HUD”), bastava saber como fazer as duas interagirem, pois boa parte dos botões da GUI cria um objeto para ser posto no mundo.

Antes de tudo, configurei o objeto Camera dessa nova cena para ser ortográfica, pois é a visão comumente utilizada para visualização de modelos bidimensionais.



2.2.2 Interação entre Game e GUI

A BGE possui referências para as cenas sendo executadas e as armazena em uma lista, que pode ser obtida pelo método “bge.logic.getSceneList()”. Cada cena também possui um atributo que armazena o nome especificado para ela no editor do Blender.

Dessa forma, poderia obter uma referência para o mundo a partir da interface desta maneira:

```
for c in bge.logic.getSceneList():
    if c.name == "Gameplay":
        cena = c
```

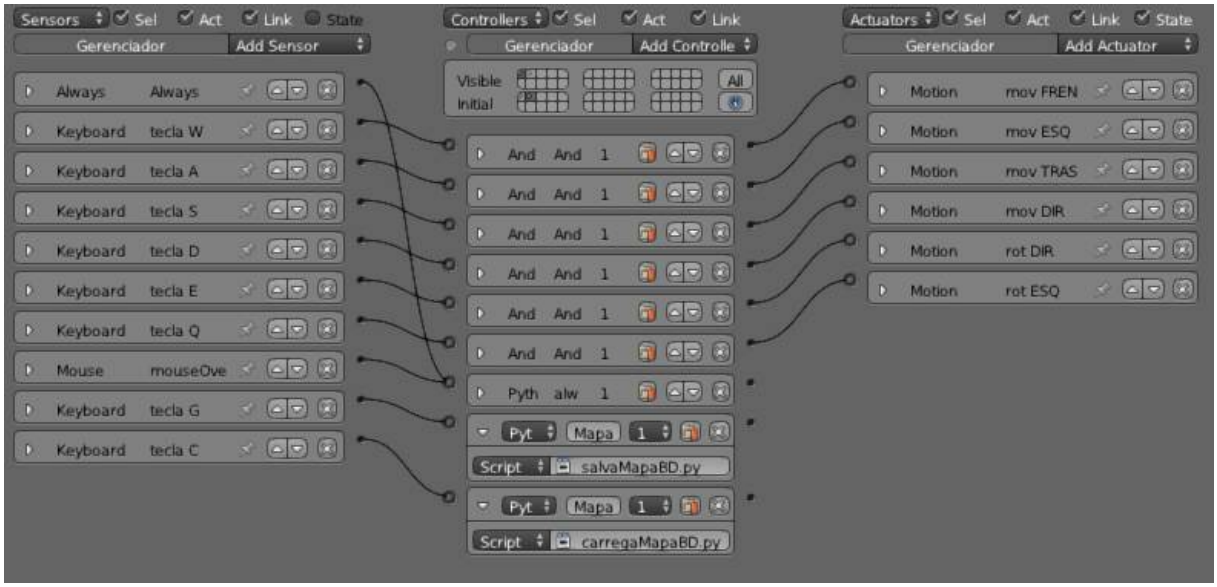
A variável “c”, especificada no comando “for” da linha 1, percorre cada cena da lista e verifica se o seu nome é igual ao da cena que quero obter (neste caso “Gameplay”). Assim que a encontra, armazena sua referência em outra variável (linha 3).

Com a referência da cena do mundo, era possível especificar uma construção a ser criada no mapa a partir de um clique em um botão da interface, que, por motivos de teste, representei com um plano simples e sem material que nomeei como “btCasa”. Logo, criei uma nova propriedade para este objeto e o “Gerenciador”, que chamei de “construcao”, e guarda o nome do objeto ser adicionado.

Para este, adicionei um sensor para detectar se o mouse está o encostando e para detectar o clique esquerdo. Ambos foram ligados a um controlador que executa o seguinte script:

```
import bge
#Referencia a cena de jogo, procurando pelo seu nome na lista de cenas ativadas
for c in bge.logic.getSceneList():
	if c.name == "Gameplay":
		cena = c
#Referencia o proprio objeto
own = bge.logic.getCurrentController().owner
#Obtem os sensores
mouseSobre = own.sensors["mouseOver"]
cliqueEsquerdo = own.sensors["leftButton"]
#Testa se o mouse se encontra por cima do objeto e se houve clique
if cliqueEsquerdo.positive and mouseSobre.positive:
	#Referencia o objeto gerenciador da cena de jogo
	gerenciador = cena.objects["Gerenciador"]
	#Altera o atributo do objeto gerenciador que marca o objeto a ser criado
	gerenciador['construcao'] = own['construcao']
```

Essa nova propriedade levou a uma pequena alteração na criação da casa, pois o método “addObject()” usa o nome do objeto para adicioná-lo à cena, e com a propriedade “construção” era possível armazenar o nome de qualquer uma delas. Caso contrário, seria necessário um código para a adição de cada construção. A estrutura de logic bricks do objeto Gerenciador e o script “always.py” deste objeto, que é responsável pelo posicionamento do cursor no terreno e por adicionar o objeto a ser fixado no mapa, ficaram da seguinte forma:



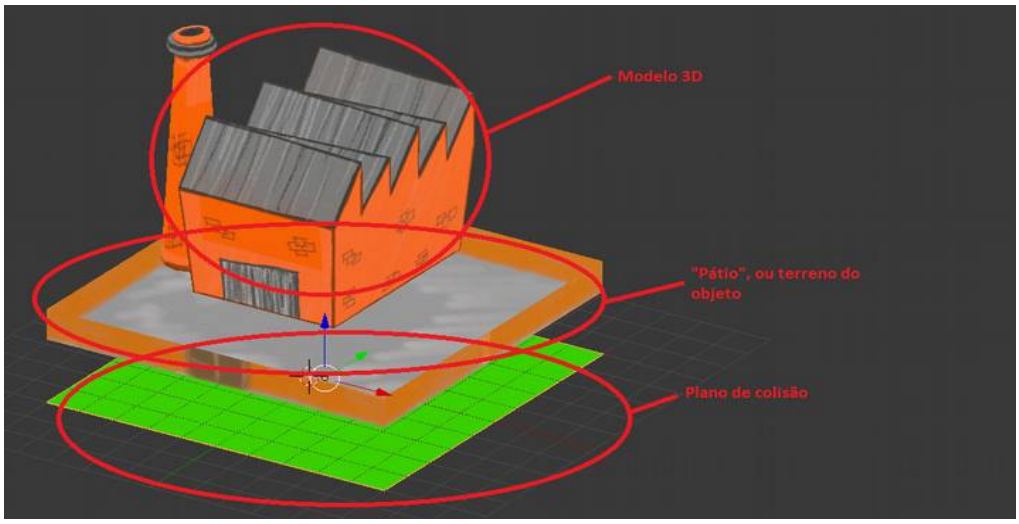

```
#Obtem os objetos Cursor e Camera
cursor = obj["Cursor"]
camera = obj["Camera"]
#Pega a posiçao no mundo 3D em que o mouse esta
hitPos = mouseOverAny.hitPosition
hitObj = mouseOverAny.hitObject
#Camera envia um raycast da posicao do mouse para o Chao
rayObj, rayHitPoint, rayHitNormal = camera.rayCast(hitPos, None, 125, "chao", 0, 1)
#Se houve raycast
if hitObj != None and cursor['ativado'] == True:
	#Passa as informacoes das coords x e y para variaveis
	x = rayHitPoint[0]
	y = rayHitPoint[1]
	#Converte essas posicoes para pixels
	x = int(x+50.0)
	if x > 99:
		x = 99
	elif x < 0:
		x = 0
	y = int(y+50.0)
	if y > 99:
		y = 99
	elif y < 0:
		y = 0
	#Define a posicao do cursor conforme a posicao do raycast do mouse, convertido para tiles
	cursor.localPosition = mathutils.Vector((x-50+0.5, y-50+0.5, 0.005))
#Caso contrario, deve "esconder" o cursor
else:
	cursor.localPosition[2] = 1000
# ”own['construcao']” verifica se ha um objeto a ser posto no cenário, e own['constroi'] não permite
# que o objeto seja criado de forma duplicada
if own['construcao'] != "Nenhum" and own['constroi'] == False:
	objeto = sce.addObject(own['construcao'], cursor)
	objeto.applyRotation([1.570796326794897,0.0,0.0], True)
	objeto.scaling = [0.5,0.5,0.5]
	own['constroi'] = True
```

Fora a interação da cena de HUD com a cena do mundo 3D, o script que coloca o objeto no mapa não pode detectar o clique com botão esquerdo caso o mouse esteja sobre algum painel da interface. Para isso, foi definido que, ao colocar o mouse sobre o painel, o cursor que percorre o mapa tenha sua posição em Z alterada para 1000. Assim, se a posição do cursor for esta, no momento do clique esquerdo, o objeto não será colocado.

2.3. Alterando a colocação de objetos

Com todos os recursos para se colocar o objeto já implementados, botamos mais opções de objetos no projeto, como prédio e escola. Porém, vimos que a forma em que a colocação foi implementada, apesar de servir para qualquer objeto, tornava o trabalho de colocar novos objetos muito demorado. E após algumas reuniões com o pessoal responsável pela parte pedagógica do projeto, foi definido que cada objeto deveria ter um "pátio" em sua base.

Para solucionar estes problemas, resolvemos dividir cada construção em três objetos: o plano de colisão, para assegurar que o objeto não será colocado por cima de uma área indevida, o "pátio", e o modelo 3D da construção de fato, que também carregará consigo os atributos do objeto a ser posto. Este plano de colisão será redimensionado conforme a quantidade de tiles que a construção ocupa em X e em Y, valores estes armazenados em game properties do modelo 3D da construção a ser posta. O modelo também deve armazenar os IDs desses dois objetos auxiliares, de forma que o script que define sua posição possa definir destes também.



2.4. Alteração de terreno

2.4.1. O algoritmo de deformação de malha

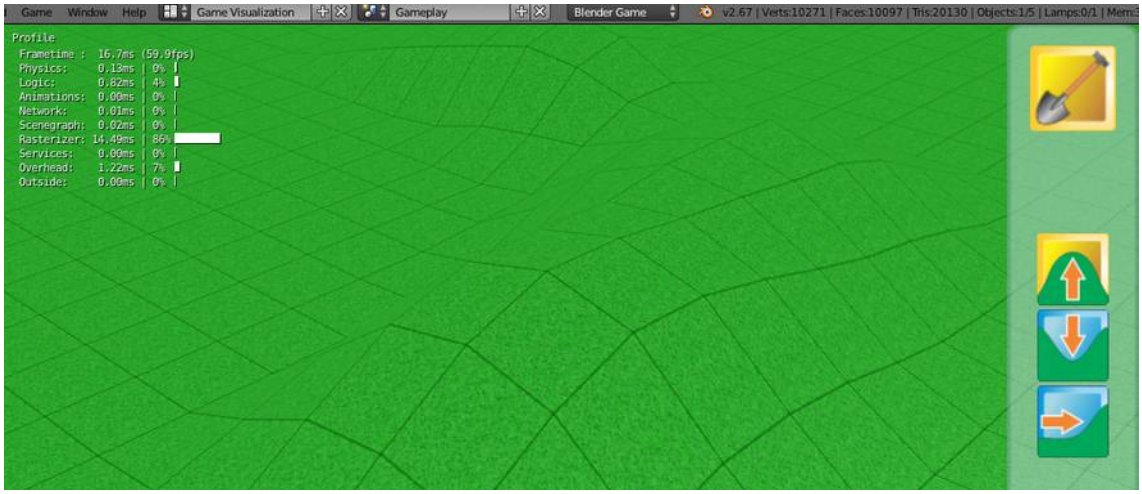
O próximo passo então era implementar a possibilidade de alteração do terreno. Baseando-se no algoritmo para Blender escrito por Captain Oblivion no fórum Blender Artists (http://blenderartists.org/forum/showthread.php?133308-reinstancePhysicsMesh%28%29-test), tivemos de dividir a malha do terreno em 10201 vértices, pois usamos um terreno referenciado por 100x100 tiles. O algoritmo então realiza alterações no eixo Z dos vértices que estiverem em um determinado raio em volta do cursor do mouse.

Foi então evidenciado um problema com a utilização deste código. Toda vez em que o terreno era clicado, o script verificava cada vértice da malha para obter sua posição e poder realizar a alteração. Considerando que a malha do terreno do Città teria 10201 vértices, isso seria altamente custoso para o processamento.

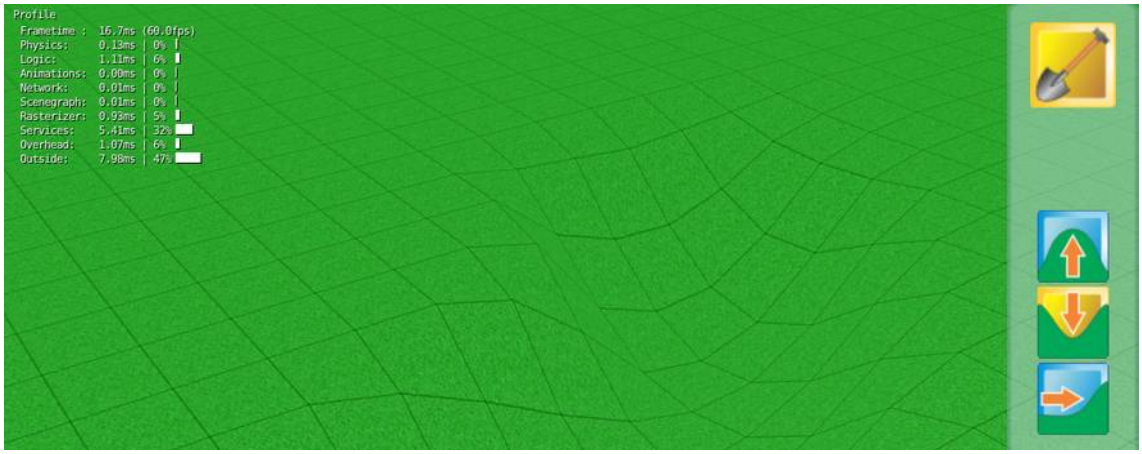
A solução então foi mapear a lista de vértices do objeto para uma matriz 101x101, ordenada conforme a posição XY de cada vértice. Dessa forma, conseguiríamos fazer o acesso direto ao vértice de uma determinada posição, sem ter de percorrer a lista de vértices.

Feito isso, tínhamos um botão da interface para realizar:

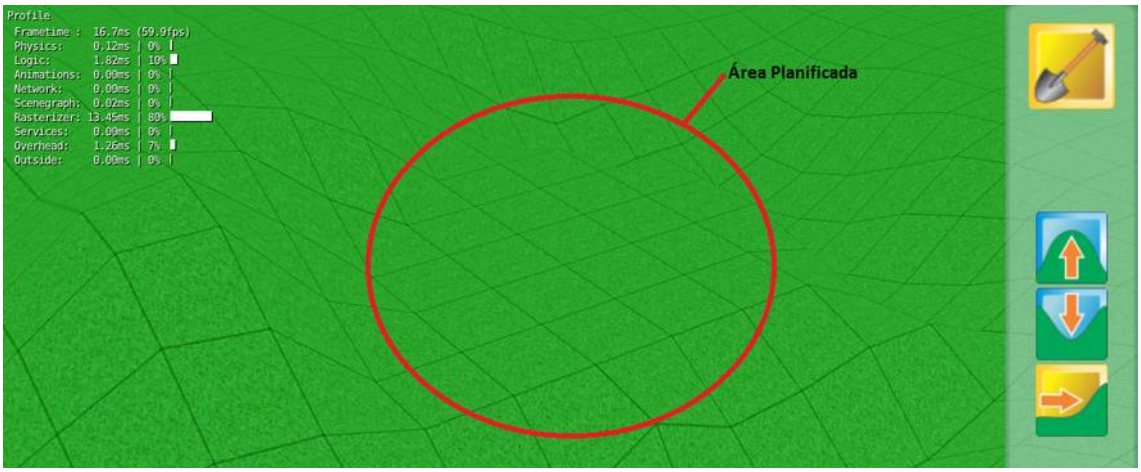
Aclives (função que adiciona valor de Z aos vértices):



Declives (que subtrai o valor de Z):



E planificação, que adiciona valor à posição Z do vértice se esta for abaixo de 0, e subtrai se for maior que 0, normalizando a malha naquela posição:



Essas alterações são todas feitas gradualmente, de forma a se ter uma malha suavizada.

2.4.2. Interação entre terreno e objetos

Outro desafio era fazer com que o jogo impedisse o jogador de realizar alterações onde houvesse outro objeto e de pôr um objeto onde o terreno não fosse plano.

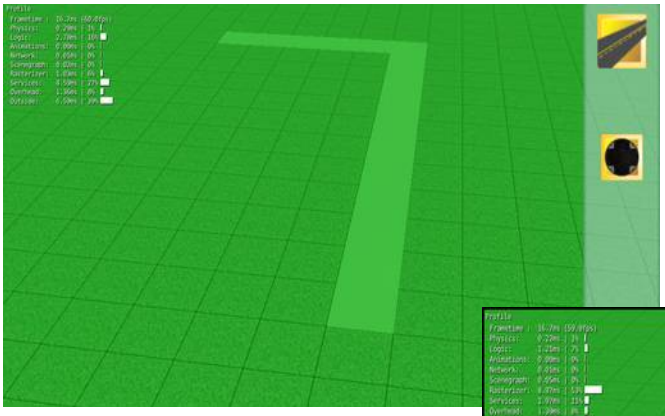
Assim, definimos uma esfera de colisão que seguiria o mouse quando o usuário fosse alterar o terreno, e no caso de colisão com algum objeto, o script não permitiria a alteração. E o colisor das construções deveria verificar se a posição em Z de cada vértice no espaço onde o objeto seria colocado fosse 0. Se sim, o objeto poderia ser posto ali.

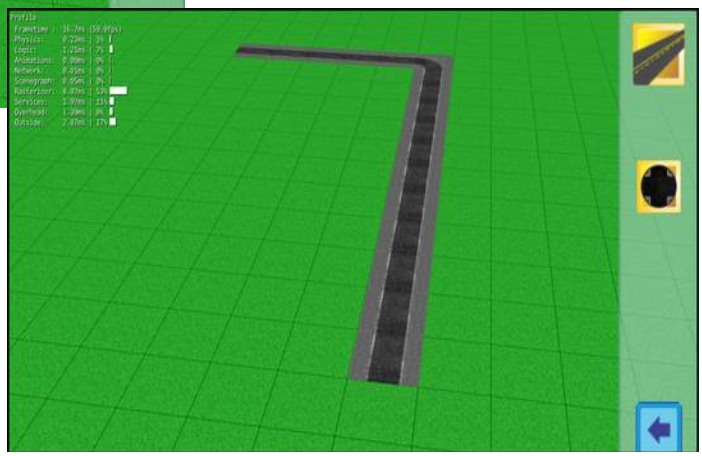
2.5. Tentativa de algoritmo de estradas

Então, o último passo para se ter uma versão “jogável” do simulador, era possibilitar a construção de estradas, de forma intuitiva.

Primeiramente, tentamos um algoritmo que alterava a escala de dois planos auxiliares conforme a posição selecionada com um clique e a posição do cursor, e que então deveria criar cada tile de estrada conforme o tamanho destes objetos. O algoritmo, porém, tinha dificuldades em realizar colisões com outros objetos, pela forma que o Blender lida com malhas físicas, tornando-o instável, ainda que utilizável. Assim, partimos para um próximo algoritmo.

Este então deveria guardar a posição do cursor no momento do clique do jogador, e a partir dessa posição com a posição onde o mouse estiver, criar e posicionar os tiles de estrada. Cada tile de estrada então deve detectar sua conexão com seus quatro lados, e então alterar sua textura dependendo dessas conexões.





A implementação deste algoritmo ainda está em desenvolvimento. Para que pudéssemos apresentar uma versão jogável do simulador para as escolas, utilizamos o primeiro algoritmo de estradas, e realizamos algumas alterações no projeto para se adequar a este código.

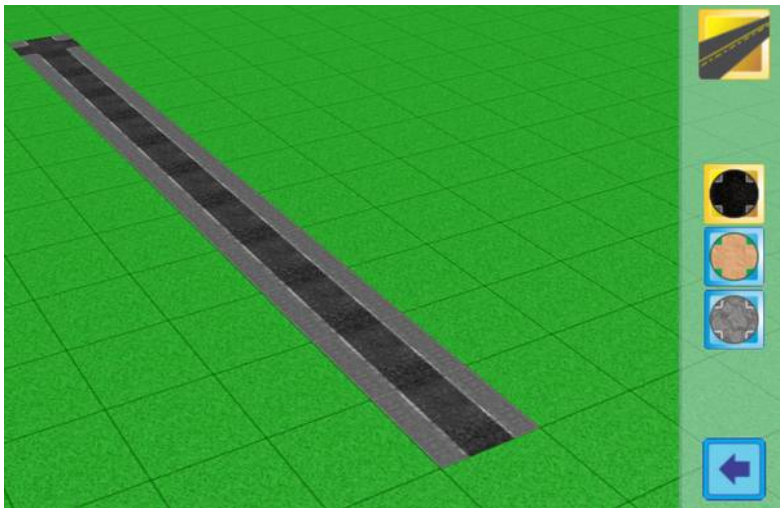
2.6. Primeira implementação de Menu Inicial 3D

Foi implementado também um novo menu inicial em 3D, com objetos interativos, que executam animações quando o mouse é colocado por cima deles, e que poderão ser clicados. Atualmente, só a mochila e o caderno podem ser clicados pelo usuário.

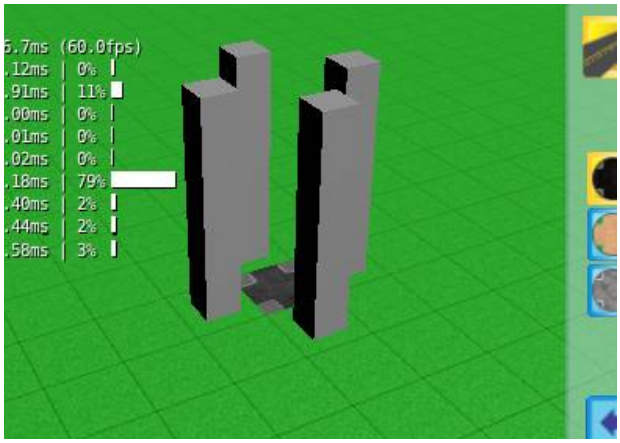
2.7. Segundo algoritmo de estradas

Após ter problemas com a implementação do algoritmo de estradas na BGE, foi trabalhada a seguinte solução:

O algoritmo ainda recebe a posição do mouse no terreno, mas agora cada tile de estrada é colocado em tempo real, conforme a distância da posição do mouse em relação à posição em que foi realizado o clique, até que o usuário solte o clique no mouse.



Cada estrada possui quatro colisores (chamados de ConectorEstrada) anexados em cada uma de suas quatro direções: cima, baixo, esquerda e direita. Estes colisores são os responsáveis por detectar as conexões entre as estradas que já estavam no mapa. Estes colisores são invisíveis para o jogador.



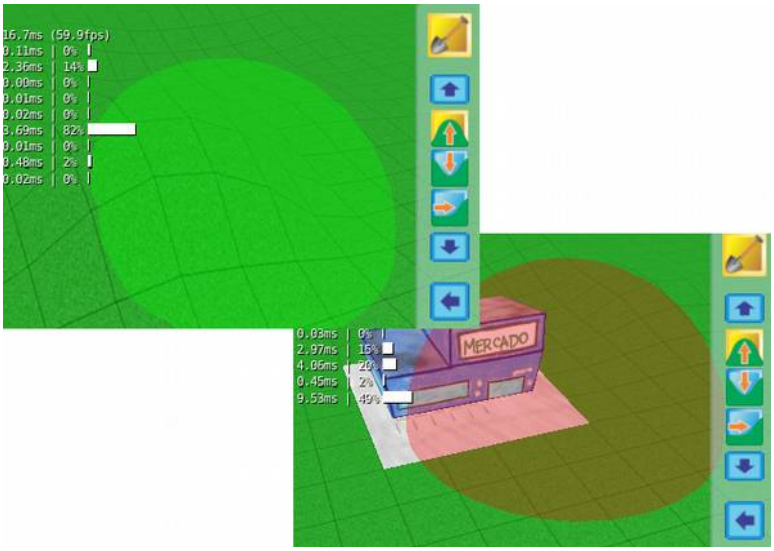
Ao soltar o botão do mouse, estes colisores detectam as colisões com outras estradas do mesmo tipo (asfalto, chão batido, paralelepípedo), e a textura da estrada é alterada conforme o número de colisões detectadas, e onde essas colisões ocorreram.

Para não haver a sobreposição de estradas com outros elementos do cenário, a estrada possui uma propriedade “colisor”, para que os colisores das construções também consigam o detectar.

Ao colocar as estradas definitivamente, cada estrada verifica se os quatro vértices do terreno correspondentes à sua posição têm sua worldPosition[2] (posição em Z) igual à 0, o que significa que aquele tile está normalizado, e a estrada será fixada ali. Caso contrário, a estrada simplesmente será apagada quando fosse ser posta no lugar indevido.

2.8. Marcador de colisão de terreno

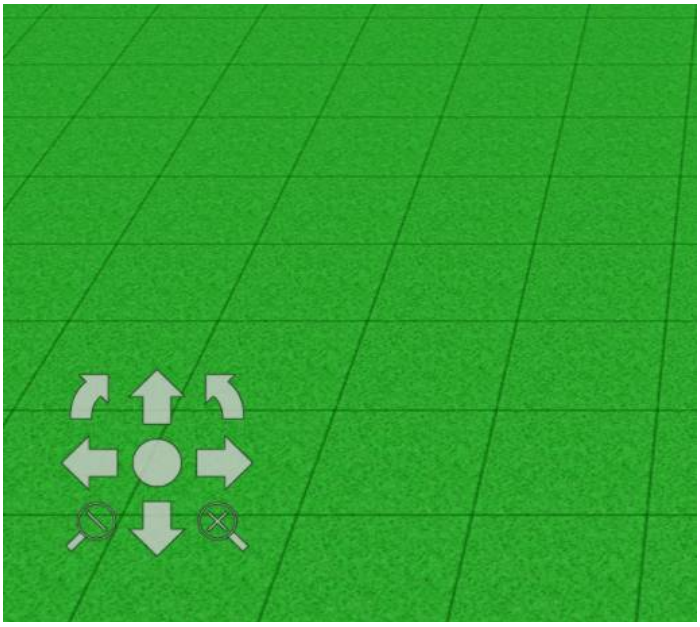
Também foi adicionada uma esfera de colisão para as alterações no terreno. Assim, o usuário pode saber o raio em que a alteração será feita, e se não há nenhum objeto (construção, estrada, água) que poderia interferir no processo, fazendo com que a esfera oscile entre as cores verde (pode alterar) e vermelho (alteração não aceita naquele local).



2.9. Interface de navegação de câmera

Por meio do feedback das crianças que tiveram a oportunidade de testar o jogo, percebemos que a interface do jogo ainda não era a adequada. Os comandos de teclado ainda não eram claros, principalmente para os comandos de navegação de câmera, e nosso objetivo era diminuir o número de possíveis inputs que o usuário deve ter à disposição para realizar qualquer tarefa, visto que o jogo foi pensado para crianças.

Tivemos então a ideia de criar uma interface para navegação de câmera, que ficaria disponível na tela de jogo, ocupando um espaço relativamente pequeno no canto inferior esquerdo, e teria comandos para mover a câmera para os quatro lados, rotação de câmera, zoom in/out e um botão direcional, em que o jogador poderia mover a câmera lentamente para todos os lados. Para utilizar esses botões, o jogador somente teria de clicar nestes.



Essa interface também seria útil para uma possível versão para dispositivos móveis, pois nesse caso o usuário somente conta com os toques na tela.

Para fazê-la, foi necessário passar as funções de rotação de câmera e zoom, que já estavam implementadas, para scripts separados, pois então poderíamos chamar cada script separadamente, dependendo do botão ou tecla que foi pressionado. O resultado foi dois scripts: “cameraMovement.py” e “cameraZoomInOut.py”.

Outra modificação foi fazer com que os objetos sempre iniciem com sua orientação voltada para a câmera, no momento de serem postos na cena. Dessa forma, não é necessário qualquer botão ou tecla para girar um objeto.

2.10. Implementação de água

Para a funcionalidade de colocar água na cena, foi usado um algoritmo bastante parecido com o da colocação de objetos, com a diferença de que temos um objeto gerenciador (ColisorAgua), e ele é o responsável por colocar a água no tile onde se encontra o mouse.

Os tiles de água possuem um colisor, assim como os objetos e as estradas, e possuem a propriedade "colisor", para serem tratados como estes outros objetos. Os tiles de água também detectam se a área onde serão postos não se encontra deformada.

Não bastava somente colocar objetos no cenário, tínhamos que também proporcionar ao usuário uma ferramenta para apagar objetos já existentes. Então criamos uma ferramenta "borracha", em que o jogador pode clicar sobre o tile em que um objeto estiver posicionado, e aquele objeto deixa de existir no mapa, tornando os tiles livres para que o usuário possa colocar um outro objeto no mesmo local.

Esse objeto deve ser configurado para conseguir apagar todo o tipo de objeto: construção, estrada e água.

2.11. Finalização do Menu Principal 3D

Trabalhamos então na finalização do Menu Principal, que deveria possuir funções para criar um novo mapa, carregar um mapa já existente, conferir os créditos, as instituições envolvidas no projeto e o tutorial.

Ao colocar o mouse por cima de uma opção, o usuário consegue ver a opção na parte superior da tela, escrita em laranja.

Para isso, foram criados objetos do tipo "Text", que foram transformados em "Mesh" extrudados, para se obter o efeito 3D no texto.

Os textos iniciam a cena invisíveis, e se tornam visíveis quando o jogador põe o mouse por cima do respectivo objeto daquela opção. Se tornam invisíveis novamente ao se tirar o mouse de cima do objeto.

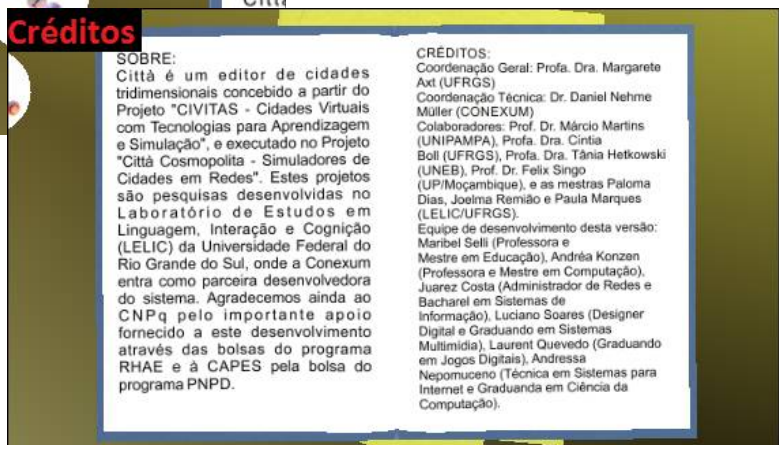
Ao receber um clique, o objeto envia uma "mensagem" para a câmera, que desabilita o clique nos outros objetos durante a animação, e inicializa as ações daquela opção.

Uma vez que a opção estiver selecionada, o usuário pode clicar fora do objeto para voltar para o início e selecionar as outras opções. Ao fazer isso, também é enviada uma mensagem para o objeto câmera, que então executa a animação para voltar ao início, e habilita novamente os cliques ao fim da animação.

O caderno possui uma página para criar novo mapa, com uma barra de texto selecionável para o usuário inserir o nome da cidade a ser criada, e uma segunda página para carregar um mapa, com uma lista de mapas que são carregados direto da pasta “mapas”, no diretório do jogo, incluindo botões de rolagem de lista. Ao escrever o nome ou selecionar o mapa, o usuário somente deve clicar no botão de “Ok” disponível na respectiva página.

O usuário também possui à disposição um mini tutorial de como jogar, acessível pelo gameboy. Com os botões do próprio objeto, o usuário pode passar ou voltar as páginas de instrução.

As demais opções são apenas de visualização, não possuindo nenhum comando adicional.



E ao clicar na mochila, o jogo termina imediatamente

O projeto é disponibilizado no SourceForge, pelo link: https://sourceforge.net/projects/civitas/

Após finalizar o Menu Principal por último, disponibilizei a versão executável 0.2.1.0. do jogo para Windows (link: https://sourceforge.net/projects/civitas/files/latest/download?source=updater), sob a licença GNU General Public License (GPL) (link: http://www.gnu.org/copyleft/gpl.html), sendo assim, um software livre. Seu código pode ser acessado pelo seu repositório no SourceForge.

3. Curso 3DS Max

Nos dias 8 e 9 de Novembro de 2012, participei do mini-curso de 3D Studio Max, promovido pela parceria com o grupo Geotec e UNEB.

O curso foi ministrado pelo designer e professor do IFBA Edson Machado, e as aulas ocorreram durante o turno da manhã e da tarde, no Laboratório de Estudos em Linguagem, Interação e Cognição (LELIC), localizado na Faculdade de Educação da Universidade Federal do Rio Grande do Sul.

Com o objetivo de introduzir a ferramenta 3DS Max e os conceitos básicos de modelagem e design 3D, foram realizadas algumas séries de exercícios, em que tivemos de modelar alguns objetos 3D usando as técnicas que nos foram ensinadas.

Nosso trabalho final foi modelar um ou mais objetos específicos, diferentes para cada participante do curso, para então gerar uma cena tridimensional unindo todos esses objetos. Para mim, a tarefa foi modelar uma mesa, uma cadeira e um copo.

4.

4. Pesquisa de campo em Estrela

No dia 23/05/2013, visitamos uma escola na cidade de Estrela, para acompanhar uma das dinâmicas do projeto Civitas, sendo aplicada pela pedagoga e colega de pesquisa Maribel Selli, em uma turma de 4º ano do ensino fundamental.

Em certo momento, foi perguntado aos alunos “O que uma cidade imaginária deve conter?”. Cada criança sugeria um tipo de construção e objeto que gostariam que a cidade

imaginada pela dinâmica tivesse. Diversos objetos, como flores, lagos, rios, casas, mercado, igreja, entre muitos outros, foram requisitados pelas crianças.

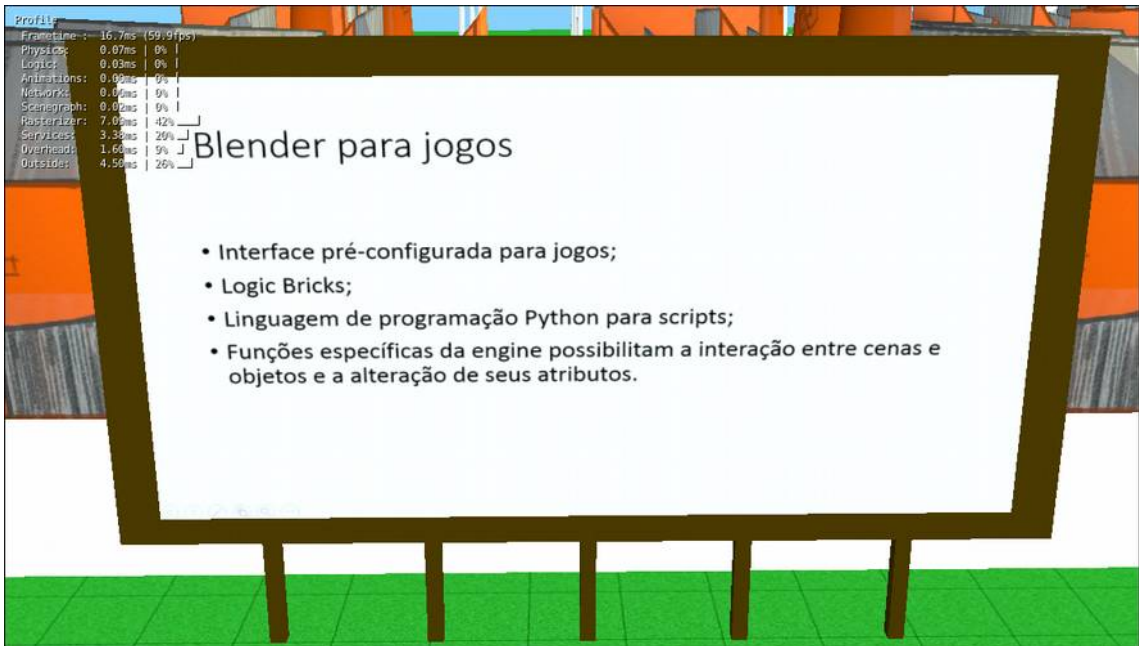
Todas essas sugestões foram anotadas, e nos permitiu visualizar o que as crianças esperam do Città, especialmente da parte visual. Esta lista também nos permitiu reorganizar os ícones de objetos em ordem de prioridade.

5. Apresentação do Città no FISL14

No dia 3 de julho de 2013, o grupo do Projeto Cosmopolita apresentou o Città na 14ª edição do Fórum Internacional do Software Livre (FISL14).

Para a apresentação, ao invés de usarmos slides, preparamos uma aplicação interativa em Blender, que simulava uma cidade feita em 3D, substituindo os slides por outdoors dessa cidade. Para cada outdoor, era realizada uma animação de movimento de câmera. Eu fiquei encarregado de programar a função de avançar/voltar outdoor, e ajustar a performance do aplicativo.

Na minha parte do discurso, discorri sobre o uso do software Blender no desenvolvimento de jogos, os ganhos e perdas que tivemos com a escolhe desse programa para desenvolver a nova versão do Città, e as novas funcionalidades que a Blender Game Engine nos permitiu implementar. Então mostrei o protótipo do jogo, que já possuía as funções de alteração de terreno e colocação de objetos.



6. Primeiro Curso de jogos em Blender em Salvador

No mês de julho, dos dias 17 a 20, fui, acompanhado de outros dois bolsistas, à cidade de Salvador, na Bahia, para trocar experiências com o Grupo de Geotecnologias, Educação e Contemporaneidade (GEOTEC), da Universidade do Estado da Bahia (UNEB), e também para ministrar um minicurso de desenvolvimento de jogos na Blender Game Engine. Chegamos lá na quarta-feira de noite e voltamos no sábado pela manhã.

Para que pudéssemos ter mais tempo para compartilhar informações com o grupo de lá, os três cursos (o meu e dos outros bolsistas) ocorreram todos no mesmo dia, pelo período da tarde.

Como tarefa do curso que ministrei, os inscritos tiveram de criar um simples jogo do tipo Labirinto, modelando um labirinto simples, e criando um objeto interativo. Este objeto tinha de detectar a colisão com as paredes do cenário e sofrer intervenção da física.

O usuário deveria poder movimentar o objeto por comandos do teclado. Essa funcionalidade tinha de ser primeiramente implementada por meio de Blocos de Lógica, e logo após, utilizando scripts em Python.

A maior parte dos inscritos conseguiu realizar as tarefas sem grandes problemas. Para aqueles que não conseguiam progredir, os bolsistas que me acompanhavam se prontificaram a tirar dúvidas durante o curso.

7. Artigo "Interface para Jogos Educacionais"

Durante o mês de julho de 2013, escrevi, juntamente com Luciano Soares e Daniel Müller, o artigo ***Interface para Jogos Educacionais: o impacto da usabilidade no simulador de cidades Città***, que seria disponibilizado no livro: *JOGOS-SIMULADORES: relatos de*

experiência sobre o desenvolvimento do Cittá e do Kimera, Daniel Nehme Müller, Tânia Maria Hetkowski, Margarete Axt e André Luiz da Silva (orgs). Salvador: EDUNEB, 2014 (no prelo).

O artigo visa a descrição do processo evolutivo da interface do Città, e a compara com suas versões anteriores, se baseando em estudos feitos na área de IHC (Interface Homem-Computador).

Fiquei encarregado de descrever e comparar a parte funcional de interface do simulador, abordando tópicos como a facilidade de uso da interface, o espaço ocupado na pela mesma, entre outros assuntos estudados na construção de interfaces para aplicações interativas.

8. Teste do Città Blender na escola

No dia 9 de dezembro de 2013, fomos para a cidade de Segredo, para instalar um dos protótipos do Città Blender nos computadores de uma das escolas onde o Projeto Civitas é aplicado, e acompanhamos de perto as crianças durante os testes com o jogo.

Tivemos problemas para instalar nos computadores da escola, pois as máquinas não identificavam os pendrives e não tínhamos acesso à internet. Para que as crianças pudessem testar o projeto, cedemos nossos notebooks com o jogo instalado.

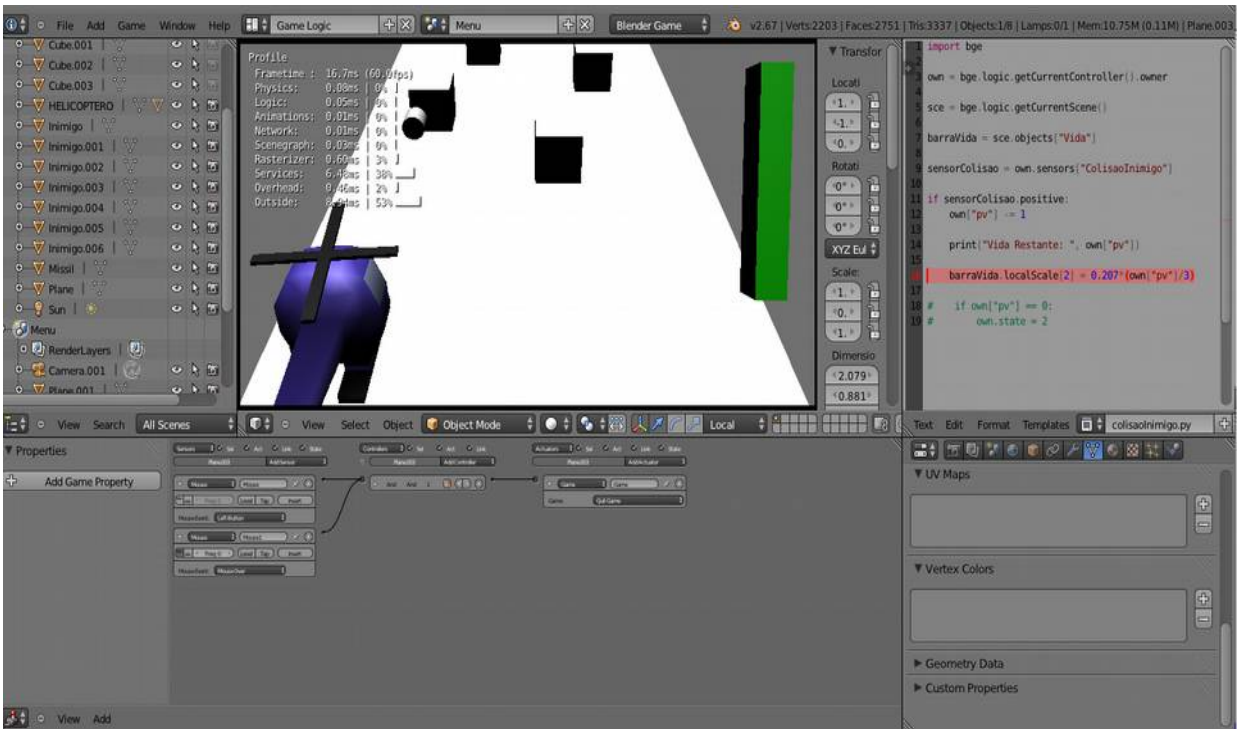
Pudemos analisar algumas falhas na interface, como a falta de informações para os comandos de teclado, alguns problemas de performance na colocação de estradas (ainda estava sendo utilizado um algoritmo provisório), e pudemos identificar um bug que ocorria na interface após a colocação de estradas no mapa, em que alguns ícones do menu ficavam sobrepostos. Também estávamos utilizando um algoritmo provisório para o salvamento do mapa criado pelas crianças.

O jogo foi elogiado pelos usuários e pela professora, pela sua funcionalidade, ainda com algumas funções não implementadas, e pela sua parte gráfica.

9. Segundo Curso de jogos em Blender em Salvador

No mês de janeiro de 2014, do dia 28 ao 31, fizemos mais uma viagem para Salvador, com a proposta de refazer o curso de desenvolvimento de jogos com Blender na UNEB.

Para isso, preparamos um cronograma completo do curso, uma introdução ao Blender e ao que o curso abrangeria, e um projeto exemplo do jogo que seria desenvolvido pelos participantes.



A primeira parte do curso se destinava ao ensinamento de algumas técnicas básicas de modelagem de objetos no Blender, e foi passada aos participantes pelo bolsista Luciano. Nessa parte, os participantes aprenderam a modelar o helicóptero que seria utilizado no jogo.

A segunda parte foi ministrada por mim, e se destinava a ensinar o funcionamento básico da BGE, os conceitos de Blocos de Lógica, a montagem da cena e o uso dos scripts no jogo.

Foram quatro dias de aulas práticas, em que os participantes acompanhavam, passo a passo, o desenvolvimento de um jogo no Blender.

10. Oficina no FISL15

No dia 8 de maio de 2014, participei da 15ª edição do FISL, como palestrante da ***Oficina de Desenvolvimento de Jogos em Blender***, com o auxílio da bolsista Andressa Nepomuceno.

Foram 180 minutos focados no desenvolvimento de um jogo simples no Blender, de forma a ensinar aos participantes o funcionamento básico da BGE.

Para a oficina, preparei um projeto-exemplo do que seria feito durante o curso, e uma apostila com o passo a passo de como esse projeto foi feito, para os alunos refazerem os passos em casa.

No início do curso, os alunos interessados receberam um projeto base em Blender, com a estrutura básica do projeto já feita, de forma a focarmos somente nos assuntos que o curso deveria cobrir, e um arquivo, também em Blender, com modelos 3D e texturas feitas pelo Luciano Soares, produzidos exclusivamente para o curso.

11. Referências

Blender Wiki. Logic, Properties and States:

http://wiki.blender.org/index.php/Doc:2.6/Manual/Game_Engine/Logic

Blender API Documentation:

http://www.blender.org/documentation/blender_python_api_2_63_17/contents.html

BlenderTips.com. Blender 3D Hotkeys, Shortcut and Reference:

http://blendertips.com/hotkeys.html

Python 2 Documentation:

http://docs.python.org/2/

Downey, A. Elkner, J. Meyers, C. (2010). Aprenda Computação com Python:

http://olamundo0.files.wordpress.com/2010/07/aprendacomputaocompython.pdf

RELATÓRIO DO PROJETO CITTÀ COSMOPOLITA

Projeto processo CNPq: *551846/2011-9*

Relatório de Atividades do Bolsista: *Andressa Cruz Nepomuceno*

Bolsista processo CNPq: *374840/2012-1 e 350783/2013-6*

Coordenador: *Daniel Nehme Müller*

Período do Relatório: *novembro/2012 a agosto/2013 e setembro/2013 a maio/2014*

Planos de Trabalho:

Estudo dos módulos e plugins de áudio que podem ser usados com Blender; acoplamento de sistema de reconhecimento de fala ao Blender; testes de interface de fala; visitas de campo nas escolas; participação de reuniões; documentação técnica.

Monitoramento e atualização dos servidores, páginas web e sistemas. Desenvolvimentos dos sistemas de chat (bate-papo) de texto e voz do jogo Città. Participação de reuniões. Participação de visitas de campo nas escolas. Elaboração de atas e relatórios.

---

1. Introdução

Neste documento estão relatadas as atividades que exerci como bolsista do CnPQ, no projeto de desenvolvimento do simulador de cidades (Città), parte do projeto CIVITAS. O Città é realizado por bolsistas e pesquisadores da empresa Conexum, em parceria com Laboratório de Estudos em Linguagem, Interação e Cognição (LELIC) da Faculdade de Educação da UFRGS.

2. Visão do Projeto

Desenvolvimento de um software gráfico interativo (jogo) para apoiar pedagogicamente na construção de conhecimento das crianças de 4º. e 5º. ano do ensino fundamental, no que se refere ao desenvolvimento de capacidades que lhes permitam pensar-construir-organizar uma cidade virtual a partir da realidade infantil e seus conhecimentos incorporados através dos conteúdos oferecidos no aprendizado dos sistemas que regulam uma cidade e suas relações.

3. Desenvolvimento

Pensando em um futuro em que o Città terá uma rede de compartilhamento de ideias, com um jogador em sua escola conectado a internet jogando com outros jogadores, em outras escolas/cidades. Crianças construindo uma cidade juntas, trocando ideias e trabalhando em grupo. Com isso, a necessidade de comunicação entre os jogadores, que estarão construindo uma cidade no Città, é tão fundamental quanto o próprio modo online.

Vi anteriormente que não poderá ser feito agora um sistema de voz para esta troca de informação entre as crianças, mas o sistema de comunicação em texto já está pronto, esperando o multiplayer para ser implementado.

Anteriomente foi feito pesquisas sobre as tecnologias existentes para fazer uma aplicação de streaming e como poderia ser desenvolvido isto na linguagem de programação interpretada de alto nível Python, dentro do software Blender. Foi estudado também a possibilidade de o Città ter uma versão para navegador, tendo assim a opção de se fazer o chat com WebRTC.

Este estudo anterior foi focado na possibilidade de se implementar as ideias, levando em consideração a limitação que temos dentro do Blender. Durante todos os testes, houve uma restrição a linguagem Python dentro do Blender, uma vez que o Blender não foi feito para uma programação muito complexa. O suporte que o Blender dá para scripts complexos em Python é pouco, e sua documentação não ajuda muito. Mesmo procurando em fóruns da comunidade, não achei nenhum exemplo ou caminho para seguir. Lembrando também que vários jogos online não têm este sistema de chat dentro do jogo, os usuários usam softwares em paralelo com o jogo para a comunicação.

Agora o objetivo é interligar o multiplayer estudado pelo bolsista Juarez Costa com o chat que eu mesmo fiz. A grandes questão que ficou no final desta parte, é que os computadores utilizados para se jogar o Città são muito defasados, então a grande dificuldade foi fazer esse sistema de intenso transmissão de dados com rapidez.

3.1. Burster

Burster é um plugin de código aberto criado pela empresa polonesa iTechnologie que executa o arquivo .blend (extensão dos projetos feitos no Blender) no navegador, utilizando uma versão modificada/adaptada do Blender Player (programa usado para executar um projeto) como base. Deve ser instalado no navegador e possui a opção de criptografia para que, mesmo fazendo o download do .blend, não seja possível modificá-lo. Existem algumas limitações, como por exemplo a compatibilidade com placa de vídeo. Antes deste plugin existir, existia outro no próprio Blender, chamado Blender Web Plugin, porém foi descontinuado por questões de segurança.

Nos testes que fiz, pegando o Citta e executando com o Burster, os cliques no mapa não funcionou. A causa disso é porque por alguma razão ali com o Burster não estava sendo reconhecido o comando Raycast. Este comando permite pegar a posição do mouse. Tudo indica que seja outra falha do Burster. Não achei nenhuma documentação em fóruns da comunidade Blender/Burster referente a esta falha.

Levamos em consideração que é um projeto que não está estável ainda, como eles mesmo, em diversos textos, falam que é um projeto em construção. Sendo assim, não será possível, ao menos por enquanto, utilizar este plugin, pelo menos até que a empresa lance uma versão atualizada sem as limitações atuais.

3.2. WebRTC

WebRTC é uma API elaborada pela World Wide Web Consortium (W3C) que permite executar aplicações de videoconferência sem a necessidade da instalação de plugins para isto. Antes de existir essa API, a maioria das aplicações com chamadas de voz/vídeo eram feitas em Flash, tecnologia defasada.

Como vimos que o Burster não está estável o suficiente para ser usado no projeto, WebRTC deixou de ser útil para o chat.

3.3. Streaming de áudio

Streaming nada mais é que um fluxo de mídia, que é a transmissão de dados, reproduzindo a mídia à medida que chega a informação no usuário. A ideia é não precisar fazer o download e salvar no HD, sendo utilizado nas chamadas de voz/vídeo. A ideia aqui era implementar isso em Python dentro do Blender.

No Python, a biblioteca usada para streaming de audio é a PyAudio e com os testes que fiz no terminal de comando (estava usando sistema operacional Linux Debian) com os scripts de exemplo vi que realmente funcionava. Porém, dentro do Blender ele fechava abruptamente quando tentava executar o mesmo. Pesquisei e achei uma proposta feita para a comunidade Blender pedindo para que seja reescrito o sistema de áudio, pois o código atual, segundo o autor da proposta, “sofre atualmente de código ruim e falta de manutenção”.

Mesmo colocando o código dentro do .blend, ou usando as bibliotecas do Python do computador, quando era executado o Blender fecha. Essa parte de áudio está sendo visto na comunidade Blender. Acredito que seja questão de tempo, assim como outros assuntos tratados neste documento.

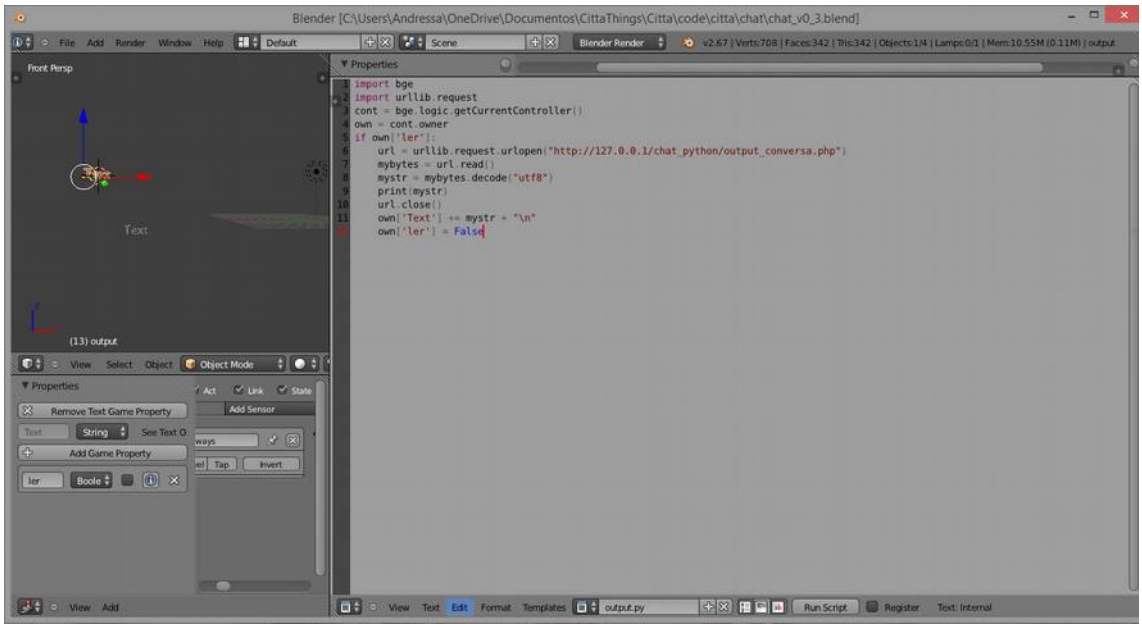
3.4. WebService e o Chat modo texto

Web Service é a solução para integrar aplicações distintas. E é exatamente o que precisamos. Como o Burster não funcionou, logo não podemos usar o WebRTC para fazer o chat texto. Então faremos dentro do blender mesmo, e a maneira de salvar informações em nuvem de uma aplicação construída no Blender, é utilizando WebService.

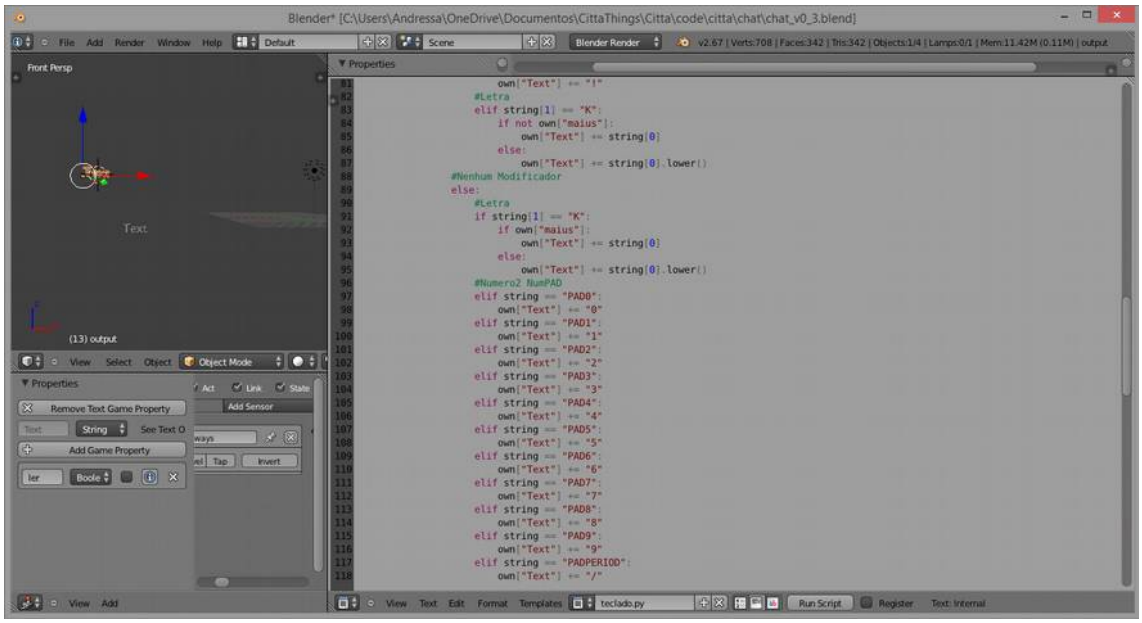
O HTTP (em português Protocolo de Transferência de Hipertexto), é um protocolo de comunicação, utilizando o Modelo OSI. Este protocolo é o que fez a internet (World Wide Web) funcionar, que é uma rede de HTTP 'conversando' entre si. Hipertexto é que se movimenta nessa operação. O HTTP utilizada este texto estruturado (que utiliza ligações lógicas, hiperlinks) para a transmissão a informações.

URL (em português Localizador-Padrão de Recursos, é o endereço de um recurso disponível em uma rede (seja internet ou intranet). Na URL especifica-se o protocolo a ser usado (protocolos que são usados para acessar certo conteúdo em um ser servidor, para a transferencial de arquivos ou acessar o site, por exemplo).

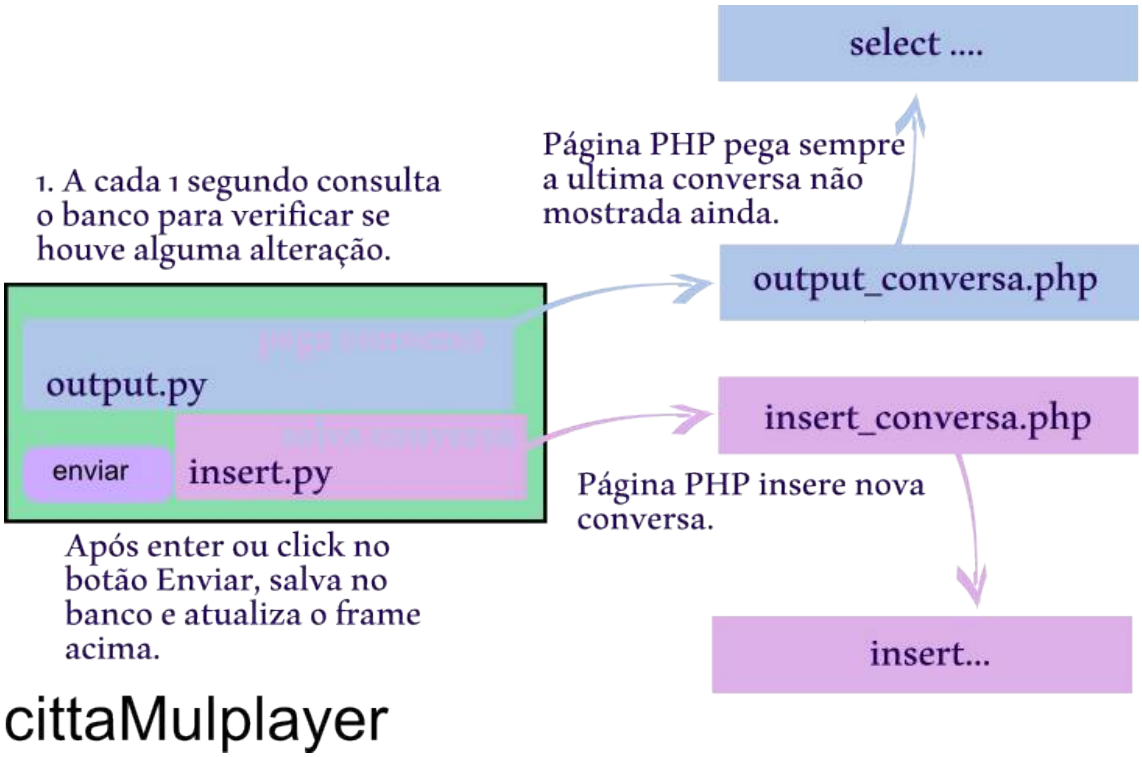
Então para as requisições que precisava ser feito (mandando informação e recebenfo resposta), utilizei o módulo urllib.request, que têm funções e classes que fazem a comunicação por meio de requisições do script Python com uma página web (PHP). Normalmente as informações trocadas entre entre servidor e cliente são em XML.



Fiz o teste no Blender em um arquivo branco mesmo, já que vai demorar até ter a versão certa para o multiplayer. Com o WebService pronto, que é a alma do chat, fiz o código para mapeamento do teclado. Demandou muito esforço de implementação, foi testado e está operacional.

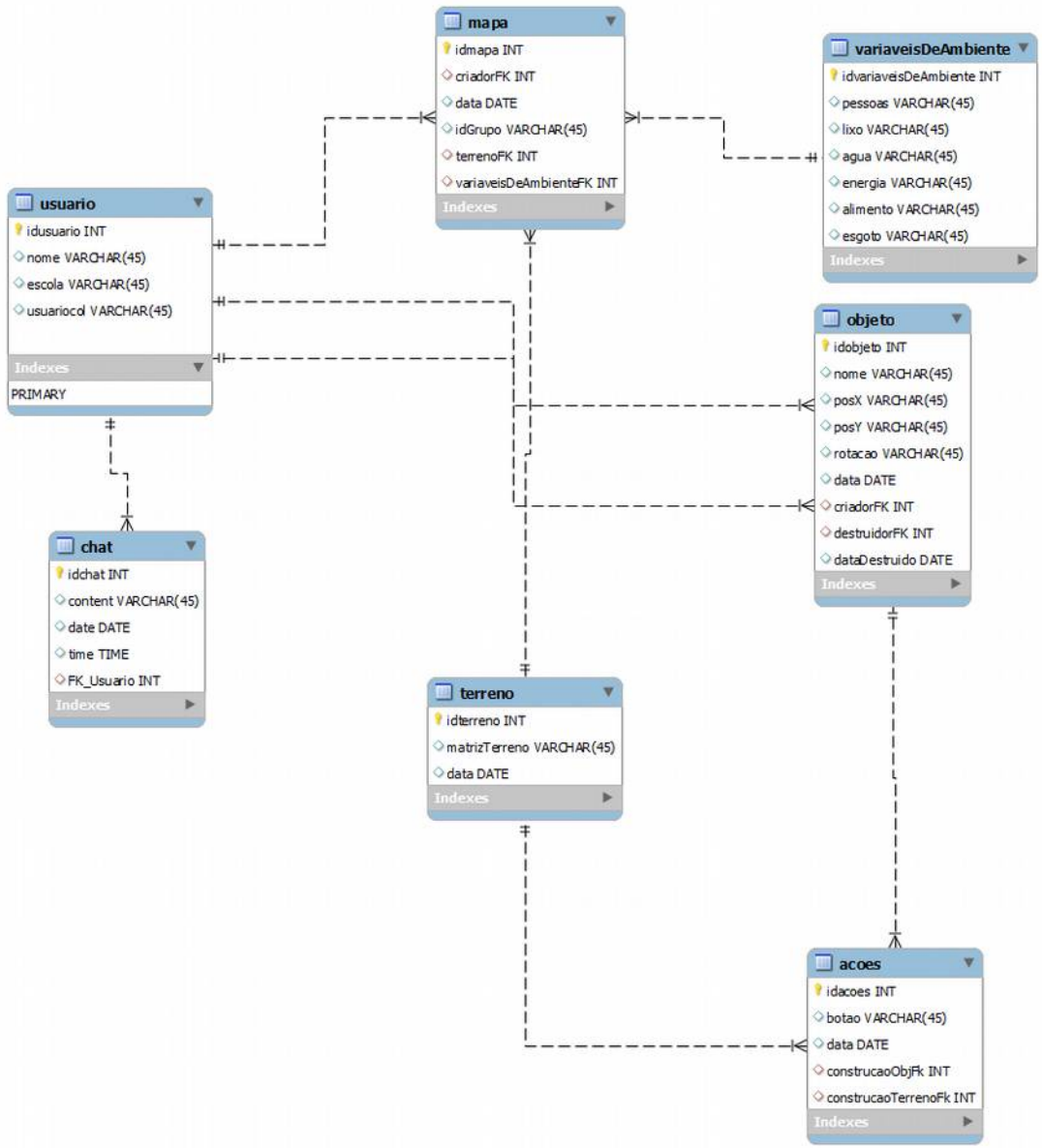


O sistema num todo funciona como na imagem a seguir:



Como a própria imagem mostra, temos dois campos principais, um que fica verificando se existe mensagem nova e outro que é para enviar uma mensagem. Como não temos o sistema Multiplayer, eu utilizei variáveis como exemplo que mais tarde, estas informações serão tiradas do sistema (nome do usuário e nome da sala/mapa compartilhada, por exemplo).

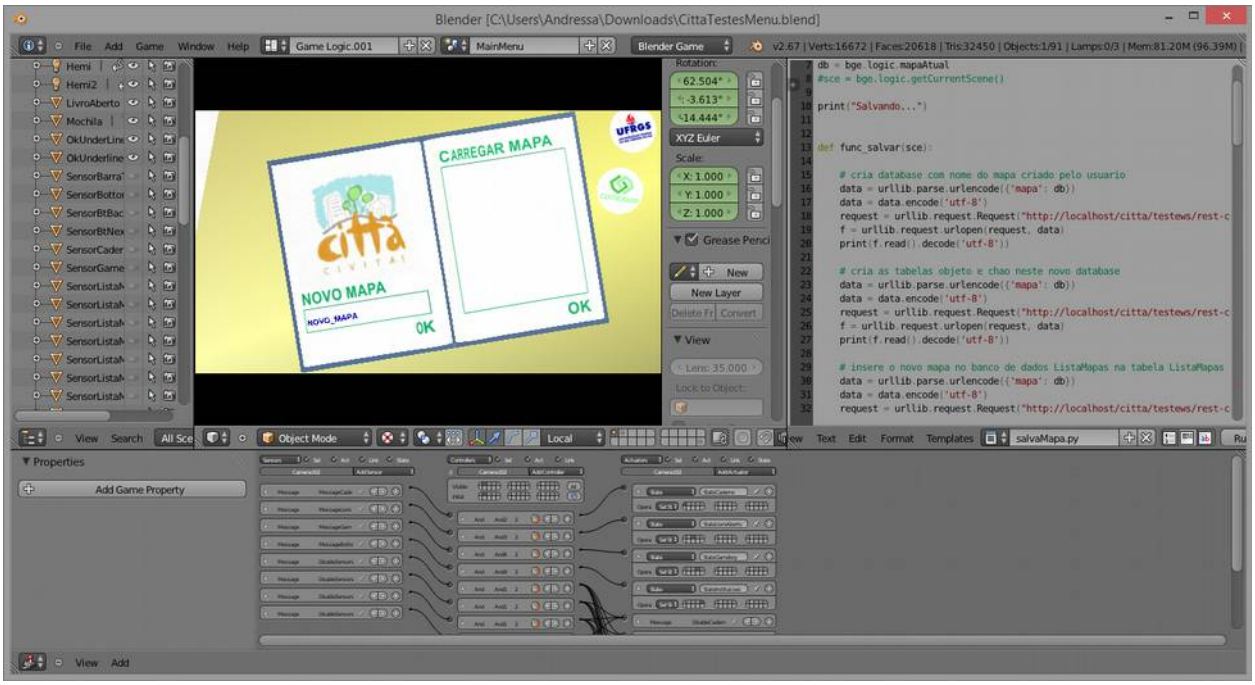
Pensando na estrutura geral de como ficará o game, tando na parte multiplayer quando no chat, e levando em consideração também as variáveis de ambiente que serão usadas depois para análise, fiz o seguinte diagrama:



3.5. Multiplayer

O bolsista Juarez Costa fez pesquisas sobre o modo multiplayer, realizando testes e escrevendo a estrutura WebService. Pegando e retornando estruturas em XML em consultas/salvamentos de mapas em nuvem. A grande dificuldade aqui foi encontrar uma maneira de não sobrecarregar o processamento do computador já que trabalhamos com computadores extremamente limitados nas escolas em que Città é utilizado.

Pensando também nesta limitação de processamento, resolvemos separar o modo multiplayer do modo monoplayer. Assim, a versão para o multiplayer tem o menu diferente do monoplayer, com o carregamento (na hora de escolher um mapa) dos mapas sendo feito através do banco, com o nome dos mapas que já existem em nuvem.



A interface do menu do Multiplayer é igual a do menu Singleplayer, a única diferença é que no SinglePlayer, para carregar a lista acima, é feito uma varredura na pasta local onde é armazenado os dados dos mapas salvos. No Multiplayer, essa consulta de mapas é feita no banco de dados em nuvem, na tabela listaMapas do banco listaMapas. Toda vez que é criado uma mapa novo no jogo, é verificado se já existe um mapa com aquele nome, se existe retorna um erro, avisando que não poderá existir mapas com o mesmo nome. Se não existe, é criado então um novo banco com o nome do mapa, e inserido o nome deste novo mapa na tabela listaMapas. A cada 2 segundos é verificado se o mapa atual está igual ao em nuvem, e se não está ele salva de novo para ficar igual. O primeiro carregamento é feito completo, e o carregamento da verificação de alteração carrega apenas aquilo que está diferente. O método de colisão foi bem complexo de se adaptar, no início ficava um objeto em cima de outro, deixando mais lento o jogo.

Nos testes, vi que ainda ficou meio lento, então o modo multiplayer não pode ser jogado por computadores muito inferiores. Também é obrigatório o uso em rede. Quanto menor a banda, mais delay vai existir entre a consulta e a alteração no mapa/terreno.

Para o salvamento do mapa, criei uma classe em python, chamada func_salva, que recebe como parâmetro a cena que o objeto se encontra. Atualmente o código não é orientado a objeto, porém na minha parte optei por fazer, esta prática além de deixar o código limpo, diminui o número de linhas feitas, já que não precisa duplicar o código em todos os lugares que tenha um salvamento no mapa. Segue abaixo parte do código feito para o salvamento dos objetos, que é responsável pela comunicação cliente-servidor:

```
# insere o novo mapa no banco de dados ListaMapas na tabela ListaMapas
data = urllib.parse.urlencode({'mapa': db})
data = data.encode('utf-8')
request = urllib.request.Request("http://localhost/citta/testews/rest-create-and-insert-listaMapa.php")
f = urllib.request.urlopen(request, data)
print(f.read().decode('utf-8'))
```

Neste caso, utilizando o comando urllib.parse.urlencode passa os dados necessários com uma solicitação POST. Passar parâmetros pelo POST faz com que separe o corpo da mensagem da URL, diferentemente do que aconteceria se usassemos GET, que parra o parâmetro pela URL. O comando Encode serve para transformar o padrão atual em UTF-8, que é o utilizado na web. O comando urllib.urlopen, como o próprio nome diz, abre uma URL, enviando os parâmetros para o endereço do servidor. O que retorna do servidor é tirado do padrão UTF-8 e impresso no prompt que executa o Blender. Este código é o mais importante, sendo repetido várias vezes, para todos os casos de salvamento ou carregamento dos objetos/terreno.

No código abaixo, está a parte responsável pelo carregamento da lista dos mapas existentes no menu inicial.

```
request = urllib.request.Request("http://localhost/citta/testews/rest-lista-mapas.php")
f = urllib.request.urlopen(request)
dom = parseString(f.read().decode('utf-8'))

for node in dom.getElementsByTagName('nome'):
    xmlTag = node.toxml()
    xmlData = xmlTag.replace('<nome>','').replace('</nome>','')
    nomeMapa = xmlData
    print(nomeMapa)
    bge.logic.mapas.append(nomeMapa)
```

A requisição para o servidor é feita, e no parseString transformamos o padrão UTF-8 para String para poder manipular o XML que retornar do servidor. Depois de transformar em String, podemos utilizar o comando getElementsByName, que como o nome diz, pega o nodo pelo nome. Este comando já trata estruturas XML. Percorremos todos os nodos nome, tiramos a estrutura XML daquele nodo e acrescentamos o nome do mapa na lista de nomes dos mapas existentes.

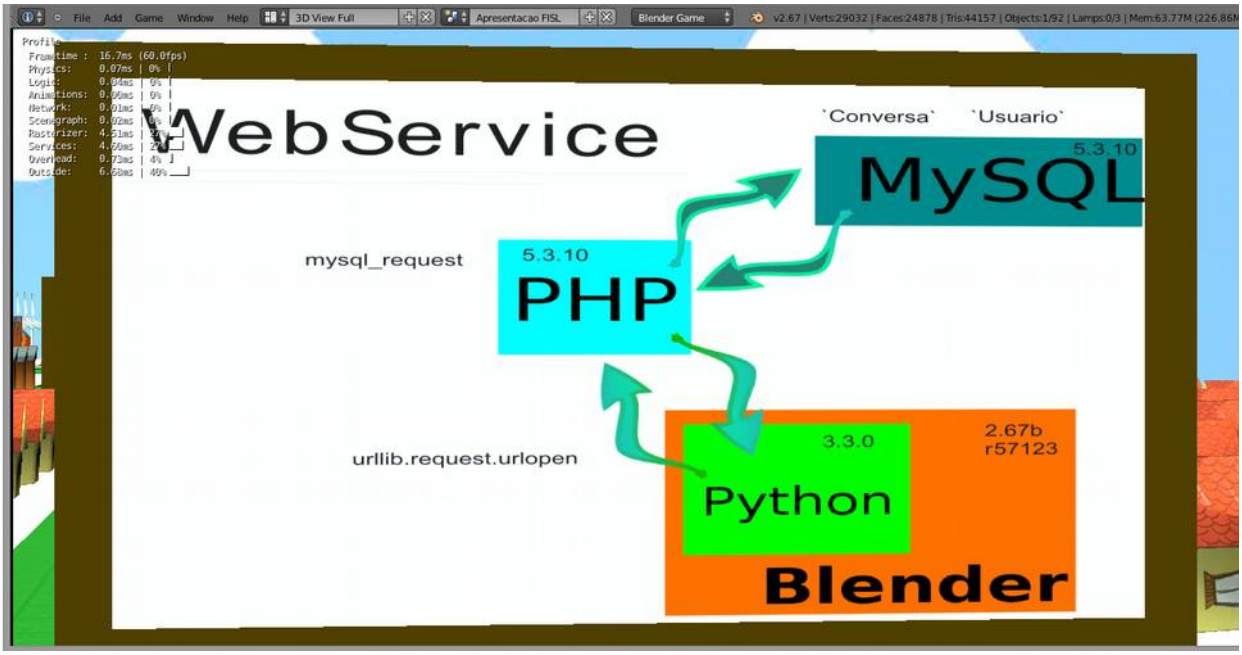
Abaixo o código PHP:

```
<?php
include "mysql_header.php";


$db = mysql_select_db("listamapas");

$query="SELECT * FROM listamapas";
$result=mysql_query($query) or die (mysql_error());

if(!$result){
    $message = "Erro: ". mysql_error() . "\n";
    $message .= "Consulta: " . $query;
    echo ($message);
}else{

    echo "<dados>";
    //Busca os dados do objeto no banco de dados:
    while($info = mysql_fetch_array( $result ))
    {
        echo "<mapa>";
        //echo "<objeto>\n";
        //monta o XML de resposta
        echo "  <nome>".$info['NOMEMAPA']."</nome>";
        echo "</mapa>";

    }
    echo "</dados>";
}
```

4. Palestra e Curso

4.1. Palestra no FISL14 (Fórum Internacional de Software Livre)

A equipe do Projeto Cosmopolita apresentou o Città na 14ª edição do Fórum Internacional do Software Livre (FISL14), em 3 de julho de 2013. Para a apresentação, ao invés de usarmos slides, preparamos uma aplicação interativa em Blender, que simulava uma cidade feita em 3D, substituindo os slides por outdoors dessa cidade. Cada outdoor era realizada uma animação de movimento de câmera. O outdoor da minha parte é mostrado na figura a seguir.



Na minha parte da apresentação, expliquei como funciona o WebService e como implementei o chat. Contei das dificuldades encontradas ao tentar importar algumas bibliotecas Python para dentro do Blender, e fiz uma demonstração de seu funcionamento.

4.2. Oficina de Desenvolvimento de Jogos em Blender – FISL15 (Fórum Internacional de Software Livre)

Em 8 de maio de 2014, participei da 15ª edição do FISL, como palestrante da ***Oficina de Desenvolvimento de Jogos em Blender***, junto com o bolsista Laurent Quevedo. Foram 180 minutos focados no desenvolvimento de um jogo simples no Blender, de forma a ensinar aos participantes o funcionamento básico da BGE.

Para a oficina, foi preparado um projeto-exemplo do que seria feito durante o curso, e uma apostila com o passo a passo de como esse projeto foi feito, para os alunos refazerem os passos em casa.

No início do curso, os alunos interessados receberam um projeto base em Blender, com a estrutura básica do projeto já feita, de forma a focarmos somente nos assuntos que o

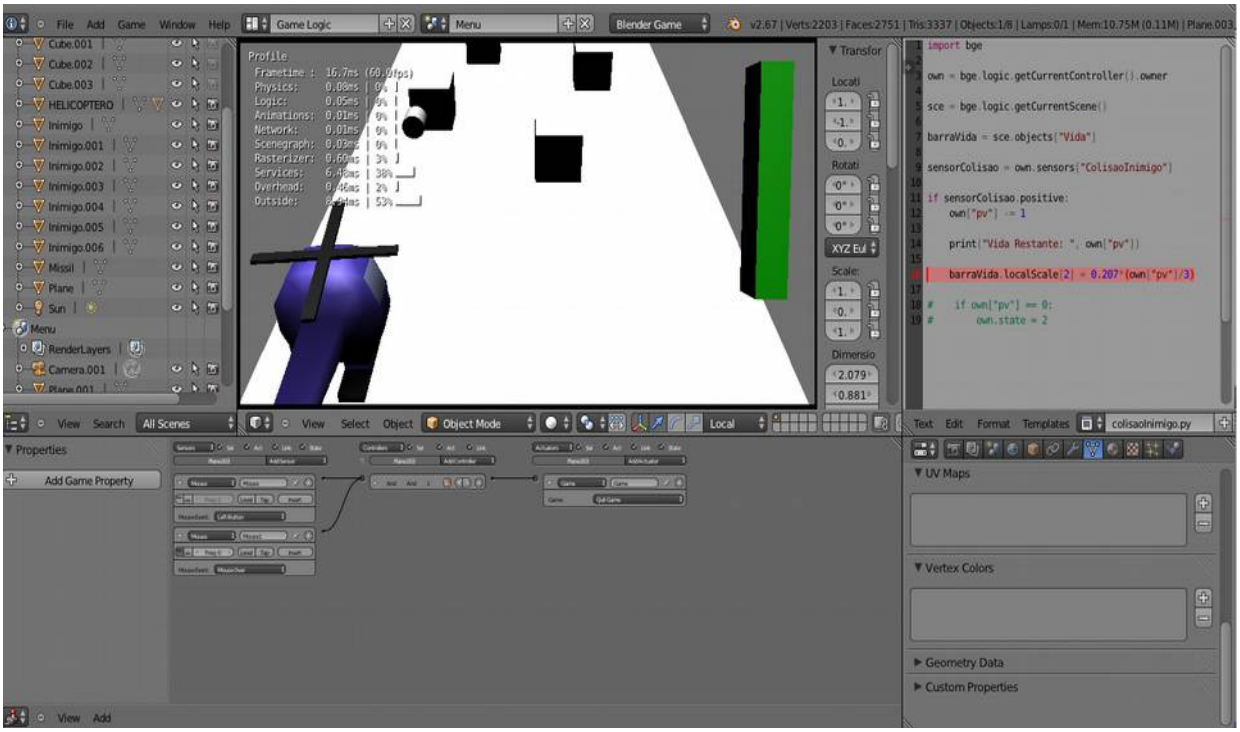
curso deveria cobrir, e um arquivo, também em Blender, com modelos 3D e texturas feitas pelo Luciano Soares, produzidos exclusivamente para o curso.

Nesta oficina auxiliei os alunos que estavam com dificuldade enquanto o Laurent continuava a oficina, tínhamos pouco tempo, então não podíamos parar por muito tempo.

4.3. Oficina de Blender na UNEB

De 28 a 31 de janeiro de 2014, fui a Salvador, juntamente com os bolsistas Laurent Quevedo e Luciano Soares, com a proposta de refazer o curso de desenvolvimento de jogos com Blender, na Universidade do Estado da Bahia.

Preparamos então um cronograma completo do curso com introdução ao Blender e um projeto exemplo do jogo que seria desenvolvido pelos participantes.



A primeira parte do curso se destinava ao ensinamento de algumas técnicas básicas de modelagem de objetos no Blender, e foi passada aos participantes pelo bolsista Luciano. Nessa parte, os participantes aprenderam a modelar o helicóptero que seria utilizado no jogo.

A segunda parte foi ministrada por Laurent, e se destinava a ensinar o funcionamento básico da BGE, os conceitos de Blocos de Lógica, a montagem da cena e o uso dos scripts no jogo.

Eu fiquei responsável pelo suporte em sala de aula, ajudei as pessoas que estavam com dificuldade. Também respondi dúvidas sobre Python e contei minhas experiências para que pudessem contornar os muitos problemas encontrados na limitação das bibliotecas Python.

Foram quatro dias de aulas práticas, em que os participantes acompanhavam, passo a passo o desenvolvimento de um jogo no Blender.

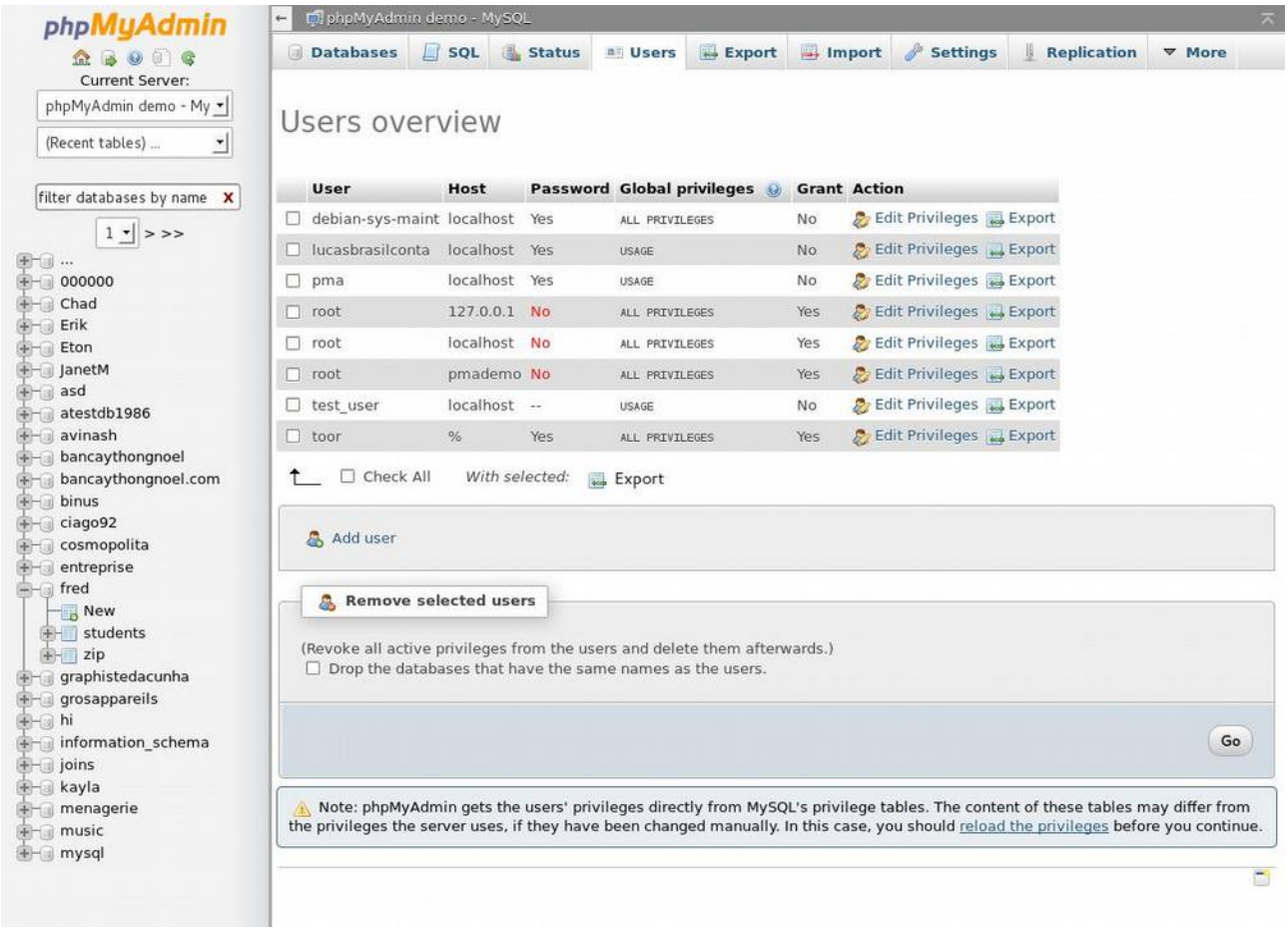
5. Reestruturação

Durante um bom tempo, fiquei estudando o código, a estrutura, o que cada arquivos faz e como poderíamos melhorar o código, deixando-o limpo e legível. Esta reestruturação mudaria o paradigma atual para paradigma orientado a objetos. Seria necessário fechar um pacote 'final' e depois reescrever todo o código. Além de limpar o código, seria feito também para implementar o sistemas de pastas e arquivos separados, para facilitar na hora do compartilhamento e sincronia do código entre os colegas de bolsa. Segue abaixo o diagrama geral feito inicialmente, e a partir deste, iriamos ter a base para fazer a nova arquitetura.

6. Servidor

Fui encarregada também de manter o servidor sempre atualizado e verificar se seus serviços estavam funcionando adequadamente (principalmente o Apache, já que no Lelic existem aplicações vigentes). Comando rotineiros eram executados praticamente todos os dias, para atualizar o sistema (apt-get update e apt-get upgrade), assim como o monitoramento dos processos/serviços em execução com o comando ps -aux.

Também era necessário o supervisionamento do banco de dados MySQL. MySQL é um sistema de gerenciamento de banco de dados (SGBD), que utiliza SQL (Structured Query Language) como interface. Para acessar o MySQL utilizada o PHPMyAdmin, que nada mais que uma interface gráfica feita em PHP para a manipulação dos dados do banco de dados. Porém é possível acessar pelo terminal de comando também. A imagem abaixo foi tirada do Google, apenas para mostrar a interface:



Bibliografia

WebRTC. Disponível em http://www.webrtc.org/

Burster

http://wiki.blender.org/index.php/Dev:Source/GameEngine/WebPlugin

http://www.blender.com.br/index.php?option=com_content&view=article&id=166:burster&catid=41&Itemid=147

http://www.blendernation.com/2010/11/02/burster-blender-web-player/

Raycast

http://www.tutorialsforblender3d.com/BGE_Python/Game_Logic/GameObject/GameObject_rayCast.html

PyAudio

http://people.csail.mit.edu/hubert/pyaudio/

http://wiki.blender.org/index.php/User:Hcube/SummerOfCode2007/Proposal

WebService

http://imasters.com.br/artigo/4245/web-services/entendendo-os-webservices/

CAPELARI, João Cláudio Junior; FORNARI, Miguel Rodrigues. WebClipping utilizando WebServices.

Blender Documentation

http://www.blender.org/documentation/blender_python_api_2_56_0/

RELATÓRIO DO PROJETO CITTÀ COSMOPOLITA

Projeto processo CNPq: *551846/2011-9*

Relatório de Atividades do Bolsista: *Juarez Dubois da Costa*

Bolsista processo CNPq: *382602/2012-9*

Coordenador: *Daniel Nehme Müller*

Período do Relatório: junho/*2012 a agosto/2013*

Plano de Trabalho:

Configuração de servidores do projeto. Administração da rede do laboratório (LELIC). Administração dos portais e demais serviços internet envolvidos no projeto. Reuniões com as equipes do projeto. Análise dos projetos Java e Python envolvidos na implementação do simulador Città Cosmopolita.

1. Introdução

Neste documento estão descritas as atividades que desenvolvi como bolsista do CNPq durante a participação do projeto Cosmopolita. Durante este período trabalhei no *Laboratório de Estudos em Linguagem, Interação e Cognição* (*LELIC*), participando na manutenção e gerenciamento dos recursos de infraestrutura de tecnologia da informação (TI) do laboratório e também no projeto e desenvolvimento do simulador de cidades (Cittá), parte do projeto Civitas. O projeto Civitas é realizado por bolsistas e pesquisadores do LELIC, localizado na Universidade Federal do Rio Grande do Sul, sob orientação da professora Margarete Axt.

1.1 Visão do projeto

Durante o período no qual estive no *Lelic* realizei a manutenção e gerenciamento dos equipamentos e recursos de infraestrutura de TI, como computadores, redes, servidores de dados, sistemas e recursos de internet com a finalidade de proporcionar os recursos para que os pesquisadores e desenvolvedores do *Lelic* pudessem utilizar e se apoiar para as suas atividades.

Participei conjuntamente com os outros bolsistas no desenvolvimento de um software gráfico interativo (jogo) com a finalidade de apoiar pedagogicamente a construção do conhecimento das crianças de 3ª e 4ª séries do ensino fundamental em relação à construção de cidades virtuais e os aspectos relativos a pensar, organizar e construir uma cidade virtual, com o auxílio de seus professores, para poderem entender os sistemas que regem as relações de

uma cidade como recursos alimentícios de produção de alimentos como fazendas, efeitos de poluição como fábricas, recursos de infraestrutura das cidades como estações de tratamento de água e esgoto, etc.

2. Desenvolvimento

Quando iniciei no projeto de desenvolvimento, o jogo *Cittá* estava sendo migrado da versão escrita na linguagem de programação *Java* para a versão feita no software de criação 3D chamado *Blender*.

O software desenvolvido em *Java* possuía diversas limitações em termos de implementação de alguns recursos que dificultavam enormemente o desenvolvimento do software, como a rotação de um objeto no cenário, por exemplo.

Assim o software começou a ser reescrito na arquitetura do *Blender* para facilitar o desenvolvimento do jogo pelo fato do *Blender* já possuir recursos voltados a jogos como a funcionalidade *Blender Game Engine* já incorporado ao *Blender*.

Participei da parte de desenvolvimento do jogo com relação ao quesito de mapeamento objeto-relacional e salvamento, chamado de persistência, dos objetos do jogo em um sistema de banco de dados.

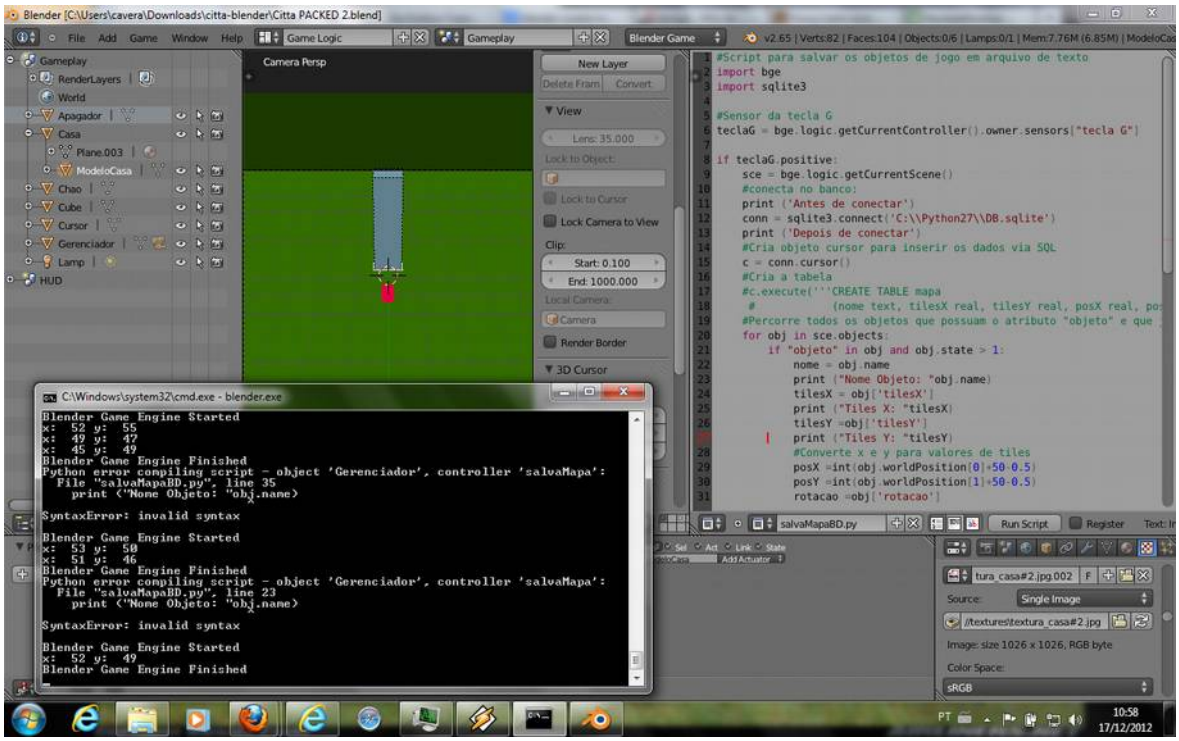


Imagem do programa Blender

O mapeamento objeto-relacional consiste em fazer com que os objetos 3D inseridos pelos usuários do jogo fiquem armazenados em um sistema gerenciador de banco de dados, sendo este armazenamento o aspecto de "*persistência*" do jogo, para que na próxima execução do jogo os objetos possam ser observados da mesma maneira que foram deixados na execução anterior do jogo.

Exemplo de código na linguagem de programação "*python*" mostrando um exemplo de persistência dos objetos no jogo *Cittá* na versão em *Blender*:

```
#Script para salvar os objetos de jogo em arquivo de texto
import bge
import sqlite3
#Sensor da tecla G
teclaG = bge.logic.getCurrentController().owner.sensors["tecla G"]
if teclaG.positive:
    sce = bge.logic.getCurrentScene()
    #conecta no banco:
    print ('Antes de conectar')
    conn = sqlite3.connect('C:\\Python27\\DB.sqlite')
    print ('Depois de conectar')
    #Cria objeto cursor para inserir os dados via SQL
    c = conn.cursor()
    #Cria a tabela
    #c.execute('''CREATE TABLE mapa
      #          (nome text, tilesX real, tilesY real, posX real, posY real, rotacao real, pessoas real, agua real,
energia real, alimento real, lixo real, esgoto real)''')
    #Percorre todos os objetos que possuam o atributo "objeto" e que ja estejam postos no mapa (state > 1)
    for obj in sce.objects:
        if "objeto" in obj and obj.state > 1:
            nome = obj.name
            print ("Nome Objeto: "obj.name)
            tilesX = obj['tilesX']
            print ("Tiles X: "tilesX)
            tilesY =obj['tilesY']
            print ("Tiles Y: "tilesY)
            #Converte x e y para valores de tiles
            posX =int(obj.worldPosition[0]+50-0.5)
            posY =int(obj.worldPosition[1]+50-0.5)
            rotacao =obj['rotacao']
            pessoas = 4
            agua = 4
            energia = 4
```

```
    alimento = 4
    lixo = 4
    esgoto = 3
    objeto = (nome, tilesX, tilesY, posX, posY, rotacao, pessoas, agua, energia, alimento, lixo, esgoto)
    cursor.execute("INSERT INTO mapa VALUES (?,?,?,?,?,?,?,?,?,?,?,?)", objeto)
#SALVAR POS DA CAMERA ANTES DE COMMIT na tabela Cenario
#commit e fecha a conexao
conn.commit()
cursor.close()
conn.close()
print("Objetos inseridos no banco de dados")
```

A persistência dos objetos em um sistema gerenciador de banco de dados foi realizada no banco de dados *SQLite*.

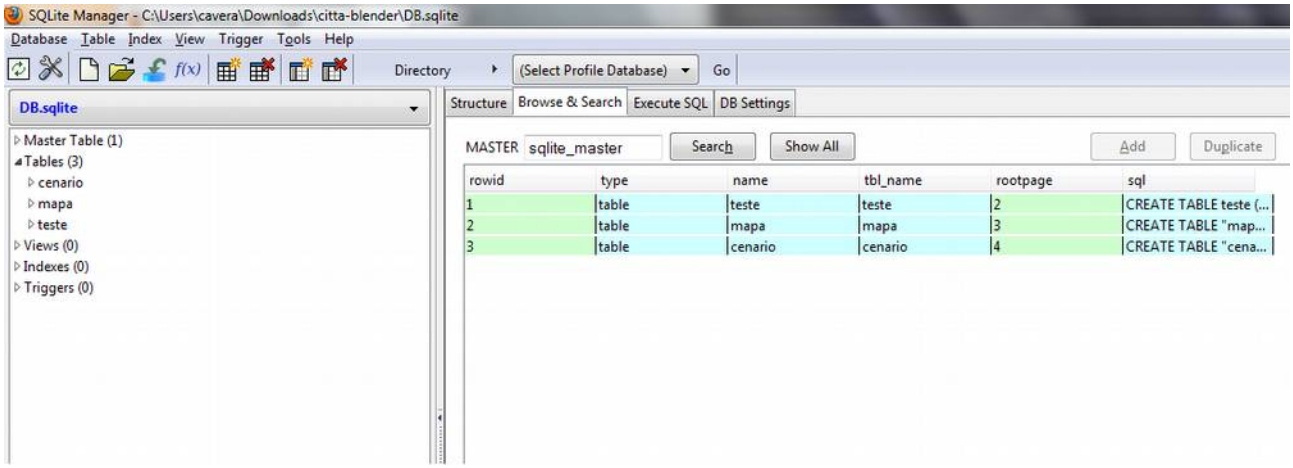


Imagem do SQLite Manager mostrando um exemplo de banco de dados

Ajudei no projeto e criação das tabelas e atributos do banco de dados, provendo uma estrutura de armazenamento que possa ter os objetos do jogo facilmente salvos e adquiridos para a posterior realocação em uma nova execução do jogo exibindo os objetos tais quais foram organizados em uma execução prévia.

2.1. Multiplayer

Consegui alguns videos de tutoriais no youtube mas alguns são em alemão que vou colocar ao lado como [Ger] e em inglês que vou colocar ao lado do link um símbolo [En]:

Blender Game Engine Tutorial Multiplayer [Ger]:

http://www.youtube.com/watch?v=mzKrKwhpmJQ

Blender Multiplayer Online Component [1.7] [En]:

http://www.youtube.com/watch?v=ctNbvAj_nzY

Para instalar o plugin o mesmo deve ser baixado em:

http://blenderartists.org/forum/showthread.php?242740-Blender-Game-Engine-Multiplayer-Add-on&highlight=multiplayer+add-on

Eu baixei quando estava na versão 1.6.4.

Para instalá-lo basta abrir o Blender e ir no menu "File" e a seguir em "User Preferences". Na janela que abre clique sobre a guia/botão "Addons".

A seguir clique no botão "Install from file" na parte inferior esquerda da janela. Na janela que abre procure pelo arquivo que você baixou, no meu caso é "addon_1.6.4.zip", selecione o arquivo e clique sobre o botão "install from file...".

A seguir será retornado para a janela de "User preferences". Para habilitar o plugin você deve ir até a guia de "addons" e clicar na Categoria "Game engine" e a seguir será exibida na direita da janela os addons instalados para a Categoria Game engine.

Clique no triângulo do "Game Engine: Multiplayer" para abrir o quadro. A seguir habilite a opção na direita do quadro ao lado do ícone do boneco correndo e o plugin estará habilitado.

23/01/13:

Pelo que eu andei pesquisando sobre o add-on existe a primeira camada que é a representação do usuário no mundo dele e a segunda camada que é a representação do usuário no universo multiplayer.

Ao pesquisar sobre o add-on no youtube só achei exemplos de um quadrado no universo singleplayer que no multiplayer é representado como uma esfera. No mais só achei um outro que faz um multiplayer com helicópteros. Além destes achei um exemplo de tanques (armature test) que é bem parecido com o exemplo do cubo e da esfera.

A minha ideia é fazer o add-on representar graficamente em tempo real as peças de ambos os usuários e ambos utilizarem a mesma base de dados remota para salvar o mapa feito em conjunto.

Mas para isso ainda necessito estudar mais com relação à representação em comum do cenário.

20/03/2013:

Segue a página de documentação do add-on do multiplayer para a Blender Game Engine:

http://code.google.com/p/bge-networking/

02/04/13

Realizado teste de utilização da mesma base de dados para mais de um cliente em uma base de dados SQLite local. Funciona direitinho, mas tem que teclar 'G' para gravar e o outro jogador tem que teclar 'C' para carregar as alterações. Então vou ter que verificar um método de mudança de estados para automatizar a sincronicidade entre os jogadores.

Sensors:

http://wiki.blender.org/index.php/Doc:PT/2.4/Manual/Game_Engine/Logic/Sensors

05/04/13

Estive pesquisando e a lib de conexão com MySQL é desenvolvido para o Python 2.7 enquanto ainda está para ser desenvolvida a lib para a versão 3.3 do python que é a que o Blender 2.65 utiliza.

Então vou verificar outra lib de conexão para utilizar com PostgreSQL.

10/04/13

Achei um conector para MySQL feito pela Oracle (http://dev.mysql.com/downloads/connector/python/), a instalação foi tão rápida que eu desconfiei que não tinha instalado. Mas ele compilou com o Python 3.3 que eu tinha instalado sem o Blender. Foi instalado dentro do diretório do Python "python/lib/site-packages/mysql" e para verificar se funcionaria no Blender copiei o diretório mysql para o diretório "Blender/2.65/python/lib/site-packages". Mas como não funcionou com o código alterado para utilizar as libs do MySQL Connector copiei a pasta mysql para a pasta lib ("Blender/2.65/python/lib") e aí funcionou.

O problema está residindo agora no fato de estar gravando apenas null no banco de dados. Recriei a tabela com campos "not null" e a tabela foi preenchida com zeros. Estes casos aconteceram tanto no banco de dados do servidor local como no banco de dados do servidor.

Ainda estou verificando possíveis soluções.

11/04/13

Inserção manual via python (do Blender):

```
C:\Program Files\Blender Foundation\Blender\2.65\python\lib>python
Python 3.3.1 (v3.3.1:d9893d13c628, Apr 6 2013, 20:30:21) [MSC v.1600 64 bit (AM]
D64) on win32
Type "help", "copyright", "credits" or "license" for more information.
    import mysql.connector
    Traceback (most recent call last):
    File "<stdin>", line 1, in <module>
    File ".\mysql\connector__init__.py", line 33, in <module>
    from .connection import MySQLConnection
    File ".\mysql\connector\connection.py", line 32, in <module>
    from .network import MySQLUnixSocket, MySQLTCPSocket
    File ".\mysql\connector\network.py", line 28, in <module>
    import socket
    File ".\socket.py", line 47, in <module>
    import _socket
    ImportError: DLL load failed: Uma rotina de inicialização da biblioteca de víncu
    lo dinâmico (DLL) falhou.
    quit()
```

Ao rodar direto da pasta C:\Python33

```
C:\Python33>python
Python 3.3.1 (v3.3.1:d9893d13c628, Apr 6 2013, 20:30:21) [MSC v.1600 64 bit (AM]
D64) on win32
Type "help", "copyright", "credits" or "license" for more information.
    import mysql.connect
    Traceback (most recent call last):
```

```
File "<stdin>", line 1, in <module>
ImportError: No module named 'mysql.connect'
import mysql.connector
conn = mysql.connector.connect(user='citta', password='teste', host='127.0.0
.1', database='cittateste')
c = conn.cursor()
c.execute("INSERT INTO mapa VALUES ('teste',43.5,39.5,0.0,1,2,3,4,5,6)")
c.commit()
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
AttributeError: 'MySQLCursor' object has no attribute 'commit'
conn.commit()
c.close()
True
conn.close()
```

Sem erros.

E gravou no banco sem ser zerado!

Vou ver novas tentativas de instalação da lib no Blender.

15/04/13

Instalei o Blender novamente em "C:\Blender", instalei o python em "C:\Blender\2.65\python" sobreescrevendo o python do Blender, instalei a lib do mysql e caí no mesmo problema do dia 10/04/13 acima. Roda o Cittá sem erros mas grava tudo zerado.

Vou verificar se existe outro conector para MySQL em Python.

Enquanto isso segue um fórum sobre código para o Server Lobby:

http://blenderartists.org/forum/showthread.php?242160-Multiplayer-Server-Lobbying

URLs importantes:

https://github.com/farcepest/moist

https://github.com/farcepest/MySQLdb1/tree/MySQLdb-1.3

Sensores (para mudança de estados):

http://wiki.blender.org/index.php/Doc:PT/2.4/Manual/Game_Engine/Logic/Sensors

2.1.1. Testes com webserver

Após termos tido problemas com a camada de dados do Blender, que não estava repassando os dados para o banco de dados, resolvemos tentar a abordagem de persistência através de WebServices.

Após pesquisar diversos tipos de webservices em PHP, como RPC, SOAP e RESTful, fiquei com o padrão REST webservice pois não precisa de bibliotecas externas, porém necessita que o desenvolvedor faça todo o código de retorno em XML manualmente (se você entende de WebServices você sabe do que estou falando).

Então fiz dois arquivos em PHP para testar se o webservice grava no banco:

Um cliente simples:

rest-insert-citta-client.php

```
<?php
$rs = "http://localhost/testews/rest-insert-citta.php";
$qs = "";
$param = array('tipo' => 'casa', 'posX' => "44.5", 'posY' => "54.5", 'rotacao' => "0.0", 'pessoas' => "4", 'agua' => "4", 'energia' => "4", 'alimento' => "4", 'lixo' => "4", 'esgoto' => "4");
foreach($param as $key=>$value){
$qs = $qs."$key=".urlencode($value)."&";
}
$url="$rs?$qs";
$xml = file_get_contents($url);
echo $xml;
```

```
?>
```

E um arquivo 'servidor':

rest-insert-citta.php

```
<?php
include "mysql_header.php";


$tipoObjeto=$_GET["tipo"];
$posX=$_GET["posX"];
$posY=$_GET["posY"];
$rotacao=$_GET["rotacao"];
$pessoas=$_GET["pessoas"];
$agua=$_GET["agua"];
$energia=$_GET["energia"];
$alimento=$_GET["alimento"];
$lixo=$_GET["lixo"];
$esgoto=$_GET["esgoto"];
switch ($tipoObjeto) {
    case "casa":
                                                $query="INSERT INTO mapa (nome, posX,posY,rotacao,pessoas,agua,energia,alimento,lixo,esgoto) VALUES ('".$tipoObjeto."',".$posX.",".$posY.",".$rotacao.",".$pessoas.",".$agua.",".$energia.",".$alimento.",".$lixo.",".$esgoto.")";
        $result=mysql_query($query);
        if(!$result){
            $message = "erro: ". mysql_error() . "\n";
            $message .= "Consulta: " . $query;
            echo ($message);
        }else{
            //monta o XML de resposta
            echo "<?xml version=\"1.0\"?>";
```

```
            echo "<taxinfo>";
            echo "<result>inseriu</result>";
            echo "</taxinfo>";
        }
      break;
        case 1:
        echo "i equals 1";
        break;
        case 2:
        echo "i equals 2";
        break;
        default:
        //monta o XML de resposta
        echo "<?xml version=\"1.0\"?>";
        echo "<taxinfo>";
        echo "<result>nenhum tipo de objeto detectado!</result>";
        echo "</taxinfo>";
        }
    ?>
```

E inseriu com sucesso no banco de dados.

Agora falta ver como faço para chamar um webservice a partir do python do Blender e como ele se comporta.

Criado um script em Python para a inserção dos dados da BGE no Banco de Dados:

Script para salvar os objetos de jogo em um WebService:

```
import bge
import urllib.request
Sensor da tecla G
```

```
teclaG = bge.logic.getCurrentController().owner.sensors["tecla G"]
if teclaG.positive:


    sce = bge.logic.getCurrentScene()
    #conecta no banco:
    print ('Antes de conectar')
    #mapa = "mapas\" + bge.logic.mapaAtual + ".sqlite"
    #print ('Depois de conectar')
      #Percorre todos os objetos que possuam o atributo "objeto" e que ja estejam postos no mapa (state > 1)
    for obj in sce.objects:
        print ("entrou na iteracao")
        if "objeto" in obj and obj.state > 1:
            print ("entrou na condicao")
            nome = obj.name
            #Converte x e y para valores de tiles
            #posX =int(obj.worldPosition[0]+50-0.5)
            #posY =int(obj.worldPosition[1]+50-0.5)
            posX = obj.worldPosition[0] + 50 + (0.5-((obj['tilesX']%2)/2))
            posY = obj.worldPosition[1] + 50 - (0.5-((obj['tilesY']%2)/2))
            print("PosX: ", posX)
            print("PosY: ", posY)
            rotacao =obj['rotacao']
            pessoas = 4
            agua = 4
            energia = 4
            alimento = 4
            lixo = 4
```

```
esgoto = 3

objeto = (nome, posX, posY, rotacao, pessoas, agua, energia, alimento, lixo, esgoto)

urlcompleta="http://143.54.192.213/testews/rest-insert-citta.php?tipo="+nome+"&posX="+str(posX)+"&posY="+str(posY)+"&rotacao="+str(rotacao)+"&pessoas=4&agua=4&energia=4&alimento=4&lixo=4&esgoto=2"

resposta = urlopen(urlcompleta)

print (resposta.headers.as_string())

#SALVAR POS DA CAMERA ANTES DE COMMIT na tabela Cenario

#commit e fecha a conexao

print("Objetos inseridos no banco de dados")
```

Além deste foi criado um WebService de consulta no Banco de Dados:

No Servidor temos o seguinte código no arquivo buscaDadosTabela.php:

```
<?php
include "mysql_header.php";
$tipoDado=$_GET["tipo"];
$query="SELECT * FROM mapa";
$result=mysql_query($query) or die (mysql_error());
if(!$result){
    $message = "erro: ". mysql_error() . "\n";
    $message .= "Consulta: " . $query;
    echo ($message);
  }else{
  if($tipoDado=="xml"){
        $identif=0;
        echo "<?xml version=\"1.0\"?>\n";
        echo "<Dados>\n";
        //Busca os dados do objeto no banco de dados:
```

```
while($info = mysql_fetch_array( $result ))
{
        echo "<objeto id='".$identif."'>\n";
        //echo "<objeto>\n";
        //monta o XML de resposta
        echo "  <nome>".$info['nome']."</nome>\n";
        echo "  <posX>".$info['posX']."</posX>\n";
        echo "  <posY>".$info['posY']."</posY>\n";
        echo "  <rotacao>".$info['rotacao']."</rotacao>\n";
        echo "  <pessoas>".$info['pessoas']."</pessoas>\n";
        echo "  <agua>".$info['agua']."</agua>\n";
        echo "  <energia>".$info['energia']."</energia>\n";
        echo "  <alimento>".$info['alimento']."</alimento>\n";
        echo "  <lixo>".$info['lixo']."</lixo>\n";
        echo "  <esgoto>".$info['esgoto']."</esgoto>\n";
        echo "</objeto>\n";
        $identif++;
    }
    echo "</Dados>";
}else{
    print "<table border cellpadding=3>";
    //Busca os dados do objeto no banco de dados:
    while($info = mysql_fetch_array( $result ))
    {
        //monta a tabela de resposta
        print "<tr>";
        print "<th>Nome:</th> <td>".$info['nome']."</td>";
        print "<th>posX:</th> <td>".$info['posX']."</td>";
        print "<th>posY:</th> <td>".$info['posY']."</td>";
```

```
            print "<th>rotacao:</th> <td>".$info['rotacao']."</td>";
            print "<th>pessoas:</th> <td>".$info['pessoas']."</td>";
            print "<th>agua:</th> <td>".$info['agua']."</td>";
            print "<th>energia:</th> <td>".$info['energia']."</td>";
            print "<th>alimento:</th> <td>".$info['alimento']."</td>";
            print "<th>lixo:</th> <td>".$info['lixo']."</td>";
            print "<th>esgoto:</th> <td>".$info['esgoto']."</td></tr>";
        }
        print "</table>";
    }
}
?>
```

E foi criado um script em Python, para ser utilizado no Blender, que busca os dados da tabela no BD através do WebService criado acima:

Script para carregar os objetos de jogo do WebService:

```
import bge
import pickle
import xml.dom.minidom as minidom
from urllib.request import urlopen
from xml.dom.minidom import parseString
#Sensor da tecla C
teclaC = bge.logic.getCurrentController().owner.sensors["tecla C"]
if teclaC.positive:
    sce = bge.logic.getCurrentScene()
    arquivo = urlopen('http://143.54.192.213/testews/buscaDadosTabela.php?tipo=xml')
    data = arquivo.read()
        dom = parseString(data)
```

```
doc = minidom.parse(arquivo)
for node in doc.getElementsByTagName("objeto")
    xmlTag = node.getElementsByTagName('nome')[0].toxml()
    xmlData = xmlTag.replace('<nome>','').replace('</nome>','')
    nomeObj = xmlData
    objeto = sce.addObject(nomeObj, sce.objects['Cursor'])
    xmlTag = node.getElementsByTagName('posX')[0].toxml()
    xmlData = xmlTag.replace('<posX>','').replace('</posX>','')
    posX = xmlData
    xmlTag = node.getElementsByTagName('posY')[0].toxml()
    xmlData = xmlTag.replace('<posY>','').replace('</posY>','')
    posY = xmlData
    #Arruma a rotacao do objeto em X e a escala
      objeto.applyRotation([1.570796326794897,0.0,0.0], True)
      objeto.scaling = [0.5,0.5,0.5]
      #Passa a posicao do objeto no mundo
      objeto.worldPosition[0] = posX - 50
      objeto.worldPosition[1] = posY - 50
      objeto.localPosition[2] += 0.001
    xmlTag = node.getElementsByTagName('rotacao')[0].toxml()
    xmlData = xmlTag.replace('<rotacao>','').replace('</rotacao>','')
    rotacao = xmlData
    #Passa a rotacao e ja aplica no objeto
      objeto['rotacao'] = rotacao
      objeto.applyRotation([0.0,1.570796326794897*(objeto['rotacao']/90),0.0], True)
      #Corrige o valor de tiles em X e Y conforme a rotacao
      if objeto['rotacao'] == 90 or objeto['rotacao'] == 270:
          valor = objeto['tilesX']
          objeto['tilesX'] = objeto['tilesY']
```

```
        objeto['tilesY'] = valor
        #Corrige a posicao do objeto conforme a rotacao
        objeto.worldPosition[0] -= (0.5-((objeto['tilesX']%2)/2))
        objeto.worldPosition[1] += (0.5-((objeto['tilesY']%2)/2))
    #Passa os demais atributos para o objeto
xmlTag = node.getElementsByTagName('pessoas')[0].toxml()
xmlData = xmlTag.replace('<pessoas>','').replace('</pessoas>','')
pessoas = xmlData
objeto['pessoas'] = pessoas
xmlTag = node.getElementsByTagName('agua')[0].toxml()
xmlData = xmlTag.replace('<agua>','').replace('</agua>','')
agua = xmlData
objeto['agua'] = agua
xmlTag = node.getElementsByTagName('energia')[0].toxml()
xmlData = xmlTag.replace('<energia>','').replace('</energia>','')
energia = xmlData
objeto['energia'] = energia
xmlTag = node.getElementsByTagName('alimento')[0].toxml()
xmlData = xmlTag.replace('<alimento>','').replace('</alimento>','')
alimento = xmlData
objeto['alimento'] = alimento
xmlTag = node.getElementsByTagName('lixo')[0].toxml()
xmlData = xmlTag.replace('<lixo>','').replace('</lixo>','')
lixo = xmlData
objeto['lixo'] = lixo
xmlTag = node.getElementsByTagName('esgoto')[0].toxml()
xmlData = xmlTag.replace('<esgoto>','').replace('</esgoto>','')
esgoto = xmlData
objeto['esgoto'] = esgoto
```

```
        objeto.state = 2

    print (nome, posX,posY,rotacao,pessoas,agua,energia,alimento,lixo,esgoto)
```

E isto ainda necessita ser testado, o que será feito no próximo experimento.

30/04/13

Efetuei os testes com o código de carregamento dos objetos em banco de dados através de web services mostrado acima e necessitou se fazer algumas mudanças. O código que está funcionando, devidamente identado dentro do Blender, segue abaixo:

*carregaMapaWS.py

```
#Script para carregar os objetos de jogo do WebService
import bge
import pickle
import xml.dom.minidom as minidom
from urllib.request import urlopen
from xml.dom.minidom import parseString
#Sensor da tecla C
teclaC = bge.logic.getCurrentController().owner.sensors["tecla C"]
if teclaC.positive:
    sce = bge.logic.getCurrentScene()
    arquivo = urlopen('http://143.54.192.213/testews/buscaDadosTabela.php?tipo=xml')
    data = arquivo.read()
    dom = parseString(data)
    #doc = minidom.parse(arquivo)
    for node in dom.getElementsByTagName('objeto'):
        xmlTag = node.getElementsByTagName('nome')[0].toxml()
        xmlData = xmlTag.replace('<nome>','').replace('</nome>','')
        nomeObj = xmlData
        objeto = sce.addObject(nomeObj, sce.objects['Cursor'])
```

```
  xmlTag = node.getElementsByTagName('posX')[0].toxml()
  xmlData = xmlTag.replace('<posX>','').replace('</posX>','')
  posX = xmlData
  xmlTag = node.getElementsByTagName('posY')[0].toxml()
  xmlData = xmlTag.replace('<posY>','').replace('</posY>','')
  posY = xmlData
#Arruma a rotacao do objeto em X e a escala
objeto.applyRotation([1.570796326794897,0.0,0.0], True)
objeto.scaling = [0.5,0.5,0.5]
#Passa a posicao do objeto no mundo
objeto.worldPosition[0] = float(posX) - 50
objeto.worldPosition[1] = float(posY) - 50
objeto.localPosition[2] += 0.001
xmlTag = node.getElementsByTagName('rotacao')[0].toxml()
xmlData = xmlTag.replace('<rotacao>','').replace('</rotacao>','')
rotacao = xmlData
#Passa a rotacao e ja aplica no objeto
objeto['rotacao'] = rotacao
objeto.applyRotation([0.0,1.570796326794897*(objeto['rotacao']/90),0.0], True)
#Corrige o valor de tiles em X e Y conforme a rotacao
if objeto['rotacao'] == 90 or objeto['rotacao'] == 270:
        valor = objeto['tilesX']
        objeto['tilesX'] = objeto['tilesY']
        objeto['tilesY'] = valor
        #Corrige a posicao do objeto conforme a rotacao
        objeto.worldPosition[0] -= (0.5-((objeto['tilesX']%2)/2))
        objeto.worldPosition[1] += (0.5-((objeto['tilesY']%2)/2))
        #Passa os demais atributos para o objeto
    xmlTag = node.getElementsByTagName('pessoas')[0].toxml()
```

```
xmlData = xmlTag.replace('<pessoas>','').replace('</pessoas>','')

pessoas = xmlData

objeto['pessoas'] = pessoas

xmlTag = node.getElementsByTagName('agua')[0].toxml()

xmlData = xmlTag.replace('<agua>','').replace('</agua>','')

agua = xmlData

objeto['agua'] = agua

xmlTag = node.getElementsByTagName('energia')[0].toxml()

xmlData = xmlTag.replace('<energia>','').replace('</energia>','')

energia = xmlData

objeto['energia'] = energia

xmlTag = node.getElementsByTagName('alimento')[0].toxml()

xmlData = xmlTag.replace('<alimento>','').replace('</alimento>','')

alimento = xmlData

objeto['alimento'] = alimento

xmlTag = node.getElementsByTagName('lixo')[0].toxml()

xmlData = xmlTag.replace('<lixo>','').replace('</lixo>','')

lixo = xmlData

objeto['lixo'] = lixo

xmlTag = node.getElementsByTagName('esgoto')[0].toxml()

xmlData = xmlTag.replace('<esgoto>','').replace('</esgoto>','')

esgoto = xmlData

objeto['esgoto'] = esgoto

objeto.state = 2

print (nomeObj, posX,posY,rotacao,pessoas,agua,energia,alimento,lixo,esgoto)
```

A seguir percebi que necessitaríamos de um método que removesse do banco de dados um registro específico. Então criei um webservice que executa um comando SQL de deleção. Segue abaixo o código:

rest-delete-citta.php

```
<?php
include "mysql_header.php";
$tipoObjeto=$_GET["tipo"];
$posX=$_GET["posX"];
$posY=$_GET["posY"];
$rotacao=$_GET["rotacao"];
$pessoas=$_GET["pessoas"];
$agua=$_GET["agua"];
$energia=$_GET["energia"];
$alimento=$_GET["alimento"];
$lixo=$_GET["lixo"];
$esgoto=$_GET["esgoto"];
switch ($tipoObjeto) {
   case "Casa":
                $query="DELETE FROM mapa WHERE nome='".$tipoObjeto."' and posX=".
$posX." and posY=".$posY."and rotacao=".$rotacao." and pessoas=".$pessoas." and agua=".$agua."
and energia=".$energia." and alimento=".$alimento." and lixo=".$lixo." and esgoto=".$esgoto;
            $result=mysql_query($query);
            if(!$result){
                $message = "erro: ". mysql_error() . "\n";
                $message .= "Codigo SQL: " . $query;
                echo ($message);
            }else{
                //monta o XML de resposta
                echo "<?xml version=\"1.0\"?>";
                echo "<info>";
                echo "<result>removeu com sucesso</result>";
                echo "</info>";
            }
        break;
```

```
case 1:
echo "i equals 1";
break;
case 2:
echo "i equals 2";
break;
default:
//monta o XML de resposta
echo "<?xml version=\"1.0\"?>";
echo "<info>";
echo "<result>nenhum tipo de objeto detectado!</result>";
echo "</info>";
}
?>
```

Falta ainda colocar as funcionalidades de inserção, remoção de objetos e colocar um sensor em modo de detecção de mudança de estados do cenário.

*09/05/2013

Criado o Webservice para a criação de tabelas de mapas:

*rest-create-table.php

```
<?php
include "mysql_header.php";
$nomemapa=$_GET["mapa"];
 $consulta = "CREATE TABLE IF NOT EXISTS mapa_".$nomemapa." (nome varchar(40), posX float, posY float, rotacao float, pessoas float, agua float, energia float, alimento float, lixo float, esgoto float)";
$result=mysql_query($consulta);
if(!$result){
    $message = "erro: ". mysql_error() . "\n";
```

```
    $message .= "Consulta: " . $consulta;

    echo ($message);

  }else{

  //monta o XML de resposta

  echo "<?xml version=\"1.0\"?>";

  echo "<info>";

  echo "<result>criada a tabela mapa_".$nomemapa."</result>";

  echo "</info>";

  }

?>
```

Vou começar a trabalhar com o Laurent para criarmos a funcionalidade de seleção de jogo singleplayer e multiplayer e colocar os códigos conforme o tipo de jogo a partir de uma variável booleana chamada "bge.logic.multiplayer".

Quando esta variável for verdadeira, conforme a seleção a partir do menu, será executado um ou outro código de inicialização, carregamento e salvamento de objetos.

Além disso preciso ver como implementar via código um sensor que verifique mudanças no mapa.

2.2. Estudo: SQLite no Blender Game Engine

Referências de implementação do SQLite no BGE:

Implementando um banco de dados SQLite no Blender Game Engine (Parte 1)

http://infomoz.net/51/

Implementando um banco de dados SQLite no Blender Game Engine (Parte 2)

http://infomoz.net/52/

Implementando um banco de dados SQLite no Blender Game Engine (Parte 3)

http://infomoz.net/57/

Vou estudar a versão implementada no Java em termos de tabelas e nas funções de persistência e ver como podemos aplicar no BGE.

Pelo que vi na versão atual do Blender, versão 2.6, já tem suporte nativo ao SQLite3. Mas precisamos testar ainda.

Então vamos fazer um teste de criação de arquivo de banco de dados no init do jogo.

Os tipos de dados do SQLite são NULL, INTEGER, REAL, TEXT e BLOB.

As datas podem ser inseridas de 3 formas:

Armazenar a data como Texto em um campo TEXT no formato YYYY-MM-DD HH:MM:SS.SSS porém eu vejo o inconveniente de termos que executar um parse para pegar cada um dos dados de ano, mês, dia, hora, minuto e segundo.

Outra possibilidade de armazenar é em um campo inteiro, INTEGER, no qual armazenaríamos a data no formato "Unix type" onde contamos cada segundo desde 00:00 de 1 de Janeiro de 1970. O porém é que deveríamos realizar a conversão para um formato legível.

Então temos que verificar quais funções o python do Blender usa que faça estas conversões de dados de datas de forma transparente.

Vou verificar as funções de python que podem fazer estas conversões de datas em tipos de dados do SQLite e vice-versa.

Funções do Python para utilizar datas:

http://docs.python.org/3/library/datetime.html

3. Organização

Os arquivos de desenvolvimento do sistema feito no *Blender* necessitavam ser colocados em um repositório de dados diferente do que estávamos utilizando, utilizávamos o sistema de versionamento conhecido como *subversion* ou ainda *SVN*, e passamos para o repositório do tipo *GIT*. Realizei a troca de sistema de repositório do *SVN* para o *GIT* e ajudei a manter a organização dos dados nestes repositórios e nos sistemas disponibilizados pela plataforma do *Sourceforge.net*.

Ademais a documentação necessitava ser reorganizada em um sistema do tipo '*Wiki*' de enciclopédia colaborativa, o mesmo tipo de sistemas utilizado por enciclopédias colaborativas online como a *Wikipedia* <*http://pt.wikipedia.org/*>, o qual implementamos no sistema de *wiki* do *sourceforge.net*.

4. Redes e servidor

Realizei a instalação do sistema web *Joomla* para onde migrei a página principal do Lelic <http://www.lelic.ufrgs.br> junto ao servidor de páginas de internet do CPD da universidade.

Página inicial do Lelic

Efetuei a assessoria de aquisição do servidor junto ao fornecedor e junto ao setor de aquisições, patrimônio e entrega do equipamento. Incluí um sistema de *wiki* interno para a documentação dos dados dos projetos do Lelic e documentação de infraestrutura da sala.

Organizei a reestruturação de arquivos e de sistemas do servidor interno <http://lab.lelic.ufrgs.br> ao laboratório do Lelic na Universidade Federal do Rio Grande do Sul.

Realizei a organização de alguns sistemas do servidor principal presente no CPD da Universidade, bem como dos bancos de dados utilizados nos dois servidores, tanto interno ao laboratório quanto no servidor do CPD.

Realizei ainda a migração de alguns sistemas que estavam no servidor do CPD para o servidor do laboratório visto que as versões de alguns sistemas que estávamos utilizando eram muito antigas e viriam a não serem mais suportadas pelo servidor do CPD visto que o mesmo estava migrando as versões de sistemas de provedor de páginas e banco de dados, fazendo com que as páginas dos sistemas mais antigos ficassem sem poder funcionar corretamente.

Página inicial do servidor do laboratório

5. Infraestrutura

Organizei a infraestrutura da sala do laboratório do Lelic. Ajudei na assessoria de aquisição de equipamentos de infraestrutura de redes de dados e de rede elétrica do laboratório bem como na aquisição de equipamentos de informática do laboratório como computadores e manutenções dos mesmos.

Efetuei a troca de equipamentos físicos de “hardware” como discos rígidos e memórias dos computadores. Efetuei ainda a instalação de novos computadores adquiridos, instalação de sistemas operacionais do servidor de dados.

Provi ainda ajuda aos usuários do laboratório com relação à utilização dos computadores, impressoras, recursos de internet, e também na ajuda com a utilização de softwares de edição de textos, planilhas eletrônicas, navegadores de internet, sistemas de impressão.

Efetuei ainda o mapeamento de todas as tomadas de rede elétrica e de rede de dados no padrão “*ethernet*” para verificar quais estavam funcionando corretamente e documentar no mapa da sala no sistema de documentação do tipo “*Wiki*”.

6. Relatos mais recentes das atividades no LELIC

- ✔ 15/01/13 a 29/01/13:
  - ✔ Registrando trabalhos aqui na Wiki do Sourceforge e dotproject;
  - ✔ Pesquisa de add-on Multiplayer para Blender Game Engine;
  - ✔ Interação entre os jogadores multiplayer;

- ✔ Armazenamento de cada mapa no banco de dados do servidor;
- ✔ Testar add-on Multiplayer em 2 máquinas;
- ✔ Implementar tabelas de registros de ações do usuário;
- ✔ Ver como gravar câmera no SQLite;
- ✔ Teste de string da interface para criar e salvar arquivo do SQLite.

✔ 18/03/13 a 29/03/13

- Pesquisa para instanciar um servidor multiplayer manualmente via código-fonte;
- Pesquisa para conectar um cliente ao servidor multiplayer manualmente via código-fonte;
- Implementar: gravar a câmera no banco de dados. (Fazer tabela e código);
- Testar interface de seleção de mapas salvos em disco.
- Documentar o Citta versão Blender, mas preciso que ajudem a documentar.

✔ 01/04/13 a 05/04/13

- Reestruturação da Wiki (esta);
- Teste de Multiplayer acessando o mesmo banco de dados (na mesma máquina);
- Teste de Multiplayer acessando o mesmo banco de dados (MySQL em rede);
- Auxiliar na documentação do projeto.

✔ 08/04/13 a 12/04/13

- Testes com MySQL no Blender.

✔ 12/04/13 a 19/04/13

- Pesquisa de implementação de WebServices em PHP padrão REST;
- Pesquisa de integração do MySQL com PHP;

- Além das outras atividades do laboratório;
- Implementação de WebService de gravação no banco de dados;
- Implementação do script de inserção de objetos do jogo no BD via WebServices.

✔ 22/04/13 a 26/04/13

- Pesquisa de desenvolvimento do WebService de consulta do BD;
- Pesquisa de parser XML no Python com minidom;
- Problemas de redes e servidores de relay de email para o moodle no laboratório;
- Implementação de script de consulta do BD atraves de WebServices.

✔ 29/04/13 a 03/05/13

- Testes de implementação dos scripts de inserção e consulta ao BD via WebServices no Blender;
- Testes de carga e de velocidade de inserção e consulta via WebServices;
- Outras atividades relacionadas ao laboratório sempre surgem.

✔ 10/05/13 a 17/05/13

- Testes de códigos de Webservices no jogo com a variável booleana multiplayer;
- Correção de bugs que vão aparecer.

✔ 20/05/13 a 30/05/13

- Teste do modelo Webservice (implementar o sensor de alteração do mapa);
- Escrever o artigo de multiplayer;
- Corrigir bugs;
- Estudar versão mobile do Cittá;

- Gerir os recursos do laboratório, servidor, computadores, usuários, etc.

7. Publicação e palestra

Fui coautor do capítulo "Desenvolvimento de Módulo Multijogador no Editor de Cidades Città" juntamente com o coordenador Daniel Müller, no livro: *JOGOS-SIMULADORES: relatos de experiência sobre o desenvolvimento do Cittá e do Kimera*, Daniel Nehme Müller, Tânia Maria Hetkowski, Margarete Axt e André Luiz da Silva (orgs). Salvador: EDUNEB, 2014 (no prelo).

Participei também com a equipe do Cosmopolita no Fórum Internacional do Software Livre 2013, com a palestra "Città Cosmopolita – simuladores de cidades em redes", dia 03/07/2013, das 16:00 às 17:00, na sala 40A da PUCRS.

RELATÓRIO DO PROJETO CITTÀ COSMOPOLITA

Projeto processo CNPq: *551846/2011-9*

Relatório de Atividades do Bolsista: *Pablo Felipe Leonhart*

Bolsista processo CNPq: *350266/2014-0*

Coordenador: Daniel Nehme Müller

Período do Relatório: *setembro/2013 a março/2014*

Plano de Trabalho:

Implementação em Blender dos recursos de agente inteligente. Validação da comunicação entre as cidades. Participação das reuniões e saídas de campo. Elaboração de relatórios.

---

1. Introdução

Neste documento são descritas as atividades realizadas como bolsista CNPq, durante o período referido, no projeto de desenvolvimento do simulador de cidades (Città), parte do projeto CIVITAS. O Città é realizado por bolsistas e pesquisadores da empresa Conexum, em parceria com Laboratório de Estudos em Linguagem, Interação e Cognição (LELIC) da Faculdade de Educação da UFRGS.

2. Tarefas

2.1. Contextualização

Pelo fato de ter iniciado no projeto com o mesmo já em andamento, necessitei de um período para conhecer a ideia do mesmo, qual o seu propósito, e claro estudar o código que já havia sido desenvolvido. Na época, o Città ainda estava rodando na versão Java com renderização pelo Blender, a qual é de complexo entendimento, pois se tratavam de muitos arquivos de código devido a transição do Java para o Blender.

Pouco tempo depois recebo a informação de que o projeto passaria então a ser trabalhado com a linguagem Python, a qual já vem integrada no próprio Blender, o que facilita praticamente tudo, a codificação, testes, depuração, etc. Inicialmente até fiquei meio assustado, pelo fato de ter iniciado os estudos com a versão anterior, e até por não conhecer a nova linguagem de programação, temia então que fosse demorada a readaptação, mas com a prática vi que era interessante trabalhar desta maneira, e hoje vejo o projeto mais limpo, mais enxuto nesta nova versão.

2.2. Inserção do Agente inteligente no jogo

Uma das tarefas iniciais foi a de inserir o agente no ambiente do Città, o qual irá ajudar e dar dicas aos alunos durante o jogo. Esta tarefa foi dividida em duas etapas, a primeira de reconhecimento e tratamento das variáveis no ambiente do simulador, e a segunda a da inserção na tela do agente, no caso a Maga Vitta.

Variáveis do ambiente

Baseado no que havia sido previamente construído na versão Java, iniciei a construção dos módulos que controlariam os índices que eram alterados com a realização de novas construções. Para isso adicionei 2 scripts: atualizadorAmbiente e gerenciadorAmbiente, como as classes que estavam na versão em Java.

Neles eu tenho funções definidas por "def". As variáveis da cidade ficam no atualizadorAmbiente e são alteradas por essas funções definidas. Quando temos a criação de um objeto, o script placeObject chama uma função do gerenciadorAmbiente e atualiza as variáveis. Da mesma forma, quando um objeto é apagado, o script eraseObject chama uma função para atualizar os dados. Também estão implementados getters e setters de cada variável.

Maga Vitta

Terminada a primeira etapa, foi iniciada a inserção do agente no ambiente, em atividade exercida em conjunto com o outro bolsista da UNISC, Yuri Dias Meira. Inicialmente foram desenvolvidos simples testes para verificar como seria o comportamento da Maga e em que circunstâncias ela iria se manifestar.

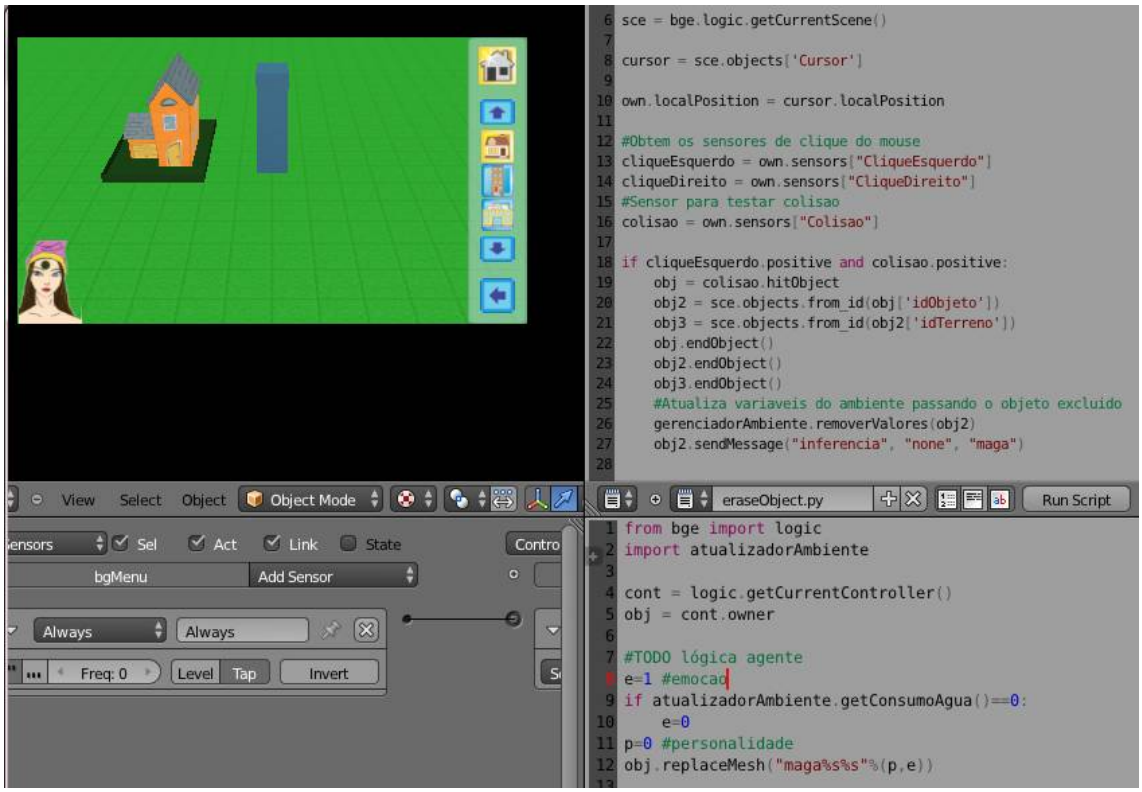


Figura 1. Maga Vitta no ambiente do Città

Essa etapa ainda está em curso, uma vez que as decisões tomadas pelo agente são resultado da rede bayesiana, a qual está realizando-se pesquisa, escolha e configuração das bibliotecas a serem usadas para a implementação de rede bayesiana em Python. A implementação do avatar do agente e sistema de substituição do avatar, pode ser de tempo em tempo e/ou a cada construção ou remoção de objeto. Essa implementação já está preparada para a inclusão de novos avatares sem a necessidade de mudar o código.

2.3. Pôster

Não menos importante, outro item desenvolvido foi o pôster do Città apresentado pela professora Andréa Konzem no evento CBIE realizado em novembro de 2013.

Figura 2. Pôster Città

3. Desenvolvimento

A mais recente meta do projeto foi a de desenvolver um componente para passar um mapa do Città do cliente para o servidor e vice-versa, tarefa a qual será realizada quando o jogo for executado na forma multiplayer, e também passar o mapa do jogo para o banco local quando o usuário está em modo singleplayer. Este componente abrangeu as seguintes tarefas:

- importar/exportar tabela do banco sqlite (em python);

- transferir a tabela de/para o cliente/servidor (em php);

- importar/exportar tabela do mysql (em php).

Nas seções seguintes são explicados o desenvolvimento de cada uma delas.

3.1. Banco SQLite

No modo de um jogador por mapa as questões envolvendo o banco de dados ocorre simplesmente nas ações de salvar e carregar localmente o arquivo .sqlite do mapa, não se faz necessária a interação com um servidor neste caso. Neste módulo os scripts já estavam implementados, eles apenas foram aperfeiçoados e ligados aos sensores do jogo para que entrassem em execução.

No arquivo salvaMapaBD.py é feita a conexão com o banco de dados, em seguida cada objeto da cena do jogo que for considerado uma construção tem suas características coletadas, as quais são montadas em instruções insert que na sequência são executadas, atualizando a base. Esta operação é realizada quando o jogador pressiona a tecla G quando está no ambiente de sua cidade.

Já no arquivo que recupera os dados, carregaMapaBD.py, também se faz a conexão com o banco, o qual tem o nome da cidade em questão, após isso é feito um *select* de tudo o que está na base. Para cada linha retornada, é adicionado um objeto cujas características são definidas pela busca feita no banco, sendo assim o esquema da cidade é restaurado. A tecla responsável por ativar essa operação é a C.

Dessa maneira a tarefa de importar e exportar a tabela do banco sqlite no python é realizada sem maiores dificuldades.

3.2. Transferindo tabela do cliente para servidor

Neste módulo a tarefa é exportar os dados contidos no banco do cliente, os quais se apresentam no formato SQLite e enviá-los via PHP para o servidor, que por sua vez trabalha com banco MySQL. Optou-se por uma maneira que melhor trouxe o resultado esperado, mas que nem por isso seja a mais otimizada ou mais simples de programar. A seguir o processo é descrito.

No cliente, quando o jogador salvar seu mapa o script salvaMapa.py irá criar um arquivo .sql no qual estarão todos os dados dos objetos que estão ativos em cena, estas

informações são montadas em strings de modo que cada objeto tenha um insert correspondente. Após ter sido varrido todo o mapa é feito o acesso à página importarSQLite.php, a qual está no servidor e será responsável por processar os dados enviados pelo cliente.

Para ter acesso à páginas no python é usado o módulo urllib, o qual fornece meios de acessar servidores HTTP por meios de requests GET e POST de forma direta. Neste caso optou-se por usar o método GET para transferir o conteúdo do arquivo sql gerado anteriormente.

```
46    arquivo.close()
47    #Abre o arquivo sql
48    arquivo = open(mapa, 'r')
49
50    #pagina de requisicao
51    url = 'http://localhost/importarSQLite.php/'
52    #parametros: nome do banco e o conteudo do arquivo
53    params = {'dbName': bge.logic.mapaAtual, 'conteudo': arquivo.read()}
54    #codifica os parametros
55    params = urllib.parse.urlencode(params)
56    #abre a pagina referida passando os parametros via GET
57    f = urllib.request.urlopen(url+'?'+params)
58    f.close()
59
60    #Fecha o arquivo
61    arquivo.close()
```

Figura 1. Acesso à url via método GET com Python

Como pode ser visto no trecho de código acima é possível acessar qualquer url e ainda enviar parâmetros a ela de forma direta. Dessa maneira o mapa do cliente é enviado para o servidor.

O passo seguinte é o tratamento desses dados no php. A página já citada, importarSQLite, recebe dois parâmetros via GET, o nome da base de dados (‘dbName’), isto é, o nome da cidade do jogador e o conteúdo do arquivo sql (‘conteudo’), o qual contém os dados de todos os objetos já montados na forma de inserts. Primeiramente faz-se o acesso ao servidor, depois a partir do nome da cidade informado é verificada a existência de uma base com este nome, caso ela não exista, ou seja, é a primeira vez que aquele mapa é salvado, então cria-se um banco com aquele nome.

Em seguida, a tabela mapa (nome default para todas as cidades) é excluída caso ela já exista, e criada novamente, ou pela primeira vez se for o caso. E por fim o conteúdo recebido é quebrado em pedaços menores de tal modo que cada um deles contenha uma operação insert, e estes então são executados na nova tabela. Note que esta quebra ocorre porque o conteúdo recebido é uma string apenas, e esta se conter mais de uma operação sql não pode ser executada pelo comando query do MySQL. A Figura 2 mostra o trecho de código correspondente.

```
22      #Chama a funcao que ira separar os INSERTS, os quais estao na variavel conteudo
23      if (mysql_multiple_query($_GET['conteudo'], $con)or die(mysql_error())){
24          echo "objetos inseridos";
25      }
26
27      function mysql_multiple_query($query, $link_identifier = null){
28          $result = false;
29          #Separa em tokens o token maior (conteudo) conforme apareca um \n
30          $tok = strtok($query, ";;\n");
31
32          #Enquanto houver tokens executa as queries
33          while ($tok) {
34                  if ($link_identifier !== null) {
35                          $result = mysql_query($tok, $link_identifier);
36                  } else {
37                          $result = mysql_query($tok);
38                  }
39
40                  $tok = strtok(";;\n");
41          }
42
43          return $result;
44      }
```

Figura 2. Código PHP para dividir queries MySQL

Com isso o processo de exportar o mapa do cliente para o servidor está completo, tanto o usuário que salvou o mapa quanto os demais usuários da mesma cidade podem agora acessar os dados contidos no servidor.

3.3. Transferindo tabela do servidor para cliente

Após concluir o esquema para salvar os dados no servidor, agora deseja-se carregá-los para o banco do cliente. O módulo aqui trabalhado faz o processo inverso ao que foi visto acima, nesse caso o conteúdo do banco MySQL que está no servidor deve ser consultado e o resultado deve ser recebido pelo cliente para que atualize sua base de dados e consequentemente o estado de sua cidade.

Inicialmente o carregamento do mapa se dá apenas quando o usuário tecla C, isto é, apenas quando ele desejar carregar seu jogo, não havendo ainda uma atualização em tempo real do que está acontecendo na cidade por parte de outros jogadores. De qualquer forma a iniciativa de carregar o mapa do jogo deve partir do cliente, seja pelo usuário ou por código, uma vez que o servidor não sabe quando isso deve ser feito.

Sendo assim, quando o jogador solicitar o carregamento do mapa a partir do script que controla isso (carregaMapaBD.py), é feita uma solicitação via GET a uma página do servidor responsável por isso, a qual chama-se GetDatabase. Como se vê na Figura 3, testa-se o modo de jogo (multiplayer ou singleplayer), caso seja o primeiro abre a página do servidor enviando como parâmetro o nome do banco que se deseja carregar.

```
24      #Se o mapa eh alterado por mais de um jogador carrega o conteudo do banco do servidor
25      if bge.logic.modoJogo == "multiJogador":
26
27          #Envia solicitacao para pagina do servidor informando via GET
28          #o nome do banco (nome do mapa em que se esta) ao qual deseja se conectar
29
30          url = 'http://localhost/GetDatabase.php/'    #pagina de requisicao
31          params = {'dbName': bge.logic.mapaAtual}     #parametros: nome do banco
32          params = urllib.parse.urlencode(params)
```

Figura 3. Enviando parâmetros via GET com Python

Feito isso, a página php agora é responsável por buscar os dados no servidor. Para tal, recebe via GET o nome da base ('dbName') e com isso conecta-se ao banco que está em MySQL. Com a conexão aberta são listadas todas as tabelas do banco, sabe-se que em todo e qualquer banco gerado pelo Città só existe uma tabela "mapa", mas se futuramente vierem a existir novas tabelas esta função já está preparada para capturá-la também.

Para cada tabela é selecionado todo seu conteúdo, e cada linha da tabela é montada na maneira de um insert, e todas são unidas numa string. Essa string está pronta para ser processada pelo código Python para que atualize o banco do cliente. Depois de finalizada a busca na tabela, a string final é então escrita num arquivo do servidor, o qual trata-se de uma página php também mas que não contém nenhum tipo de código nela, apenas a string com os inserts do banco. Essa página criada recebe o mesmo nome da cidade de modo que facilite o seu acesso por todos os clientes da mesma e que não cause conflitos com clientes de outros mapas.

Depois disso volta-se ao código Python, nele o cliente acessa a página que foi recém criada ou atualizada, mas não requisita nada nem recebe dado algum, apenas lê o conteúdo dela, isto é, percorre o "código" php da página, que no caso é apenas uma string com os inserts. Fala-se em string porém, o conteúdo apresenta-se no formato de bytes, sendo então necessária a conversão deste para string de fato.

```
44        #Pagina para leitura dos INSERTS sql gerado pela pagina anterior
45        url = urllib.request.urlopen("http://localhost/"+bge.logic.mapaAtual+".php")
46
47        c.execute("DROP TABLE if EXISTS mapa")
48        c.execute('''CREATE TABLE mapa
49              (nome text, posX real, posY real, posZ real, rotacao real, pessoas real, agua real,
50
51        while True:
52            s = bytes.decode(url.readline())
53        #I'm guessing this would output the html source code?
54            if len(s) == 0:
55                break
56            print(s)
57            c.execute(s)
```

Figura 4. Lendo dados de uma url

Na Figura 4 vemos o acesso à página conforme o nome do mapa atual, em seguida há um laço de repetição para ler todas as linhas da página, cada linha é convertida em string e enfim executada a instrução contida nela. Desse modo ocorre a atualização do banco local sqlite. O trecho de código seguinte já foi descrito nos primeiros parágrafos, é feita uma conexão com o banco, lido todo seu conteúdo atualizado e convertido em objetos, os quais são adicionados à cena.

3.4. Inicialização do mapa

Foram também alterados arquivos como defineMap.py e initialize.py de modo que conforme as escolhas do jogador seja ela mutliplayer ou singleplayer e se é um novo mapa ou não, o mapa possa ser inicializado de forma adequada. Por exemplo, se o usuário quer jogar com outras pessoas e decide carregar um mapa, ao inicializa-lo já busca-se as informações contidas banco de dados do servidor.

3.5. Problemas existentes e enfrentados

Objeto é restaurado, mas não "existe", isto é, os objetos são carregados adequadamente do banco em suas posições corretas, porém não é possível exclui-los por

exemplo. Eles aparecem no mapa, mas não são obstáculos para outros objetos. Isso gera conflito, pois estes objetos carregados quando ocorrer uma nova operação salvar não serão inclusos no banco, já que eles não existem na cena do jogo. Essa situação acontece quando a tabela do mapa é excluída a cada operação salvar, por outro lado, se não excluir a tabela ao iniciar um mapa ou carregá-lo tudo que já existiu no mapa uma vez é "criado" novamente, ou seja, se excluir um objeto ele reaparecerá quando o jogo for carregado.

Outra questão refere-se a forma como os dados são transferidos. Acredita-se que uma maneira melhor para enviar os dados do cliente para o servidor seja lendo a base de dados local com o uso de funções do SQLite no PHP. Porém, por algum motivo de incompatibilidade de versões de ambos não foi possível o uso das tais funcionalidades.

4. Referências

- **Aprendendo Python: Entrada/Saída**. Link disponível em: *http://homepages.dcc.ufmg.br/~joaoreis/Site%20de %20tutoriais/aprendendopython/entrada-saida.html*

- **Rede em Python**. NOGUEIRA, Antonio Sérgio. Link disponível em: *http://pt.slideshare.net/asergionogueira/acessando-a-rede-com-o-python*

- **Python v2.7.2 documentation**. Link disponível em: *http://turing.com.br/pydoc/2.7/tutorial/stdlib.html#acesso-a-internet*

- **PHP: Upload de Arquivos.** BELEM, Tiago. Link disponível em: *http://www.linhadecodigo.com.br/artigo/3578/php-upload-de-arquivos.aspx*

- **Stackoverflow - AttributeError: 'module' object has no attribute 'urlopen'**. Link disponível em*: http://stackoverflow.com/questions/3969726/attributeerror-module-object-has-no-attribute-urlopen*

- **Stackoverflow - Simple URL GET/POST function in Python**. Link disponível em: *http://stackoverflow.com/questions/4476373/simple-url-get-post-function-in-python*

- **Python: HTTP GET e POST com urllib**. Link disponível em: *http://codare.net/2006/09/20/python-http-get-e-post-com-urllib/*

- **Como passar informação entre páginas em PHP (Parte II)**. Link disponível em: *http://www.escolacriatividade.com/como-passar-informacao-entre-paginas-em-php-parte-ii/*

- **Converting str to bytes via Methods**. Link disponível em: *http://python.about.com/od/python30/ss/30_strings_3.htm*

- **Webmasters online - Várias Instruções Sql Ao Mesmo Tempo**. Link disponível em: *http://forum.wmonline.com.br/topic/179648-varias-instrucoes-sql-ao-mesmo-tempo/*

RELATÓRIO DO PROJETO CITTÀ COSMOPOLITA

Projeto processo CNPq: *551846/2011-9*

Relatório de Atividades do Bolsista: *Yuri Dias Meira*

Bolsista processo CNPq: *375951/2013-0 e 350266/2014-0*

Coordenador: Daniel Nehme Müller

Período do Relatório: *setembro/2013 a fevereiro/2014 e março a junho/2014*

Planos de Trabalho:

Execução de testes com o agente inteligente. Testagens dos recursos multijogador. Realização de simulações de uso para interação com agente inteligente. Apoio nas implementações em Blender.

Implementação computacional do módulo de controle e conexão entre bancos de dados das versões do servidor e do cliente no sistema Città.

---

1. Introdução

Neste relatório são descritas as atividades realizadas, dentro do período de janeiro à junho de 2014, para o desenvolvimento do agente do jogo Città, parte do projeto Cosmopolita.

Fiquei encarregado pela implementação da lógica do agente, em Python, no Blender. O agente já havia sido implementado em java com o uso de rede bayesiana.

1.1. Visão do Projeto

Desenvolvimento de um software gráfico interativo (jogo) para apoiar pedagogicamente na construção de conhecimento das crianças de 4º. e 5º. ano do ensino fundamental, no que se refere ao desenvolvimento de capacidades que lhes permitam pensar-construir-organizar uma cidade virtual a partir da realidade infantil e seus conhecimentos incorporados através dos conteúdos oferecidos no aprendizado dos sistemas que regulam uma cidade e suas relações.

2. Desenvolvimento

As duas primeiras semanas no projeto foram usadas para a leitura da wiki do projeto e familiarização com a versão do Città em Java e da versão em Python do Blender conforme era implementada. Período também usado para o aprendizado e prática de Python, assim como o estudo de redes Bayesianas.

Já familiarizado com o ambiente foram realizadas, em conjunto com outro bolsista, as primeiras atividades no desenvolvimento do agente no Blender que envolveram a implementação dos scripts gerenciadorAmbiente.py e atualizadorAmbiente.py. Os dois scripts foram baseados nas classes Java de mesmo nome que já eram implementadas na versão anterior.

O script atualizadorAmbiente.py contém as variáveis que são utilizadas pelo agente e seus respectivos getters e setters. Já o gerenciadorAmbiente.py faz a interface com o ambiente e as variáveis, cada mudança no mapa do jogo é captada pelo gerenciador que atualiza as variáveis de acordo com a mudança. O gerenciador foi melhorado em relação à versão anterior, em java as variáveis dos objetos modificados deviam ser programadas especificamente para cada tipo de objeto. Na versão do Blender isso já não é mais necessário, o script sabe como tratar as variáveis de qualquer tipo de objeto e ajustar seus valores de maneira correta.

Já com a gerência do ambiente completa, foi criado o script maga.py que é o responsável pela tomada de decisão do agente. O agente faz uso de uma rede Bayesiana para suas decisões. Até o momento o script só é capaz de gerenciar qual emoção e personalidade o agente terá, porém sem a lógica de inferência devido à dificuldade de incluir no ambiente do Blender as bibliotecas necessárias para sua implementação.

As emoções e personalidades do agente são adicionados como novos objetos no Blender. O script já está preparado para receber qualquer número de emoções e personalidades, ele foi implementado de forma semelhante à uma matriz. Os objetos devem seguir o nome magaxy onde x é a personalidade do agente e y a emoção. Por exemplo, se a personalidade atual for normal = 0 e a emoção for irritada = 1, o objeto que a representa deve ser adicionado com o nome maga01. Assim, o script já saberá qual objeto deve procurar para substituir a emoção sendo apresentada pelo agente no momento.

Após a implementação dos scripts foi necessária a edição dos arquivos placeObject.py e eraseObject.py, assim no momento em que um objeto é criado ou deletado, o script de gerenciamento do ambiente é notificado para atualizar as variáveis do ambiente, como consumo e produção de água, lixo, energia, etc. O mesmo acontece com o script maga.py que deve ser chamado quando ocorrem mudanças no ambiente para realizar a inferência, mas neste caso o script está ligado ao objeto maga. O objeto contém um sensor de mensagem conectado ao script, então para notificá-lo é necessário o uso do método sendMessage("inferencia", "none", "maga") a partir dos scripts placeObject e eraseObject.

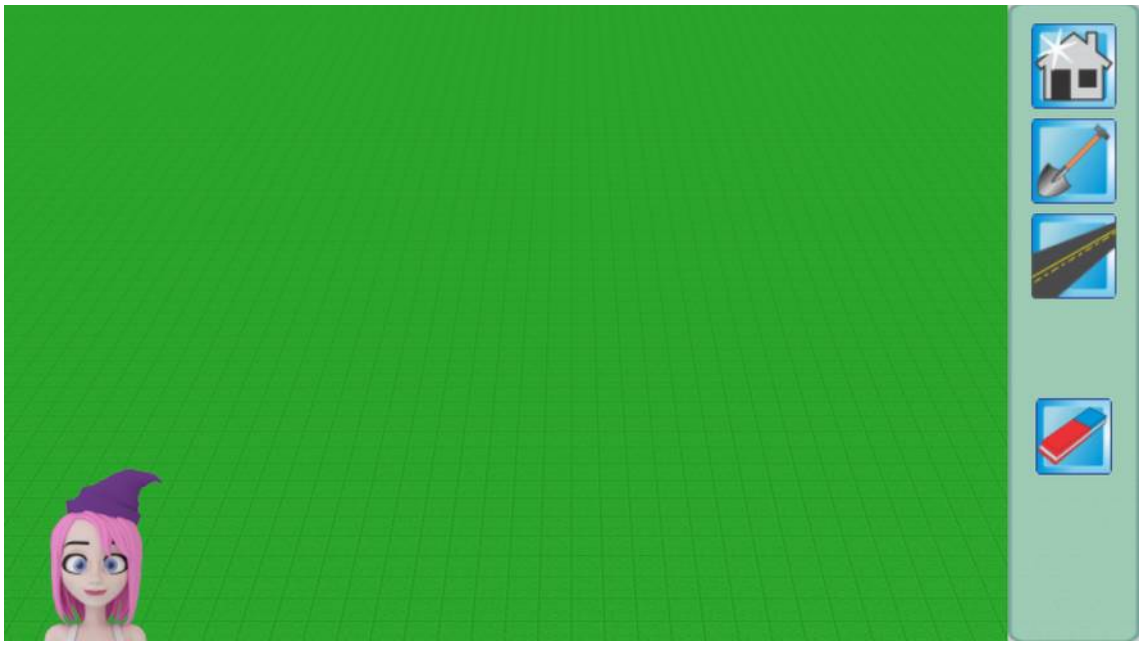
Após estas atividades foi pedido o desenvolvimento de importação e exportação de tabelas sqlite do blender para o servidor PHP e vice versa. Primeiramente foi necessário tornar funcional o salvamento e carregamento local dos dados no banco. Logo após foi desenvolvido um protótipo da parte de exportação de tabelas apenas do Blender para o servidor, estando pendente o caminho inverso.

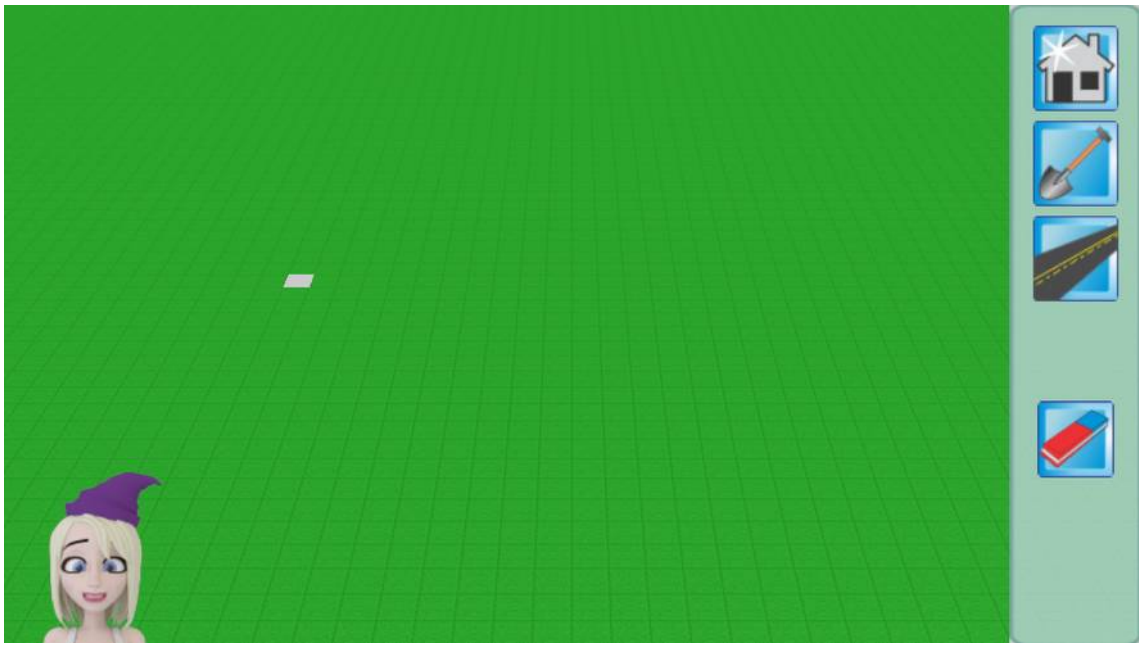
Foi dada continuidade à implementação da rede bayesiana em Python, assim como o seu treinamento. O treinamento consiste na execução de casos dentro da rede com resultados já conhecidos, para que quando seja feito o cálculo para casos inéditos sejam usados os dados

já encontrados anteriormente, afetando as probabilidades dos mesmos. Quanto maior a quantidade de casos presentes no treinamento do agente, maior será a sua precisão no momento da inferência da rede.

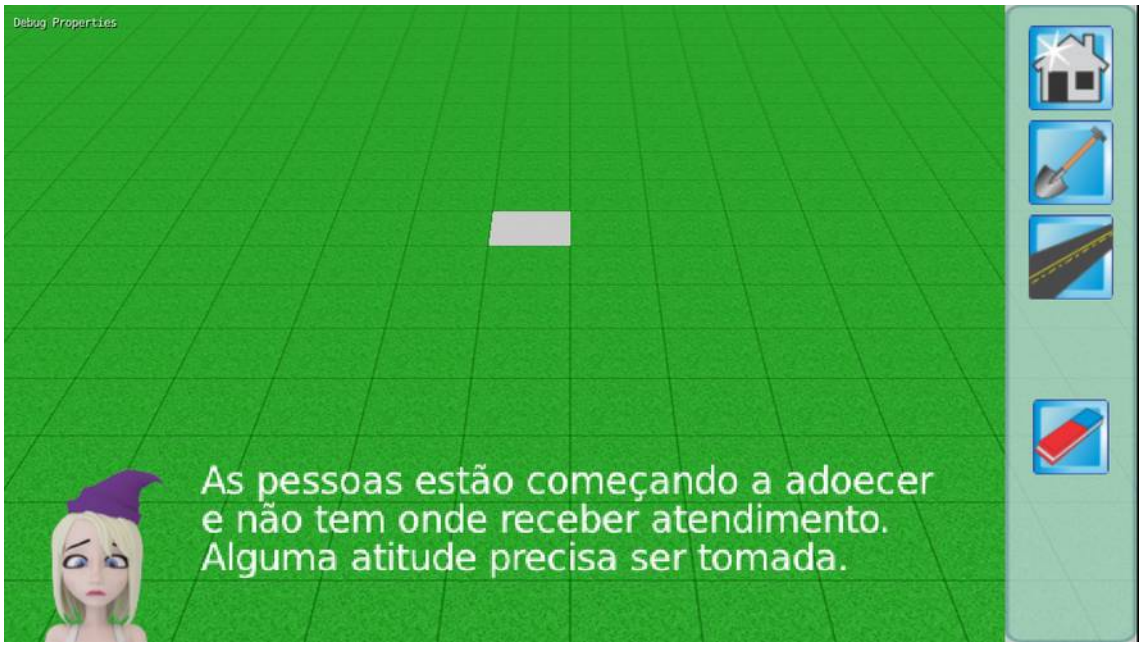
A implementação da rede se encontra nos scripts maga.py e NaiveBayes.py, já o treinamento da rede foi realizado no script treinamentoMaga.py, apresentando diversos casos básicos com resultados pré definidos. O script maga.py é o responsável pela inferência em si, fazendo uso do NaiveBayes.py para a lógica de como ela deve ser feita e o treinamentoMaga.py para as probabilidades que serão caculadas.

Já para a inserção das novas imagens de avatar fornecidas pela equipe, representando as diferentes emoções da maga são usados objetos da Blender Game Engine, foram criados objetos específicos para cada emoção, sendo eles controlados pelo objeto principal que está presente na cena HUD no projeto Blender do Città, este com o nome de “maga”, e os específicos de cada emoção seguindo a nomenclatura “magape”, sendo o "p" substituído pelo número da personalidade, funcionalidade ainda não implementada na versão Blender, porém já compatível com fúturas alterações, e o “e" substituído pelo número da emoção, o que é de momento utilizado nos scripts, conforme seu valor uma nova emoção para o agente será representada na cena “HUD”.

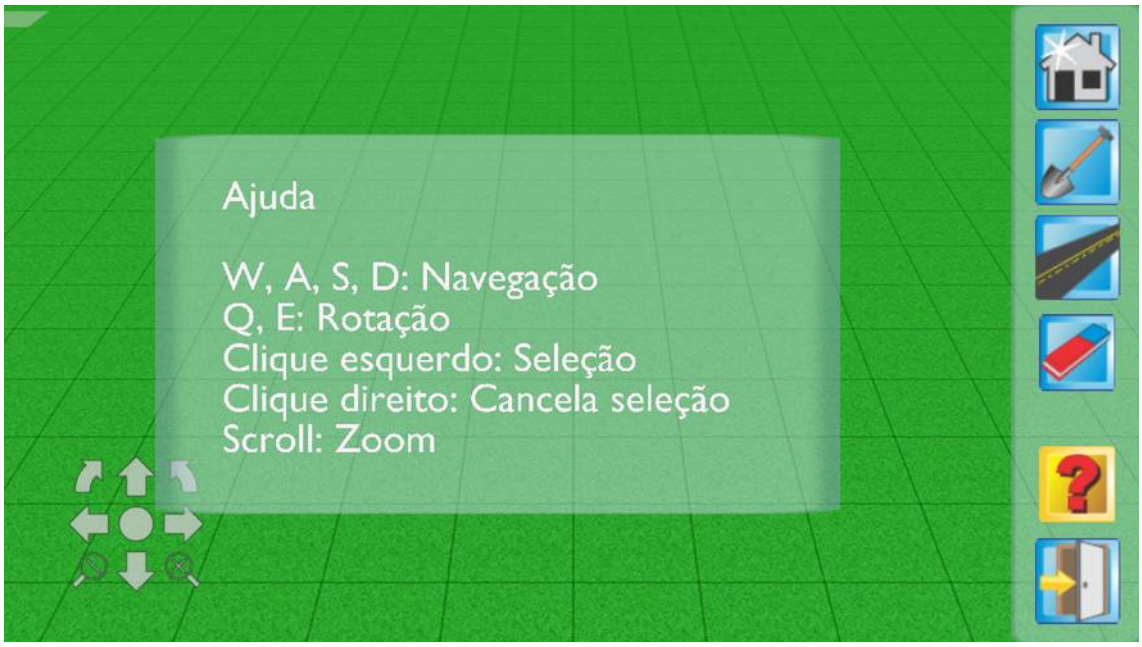




Além da representação das emoções da maga, também foram incluídas suas falas, através do uso do objeto de texto do Blender e scripts em Python responsáveis por controlá-lo, as frases aparecem na tela conforme a situação do ambiente e ações do usuário, esclarecendo a interpretação do agente sobre o ambiente e reforçando a expressão da sua emoção através do texto.



Outra atividade foi a criação de um botão de ajuda para auxiliar o usuário durante a experiência do jogo. Quando pressionado exibe uma tela contendo os comandos básicos que podem ser executados pelo jogador. Foi implementado tomando como base os outros botões já presentes na barra lateral da cena HUD, faz uso dos estados que os objetos podem possuir no Blender, ele inicia em um estado normal, que fica aguardando um clique sobre ele para realizar a ação de exibir a janela de ajuda, quando pressionado ele muda o estado para ativo, neste estado o botão troca de cor e exibe a janela de ajuda, quando o jogador realiza o clique do mouse em qualquer área da tela a janela de ajuda é fechada e o botão retorna ao estado normal.



Também foi realizada uma alteração no menu principal, que consiste na possibilidade de desfazer a seleção do caderno. O menu é uma cena com uma mesa e diferentes objetos

sobre ela, de momento apenas o caderno e a mochila são funcionais, sendo o caderno para inicializar o mapa, sendo ele um novo ou um mapa salvo e a mochila para fechar o jogo. Anteriormente após a seleção do caderno para iniciar um novo mapa, não era mais possível voltar atrás e sair do jogo. Por isso foi implementada a possibilidade de desfazer a seleção com um clique direito do mouse.

Após o clique esquerdo sobre o caderno para selecioná-lo, a câmera foca apenas o ele e habilita os campos para seleção do nome do mapa e botão de iniciar jogo, neste momento se houver um clique direito em qualquer parte da tela, a câmera retorna ao seu estado inicial e os campos e botões são novamente desabilitados para que o jogo não seja iniciado por acidente. Isto é feito com o uso dos sensores do Blender e o auxílio de scripts Python para realizar as alterações de estado do caderno e câmera. Já a animação de foco no caderno e retorno ao estado inicial fazem uso dos atuadores.

A figura acima representa a tela inicial, onde é possível selecionar o caderno para dar início ao jogo ou a mochila para sair, quando a seleção do caderno é desfeita esta tela volta a aparecer. A figura da página seguinte mostra o foco no caderno após a seleção.

Incluída também a Licença Pública Geral, GNU GPL, idealizada por Richard Matthew Stallman em 1989 e atualmente a licença de maior utilização por projetos de software livre. Baseia-se em 4 liberdades:

0. A liberdade de executar o programa, para qualquer propósito
1. A liberdade de estudar como o programa funciona e adaptá-lo para as suas necessidades, sendo o acesso ao código-fonte um pré-requisito.
2. A liberdade de redistribuir cópias de modo que você possa ajudar ao próximo.
3. A liberdade de aperfeiçoar o programa, e liberar os seus aperfeiçoamentos, de modo que toda a comunidade se beneficie deles, sendo também o acesso ao código-fonte um pré-requisito.

Os direitos do autor devem ser mantidos, assim como não podem ser impostas restrições que impeçam o código de ser redistribuído da mesma maneira que foi adquirido. A licença é redigida em inglês e não aceita traduções, pois erros na sua tradução podem torná-la inválida.

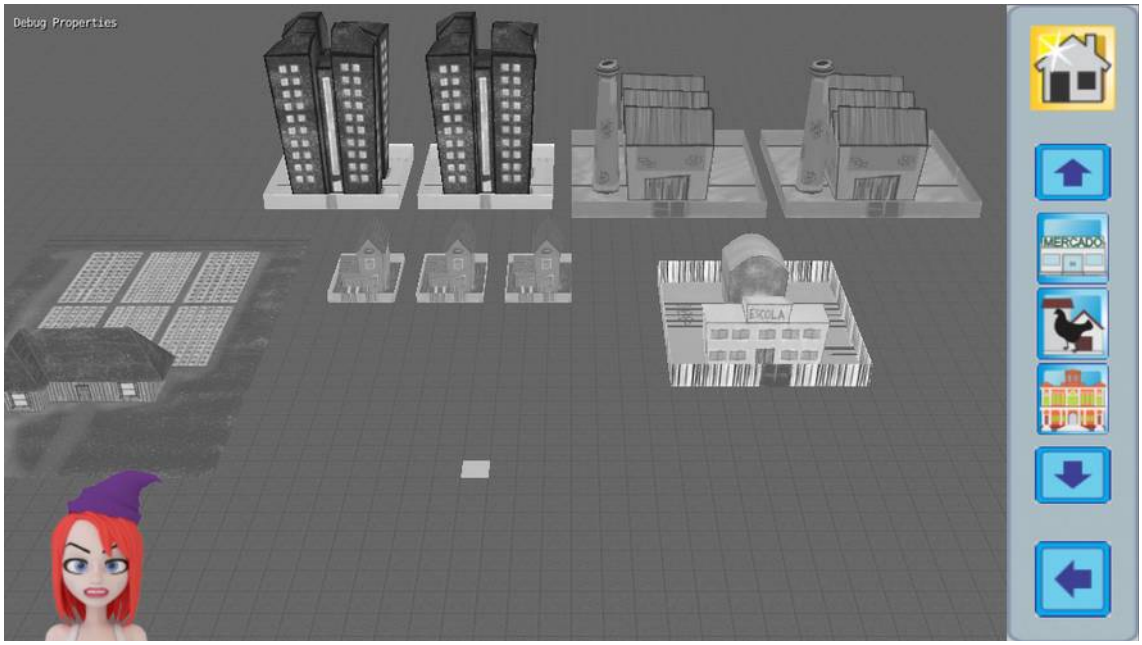
Foram incluídos, no projeto do Blender, os arquivos “LEIAME" e “GnuGpl”, assim como um pequeno cabeçalho em todos os scripts contidos no projeto. Os conteúdos dos arquivos constam respectivamente no anexo I e no link http://gnu.org/licenses/gpl.html.

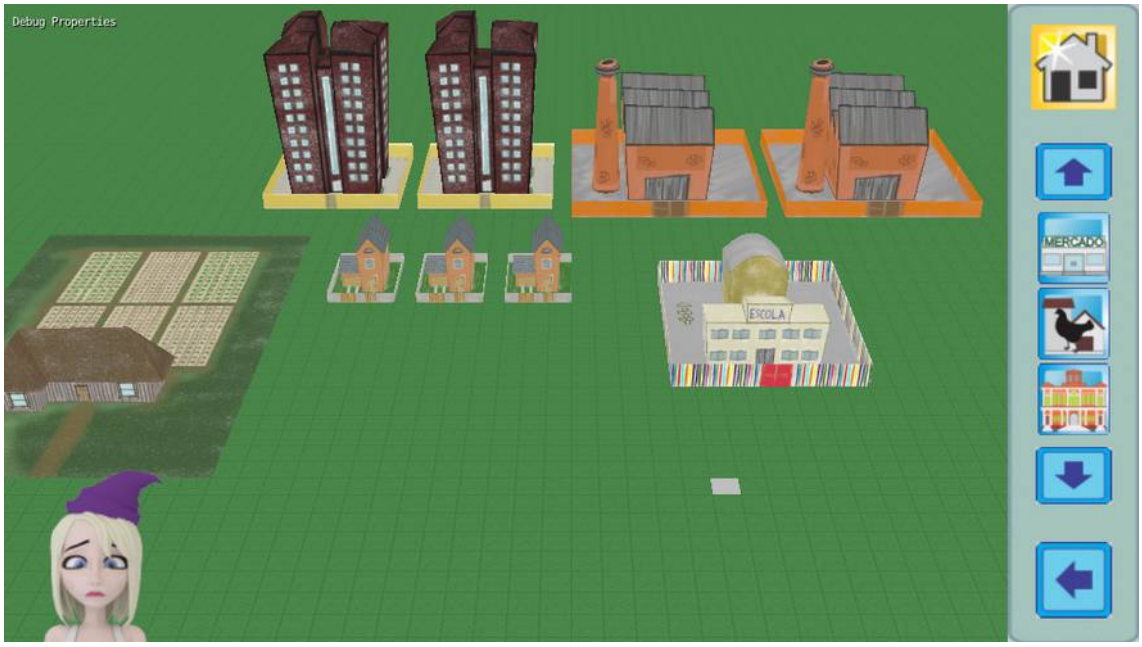
Implementada ainda a alteração do ambiente de jogo conforme a emoção do agente, tornando o ambiente mais escuro conforme o quão negativa for a emoção do agente, ou mais vibrante para emoções positivas. Como o Blender não suporta alteração ou importação de texturas em tempo real, para essa funcionalidade fez-se necessária a inclusão de diversos objetos novos para representar as diferentes tonalidades que poderiam ser atribuídas. No momento em que o agente muda de emoção, é realizada uma varredura em todos os objetos que foram construídos no ambiente e para cada objeto encontrado é feita a substituição pelo

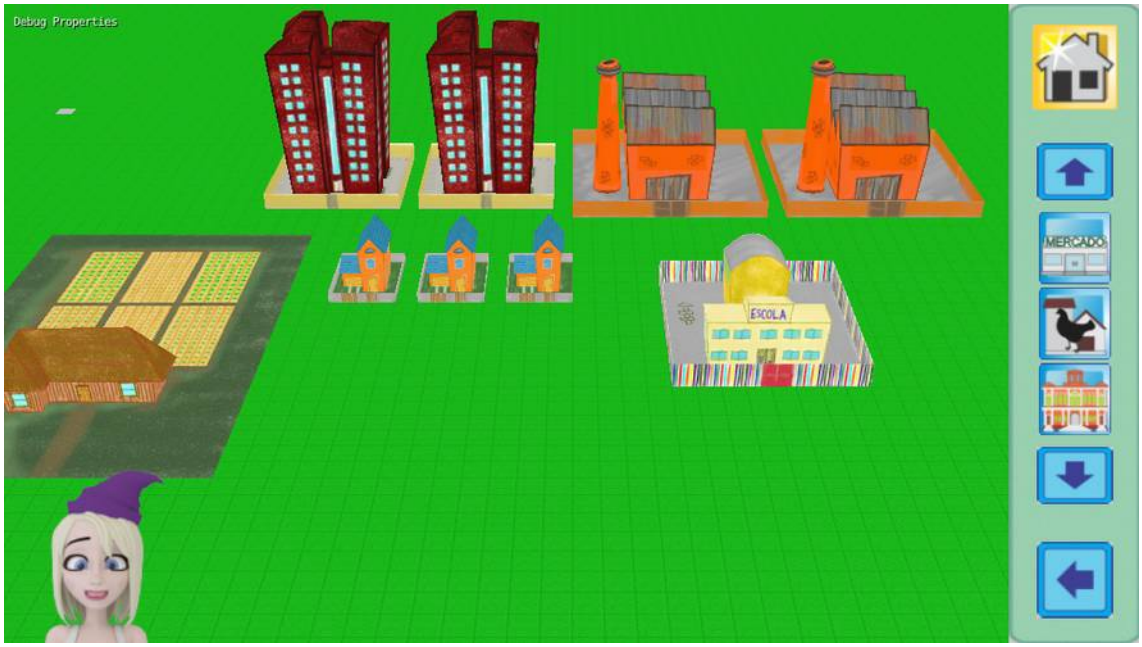
objeto correspondente a emoção atual do agente, processo feito através do script “trocaTexturas.py”.

trocaTexturas.py

```
import bge


def troca(e):
    cenas = bge.logic.getSceneList()
    for cenaAux in cenas:
        if cenaAux.name == "Gameplay":
            cena = cenaAux
    for obj in cena.objects:
        print (obj.name)
        print (obj['objeto'])
        if obj["objeto"]:
            obj.replaceMesh("%s%s"%(obj.name.lower(),e))
        elif "terreno" in obj:
            if e>1:
                obj.replaceMesh("%s%s"%(obj.name.lower(),0))
            else:
                obj.replaceMesh("%s%s"%(obj.name.lower(),1))
```


Debug Properties
MERCADO
ESCOLA


Debug Properties
MERCADO
ESCOLA



Além das ativadas descritas também foi necessário o auxílio de alunos da graduação que desenvolveram seus trabalhos de conclusão de curso ao redor do projeto Città. Estes

alunos implementaram diversas funcionalidades ao ambiente, incluindo partes implementadas ainda na versão java.

5. *Referências*

Blender Wiki. Logic, Properties and States:
http://wiki.blender.org/index.php/Doc:2.6/Manual/Game_Engine/Logic
Blender API Documentation:
http://www.blender.org/documentation/blender_python_api_2_63_17/contents.html
BlenderTips.com. Blender 3D Hotkeys, Shortcut and Reference:
http://blendertips.com/hotkeys.html
Python 2 Documentation:
http://docs.python.org/2/
Introducing Bayesian Networks:
http://www.csse.monash.edu.au/bai/book/BAI_Chapter2.pdf

Anexo I

CIVITAS é um jogo educacional colaborativo para simulação de cidades. O objetivo deste jogo é servir como apoio educacional para a construção de cidades virtuais criadas por grupos de alunos.

Direitos Autorais Reservados (c) 2008 LELIC - Laboratório de Estudos em Linguagem Interação e Cognição - Faculdade de Educação - Universidade Federal do Rio Grande do Sul.

Este programa é software livre; você pode redistribuí-lo e/ou modificá-lo sob os termos da Licença Pública Geral GNU conforme publicada pela Free Software Foundation; tanto a versão 3 da Licença, como (a seu critério) qualquer versão posterior.

Este programa é distribuído na expectativa de que seja útil, porém, SEM NENHUMA GARANTIA; nem mesmo a garantia implícita de comerciabilidade ou adequação a uma finalidade específica.

Consulte a Licença Pública Geral do GNU, disponível no arquivo GnuGPL deste projeto e no endereço ***http://www.gnu.org/licenses/gpl.html*** para mais detalhes.

Você deve ter recebido uma cópia da Licença Pública Geral do GNU junto com este programa; se não, escreva para Laboratório de Estudos em Linguagem, Interação e Cognição (LELIC) - Prédio 12201 - sala: 801- Av. Paulo Gama, 110 - CEP 90040-060 - Tel: (51) 3308-4149

RELATÓRIO DO PROJETO CITTÀ COSMOPOLITA

Projeto processo CNPq: *551846/2011-9*

Relatório de Atividades do Bolsista: *Andréa Aparecida Konzen*

Bolsista processo CNPq: *383987/2013-0*

Coordenador: Daniel Nehme Müller

Período do Relatório: *setembro/2013 a maio de 2014*

Plano de Trabalho:

Apoio no acompanhamento e gerenciamento das atividades dos demais bolsistas. Inserção do agente inteligente Maga Vitta no jogo Città na nova versão Blender. Coordenação de reuniões. Participação das visitas de campo.

---

1. Introdução

Neste relatório são descritas as atividades realizadas que contribuíram para o desenvolvimento do agente do jogo Città, parte integrante do projeto Cosmopolita além das atividades gerais no projeto.

1.1. Visão geral do Projeto

Desenvolvimento de um software gráfico interativo (jogo) para apoiar pedagogicamente na construção de conhecimento das crianças de 4º. e 5º. Ano do ensino fundamental, no que se refere ao desenvolvimento de capacidades que lhes permitam pensar-construir-organizar uma cidade virtual a partir da realidade infantil e seus conhecimentos incorporados através dos conteúdos oferecidos no aprendizado dos sistemas que regulam uma cidade e suas relações.

2. Atividades realizadas

- Auxílio na organização das reuniões.
- Modelagem do agente: emoções e personalidade.
- Estudo do vínculo das emoções e personalidade com os diálogos já criados.
- Estudo do questionário a ser aplicado para a busca dos traços de personalidade.
- Estudos das expressões faciais do agente relacionadas à personalidade e às emoções.
- Análise do material existe, coletado nas saídas de campo a fim de extrair parâmetros a serem usados nos estudos previamente apresentados.

- Análise de trabalhos já existentes a fim de buscar características que podem ser úteis ao desenvolvimento desta etapa e a fim de mostrar o diferencial que este agente terá em relação aos outros.
- Orientação de dois bolsistas (Pablo e Yuri) responsáveis pelo desenvolvimento do agente pedagógico afetivo e responsáveis por auxiliar outros bolsistas (Andressa e Laurent) no desenvolvimento dos módulos do jogo Città.
- Estudos de estratégias cognitivas e afetivas a serem utilizadas pelo agente pedagógico afetivo.
- Estudo da possibilidade de se trabalhar com a personalidade do estudante na interação com o agente pedagógico afetivo.
- Auxílio na organização das reuniões quinzenais do grupo de pesquisa.
- Participação nas discussões de desenvolvimento dos diálogos do agente pedagógico afetivo e de desenvolvimento do jogo e seus objetos.
- Participação nos testes do uso do jogo na escola com os estudantes, acompanhando a bolsista Maribel.
- Teste de usabilidade do jogo com os bolsistas desenvolvedores.
- Escrita de artigos para eventos científicos.

3. Referências

AXT, M. et al. Maga Vitta: conversational ecological agent in an interactive collective construction environment for basic education. In VICARI, Rosa; JAQUES, Patricia A. (Orgs.). **Agents-based Tutoring Systems for Affective and Cognitive Modelling**. Ed. Idea, 2008, chap. IV.

BAKHTIN, M. **Estética da criação verbal.** São Paulo: Martins Fontes, 1997.

BAKHTIN, M. **Marxismo e Filosofia da Linguagem.** São Paulo: Hucitec, 1981.

BERCHT, Magda. **Em Direção a Agentes Pedagógicos com Dimensões Afetivas**. Tese (Doutorado em Ciência da Computação). Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2001.

BERGER, Alexandra; PESTY, Sylvie. **Towards a Conversational Language for Artificial Agents in Mixed Community.** In Fourth International Central and Eastern European Conference on Multi-Agent Systems CEEMAS'05, Springer Verlag, Budapest, 2005, pages 31-40.

BURLESON, Winslow. **Affective Learning Companions:** Strategies for Empathetic Agents with Real-Time Multimodal Affective Sensing to Foster Meta-Cognitive and Meta-Affective Approaches to Learning, Motivation, and Perseverance. MIT PhD Thesis, 2006.

CONATI, Cristina. **Probabilistic Assessment of User's Emotions in Educational Games.** Applied Artificial Intelligence, Philadelphia, v. 16, n. 7-8, 2002.

GOLEMAN, Daniel. **Emotional Intelligence**, New York: Bantam Books, 1995.

IZARD, Carrol E. Emotion-cognition relationships and human development. In: IZARD, C.; KAGAN, J.; ZAJONC, R.B. (Ed.). **Emotions, cognition, and behavior.** New York: Cambridge University Press, 1984. p. 17-37.

JAQUES, Patricia A. **Using an Animated Pedagogical Agent to Interact Affectively with the Student.** 2004. Tese (Doutorado em Ciência da Computação). Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2004.

LEONAHRDT, M.; VICARI, R.; PESTY, S. **Enhancing Affective Communication in Embodied Conversational Agents.** In COLIBRI - Colóquio em Informática, 2009, Bento Gonçalves. [Colibri] - Anais do Colóquio em Informática: Brasil / INRIA, Cooperações, Avanços e Desafios, 2009.

LESTER, J. et al. **Cosmo: A life-like animated pedagogical agent with deictic believability.** In: IJCAI97 Workshop on Animated Interface Agents: Making them Intelligent. **Proceedings.** Nagoya, 1997.

McQUINGGAN, Scott W.; LESTER, James C. **Modeling and evaluating empathy in embodied companion agents.** In International Journal of Human-Computer Studies/International Journal of Man-Machine Studies. 65 (4): 348-360, 2007.

MÜLLER, D. N. ET AL. **Cidades virtuais como ambiente educacional colaborativo**. In: Digital Games 2008. **Proceedings.** Porto/Portugal, 2008.

ORTONY, A.; CLORE, G. L.; COLLINS, A. **The cognitive structure of emotions**. Cambridge: Cambridge University Press, 1988.

PELACHAUD, Catherine. Contextually Embodied Agents. In MAGNENAT-THALMANN, N. THALMANN, D. (eds.). **Deformable Avatars**. Kluwer Publishers, 2001.

PICARD, Rosalind. **Affective Computing.** Cambridge: MIT Press, 1997.

SCHERER, Klaus. Appraisal Theory. In DALGLEISH, T.; POWER, M. (ed.). **Handbook of Cognition and Emotion.** New York: John Wiley & Sons, 1999.

VILHJALMSSON, H. et al. **The behavior markup language: recent developments and challenges.** In 7tg International Conference on Intelligent Virtual Agents: Electronic Notes in Artificial Intelligence. **Proceedings.** Berlin: Springer, 2007. p. 90-111.

VYGOTSKY, Lev S. **Thought and Language. Cambridge**. MA: MIT Press, 1962.

WEIZENBAUM, Joseph. **ELIZA - A Computer Program for the Study of Natural Language Communication between Man and Machine**. Communication of the ACM. 25th Anniversary Issue. 26 (1): 23-28. Cambridge, MA. USA, 1983.

YUNG, Hsin. I. **Effects of animated agent with instructional strategies in facilitating student achievement of different educational objectives in multimedia learning.** In Journal of Educational Multimedia and Hypermedia, 18(4), 453-466. Chesapeake, VA: AACE, 2009.

RELATÓRIO DO PROJETO CITTÀ COSMOPOLITA

Projeto processo CNPq: *551846/2011-9*

Relatório de Atividades do Bolsista: *Alexsandro Rosa dos Anjos*

Bolsista processo CNPq: *385529/2011-2*

Coordenador: Daniel Nehme Müller

Período do Relatório: *dezembro/2011 a maio/2012*

Plano de trabalho:

Acompanhamento dos bolsistas estudantes no projeto, análise e desenvolvimento dos sistemas envolvidos no projeto.

---

1. Introdução

Neste relatório são descritas as atividades realizadas que contribuíram para o desenvolvimento do agente do jogo Città, parte integrante do projeto Cosmopolita além das atividades gerais no projeto.

1.1. Visão geral do Projeto

Desenvolvimento de um software gráfico interativo (jogo) para apoiar pedagogicamente na construção de conhecimento das crianças de 4º. e 5º. Ano do ensino fundamental, no que se refere ao desenvolvimento de capacidades que lhes permitam pensar-construir-organizar uma cidade virtual a partir da realidade infantil e seus conhecimentos incorporados através dos conteúdos oferecidos no aprendizado dos sistemas que regulam uma cidade e suas relações.

2. Atividades realizadas

- Acompanhamento dos bolsistas responsáveis pelo desenvolvimento dos módulos do jogo Città.
- Organização do fluxo de informações do projeto, utilizando para isso ferramentas de gerenciamento de projetos.
- Configuração do servidor do projeto, na sala do LELIC.
- Atualização dos sistemas dos computadores do LELIC.
- Apoio às demais atividades dos bolsistas.

RELATÓRIO DO PROJETO CITTÀ COSMOPOLITA

Projeto processo CNPq: *551846/2011-9*

Relatório de Atividades do Bolsista: *Vinícius Brum Bertoldo*

Bolsista processo CNPq: *350705/2011-9*

Coordenador: *Daniel Nehme Müller*

Período do Relatório: *dezembro/2011 a novembro/2012*

Plano de Trabalho:

Programação dos módulos de conexão e máquina de jogo.

---

1. *Introdução*

Neste relatório são descritas as atividades realizadas para o desenvolvimento do jogo Città, parte do projeto Cosmopolita.

Fiquei encarregado pela atualização do Città da versão Java para a nova concepção em Blender.

1.1. Visão do Projeto

Desenvolvimento de um software gráfico interativo (jogo) para apoiar pedagogicamente na construção de conhecimento das crianças de 4º. e 5º. ano do ensino fundamental, no que se refere ao desenvolvimento de capacidades que lhes permitam pensar-construir-organizar uma cidade virtual a partir da realidade infantil e seus conhecimentos incorporados através dos conteúdos oferecidos no aprendizado dos sistemas que regulam uma cidade e suas relações.

2. *Desenvolvimento*

2.1. Cittá versão Java

No início do projeto vimos que o Cittá na versão java estava estagnado na questão do desempenho. Além disso, sua manutenção estava cada vez mais complexa.

Tanto o uso de modelos gráficos como sua transformação no ambiente gráfico exigem muito esforço de programação para otimização e desenvolvimento de novas funcionalidades ou correções. Assim, no início do projeto decidimos mudar a plataforma de desenvolvimento, e o Blender foi escolhido por ter todas as características que precisávamos para o Città.

Continuamos a dar suporte para a versão java usada nas escolas. Em paralelo começamos a estudar as funcionalidades do Blender e reescrever todo o Città nesta nova plataforma.

Para avaliar a versão java fomos, em 07/08/2012, à cidade de Estrela (RS) para fazer uma oficina com professores que atuavam no Projeto CIVITAS e que iam iniciar o uso do Città com seus alunos. Eles tinham disponíveis um netbook para cada professor, nos quais o Città já havia sido instalado pela equipe de informática da prefeitura. Lá anotamos algumas melhorias para o Città java. Geramos outra versão java com melhorias após a visita.

2.2. Cittá versão Blender

O Cittá versão Blender está dividido em partes que são o Blender em si e o recurso da Game Engine, os scripts em Python, objetos de jogo e blocos de lógica da Blender Game Engine.

Cenas:

O Blender disponibiliza várias cenas nas quais é possível modelas os objetos 3D. Cada cena possui uma ou mais camadas (layers) e blocos de lógica (logic brics). No Cittá estamos utilizando três cenas básicas que são:

Gameplay:

Nesta cena está modelado o jogo em si no qual estão modelados na primeira camada o terreno e os objetos de controle. Na segunda camada está modelado o terreno e na terceira camada estão os objetos do jogo como casa e demais objetos que o jogador coloca no mapa do terreno durante o gameplay.

Blocos de lógica:

Os blocos de lógica da cena Gameplay são baseados no objeto Gerenciador e começam no estado 2 para uma única execução dos blocos de lógica que a seguir passam para o estado 1.

No estado 2 estão colocados os seguintes blocos de lógica:

Para o objeto Gerenciador:

1) Sempre executa o código initialize.py mostrado a seguir:

```
import bge
import pickle
import sqlite3
#Script de inicialização

bge.render.showMouse(True)
#Pega o controlador
cont = bge.logic.getCurrentController()
```

```
own = cont.owner
#Cria a matriz de jogo em escopo global
bge.logic.matrizJogo = [[0]*100 for x in range(100)]
#Muda o estado do objeto
own.state = 1
if bge.logic.operacaoInicial == "salvar":
    sce = bge.logic.getCurrentScene()
    #conecta no banco:
    print ('Antes de conectar')
    mapa = "mapas\\" + bge.logic.mapaAtual + ".sqlite"
    conn = sqlite3.connect(mapa)
    print ('Depois de conectar')
    #Cria objeto cursor para inserir os dados via SQL
    c = conn.cursor()
    #Cria a tabela
    c.execute("DROP TABLE if EXISTS mapa")
    c.execute('''CREATE TABLE mapa
                (nome text, posX real, posY real, rotacao real, pessoas real, agua real, energia real, alimento
real, lixo real, esgoto real)''')
    #Percorre todos os objetos que possuam o atributo "objeto" e que ja estejam postos no mapa (state >
1)
    for obj in sce.objects:
        print ("entrou na iteracao")
        if "objeto" in obj and obj.state > 1:
            print ("entrou na condicao")
            nome = obj.name
            #Converte x e y para valores de tiles
            #posX =int(obj.worldPosition[0]+50-0.5)
            #posY =int(obj.worldPosition[1]+50-0.5)
            posX = obj.worldPosition[0] + 50 + (0.5-((obj['tilesX']%2)/2))
            posY = obj.worldPosition[1] + 50 - (0.5-((obj['tilesY']%2)/2))
            print("PosX: ", posX)
            print("PosY: ", posY)
            rotacao =obj['rotacao']
            pessoas = 4
            agua = 4
            energia = 4
            alimento = 4
            lixo = 4
```

```
        esgoto = 3
        objeto = (nome, posX, posY, rotacao, pessoas, agua, energia, alimento, lixo, esgoto)
        c.execute("INSERT INTO mapa VALUES (?,?,?,?,?,?,?,?,?,?)", objeto)
    #SALVAR POS DA CAMERA ANTES DE COMMIT na tabela Cenario
    #commit e fecha a conexao
    conn.commit()
    c.close()
    conn.close()
    print("Objetos inseridos no banco de dados")
else:
    sce = bge.logic.getCurrentScene()
    #Abre o arquivo especificado em modo binário de leitura ("r = leitura", "b = modo binário")
    mapa = "mapas\\" + bge.logic.mapaAtual + ".sqlite"
    conn = sqlite3.connect(mapa)
    with conn:
        cur=conn.cursor()
        cur.execute("SELECT * FROM mapa")
        #Deve ler o conteúdo da tabela até o ultimo registro
        rows = cur.fetchall()
        for row in rows:
            nomeObj = row[0]
            objeto = sce.addObject(nomeObj, sce.objects['Cursor'])
            #Arruma a rotacao do objeto em X e a escala
            objeto.applyRotation([1.570796326794897,0.0,0.0], True)
            objeto.scaling = [0.5,0.5,0.5]
            #Passa a posicao do objeto no mundo
            objeto.worldPosition[0] = row[1] - 50
            objeto.worldPosition[1] = row[2] - 50
            objeto.localPosition[2] += 0.001
            #Passa a rotacao e ja aplica no objeto
            objeto['rotacao'] = row[3]
            objeto.applyRotation([0.0,1.570796326794897*(objeto['rotacao']/90),0.0], True)
            #Corrige o valor de tiles em X e Y conforme a rotacao
            if objeto['rotacao'] == 90 or objeto['rotacao'] == 270:
                valor = objeto['tilesX']
                objeto['tilesX'] = objeto['tilesY']
                objeto['tilesY'] = valor
            #Corrige a posicao do objeto conforme a rotacao
            objeto.worldPosition[0] -= (0.5-((objeto['tilesX']%2)/2))
```

```
        objeto.worldPosition[1] += (0.5-((objeto['tilesY']%2)/2))
        #Passa os demais atributos para o objeto
        objeto['pessoas'] = row[4]
        objeto['agua'] = row[5]
        objeto['energia'] = row[6]
        objeto['alimento'] = row[7]
        objeto['lixo'] = row[8]
        objeto['esgoto'] = row[9]
        objeto.state = 2
    #deve ler a posicao da camera antes de fechar
    #Fecha o banco
    cur.close()
    conn.close()
```

Este código de programação acima efetua a mudança para o estado 1 dos blocos de lógica.

2) Além deste bloco de lógica está definido um segundo bloco de lógica que adiciona a cena HUD.

No estado 1 estão colocados os seguintes blocos de lógica:

2.1) Sempre executa o código definido em always.py descrito abaixo:

```
import bge
import mathutils
#Obtem o controlador que roda o script
cont = bge.logic.getCurrentController()
#Obtem quem esta rodando o controlador (este
#proprio objeto)
own = cont.owner
#Obtem a cena
sce = bge.logic.getCurrentScene()
#Obtem a lista (vetor) de objetos da cena
obj = sce.objects
#Obtem os sensores de verificaçao de mouse
#sobre alguma coisa e clique com botao esquerdo
mouseOverAny = own.sensors["mouseOverAny"]
#Obtem os objetos Cursor e Camera
cursor = obj["Cursor"]
camera = obj["Camera"]
#Pega a posiçao no mundo 3D em que o mouse esta
```

```
hitPos = mouseOverAny.hitPosition
hitObj = mouseOverAny.hitObject
#Camera envia um raycast da posicao do mouse
#para o Chao
rayObj, rayHitPoint, rayHitNormal = camera.rayCast(hitPos, None, 125, "chao", 0, 1)
#Se houve raycast
if hitObj != None and cursor['ativado'] == True:
    #Passa as informacoes das coords x e y para
    #variaveis
    x = rayHitPoint[0]
    y = rayHitPoint[1]
    #Converte essas posicoes para pixels
    x = int(x+50.0)
    if x > 99:
        x = 99
    elif x < 0:
        x = 0
    y = int(y+50.0)
    if y > 99:
        y = 99
    elif y < 0:
        y = 0
    #Define a posicao do cursor conforme a
    #posicao do raycast do mouse, convertido
    #para tiles
    cursor.localPosition = mathutils.Vector((x-50+0.5, y-50+0.5, 0.005))
#Caso contrario, deve "esconder" o cursor
else:
    cursor.localPosition[2] = 1000
if own['construcao'] != "Nenhum" and own['constroi'] == False:
    objeto = sce.addObject(own['construcao'], cursor)
    objeto.applyRotation([1.570796326794897,0.0,0.0], True)
    objeto.scaling = [0.5,0.5,0.5]
    own['constroi'] = True
    if own['construcao'] != "Apagador":
        cursor.state = 2
```

2.2) Gerenciador de evento quando o cursor do mouse passa sobre os objetos executa o codigo always.py definido acima;

2.3) Teclas G para gravar as alterações do mapa e C para carregar o mapa salvo (ver se vai haver mudanças nestas funcionalidades);

2.3.1) A tecla G executa o script salvaMapaBD.py definido a seguir:

```
#Script para salvar os objetos de jogo em arquivo de texto
import bge
import sqlite3
#Sensor da tecla G
teclaG = bge.logic.getCurrentController().owner.sensors["tecla G"]
if teclaG.positive:
    sce = bge.logic.getCurrentScene()
    #conecta no banco:
    print ('Antes de conectar')
    mapa = "mapas\\" + bge.logic.mapaAtual + ".sqlite"
    conn = sqlite3.connect(mapa)
    print ('Depois de conectar')
    #Cria objeto cursor para inserir os dados via SQL
    c = conn.cursor()
    #Cria a tabela
    c.execute("DROP TABLE if EXISTS mapa")
    c.execute('''CREATE TABLE mapa
            (nome text, posX real, posY real, rotacao real, pessoas real, agua real, energia real, alimento real, lixo real, esgoto real)''')
    #Percorre todos os objetos que possuam o atributo "objeto" e que ja estejam postos no mapa (state > 1)
    for obj in sce.objects:
        print ("entrou na iteracao")
        if "objeto" in obj and obj.state > 1:
            print ("entrou na condicao")
            nome = obj.name
            #Converte x e y para valores de tiles
            #posX =int(obj.worldPosition[0]+50-0.5)
            #posY =int(obj.worldPosition[1]+50-0.5)
            posX = obj.worldPosition[0] + 50 + (0.5-((obj['tilesX']%2)/2))
            posY = obj.worldPosition[1] + 50 - (0.5-((obj['tilesY']%2)/2))
            print("PosX: ", posX)
            print("PosY: ", posY)
            rotacao =obj['rotacao']
```

```
        pessoas = 4
        agua = 4
        energia = 4
        alimento = 4
        lixo = 4
        esgoto = 3
        objeto = (nome, posX, posY, rotacao, pessoas, agua, energia, alimento, lixo, esgoto)
        c.execute("INSERT INTO mapa VALUES (?,?,?,?,?,?,?,?,?,?)", objeto)
    #SALVAR POS DA CAMERA ANTES DE COMMIT na tabela Cenario
    #commit e fecha a conexao
    conn.commit()
    c.close()
    conn.close()
    print("Objetos inseridos no banco de dados")
```

2.3.2) A tecla C executa o script carregaMapaBD.py definido abaixo:

```
#Script para carregar os objetos de jogo do arquivo binário
import bge
import pickle
import sqlite3
#Sensor da tecla C
teclaC = bge.logic.getCurrentController().owner.sensors["tecla C"]
if teclaC.positive:
    sce = bge.logic.getCurrentScene()
    #Abre o arquivo especificado em modo binário de leitura ("r = leitura", "b = modo binário")
    mapa = "mapas\\" + bge.logic.mapaAtual + ".sqlite"
    conn = sqlite3.connect(mapa)
    with conn:
        cur=conn.cursor()
        cur.execute("SELECT * FROM mapa")
        #Deve ler o conteúdo da tabela até o ultimo registro
        rows = cur.fetchall()

        for row in rows:
            nomeObj = row[0]
            objeto = sce.addObject(nomeObj, sce.objects['Cursor'])
            #Arruma a rotacao do objeto em X e a escala
            objeto.applyRotation([1.570796326794897,0.0,0.0], True)
            objeto.scaling = [0.5,0.5,0.5]
```

```
        #Passa a posicao do objeto no mundo
        objeto.worldPosition[0] = row[1] - 50
        objeto.worldPosition[1] = row[2] - 50
        objeto.localPosition[2] += 0.001
        #Passa a rotacao e ja aplica no objeto
        objeto['rotacao'] = row[3]
        objeto.applyRotation([0.0,1.570796326794897*(objeto['rotacao']/90),0.0], True)
        #Corrige o valor de tiles em X e Y conforme a rotacao
        if objeto['rotacao'] == 90 or objeto['rotacao'] == 270:
            valor = objeto['tilesX']
            objeto['tilesX'] = objeto['tilesY']
            objeto['tilesY'] = valor
        #Corrige a posicao do objeto conforme a rotacao
        objeto.worldPosition[0] -= (0.5-((objeto['tilesX']%2)/2))
        objeto.worldPosition[1] += (0.5-((objeto['tilesY']%2)/2))
        #Passa os demais atributos para o objeto
        objeto['pessoas'] = row[4]
        objeto['agua'] = row[5]
        objeto['energia'] = row[6]
        objeto['alimento'] = row[7]
        objeto['lixo'] = row[8]
        objeto['esgoto'] = row[9]
        objeto.state = 2
    #deve ler a posicao da camera antes de fechar
    #Fecha o banco
    cur.close()
    conn.close()
```

2.4) Teclas A, S, D e W movimentam a câmera sobre o cenário, sendo W para mover à frente, S para mover para trás, A para mover para a esquerda e D para mover para a direita.

2.5) Teclas Q e E que rotacionam a câmera sobre o cenário, sendo Q para rotacionar no sentido anti-horário e E para rotacionar no sentido horário.

Para o objeto HUD:

Nesta cena estão modelados os objetos que compõe o menu que é utilizado durante o jogo como o botão de seleção do objeto, casa por exemplo, para ser inserida no mapa do terreno durante o gameplay.

Nesta cena estão definidas quatro camadas:

1) Camada 1 está definindo o menu HUD em si, no qual são colocados os botões de seleção de objetos.

1.1) Blocos de lógica:

Sempre executa o script HUDinitialize.py que contém o seguinte código:

```
import bge
sce = bge.logic.getCurrentScene()
own = bge.logic.getCurrentController().owner
obj = sce.addObject("btConstruir", own)
obj.localPosition[0] = 2.65
obj.localPosition[1] = 1.35
obj.localPosition[2] = 0
obj.scaling = [0.225, 0.225, 1]
obj = sce.addObject("btTerreno", own)
obj.localPosition[0] = 2.65
obj.localPosition[1] = 0.85
obj.localPosition[2] = 0
obj.scaling = [0.225, 0.225, 1]
obj = sce.addObject("btEstrada", own)
obj.localPosition[0] = 2.65
obj.localPosition[1] = 0.35
obj.localPosition[2] = 0
obj.scaling = [0.225, 0.225, 1]
obj = sce.addObject("btApagar", own)
obj.localPosition[0] = 2.65
obj.localPosition[1] = -0.6
obj.localPosition[2] = 0
obj.scaling = [0.2, 0.2, 1]
own.state = 2
```

2) Na camada 2 estão modelados os botões que serão colocados sobre o menu 'HUD'.

2.1) Botão de construção do objeto casa:

É o primeiro botão da camada da cena no canto superior esquerdo, que executa, quando o mouse clicar no botão, o código do script createObject.py que contém o seguinte código:

```
import bge
#Referencia a cena de jogo, procurando pelo seu
#nome na lista de cenas ativadas
for c in bge.logic.getSceneList():
    if c.name == "Gameplay":
```

```
        cena = c
#Referencia o proprio objeto
own = bge.logic.getCurrentController().owner
#Obtem os sensores
mouseSobre = own.sensors["mouseOver"]
cliqueEsquerdo = own.sensors["leftButton"]
#Testa se o mouse se encontra por cima do objeto
#e se houve clique
if mouseSobre.positive:
    if own['estado'] != 2:
        own['estado'] = 1
    if cliqueEsquerdo.positive:
        #Referencia o objeto gerenciador da cena
        #de jogo
        gerenciador = cena.objects["Gerenciador"]
        #Altera o atributo do objeto gerenciador que
        #marca o objeto a ser criado
        gerenciador['construcao'] = own['construcao']
        if own['estado'] != 2:
            own['estado'] = 2
        else:
            own['estado'] = 1
            for obj in cena.objects:
                if ("objeto" in obj or obj.name == "Apagador") and obj.state == 1:
                    obj.endObject()
                    break
            gerenciador['constroi'] = False
            gerenciador['construcao'] = "Nenhum"
else:
    if own['estado'] != 2:
        own['estado'] = 0
```

3) Na camada 3 estão modelados outros botões de funcionalidades;

4) Na camada 4 estão definidos:

a)Menu:

Na cena chamada 'Menu' estão modelados os objetos que compõe o menu principal do jogo na tela de abertura (tela inicial do jogo). No nível 1 estão definidos os objetos que virão a conter na gameplay o menu de seleção de mapas;

Na camada 2 e 3 estão definidos os botões que compõe o menu de abertura do jogo como criar novo mapa ou abrir um mapa existente;

Na camada 4 estão definidos os objetos que conterão as telas de ajuda, créditos e "o que é?"

b)Blocos de lógica:

A Blender Game Engine (BGE) utiliza-se de blocos de lógica (logic brics) nos objetos das cenas para realizar diversas funcionalidades durante a game loop (looping do jogo). Estes blocos de lógica são divididos em vários níveis ou estados.

c)Banco de dados:

O banco de dados utilizado é o SQLite onde seus artquivos são salvos no diretório 'maps' na raiz do jogo Cittá.

As tabelas utilizadas e seus atributos são descritos nos scripts de criação de tabelas abaixo:

d)Tabela Mapa:

CREATE TABLE "mapa" ("nome" TEXT, "tilesX" INTEGER, "tilesY" INTEGER, "posX" INTEGER, "posY" INTEGER, "rotacao" INTEGER, "pessoas" INTEGER, "agua" INTEGER, "energia" INTEGER, "alimento" INTEGER, "lixo" INTEGER, "esgoto" INTEGER)

Tabela feita para salvar os objetos do cenário como casas, prédios, estradas, etc.

e)Tabela 'cenario':

CREATE TABLE "cenario" ("rowID" INTEGER, "posCamX" FLOAT, "posCamY" FLOAT, "posCamZ" FLOAT, "rotacaoCam" FLOAT, "tempo" INTEGER)

Tabela feita para salvar a posição da câmera e o tempo de jogo em segundos.

Devemos implementar ainda uma tabela para o terreno, salvando os vértices do terreno com relação a sua posição X, Y e Z.

f) INPUT DE TEXTO:

Há essencialmente duas formas de se fazer o input de texto na BGE. Uma delas é utilizando apenas os blocos de lógica, sendo esta uma forma mais simples, porém menos customizável. A segunda é por meio de scripting, que é a forma usada no Città.

Para tanto, deve-se adicionar à cena um objeto do tipo "Text". Isso pode ser feito na aba "Add".

O próximo passo é criar uma Game Property (equivalente a um atributo, uma variável) de string nesse objeto. Nas versões mais atuais do Blender, os objetos de texto já possuem uma opção específica para isso: "Add Text Game Property", bastando apenas selecionar essa opção. Lembrando que as Game Properties são exibidas na interface "Logic Editor".

Agora, cria-se um sensor para o teclado. Deve-se selecionar a caixa "All Keys" do sensor para que ele seja ativado com qualquer pressionar de tecla, de resto, deixaremos o script tomar conta.

Esse sensor terá de ser ligado à um controlador do tipo Python, para rodar o script. Abaixo, segue o script que foi feito para o input de nome do mapa do Città:

```
import bge
cont = bge.logic.getCurrentController()
own = cont.owner
#Obtem o sensor de teclado
sensor = cont.sensors["teclado"]
#Verifica cada evento de teclado possivel
for tecla, estado in sensor.events:
    #Confere se a tecla sendo verificada foi pressionada
    if estado == bge.logic.KX_INPUT_JUST_ACTIVATED:
        string = bge.events.EventToString(tecla)
        print("Tecla Pressionada: ", string)
        print("Texto: ", own["Text"])
        print("Estado: ", estado)
        tamanhoTexto = len(own["Text"])
        if tamanhoTexto < 16:
            #Conferindo se a tecla corresponde a:
            #Letra
            if string[1] == "K":
                own["Text"] += string[0]
            #Numero
            elif string[0] == "P" and string[2] == "D":
                own["Text"] += string[3]
            #Espaco
            elif string == "SPACEKEY":
                own["Text"] += "_"
        #Backspace
        if string == "BACKSPACEKEY":
            tamanhoTexto = len(own["Text"])
            if tamanhoTexto >= 1:
                texto = own["Text"]
                ultimoCaracter = texto[tamanhoTexto-1]
                own["Text"] = texto.rstrip(ultimoCaracter)
                print("Tamanho Texto: ", tamanhoTexto)
```

No exemplo acima, o código percorre todos os eventos possíveis do sensor, pois os valores armazenados em "sensor.events" não são do tipo string ou integer. Dessa forma, se o estado da tecla verificada corresponder a "bge.logic.KX_INPUT_JUST_ACTIVATED", ele deve continuar a verificação com aquela tecla.

A variável "string" recebe o nome pelo qual a tecla é identificada. Por exemplo, se pressionássemos a tecla "E", string seria igual a "EKEY", pois é por esse nome que o evento dessa tecla é reconhecido, e a função bge.events.EventToString(tecla) retorna esse nome como um valor do tipo string.

Porém ocorre um problema, dessa forma temos uma palavra e não apenas o caractere que queremos para acrescentar em nosso texto. Mas podemos observar os padrões da forma como os caracteres são reconhecidos e extrair somente a letra que queremos deles. Exemplo: todos os caracteres de letras do alfabeto seguem o padrão "(letra)" + "KEY" (AKEY, BKEY, CKEY...). No código de exemplo, o que se faz é verificar se a segunda letra da variável string é a letra "K" (if string[1] == "K"), e se for, acrescentamos a primeira letra de string à variável "Text" (aquela Game Property do tipo String que criamos em nosso objeto texto). O mesmo pode ser feito analisando-se os padrões dos outros tipos de tecla, como números e teclas especiais. O bom dessa forma é que podemos impedir que alguns caracteres sejam usados

Notas de versão:

A versão atual está apresentando um menu de jogo, cria mapas de jogos novos e abre mapas previamente salvos.

Referências:

Link para documentação da API:

http://www.blender.org/documentation/blender_python_api_2_61_0/

Esse tópico tem diversos guias e tutoriais sobre o Blender Game Engine:

http://blenderartists.org/forum/showthread.php?240729

Publicações em Eventos

Editor de Cidades Civitas Città

(Publicado no CBIE/MPIE – Trilha 1: MSIE – 2013)

Disponível em:
http://www.br-ie.org/pub/index.php/wcbie/article/view/2668

Daniel Nehme Müller[1], Laurent Le Avner Quevedo[2], Andressa Cruz Nepomuceno[3], Luciano Soares[4], Juarez Dubois da Costa[5], Maribel Suzane Selli[6] e Margarete Axt[6]
Conexum Ltda.[1]
Curso de Jogos Digitais, Universidade do Vale do Rio dos Sinos (UNISINOS)[2]
Curso de Ciência da Computação, Universidade do Vale do Rio dos Sinos (UNISINOS)[3]
Curso Produção Multimídia Digital, Universidade do Sul de Santa Catarina (UNISUL)[4]
Laboratório de Estudos em Linguagem, Interação e Cognição, Universidade Federal do Rio Grande do Sul (UFRGS)[5]
Programa de Pós Graduação em Informática na Educação, Universidade Federal do Rio Grande do Sul (UFRGS)[6]

daniel@conexum.com.br, laurentleavnerquevedo@gmail.com,andressacruzn@gmail.com, ls.druid@gmail.com, juarezdc@hotmail.com, mselli7@gmail.com, maaxt@ufrgs.br

Abstract. *The city editor Città develops the digital side of inventing cities promoted by Civitas Project, Faculty of Education / UFRGS. Citta is a free GPL software which allows the construction of a three dimensional city using Java and SQLite database. The city is monitored by an intelligent agent that uses Bayesian inference as well as recognition and speech synthesis. In this paper is being presented the expansion of the Città for use of multiplayer, text/voice chat and the rewriting the system in Blender, using multiplayer features and web access.*

Resumo. *O editor de cidades Città desenvolve o lado digital de invenção de cidades promovido pelo Projeto CIVITAS, da Faculdade de Educação/UFRGS. Città é um software livre GPL que permite a construção de uma cidade tridimensional usando Java e banco de dados SQLite. Tem como contexto de aplicação turmas de quarto ano de Escolas Municipais de Ensino Fundamental de um Município da Região do Vale do Taquari no Estado do Rio Grande do Sul. Neste artigo estarão sendo apresentadas, além da versão Java, também sua versão reescrita do sistema em Blender, com a exploração de recursos multijogador, chat por voz e texto, além do seu uso potencial em sistemas android e navegadores web.*

Introdução

O *Projeto Città Cosmopolita - Simuladores de Cidades em Redes*[19] visa à pesquisa e ao desenvolvimento dos módulos de comunicação, interação e geolocalização do jogo computacional educacional e simulador de cidades Città [Oliveira et al, 2009], que vem sendo desenvolvido pelo LELIC (Laboratório de Estudos em Linguagem, Interação e Cognição da Faculdade de Educação da UFRGS) em parceria com a empresa CONEXUM. Città é um software livre com licença GPL com fontes disponibilizados no SourceForge[20].

Os novos módulos concebidos para o Città contemplam a interação entre as cidades virtuais construídas pelos estudantes, o uso do sistema na web e em dispositivos móveis, a comunicação com o assistente inteligente Maga Vitta baseado na análise do estado afetivo

[19]Projeto financiado pelo CNPq – Conselho Nacional de Desenvolvimento Científico e Tecnológico.
[20]http://sourceforge.net/projects/civitas/

(emoção no texto e na voz), e na geolocalização das cidades. O desenvolvimento da assistente inteligente está sendo realizado em parceria ainda com pesquisadores da UNISC e UNIPAMPA, e a plataforma web com geolocalização está sendo desenvolvida com parceria com pesquisadores da UNEB. O interesse da CONEXUM está no desenvolvimento de novos sistemas para dispositivos móveis, visando interfaces facilitadas por assistentes. Ao longo do projeto as escolas atendidas pelo LELIC através do Projeto *CIVITAS – Cidades Virtuais com Tecnologias para Aprendizagem e Simulação* [Axt e Martins, 2008] terão um benefício direto com o uso do jogo em suas atividades pedagógicas.

Neste artigo serão abordados a contextualização dos jogos de simulação no ensino (seção 2), a apresentação geral do Città (seção 3), o agente inteligente (seção 4) e os novos recursos Blender que estão sendo explorados na nova versão (seção 5). Por fim, algumas considerações finais sobre o desenvolvimento e futuras aplicações derivadas do projeto.

Jogos de simulação no ensino

Os jogos digitais já vêm sendo pesquisados como recursos de ensino há muitos anos, em diversas disciplinas. Podemos encontrar jogos específicos no ensino de Biologia [Guterres et al, 2003], Física [Hornes et al, 2009] ou Geografia [Pereira et al, 2011]. Ainda em Geografia, encontramos o uso de jogos comerciais, como o SimCity [Tavares, 2012] ou livres, como o LinCity [Albuquerque, 2009]. Este último foi experimentado em várias disciplinas, mas sem muito sucesso, devido à falta de um planejamento adequado com o software.

Progressivamente os jogos de simulação, principalmente digitais, estão evoluindo rapidamente. Os pesquisadores que investigam o uso de jogos de interação e imaginação como o RPG digital aplicado a disciplinas [Araújo e Simon, 2008], jogos comerciais como *The Sims* [Oliveira, 2011], jogos para web [Tarouco et al, 2004] são unânimes em considerar a importância do preparo dos professores e o bom planejamento da aula como pontos de sucesso do uso de qualquer jogo digital. Infelizmente jogos educacionais desenvolvidos com professores e estudantes ainda são raros, sendo encontrados jogos de entretenimento adaptados a situações de sala de aula.

O jogo de simulação Città

Dentro de contextos de jogos de simulação, Città é a versão digital do simulacro da cidade que ocorre na sala de aula com alunos do 4º ano do ensino fundamental atendidos pelo Projeto CIVITAS. As crianças reinventam cidades imaginárias onde eles são cidadãos e tornam-se responsáveis por todos os setores da sociedade.

O que antes era feito apenas com papelão, isopor ou argila, agora torna-se também digital. A cidade construída pelas crianças fica ainda sobre uma mesa no canto da sala, mas a turma pode construir muitas outras cidades, sem ocupar espaço físico nem ser desgastada pelo manuseio, agora no meio digital.

Através do Città, a criança constrói prédios e vias, modela terrenos, troca cores e apaga os objetos que não servem mais. Também está sendo testado um agente inteligente para questionar a criança sobre a sustentabilidade da cidade (ver seção 4). Além disso, a cidade poderá ser construída em rede e ainda através do navegador web (ver seção 5).

O Città está constantemente sendo reprojetado, uma vez que a equipe de desenvolvimento trabalha integrada com as acadêmicas de pós-graduação da Faculdade de Educação e do Centro Interdisciplinar de Novas Tecnologias na Educação (CINTED) da UFRGS. Essas acadêmicas são integrantes do LELIC, e assessoram as professoras nas escolas atendidas pelo CIVITAS, onde Città é elaborado numa atitude de escuta e análise deste processo.

Nesse sentido, o Projeto Città Cosmopolita é o resultado de um ciclo de dez anos de

estudos e ações contínuas executadas no Projeto CIVITAS. Por esse motivo a releitura e reelaboração do Città é uma prática comum nas equipes destes projetos. Nas seções a seguir tem-se mais uma versão otimizada a partir do diálogo entre o mundo computacional e o pedagógico.

As primeiras experimentações/impressões do Città em sala de aula

Depois de algumas apresentações pontuais que ocorreram durante o desenvolvimento do Città na versão Java tivemos a primeira oficina para exploração e reconhecimento do simulador de cidades com um grupo de professoras do quarto ano de um dos municípios em que desenvolvemos o Projeto Civitas[21].

No decorrer dessa experimentação nos foi possível perceber alguns detalhes importantes que precisavam ainda ser ajustados no simulador como a cor do objeto praça, o zoom e a colocação de alguns objetos que acusavam sobreposição falando mais tecnicamente.

Por outro lado percebemos o encantamento e a empolgação das professoras ao experimentarem o simulador e suas possibilidades fazendo vários questionamentos sobre como poderiam levar para seus alunos o simulador inserindo-o no cotidiano da sala de aula como mais uma ferramenta que poderia auxiliar na construção da cidade imaginária/virtual da turma.

Sanados os pontos apontados pelas professoras, uma nova versão do simulador foi disponibilizada para as escolas e em seguida foi apresentado aos alunos que fizeram várias explorações no simulador.

Segundo o relato que obtivemos os alunos acharam extremamente importante a presença dos netbooks que fazem parte dessa proposta na sala de aula para a realização de pesquisas e da criação da cidade imaginária.

Com pressupostos para a utilização dessa ferramenta temos alguns critérios que consideramos muito importantes para a formação e aprendizagem dos alunos que têm acesso ao Città, entre eles, a cooperação que acontece desde quando os alunos fazem uso da mesma máquina em grupos de dois ou três discutindo os objetos que vão fazer parte de sua cidade; o respeito pela opinião e entendimento do outro sobre as coisas quando precisam negociar e abrir mão de uma intenção que possuíam em prol da opinião do colega; a solidariedade e o respeito à diversidade quando pensam na importância de construir uma cidade que acolha os diferentes e seja acessível a todas as pessoas que lá decidirem morar; a sustentabilidade ao planejarem uma cidade economicamente viável e ambientalmente sustentável evitando desmatamento, poluição da natureza e dos mananciais de água, desperdício de energia; ainda sobre o quesito respeito podemos destacar a preocupação com o espaço do outro e suas necessidades evitando invadir o terreno do vizinho ou mesmo construir em espaços que não são permitidos porque podem colocar a vida da cidade em risco ou comprometer a saúde dos moradores.

Entre esses pressupostos apontados buscamos durante a utilização do simulador orientar as professoras e professores para que estejam sempre atentos e acompanhando as ações das crianças para que elas percebam a importância de respeitar as regras estabelecidas e, com isso, viabilizando a vida e o convívio na cidade e, sempre que houver desentendimentos ou discordâncias que possam parar para pensar as causas que levaram a esse fato buscando juntos, ouvindo a opinião e posicionamento uns dos outros resolver o impasse da melhor forma.

[21]CIVITAS: Cidades Virtuais Tecnologias para Aprendizagem e Simulação. Projeto de Pesquisa e Formação CNPq, FINEP, UFRGS, SESU-MEC. O Civitas é desenvolvido na Educação Infantil e até os 5º anos do Ensino Fundamental de escolas municipais da cidade de Estrela, nesse caso especificamente, mas já foi desenvolvido também em outros municípios desde sua criação como Cruzeiro do Sul, Mato Leitão, Venâncio Aires e Sobradinho, todas situadas no Rio Grande do Sul. Professores e alunos interagem cooperativamente na tomada de decisões e no planejamento de uma cidade virtual, bem como na definição de normas de convivência urbana em seus aspectos socioambiental, histórico-cultural, ético-estético e político-econômico, gerando subsídios para reflexão crítica, tanto dos conteúdos curriculares quanto dos valores implicados.

A partir das experimentações realizadas em sala de aula com o simulador e os registros das crianças percebemos o quanto o mesmo contribuiu para o enriquecimento dos estudos da cidade e das práticas em sala de aula.

Os alunos demonstraram grande interesse e criatividade nas experimentações realizadas criando cidades com os mais diversos temas entre eles Cidade dos Fantasmas e Cidade dos Robôs. Além da produção da cidade imaginária física confeccionada com sucatas e da cidade virtual idealizada no simulador utilizaram outras ferramentas digitais como máquina fotográfica, gravador de voz e interação virtual através do Forchat contanto para as outras turmas o que vinha acontecendo na cidade que haviam criado e recebendo informações dos colegas sobre os fatos ocorridos em suas cidades.

As primeiras impressões expressas tanto pelos alunos como pelas professoras nos motivaram muito, pois além explorarem todos os recursos do simulador nos sugeriram vários acréscimos para a versão que estamos programando em Blender.

Módulo de Interface

A interface do Città foi totalmente remodelada e seus objetos tridimensionais foram redesenhados frente à versão anterior. Com base nos relatos de experiências com as crianças das escolas nas quais o Città foi apresentado, percebeu-se que havia necessidade de melhorar a interface gráfica primeiramente para facilitar a compreensão.

Após diversas discussões ao longo do tempo, foi decidido que além de simplificar a interface, haveria necessidade também de uma revisão e renovação geral da interface gráfica. Foi adotada a ideia de tornar a navegação dos menus clara através de imagens apenas, já que palavras muitas vezes complicadas não eram compreendidas pelas crianças, o que restringia às crianças algumas opções de maneira inconsciente.

O menu agora está menor e mais intuitivo, na lateral direita da tela em transparência (veja figura 1). Assim, ocupa menos espaço de jogo e abre em cascata de acordo com a seleção dos botões principais. A figura 2 ilustra um dos itens simplificados na nova interface com a opção para sair do jogo, que antes necessitava dois passos (à esquerda da figura) e agora é em apenas um (à direita).

Figura 1 – Menu na versão anterior (à esquerda) e remodelado (à direita).

Figura 2 – Comparação das interfaces: à esquerda, interface antiga; à direita, interface nova.

Ao implementar a nova interface, notou-se as limitações que a falta de conhecimento sobre a engine gráfica e bibliotecas não recentemente atualizadas iriam implicar em futuras atualizações mais complexas como animações. Foi considerada a utilização de uma engine completamente diferente, como o Blender (veja seção 5).

Banco de Dados

O banco de dados livre SQLite[22] foi escolhido por ser de fácil distribuição dentro com os pacotes Java do Città. Durante seu desenvolvimento, foi discutido um método mais otimizado de gravação de mapas, que também facilitaria a implementação de um histórico de eventos também mais otimizado. Este novo método consiste em um arquivo de banco de dados para cada mapa, contendo um registro do estado atual do mapa e um registro de eventos.

O registro do mapa serve para uma rápida reconstrução do mundo de jogo, independente de quantas alterações foram feitas até agora naquele mapa. O registro de eventos grava a "história" daquele mapa, indicando a data de jogo, o tipo do evento (construção/destruição/"biografia da construção") e um texto, descrevendo o evento em detalhes.

Foi definido também o critério de salvamento dos mapas. Os mapas são salvos assim que o jogador decide sair do mapa atual, não há necessidade de navegar menus ou apertar outros botões.

O agente inteligente Maga Vitta

Durante a construção da cidade virtual, o aluno conta com auxílio de um agente que observa suas ações no ambiente virtual. O agente inteligente, encorporado como uma personagem chamada Maga Vitta, não intervém na construção da cidade, mas está sempre atenta para indagar, aconselhar, informar e provocar reflexões sobre os processos de construção da cidade.

Além disso, Maga Vitta é um agente que possui como padrão interno estar eternamente preocupado com a ecologia: ao memorizar ações contra o equilíbrio ecológico, adverte sobre possíveis tragédias ambientais. Em sua interação com o jogo Città, Maga Vitta tem um motor de inferência que trabalha em conjunto com um módulo de conversação.

Inferência, diálogo e estados afetivos

A agente Maga Vitta é a consciência do jogo Città, promovendo a regulação do sistema ecológico. A sua proposta não é realizar a orientação direta, mas provocar ações internas no aluno de forma a fazê-lo se preocupar com a situação indicada no momento do jogo. Dependendo da gravidade da situação, indicada pelo nível de equilíbrio da variável de ambiente, e ainda da prioridade do

[22] http://www.sqlite.org

elemento que está sendo considerado, é indicado ao módulo de diálogo a importância e nível de expressividade que o agente terá que executar.

As variáveis de ambiente são relativas aos produzíveis e consumíveis na cidade virtual, correspondendo ao quanto é produzido ou gasto por cada objeto pertencente à ao jogo. Os objetos criados para composição da cidade virtual são: casa, prédio, escola, hospital, igreja, fábrica, prefeitura, mercado, fazenda, árvore, ETA (Estação de Tratamento de Água), usina eólica, aterro sanitário, ETE (Estação de Tratamento de Esgoto), UTLH (Unidade de Tratamento de Lixo Hospitalar).

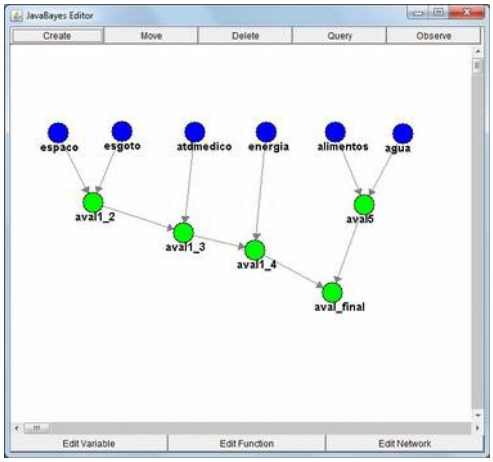


Figura 3 – Projeto de inferência no JavaBayes.

Uma rede bayesiana projetada para tomada de decisão das variáveis de ambiente, as quais têm os valores aplicados nos nodos superiores e a resposta é obtida no nodo de nível mais baixo, conforme mostra a figura 3, obtida a partir da modelagem realizada com o sistema de software livre JavaBayes[23]. Este software livre foi incorporado ao projeto Java do Città, obtendo-se sucesso no cálculo das variáveis de ambiente e obtenção das decisões necessárias às intervenções da Maga Vitta.

Atualmente está sendo trabalhado o uso do raciocínio bayesiano juntamente com o esquema de Raciocínio Baseado em Casos (RBC) elaborado para o diálogo da Maga Vitta. Como parâmetro, utiliza-se o trabalho de [Pavón et al, 2009], que o qual aborda a inferência bayesiana na busca do conjunto de soluções dentro de um sistema RBC.

Desta forma, foram pesquisadas implementações de RBCs em software livre, para composição ao projeto do jogo Città. A plataforma RBC que convergimos para integração ao nosso projeto foi a *jCOLIBRI – CBR Framework*[24]. Na fase atual de desenvolvimento está ocorrendo a integração do JCOLIBRI com o chat de diálogo entre a criança e a Maga Vitta.

Além dos módulos de inferência e diálogo, a Maga Vitta também está projetada para ter capacidade emotiva e de personalidade [Konzen et al, 2011]. Para tanto, está sendo estudado o modelo de emoções OCC, [Ortony, Clore e Collins, 1988] e o modelo *Big Five*, de representação de traços de personalidade [John, 1999]. Como este módulo está em iniciando seu desenvolvimento, ainda ocorrerá o aprofundamento destes e outros modelos para posterior implementação. Na figura 4 pode-se observar um estudo gráfico dos diferentes estados emocionais da Maga Vitta.

[23] http://sourceforge.net/projects/jbnc
[24] http://sourceforge.net/projects/jcolibri-cbr

Figura 4 – Estados emocionais da Maga Vitta.

Reconhecimento e síntese de fala

O módulo de voz consiste no reconhecimento de voz da criança usuária do Città e a síntese do agente inteligente Maga Vitta.

O reconhecimento de voz foi continuado com o aperfeiçoamento do modelo lingüístico e teste de palavras para reconhecimento, sempre dentro do modelo dos sistemas de software livre *HTK*[25] (treinamento estocástico) e *Julius*[26] (reconhecimento de voz). Neste sentido, foram feitas interações com bolsistas do Projeto Comunica[27], do Instituto de Informática da UFRGS, e com membros do Projeto FalaBrasil[28] da UFPA, no sentido de otimizar o reconhecimento.

A integração do módulo de reconhecimento de voz através do Julius ocorreu parcialmente através da inserção de botões de controle de gravação de voz ("Começar gravação" e "Parar gravação") na interface do chat com a Maga. No entanto, o módulo de reconhecimento de voz não está funcional em conjunto com os outros módulos. Foi cogitado o uso de uma ferramenta semelhante ao *grep* (GNU Project) para buscar as palavras-chave capturadas pelo Julius e repassá-las ao assistente do jogo (veja figura 5).

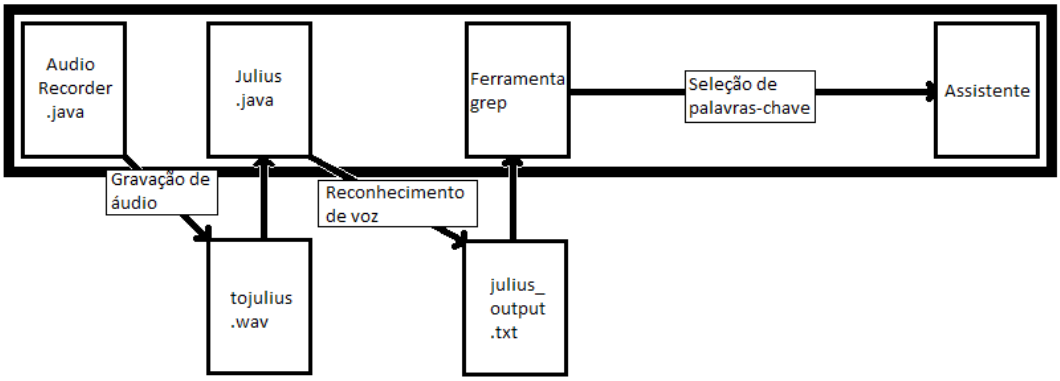


Figura 5 – Sequência de chamadas do Julius no Città.

Quanto à síntese de fala, utilizamos o sistema de software livre *MaryTTS*[29], também com apoio da equipe do Projeto Comunica, que desenvolveu um modelo acústico com voz feminina, adequada à Maga Vitta. MaryTTS é uma solução completa e configurável para síntese, incluindo a construção do modelo acústico para voz, o léxico de fonemas e a parametrização de itens como a prosódia (entonação) e velocidade de leitura (duração). Ele utiliza um padrão de serviço internet (WebService) para receber um texto e enviar um arquivo de áudio. Ao iniciar o Città é necessário

[25] http://htk.eng.cam.ac.uk
[26] http://julius.sourceforge.jp/en_index.php
[27] http://comunica.conexum.com.br
[28] http://www.laps.ufpa.br/falabrasil
[29] http://mary.dfki.de

ativar o servidor (*maryserver*) e, quando necessário, apenas solicitar o arquivo de áudio para determinada frase. Para tanto, foi adicionada uma chamada de sistema para o servidor, ao início do Città, e uma classe Java (*MaryClientUser*) para a síntese, quando esta for solicitada.

Testes realizados apenas com a classe cliente (*MaryClientUser*) possibilitou a geração de uma janela para testes, como mostra a figura 6.

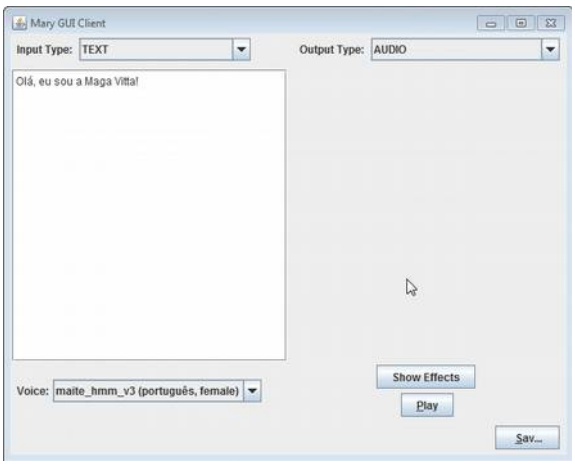


Figura 6 – Janela de testes no cliente MaryTTS.

Os modelos acústicos de síntese para o Português que vêm com o MaryTTS são primitivos, proporcionando uma voz visivelmente mecânica e de difícil audição. Neste sentido, foi necessária a gravação de mais de 600 frases (locução feminina) e geração de um léxico de 700 palavras com sua anotação fonética para treinamento do modelo acústico. Este modelo proporcionou uma síntese com boa audição, embora ainda possa melhorar, através do ajuste de parâmetros de saída do MaryTTS.

Nova engine Blender

A equipe do Città já trabalhava com o software livre Blender para modelagem dos objetos em 3D e partimos para a linguagem Python (já integrada nas versões do Blender). Pois assim seria possível excluir boa parte de problemas que tivemos de compatibilidade com programas e teríamos ainda uma ferramenta mais potente, estável e que poderíamos ter mais controle sobre o próprio desenvolvimento do software, sem o uso de diversos programas para adaptação.

Outra funcionalidade que achamos muito interessante é a possibilidade de embutir animações. Uma ideia que desde o começo do projeto era fazer a integração da Maga Vitta com algum tipo de animação ou 3D. Hoje isso é possível.

Acabamos com problemas de leitura de texturas ou materiais. Agora tem-se a possibilidade de trabalhar com a OpenGL plenamente e utilizam-se sistemas de *shaders* complexos para tratamento dos modelos em tempo real de jogo, com iluminação, sombra, luz, mapeamento alpha, etc.

A partir de agora o Città será mais estável, sendo definido na programação a quantidade de frames que irá rodar o jogo. E o jogo passará a rodar em Windows, Linux e Mac sem nenhum problema, além de ser possível seu uso em navegadores web.

Pontos decisivos para escolhermos o Blender:

- Programa em software livre;
- Motor de jogo integrado com linguagem em Python – aqui nós optamos por termos alguns problemas de incompatibilidade com Java e também por ser mais estável para diversas configurações de máquinas;

- Programa leve, rápido e eficiente para nossa proposta;
- Agilidade no processo do jogo, pois temos todas as ferramentas em um programa. Isso nos auxilia para melhor integração dos bolsistas entre si e com o projeto;
- Sistema conhecido pelos bolsistas do projeto, pois já era utilizado para desenho dos objetos 3D, e a linguagem integrada Python também familiar para a equipe;
- Possibilidade de construção da abordagem multijogador, a exemplo do jogo WSAG[30], e a constituição de redes de cidades;
- Existência de plugins para rodar um .blender em navegadores web, como o Burster[31].

Dentro dessa nova concepção do Città no Blender, estão sendo previstos novos recursos como a leitura de dados de relevo reais para modelagem do terreno da cidade. Isso significa que uma cidade poderá ser reconstruída com os dados reais em relação ao seu relevo. Outro recurso previso com o multijogador e com a interface web é a geolocalização. Através dela será possível o mapeamento das diferentes culturas, e da localização dos grupos de alunos, suas escolas e cidades (imaginadas ou reais).

Considerações finais

O Projeto Città Cosmopolita é uma parceria entre a empresa CONEXUM e o LELIC-FACED-UFRGS, para desenvolvimento de novas tecnologias digitais que promovam a inovação nos processos educacionais. Nesta parceria é desenvolvido o software livre Città, voltado à construção de cidades imaginadas por estudantes do 4º ano do ensino fundamental, dentro do contexto do Projeto CIVITAS.

Città é um software desenvolvido com pedagogas e voltado à aplicação como ferramenta de apoio à metodologia CIVITAS. Neste sentido, Città está sempre evoluindo, na interação tanto com as crianças usuárias quanto as professoras que refletem sobre a aplicação desta ferramenta sobre o pensar, sentir e agir no contexto educativo.

Do ponto de vista tecnológico, desenvolvem-se os recursos de interação gráfica, textual e vocal. Ainda são analisadas formas de monitoramento das ações do jogo de simulação e do banco de dados de interações para tomada de decisões de diálogo.

Os novos dispositivos proporcionados pelo Blender estão resultando em interações mais ricas e complexas, e, em contraponto, mais fáceis de implementar computacionalmente. Uma vez que existe uma gama de funcionalidades já incorporadas no Blender, a rapidez e o efeito obtidos nas ações do Città são muito superiores aos que foram produzidos nos últimos anos.

As aplicações que podem ser derivadas desse processo são muitas. Além de simulações que podem ser feitas com jogos empresariais, voltados à educação de adultos nas relações de trabalho, há possibilidade da criação de novos tipos de redes de convivência em ambientes tridimensionais, inclusive na internet e com dispositivos móveis.

O uso de assistentes que possibilitam o diálogo é um dado relevante, não somente do ponto de vista educacional, mas para qualquer um que queira ter suas tarefas facilitadas junto ao computador ou outro dispositivo eletrônico. Sistemas de monitoramento com diálogo, principalmente envolvendo diversas mídias combinadas de imagem, texto e voz, podem vir a compor uma nova concepção de interface.

O desenvolvimento do Città Cosmopolita vem comprovando que a pesquisa computacional caminha de mãos dadas com o estudo de campo. Apesar da formação profissional

[30] http://www.oldjim.ch/WSAG
[31] http://geta3d.com

dar um enfoque estanque aos conhecimentos acadêmicos, toda a relação de pesquisa e desenvolvimento dá-se na interação de diferentes áreas. No caso deste projeto, o diálogo entre Educação e Computação é essencial para o adequado uso do computador como instrumento de qualificação do cidadão do amanhã.

Agradecimentos. O projeto Città Cosmopolita é financiado com o apoio do CNPq – Conselho Nacional de Desenvolvimento Científico e Tecnológico. Agradecemos a toda a equipe do Projeto CIVITAS por sua assessoria a este projeto, em especial a coordenadora do LELIC profa. Dra. Margarete Axt (FACED/CINTED - UFRGS). Também agradecemos a parceria dos professores Andréa Konzen (UNISC), Marcio Martins (UNIPAMPA) e Tânia Hetkowski (UNEB).

Referências

Albuquerque, R. M. ; Cruz, D. M. ; Azevedo, V. A. (2009) Jogos eletrônicos na escola: uma experiência educativa com o LinCity-NG. In: VIII Simpósio Brasileiro de Jogos e Entretenimento Digital, 2009, Rio de Janeiro. Proceedings do VIII Simpósio Brasileiro de Jogos e Entretenimento Digital, 2009 v. 1. p. 57-60.

Araújo, C. de F. M.; Simon, E. B. (2008) Objetos de aprendizagem em educação a distância: uso de jogos educacionais no estilo RPG (Role-Playing Games) digitais. In: *Revista Paidéi@*, Vol. 1, No 1.

Axt, M. e Martins, M. A. R. (2008) Coexistir na diferença: De quando a formação em serviço pensa modos de habitar a sala de aula. In: Trindade, Iole Maria Faviero. (Ed), Múltiplas Alfabetizações e Alfabetismos, pp. 133-158, Editora da UFRGS, Porto Alegre, 1 ed., v. 1.

Guterres, J. O.; Eichler, M. L.; Del Pino, J. C. (2003) A usabilidade de Carbópolis, um software livre para a educação ambiental.. Tecnologia Educacional, Rio de Janeiro, v. 32, p. 69-82.

Hornes, A.; Grachinski, L.; Silva, S. C. R.; Koscianski, A. (2009) Os jogos computacionais no ensino de física. In: VII Encontro Nacional de Pesquisa em Educação em Ciências, 2009, Florianópolis, SC, Brasil.

John, Oliver P. (1999) "The 'Big Five' factor taxonomy: Dimensions of personality in the natural language and in questionnaires", In: Handbook of personality: Theory and research. Edited by Lawrence A. Pervin, Guilford.

Konzen, A. A. ; Braitback, O. L. B.; Kist, L. ; Anjos, A. R. ; Moraes, L. D.; Lima, C. I. L.; Müller, D. N.; Axt, M. (2011) Maga Vitta: agente conversacional aplicado ao jogo educacional Città. In: *Simpósio Brasileiro de Informática na Educação*, Aracajú.

Oliveira, C. (2011) A contribuição dos jogos de simulação na aprendizagem: uma experiência com jogo The Sims. In: *V Colóquio Internacional "Educação e Contemporaneidade"*. São Cristóvão, 21 a 23 de setembro.

Oliveira, O. L. B. de, Müller, D. N., Remião, J. A. A., Silveira, P. D., Ferreira Filho, R., Martins, Márcio A. R. e Axt, Margarete. (2009) CIVITAS: cidades virtuais na educação. In: Andrea S. Charão et all. *X workshop de Software Livre - FISL 2009*. Porto Alegre, pp. 141-146.

Ortony, A., Clore, G. L. e Collins, A. (1988), "The cognitive structure of emotions", Cambridge University Press.

Pavón, R., Díaz, F., Laza, R. e Luzón, V. (2009) "Automatic parameter tuning with a Bayesian case-based reasoning system. A case of study" In: Expert Systems with Applications 36, pages 3407–3420, Elsevier.

Pereira, F. L. F; Araújo, S. de L.; Holanda, V. C. C. de. (2011) As novas formas de se ensinar e

aprender geografia: os jogos eletrônicos como ferramenta metodológica no ensino de geografia. In: *Geosaberes*, Fortaleza, v. 2, n. 3, p. 34-47, jan. / jul.

Tavares, R. B. (2012) O Jogo da Geografia: SimCity como recurso didático no Ensino de Geografia e no Planejamento de cidades. Sistema Pec de Ensino. Disponível em www.br-ie.org/pub/index.php/wcbie/article/download/2668/2322 Acesso em maio/2012.

Tarouco, L.M.R. ; Fabre, M. ; Roland, L. C. ; Konrath, M. L. P. (2004) Jogos educacionais. RENOTE. Revista Novas Tecnologias na Educação, Porto Alegre, v.2, n.1, p. 1-7.

LELIC – Laboratório de Estudos em Linguagem, Interação e Cognição

(Publicado no CBIE/PGPIE 2013)

Daniel Nehme Müller[1], Andréa Konzen da Silva[2 3], Maribel Suzane Selli[3], Sansão Timbane[3], Otto Braitback[4], Daniel Epstein[5] , Joelma Remião[3] e Margarete Axt[6]

Conexum Ltda.[1]

Departamento de Informática, Universidade de Santa Cruz do Sul (UNISC)[2]

Laboratório de Estudos em Linguagem, Interação e Cognição, Universidade Federal do Rio Grande do Sul (UFRGS)[3]

Programa de Pós Graduação em Informática na Educação, Universidade Federal do Rio Grande do Sul (UFRGS)[4]

daniel@conexum.com.br, andrea@unisc.br, mselli7@gmail.com, sansaot@gmail.com, ottolb@gmail.com, depstein@inf.ufrgs.br, joremiao@gmail.com, maaxt@ufrgs.br

O Grupo de Pesquisa "LELIC – Laboratório de Estudos em Linguagem, Interação e Cognição: metodologias para formação em rede/EAD, sentido e autoria, criação em hipermídia" é uma equipe interdisciplinar que vem desenvolvendo objetos e ambientes virtuais de aprendizagem, bem como ações pedagógicas que considerem a interação com tecnologias digitais, enfatizando a construção do conhecimento e a produção de sentidos. Compõe o LELIC um grupo de pesquisadores da UFRGS (Universidade Federal do Rio Grande do Sul) de nível de Pós-Graduação (PPGs Educação - PPGEDU e Informática na Educação - PPGIE) e Graduação, visando o desenvolvimento de tecnologias cada vez mais integradas às demandas teórico-práticas educacionais e sociais. Investiga formação em rede (presencial/distância), efeitos da interação em espaços de linguagem e do agenciamento de tecnologias em relação de coexistência, nos planos da autoria coletiva, criação estética e construção do conhecimento, sob perspectiva ética da escuta e do cuidado com os sentidos. LINHAS DE PESQUISA: 1)PROVIA - Problematizações dos Processos de Investigação do Virtual, da Criação e da Autoria. Projetos: a)Forchat e diário virtual: autorias mestiças - apoio do edital 12/2010 - SEAD/UFRGS; b)Mapeamentos cartográficos dos planos de produção analítica em pesquisa: articulações entre o comum e o diverso - apoio do edital AT-CNPq 2010; c)Deslizamentos de sentido, posições efêmeras de autoria: o lugar da multiplicidade da formação docente em rede - Bolsa PQ/CNPq; d) Ao provia também está vinculada uma bolsa de iniciação científica PIBIC/UFRGS. 2)CIVITAS - Cidades Virtuais com Tecnologias para Aprendizagem e Simulação. Projetos: a)Brasil-Moçambique: formação continuada em serviço e prática docente em séries iniciais do ensino fundamental - CNPq Pró-África: Edital MCT/CNPq nº 12/2008; b)Convênios de cooperação pesquisa-extensão entre a UFRGS e as prefeituras de: Mato Leitão/RS e Cruzeiro do Sul/RS. 3)COSMOPOLITA: Simuladores de Cidades em Rede. Projeto: Città Cosmopolita - apoio Edital MEC/CAPES e MCT/CNPq/FINEP Nº 28/2010 (PNPD 2010), edital MCT/CNPq Nº 75/2010 (RHAE Pesquisador na Empresa) e Edital Universal - CNPq Nº 14/2011. Colaboram com o LELIC a empresa CONEXUM e a Associação dos Municípios do Vale do Rio Pardo.

Città Cosmopolita – simuladores de cidades em redes

(Apresentado no FISL 14 – Trilha Academia Livre – 2013)

Gravação em:

http://hemingway.softwarelivre.org/fisl14/high/40a/sala40a-high-201307031611.ogg

Laurent Le Avner Quevedo[1], Andressa Cruz Nepomuceno[2], Mercio Luciano Teixeira Soares[3], Juarez Dubois da Costa[4], Maribel Susane Selli[5] e Daniel Nehme Müller[6]

Curso de Jogos Digitais, Universidade do Vale do Rio dos Sinos (UNISINOS)[1]
Curso de Ciência da Computação, Universidade do Vale do Rio dos Sinos (UNISINOS)[2]
Curso Produção Multimídia Digital, Universidade do Sul de Santa Catarina (UNISUL)[3]
Laboratório de Estudos em Linguagem, Interação e Cognição, Universidade Federal do Rio Grande do Sul (UFRGS)[4]
Programa de Pós Graduação em Informática na Educação, Universidade Federal do Rio Grande do Sul (UFRGS)[5]
Conexum Ltda.[6]

laurentleavnerquevedo@gmail.com,andressacruzn@gmail.com, ls.druid@gmail.com, juarezdc@hotmail.com, mselli7@gmail.com, daniel@conexum.com.br

Resumo. *O Projeto Città Cosmopolita desenvolve o lado digital de invenção de cidades promovido pelo Projeto CIVITAS, da Faculdade de Educação/UFRGS. Città é um software livre GPL que permite a construção de uma cidade tridimensional usando Java e banco de dados SQLite.* ***O contexto de aplicação do mesmo são turmas de quarto ano de Escolas Municipais de Ensino Fundamental de um Município da Região do Vale do Taquari no Estado do Rio Grande do Sul.*** *Neste artigo está sendo apresentada a reescrita do sistema em Blender, com a ex****ploração de*** *recursos multijogador, chat* ***por voz e texto, além de seu uso em sistemas android*** *e* ***navegadores*** *web.*

TecDoc

Documentação Técnica

Città versão Blender

Versão 1.1

Autores: Andressa Cruz, Luciano Soares e Laurent Quevedo
Revisor: Daniel Nehme Müller

maio de 2013

1. Introdução

O editor de cidades Città desenvolve o lado digital de invenção de cidades promovido pelo Projeto CIVITAS, do Laboratório de Estudos em Linguagem, Interação e Cognição (LELIC) da Faculdade de Educação/UFRGS. Città é um software livre GPL que permite a construção de uma cidade tridimensional, usando Blender e banco de dados SQLite embutido. Neste documento estarão sendo apresentados, além de uma rápida explicação sobre a versão anterior em Java, também sua atual versão reescrita do sistema na plataforma *Blender*, com a exploração de recursos multijogador, chat por texto, além do seu uso potencial em sistemas Android e navegadores web.

O editor Città é desenvolvido pela empresa Conexum Ltda. com apoio do CNPq, em parceria com o LELIC.

1.1. Plataforma

O que é Blender?

Blender, também conhecido como blender3d, é um programa de computador de código aberto, desenvolvido pela Blender Foundation, para modelagem, animação, texturização, composição, renderização, edição de vídeo e criação de aplicações interativas em 3D, tais como jogos, apresentações e outros, através de seu motor de jogo integrado, o Blender Game Engine.

Tínhamos como base a primeira versão do Città desenvolvido ainda em Java para iniciar e acelerar o processo de criação da versão Blender. Como não conseguimos aproveitar praticamente nada resolvemos começar o jogo do zero. Pegamos os pontos fortes do que já tínhamos definido anteriormente e reescrevemos tudo o que ficou pendente ou o que não agradava ou não funcionava na versão Java. Os controles do jogo tiveram seus códigos reescritos em Python, pois já vem integrado com o Blender.

1.2. Delimitação da Plataforma

Um novo código merecia um novo visual, então colocamos em prática todos os testes que foram possíveis em relação à modelagem, animação, interface e tudo o que iria ser trabalhado. Pois a falta de testes mais específicos na versão anterior acabou nos atrapalhando a longo prazo no projeto. Os objetos e construções em 3D foram novamente retrabalhados, os menus, o mapa e novamente o assistente inteligente Maga Vitta, que agora irá ter sua versão em 3 dimensões com algumas animações enquanto se comunica com o aluno.

Resolvemos trabalhar com OpenGL Shading Language (GLSL), uma linguagem de shading de alto nível baseada na linguagem de programação C. Foi criada pela OpenGL ARB para dar aos desenvolvedores controle mais direto do pipeline de gráficos sem ter de usar a linguagem de assembly ou linguagens especificas de hardware. A GLSL é extremamente poderosa se utilizada da maneira correta, mas em contrapartida exige mais do hardware em que irá rodar. Começamos então um longo processo de estudo sobre materiais e texturas para GLSL e também sobre ela para adequação ao projeto, pois os computadores precisariam de um mínimo de configuração e suporte a OpenGL. O recurso GLSL usado atualmente é embutido na plataforma Blender.

1.3. Justificativa da Escolha da Plataforma

Após alcançarmos uma versão estável do Città criada em Java, percebemos que o avanço no sistema poderia ser alcançado apenas trocando-se o paradigma. Como o Java exigia muita programação, tanto o desenvolvimento quanto as correções exigiam muito tempo da equipe. Foi então realizada uma pesquisa para reescrever o Città numa nova plataforma. Após um mês de estudos, a única opção que atendia a todas as nossas necessidades era o Blender.

Já estávamos trabalhando com o Blender para modelagem dos objetos em 3D e partimos para a linguagem Python (já integrada nas versões do Blender). Pois assim poderíamos excluir boa parte de problemas que tivemos de compatibilidade com programas e teríamos ainda uma ferramenta mais potente, estável e que poderíamos ter mais controle sobre o próprio desenvolvimento do software, sem o uso de diversos programas para adaptação.

Alguns pontos decisivos para escolhermos o Blender:

- Programa com código livre pela licença OpenGL;
- Motor de jogo integrado com linguagem em Python – aqui nós optamos por termos alguns problemas de incompatibilidade com Java e também por ser mais estável para diversas configurações de máquinas;
- Programa leve, rápido e eficiente para nossa proposta;
- Agilidade no processo do jogo, pois temos todas as ferramentas em um programa. Isso nos auxilia para melhor integração dos bolsistas entre si e com o projeto;
- E não menos importante: programa conhecido por todos os bolsistas do projeto.

1.4. Método de Trabalho

Visto que o Blender utiliza uma classe padrão para todos os objetos, os códigos são estruturados em scripts, separados conforme sua função. Portanto, um script que deve mover um objeto, por exemplo, pode ser utilizado em qualquer objeto, mas queremos que ele seja usado em objetos específicos. Os scripts devem ser comentados conforme a função de cada trecho.

Para o controle de versão, usamos o sistema de versionamento GIT do SourceForge[32], de forma que todos consigam realizar alterações no mesmo projeto e manter sempre a cópia atualizada do projeto.

2. Descrição Geral do Sistema

2.1. Usuários do Sistema

Dentro de contextos de jogos de simulação, Città é a versão digital do simulacro da cidade que ocorre na sala de aula com alunos do 4º/5º ano do ensino fundamental atendidos pelo Projeto CIVITAS. As crianças reinventam cidades imaginárias onde eles são cidadãos e tornam-se responsáveis por todos os setores da sociedade.

[32] http://www.sourceforge.net

2.2. Desenvolvedores do Sistema

A equipe é formada por pesquisadores e bolsistas, tanto na área tecnológica quanto pedagógica. Segue abaixo a área de atuação de cada um:

Daniel Nehme Müller, coordenador do projeto.

Andressa Cruz Nepomuceno, desenvolvedora, trabalha nos scripts python e banco da dados.

Mercio Luciano Teixeira Soares, designer, produz os objetos gráficos utilizados no sistema.

Laurent Le Avner Quevedo, desenvolvedor, trabalha na lógica de manipulação dos objetos do sistema.

Andréa Aparecida Konzen, pesquisadora, projeta as ações emocionais do agente inteligente.

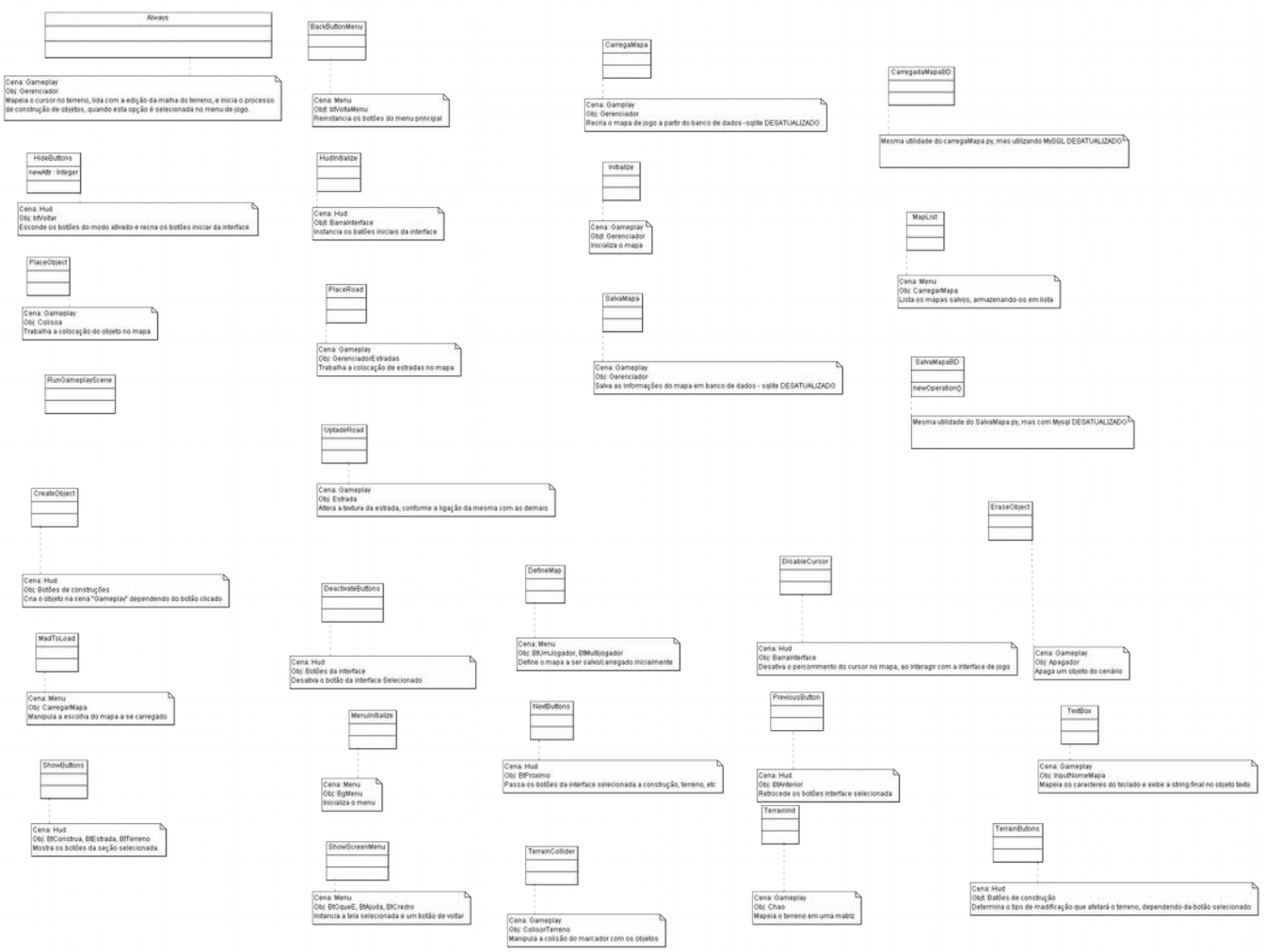
Pablo Felipe Leonhart, desenvolvedor, atua na programação do assistente inteligente.

Yuri Dias Meira, desenvolvedor, participa da programação da lógica de ação do assistente inteligente.

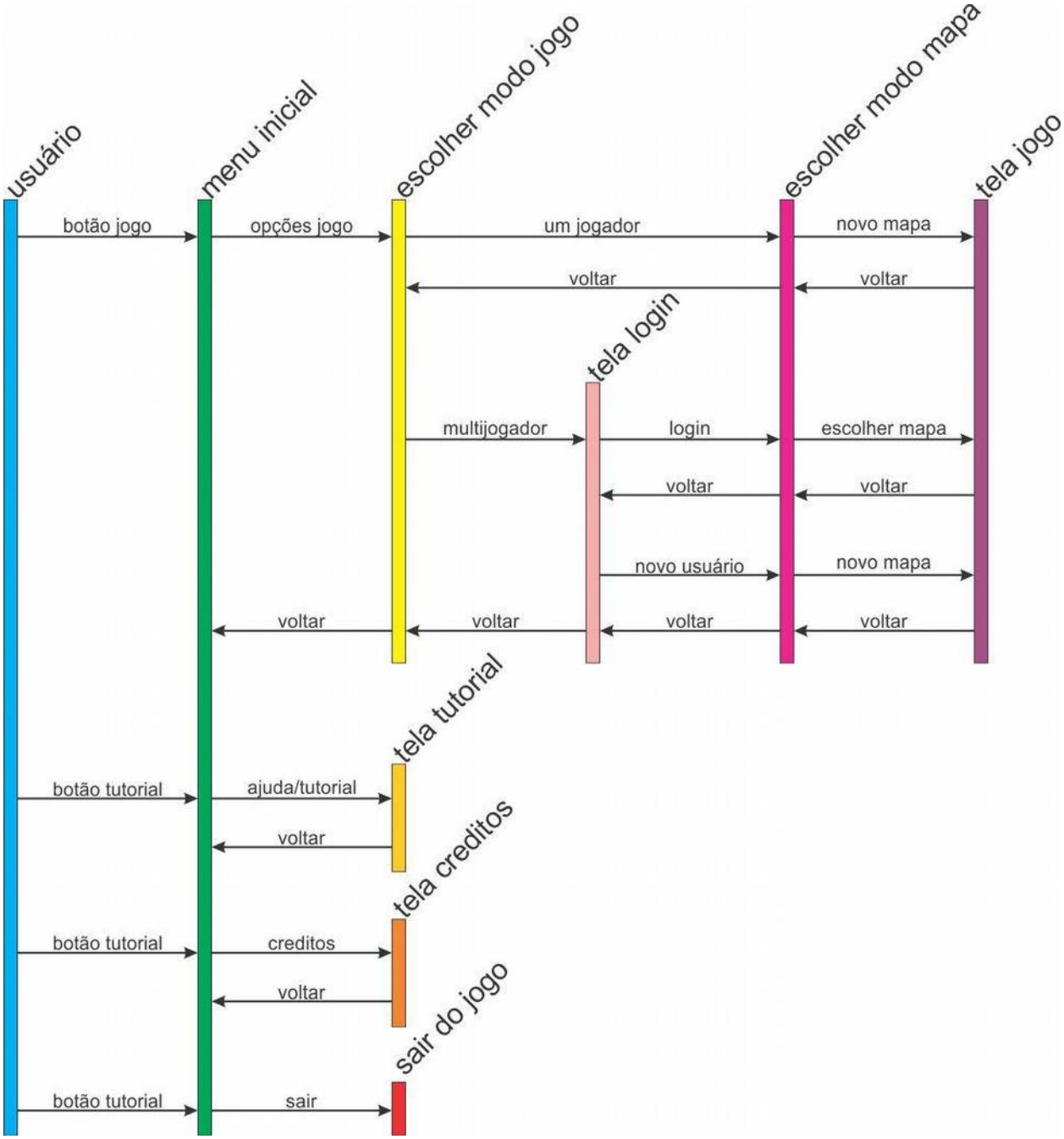
Maribel Susane Selli, pesquisadora, acompanha as ações do projeto nas escolas.

2.3. Diagramas

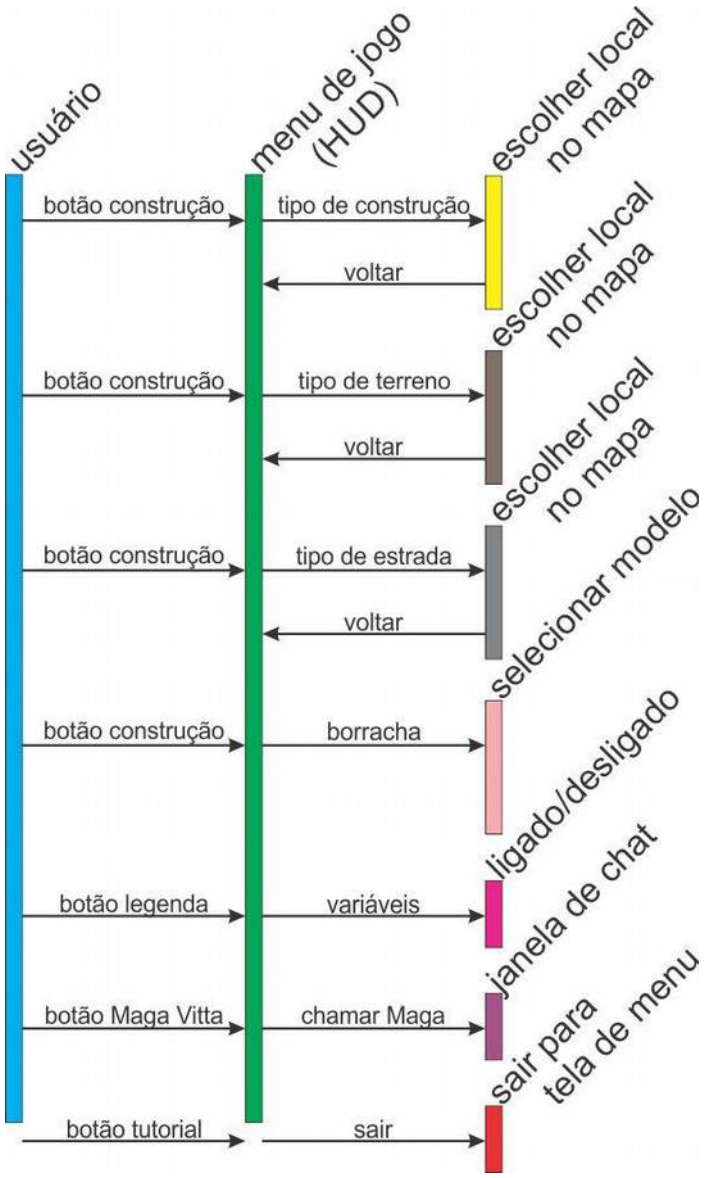
2.3.1. Diagrama Geral do Sistema



2.3.2. Diagrama de sequência - Menu Inicial



2.3.3. Diagrama de sequência - Menu de jogo (HUD)



3. Versões do Sistema

Como sabemos que em algumas escolas possuem em seus laboratórios computadores bastante defasados, devemos ter uma versão resumida do sistema. Esta versão será a mais compacta possível, para que todos possam ter a experiência de criar a sua cidade, independente da máquina em que esteja criando.

3.1 Protótipo

Existem basicamente três versões do Città: a monojogador, a multijogador e a versão reduzida.

3.1.1. Versão monojogador

Esta versão do simulador possuirá todas as funcionalidades necessárias para a construção da cidade virtual, porém, não irá dispor da conexão entre usuários, e da possibilidade de intervenção do agente inteligente.

Para se ter a referência das posições dos objetos no mundo 3D no formato de "tiles", estipulamos que cada tile do terreno terá o tamanho 1x1 em (x,y), no sistema de coordenadas oferecido pelo Blender. Dessa forma, podemos ter um objeto cursor, que percorrerá o mapa tendo sua posição em (x,y) arredondada para números inteiros, e usar este objeto como referência para o posicionamento de outros objetos no mundo.

Cada construção (casas, prédios, etc) feita sobre o terreno de jogo será composta de três objetos 3D: o plano de colisão, para assegurar que o objeto não será colocado por cima de uma área indevida, o "pátio", para facilitar visualização da área ocupada pela construção, e modelo da construção de fato, que carregará consigo os atributos do objeto a ser posto. Este plano de colisão será redimensionado conforme a quantidade de tiles que a construção ocupa em X e em Y, valores estes armazenados em *game properties* do objeto a ser posto.

Para a colocação de estradas, o algoritmo deve guardar a posição do cursor no momento do clique do jogador, e a partir dessa posição com a posição onde o mouse estiver, criar e posicionar os tiles de estrada. Cada tile de estrada então deve detectar sua conexão com seus quatro lados, e então alterar sua textura dependendo dessas conexões.

Tendo em mente que, para se pensar em um algoritmo que faça a alteração do terreno de forma que não comprometa o desempenho do software, os vértices da malha a ser alterada devem estar ordenados de uma determinada forma. Para isso, a lista de vértices do objeto deve ser primeiramente mapeada para uma matriz. Com isso, é possível desenvolver um script que realize alterações no eixo Z dos vértices que estiverem em um determinado raio em volta do cursor do mouse.

A interface de jogo (HUD) é uma cena que sobrepõe à de jogo. Assim, podemos ter duas câmeras lidando com a renderização dos elementos: a da cena onde se encontram o terreno e os objetos de jogo, sendo uma câmera perspectiva, e da cena de interface, realizando por sua vez a renderização ortográfica dos paineis 2D, assim como seus botões.

O salvamento do mapa e dos objetos será feito em banco de dados SQLite, já que este realiza o armazenamento das informações de forma local. Dessa forma, conseguimos manter a coerência entre as tabelas das duas versões.

Dessa forma, temos tabelas distintas para o salvamento dos objetos e para os vértices do terreno alterado pelo usuário.

3.1.2. Versão Multijogador

Com o objetivo de interligar diversas crianças de diferentes cidades, nesta nova versão do Città foi acrescentada mapas compartilhados. Neste mapa é possível conversar por chat modo texto e contruir o mapa em conjunto, com vários usuários para modificar o mapa carregado/criado.

Pensando também nesta limitação de processamento, resolvemos separar o modo multiplayer do modo monoplayer. Assim, a versão para o multiplayer tem o menu diferente do monoplayer, com o carregamento (na hora de escolher um mapa) dos mapas sendo feito através do banco, com o nome dos mapas que já existem em nuvem.

A interface do menu do Multiplayer é igual a do menu Singleplayer, a única diferença é que no SinglePlayer, para carregar a lista acima, é feito uma varredura na pasta local onde é armazenado os dados dos mapas salvos. No Multiplayer, essa consulta de mapas é feita no banco de dados em nuvem, na tabela listaMapas do banco listaMapas. Toda vez que é criado uma mapa novo no jogo, é verificado se já existe um mapa com aquele nome, se existe retorna um erro, avisando que não poderá existir mapas com o mesmo nome. Se não existe, é criado

então um novo banco com o nome do mapa, e inserido o nome deste novo mapa na tabela listaMapas. A cada 2 segundos é verificado se o mapa atual está igual ao em nuvem, e se não está ele salva de novo para ficar igual. O primeiro carregamento é feito completo, e o carregamento da verificação de alteração carrega apenas aquilo que está diferente.

No modo multiplayer, existe o chat de texto, para que todos que estão alterando aquele mapa possa conversar e trocar informações. Tanto o multiplayer em si quanto o chat foram feitos a partir de WebService, que serve justamente para trocar informações com aplicações de diferentes linguagens.

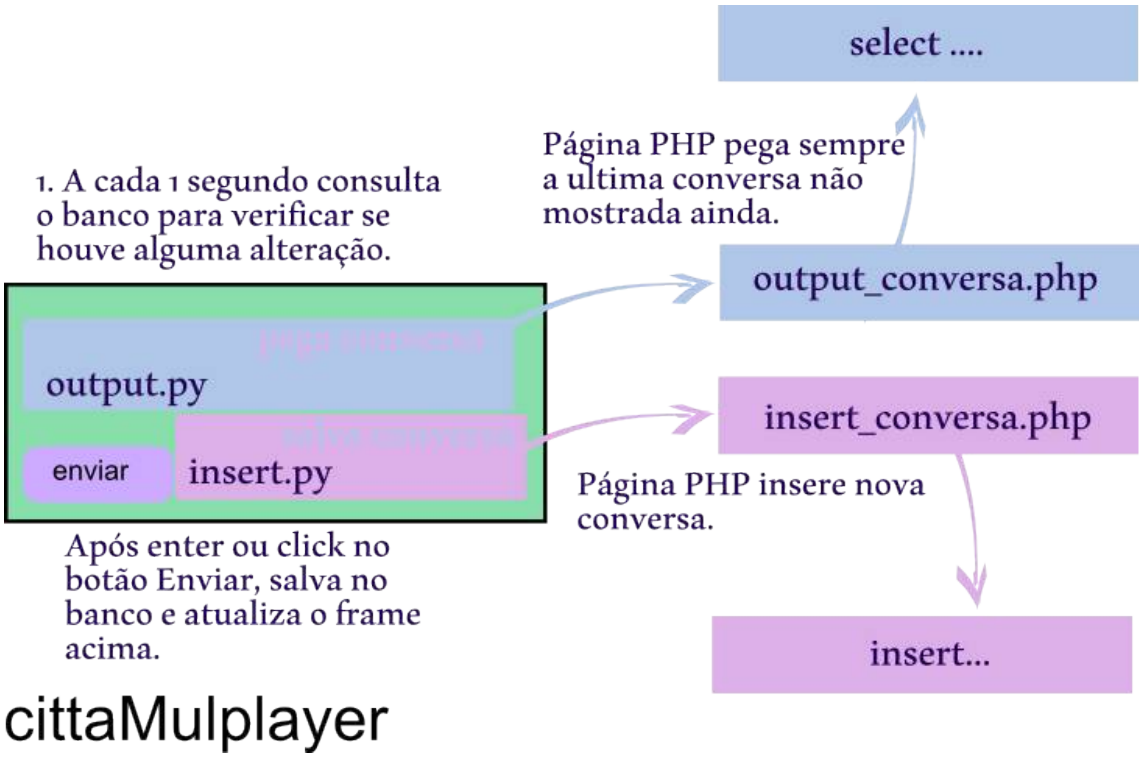
3.1.2.1. Sobre WebService e o Chat modo texto

Web Service é a solução para integrar aplicações distintas. E é exatamente o que precisamos. Como o Burster não funcionou, logo não podemos usar o WebRTC para fazer o chat texto. Então faremos dentro do blender mesmo, e a maneira de salvar informações em nuvem de uma aplicação construída no Blender, é utilizando WebService.

O HTTP (em português Protocolo de Transferência de Hipertexto), é um protocolo de comunicação, utilizando o Modelo OSI. Este protocolo é o que fez a internet (World Wide Web) funcionar, que é uma rede de HTTP 'conversando' entre si. Hipertexto é que se movimenta nessa operação. O HTTP utilizada este texto estruturado (que utiliza ligações lógicas, hiperlinks) para a transmissão a informações.

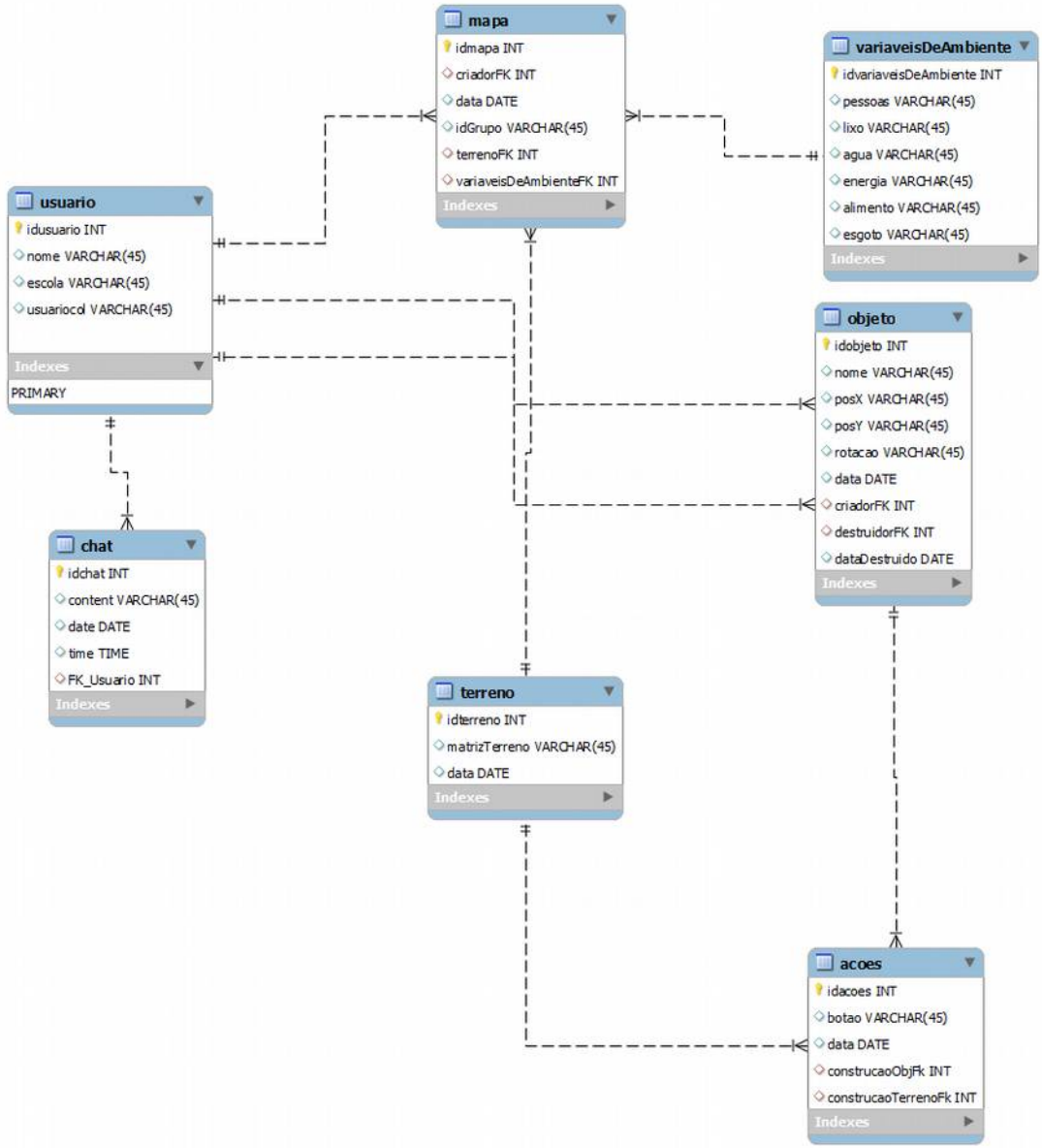
URL (em português Localizador-Padrão de Recursos, é o endereço de um recurso disponível em uma rede (seja internet ou intranet). Na URL especifica-se o protocolo a ser usado (protocolos que são usados para acessar certo conteúdo em um ser servidor, para a transferencial de arquivos ou acessar o site, por exemplo).

O sistema num todo funciona como na imagem a seguir:



Como a própria imagem mostra, temos dois campos principais, um que fica verificando se existe mensagem nova e outro que é para enviar uma mensagem. Como não temos o sistema Multiplayer, eu utilizei variáveis como exemplo que mais tarde, estas informações serão tiradas do sistema (nome do usuário e nome da sala/mapa compartilhada, por exemplo).

Pensando na estrutura geral de como ficará o game, tando na parte multiplayer quando no chat, e levando em consideração também as variáveis de ambiente que serão usadas depois para análise, fiz o seguinte diagrama:



3.1.2. Versão Reduzida

Devido às limitações da configuração dos computadores encontrados nas escolas públicas, temos uma versão com recursos limitados, permitindo um bom desempenho, mesmo em máquinas antigas.

Esta versão utiliza sistema Multitexture de shading, e não possui animações, de forma a não comprometer a performance.

Referências

Albuquerque, R. M. ; Cruz, D. M. ; Azevedo, V. A. (2009) Jogos eletrônicos na escola: uma experiência educativa com o LinCity-NG. In: VIII Simpósio Brasileiro de Jogos e II Congresso Brasileiro de Informática na Educação (CBIE 2013) Workshops (WCBIE 2013).

Araújo, C. de F. M.; Simon, E. B. (2008) Objetos de aprendizagem em educação a distância: uso de jogos educacionais no estilo RPG (Role-Playing Games) digitais. In: Revista Paidéi@, Vol. 1, No 1.

Axt, M. e Martins, M. A. R. (2008) Coexistir na diferença: De quando a formação em serviço pensa modos de habitar a sala de aula. In: Trindade, Iole Maria Faviero. (Ed), Múltiplas Alfabetizações e Alfabetismos, pp. 133-158, Editora da UFRGS, Porto Alegre, 1 ed., v. 1.

Entretenimento Digital, 2009, Rio de Janeiro. Proceedings do VIII Simpósio Brasileiro de Jogos e Entretenimento Digital, 2009 v. 1. p. 57-60.

Guterres, J. O.; Eichler, M. L.; Del Pino, J. C. (2003) A usabilidade de Carbópolis, um software livre para a educação ambiental.. Tecnologia Educacional, Rio de Janeiro, v. 32, p. 69-82.

Hornes, A.; Grachinski, L.; Silva, S. C. R.; Koscianski, A. (2009) Os jogos computacionais no ensino de física. In: VII Encontro Nacional de Pesquisa em Educação em Ciências, 2009, Florianópolis, SC, Brasil.

John, Oliver P. (1999) “The ‘Big Five’ factor taxonomy: Dimensions of personality in the natural language and in questionnaires”, In: Handbook of personality: Theory and research. Edited by Lawrence A. Pervin, Guilford.

Konzen, A. A. ; Braitback, O. L. B.; Kist, L. ; Anjos, A. R. ; Moraes, L. D.; Lima, C. I. L.; Müller, D. N.; Axt, M. (2011) Maga Vitta: agente conversacional aplicado ao jogo educacional Città. In: Simpósio Brasileiro de Informática na Educação, Aracajú.

Oliveira, C. (2011) A contribuição dos jogos de simulação na aprendizagem: uma experiência com jogo The Sims. In: V Colóquio Internacional "Educação e Contemporaneidade". São Cristóvão, 21 a 23 de setembro.

Oliveira, O. L. B. de, Müller, D. N., Remião, J. A. A., Silveira, P. D., Ferreira Filho, R., Martins, Márcio A. R. e Axt, Margarete. (2009) CIVITAS: cidades virtuais na educação. In: Andrea S. Charão et all. X workshop de Software Livre - FISL 2009. Porto Alegre, pp. 141-146.

Ortony, A., Clore, G. L. e Collins, A. (1988), “The cognitive structure of emotions”, Cambridge University Press.

Pavón, R., Díaz, F., Laza, R. e Luzón, V. (2009) “Automatic parameter tuning with a Bayesian case-based reasoning system. A case of study” In: Expert Systems with Applications 36, pages 3407–3420, Elsevier.

Pereira, F. L. F; Araújo, S. de L.; Holanda, V. C. C. de. (2011) As novas formas de se ensinar e aprender geografia: os jogos eletrônicos como ferramenta metodológica no ensino de geografia. In: Geosaberes, Fortaleza, v. 2, n. 3, p. 34-47, jan. / jul.

Tavares, R. B. (2012) O Jogo da Geografia: SimCity como recurso didático no Ensino de Geografia e no Planejamento de cidades. Sistema Pec de Ensino. Disponível em www.br-ie.org/pub/index.php/wcbie/article/download/2668/2322 Acesso em maio/2012.

Tarouco, L.M.R. ; Fabre, M. ; Roland, L. C. ; Konrath, M. L. P. (2004) Jogos educacionais. RENOTE. Revista Novas Tecnologias na Educação, Porto Alegre, v.2, n.1, p. 1-7.