



UNIVERSIDADE ESTADUAL DE CAMPINAS  
FACULDADE DE ENGENHARIA ELÉTRICA E DE  
COMPUTAÇÃO

## **Late Redemption – Game Design Document**

**Introdução ao Projeto de Jogos Digitais (IA369A)**  
**Professor: José Mario De Martino**  
**Primeiro semestre de 2011**

**Grupo 1**  
Carla Sampaio  
Cesar Tegani Tofanini  
Jennifer Chuin Lee  
Marcos Donato da Silva  
Matheus Leonardo Parolin  
Rafael Sangalli  
Rodrigo Costa Leite

# Índice Geral

Resumo do Documento .....	1
1 Nome do Jogo .....	1
2 Visão Global do Jogo .....	1
2.1 Características Gerais.....	1
2.1.1 Estilo.....	1
2.1.2 Perspectiva.....	1
2.1.3 Modo.....	1
2.1.4 Número de fases .....	1
2.1.5 Linha do tempo .....	1
2.1.6 Idioma do jogo .....	1
2.1.7 Plataformas .....	2
2.1.8 Gráficos .....	2
2.1.9 Principal Forma de Controle .....	2
2.2 Filosofia .....	2
2.2.1 Ponto Filosófico #1.....	2
2.2.2 Ponto Filosófico #2.....	2
2.2.3 Ponto Filosófico #3.....	2
2.3 Conceito Geral.....	2
2.3.1 Descrição do Jogo.....	3
2.3.2 Referências e inspirações.....	3
2.3.2.1 Silent Hill .....	3
2.3.2.2 Resident Evil.....	4
2.3.2.3 Fatal Frame .....	5
2.3.2.4 Jogos Mortais .....	6
2.3.2.5 Fear .....	7
2.3.2.6 Condemned: Criminal Origins .....	8
2.3.3 Motivação .....	9
2.3.4 Enredo .....	10
2.3.5 Ambientes a serem simulados .....	11
2.3.6 Personagem controlado pelo jogador .....	11
2.3.7 Personagens controlados pelo computador.....	11
2.3.7.1 Inimigos.....	11
2.3.7.2 Chefe Final.....	11
2.3.7.3 Silmara .....	11
2.3.8 Principal objetivo do jogo .....	11
3 Personagens.....	12
3.1 Personagem Principal.....	12
3.1.1 Descrição Física e Psicológica.....	12
3.1.2 Concepção artística.....	12
3.1.3 Imagens de referência.....	14
3.1.4 Arma do personagem principal.....	15
3.1.4.1 Descrição física.....	15
3.1.4.2 Tipos de munição .....	15
3.1.4.3 Dinâmica de uso da arma.....	16
3.1.4.4 Danos causados pelos tiros.....	16
3.1.4.5 Animações associadas.....	17

3.1.4.6	Imagens de Referência.....	17
3.1.5	Características Gerais.....	18
3.2	Inimigos.....	19
3.2.1	Shortie.....	19
3.2.1.1	Descrição Física .....	19
3.2.1.2	Características Gerais.....	19
3.2.1.3	Concepção Artística .....	20
3.2.1.4	Imagens de Referência.....	20
3.2.2	Screamer .....	21
3.2.2.1	Descrição Física .....	22
3.2.2.2	Características Gerais.....	22
3.2.2.3	Concepção artística .....	22
3.2.2.4	Imagens de Referência.....	23
3.2.3	Butcher.....	24
3.2.3.1	Descrição Física .....	24
3.2.3.2	Características Gerais.....	25
3.2.3.3	Concepção Artística .....	25
3.2.3.4	Imagens de Referência.....	26
3.2.4	Crusher.....	27
3.2.4.1	Descrição Física .....	27
3.2.4.2	Características Gerais.....	27
3.2.4.3	Concepção Artística .....	28
3.2.4.4	Imagens de referência .....	28
3.3	Chefe final - Diabolus .....	30
3.3.1	Descrição Física .....	30
3.3.2	Características Gerais.....	30
3.3.3	Concepção Artística.....	31
3.3.4	Imagens de referência.....	31
3.3.4.1	Diablo .....	31
3.3.4.2	Jericho .....	33
3.3.4.3	Mass Effect.....	34
3.3.4.4	Samara, do filme O Chamado .....	35
3.4	NPC's .....	36
3.4.1	Silmara.....	36
3.4.1.1	Descrição Física .....	36
3.4.1.2	Concepção Artística .....	36
3.4.1.3	Imagens de Referência.....	37
4	Fases.....	38
4.1	Descrição das Fases.....	38
4.1.1	Sequência .....	38
4.1.2	Planta Baixa Geral .....	39
4.2	Fase 1: Sala da Mansão .....	39
4.2.1	Descrição Física .....	40
4.2.2	Dinâmica da Fase .....	40
4.2.3	Descrição dos itens da fase .....	41
4.2.3.1	Com interação.....	41
4.2.3.2	Sem interação .....	46
4.2.3.3	Pensamentos do personagem principal.....	48
4.2.4	Flowchart .....	48
4.2.5	Planta Baixa.....	49
4.3	Fase 2: Interior da Mansão .....	50
4.3.1	Flowchart .....	50

4.3.2	Corridor .....	51
4.3.2.1	Descrição Física .....	52
4.3.2.2	Planta Baixa.....	52
4.3.2.3	Imagens de referência .....	52
4.3.3	Main Hall .....	54
4.3.3.1	Descrição Física .....	54
4.3.3.2	Planta Baixa.....	54
4.3.3.3	Imagens de referência .....	55
4.4	Fase 3: Library .....	57
4.4.1	Descrição Física .....	57
4.4.2	Flowchart .....	58
4.4.3	Planta baixa.....	58
4.4.4	Concepção Artística.....	58
4.4.5	Imagens de referência.....	59
5	Fluxo do Jogo.....	61
5.1	Progressão do Jogo .....	61
5.2	Cenas entre Fases (cinematics / cutscenes).....	62
5.2.1	Cena 1: Introdução ao Jogo.....	62
5.2.1.1	Parte 1: Apresentação.....	62
5.2.1.2	Parte 2: Enredo .....	63
5.2.1.3	Parte 3: Nome do Jogo.....	66
5.2.2	Cena 2: Início do jogo até a entrada na sala do puzzle .....	67
5.2.3	Cena 3: Transição entre a fase 1 e fase 2.....	67
5.2.4	Cena 4: Transição entre fase 2 e fase 3.....	67
5.2.5	Cena 5: Final do jogo com derrota.....	68
5.2.5.1	Derrota por Tempo .....	68
5.2.5.2	Derrota por Dano .....	68
5.2.6	Cena 6: Final do jogo com vitória.....	69
5.2.7	Cena 7: Menu principal .....	69
5.2.8	Cena 8: Níveis de dificuldade .....	70
5.2.9	Cena 9: Controles disponíveis.....	70
5.2.10	Cena 10: Carregar Jogos.....	70
5.2.11	Cena 11: Ranking.....	70
5.2.12	Cena 12: Menu 2 .....	71
5.2.13	Cena 13: Salvar Jogo .....	71
5.2.14	Cena 14: Puzzle do quadro de distribuição de energia.....	71
5.2.15	Cena 15: Créditos .....	71
5.3	Horas de Jogo .....	72
5.4	Condições de Vitória .....	72
5.5	Morte de Marshall Gory.....	72
5.6	Salvar e Carregar .....	73
6	Mecânica do jogo .....	74
6.1	Jogabilidade .....	74
6.2	Controles.....	74
6.2.1	Movimentação do personagem principal .....	74
6.2.2	Mira .....	75
6.2.3	Atirar .....	75
6.2.4	Recarregar.....	75
6.2.5	Alternar entre os tipos de munição normal e especial.....	75
6.2.6	Ações do Personagem Principal .....	75
6.2.7	Acessar tela do inventário.....	75
6.2.8	Acessar menu do jogo.....	75

6.2.9	Salvar o jogo.....	75
6.2.10	Carregar o jogo.....	75
6.3	Níveis de Dificuldade.....	76
6.4	Sistema de Combate .....	77
6.4.1	Classificação de tipos de ataque .....	77
6.5	Sistema de danos .....	78
6.5.1	Dano causado no personagem principal.....	78
6.5.1.1	Sistema de dano na dificuldade “Go Easy On Me” .....	78
6.5.1.2	Sistema de dano na dificuldade “Hurt Me Plenty” .....	79
6.5.2	Dano causado por elementos do cenário .....	79
6.5.3	Dano causado nos inimigos.....	79
6.6	Estatísticas de jogo e classificação de jogadores .....	79
6.6.1	Exemplo.....	79
7	Interface .....	81
7.1	HUD (Heads-up Display).....	81
7.1.1	Exibição da vida Marshall.....	82
7.2	Menus .....	82
7.2.1	Menu principal .....	82
7.2.2	Menu 2 .....	82
7.3	Inventário do personagem principal.....	82
8	Músicas e Efeitos Sonoros .....	84
8.1	Músicas .....	84
8.2	Efeitos Sonoros .....	84
9	Inteligência Artificial .....	86
9.1	Shortie .....	86
9.1.1	Movimentação de Ataque/Defesa .....	86
9.1.2	Evento de Acionamento.....	86
9.1.3	Controle do Nível de Dificuldade .....	86
9.2	Screamer .....	87
9.2.1	Movimentação de Ataque/Defesa .....	87
9.2.2	Evento de Acionamento.....	87
9.2.3	Controle do Nível de dificuldade.....	87
9.3	Butcher.....	88
9.3.1	Movimentação de Ataque/Defesa .....	88
9.3.2	Evento de Acionamento.....	88
9.3.3	Controle do Nível de dificuldade.....	88
9.4	Crusher.....	88
9.4.1	Movimentação de Ataque/Defesa .....	88
9.4.2	Evento de Acionamento.....	89
9.4.3	Controle do Nível de dificuldade.....	89
9.5	Chefe final .....	89
9.5.1	Movimentação de Ataque/Esquiva.....	89
9.5.1.1	Golpes com garras.....	89
9.5.1.2	Bolas de Fogo .....	89
9.5.2	Evento de Acionamento.....	90
9.5.3	Controle do Nível de dificuldade.....	90
10	Detalhamento Técnico .....	91
10.1	Requisitos mínimos para execução do jogo.....	91
10.2	Tecnologias Utilizadas.....	91
10.3	Engine.....	91
11	Testes .....	93
11.1	Estratégia de Testes .....	93

11.2	Processo .....	93
11.2.1	Identificação de erros .....	94
11.2.2	Correção de erros.....	94
11.2.3	Re-teste de erros corrigidos.....	94
11.3	Definições.....	95
11.4	Fontes.....	95
12	Produção .....	96
12.1	Milestones do Projeto.....	96
12.2	Listas de Tarefas e Cronograma.....	96
13	Histórico de modificação do documento.....	99

# **Lista de Figuras**

<b>Figura 2.1</b> - Cena do jogo Silent Hill 3, Konami - 2003 .....	3
<b>Figura 2.2</b> - Cena do jogo Silent Hill 3, Konami – 200.....	4
<b>Figura 2.3</b> - Cena do jogo Resident Evil 5, Capcom – 2009 .....	5
<b>Figura 2.4</b> - Cena do jogo Resident Evil 5, Capcom – 2009 .....	5
<b>Figura 2.5</b> - Cena do jogo Fatal Frame 3, Tecmo – 2005.....	6
<b>Figura 2.6</b> - Cena do jogo Fatal Frame 3, Tecmo – 2005.....	6
<b>Figura 2.7</b> - Cenas dos filmes da série Jogos Mortais .....	7
<b>Figura 2.8</b> - Cena de um dos filmes da série Jogos Mortais.....	7
<b>Figura 2.9</b> - Cena do jogo F.E.A.R., Monolith Productions – 2006 .....	8
<b>Figura 2.10</b> - Cena do jogo F.E.A.R., Monolith Productions – 2006 .....	8
<b>Figura 2.11</b> - Cena do jogo Condemned: Criminal Origins , Monolith Productions – 2005 .....	9
<b>Figura 2.12</b> - Cena do jogo Condemned: Criminal Origins , Monolith Productions – 2005 .....	9
<b>Figura 3.1</b> - Marshall Gory, personagem principal do jogo Late Redemption .....	13
<b>Figura 3.2</b> - Personagem Leon do jogo Resident Evil 4, Capcom – 2005 .....	14
<b>Figura 3.3</b> - Personagem Guile do jogo Street Fighter 2, Capcom – 1991 .....	14
<b>Figura 3.4</b> - Personagem Ryu Hazuki do jogo Shenmue, Sega – 1999 .....	15
<b>Figura 3.5</b> - Referência para a arma.....	17
<b>Figura 3.6</b> -Referência para a munição .....	17
<b>Figura 3.7</b> -Referência – Arma em Punho.....	18
<b>Figura 3.8</b> - Shortie, um dos inimigos de Marshall Gory no jogo Late Redemption .....	20
<b>Figura 3.9</b> - Monstro Jogo Doom .....	20
<b>Figura 3.10</b> - Referência para as garras.....	21
<b>Figura 3.11</b> -Referência para as garras .....	21
<b>Figura 3.12</b> - Referência-1 para os dentes.....	21
<b>Figura 3.13</b> - Screamer, um dos inimigos de Marshall Gory no jogo Late Redemption	23
<b>Figura 3.14</b> - Imagem de uma bola de fogo.....	23
<b>Figura 3.15</b> - Imagem do personagem Imp, do jogo Doom .....	24
<b>Figura 3.16</b> - Imagem de um personagem de filme lançando uma bola de fogo .....	24
<b>Figura 3.17</b> - Imagem de do persoangem Dhalsim do jogo Street Fighter .....	24
<b>Figura 3.18</b> - Butcher, um dos inimigos de Marshall Gory no jogo Late Redemption ...	25
<b>Figura 3.19</b> - Imagem de referênciia para o Cutelo .....	26
<b>Figura 3.20</b> - Referência para o Butcher .....	26
<b>Figura 3.21</b> - Referência para o Bucher.....	26
<b>Figura 3.22</b> - Referência para o Butcher .....	27
<b>Figura 3.23</b> - Referência para o Butcher .....	27
<b>Figura 3.24</b> - Crusher, um dos inimigos de Marshall Gory no jogo Late Redemption ...	28
<b>Figura 3.25</b> - Imagem de referênciia para o Tacape .....	28
<b>Figura 3.26</b> - Imagem de Referênciia – Crusher.....	29
<b>Figura 3.27</b> - Imagem do jogo God of War.....	29
<b>Figura 3.28</b> - Imagem de referênciia - Atitude Crusher .....	29
<b>Figura 3.29</b> -Chefe final Diabolus.....	31
<b>Figura 3.30</b> - Cena do jogo Diablo, Blizzard – 1996.....	32
<b>Figura 3.31</b> - Cena do jogo Diablo, Blizzard – 1996.....	32
<b>Figura 3.32</b> - Cena do jogo Jericho, Codemasters – 2005 .....	33
<b>Figura 3.33</b> - Cena do jogo Jericho, Codemasters – 2005 .....	33
<b>Figura 3.34</b> - Cena do jogo Mass Effect, Bioware – 2007.....	34
<b>Figura 3.35</b> - Cena do jogo Mass Effect, Bioware – 2007.....	34
<b>Figura 3.36</b> - Samara, do filme O Chamado .....	35

<b>Figura 3.37</b> - Samara, do filme O Chamado .....	35
<b>Figura 3.38</b> - Silmara, a menina raptada por Diabolus e que deve ser salva por Marshall .....	36
<b>Figura 3.39</b> - Modelo de inspiração para a personagem Silmara .....	37
<b>Figura 4.1</b> - Sequência de fases no jogo Late Redemption .....	38
<b>Figura 4.2</b> - Planta baixa geral da mansão onde o jogo é ambientado.....	39
<b>Figura 4.3</b> - Quadro de distribuição de força com problemas .....	42
<b>Figura 4.4</b> - Referência para o papel com a dica de senha .....	42
<b>Figura 4.5</b> - Referência do visor do dispositivo de segurança do cofre .....	43
<b>Figura 4.6</b> - Referência para o cofre.....	43
<b>Figura 4.7</b> - Referência para a pintura na frente do cofre .....	43
<b>Figura 4.8</b> - Referência para a chave do armário.....	44
<b>Figura 4.9</b> - Referência para o papel com dica para apagar o fogo .....	44
<b>Figura 4.10</b> - Referência para o armário contendo galões de água.....	44
<b>Figura 4.11</b> - Referência para armário vazio .....	45
<b>Figura 4.12</b> - Referência para os galões de água .....	45
<b>Figura 4.13</b> - Referência para a chave da porta do quarto .....	45
<b>Figura 4.14</b> - Referência para a lareira.....	46
<b>Figura 4.15</b> - Referência para o fogo da lareira .....	46
<b>Figura 4.16</b> - Referência para o sofá .....	46
<b>Figura 4.17</b> - Referência para a o cadáver na fase 1 .....	47
<b>Figura 4.18</b> - Referência para o tapete de pele animal que decora a sala .....	47
<b>Figura 4.19</b> - Referência para mesa de centro da sala.....	47
<b>Figura 4.20</b> - Referência para a pintura na frente do cofre .....	47
<b>Figura 4.21</b> - Referência para a janela da sala .....	48
<b>Figura 4.22</b> - Flowchart da fase 1.....	49
<b>Figura 4.23</b> - Planta baixa da fase 1 .....	50
<b>Figura 4.24</b> - Flowchart da fase 2.....	51
<b>Figura 4.25</b> - Planta baixa do cenário Corridor da fase 2.....	52
<b>Figura 4.26</b> - Imagem do corredor de um hotel.....	53
<b>Figura 4.27</b> - Imagem do corredor de um hotel.....	53
<b>Figura 4.28</b> - Planta baixa do cenário Main Hall da fase 2.....	55
<b>Figura 4.29</b> - Imagem do saguão de um hotel .....	55
<b>Figura 4.30</b> - Imagem do saguão de um hotel .....	56
<b>Figura 4.31</b> - Imagem do saguão de um hotel .....	56
<b>Figura 4.32</b> - Imagem do saguão de um hotel .....	57
<b>Figura 4.33</b> - Flowchart da Fase 3 - Library .....	58
<b>Figura 4.34</b> - Planta baixa da Fase 3 - Library .....	58
<b>Figura 4.35</b> - Concepção artística da Fase 3 - Library .....	59
<b>Figura 4.36</b> - Imagem de uma biblioteca do Reino Unido .....	59
<b>Figura 4.37</b> - Imagem de uma biblioteca do Canadá .....	60
<b>Figura 4.38</b> - Imagem de uma biblioteca da França.....	60
<b>Figura 5.1</b> - Transições entre cenas e fases. ....	62
<b>Figura 5.2</b> - Referência - Uma foto de Família .....	63
<b>Figura 5.3</b> - Referência - Foto de Militar .....	64
<b>Figura 5.4</b> - Referência - Quarto vazio (sequestro...).....	64
<b>Figura 5.5</b> - Referência - Funeral/Cemitério .....	64
<b>Figura 5.6</b> - Referência – Desilusão.....	65
<b>Figura 5.7</b> - Referência - Buscando ajuda .....	65
<b>Figura 5.8</b> - Referência – Recomeço .....	65
<b>Figura 5.9</b> - Referência - Investigação de Desaparecimento .....	66
<b>Figura 5.10</b> - Referência - Primeiro contato com a mansão .....	66

<b>Figura 7.1</b> - Elementos do HUD.....	81
<b>Figura 7.2</b> – Inventário de Marshall .....	83
<b>Figura 11.1</b> - Estados das entradas do bugtracking .....	95
<b>Figura 12.1</b> - Cronograma preliminar da fase de desenvolvimento do jogo .....	97

## **Lista de Tabelas**

<b>Tabela 3.1</b> - Dano causado pela arma Glock 34 semi-automática 9 mm .....	17
<b>Tabela 3.2</b> -Características gerais de Marshall Gory.....	18
<b>Tabela 3.3</b> -Características gerais de um Shortie.....	19
<b>Tabela 3.4</b> -Características gerais de um Screamer.....	22
<b>Tabela 3.5</b> -Características gerais de um Butcher.....	25
<b>Tabela 3.6</b> -Características gerais de um Crusher.....	28
<b>Tabela 3.7</b> -Características gerais de Diabolus .....	31
<b>Tabela 6.1</b> -Níveis de dificuldade do jogo Late Redemption .....	77
<b>Tabela 6.2</b> -Exemplo da pontuação final de um jogador .....	80
<b>Tabela 12.1</b> - Milestones do projeto.....	96
<b>Tabela 13.1</b> – Histórico de modificações do documento .....	100

# **Resumo do Documento**

Este documento, em sua versão final, apresenta o Game Design Document (GDD) do jogo Late Redemption. Trata-se de um guia contendo as diretrizes sobre diversos aspectos que foram abordados no desenvolvimento do projeto, tais como: nome do jogo, visão global do jogo, fluxo do jogo, interface do usuário, fases, personagens, armas e itens, músicas e efeitos sonoros, inteligência artificial e a arte do jogo, além de exibir o planejamento com prazos utilizados para o desenvolvimento de cada etapa de produção.

## **1      Nome do Jogo**

Late Redemption

## **2      Visão Global do Jogo**

Este capítulo apresenta uma visão geral do jogo, sendo que a maioria dos itens abordados aqui serão aprofundados nos outros capítulos deste documento.

### **2.1    Características Gerais**

As características gerais do jogo são apresentadas nas seções abaixo.

#### **2.1.1   Estilo**

Survival Horror

#### **2.1.2   Perspectiva**

3ª Pessoa (Shoulder Camera)

#### **2.1.3   Modo**

Single player

#### **2.1.4   Número de fases**

O jogo terá um total de 3 fases

#### **2.1.5   Linha do tempo**

Atual / contemporânea

#### **2.1.6   Idioma do jogo**

Inglês

## **2.1.7 Plataformas**

Windows

## **2.1.8 Gráficos**

3D

## **2.1.9 Principal Forma de Controle**

Uso de teclado e mouse em conjunto.

## **2.2 Filosofia**

Esta seção descreve os principais fatores que nortearam o desenvolvimento do jogo, desde sua concepção até sua finalização.

### **2.2.1 Ponto Filosófico #1**

“Jogo de terror, para jogadores que gostam de tomar sustos, de resolver puzzles e de combates de tiro com inimigos.”

O jogo deve apresentar as 3 características descritas acima integradas, de forma que agrade jogadores desses diversos estilos. A ambientação deve ser uniforme, tal que o jogador não sinta que está jogando 3 jogos diferentes, mas sim um único, com características que se complementam.

### **2.2.2 Ponto Filosófico #2**

“Controle e jogabilidade simples.”

Através de comandos simples de teclado e mouse, o jogador deve executar todas as ações necessárias no jogo. Da mesma forma, o jogador não deve se preocupar em colher itens para a regeneração de pontos de vida do personagem principal, mas apenas em não receber danos em demasiao, proporcionando maior foco nas estratégias do jogo em si, e também na sua apreciação.

### **2.2.3 Ponto Filosófico #3**

“Divertido e gratificante para o grupo de desenvolvimento.”

O desenvolvimento do jogo deve ser desafiador e gratificante ao mesmo tempo para as pessoas da equipe de produção, de forma que elas possam desfrutar de uma ótima experiência de aprendizado e trabalho de equipe, podendo expor suas idéias de maneira livre e fazendo uso de sua criatividade.

## **2.3 Conceito Geral**

Este capítulo descreve de forma resumida o conceito, enredo e personagens do jogo. Os mesmos assuntos serão abordados em mais detalhes em outros capítulos.

### 2.3.1 Descrição do Jogo

Late Redemption é um jogo do estilo Survival Horror, no qual o personagem principal deve cumprir um objetivo que é sobreviver a uma série de ataques de inimigos, solucionar puzzles ou decifrar mistérios.

A ambientação de jogos desse gênero costuma ser sombria, com o objetivo de envolver o jogador completamente, passando a impressão de que está vivendo uma realidade, e com isso, é mais fácil de utilizar mecanismos próprios do estilo para manter presa a atenção do jogador, como sustos, expectativa e até medo.

### 2.3.2 Referências e inspirações

Diversos jogos e até mesmo filmes atuais são ótimos representantes deste gênero, e entre os que serviram de inspiração para o trabalho do grupo estão os listados a seguir.

#### 2.3.2.1 Silent Hill

Silent Hill ([http://en.wikipedia.org/wiki/Silent\\_Hill](http://en.wikipedia.org/wiki/Silent_Hill)) é uma série de jogos de videogame no estilo Survival Horror, desenvolvida e publicada pela Konami. Na série, o jogador controla uma pessoa comum que transita pelos escombros da cidade de Silent Hill. Geralmente a história revela um conflito psicológico envolvendo os personagens principais e a cidade. A **Figura 2.1** e a **Figura 2.2** abaixo mostram cenas do jogo.

**Principal influência:** inspiração como um jogo de terror, com situações de surpresa e sustos.



**Figura 2.1 - Cena do jogo Silent Hill 3, Konami - 2003**  
(fonte:

[http://2.bp.blogspot.com/\\_GGAqK6BvRxk/S7z1LbIqBPI/AAAAAAAEE0/AxH97\\_mb5Ew/s1600/silenthill3\\_screen017%5B1%5D.jpg](http://2.bp.blogspot.com/_GGAqK6BvRxk/S7z1LbIqBPI/AAAAAAAEE0/AxH97_mb5Ew/s1600/silenthill3_screen017%5B1%5D.jpg)



**Figura 2.2** - Cena do jogo Silent Hill 3, Konami – 200.  
(fonte: [http://silenthill.net46.net/web\\_images/silenthill11.jpg](http://silenthill.net46.net/web_images/silenthill11.jpg))

### 2.3.2.2 Resident Evil

Resident Evil ([http://en.wikipedia.org/wiki/Resident\\_Evil\\_%28video\\_game%29](http://en.wikipedia.org/wiki/Resident_Evil_%28video_game%29)) é uma série de jogos de videogame no estilo Survival Horror, desenvolvida e publicada pela Capcom. Na série, o jogador controla um membro de uma força especial (policial ou militar, dependendo do jogo). Geralmente a história evolui em torno do desenvolvimento de armas químicas e das complicações vindas de tal fato, com o surgimento de seres contaminados e mutados por tais armas. A **Figura 2.3** e a **Figura 2.4** abaixo mostram cenas do jogo.

**Principal influência:** inspiração como um jogo em que se deve sobreviver a confrontos com diversos inimigos.



**Figura 2.3** - Cena do jogo Resident Evil 5, Capcom – 2009  
(fonte: <http://www.newmam.blogger.com.br/ResidentEvil5..jpg>)



**Figura 2.4** - Cena do jogo Resident Evil 5, Capcom – 2009  
(fonte: [http://colunistas.ig.com.br/gameover/files/2009/03/resident-evil-5\\_26-07-08\\_01.jpg](http://colunistas.ig.com.br/gameover/files/2009/03/resident-evil-5_26-07-08_01.jpg))

### 2.3.2.3 Fatal Frame

Fatal Frame ([http://en.wikipedia.org/wiki/Fatal\\_Frame](http://en.wikipedia.org/wiki/Fatal_Frame)) é uma série de jogos de videogame no estilo Survival Horror, desenvolvida e publicada pela Tecmo. Seguindo a linha de terror oriental, a série lida com fantasmas, exorcismos e rituais Xintoístas. A **Figura 2.5** e a **Figura 2.6** abaixo mostram cenas do jogo.

**Principal influência:** inspiração como um jogo de terror, com situações de surpresa e sustos.



**Figura 2.5** - Cena do jogo Fatal Frame 3, Tecmo – 2005

(fonte:

[http://3.bp.blogspot.com/\\_hko3jWmZfU8/SwRsNIQsqBI/AAAAAAAAlk/NmlIbKpbfFE/s1600/fatal-frame-iii-the-tormented-20051116105804515%5B1%5D.jpg](http://3.bp.blogspot.com/_hko3jWmZfU8/SwRsNIQsqBI/AAAAAAAAlk/NmlIbKpbfFE/s1600/fatal-frame-iii-the-tormented-20051116105804515%5B1%5D.jpg))



**Figura 2.6** - Cena do jogo Fatal Frame 3, Tecmo – 2005

(fonte:

[http://3.bp.blogspot.com/\\_j3YXdUSqc\\_8/SEvapvULZXI/AAAAAAAABLY/x4wTKRsUXxE/s320/fatal\\_f rame\\_3\\_the\\_tormented\\_002.jpg](http://3.bp.blogspot.com/_j3YXdUSqc_8/SEvapvULZXI/AAAAAAAABLY/x4wTKRsUXxE/s320/fatal_f rame_3_the_tormented_002.jpg))

#### 2.3.2.4 Jogos Mortais

Jogos Mortais é uma série de horror com filmes e jogos de videogame. A série gira em torno do personagem John Kramer, conhecido como “Assassino Jigsaw” ou simplesmente “Jigsaw”. Ao invés de matar suas vítimas de forma direta, Jigsaw as prende em situações que chama de “testes” ou “jogos”, para testar a vontade e determinação das vítimas para sobreviver, através de tortura física ou psicológica. Os testes ou jogos são sempre pensados e executados na forma de enigmas, em que a vítima deve abrir mão de alguma coisa valiosa (muitas vezes uma parte do seu corpo ou até a vida de outra pessoa) para resolver os enigmas, caso contrário ela encontra a morte. A **Figura 2.7** e a **Figura 2.8** abaixo mostram cenas de filmes da série.

**Principal influência:** inspiração para puzzles e situações de pressão.



**Figura 2.7 - Cenas dos filmes da série Jogos Mortais**  
 (fonte: [http://2.bp.blogspot.com/\\_JeRm5ER-vXA/TFC31UgV3iI/AAAAAAAACIO/7Oy58H3F10c/s1600/jogos-mortais-fotos2.jpg](http://2.bp.blogspot.com/_JeRm5ER-vXA/TFC31UgV3iI/AAAAAAAACIO/7Oy58H3F10c/s1600/jogos-mortais-fotos2.jpg))



**Figura 2.8 - Cena de um dos filmes da série Jogos Mortais**  
 (fonte:  
[http://1.bp.blogspot.com/\\_MC9XvQiuf24/TNB5mB12YVI/AAAAAAAAPQI/g\\_PvD7APZSQ/s1600/the-rack.jpg](http://1.bp.blogspot.com/_MC9XvQiuf24/TNB5mB12YVI/AAAAAAAAPQI/g_PvD7APZSQ/s1600/the-rack.jpg))

### 2.3.2.5 Fear

F.E.A.R. (<http://en.wikipedia.org/wiki/F.E.A.R.>), abreviação para First Encounter Assault Recon, é um jogo de tiro em primeira pessoa com temática de horror, desenvolvido pela Monolith Productions e publicado pela Vivendi Universal. A história do jogo gira em torno de um fenômeno sobrenatural, para o qual o time de forças especiais F.E.A.R. é chamado para conter. O jogador assume o papel de um integrante do grupo que possui reflexos super-humanos, e deve descobrir os segredos de uma ameaça sobrenatural na forma de uma pequena garota. A **Figura 2.9** e a **Figura 2.10** abaixo mostram cenas de jogo.

**Principal influência:** inspiração como um jogo de terror, com situações de surpresa e sustos.



**Figura 2.9 - Cena do jogo F.E.A.R., Monolith Productions – 2006**  
(fonte: [http://www.fpssteam.it/img2005/fear/recensione/fear\\_09.jpg](http://www.fpssteam.it/img2005/fear/recensione/fear_09.jpg))



**Figura 2.10 - Cena do jogo F.E.A.R., Monolith Productions – 2006**  
(fonte:  
[http://2.bp.blogspot.com/\\_hDzxow9BwBU/TC3peDuCNOI/AAAAAAAACJ4/\\_2MtF2DiGUG/s1600/ps3\\_fear\\_hi.jpg](http://2.bp.blogspot.com/_hDzxow9BwBU/TC3peDuCNOI/AAAAAAAACJ4/_2MtF2DiGUG/s1600/ps3_fear_hi.jpg))

### 2.3.2.6 Condemned: Criminal Origins

Condemned: Criminal Origins ([http://en.wikipedia.org/wiki/Condemned:\\_Criminal\\_Origins](http://en.wikipedia.org/wiki/Condemned:_Criminal_Origins)) é um videogame de horror psicológico com elementos de combate. O jogo foi desenvolvido pela Monolith Productions e publicado pela Sega. O jogo dá ênfase no combate corpo-a-corpo e na solução de enigmas. Em Condemned, o jogador assume o papel do investigador Ethan Thomas que, acusado pelo assassinato de um policial, tenta solucionar o crime e caçar o assassino em série. Como Ethan, o jogador investiga cenas de crime usando ferramentas e técnicas de investigação forense, enquanto luta com pessoas afetadas por um fenômeno sobrenatural, e tenta sair vivo e se inocentar das acusações. A **Figura 2.11** e a **Figura 2.12** abaixo mostram cenas de jogo.

**Principal influência:** inspiração como um jogo de terror, com situações de surpresa e sustos, além de solução de enigmas.



**Figura 2.11** - Cena do jogo *Condemned: Criminal Origins* , Monolith Productions – 2005  
(fonte: <http://xbox360media.ign.com/xbox360/image/article/659/659056/condemned-criminal-origins-20051017070128260.jpg>)



**Figura 2.12** - Cena do jogo *Condemned: Criminal Origins* , Monolith Productions – 2005  
(fonte:  
[http://1.bp.blogspot.com/\\_jJxCUbR0\\_04/SwkvCO299KI/AAAAAAAAMk/18NmmDhavzc/s1600/s26690\\_pc\\_71.jpg](http://1.bp.blogspot.com/_jJxCUbR0_04/SwkvCO299KI/AAAAAAAAMk/18NmmDhavzc/s1600/s26690_pc_71.jpg))

### 2.3.3 Motivação

O jogo Late Redemption será desenvolvido como trabalho para a disciplina IA369 do primeiro semestre de 2011, cujo objetivo é um exercício prático do projeto de um jogo digital, cobrindo todas as etapas de elaboração do mesmo, desde a sua concepção até sua finalização e acabamento.

A idéia do jogo surgiu de comum acordo entre os membros do grupo, sendo um reflexo dos gostos e interesses compartilhados. A proposta era fazer um jogo que todos gostassem para que pudessem contribuir de maneira espontânea e assim, tornar esta experiência de aprendizado mais proveitosa e interessante para todos. Jogos do estilo Survival Horror, como Silent Hill e Resident Evil, foram a base de influência, porém outros jogos e até outras formas

de entretenimento (filmes, livros) também influenciaram, afinal, faziam parte da vivência dos integrantes do grupo.

### **2.3.4 Enredo**

Marshall Gory atualmente é um Detetive, especialista em seguir pistas de assassinatos e desaparecimento. Sua aparente tranquilidade esconde uma terrível tragédia que mudou sua vida para sempre...

Ainda no início da sua juventude, alistou-se no exército e construiu uma brilhante carreira, sendo plenamente respeitado e admirado pela sua dedicação e competência.

O sucesso não era restrito ao trabalho pois tinha uma linda família e, nada no mundo era capaz de trazer mais alegria do que os momentos ao lado de sua bela esposa e amável filhinha.

Mas em uma noite fria e chuvosa, seu sono e sua paz são interrompidos por um grito desesperado vindo do quarto de sua pequena filha seguido de estranhos grunhidos e gemidos.

Com o coração aos sobressaltos, consegue alcançar a sua arma na penumbra do seu quarto, apenas iluminado pela luz vermelha de seu despertador digital que indicava exatamente três horas e trinta e três minutos da madrugada, e parte em disparada na direção do quarto de sua filha.

Tarde demais... sua vida nunca mais será a mesma e as perguntas “Porque?” e “Como?” jamais abandonarão a sua mente.

A terrível sensação de ter sido incapaz de salvar a própria filha acaba com sua vontade de viver e, Marshall Gory afasta-se de tudo e de todos... transforma-se em um ser errante sem emoções.

Apesar de ele ter desistido de todos, nem todos desistiram dele e, com muita ajuda ele vai conseguindo aos poucos recuperar a coragem de encarar o mundo e finalmente um novo trabalho, desta vez como detetive, o ajuda a olhar para frente e sentir-se vivo.

Sua vida parecia recuperar um rumo até o dia em que uma mulher, desesperada com a foto de uma criança nas mãos, vem a seu escritório após o repentina desaparecimento de sua pequena filha.

Ao ver a angústia na face desta mãe, pela primeira vez em anos, Marshall sente compaixão e, ao olhar para a foto desta menina, o chão parece sumir sob seus pés... Como poderiam ser tão parecidas?

Estaria o destino dando-lhe uma segunda chance ou estaria apenas pregando-lhe uma peça para acabar de vez com sua vida?

Sua obsessão pelo caso o leva à uma sombria mansão, onde ele espera deparar-se com a origem dos crimes e as respostas para as perguntas que tanto o atormentam.

Inicia-se então uma corrida contra o tempo para encontrar a menina desaparecida e com isso alcançar a sua própria redenção... ou não...

### **2.3.5 Ambientes a serem simulados**

O interior de uma mansão mal-assombrada. Cada fase do jogo se passará em um ambiente específico da mansão, sendo eles:

- Primeira fase: uma sala fechada, onde o jogador deve resolver diversos puzzles. Este quarto conterá elementos de cenário como portas, janelas, sofá, armários, entre outros.
- Segunda fase: um corredor amplo da mansão seguido de um grande saguão. O jogador deve avançar a partir do quarto da primeira fase em direção à biblioteca da terceira fase, eliminando qualquer inimigo ou obstáculo que tente impedir seu progresso.
- Terceira fase: a biblioteca da mansão, contendo mesas para leitura, dois andares repletos de prateleiras com livros, escadas em círculo para acesso.

Cada ambiente será descrito com mais detalhes no capítulo 4 - **Fases**.

### **2.3.6 Personagem controlado pelo jogador**

Marshall Gory é o personagem principal do jogo. Detetive ex-militar com uma mente aguçada, raciocínio rápido e grande habilidade para colher pistas e resolver crimes. Atormentado pela trágica morte de sua única filha, procura em seu trabalho uma maneira de reencontrar paz de espírito.

O personagem principal é descrito em detalhes no capítulo 3 - **Personagens**.

### **2.3.7 Personagens controlados pelo computador**

Os personagens controlados pelo computador são descritos em detalhes no capítulo 3 - **Personagens**.

#### **2.3.7.1 Inimigos**

Criaturas sobrenaturais demoníacas que tentam impedir o avanço de Marshall pelos cenários da segunda fase. São: Shorties, Screamers, Butchers e Crushers.

#### **2.3.7.2 Chefe Final**

Misterioso chefe final, Diabolus é o responsável pelo desaparecimento da vítima e pelos assassinatos em série.

#### **2.3.7.3 Silmara**

Sequestrada pelo Chefe Final, corre risco de morte, e a missão de Marshall é resgatá-la.

### **2.3.8 Principal objetivo do jogo**

Sobreviver às situações apresentadas pelo jogo, fugindo da mansão mal-assombrada, derrotando os inimigos e salvando a vítima.

# **3 Personagens**

Este capítulo descreve em detalhes os personagens do jogo, sendo estes classificados em personagem principal, inimigos e chefe final. Serão encontradas aqui informações sobre a descrição dos personagem, concepções artísticas e características do personagem importantes para a mecânica do jogo.

## **3.1 Personagem Principal**

Nesta seção serão encontradas informações tanto do personagem principal como da arma utilizada por ele.

### **3.1.1 Descrição Física e Psicológica**

O personagem principal será controlado pelo jogador. Seu nome é Marshall Gory e ele é um homem com idade em torno dos 35 anos. Possui 1,80 metro de altura e pesa 85 quilos. Possui cabelos cor castanho escuro e curtos, quase raspados devido ao estilo de vida adquirido no tempo que era um militar. Usa uma camiseta branca desbotada com aspecto de velha junto com um camisa preta e uma jaqueta de couro marrom. Utiliza um cinto com uma fivela de metal acinzentado e veste calças de cor verde musgo. Calça um coturno preto. Possui personalidade forte e é muito céptico em relação a assuntos que não podem ser explicados. É amargurado devido a morte da filha. Sempre que está envolvido em algum caso anda munido de uma arma Glock modelo 34, calibre de 9 mm.

### **3.1.2 Concepção artística**

A arte do personagem principal foi concebida baseando-se nos personagens Leon do jogo Resident Evil 4 e Guile do jogo Street Fighter. O detalhe da jaqueta de couro marrom foi baseada na jaqueta do personagem Ryu Hazuki do jogo Shenmue, conforme referências a seguir.



**Figura 3.1** - Marshall Gory, personagem principal do jogo Late Redemption

### 3.1.3 Imagens de referência



**Figura 3.2** - Personagem Leon do jogo Resident Evil 4, Capcom – 2005

(fonte:

[http://www.ugo.com/ugo/html/gallery/?id=51&gallery=residentevil4characters\\_games](http://www.ugo.com/ugo/html/gallery/?id=51&gallery=residentevil4characters_games))



**Figura 3.3** - Personagem Guile do jogo Street Fighter 2, Capcom – 1991

(fonte:

<http://surbrook.devermore.net/adaptationsvideogame/streetfighter/sfquile.html>)



**Figura 3.4 - Personagem Ryu Hazuki do jogo Shenmue, Sega – 1999**  
(fonte: <http://www.segabits.com/?p=5399>)

### 3.1.4 Arma do personagem principal

Atualmente, a melhor amiga de Marshall é sua arma que o acompanha onde quer que ele vá. Sendo ele um ex-militar, isto não é problema uma vez que ele possui porte de arma e, agora na nova fase de sua vida, trabalhando como detetive particular especialista em assassinatos e desaparecimentos, uma arma bem balanceada torna-se fundamental.

#### 3.1.4.1 Descrição física

A arma de Marshall é uma série especial desenvolvida para o exército, baseada em uma Glock 34, de calibre 9 mm.

Esta é uma arma semi-automática. Quando um tiro é disparado, automaticamente o próximo projétil disponível no pente é carregado na agulha e fica pronto para o próximo disparo. A capacidade da arma é de 13 balas.

#### 3.1.4.2 Tipos de munição

No que diz respeito aos itens vinculados às armas, balas de dois tipos diferentes e em quantidades variadas estarão espalhadas pelo jogo.

Quanto aos dois tipos distintos de munição, teremos: a munição padrão, do mesmo tipo daquela que Marshall normalmente usa em sua arma, e uma munição especial, que contém pontas de prata cujo dano causado aos monstros será o dobro daquele causado pela munição padrão.

Esta munição extra está intimamente associada aos corpos encontrados pela casa pois estes na maioria dos casos pertenciam àqueles que antes de Marshall, também foram atraídos para a mansão mas não sobreviveram para contar a sua experiência.

Conforme será descrito na seção **6.3 - Níveis de Dificuldade**, a quantidade de munição disponível pela casa varia de acordo com a dificuldade selecionada ao início do jogo: em

ambos os níveis de dificuldade, a munição será ilimitada, pois o item representando a munição reaparecerá periodicamente; o que varia de um nível para o outro é a quantidade de locais nos quais as munições podem ser encontradas e o tempo que leva até a munição reaparecer. A quantidade de munição especial também é menor no nível de dificuldade “Hurt Me Plenty”, além de não ser possível obtê-la na fase 3, enquanto isto é possível no nível “Go Easy On Me”.

Sempre que Marshall avistar o item correspondente à munição, ele terá a oportunidade de pegá-lo, a partir do comando correspondente à ação de pegar objetos.

No início do jogo, a arma estará completamente carregada e Marshall ainda terá um pente extra, tendo, portanto, 26 balas.

### **3.1.4.3    Dinâmica de uso da arma**

#### **Recarga da arma**

Mesmo que o pente não esteja completamente vazio, o jogador poderá ter a opção de recarregar a sua arma a qualquer momento (veja as seções **6.1 - Jogabilidade** e **6.2 - Controles** para o comando associado com a recarga).

O tempo de recarga será sempre o mesmo, por volta de 2 segundos e, durante este tempo, Marshall estará vulnerável aos ataques dos monstros.

Um máximo de 52 balas de cada tipo podem ser carregados por Marshall em um dado momento, sendo que, a quantidade atualmente carregada no pente é levada em conta neste cálculo. Basicamente, a quantidade máxima de balas que Marshall pode carregar será o suficiente para preencher 4 pentes de cada tipo de munição.

Caso seja encontrado um item correspondente aos projéteis cuja quantidade ultrapasse este limite, este não poderá ser captado e a mensagem *Not Enough Room* deverá ser mostrada.

#### **Disparos**

Durante a primeira fase, podem ocorrer disparos, mas, estes disparos não interagem com nenhum elemento encontrado dentro da sala onde os puzzles deverão ser resolvidos.

O único efeito prático que é obtido quando tiros são disparados nesta fase é a diminuição da quantidade de munição disponível para Marshall enfrentar os primeiros inimigos assim que ele sair da sala. Já na segunda e terceira fases, Marshall está livre para disparar seus projéteis na direção de todos os monstros que cruzem o seu caminho. É permitido mirar tanto em posição horizontal como vertical. Sendo uma arma semi-automática, a cada comando de atirar executado pelo jogador, um novo tiro será disparado.

### **3.1.4.4    Danos causados pelos tiros.**

Desde que um tiro disparado pela arma de Marshall acerte um monstro, independente da região do seu corpo que este projétil atinja, o dano causado sempre levará à mesma diminuição no valor dos pontos de vida do monstro – ou seja, indiferente do local onde o tiro atinge o monstro, o dano será constante.

Como referência para o cálculo dos pontos de dano, a tabela abaixo deverá ser usada e, combinada com o valor total dos pontos de vida de cada monstro:

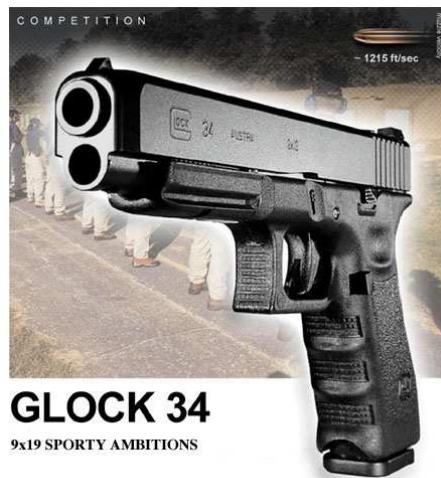
Tipo de munição	Dano causado
Normal	20 Pontos de Vida
Especial	40 Pontos de Vida

*Tabela 3.1 - Dano causado pela arma Glock 34 semi-automática 9 mm*

### 3.1.4.5 Animações associadas

- Enquanto a arma não estiver em punho, ela estará guardada no coldre escondido dentro da jaqueta de Marshall.
- Sempre que o comando de recarga da arma for acionado, uma animação mostrando Marshall carregando sua arma será mostrado.
- Sempre que o tipo de munição for modificado, a animação de recarga deverá ser acionada pois, a modificação do tipo de munição implica na troca do pente.

### 3.1.4.6 Imagens de Referência



*Figura 3.5 - Referência para a arma  
(fonte:  
<http://dynamicarmament.com/images/glockmag/Glock%2033RD%20Mag.jpg>)*



*Figura 3.6 -Referência para a munição  
(fonte: <http://www.baixaki.com.br/imagens/62504/64061.jpg>)*



**Figura 3.7 -Referência – Arma em Punho**  
(fonte: [http://ep.yimg.com/ca/I/artoftoy\\_2147\\_138532890](http://ep.yimg.com/ca/I/artoftoy_2147_138532890))

### 3.1.5 Características Gerais

A tabela abaixo exibe características gerais de Marshall Gory. Algumas características variam de acordo com o nível de dificuldade do jogo, o que está representado pelas colunas “Go Easy on Me” e “Hurt Me Plenty”:

Característica x Dificuldade	Go Easy On Me	Hurt Me Plenty
Pontos de vida	100	85
Recuperação de danos recebidos	Recuperação automática de pontos de vida (10 pontos de vida recuperadas a cada 1 segundo após ficar 5 segundos sem tomar dano)	Recuperação automática de pontos de vida (3 pontos de vida recuperadas a cada 1 segundo após ficar 15 segundos sem tomar dano)
Forma de ataque	Glock 34 semi-automática, de calibre 9 mm	Glock 34 semi-automática, de calibre 9 mm
Tipo de dano	Perfurante	Perfurante
Valor de dano da munição normal	20	20
Valor de dano da munição normal	40	40
Alcance do ataque	15 metros (longa distância)	15 metros (longa distância)
Velocidade do ataque	Aproximadamente 3 disparos a cada segundo	Aproximadamente 3 disparos a cada segundo
Quantidade inicial de munição	26 balas normais	26 balas normais
Quantidade máxima de munição	52 balas normais e 52 balas especiais	52 balas normais e 52 balas especiais

**Tabela 3.2 -Características gerais de Marshall Gory**

## 3.2 Inimigos

Os personagens controlados pelo computador que atacarão Marshall Gory serão inimigos que foram criados e modelados a partir da idéia de criaturas sobrenaturais demoníacas, ou seja, monstros. Incentivou-se dentro do grupo a criação de inimigos de diferentes tamanhos e formas para a maior diversificação de seus comportamentos e ataques no jogo.

Dos inimigos encontrados na Fase 2, temos Shortie, Screamer, Butcher e Crusher.

### 3.2.1 Shortie

A menor criatura do jogo, Shortie não é o inimigo mais ameaçador quando se aventura solitário pelos cantos da Mansão. Mas não se pode dizer o mesmo ao encontrar um grupo deles, eles sempre andam em grupos de 3, representando uma ameaça que deve ser cuidadosamente considerada. Essas criaturas são perversos pequenos demônios que adoram devorar qualquer um que cruze o seu caminho. Podem ser encontrados perto de cadáveres, pois possuem uma fome gigantesca, e não descansam até se darem por satisfeitos.

#### 3.2.1.1 Descrição Física

Com aproximadamente 70 cm de tamanho, Shortie tem uma mandíbula forte com uma boca grande e dentes afiados. Toda a parte superior de seu corpo é grande em comparação com o restante do seu corpo, evidenciando a especialização da criatura em seu ataque frontal, que nada mais é do que dar mordidas, juntamente com golpes desferidos por seus braços troncudos e suas garras pontudas. Shortie tem sua pele enrugada e rosada, e não possui olhos, por ser capaz de identificar uma presa pelo cheiro.

#### 3.2.1.2 Características Gerais

A tabela abaixo exibe características gerais de um Shortie. Algumas características variam de acordo com o nível de dificuldade do jogo, o que está representado pelas colunas "Go Easy on Me" e "Hurt Me Plenty":

Característica x Dificuldade	Go Easy On Me	Hurt Me Plenty
Pontos de vida	60	80
Forma de ataque	Golpes com garras e mordidas	Golpes com garras e mordidas
Tipo de dano	Cortante	Cortante
Valor de dano	4	5
Alcance do ataque	1 metro (curta distância)	1 metro (curta distância)
Velocidade do ataque	1 a cada segundo	1 a cada 0.8 segundo
Comportamento de ataque	Veja Capítulo 9.1	Veja Capítulo 9.1
Velocidade de movimentação (relação ao Marshall)	120%	150%

Tabela 3.3 -Características gerais de um Shortie

### 3.2.1.3 Concepção Artística

A figura abaixo exibe a concepção artística do Shortie:



*Figura 3.8 - Shortie, um dos inimigos de Marshall Gory no jogo Late Redemption*

### 3.2.1.4 Imagens de Referência



*Figura 3.9 - Monstro Jogo Doom*

(fonte: <http://forums.gametrailers.com/thread/gears-of-war-is-so-original-/520921>)



**Figura 3.10** - Referência para as garras  
(fonte: <http://spbynights.files.wordpress.com/2009/10/garras.jpg?w=320&h=286>)



**Figura 3.11** -Referência para as garras  
(fonte: <http://www.maskworld.com/pix/masks/r963-monster-klauen-gruen-monster-claws-green.jpg>)



**Figura 3.12**- Referência-1 para os dentes  
(fonte:  
[http://diholinda.files.wordpress.com/2008/09/monstro\\_dentes\\_garras.jpg?w=341&h=355](http://diholinda.files.wordpress.com/2008/09/monstro_dentes_garras.jpg?w=341&h=355))

### 3.2.2 Screamer

Vagando pelos cômodos da Mansão, Screamer é solitário, talvez pelo fato de não possuir a parte de baixo do corpo. Ele é uma criatura que flutua e tem ataques sobrenaturais através do seu grito aterrorizador. Nascido do submundo, seu único objetivo é amaldiçoar qualquer alma que se encontrar com ele.

### **3.2.2.1 Descrição Física**

O Screamer tem a aparência de um ser humano grande pálido e sombrio. Seu corpo é magro e possui braços anormalmente longos e finos, do mesmo modo que sua cabeça, e não possui a parte de baixo da cintura. Também sem olhos, Screamer possui uma boca comprida e maior que a do Shortie, porém é usada para atacar o personagem principal com uma bola de fogo junto ao seu grito.

### **3.2.2.2 Características Gerais**

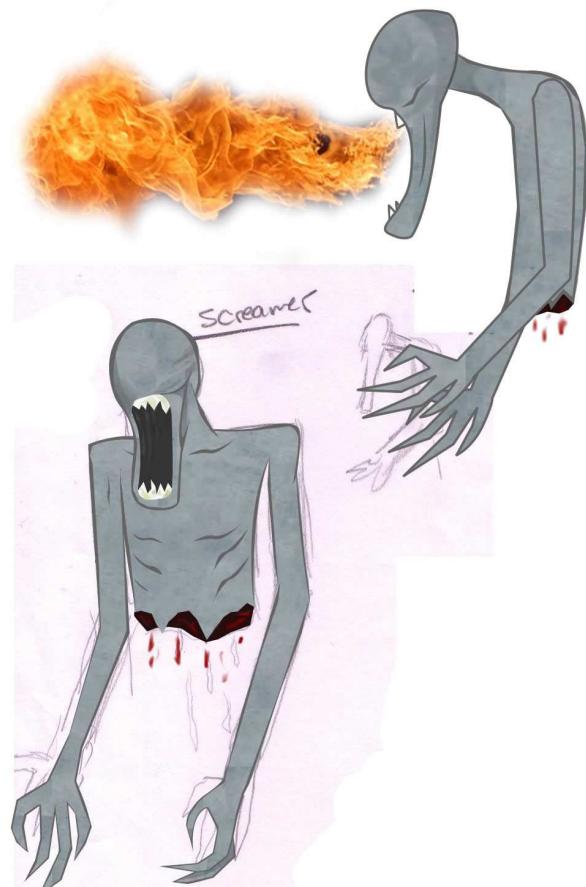
A tabela abaixo exibe características gerais de um Screamer. Algumas características variam de acordo com o nível de dificuldade do jogo, o que está representado pelas colunas “Go Easy on Me” e “Hurt Me Plenty”:

<b>Característica/Dificuldade</b>	<b>Go Easy On Me</b>	<b>Hurt Me Plenty</b>
Pontos de vida	100	120
Forma de ataque	Bolas de fogo	Bolas de fogo
Tipo de dano	Dano por elemental	Dano por elemental
Valor de dano	20	25
Alcance do ataque	10 metros (longa distância)	10 metros (longa distância)
Velocidade do ataque	1 a cada 3 segundos	1 a cada 2.5 segundos
Comportamento de ataque	Veja Capítulo 9.2	Veja Capítulo 9.2
Velocidade de movimentação (relação ao Marshall)	130%	160%

*Tabela 3.4 -Características gerais de um Screamer*

### **3.2.2.3 Concepção artística**

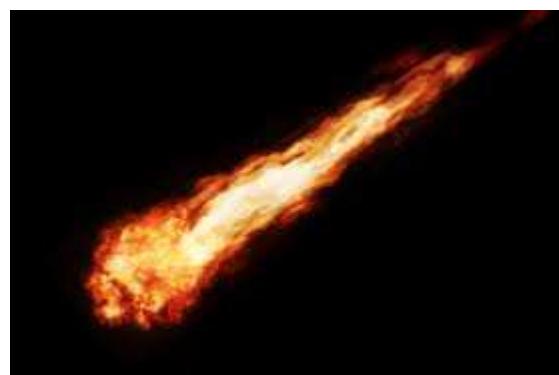
A figura abaixo exibe a concepção artística do Screamer:



**Figura 3.13** - Screamer, um dos inimigos de Marshall Gory no jogo Late Redemption

#### 3.2.2.4    Imagens de Referência

Como base para o visual e os ataques do personagem inimigo Screamer, os jogos da série Doom, Street Fighter e alguns personagens de filmes foram usados como referência. As figuras a seguir ilustram tais influências:



**Figura 3.14** - Imagem de uma bola de fogo.  
(fonte:

<http://my.opera.com/AOTEAROAnz/albums/showpic.dml?album=611050&picture=8910743>)



**Figura 3.15** - Imagem do personagem Imp, do jogo Doom  
(fonte: [http://www.giantbomb.com/doom/61-7326/all-images/52-179317/6108\\_doom2/51-215270](http://www.giantbomb.com/doom/61-7326/all-images/52-179317/6108_doom2/51-215270))



**Figura 3.16** - Imagem de um personagem de filme lançando uma bola de fogo  
(fonte: [http://images.wikia.com/farscape/images/a/a0/Sheyana\\_Fireball.jpg](http://images.wikia.com/farscape/images/a/a0/Sheyana_Fireball.jpg))



**Figura 3.17** - Imagem do personagem Dhalsim do jogo Street Fighter  
(fonte: [http://wikicheats.gametrailers.com/Image:SFIV\\_screenDhalsim03.jpg](http://wikicheats.gametrailers.com/Image:SFIV_screenDhalsim03.jpg))

### 3.2.3 Butcher

O Butcher é o inquisidor dentro do grupo de monstros e é ele que decide quem deve passar por ele. Forte e imponente, com sua única arma, o cutelo, consegue barrar a passagem de qualquer um.

#### 3.2.3.1 Descrição Física

Com quase 2 metros de altura e obeso, tem um porte grande e forte. O Butcher não possui cabeça (ela foi decepada) e usa um avental, luvas e botas de açougueiro sujas de sangue. Além disso, possui quatro patas gigantescas de inseto, rígidas e pontiagudas saíndo de suas costas.

### 3.2.3.2 Características Gerais

A tabela abaixo exibe características gerais de um Butcher. Algumas características variam de acordo com o nível de dificuldade do jogo, o que está representado pelas colunas “Go Easy on Me” e “Hurt Me Plenty”:

Característica x Dificuldade	Go Easy On Me	Hurt Me Plenty
Pontos de vida	200	240
Forma de ataque	Golpes com cutelo	Golpes com cutelo
Tipo de dano	Cortante	Cortante
Valor de dano	30	35
Alcance do ataque	1,5 metro (curta distância)	1,5 metro (curta distância)
Velocidade do ataque	1 a cada segundo	1 a cada 0.8 segundo
Comportamento de ataque	Veja Capítulo 9.3	Veja Capítulo 9.3
Velocidade de movimentação (relação ao Marshall)	50% (Normal) / 150% (Investida)	60% (Normal) / 180% (Investida)

*Tabela 3.5 -Características gerais de um Butcher*

### 3.2.3.3 Concepção Artística

A figura abaixo exibe a concepção artística do Butcher:



*Figura 3.18 - Butcher, um dos inimigos de Marshall Gory no jogo Late Redemption*

### 3.2.3.4 Imagens de Referência



**Figura 3.19 - Imagem de referência para o Cutelo**  
(fonte: [http://www.magazinedacasa.com.br/Imagens/produtos/00/006-0134-000-000/006-0134-000-000\\_Ampliada.jpg](http://www.magazinedacasa.com.br/Imagens/produtos/00/006-0134-000-000/006-0134-000-000_Ampliada.jpg))



**Figura 3.20 - Referência para o Butcher**  
(fonte: <http://battlemind.files.wordpress.com/2009/12/ogre-kingdom-butcher.jpg>)



**Figura 3.21 - Referência para o Bucher**  
(fonte:  
[http://2.bp.blogspot.com/\\_8L7\\_ydYa4BY/SsYMZusTOZI/AAAAAAAeU/r24GzLWW2ug/s1600-h/qd2.JPG](http://2.bp.blogspot.com/_8L7_ydYa4BY/SsYMZusTOZI/AAAAAAAeU/r24GzLWW2ug/s1600-h/qd2.JPG))



**Figura 3.22 - Referência para o Butcher**  
(fonte: <http://www.myfreewallpapers.net/games/pages/splatterhouse-original.shtml>)



**Figura 3.23 - Referência para o Butcher**  
(fonte: [http://www.profantasy.com/images/cc3/butcher\\_demon.gif](http://www.profantasy.com/images/cc3/butcher_demon.gif))

### 3.2.4 Crusher

Crusher é a personificação da força bruta, dentro do grupo de inimigos do jogo. De força descomunal, nenhum oponente que ousa enfrentá-lo sobrevive; nem mesmo outros demônios ousam desafiá-lo ou contrariá-lo. Por isso, ocupa uma posição de general dentre as forças do chefe final Diabolus, respondendo diretamente a este.

#### 3.2.4.1 Descrição Física

Com 2 metros e meio de altura, tem um porte grande e forte, requisito necessário para manipular sua arma preferida, um tacape gigante e quase tão grande quanto o dono. Tal arma é extremamente mortal nas mãos do Crusher, não precisando de mais do que 2 ou 3 golpes para matar um adversário comum. Porém, esse poder todo tem um custo, velocidade. Crusher se desloca de forma lenta, tanto por causa de seu tamanho, quanto pelo peso de sua arma. Mas esse demônio perigoso tem um trunfo para suprir tal desvantagem: ao atacar com golpes de seu tacape, a força e impulso empregados são tão grandes, que deslocam o gigante vários metros à frente, tornando uma tarefa mais difícil ainda escapar de suas investidas.

#### 3.2.4.2 Características Gerais

A tabela abaixo exibe características gerais de um Crusher. Algumas características variam de acordo com o nível de dificuldade do jogo, o que está representado pelas colunas “Go Easy on Me” e “Hurt Me Plenty”:

Característica/Dificuldade	Go Easy On Me	Hurt Me Plenty
----------------------------	---------------	----------------

Pontos de vida	400	500
Forma de ataque	Golpes com tacape	Golpes com tacape
Tipo de dano	Contusão	Contusão
Valor de dano	35	40
Alcance do ataque	2,5 metro (curta distância)	2,5 metro (curta distância)
Velocidade do ataque	1 a cada segundo	1 a cada 0.8 segundo
Comportamento de ataque	Veja Capítulo 9.4	Veja Capítulo 9.4
Velocidade de movimentação (relação ao Marshall)	80%	100%

*Tabela 3.6 -Características gerais de um Crusher*

### 3.2.4.3 Concepção Artística

A figura abaixo exibe a concepção artística do Crusher:



*Figura 3.24 - Crusher, um dos inimigos de Marshall Gory no jogo Late Redemption*

### 3.2.4.4 Imagens de referência



*Figura 3.25 - Imagem de referência para o Tacape  
(fonte:*

[http://4.bp.blogspot.com/\\_DLtasQWmtko/S19fEwunnVI/AAAAAAAANA/a7OXvlotGuo/s1600-h/Tacape+de+Guerra.jpg](http://4.bp.blogspot.com/_DLtasQWmtko/S19fEwunnVI/AAAAAAAANA/a7OXvlotGuo/s1600-h/Tacape+de+Guerra.jpg)



**Figura 3.26 - Imagem de Referência – Crusher**  
(fonte:[http://animewallpapers.it/Fantasy/Fantasy-art-series/MINOTAUR-CJC-FI-64801p.html/\(mode\)/search/\(keyword\)/Warhammer+wood+elves/\(sort\)/relevanceasc](http://animewallpapers.it/Fantasy/Fantasy-art-series/MINOTAUR-CJC-FI-64801p.html/(mode)/search/(keyword)/Warhammer+wood+elves/(sort)/relevanceasc))



**Figura 3.27 - Imagem do jogo God of War**  
(fonte:  
[http://t3.gstatic.com/images?q=tbn:ANd9GcQ7LeOGh9KK8cWn2QUe4kf4cN\\_kUnFEHj\\_JPp2bB3nwTcBdeDHrQA&t=1](http://t3.gstatic.com/images?q=tbn:ANd9GcQ7LeOGh9KK8cWn2QUe4kf4cN_kUnFEHj_JPp2bB3nwTcBdeDHrQA&t=1))



**Figura 3.28 - Imagem de referência - Atitude Crusher**  
(fonte: <http://www.ps3vault.com/wp-content/uploads/2010/12/Splatterhouse-Tease.jpg>)

### 3.3 Chefe final - Diabolus

Diabolus (diabo, em Latim - <http://pt.wikipedia.org/wiki/Diabo>) é o chefe final do jogo. Maquiavélico e manipulador, é ele o responsável por toda a série de acontecimentos que trazem Marshall Gory até a Mansão em que se passa o jogo. Como um demônio de grande poder, Diabolus sabe do trauma que marcou a vida de Marshall, e usa esse fato para atraí-lo e capturar a sua alma, usando a menina Silmara como isca.

#### 3.3.1 Descrição Física

Como um demônio de imenso poder, Diabolus pode assumir a forma física que quiser. No decorrer do jogo, ele assume duas formas. A primeira delas é a da menina Silmara, para atrair Marshall Gory até seus domínios. Porém, essa forma assumida é maculada pelo mal, o que afeta os seus atributos físicos, acrescentando à imagem pura da menina olheiras, cabelos despenteados e sujos, vestido rasgado e semblante aterrorizador.

Em sua verdadeira forma, Diabolus possui quase 15 metros de altura. Seu grande porte físico, juntamente com seu poder, o fazem dominar outras criaturas demoníacas mais fracas que ele (caso dos outros inimigos do jogo). Com grandes garras, chifres que saem de suas costas, dentes afiados e olhos incandescentes, é uma figura amedrontadora. No centro do seu peito encontra-se a fonte de seu poder: seu coração, feito das mais puras energias negativas. Entretanto, ao mesmo tempo que é a fonte de seu poder, também é seu único ponto fraco. Destruindo seu coração, destrói-se também Diabolus.

#### 3.3.2 Características Gerais

A tabela abaixo exibe características gerais de Diabolus. Algumas características variam de acordo com o nível de dificuldade do jogo, o que está representado pelas colunas “Go Easy on Me” e “Hurt Me Plenty”:

Característica x Dificuldade	Go Easy On Me	Hurt Me Plenty
Pontos de vida da proteção no peito	100 x 5 (Será necessário quebrar a proteção no mínimo 5 vezes para matar o chefe)	200 x 5 (Será necessário quebrar a proteção no mínimo 5 vezes para matar o chefe)
Pontos de vida do coração	500	1000
Forma de ataque 1	Golpes com garras	Golpes com garras
Tipo de dano 1	Cortante	Cortante
Valor de dano 1	20 x 2	25 x 2
Alcance do ataque 1	4 metros (curta distância)	4 metros (curta distância)
Velocidade do ataque 1	1 a cada segundo	1 a cada 0.8 segundo
Forma de ataque 2	Bolas de fogo	Bolas de fogo
Tipo de dano 2	Dano por elemental	Dano por elemental

Valor de dano 2	25 x 3	25 x 3
Alcance do ataque 2	12 metros (longa distância)	12 metros (longa distância)
Velocidade do ataque 2	1 a cada segundo	1 a cada 0.8 segundo
Comportamento de ataque	Veja Capítulo 9.5	Veja Capítulo 9.5
Velocidade de movimentação (relação ao Marshall)	80%	100%

*Tabela 3.7 -Características gerais de Diabolus*

### 3.3.3 Concepção Artística

A **Figura 3.29** mostra a arte que representa Diabolus:



*Figura 3.29 -Chefe final Diabolus*

### 3.3.4 Imagens de referência

#### 3.3.4.1 Diablo

Diablo ([http://en.wikipedia.org/wiki/Diablo\\_%28video\\_game%29](http://en.wikipedia.org/wiki/Diablo_%28video_game%29)) é um jogo de RPG de ação desenvolvido e publicado pela Blizzard. Ambientado no reino fictício de Khanduras, no mundo de Sanctuary, Diablo situa o jogador no controle de um herói solitário em uma luta

para derrotar o Lorde do Terror, Diablo. Nos subterrâneos da cidade de Tristam, o jogador deve superar 16 níveis de calabouços e catacumbas, até que finalmente chegue ao próprio inferno para enfrentar Diablo. A **Figura 3.30** e **Figura 3.31** mostram cenas do jogo.

**Principais influências:** inspiração para o chefe final, regras de combate e dano, textos a serem usados nos puzzles



*Figura 3.30 - Cena do jogo Diablo, Blizzard – 1996  
(fonte: <http://s.glbimg.com/po/tt/f/original/2010/12/22/diablo1.gif>)*



*Figura 3.31 - Cena do jogo Diablo, Blizzard – 1996  
(fonte: [http://api.ning.com/files/9\\*In8TNG2ONbx-dnb1Fcj\\*ARxX-UETqFaiUnUOuGBcV2Nmt40SmA-wh7MG0fFtx9mI2wqhFJC8zYENvCm-SpER67fMUy9ags/screendiablo2.jpg](http://api.ning.com/files/9*In8TNG2ONbx-dnb1Fcj*ARxX-UETqFaiUnUOuGBcV2Nmt40SmA-wh7MG0fFtx9mI2wqhFJC8zYENvCm-SpER67fMUy9ags/screendiablo2.jpg))*

### 3.3.4.2 Jericho

Jericho ([http://en.wikipedia.org/wiki/Clive\\_Barker's\\_Jericho](http://en.wikipedia.org/wiki/Clive_Barker's_Jericho)) é um videogame de tiro em primeira pessoa com temática de horror. O jogo foi desenvolvido pela MercurySteam e publicado pela Codemasters. A história do jogo gira em torno da primeira criação de Deus antes de Adão e Eva, o chamado Firstborn. Como resultado imperfeito da criação, o Firstborn é aprisionado numa realidade alternativa, de onde tenta escapar. Cabe ao esquadrão Jericho manter a criatura aprisionada. A **Figura 3.32** e **Figura 3.33** mostram cenas do jogo.

**Principal influência:** inspiração para o chefe final.



**Figura 3.32** - Cena do jogo Jericho, Codemasters – 2005  
(fonte: <http://www.geek.com/wp-content/uploads/2007/10/cbj1.jpg>)



**Figura 3.33** - Cena do jogo Jericho, Codemasters – 2005  
(fonte: [http://www.joblo.com/images\\_arrownews/jericho\\_4.jpg](http://www.joblo.com/images_arrownews/jericho_4.jpg))

### 3.3.4.3 Mass Effect

Mass Effect ([http://en.wikipedia.org/wiki/Mass\\_Effect](http://en.wikipedia.org/wiki/Mass_Effect)) é um jogo de RPG de ação desenvolvido pela BioWare. O jogo se passa no ano de 2183, e o jogador assume o papel de um soldado de elite humano chamado Comandante Shepard, que se prepara para explorar a galáxia numa espaçonave, a SSV Normandy. As **Figura 3.34** e **Figura 3.35** mostram cenas do jogo.

**Principal influência:** inspiração para o chefe final.



*Figura 3.34 - Cena do jogo Mass Effect, Bioware – 2007*

(fonte:

[http://th08.deviantart.net/fs71/PRE/i/2010/358/c/2/shadow\\_broker\\_wallpaper\\_by\\_michawha-d35k4h5.jpg](http://th08.deviantart.net/fs71/PRE/i/2010/358/c/2/shadow_broker_wallpaper_by_michawha-d35k4h5.jpg))



*Figura 3.35 - Cena do jogo Mass Effect, Bioware – 2007*

(fonte: [http://media.giantbomb.com/uploads/0/5911/1515675-broker\\_shields\\_up\\_super.jpg](http://media.giantbomb.com/uploads/0/5911/1515675-broker_shields_up_super.jpg))

### 3.3.4.4 Samara, do filme O Chamado



**Figura 3.36 - Samara, do filme O Chamado**  
(fonte: <http://www.popmag.com.br/posts/001/halloween-04.jpg>)



**Figura 3.37 - Samara, do filme O Chamado**  
(fonte: [http://3.bp.blogspot.com/\\_xvXuYY2JA4/TU92wtAoFUI/AAAAAAAUA4/zf\\_eNohb3OQ/s1600/kayako\\_saeki\\_in\\_the\\_grudge\\_21.png](http://3.bp.blogspot.com/_xvXuYY2JA4/TU92wtAoFUI/AAAAAAAUA4/zf_eNohb3OQ/s1600/kayako_saeki_in_the_grudge_21.png))

## 3.4 NPC's

Os NPC's, ou Non-Player Character, são os personagens existentes no jogo que não são controlados pelo jogador, porém também não inimigos.

### 3.4.1 Silmara

Silmara é a menina que foi raptada por Diabolus com o propósito de atrair Marshall Gory para capturar a sua alma. Uma criança indefesa, não tem como lutar ou se defender, permanecendo presa em uma jaula presa no teto na Fase 3, Library.

Representa um apelo muito forte para o personagem principal, uma vez que lembra sua filha já falecida.

Seu papel no jogo é muito importante, não só pela motivação de Marshall, mas também pelo fato de seus gritos guiarem Marshall em sua busca para resgatá-la. À medida que Marshall se aproxima do local onde está presa, os gritos de Silmara ficam cada vez mais altos.

#### 3.4.1.1 Descrição Física

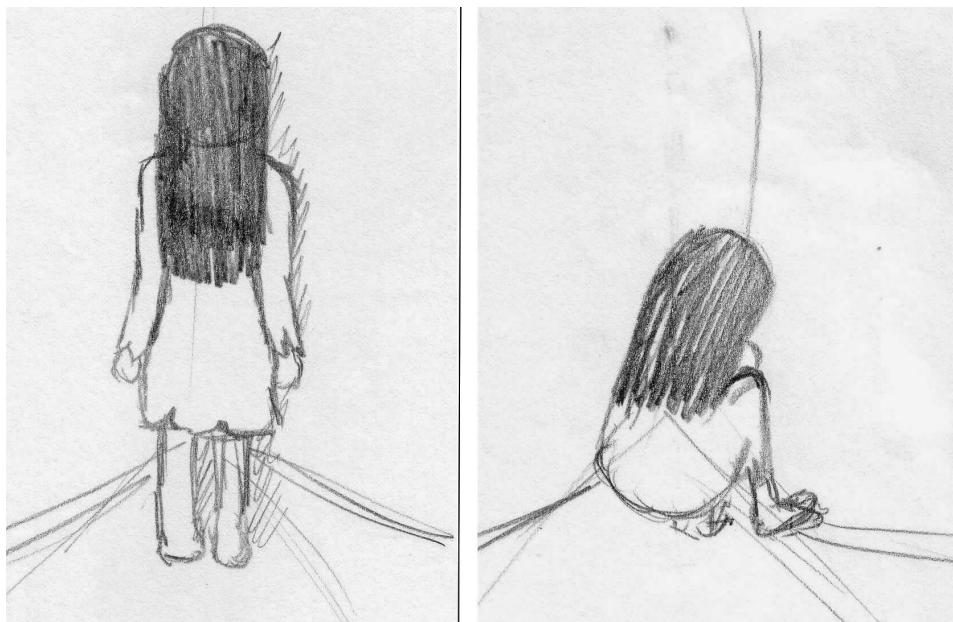
Com aproximadamente 1 metro e 40 centímetros de altura, Silmara tem o aspecto de uma criança jovem e saudável. Porém, a sua captura a deixou em estado de choque. Incapaz de fugir sozinha de tal situação, tudo o que lhe resta a fazer é gritar por socorro, esperando que alguém possa libertá-la.

Suas principais características físicas são os longos cabelos castanhos, um vestido branco que vai até seus joelhos, meias brancas e uma sapatilha preta.

Silmara baseia-se na personagem Samara, da série de filmes O Chamado ([http://pt.wikipedia.org/wiki/Samara\\_Morgan](http://pt.wikipedia.org/wiki/Samara_Morgan)).

#### 3.4.1.2 Concepção Artística

A figura abaixo exibe a concepção artística de Silmara



**Figura 3.38 - Silmara, a menina raptada por Diabolus e que deve ser salva por Marshall**

### 3.4.1.3 Imagens de Referência



**Figura 3.39** - Modelo de inspiração para a personagem Silmara  
(fonte: [http://4.bp.blogspot.com/\\_0cTS6V3MiWA/S17iCCz-3I/AAAAAAAARw/pc1KEsWIPuc/s400/samara.jpg](http://4.bp.blogspot.com/_0cTS6V3MiWA/S17iCCz-3I/AAAAAAAARw/pc1KEsWIPuc/s400/samara.jpg))

## 4 Fases

Levado até a mansão pelas pistas do desaparecimento, Marshall Gory, ao ouvir os gritos de socorro da vítima, deve tentar salvá-la o mais rápido possível. Mal sabe ele que se até às 3h33m não conseguir salvá-la, conhecerá a hora do demônio e todo seu esforço será em vão.

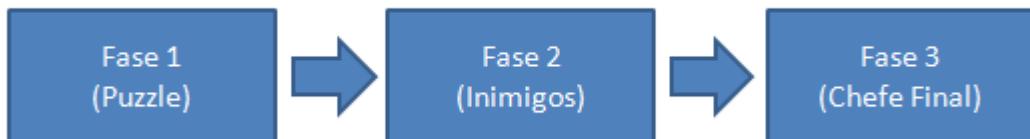
### 4.1 Descrição das Fases

O jogo Late Redemption é composto por 3 fases, cada uma com uma característica específica. A primeira foca na resolução de puzzles, enquanto a segunda se concentra no combate a inimigos normais; já a terceira fase lida apenas com o combate ao chefe final.

#### 4.1.1 Sequência

Conforme explicado nesta seção e na próxima, **Planta Baixa Geral**, o jogo todo se passa no interior de uma grande mansão. Nas fases do jogo, Marshall Gory percorre diversos cômodos da mansão solucionando enigmas e vencendo desafios a fim de alcançar o objetivo do jogo, que é o de sobreviver, escapando da mansão mal-assombrada, derrotando os inimigos e salvando a vítima.

Dessa forma, existe uma única sequência possível a ser seguida por Marshall no jogo, que é a descrita na figura abaixo.



*Figura 4.1 - Sequência de fases no jogo Late Redemption*

No início do jogo, Marshall encontra-se numa sala com uma mesa, armários e uma lareira, à busca de pistas para encontrar a vítima desaparecida. Ao perceber que está trancado dentro desta sala, e ao ouvir gritos que podem ser da vítima, ele deve encontrar uma maneira de sair da sala, objetivo que só se cumprirá com a resolução de alguns puzzles.

Uma vez fora da sala, Marshall vê ao longe a figura de uma pequena garota, e decide ir atrás dela seguindo pela direção esquerda no corredor principal, quando então encontra diversos personagens inimigos que ameaçam a sua vida. Ele deve agora lutar por sua vida, enquanto continua avançando na direção da garota e dos seus gritos. Chegando ao Main Hall da mansão, um saguão enorme com diversos móveis e pilares, deve continuar sua busca, agora enfrentando mais inimigos, em maior quantidade e diversidade. Após derrotar todos os inimigos no Main Hall, Marshall percebe que a garota está parada no topo da escada que dá acesso à biblioteca da mansão. Ao ir em sua direção, acaba perdendo-a de vista quando ela entra na biblioteca, fechando as grandes portas de madeira logo atrás de si.

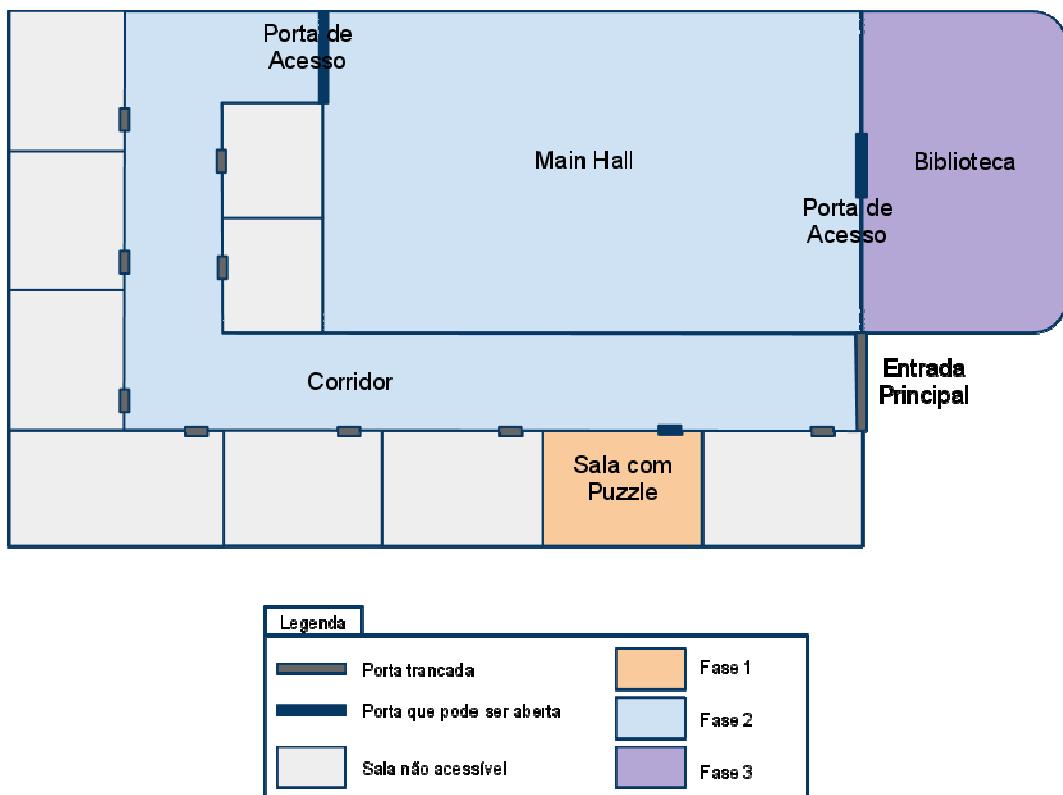
Ao abrir novamente as portas e entrar na grande biblioteca, Marshall se depara com o chefe final, responsável pelo desaparecimento da vítima. Marshall também vê a menina, até então desaparecida, presa dentro de uma gaiola suspensa no ar. Neste momento, o chefe final

inicia um diálogo com Marshall, explicando toda a trama até ali. É então que se inicia a batalha final que decidirá o destino de Marshall e da menina Silmara. Se o chefe final for derrotado, Marshall salvará Silmara e sairá pela porta dos fundos, deixando por fim a mansão mal-assombrada. Nesse momento, o jogo termina.

#### 4.1.2 Planta Baixa Geral

A planta abaixo demonstra o ambiente completo percorrido por Marshall Gory. As três fases do jogo são apresentadas na planta em cores diferentes, sendo estas descritas na legenda.

*Planta Baixa - Visão Geral do Jogo*



*Figura 4.2 - Planta baixa geral da mansão onde o jogo é ambientado*

#### 4.2 Fase 1: Sala da Mansão

No início da fase 1, a frase introdutória abaixo é apresentada:

“Existe uma barreira entre o real e o sobrenatural. Um local tão vasto quanto o espaço e tão desprovido de tempo quanto o infinito. Este lugar desafia o crescente conhecimento humano, levando a humanidade a confrontar seus medos e colocando em dúvida suas maiores certezas.”

Uma animação mostra Marshall Gory entrando na sala, ele possui uma arma e poderá usá-la durante esta etapa, entretanto seus tiros não terão efeito algum para ajudá-lo na fase.

#### 4.2.1 Descrição Física

A sala apresenta características de uma grande mansão ou de um hotel de luxo: chão acarpetado, porta de madeira nobre com maçaneta dourada e iluminação suave. Os móveis são levemente rústicos. Há uma janela onde pode-se avistar as condições climáticas: é noite e chove bastante.

Nesta sala, Marshall terá que utilizar os itens disponíveis para conseguir abrir a porta que foi trancada e passar para a segunda fase.

#### 4.2.2 Dinâmica da Fase

As referências deste texto podem ser encontradas na planta baixa da fase 1, na Figura 4.23.

Ao entrar na sala, a energia elétrica está funcionando. A porta (**i1**) se tranca e há uma explosão no quadro de distribuição do quarto (**i3**), a luz se apaga. Faíscas saem do quadro de distribuição (**i3**), dando a entender que a energia está com problemas. Neste momento, o silêncio é quebrado pelos gritos de uma menina atrás da porta. O jogo começa.

Consertar o quadro de distribuição (**i3**) fazendo a ligação correta de fios vai permitir que a energia volte a funcionar. A cada ligação errada, uma descarga causará choque, diminuindo os pontos de vida de Marshall em 10 pontos. Quando a energia voltar a funcionar, será possível ver um quadro com um dispositivo de segurança ativado (**i5**). A visualização será destacada por um pequeno LED vermelho piscando. Ao lado do LED, pode-se ver uma trava com um visor para entrada de três dígitos, onde uma senha deve ser informada para abrir o dispositivo. O valor padrão que está na trava é 666.

Atrás do sofá (**s1**) há um cadáver (**s2**), que não pode ser visto da porta da sala (**i1**) e nem do quadro de distribuição (**i3**). Quando Marshall chega perto dele, vem em sua lembrança uma notícia vista no jornal, sobre uma série de assassinatos e desaparecimentos que têm acontecido (**s2**). Essa lembrança será mostrada no jogo na forma de mensagens exibidas na tela. Ao lado do corpo haverá um papel em branco (**i4**). Este papel só poderá ser visto quando se chega perto do fogo da lareira (**i2**). Dependendo do texto no papel (**i4**), o segredo do dispositivo de segurança (**i5**) muda. Toda vez que a fase for jogada, este papel será sorteado entre algumas opções. O texto sempre começa com a frase “Se você está lendo este recado, você também foi atraído pelas sombras”. A frase seguinte é a senha do dispositivo de segurança e pode assumir os seguintes valores:

- Número **133** (Screamer) - Aqui o perigo espreita, mesmo com aparência caquética, com apenas um grito, é capaz e arrepia os pelos das vértebras de qualquer humano.
- Número **123** (Butcher) - Banhado em sangue será o anúncio da morte. Empunhando um cutelo duas vezes ele me atacou. Corri antes da terceira pois com certeza seria a última.
- Número **313** (Shortie) - A morte nem sempre é solitária. Em trios, eles parecem inofensivos. Mas cuidado, como apenas um, os três atacam.
- Número **212** (Crusher) - Com tamanho de dois homens e a força de uma dúzia, lentamente traz a destruição daqueles que o desafiam.
- Número **142** - Muitos são os inimigos daquele que vence a barreira. Em um corredor, quatro inimigos eu vi, mas com medo da morte, do segundo fui.

O texto sempre termina com a frase “Corra! Às 3h33m, na hora do demônio, seu destino poderá ser igual ao meu!”. Ao ler este papel na lareira, o tempo para término do jogo será iniciado, conforme explicado na seção **7.1 - HUD (Heads-up Display)**.

Com a senha descoberta, é possível realizar a abertura do dispositivo de segurança (**i5**). Caso o jogador não tenha lido o papel(**i4**), o tempo inicia-se neste momento.

Percebe-se então que o quadro era a porta de um pequeno cofre. Dentro do cofre, haverá uma chave (**i6**) e um papel (**i7**). Para ver o que está no papel, precisará chegar com ele perto da lareira (**i2**). No papel vai estar escrito: “Não substime o poder da água. Ela dá vida para todos os seres vivos e pode acabar com um dos elementos mais devastadores da natureza: o fogo. Não perca tempo e usufrua deste poder! Mas cuidado, esta ação pode levá-lo a cruzar a fronteira do real com o sobrenatural”.

A chave (**i6**) obtida no cofre consegue abrir um armário (**i8**) que está trancado. No armário haverá um galão de água (**i10**). Este galão será usado para apagar o fogo da lareira. Após apagar o fogo, poderá ser visto algo brilhante, outra chave (**i11**). Usa-se a chave para abrir a porta (**i1**) de saída da sala.

Haverá um armário (**i9**) destrancado e vazio, que poderá ser aberto pelo personagem. Este armário terá características similares ao armário trancado (**i8**).

Além disso, haverá mais três itens que não possuirão interação mas serão importantes para a caracterização da sala. Um tapete de pele de animal (**s3**), uma mesa de centro (**s4**) sobre o tapete e uma janela (**s5**). A janela apresentará as condições climáticas do lado de fora da mansão.

### **4.2.3 Descrição dos itens da fase**

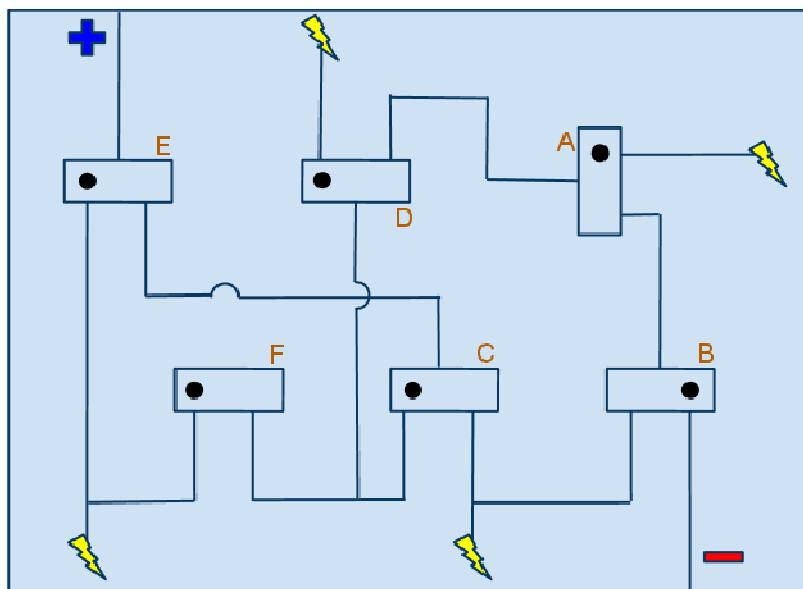
Os itens que Marshall Gory pode visualizar ou interagir na fase 1 estão descritos abaixo.

#### **4.2.3.1 Com interação**

Porta (**i1**) - Porta que será usada para a entrada do personagem (animação). Após a entrada, a porta permanece trancada até a chave (**i6**) ser encontrada.

Quadro de distribuição (**i3**) - Ele possui vários pontos em curto circuito, alguns dos quais possuem chaveadores acoplados. Marshall terá que identificar os que permitam a passagem da energia desviando dos pontos em curto circuito. Para isso, poderá fazer a mudança das chaves (pontos pretos).

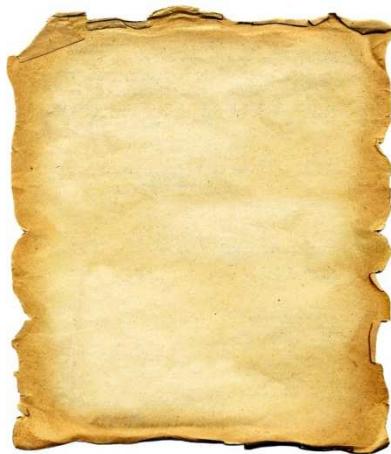
## *Quadro de distribuição com problemas.*



*Figura 4.3 - Quadro de distribuição de força com problemas*

O quadro possui vários circuitos emaranhados e partes desses circuitos estão em curto. O jogador deve perceber em que parte do circuito está ocorrendo o curto e mudar a posição dos chaveadores de forma que a corrente elétrica não passe pelas partes danificadas. Cada vez que a energia for chaveada para uma parte em curto do circuito, o jogador tomará um choque ocasionando-lhe perda de pontos de vida

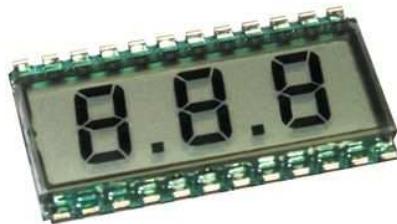
Papel com dica da senha (i4) - Esta dica será encontrada junto ao corpo (**s1**) que está atrás do sofá (**s2**). Inicialmente será um papel em branco que apenas trará informação quando for colocado perto do fogo.



*Figura 4.4 - Referência para o papel com a dica de senha  
(fonte: <http://www.angulopesquisa.com.br/img/old-paper21.jpg>)*

Quadro com dispositivo de segurança com três dígitos (trava) (i5) - O dispositivo de segurança fica ao lado direito (do personagem) do quadro. No visor do dispositivo de segurança está o numero 666. O dispositivo tem um pequeno LED vermelho piscando indicando sua atividade. Esse LED só estará piscando depois que Marshall reativar o quadro de energia (**i1**). Para informar a senha, deve-se clicar sobre o número desejado. A cada clique, o

número avança em um. Depois do 9, o número volta para zero. Ao informar a senha correta, um barulho é ouvido e a porta do cofre fica entreaberta.



**Figura 4.5 - Referência do visor do dispositivo de segurança do cofre**  
(fonte:  
<http://www.newark.com/productimages/nio/standard/4395380.jpg>)



**Figura 4.6 - Referência para o cofre**  
(fonte:  
[http://repositorio.intelidus.net/imagens/10000/1065/05\\_thumb.jpg](http://repositorio.intelidus.net/imagens/10000/1065/05_thumb.jpg))



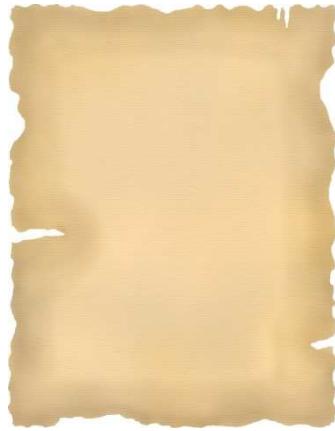
**Figura 4.7 - Referência para a pintura na frente do cofre**  
(fonte:  
[http://cdn.picapp.com/ftp/Images/5/7/f/9/Dante\\_Alighieri\\_Inferno\\_dafb.jpg?adImageId=8515335&imageId=7309837](http://cdn.picapp.com/ftp/Images/5/7/f/9/Dante_Alighieri_Inferno_dafb.jpg?adImageId=8515335&imageId=7309837))

**Chave que abre o armário (i6)** - Esta chave é encontrada dentro do cofre (i5) que foi aberto com a senha. É uma chave de cadeado comum



**Figura 4.8 - Referência para a chave do armário**  
(fonte: [http://marcelochaveiro.com.br/images/imagem\\_chave\\_comum.png](http://marcelochaveiro.com.br/images/imagem_chave_comum.png))

**Papel com dica para apagar fogo (i7)** - Este papel será encontrado juntamente com a chave mencionada anteriormente, ambos dentro do cofre (i5) que foi destrancado.



**Figura 4.9 - Referência para o papel com dica para apagar o fogo**  
(fonte: [http://www.antiquemallofstclair.com/old\\_paper.sized.jpg](http://www.antiquemallofstclair.com/old_paper.sized.jpg))

**Armário com água, trancado (i8)** - As portas da parte de baixo do armário estão trancadas. Não tem correntes nem outras coisas impedindo a abertura da porta, apenas uma pequena fechadura que deverá ser aberta com a chave encontrada dentro do cofre. A abertura revelará a existência de um galão de água (i10).



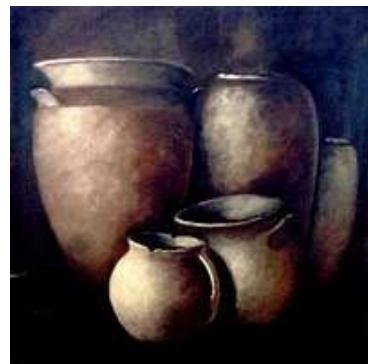
**Figura 4.10 - Referência para o armário contendo galões de água**  
(fonte: [http://images04.olx.pt/ui/2/80/30/19498030\\_1.jpg](http://images04.olx.pt/ui/2/80/30/19498030_1.jpg))

Armário destrancado vazio (i9) - Embora esteja com as portas fechadas, nenhuma delas está trancada. Marshall pode vasculhar o armário mas nada de relevante será encontrado.



**Figura 4.11** - Referência para armário vazio  
(fonte: [http://images04.olx.pt/ui/2/80/30/19498030\\_2.jpg](http://images04.olx.pt/ui/2/80/30/19498030_2.jpg))

Galão de água (i10) - Dentro do armário aberto (i8) com a chave (i6) encontrada dentro do cofre (i5), Marshall encontrará um galão de água que servirá para apagar o fogo da lareira (i2).



**Figura 4.12** - Referência para os galões de água  
(fonte: [http://4.bp.blogspot.com/\\_R8LYI6jvH1U/S-7oNSyeAvI/AAAAAAAAD\\_A/PONi7SW-apw/s200/O+Pote+Rachado.jpg](http://4.bp.blogspot.com/_R8LYI6jvH1U/S-7oNSyeAvI/AAAAAAAAD_A/PONi7SW-apw/s200/O+Pote+Rachado.jpg))

Chave que abre a porta do quarto (i11) - Comparada com a chave descrita anteriormente, esta será visualmente melhor desenhada, pois é a chave de uma das portas da mansão. Ela será encontrada dentre as cinzas da lareira (i2), assim que o fogo for extinto.



**Figura 4.13** - Referência para a chave da porta do quarto  
(fonte: [http://4.bp.blogspot.com/\\_sjX51bjYiMA/SutOWj2cYwI/AAAAAAAAljq/KfiYtdotENO/s400/chave.jpg](http://4.bp.blogspot.com/_sjX51bjYiMA/SutOWj2cYwI/AAAAAAAAljq/KfiYtdotENO/s400/chave.jpg))

Fogo da lareira (i2) - Ao entrar no quarto do puzzle, como a luz elétrica apaga-se, é a luz do fogo da lareira o responsável pela iluminação superficial do quarto. Depois que Marshall usa a água encontrada (i10) para apagar o fogo da lareira, a chave (i11) da porta surgirá dentre as cinzas do fogo extinto.



**Figura 4.14** - Referência para a lareira  
(fonte:

[http://www.colonialchurrasqueiras.com.br/painel//paqauto\\_files/323.25/image/07-lareira.jpg](http://www.colonialchurrasqueiras.com.br/painel//paqauto_files/323.25/image/07-lareira.jpg)



**Figura 4.15** - Referência para o fogo da lareira  
(fonte:

[http://3.bp.blogspot.com/\\_KnARo\\_MMfwU/R7lc46CI/AAAAAAAABVw/YQxb0Lk7DC8/s400/lareira2.jpg](http://3.bp.blogspot.com/_KnARo_MMfwU/R7lc46CI/AAAAAAAABVw/YQxb0Lk7DC8/s400/lareira2.jpg)

#### 4.2.3.2 Sem interação

Os elementos a seguir são classificados como sem interação com o personagem principal por não terem uma função direta nos puzzles. Entretanto, estes itens possuem características físicas reais e podem, por exemplo, ser derrubados por Marshall.

Sofá (s1) - Sofá em “L” que decora a sala e é usado para ocultar um cadáver atrás dele.



**Figura 4.16** - Referência para o sofá  
(fonte: <http://www.sofasnyc.com/smartimage/Grey-Fabric-Modern-Sectional-Sofa-with-Metal-Legs-img-5151.jpg>)

Corpo no chão (s2) - Cadáver de uma pessoa que morreu após os ataques dos inimigos no corredor. É ele quem possivelmente tenha deixado as mensagens pela sala.



**Figura 4.17** - Referência para o cadáver na fase 1

Tapete (s3) - Tapete de pele de animal que decora a sala.



**Figura 4.18** - Referência para o tapete de pele animal que decora a sala

(fonte:

[http://user.img.todaoferta.uol.com.br/B/6/MO/8RLXHA/1194128427429\\_bigPhoto\\_5.jpg](http://user.img.todaoferta.uol.com.br/B/6/MO/8RLXHA/1194128427429_bigPhoto_5.jpg))

Mesa de centro (s4) - Mesa de centro que decora a sala.



**Figura 4.20** - Referência para mesa de centro da sala

(fonte:

[http://imagenes.solostocks.com/z1\\_3222\\_822/mesa-centro-rustica-celta-acacia.jpg](http://imagenes.solostocks.com/z1_3222_822/mesa-centro-rustica-celta-acacia.jpg))

**Figura 4.19** - Referência para mesa de centro da sala  
(fonte:  
[http://www.portalsousas.com.br/\\_upload/rteimages/\\_move103.jpg](http://www.portalsousas.com.br/_upload/rteimages/_move103.jpg))

Janela (s5) - Janela que mostra as condições climáticas do lado de fora da mansão.



*Figura 4.21 - Referência para a janela da sala*  
(fonte: <http://fragmentosdevida.files.wordpress.com/2008/08/janela.jpg>)

#### 4.2.3.3 Pensamentos do personagem principal

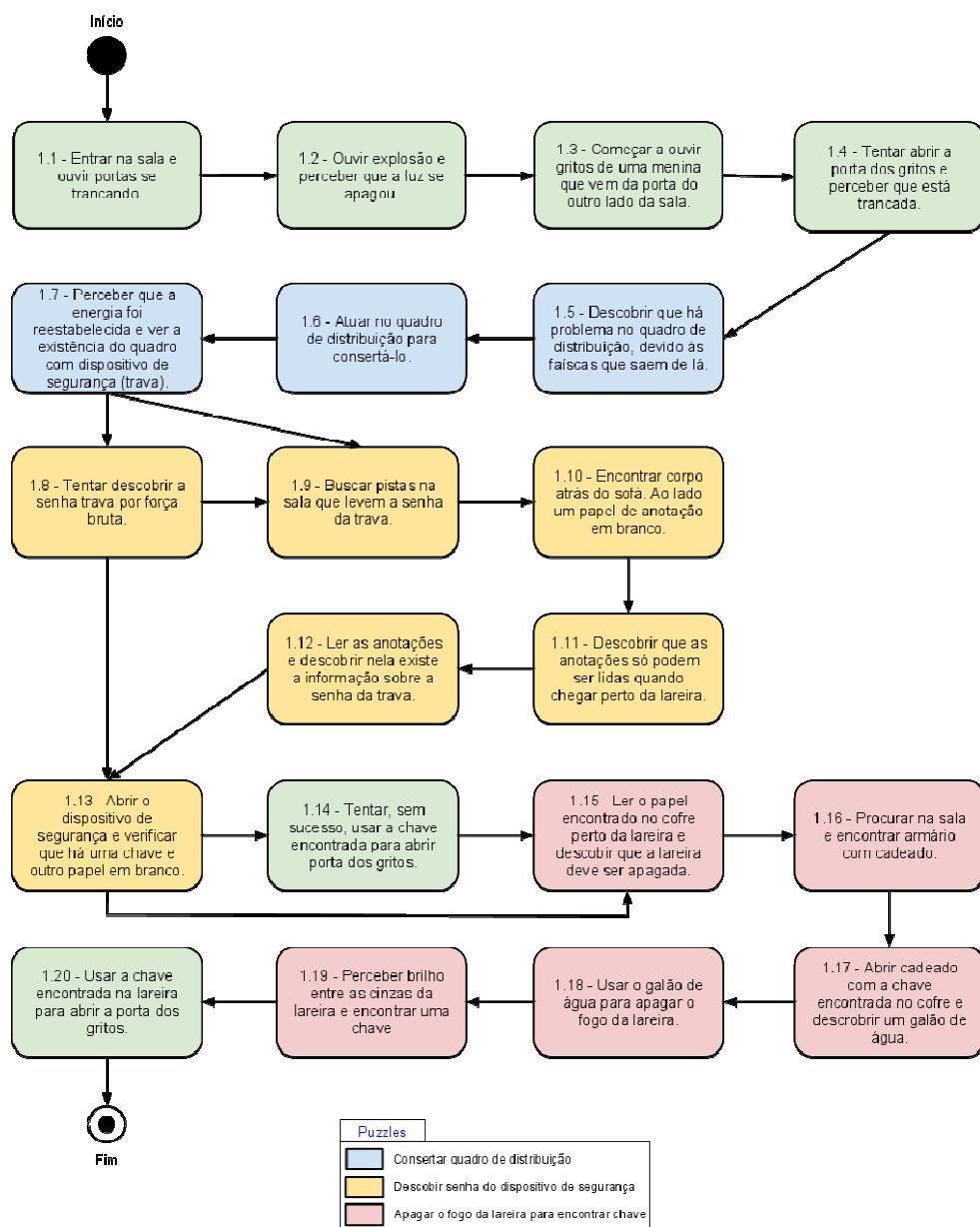
Para auxiliar na saída da sala, ao interagir com alguns itens da sala, uma mensagem aparece na tela, como se fosse o pensamento de Marshall. Abaixo segue a relação de pensamentos por itens que serão apresentados.

- (s1) - “Meu Deus! Eu vi esta pessoa no noticiário. É mais um caso de desaparecimento relacionado àquele misterioso assassino. Onde fui me meter?”
- (i1) - “Droga! A porta está trancada. E estes gritos? Será que é a menina que estou procurando?”
- (i3) - “Nesta escuridão não vou conseguir sair daqui. Este painel deve controlar a energia do quarto. Talvez eu consiga consertá-lo”. Somente no nível de dificuldade normal.
- (i4) e (i7) - “Papel em branco? Deve haver algo que eu possa fazer com ele para ver se há algo escrito”. Somente no nível de dificuldade normal.
- (i5) - “Que pintura estranha. Deve haver algo aí atrás. Preciso descobrir a senha para abrí-lo”. Somente no nível de dificuldade normal.
- (i6) e (i11) - “Uma chave! Tenho que descobrir o que ela abre...”. Somente no nível de dificuldade normal.
- (i8) - “Que estranho! Por que está trancado?”

#### 4.2.4 Flowchart

O diagrama abaixo representa o fluxo de ações possíveis na fase 1.

## Flowchart de ações - Fase 1



**Figura 4.22 - Flowchart da fase 1**

### 4.2.5 Planta Baixa

A figura abaixo exibe a planta baixa da fase 1:

## Planta Baixa - Fase 1 - Sala

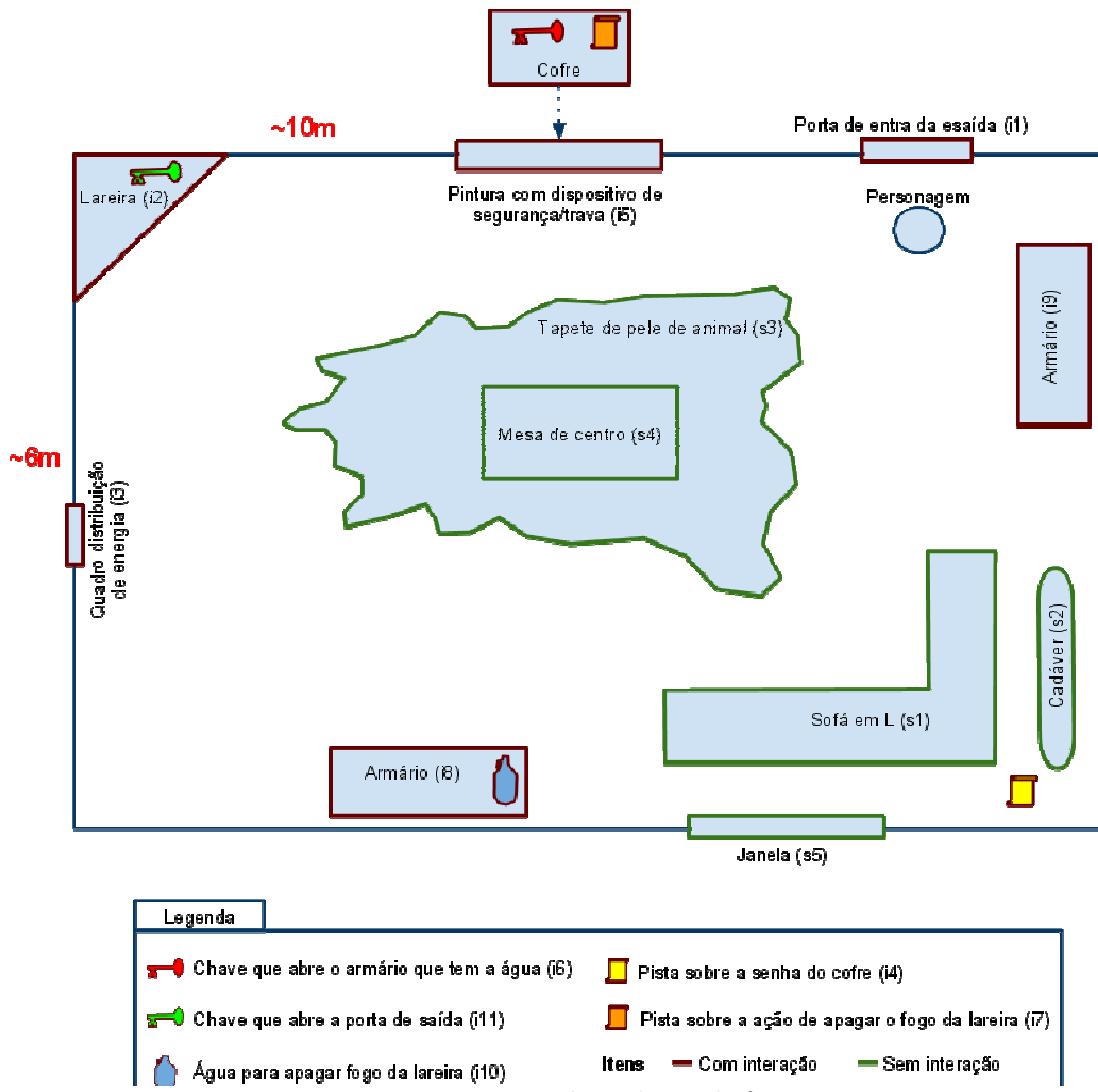


Figura 4.23 - Planta baixa da fase 1

## 4.3 Fase 2: Interior da Mansão

Uma vez fora do quarto, Marshall Gory deve seguir na direção dos gritos para tentar encontrar a vítima desaparecida. A direção será indicada pelos sons de gritos, que vão ficando mais fortes à medida que Marshall avança na direção da porta que separa o *Corridor* do *Main Hall*. Ao percorrer o interior da mansão, encontrará inimigos que deverá derrotar para avançar em sua busca. A segunda fase do jogo é composta por dois cenários: *Corridor* e *Main Hall*.

### 4.3.1 Flowchart

O flowchart abaixo descreve a sequência de ações do personagem principal ao seguir pelos cenários da segunda fase do jogo, considerando o melhor caso, em que ele não seja derrotado por nenhum inimigo, ou que o tempo limite do jogo não acabe. As ações descritas nas caixas com contorno colorido indicam o confronto com um inimigo, sendo uma cor diferente para cada tipo de inimigo.

## Flowchart de ações - Fase 2

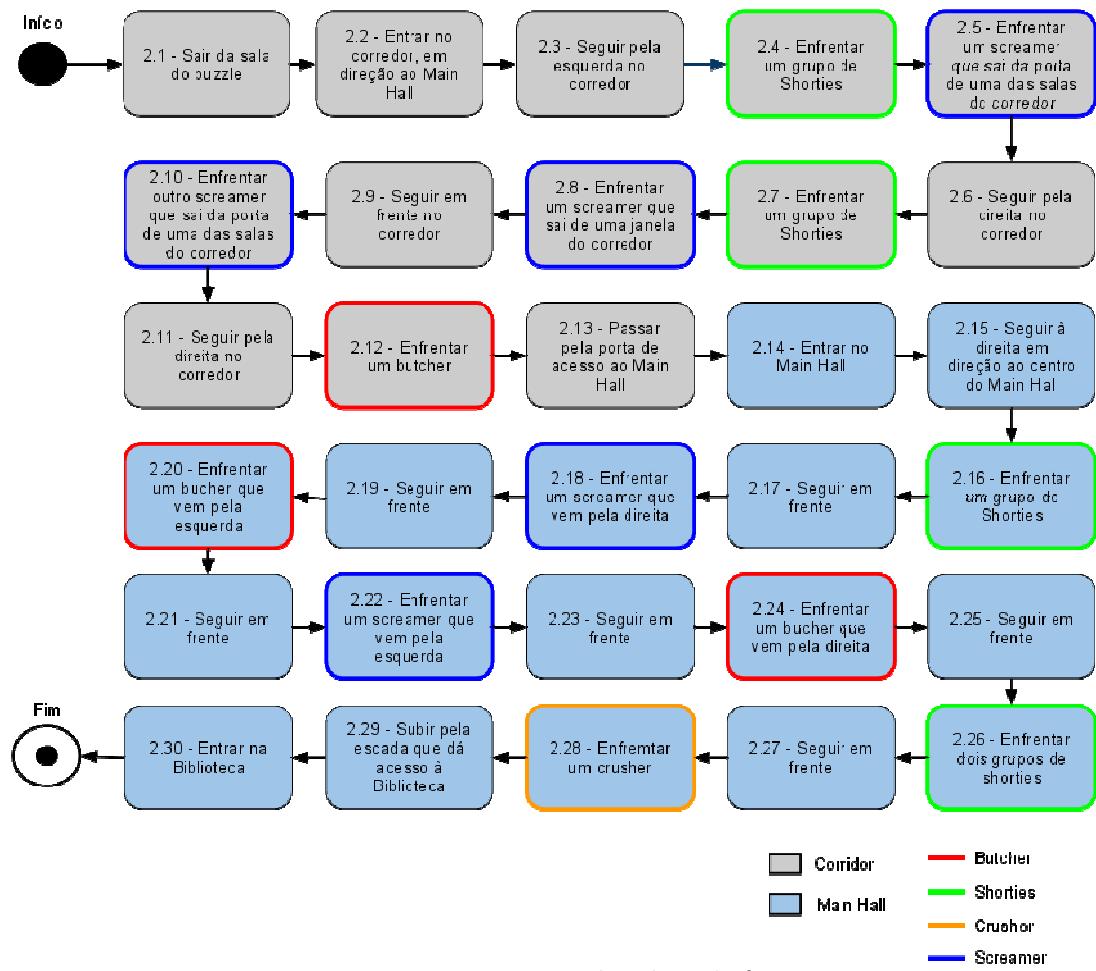


Figura 4.24 - Flowchart da fase 2

### 4.3.2 Corridor

O cenário *Corridor*, dentro da segunda fase, serve de palco para o primeiro encontro de Marshall Gory com os inimigos. É quando ele se dá conta de que algo de sobrenatural realmente está acontecendo. Neste cenário, Marshall deverá se deslocar pelo corredor em forma de U. Uma vez que entre no corredor, Marshall não poderá mais voltar para a sala da primeira fase.

Os inimigos presentes neste cenário serão: Shorties, Screamers e o Butcher. Eles estão espalhados de forma estratégica, sendo que Marshall deverá passar por todos eles obrigatoriamente, para ir em direção à saída do cenário. O número total de personagens inimigos no nível de dificuldade “Go Easy On Me” é 10, distribuídos da seguinte maneira: 2 grupos de Shorties (totalizando 6 inimigos), 3 Screamers e um Butcher, no papel de subchefe de fase.

Da mesma forma, armas e munição estão espalhadas pela fase em quantidade adequada, de forma que o personagem principal não fique desprovido de formas para derrotar os inimigos presentes.

Para sair desse cenário, Marshall deverá cumprir as seguintes condições: derrotar o inimigo Butcher, coletar a chave que está com ele e abrir a porta de acesso ao Main Hall, dentro do tempo hábil.

#### 4.3.2.1 Descrição Física

O Corridor apresenta as mesmas características do corredor de uma grande mansão ou de um hotel de luxo: chão acarpetado, portas de madeira nobre com maçanetas douradas, vasos de plantas no chão, candelabros nas paredes e iluminação suave. Na parte superior do corredor, no acesso ao Main Hall, as paredes do lado externo possuem janelas que dão para o jardim externo da Mansão, podendo-se ver plantas e grandes árvores. Através dessas janelas pode-se ver raios de uma tempestade, que também iluminam o interior do ambiente.

#### 4.3.2.2 Planta Baixa

A figura abaixo exibe a planta baixa do cenário *Corridor* da fase 2.

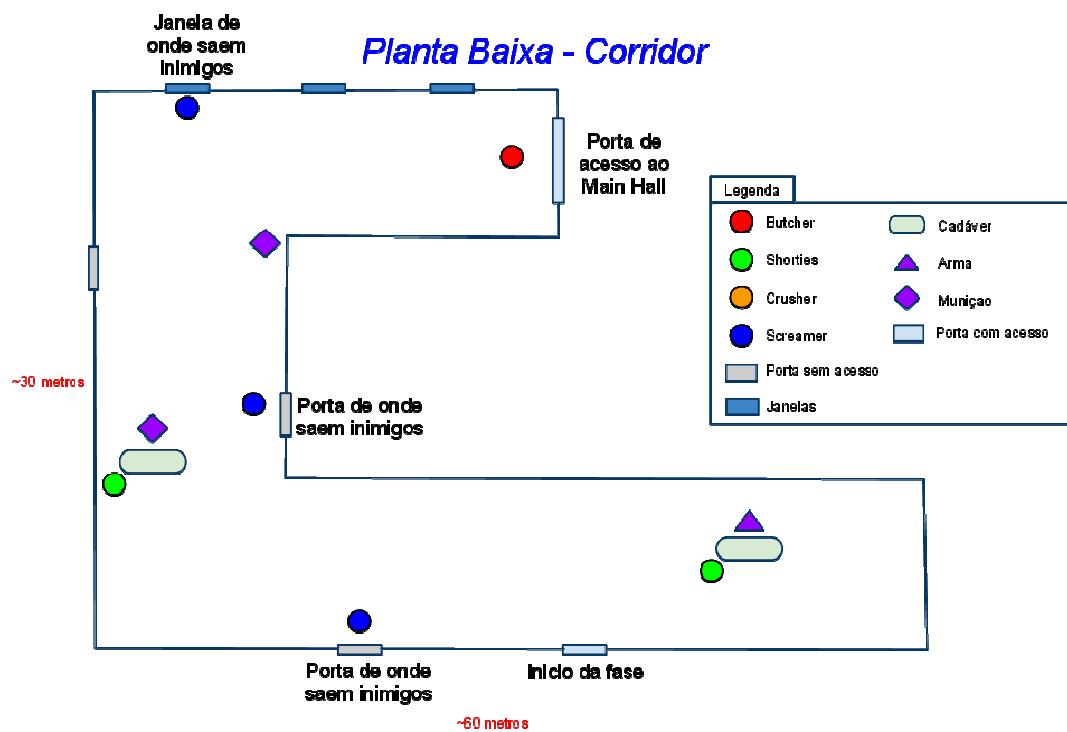


Figura 4.25 - Planta baixa do cenário Corridor da fase 2

#### 4.3.2.3 Imagens de referência

Como base para o ambiente de corredor da segunda fase, as imagens a seguir serviram como inspiração:



**Figura 4.26** - Imagem do corredor de um hotel  
(fonte:

[http://www.hoteldesigns.net/library/1184713200/original\\_corridor\\_carefully\\_restored\\_with\\_a\\_gallery\\_of\\_past\\_visitors\\_to\\_the\\_hotel.jpg](http://www.hoteldesigns.net/library/1184713200/original_corridor_carefully_restored_with_a_gallery_of_past_visitors_to_the_hotel.jpg))



**Figura 4.27** - Imagem do corredor de um hotel  
(fonte: <http://www.ignorancia.org/uploads/images/hotel/hotel.jpg>)

Jaime Vives Piqueres, (?)2001.

### **4.3.3 Main Hall**

O *Main Hall* é o principal cenário de batalha do jogo com inimigos normais. Os quatro tipos de inimigos (Shorties, Screamers, Butchers e Crushers) estão presentes, espalhados de forma estratégica, ou seja, não importa por onde o personagem principal andar no cenário, ele sempre vai encontrar um grupo variado de inimigos. Especialmente para sair do cenário, em direção à entrada da próxima fase, a Biblioteca, Marshall obrigatoriamente terá que passar por 2 grupos de Shorties, e um Crusher.

O número total de personagens inimigos no nível de dificuldade “Go Easy On Me” é 14, distribuídos da seguinte maneira: 3 grupos de Shorties (totalizando 9 inimigos), 2 Screamers, 2 Butchers e um Crusher, no papel de subchefe de fase.

Armas e munição estão espalhadas pela fase em quantidade adequada, de forma que Marshall não fique desprovido de formas para derrotar os inimigos presentes.

Para sair desse cenário, Marshall deverá cumprir as seguintes condições: derrotar o inimigo Crusher, coletar a chave que está com ele, subir a escada e abrir a porta de acesso à biblioteca, dentro do tempo hábil.

#### **4.3.3.1 Descrição Física**

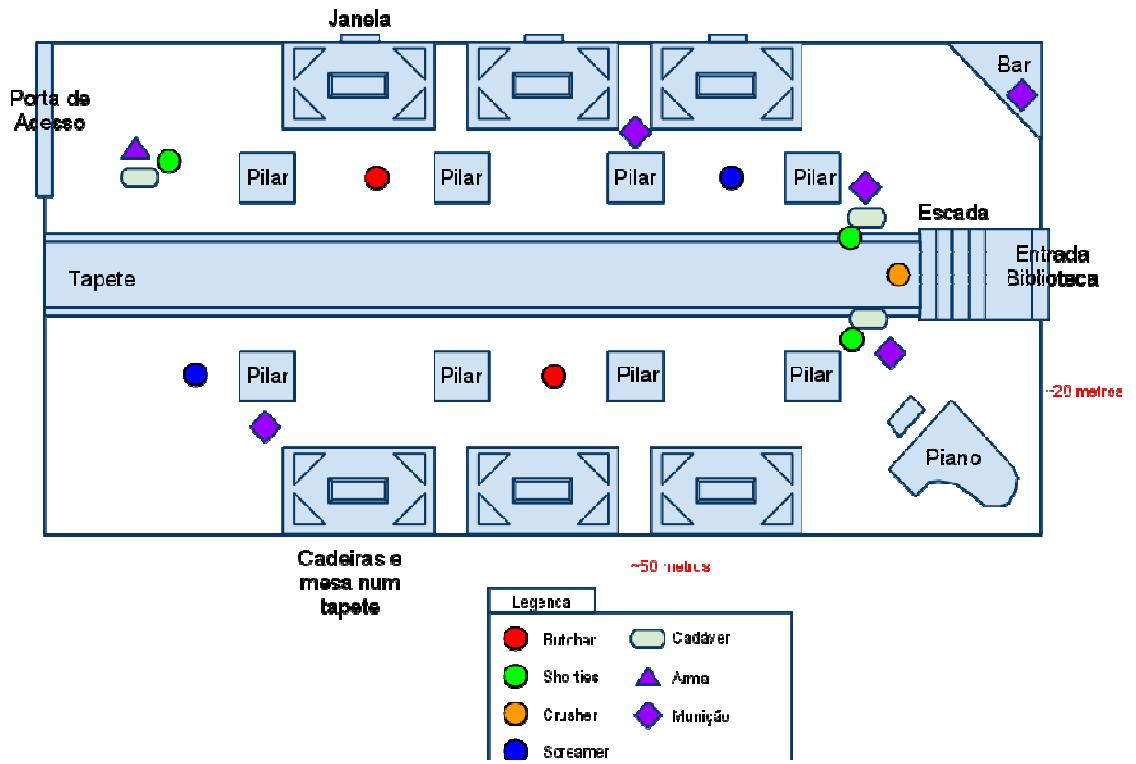
O salão principal da mansão é um saguão grande, utilizado principalmente para entreter os frequentadores do espaço ou para dar festas. O ambiente apresenta uma grande porta de madeira como entrada principal, um longo tapete cobrindo toda a extensão do caminho principal, cortando o salão de uma ponta a outra, até a escada que dá acesso à biblioteca, entrada esta que também tem uma grande porta de madeira. Pelo salão ainda espalham-se grandes pilares, que servem de sustentação para o teto alto. Estes pilares servem como elementos de jogabilidade para executar manobras de desvio e proteção, para que o personagem principal possa evitar ataques inimigos.

Como mobiliário, o salão contém 6 espaços de convivência, pequenas áreas onde pessoas podem se agrupar e socializar. Cada um desses espaços é composto por um tapete, sobre o qual encontram-se uma mesa e quatro cadeiras, dispostas nos cantos do tapete. As cadeiras e mesas possuem física real, ou seja, se o personagem principal esbarrar nesses elementos, eles serão empurrados ou até mesmo cairão no chão, dando maior sensação de realismo ao ambiente. No fundo do salão encontram-se um grande piano de cauda num dos lados, e um balcão com bar de bebidas no outro.

#### **4.3.3.2 Planta Baixa**

A figura abaixo exibe a planta baixa do cenário *Main Hall* da fase 2:

## Planta Baixa - Main Hall



*Figura 4.28 - Planta baixa do cenário Main Hall da fase 2*

### 4.3.3.3 Imagens de referência

Como base para o ambiente de saguão de mansão da segunda fase, as figuras a seguir serviram como inspiração.



*Figura 4.29 - Imagem do saguão de um hotel*

(fonte: [http://static.laterooms.com/hotelphotos/laterooms/187/gallery/barcelo-shrigley-hall-hotel-macclesfield\\_030320091942057755.jpg](http://static.laterooms.com/hotelphotos/laterooms/187/gallery/barcelo-shrigley-hall-hotel-macclesfield_030320091942057755.jpg))



**Figura 4.30** - Imagem do saguão de um hotel  
(fonte: <http://www.malapronta.com.br/fotos/496/hotel-foz-do-iguacu-496-hall-de-entrada-0.jpeg>)



**Figura 4.31** - Imagem do saguão de um hotel  
(fonte:  
<http://corinthiagrand.luxuryhotelsbudapest.com/images/Marble%20covered%20hall%20of%20Corinthia%20Grand%20Hotel%20Royal%20Budapest%20Hungary.png>)



**Figura 4.32 - Imagem do saguão de um hotel**  
(fonte: <http://www.historic-hotels-lodges.com/new-hampshire/mount-washington-hotel/photos/great-hall.jpg>)

## 4.4 Fase 3: Library

O cenário final do jogo é a Library, ou biblioteca. Nele, Marshall Gory encontrará o responsável pelo desaparecimento de Silmara, e deverá derrotá-lo para ter a chance de salvar a menina, e com isso encontrar sua própria redenção.

### 4.4.1 Descrição Física

O ambiente da Library consiste de um grande salão em semi-círculo, com dois andares de prateleiras cheias de livros estendendo-se pelas paredes do semi-círculo. A grande porta principal é de madeira, e dá acesso a quem vem do Main Hall. Existe outra porta, menor, que serve de saída de emergência, e se encontra na lateral superior da construção.

O centro da Library costumava ter várias mesas e cadeiras para leitura, mas por causa do grande buraco aberto no chão por Diabolus, apenas algumas mesas sobraram espalhadas pela sala. Tal buraco tem um grande diâmetro, e é o meio de acesso da forma física de Diabolus a este mundo, servindo como um portal entre o inferno e a Terra. Dessa forma, sua profundidade é imensa, alcançando o reino do grande demônio. Grandes nuvens de vapor e cheiro de enxofre saem desse buraco.

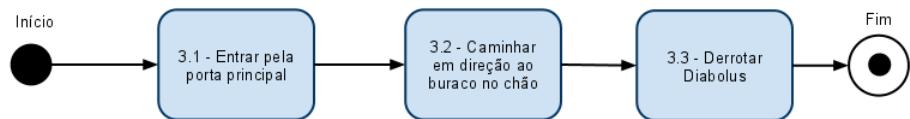
A única maneira de se locomover dentro da Library é no espaço livre entre a porta de entrada e o buraco, e as estreitas passagens que existem entre as paredes e o grande buraco, conforme pode ser visto na planta baixa.

Marshall Gory pode ainda usar os mesas restantes como elementos de proteção contra os ataques de Diabolus.

#### 4.4.2 Flowchart

O diagrama a seguir descreve o fluxo de ações de Marshall na fase 3:

*Flowchart de ações - Fase 3*

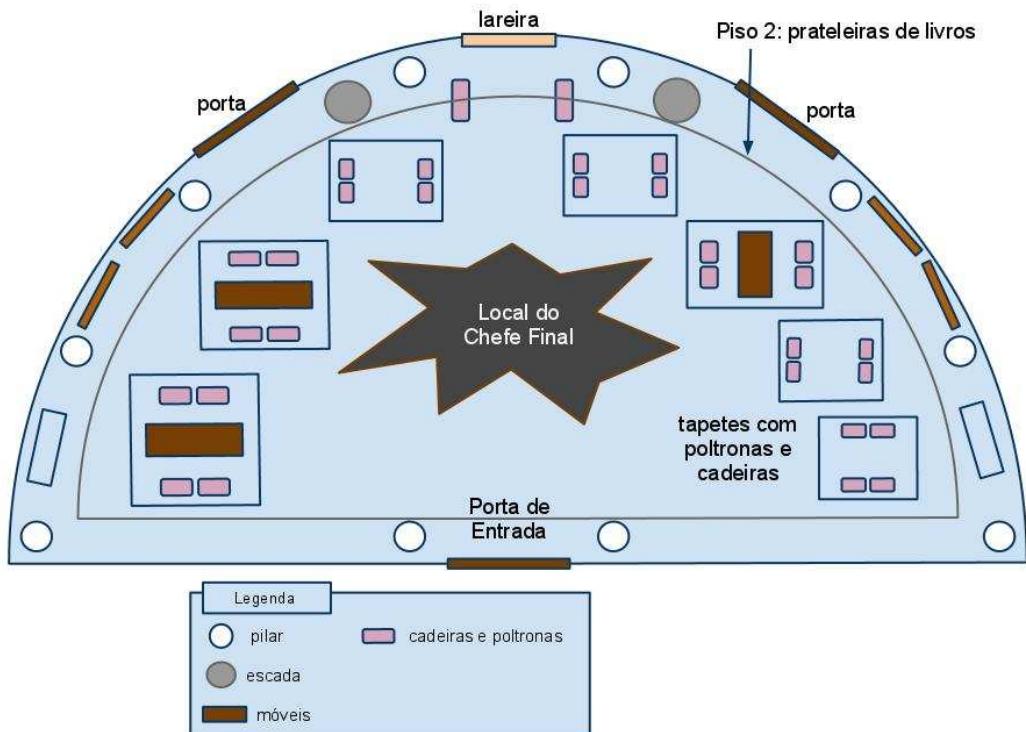


*Figura 4.33 - Flowchart da Fase 3 - Library*

#### 4.4.3 Planta baixa

A figura abaixo exibe a planta baixa da fase 3:

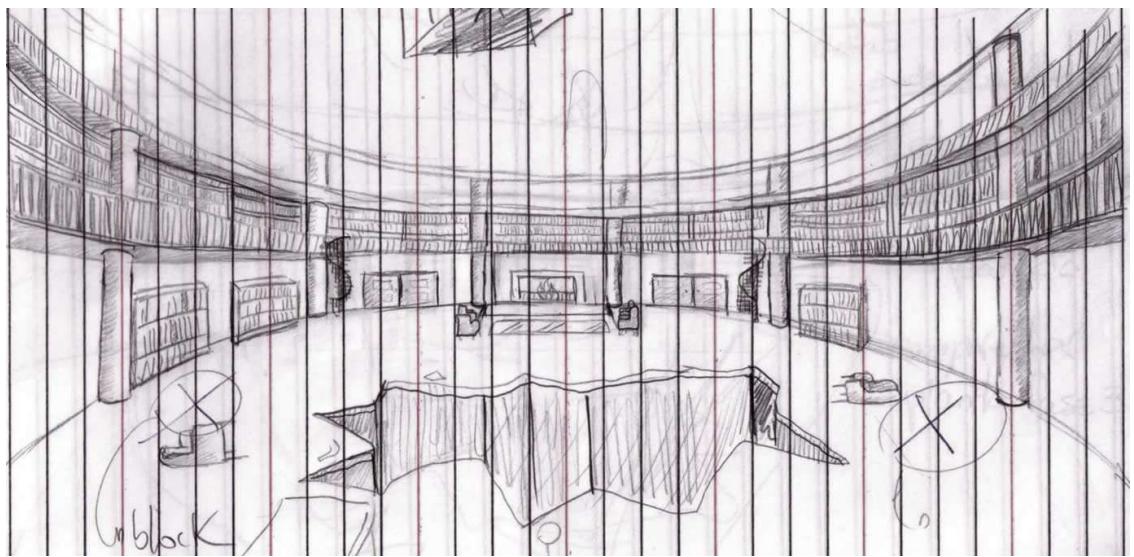
*Planta Baixa - Library*



*Figura 4.34 - Planta baixa da Fase 3 - Library*

#### 4.4.4 Concepção Artística

A concepção artística da fase 3 está ilustrada na imagem a seguir:



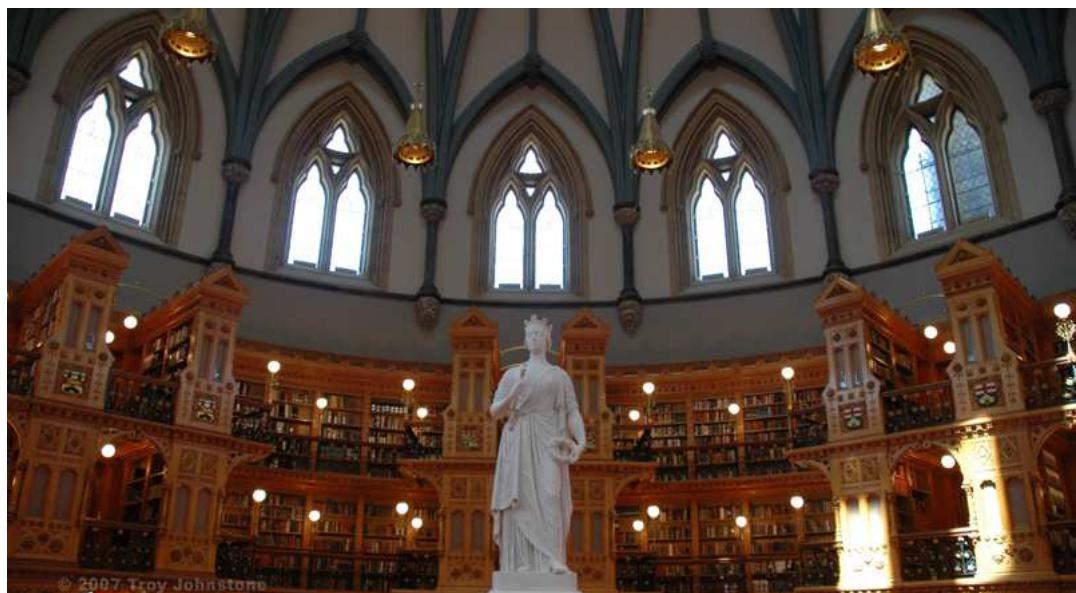
**Figura 4.35 - Concepção artística da Fase 3 - Library**

#### 4.4.5    Imagens de referência

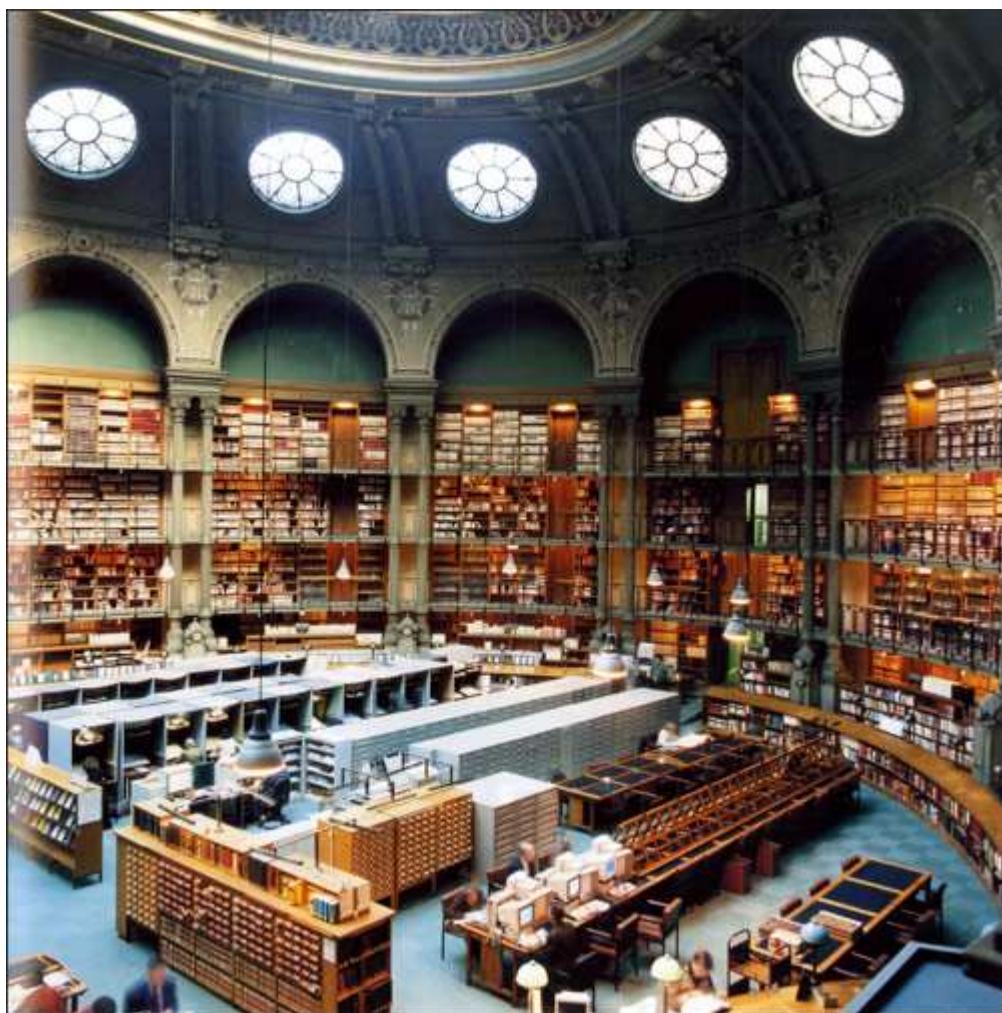
Como base para o ambiente da biblioteca da mansão da terceira fase, as figuras exibidas nesta seção serviram como inspiração.



**Figura 4.36 - Imagem de uma biblioteca do Reino Unido**  
(fonte: <http://www.worldinterestingfacts.com/wp-content/uploads/2009/11/UK-Library-2.jpg>)



**Figura 4.37 - Imagem de uma biblioteca do Canadá**  
(fonte: <http://www.worldinterestingfacts.com/wp-content/uploads/2009/11/Canada-Library-1.jpg>)



**Figura 4.38 - Imagem de uma biblioteca da França**  
(fonte: <http://www.worldinterestingfacts.com/wp-content/uploads/2009/11/France-Library-2.jpg>)

# 5 Fluxo do Jogo

Este capítulo apresenta uma visão detalhada de como os diversos elementos do jogo estão conectados, descrevendo o que o jogador pode ver e fazer desde o momento em que iniciar o jogo e incluindo todas as principais opções às quais ele tem acesso.

## 5.1 Progressão do Jogo

No início do jogo, o nome do jogo poderá ser visualizado, em seguida a cena 1 é apresentada. Após a apresentação da cena, o jogador visualizará o menu principal (cena 7), onde haverá as seguintes opções:

1. New Game. Após selecionar esta opção, o jogador deverá selecionar um dos níveis de dificuldade (cena 8). Em seguida, o jogo se inicia na fase 1.
2. Input Keys. Opção usada para apresentar os controles disponíveis no jogo (cena 9).
3. Load Game. Após selecionar esta opção o jogador deverá escolher um dos jogos gravados anteriormente (cena 10). Em seguida, o jogo se inicia no ponto de salvamento.
4. Ranking. Após selecionar esta opção, o jogador deverá selecionar um dos níveis de dificuldade. Em seguida, uma lista com os 5 melhores desempenhos naquele nível é apresentada (cena 11).
5. Credits. Mostra os créditos, listando os nomes e fotos dos desenvolvedores (cena 15).
6. Quit. Esta opção fecha o jogo.

Ao iniciar um novo jogo, a cena 2 é apresentada. Após a cena, a fase 1 começa. Nesta fase o jogador deverá descobrir como sair da sala, usando os itens existentes, conforme descrito na seção **4.2 - Fase 1: Sala da Mansão**.

Ao término da primeira fase, a cena 3 é apresentada. Após a cena, a fase 2 começa. Nesta fase Marshall deverá caminhar por um corredor e pelo saguão da mansão, enfrentando inimigos sobrenaturais. O objetivo é matar todos os oponentes no caminho até a entrada da biblioteca, conforme descrição na seção **4.3 - Fase 2: Interior da Mansão**.

Ao término da segunda fase, a cena 4 é apresentada. Após a cena, a fase 3 começa. Nesta fase Marshall deverá derrotar o chefe final para salvar a menina sequestrada, conforme descrição na seção **4.4 - Fase 3: Library**. Caso tenha sucesso nesta fase, a cena 6 é apresentada. Em seguida, os créditos do jogo são apresentados na cena 15. Após os créditos, a pontuação obtida pelo jogador e o ranking atualizado são mostrados na cena 11.

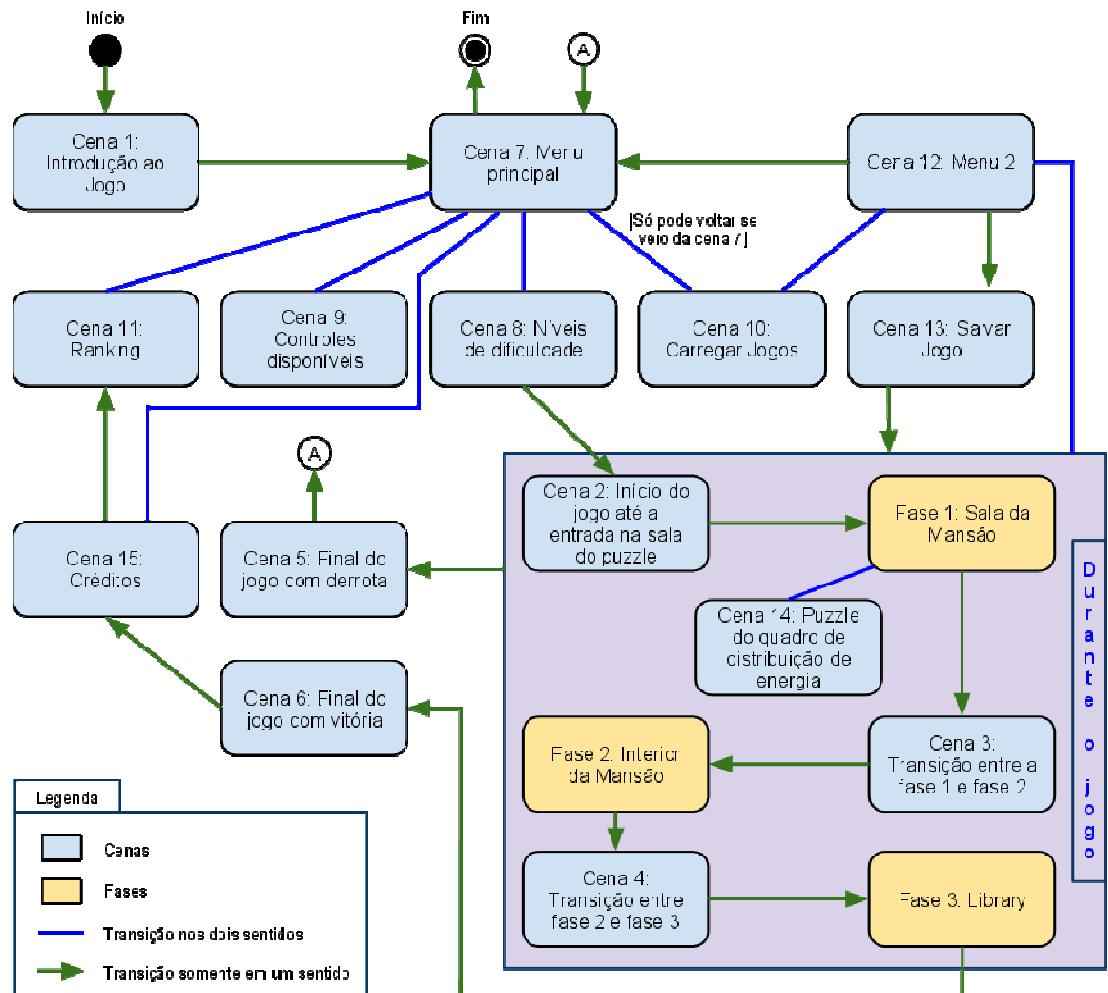
Durante o jogo, caso o jogador desejar, poderá acessar a cena 12 onde terá as seguintes opções:

1. Resume. Retorna ao jogo.
2. Quit. Para o jogo sem salvar, retorna ao menu principal (cena 7).
3. Save Game. Realiza ação de salvar o jogo. Ao selecionar esta opção o jogador deverá informar um nome para o ponto de salvamento (cena 13).
4. Load Game. Realiza ação de carregar um jogo, conforme descrito no menu principal.

Em qualquer momento do jogo, caso ocorra uma condição de derrota a cena 5 é apresentada.

A **Figura 5.1** exibe as possibilidades de transições descritas.

### Transições entre cenas e fases



**Figura 5.1** - Transições entre cenas e fases.

## 5.2 Cenas entre Fases (cinematics / cutscenes)

As diversas cenas existentes no jogo serão descritas nesta seção.

### 5.2.1 Cena 1: Introdução ao Jogo

Esta cena introdutória será dividida em três partes:

#### 5.2.1.1 Parte 1: Apresentação

Assim que o jogo é inicializado, aparecerá o logo da UDK (*Unreal Development Kit*) pois ele é padrão para jogos produzidos nesta engine.

Na sequência, será mostrada a imagem de apresentação do grupo de desenvolvimento por trás deste jogo. Esta imagem será composta de:

- Logotipo da Universidade
- Informações a respeito da Disciplina (Nome, código, professor)
- Identificação do Grupo que desenvolveu este jogo (sem os nomes num primeiro momento pois estes irão aparecer nos créditos).

### 5.2.1.2 Parte 2: Enredo

Logo em seguida, entrará o vídeo responsável pela apresentação do enredo do jogo e a história do personagem principal (Marshall Gory), de uma forma bastante sutil, dando indícios que levem os jogadores a entender o porquê da mansão ser o palco do jogo.

Os elementos que serão transmitidos nesta introdução estão relacionados ao passado do personagem principal que era feliz, tanto no ambiente profissional quanto familiar, até o dia em que sua filha é raptada e assassinada.

A partir deste ponto, a dor e revolta transformam a sua vida em uma existência sem sentido e, com ajuda e um novo propósito de vida ele encontra forças para tentar recomeçar.

Enquanto uma música de fundo é apresentada, uma série de fotos serão mostradas, de forma semelhante a uma apresentação de slides, contando um pouco da história de vida de Marshall. Em paralelo à exibição destas fotos, mensagens curtas e objetivas serão apresentadas ao jogador assim como informações da equipe de desenvolvimento do jogo também serão apresentados neste interim:

#### 1ª Imagem: Uma foto de família



*Figura 5.2 - Referência - Uma foto de Família  
(fonte:*

[http://4.bp.blogspot.com/\\_N7gno7a5\\_DU/SceZaeQhVBI/AAAAAAAfkfU/L4iSpnDIINE/s1600/familia1.jpg](http://4.bp.blogspot.com/_N7gno7a5_DU/SceZaeQhVBI/AAAAAAAfkfU/L4iSpnDIINE/s1600/familia1.jpg)

Frase: Marshall Gory era uma pessoa feliz, com uma linda família...

#### 2ª Imagem: Uma foto de militar



**Figura 5.3 - Referência - Foto de Militar**  
(fonte: [http://oglobo.globo.com/fotos/2008/06/19/19\\_MVG\\_rio\\_exercito02.jpg](http://oglobo.globo.com/fotos/2008/06/19/19_MVG_rio_exercito02.jpg))

Frase: E no trabalho, era admirado e respeitado...

O tom da música começa a mudar, tornando-se mais pesado.

**3ª Imagem: Uma foto com Marshall abrindo a porta do quarto da filha e não a encontrando**



**Figura 5.4 - Referência - Quarto vazio (sequestro...)**  
(fonte: [http://2.bp.blogspot.com/\\_99Sa9PJkISw/TIG4rRpR6xI/AAAAAAAAPU/k9u8KZUre-Y/s1600/cama.jpg](http://2.bp.blogspot.com/_99Sa9PJkISw/TIG4rRpR6xI/AAAAAAAAPU/k9u8KZUre-Y/s1600/cama.jpg))

Frase: Até que um dia... sua paz e sua filha desaparecem ao mesmo tempo...

**4ª Imagem: Uma foto de um funeral**



**Figura 5.5 - Referência - Funeral/Cemitério**  
(fonte: [http://guidesmedia.ign.com/guides/811232/images/heavy\\_rain\\_48.jpg](http://guidesmedia.ign.com/guides/811232/images/heavy_rain_48.jpg))

Frase: A dor da perda endureceu seu coração e sua vida perdeu o sentido...

**5ª Imagem: Marshall desesperado**



**Figura 5.6 - Referência – Desilusão**  
(fonte: [http://2.bp.blogspot.com/\\_zIMqLOOY-3o/TUDMhhRPWI/AAAAAAAAN8/KJcYqbrSY4/s1600/dese sperado.JPG](http://2.bp.blogspot.com/_zIMqLOOY-3o/TUDMhhRPWI/AAAAAAAAN8/KJcYqbrSY4/s1600/dese sperado.JPG))

Frase: E a solidão, sem família, trabalho ou amigos, tomou conta de sua vida...

**6ª Imagem: Deitado em um Divã numa consulta a um psicólogo**



**Figura 5.7 - Referência - Buscando ajuda**  
(fonte: <http://i.ehow.com/images/a00/0h/m5/become-psychiatrist-800X800.jpg>)

Frase: E com ajuda começou a exorcizar seus fantasmas...

**7ª Imagem: Em frente a uma agência de Investigações**



**Figura 5.8 - Referência – Recomeço**  
(fonte: [http://images03.olx.com.mx/ui/9/69/22/1289171350\\_88776722\\_2-Fotos-de--detective-e--investigador-privado-Oscar-3112617150-infieles-tepic-1289171350.jpg](http://images03.olx.com.mx/ui/9/69/22/1289171350_88776722_2-Fotos-de--detective-e--investigador-privado-Oscar-3112617150-infieles-tepic-1289171350.jpg))

Frase: E um novo trabalho o ajudou a olhar pra frente...

Mais uma vez, o tom da música é mudado para inspirar coragem.

**8ª Imagem: Recebe o caso de uma foto de uma menina**



**Figura 5.9 - Referência - Investigação de Desaparecimento**  
(fonte: [http://www.fareastgizmos.com/entry\\_images/0108/31/think-geek-digital-photo-frame.jpg](http://www.fareastgizmos.com/entry_images/0108/31/think-geek-digital-photo-frame.jpg))

Frase: Mas será que o destino estaria lhe dando uma nova chance ou pregando uma peça?

**9ª Imagem: A mansão assombrada**



**Figura 5.10 - Referência - Primeiro contato com a mansão**  
(fonte: <http://www.tu-pc.com/fondos/media/2817.jpg>)

Frase: A esperança de salvar esta garota e alcançar a sua redenção o trazem a este lugar, onde sua história será reescrita...

### 5.2.1.3 Parte 3: Nome do Jogo

A tela mostrando a mansão será congelada e, lentamente o nome do jogo (“Late Redemption”), escrito em letras que lembrem sangue escorrendo será mostrado juntamente com a mensagem: “Press Enter key”.

Ao apertar qualquer tecla, a Tela-7 (veja descrição abaixo) será mostrada.

Se o jogador pressionar a tecla “Enter” durante a apresentação do enredo, a Tela-7 também será mostrada.

### 5.2.2 Cena 2: Início do jogo até a entrada na sala do puzzle

Esta cena mostra a mansão. Marshall Gory, entra com cautela pela porta da frente, tenta abrir a primeira porta de um corredor, mas ela está trancada; em seguida consegue abrir uma segunda porta e entra. Neste momento, ele ouve um barulho de porta se fechando e claramente é possível notar uma explosão no quadro de distribuição de energia da sala e a luz se apaga (ficando a iluminação restrita àquela produzida pela lareira presente na sala), gritos de uma menina são ouvidos atrás da porta e a cena termina, retornando o controle ao jogador.

Nesta cena, a câmera seguirá o mesmo esquema adotado durante o jogo (*shoulder camera*) e, a todo o momento, Marshall Gory está com a sua arma em punho. Sua movimentação é cuidadosa, uma vez que ele ainda não tem idéia do que o espera.

A fim de transmitir a sensação de suspense e tensão desejada, os efeitos sonoros de porta fechando, explosão do quarto de luz e, principalmente os gritos da menina serão fundamentais.

### 5.2.3 Cena 3: Transição entre a fase 1 e fase 2

Assim que Marshall Gory consegue desvendar todos os elementos do puzzle da Fase 1 ele encontra a chave que será usada para abrir a porta e sair do quarto.

No momento em que esta chave é usada, inicia-se esta cena que mostra Marshall saindo do quarto do puzzle e voltando ao corredor principal da mansão.

Alguma coisa está claramente diferente e, seus olhos deparam-se com um rastro de sangue no chão imediatamente à sua frente. Seguindo este rastro de sangue, é possível observar que ele parece terminar em uma das portas do corredor.

Ainda imóvel e com o olhar fixo nesta porta, Marshall Gory percebe um barulho que assemelha-se ao som de porcos alimentando-se vindo do outro lado do corredor e, conforme seus olhos viram-se acompanhando o rastro de sangue agora, na direção contrária, ele terá a visão do primeiro grupo de *Shorties* devorando um cadáver próximo da porta de entrada da Mansão.

A cena termina com os *Shorties* detectando a presença de Marshall Gory e, iniciando um ataque contra ele.

### 5.2.4 Cena 4: Transição entre fase 2 e fase 3

Depois de vencer o guardião da fase final (Crusher), Marshall Gory terá acesso ao último aposento da mansão: *The Library*.

Esta cena inicia-se quando Marshall, ao ver a menina Silmara entrando pela porta da *Library*, decide seguí-la, para alcançá-la. Ao abrir a grande porta de acesso à *Library*, Marshall vê a pequena garota andando na direção de um enorme buraco aberto no centro da biblioteca. É um buraco muito grande e profundo, e pelas características de suas bordas, subentende-se que foi aberto de dentro para fora. Fumaça e um cheiro forte de enxofre saem de dentro desse buraco.

Ao ouvir os gritos de Silmara, Marshall percebe que eles não vêm da direção da menina em sua frente, mas sim de cima. Quando olha na direção dos gritos, vê uma jaula pendurada no teto, com uma menina parecida com a que está caminhando na sua frente, porém de semblante mais limpo, ameno. Surpreso, Marshall se pergunta se seriam duas meninas que teriam sido raptadas.

Enquanto caminha na direção da garota à sua frente, Marshall se espanta ao perceber que ela parece não ter intenção de parar quando está chegando perto do buraco. Ao tentar correr para evitar que ela caia, Marshall tem uma grande surpresa: a menina começa a flutuar sobre o buraco, como se estivesse andando normalmente sobre um piso comum. Ainda sem se recuperar do espanto, Marshall fica atônito quando a menina vira e fica frente a frente com ele, momento em que levanta a cabeça e o olha nos olhos. Nesse instante, a fumaça saindo do buraco aumenta de forma espantosa, encobrindo a menina e chegando até a clarabóia da grande sala.

É no momento em que a fumaça começa a se dissipar que Marshall sente um terrível arrepião na espinha. Ele vê dois olhos grandes e vermelhos, com chamas incandescentes, olhando na sua direção. A fumaça termina de se dissipar e Marshall pode ver claramente o tronco de um monstro gigante à sua frente. Com grandes garras e chifres saindo de suas costas, o monstro é tão grande que só a parte de cima de sua cintura cabe dentro da sala, a parte de baixo tendo que ficar dentro do buraco.

Num breve intervalo, o grande demônio se apresenta, dizendo se chamar Diabolus, e que ele está ali para capturar a alma atormentada de Marshall, e que este não pode fazer nada para impedí-lo. Nesse momento, a batalha final tem início.

### **5.2.5 Cena 5: Final do jogo com derrota**

Existem duas versões para esta cena: uma variação no caso da derrota por tempo ou derrota por danos.

#### **5.2.5.1 Derrota por Tempo**

Assim que o tempo termina, o relógio mostrando exatamente três horas e trinta e três minutos é mostrado na tela que vai tornando-se escura, mantendo apenas Marshall Gory.

Ao fundo, ouve-se o grito desesperado de Silmara e, os mesmos grunhidos e gemidos que ele ouviu naquela terrível noite.

Imediatamente um Flash-Back vem à sua mente e ao perceber que falhou pela segunda vez o desespero toma conta dele que cai de joelhos...

A cena vai apagando-se com Marshall olhando para o cano de sua arma e aos poucos, sua imagem vai ficando escura até desaparecer completamente.

Já com a tela totalmente escura, ouve-se um tiro e, em seguida a tela Game Over é mostrada.

#### **5.2.5.2 Derrota por Dano**

Dependendo da fase onde o personagem principal seja morto, uma versão diferente desta cena será mostrada:

Caso ele morra na Fase-1, depois de levar inúmeros choques no puzzle do quadro de distribuição de energia, uma animação mostrando ele sendo eletrocutado junto ao quadro de

energia será mostrada e, em seguida, seu corpo ficará estirado no chão. O ambiente ficará escuro, restando apenas a imagem de Marshall Gory inerte (ele passará a ser mais um dos cadáveres espalhados pela mansão).

Caso ele morra na Fase-2, em decorrência dos ataques dos monstros, uma animação mostrando um grupo de Shorties aproximando-se de seu corpo caído no chão para devorá-lo será mostrada (para esta versão, parte da Cena-3, onde também existe um grupo de Shorties devorando um cadáver, poderá ser usada como referência).

De forma semelhante à descrita na versão anterior, de agora em diante, Marshall Gory será apenas mais um cadáver espalhado pela mansão.

Caso ele morra na Fase-3, em consequência dos ataques do Diabolus, a animação mostrará Diabolus segurando Marshall Gory em suas mãos e, extraíndo a sua alma (uma fumaça branca poderá representar a alma de Marshall Gory deixando o seu corpo e, sendo absorvida pelo corpo do Diabolus). Em seguida, Diabolus soltará o corpo sem vida de Marshall Gory no chão e, olhará para a pequena Samara, presa em sua jaula que começará a gritar e, abruptamente a cena será interrompida.

Ao final de qualquer uma das versões, a mensagem “You are dead” será seguida da mensagem “Game Over”.

### 5.2.6 Cena 6: Final do jogo com vitória

Assim que o último tiro acertar o coração do Diabolus, esta cena será ativada e, mostrará o monstro sendo atingido por este último tiro e urrando de dor sua cabeça e seus braços perdem forças e, ele fica aparentemente imóvel e sem vida.

A seguir, um processo inverso ao descrito na Cena-4 será executado onde o Monstro irá desaparecer em meio à névoa que toma conta do ambiente e a frase: “Your soul...” é ouvida.

Ao aproximar-se do buraco, Marshall Gory não vê mais sinais dos monstros e, percebe que a chuva torrencial que caía cessou e uma linda lua cheia pode ser vista a partir das janelas da biblioteca.

Ele desprende as amarras que seguram a jaula onde Samara está presa e, assim que a jaula toca o chão, Samara sai de sua gaiola e corre em direção a Marshall Gory, seu salvador.

A cena termina com Marshall Gory abraçando a menina Samara e, entram os créditos (Cena-15).

### 5.2.7 Cena 7: Menu principal

Cena com as opções do menu principal. Possui as opções: “New Game”, “Input Keys”, “Load Game”, “Ranking”, “Credits” e “Quit”.

- A opção “New Game” levará o jogo à Cena-8.
- A opção “Input Keys” levará o jogo à Cena-9.
- A opção “Load Game” levará o jogo à Cena-10.
- A opção “Ranking” levará o jogo à Cena-11.
- A opção “Credits” levará o jogo à Cena-15.
- A opção “Quit” sairá do jogo.

O Background desta cena será o mesmo descrito ao final da Cena 1: A mansão onde desenvolve-se o jogo e em destaque, o nome: Late Redemption.

### 5.2.8 Cena 8: Níveis de dificuldade

Cena com as opções para escolha do nível de dificuldade. Possui as opções: “Go Easy On Me” (previamente selecionada) e “Hurt me Plenty”.

Está presente também a opção “Continue” que ao ser selecionada, levará o jogo diretamente para a Cena-2.

O Background desta cena será o mesmo descrito ao final da Cena 1: A mansão onde desenvolve-se o jogo e em destaque, o nome: Late Redemption.

### 5.2.9 Cena 9: Controles disponíveis

Como não será possível alterar as configurações pré-determinadas, esta cena irá mostrar uma figura contendo um teclado e um mouse com setas indicando a função de cada uma das teclas/botões determinados na seção **6.2**.

A opção “Back” retorna o jogo à Cena a partir da qual a chamada para a Cena-9 foi estabelecida.

A imagem escolhida ao fundo deverá ser compatível com a ambientação do jogo.

### 5.2.10 Cena 10: Carregar Jogos

Cena que apresenta a lista de jogos salvos.

Se algum dos jogos listados for selecionado e, a opção “Continue” for acionada, o jogo será automaticamente redirecionado para aquele escolhido.

A opção “Back” retorna o jogo à Cena a partir da qual a chamada para a Cena-10 foi estabelecida.

A imagem escolhida ao fundo deverá ser compatível com a ambientação do jogo.

### 5.2.11 Cena 11: Ranking

Cena com o Ranking do jogo, separando entre o nível normal e difícil mostrando uma lista com as 5 melhores pontuações em cada modo de jogo.

- Os Ranking pode ser apresentado em dois momentos distintos:
- A partir do menu principal : Aqui, ao pressionar a opção “Continue”, o jogo retornará automaticamente à Cena-7.
- Quando o jogo é finalizado com vitória: Logo depois que a apresentação dos créditos é finalizada, duas situações podem ocorrer:
  - A pontuação do jogador está entre as 5 melhores para aquele nível: Será mostrada uma tela informando a pontuação atingida e, solicitado ao Jogador inserir informações para identificar a sua pontuação. Na sequência, a Cena mostrando o Ranking (já atualizado) será apresentada na tela.
  - A pontuação do jogador não está entre as 5 melhores para aquele nível: Sua pontuação será mostrada na tela apenas com a opção “Continue” que ao ser acionada levará à tela informando o Ranking atual.

A imagem escolhida ao fundo deverá ser compatível com a ambientação do jogo.

### 5.2.12 Cena 12: Menu 2

Cena com o menu 2, apresentando somente durante o jogo.

Possui as opções: “Resume”, “Quit”, “Input Keys”, “Save Game” e “Load game”.

- A opção “Resume” retornará normalmente ao jogo.
- A opção “Quit” sairá do jogo.
- A opção “Input Keys” levará o jogo à Cena-9.
- A opção “Save Game” levará o jogo à Cena-13.
- A opção “Load Game” levará o jogo à Cena-10.

O Background desta cena será mostrará Marshall Gory com a sua arma em punho.

### 5.2.13 Cena 13: Salvar Jogo

Cena que pede o nome do ponto de salvamento para gravar a situação do jogo.

Tanto nos casos em que o procedimento seja completado com sucesso (ou seja, o jogo tenha sido salvo) ou quando opção “Back” é acionada, o jogo retornará à Cena-12.

A imagem escolhida ao fundo deverá ser compatível com a ambientação do jogo.

### 5.2.14 Cena 14: Puzzle do quadro de distribuição de energia

Cena que apresenta o quadro de distribuição e permite a mudança de lado das chaves, conforme figura **Figura 5.3**.

Uma vez que esta cena fará parte do mecanismo deste *puzzle*, ela será composta de várias imagens com as posições possíveis das chaves que serão carregadas à medida que o jogador prossegue na resolução do *puzzle* ao modificar as combinações entre as chaves.

Caso a combinação resultante dispare uma descarga elétrica (de acordo com o mecanismo do *puzzle* descrito na fase 5.2.2), o som característico será emitido, seguido de um grito de Marshall Gory, indicando que ele recebeu esta descarga.

### 5.2.15 Cena 15: Créditos

Nesta cena, lentamente o nome de cada um dos desenvolvedores do jogo será apresentado na tela. O nome será acompanhado de sua foto e uma pequena lista das principais atividades desenvolvidas durante o projeto.

O fundo desta cena será preto, e, imagens do concept art criadas para o jogo também serão intercaladas com as fotos dos desenvolvedores.

Os Créditos podem ser apresentados em dois momentos distintos:

- Quando o jogo é finalizado com vitória: Neste caso, a mensagem “Happy End” será mostrada com Marshall Gory e Silmara, de costas, andando de mãos dadas. Na sequência, o procedimento relacionado ao *Ranking* do jogo será ativado.
- A partir do menu principal : Aqui, ao invés de mostrar a tela “Happy End” o jogo retornará automaticamente à Cena-7.

## 5.3 Horas de Jogo

Como limite máximo de duração do jogo, inicialmente está previsto um tempo de 20 minutos. Dentro do universo do jogo, este tempo será de 3 horas e 33 minutos. O tempo real deverá ser convertido para o tempo do jogo. Para o personagem principal, o jogo é iniciado à meia-noite e o prazo limite para término é às 3h33m da manhã (hora do demônio).

Caso a condição de vitória não tenha sido atingida até 20 minutos, o jogo terminará, caracterizando uma derrota por tempo.

## 5.4 Condições de Vitória

O jogador será considerado vencedor do jogo se permanecer vivo e vencer os obstáculos apresentados nas três fases.

Na fase 1, Marshall deverá descobrir como sair da sala da mansão. Duas condições devem ser satisfeitas para que o jogador vença esta fase:

1. Marshall não pode morrer pelos danos causados pelo quadro de distribuição;
2. Deve sair do quarto antes das três horas e trinta e três minutos da manhã.

Na fase 2, Marshall deverá percorrer parte da mansão, enfrentando inimigos sobrenaturais. Duas condições devem ser satisfeitas para que o jogador vença esta fase:

1. Marshall não pode morrer pelos danos causados pelos inimigos;
2. Deve passar pelo *Corridor* e pelo *Main Hall*, entrando na biblioteca antes das três horas e trinta e três minutos da manhã.

Na fase 3, Marshall deverá enfrentar Diabolus. Duas condições devem ser satisfeitas para que o jogador vença esta fase:

1. Marshall não pode morrer pelos danos causados por Diabolus;
2. Deve derrotar Diabolus, salvando Silmara antes das três horas e trinta e três minutos da manhã.

Ao vencer as três fases, o jogador será considerado vitorioso.

## 5.5 Morte de Marshall Gory

Caso o personagem morra na fase 1, mas já resolveu parte do puzzle ou conseguiu alguns dos itens necessários para a resolução do mesmo, ele volta ao jogo no estado antes da morte, porém, o tempo de jogo é reiniciado para a 0 hora.

Caso o personagem morra na fase 2 na parte do Corridor, ele volta ao jogo em frente à porta da qual ela saiu da fase 1 com a munição daquele momento (é feito um checkpoint quando ele passa da fase 1 para a fase 2, salvando seu estado), porém, o tempo de jogo é reiniciado para a 1 hora da manhã. Caso o personagem morra na parte do *Main Hall*, ele volta ao jogo em frente à porta da qual ele saiu do *Corridor* com a munição daquele momento (é feito um checkpoint quando ele passa do *Corridor* para o *Main Hall*, salvando seu estado), porém o tempo de jogo é reiniciado para as 2 horas da manhã.

Caso o personagem morra na fase 3, ele volta ao jogo em frente à porta da *Library* com a munição daquele momento (é feito um checkpoint quando ele passa da fase 2 para a fase 3, salvando seu estado), porém, o tempo de jogo é reiniciado para as 3 horas da manhã.

## 5.6 Salvar e Carregar

O jogador poderá salvar o jogo a qualquer momento. Para isso deverá acionar um menu específico para esta ação e informar um nome para a unidade de salvamento de informações do jogo. Esta unidade conterá todas as informações sobre o posicionamento dos itens, inimigos, inventário e tempo de jogo presentes no momento da ação de salvar.

O jogador poderá reiniciar um jogo salvo anteriormente a qualquer momento. Para isso deverá acionar um menu específico para esta ação e escolher o nome da unidade de salvamento de jogo desejada.

Haverá também alguns pontos salvos automaticamente pelo jogo. Em caso de morte por danos, o personagem principal voltará para os pontos estabelecidos e haverá um acerto no tempo de jogo. Esta dinâmica está detalhada na seção **5.5 - Morte de Marshall Gory**.

# 6 Mecânica do jogo

Este capítulo apresenta os princípios de como o jogo funciona, ou seja, as regras envolvidas na maneira de jogar.

## 6.1 Jogabilidade

Conforme já foi citado, o jogador sempre visualiza o personagem principal em um modelo chamado *Shoulder Camera*, em terceira pessoa e por cima do ombro do personagem. A jogabilidade aqui definida visa propiciar uma forma agradável de jogar ao utilizar *Shoulder Camera* no contexto do jogo *Late Redemption*. Os tópicos abaixo descrevem os movimentos e ações que Marshall Gory, controlado pelo jogador, pode executar:

- Movimentação básica – Marshall anda sempre em um ritmo de corrida leve. É possível andar para frente, de costas, lateralmente, e virar para os lados mudando a direção para onde está andando e olhando. Faz parte da movimentação básica também ficar em um ponto fixo, apenas girando o corpo e olhando o ambiente em volta.
- Interação com objetos dos cenários – ao chegar perto de alguns objetos de cenários, é possível que apareça um texto indicando que é possível interagir com os mesmos. Esta interação pode servir, por exemplo, para pegar um objeto e guardá-lo no inventário do personagem, obter alguma informação a respeito do item ou executar uma ação, tal como abrir uma porta. Algumas ações eventualmente podem solicitar o uso de um dos itens do inventário como, por exemplo, uma chave.
- Utilização de armas e ações de combate – Marshall pode usar uma arma de fogo que tem consigo desde o início do jogo. A seção **6.4 - Sistema de Combate** descreve em mais detalhes as ações que podem ser executadas utilizando armas. Ao empunhar a arma de fogo o personagem é capaz de atirar em um ponto onde mirar. Além disso, as armas podem ser recarregadas a qualquer momento. É possível sacar e guardar a arma.
- Escolha do tipo de munição utilizado - uma vez que o jogador pode utilizar tanto munição normal como especial para sua arma, ele pode também escolher o tipo de munição que deseja utilizar no momento. A ação de escolha de munição equivale em tempo a uma ação de recarregar a arma.
- Acesso ao inventário - o jogador pode visualizar um inventário contendo os itens que Marshall carrega consigo. Ao acessar o inventário, o jogo se manterá paralisado.

## 6.2 Controles

*Late Redemption* deve ser jogado utilizando-se o mouse e o teclado em conjunto. Descreve-se abaixo o conjunto de ações que podem ser executadas no jogo, e os seus respectivos comandos.

### 6.2.1 Movimentação do personagem principal

As teclas “W”, “S”, “A” e “F” são utilizadas para movimentar Marshall Gory nas 4 direções básicas, respectivamente: para frente, para trás, para a esquerda e para a direita. A mudança de direção é feita utilizando movimentos do mouse.

## **6.2.2 Mira**

Movimentos do mouse.

## **6.2.3 Atirar**

Botão esquerdo do mouse.

## **6.2.4 Recarregar**

Tecla “R” ou botão direito do mouse.

## **6.2.5 Alternar entre os tipos de munição normal e especial**

Teclas “1” e “2” ou rolagem no mouse *wheel*.

## **6.2.6 Ações do Personagem Principal**

As seguintes ações poderão ser executadas pressionando a tecla “E”

- coletar itens;
- abrir portas;
- interagir com elementos de cenário.

## **6.2.7 Acessar tela do inventário**

Tecla <TAB>.

## **6.2.8 Acessar menu do jogo**

Tecla <ESC>.

## **6.2.9 Salvar o jogo**

Pode-se salvar o jogo por duas maneiras:

- acessando-se a opção “Save Game” dentro do menu 2 do jogo;
- utilizando-se o atalho de teclado F2.

Todos os itens e munição encontrados pelo personagem são salvos.

## **6.2.10 Carregar o jogo**

Pode-se carregar um jogo previamente salvo por duas maneiras:

- acessando-se a opção “Load Game” dentro do menu principal ou do menu 2 do jogo.
- utilizando-se o atalho de teclado F3

O ato de carregar o jogo depende do ponto onde ele foi salvo.

Se o jogo foi salvo na 1<sup>a</sup> fase, ao carregar o jogo, o personagem aparece na sala com todos os itens e munições salvos. Apenas o detalhe que caso ele já tenha acionado o evento de contagem de tempo do jogo antes de salvar (0 hora até as 3 horas e 33 minutos da manhã), então ao carregar o jogo o tempo será reiniciado para a 0 hora.

Se o jogo foi salvo no Corridor da 2ª fase, ao carregar o jogo, o personagem aparece na frente da porta da qual ele saiu da 1ª fase com o tipo e quantidade de munições salvas, porém, o tempo de jogo é reiniciado para a 1 hora da manhã.

Se o jogo foi salvo no Main Hall da 2ª fase, ao carregar o jogo, o personagem aparece na frente da porta da qual ele saiu do Corridor com o tipo e quantidade de munições salvas, porém, o tempo de jogo é reiniciado para as 2 horas da manhã.

Se o jogo foi salvo no 3ª fase, ao carregar o jogo, o personagem aparece na frente da porta da Library com o tipo e quantidade de munições salvas, porém, o tempo de jogo é reiniciado para as 3 horas da manhã.

### 6.3 Níveis de Dificuldade

O jogo apresenta dois níveis de dificuldade entre os quais o jogador pode escolher: “Go Easy On Me” (classificado como normal) e “Hurt Me Plenty” (classificado como difícil). Os aspectos do jogo afetados pela escolha da dificuldade são explicados na tabela abaixo:

Item	“Go Easy On Me”	“Hurt Me Plenty”
Puzzle	4 dicas em forma de pensamentos do personagem principal são apresentados para ajudar o jogador a sair da sala	Nenhum pensamento do personagem principal ajudará na resolução dos puzzles.
Número de Inimigos na Segunda Fase	10 no Corridor, 14 no Main Hall	15 no Corridor, 20 no Main Hall
Tempo disponível para completar o jogo	20 minutos	20 minutos
Sistema de Danos - Personagem Principal	Recuperação automática de pontos de vida (10 pontos de vida recuperadas a cada 1 segundo após ficar 5 segundos sem tomar dano)	Recuperação automática de pontos de vida (3 pontos de vida recuperadas a cada 1 segundo após ficar 15 segundos sem tomar dano)
Quantidade inicial de pontos de vida do personagem principal	100	85
Quantidade de pontos de vida, dano causado, velocidade de ataque e velocidade de deslocamento dos inimigos	Valores descritos no capítulo de inimigos	Valores descritos no capítulo de inimigos, sempre maior do que na dificuldade Go Easy On Me
Quantidade de pontos de vida, dano causado, velocidade de ataque do	Valores descritos no capítulo do chefe final	Valores descritos no capítulo de inimigos, sempre maior do que na dificuldade

chefe final		Go Easy On Me
Tempo de reaparecimento de munição normal	Os pentes de munição normal e especial reaparecem depois de 45 segundos de terem sido recolhidos.	Os pentes de munição normal reaparecem depois de 90 segundos de terem sido recolhidos. Os pentes de munição especial não reaparecem depois de terem sido recolhidos.
Quantidade de locais onde aparecem munição normal	6 locais, 2 no Corridor, 2 no Main Hall e 2 na Library.	4 locais, 1 no Corridor, 1 no Main Hall e 2 na Library.
Quantidade de locais onde aparecem munição especial	2 locais, 1 no <i>Main Hall</i> e 1 na <i>Library</i> .	1 local no <i>Main Hall</i> .

*Tabela 6.1 -Níveis de dificuldade do jogo Late Redemption*

## 6.4 Sistema de Combate

Durante o jogo, Marshall Gory irá se deparar com diversos inimigos e inevitavelmente precisará combater alguns deles. Para isso, ele pode utilizar sua Glock 34 semi-automática 9 mm que possui desde o início do jogo. Os inimigos, por sua vez, atacarão o personagem principal utilizando uma ou mais das maneiras descritas abaixo:

- desferindo golpes utilizando partes do seu corpo;
- por meio de magias.;

Alguns inimigos podem estar em grupos, e neste caso, é possível que vários inimigos ataquem o personagem principal simultaneamente.

Cada ataque no jogo, tanto do personagem principal quanto dos inimigos, inflige uma quantidade de dano à vida do oponente, fato que será descrito nas próximas seções. Os tópicos abaixo descrevem em mais detalhes alguns elementos envolvidos no combate.

### 6.4.1 Classificação de tipos de ataque

Os ataques podem ser classificados conforme descrito abaixo.

Quanto ao estilo do ataque:

- Combate desarmado – combate sem utilização de armas, utilizados por alguns inimigos.
- Armas brancas – armas cortantes ou contundentes (facas, porretes, pedaços de madeira). Este tipo de ataque é utilizado apenas por inimigos.
- Armas de fogo – armas que utilizam pólvora. A arma disponível para o personagem principal é uma Glock 34 semi-automática 9 mm
- Magias – ataques mágicos que podem ser conjurados por alguns inimigos

Quanto ao tipo de dano:

- Dano por perfuração ou corte – dano causado pelas armas de fogo, por armas brancas que possuem lâmina, ou por inimigos que possuam garras.

- Dano por contusão – dano causado pelo ataque desarmado e pelas armas brancas que não possuem lâmina.
- Dano por elemental – alguns inimigos serão capazes de conjurar magias (ex: bola de fogo), e o dano causado por estas magias é chamado de dano por elemental.

Quanto à distância:

- Ataque de curta distância – utilizado quando os oponentes estão frente a frente, próximos um do outro. Entram nesta categoria o combate desarmado e as armas brancas que não são lançadas na direção do oponente. Os ataques desarmados e de armas brancas utilizados pelos inimigos são ataques de curta distância.
- Ataque de longa distância – pode ser utilizado quando o inimigo se encontra a mais de um metro e meio de distância. Entram nesta categoria as armas de fogo e ataques mágicos, como as bolas de fogo lançadas por alguns inimigos.

## 6.5 Sistema de danos

A forma como o dano é tratado no jogo está descrita abaixo. Note que a quantidade de dano causada por cada elemento não está descrita neste capítulo e sim nas seções que descrevem os próprios elementos (personagem principal, armas, inimigos, etc).

### 6.5.1 Dano causado no personagem principal

A maneira como o dano causado no personagem principal é tratada de acordo com a dificuldade selecionada no jogo, conforme será descrito a seguir.

#### 6.5.1.1 Sistema de dano na dificuldade “Go Easy On Me”

Marshall tem no início do jogo 100 pontos de vida. Durante o jogo, caso ele receba danos provenientes de ataques de inimigos ou armadilhas, sua quantidade de vida irá diminuir de acordo com o tipo de ataque recebido.

A recuperação dos danos recebidos ocorre automaticamente pela regeneração por tempo: ao receber algum dano, após passar o tempo de 5 segundos, os pontos de vida de Marshall começam a se regenerar automaticamente. Se no período em que ainda não se regenerou totalmente Marshall receber mais danos, os danos são computados com base no valor atual dos pontos de vida do mesmo, ou seja, se o personagem principal tiver apenas 30 pontos de vida e receber 3 danos de 10 pontos cada, sem que haja tempo para se recuperar, terá seus pontos de vida esgotados e morrerá. Note que a recuperação do dano sofrido só é iniciada após 5 segundos sem o sofrer novos danos, assim como descrito na seção **6.3 - Níveis de Dificuldade**.

Este sistema de danos incentiva o jogador a ser cauteloso e estratégico em suas ações, para que não receba muito dano em um curto espaço de tempo. Nesta abordagem, o personagem principal não precisa se preocupar em achar itens de cura para o restabelecimento dos seus pontos de vida a todo momento durante o andamento do jogo, tornando o jogo mais dinâmico e focado no objetivo principal.

### **6.5.1.2 Sistema de dano na dificuldade “Hurt Me Plenty”**

A diferença no sistema de danos neste nível de dificuldade Marshall só irá começar a recuperar-se dos danos recebidos após ficar 15 segundos sem tomar danos, e ao invés de recuperar 10 pontos de vida por segundo, irá recuperar apenas 3.

### **6.5.2 Dano causado por elementos do cenário**

Não somente os inimigos são capazes de causar dano. Em alguns casos, a própria interação com elementos do cenário pode causar dano, como é o caso, por exemplo, do quadro de distribuição de energia descrito na fase 1, onde um erro do personagem na execução do desafio do puzzle causará uma quantidade determinada de dano.

### **6.5.3 Dano causado nos inimigos**

Os inimigos não têm a capacidade de regeneração automática de pontos de vida. Por este motivo, a quantidade de vida de alguns inimigos, principalmente dos mais fortes, será maior do que a do personagem principal.

Cada tipo de ataque ou arma utilizada por Marshall causa uma quantidade de dano pré-determinada conforme determinado na seção que descreve a arma do personagem. Porém, um determinado inimigo pode ter resistência maior ou menor a um determinado tipo de ataque, o que altera a quantidade de dano causada.

## **6.6 Estatísticas de jogo e classificação de jogadores**

O jogo possuirá um ranking que mostra os 5 melhores desempenhos obtidos no jogo. Este ranking possuirá uma divisão entre o nível normal e o nível difícil.

Ao finalizar o jogo em condição de vitória, o jogador deverá informar um texto para identificar sua pontuação no ranking de jogadores.

O ranking poderá ser consultado a qualquer momento através de uma opção de menu.

A pontuação que medirá o desempenho dos jogadores, é baseada nos seguintes itens:

- Tempo gasto, em segundos, para salvar a menina sequestrada (T);
- Percentual de ataques certeiros (A);
- Quantidade de danos tomados (D).

Com base nestes itens, a seguinte fórmula será usada para calcular os pontos (P) obtidos pelo jogador:

$$P = (1200 - T) * 6 + A * 66 - D * 6$$

### **6.6.1 Exemplo**

Na tabela abaixo é apresentado o exemplo de pontuação de um jogador que terminou as três fases em 18 minutos e 30 segundos, tomou 300 pontos de dano e teve uma precisão de 90% nos ataques.

Item	Quantidade	Pontuação
------	------------	-----------

Tempo gasto	18m30s	540
% de ataques certeiros	90	5940
Danos tomados	300	1800
	Pontuação final	4680

*Tabela 6.2 -Exemplo da pontuação final de um jogador*

## 7 Interface

Os seguintes elementos de interação entre o jogador e o jogo são descritos neste capítulo: heads-up display, menus e inventário do personagem.

### 7.1 HUD (Heads-up Display)

O jogo fornecerá na tela informações ao jogador sobre a arma e as munições que o personagem possui. Além disso, a informação da hora fictícia do jogo será demonstrada no canto superior direito da tela de tempos em tempos para que o jogador esteja ciente desse fato. Essas informações são demonstradas através do HUD (Head up Display). A **figura 8.1** mostra a disposição do HUD em na tela.



*Figura 7.1 - Elementos do HUD*

No canto inferior direito da tela haverá um ícone da arma do personagem junto com um valor numérico em um círculo. Esse valor representa a quantidade de munição na arma.

A barra circular verde indica que a arma está totalmente carregada e conforme vai sendo descarregada, essa barra vai mudando de cor, avermelhando-se até a quantidade de munição na arma for zero.

Ao lado temos o ícone da munição corrente que está sendo utilizada na arma, representado por um pente em um círculo menor. Ela indica a quantidade de munição que o jogador possui para quando acabar a munição da arma. Quando a munição da arma chega a zero, é feita uma transferência dessa munição para a arma, incrementando o valor da arma e decrementando o valor do pente.

Um pouco acima desse ícone temos um outro representado por um pente de munição também, porém, fora de um círculo. Ela indica um outro tipo de munição onde o jogador pode trocar de munição quando desejar.

No canto superior direito haverá um relógio que mostrará o horário em que se passa o jogo, mas o tempo que aparece não se trata do tempo real. Ele é iniciado à meia-noite e vai até as 3 horas e 33 minutos. O relógio não estará presente na tela o tempo todo, de modo que alguns eventos irão ativar temporariamente sua exibição: badaladas de hora em hora (meia-noite, 1, 2 e 3 da manhã), alguns trovões e gritos de socorro de Silmara que ocorrerão randomicamente. Ele será exibido por 10 segundos e depois desaparece da tela. A ideia do relógio não ser exibido sempre é justamente causar uma sensação de medo no jogador, que não terá certeza em todo momento sobre o tempo que ainda lhe resta.

### **7.1.1 Exibição da vida Marshall**

Não existe uma barra de vida para o personagem. Ao invés disso, quando ele sofre dano, a tela vai escurecendo no tom vermelho até essa cor preencher toda a tela e exibir a mensagem "You are dead" indicando que o personagem morreu. Durante o tempo em que o personagem sofre dano e a tela vai escurecendo, caso ele consiga de alguma forma evitar o dano (derrotando o inimigo, ou se escondendo) a tela vai, aos poucos, retomando sua tonalidade normal indicando que o personagem está se curando.

## **7.2 Menus**

O jogo possuirá dois menus para interação com as opções do jogo. O menu principal será apresentado logo após o carregamento do jogo. O menu 2 só poderá ser acessado durante o jogo.

### **7.2.1 Menu principal**

Este menu será usado para selecionar as opções iniciais do jogo. Possuirá as seguintes opções: "Load Game", "Input Keys", "Load Game", "Ranking" e "Quit". Para selecionar uma das opções, deverá ser utilizado o mouse.

A descrição física deste menu pode ser encontrada na seção **5.2.7 - Cena 7: Menu principal**. A descrição das funcionalidades está na seção **5.1 - Progressão do Jogo**.

### **7.2.2 Menu 2**

Este menu só poderá ser acessado durante o jogo e permitirá que o jogador realize uma das seguintes opções: "Resume", "Quit", "Load Game" e "Save Game".

A descrição física deste menu pode ser encontrada na seção **5.2.12 - Cena 12: Menu 2**. A descrição das funcionalidades está na seção **5.1 - Progressão do Jogo**.

## **7.3 Inventário do personagem principal**

Conforme descrito na seção **6.2 - Controles**, o jogador pode acionar um comando para acessar o inventário de Marshall Gory. Ao acessá-lo, o jogo fica temporariamente paralisado, voltando ao estado anterior quando o jogador sair do inventário. É possível visualizar os itens a seguir:

- Nome do cenário atual - o jogador pode visualizar o nome do cenário em que Marshall se encontra;
  - Itens de puzzle obtidos por Marshall - o jogador visualiza todos os itens Marshall tenha obtido no puzzle e ainda não tenha utilizado. É possível selecionar um destes itens para torná-lo o item ativo, o qual então pode ser utilizado no jogo;
  - Arma de Marshall Gory - é possível visualizar a arma de Marshall, porém não há interação com este item. Juntamente com a arma, é possível ver a munição em uso;
  - Munição armazenada - o jogador pode visualizar tanto a munição normal como a munição especial que Marshall leva consigo, de maneira semelhante a apresentada no HUD. Não há interação com este item no inventário.

A **Figura 7.2** demonstra uma arte conceitual de como será o inventário:



**Figura 7.2 – Inventário de Marshall**

# 8 Músicas e Efeitos Sonoros

Este capítulo lista e descreve brevemente as músicas e efeitos sonoros existentes no jogo. Um conceito importante neste sentido é que o jogo não terá músicas tocando sempre. Ao invés disso o silêncio se fará presente na maioria do tempo, porém com destaque para sons ambientes, como gritos, portas, combate e monstros, buscando assim alcançar um terror psicológico maior. As músicas serão adicionadas em momentos relevantes ou onde ela possa ser responsável por aumentar o suspense ou o terror.

## 8.1 Músicas

**Jingles de Eventos:** músicas para casos de derrota, morte, vitória.

**Shell Screens:** ambientação para telas de título e créditos.

**Título:** Música tenebrosa e impactante, com profunda ligação com o título do jogo e sua temática.

**Créditos:** Música impactante mas sem a mesma intensidade da música de título que deve passar mais fortemente a ligação com o jogo.

**Música para cutscenes:** As transições terão músicas que abrangem as duas cenas. A música começa com uma temática identificada com a primeira cena e aos poucos vai mudando para a temática ligada a segunda cena.

Ex. O personagem passa da fase 1 para a 2. A cena 1 seria o personagem achando que se livrou de seu problema e de repente se depara com um corredor mais perigoso e misterioso do que o quarto em que se encontrava.

Nesse caso a música começaria com tons mais vibrantes e passaria para melodias mais dramáticas e pesadas..

## 8.2 Efeitos Sonoros

**GUI:** É a parte que dará maior ambientação ao jogo. Possíveis efeitos são:

**Porta abrindo:** Porta rangendo enquanto se abre lentamente

**Porta trancada:** Barulho da maçaneta sendo forçada e a porta não se abre

**Porta destrancando:** Chave sendo virada e destrancando porta

**Personagem andando:** Som forte, curto e seco de pesadas botas (coturno) andando sobre assoalho.

**Personagem andando sobre carpete:** Som forte, curto e seco porém abafado devido ao tapete de pele

**Cadeira caindo:** som de madeira pesada caindo em um assoalho

**Trovão:** Estrondo forte e curto de aproximadamente 1 segundo de duração mais fade in de aproximadamente 3 segundos do mesmo som.

**Grito Menina:** grito estridente de medo e dor, distante e de curta duração

**Grito Screamer:** grito rouco, alto e curto

**Andar shortie:** som de passos curtos com unhas (garras) grandes batendo pelo chão

**Ataques garras shortie:** som de garras (ou unhas bem afiadas) rasgando a carne

**Ataques mordida shortie:** som de mordidas (ou algo bem afiado) rasgando a carne

**Andar Butcher:** um passo pesado e vagaroso seguido de um arrastar de uma perna prolongado.

**Ataque Butcher:** som de uma espada ou uma grande faca cortando o ar

**Andar Crusher:** passadas pesadas graves, como um grande estrondo que ecoa pelo ambiente

**Ataque Crusher:** passos pesados e vagarosos que ecoam a uma curta distância

Personagem sofrendo dano: um gemido abafado e curto

**Ataque garras Chefão:** uma grande pancada com som abafado

**Ataque distância chefão:** som parecido com o de um lança chamas ou maçarico.

**Grunhidos chefão:** grunhidos grotescos e roucos, volume baixo e de curta duração

**Arma disparando:** estampido seco com um pouco reverberação

**Reload arma:** som de pequenas peças de metal se chocando levemente

**Capturar item:** som de pequenas peças de metal se chocando levemente porém de duração mais curta que o som de reload

**Fogo lareira:** fogo crepitando

**Fogo apagando:** chiado longo com fade in (som vai diminuindo)

**Digitação senha dispositivo segurança:** som agudo e rápido para cada numero digitado (PI PI PI) - Senha digitada errada à um estridente pééééé

**Cofre destravando:** estampido seco de ferros se chocando

**Curto-circuito:** um zumbido baixo e continuo e pequenos estalos

# **9      Inteligência Artificial**

Conforme descrito a seguir, cada um dos monstros possui características próprias de ataque, o que exigirá um trabalho consistente na elaboração da Inteligência Artificial deste jogo, pois como demonstrado vários parâmetros deverão ser levados em conta.

Separadamente, serão descritos os padrões relacionados com cada um dos monstros, incluindo o chefe final.

## **9.1    Shortie**

### **9.1.1    Movimentação de Ataque/Defesa**

Como estes monstros iniciam seus ataques em grupos (três monstros), ao perceber a presença de Marshall (mais detalhes sobre este mecanismo no item a seguir), eles devem seguir em sua direção e iniciar uma perseguição, sendo que cada monstro irá procurar manter uma certa distância para os outros monstros do grupo (usar como referência de distância mínima entre um monstro e outro a metade da distância entre este monstro e Marshall). Deste modo, a estratégia de Marshall para escapar de tais ataques será mais complexa pois os inimigos atacarão a partir de ângulos diversos.

Com Marshall dentro do seu campo de ataque, o grupo de Shorties atacará com uma sequência de 3 ataques seguidos, com intervalo de 2 segundos entre cada sequência de ataques, na dificuldade “Go Easy on Me”, e uma sequência de 3 ataques seguidos, com intervalo de 1 segundo entre cada sequência de ataques na dificuldade “Hurt Me Plenty”. Sempre que terminar uma sequência de ataques, os Shorties recuarão, momento este em que ficarão vulneráveis aos ataques de Marshall.

Sempre que um monstro do grupo for atingido, todos eles, em um comportamento irracional, irão dirigir-se com todo o vigor na direção de Marshall, a fim de revidar.

Uma vez que este monstros iniciarem a perseguição ao pobre Marshall, eles irão parar em apenas duas situações:

- Quando eles morrerem
- Quando Marshall morrer

### **9.1.2    Evento de Açãoamento**

Uma vez que estes monstros estão sempre ocupados alimentando-se dos cadáveres espalhados pelos corredores, eles apenas iniciarão os seus ataques nas situações a seguir:

- Quando um tiro for disparado contra eles e, atingir qualquer um dos três elementos do grupo
- Quando Marshall chegar a menos de dois metros deles pois, neste caso, o seu cheiro será notado pelo grupo sedento por sangue fresco.

### **9.1.3    Controle do Nível de Dificuldade**

Para variar o nível de dificuldade dos ataques destes monstros, além das variações no dano causado por seus ataques, estes monstros terão os temporizadores variáveis associados aos seus comportamentos de ataque ou esquiva.

Conforme descrito anteriormente, o seu mecanismo de movimentação nos modos de ataque ou esquiva será composto por um conjunto de regras que podem ser modelados a partir de uma máquina de estados finita. Temporizadores estarão atuando a fim de modificar estes estados e, assim, modificar o comportamento dos monstros.

Assim sendo, no nível de maior dificuldade, os tempos de modificação de comportamento destes monstros serão menores se comparados com aqueles encontrados no nível com dificuldade menor (referência: 0,5 segundo a 1,5 segundos) e portanto, o comportamento observado poderá variar de: *totalmente selvagem* (nível difícil) até *monstro acéfalo* (nível fácil).

## 9.2 Screamer

### 9.2.1 Movimentação de Ataque/Defesa

Enquanto Marshall estiver no seu campo de visão, o monstro ativará o seu modo de ataque que consiste em manter uma distância segura para Marshall (ele tentará afastar-se de Marshall quando este estiver muito próximo e isto deverá ser feito em uma velocidade reduzida, para que seja possível a aproximação de Marshall) e, em intervalos de 3 segundos iniciar o procedimento de disparo que consiste em emitir um grito e em seguida emitir uma bola de fogo. O grito servirá de aviso para o personagem tentar esquivar-se da bola de fogo e, enquanto o monstro grita e dispara a bola de fogo, ele ficará imóvel (cerca de 2 segundos), contribuindo para a aproximação entre o monstro e Marshall.

Sempre que este monstro for atingido por um disparo da arma de Marshall, imediatamente uma bola de fogo será atirada contra Marshall, ignorando neste momento o parâmetro de dois segundos descrito acima e em seguida iniciará durante dois segundos uma movimentação preferencialmente lateral, para que as chances de ser atingido seguidamente sejam reduzidas.

Ele não causará dano a Marshall por contato e, por isto, a sua movimentação será preferencialmente a média-longa distância.

Todo disparo será seguido de uma pequena movimentação lateral, para dificultar o mecanismo de ataque/defesa de Marshall .

### 9.2.2 Evento de Açãoamento

Como este monstro possui um campo de ataque longo (seu campo de ataque pode chegar a 10 metros), assim que Marshall entrar em seu campo de visão, usando-se esta distância como referência, seus disparos serão iniciados.

### 9.2.3 Controle do Nível de dificuldade

Além dos parâmetros padronizados de dano sofrido/dano causado já explicados anteriormente, a variação do nível de dificuldade do jogo modificada também a velocidade de movimentação deste monstro. Ele terá uma velocidade padrão no nível de dificuldade “Go Easy On Me” e, mais rápido no nível de dificuldade “Hurt Me Plenty” (inspiração nos fantasmas do clássico jogo PAC-MAN da NAMCO).

## **9.3 Butcher**

### **9.3.1 Movimentação de Ataque/Defesa**

Como este monstro possui mobilidade reduzida, se comparado ao Screamer e ao Shortie, ele a princípio não irá perseguir Marshall, a não ser que ele esteja no seu restrito campo de atuação (outras exceções estão descritas no item a seguir). Desta forma, enquanto Marshall estiver no seu campo de atuação, ele tentará uma lenta aproximação e, uma vez que a distância mínima de ataque for alcançada, ele ativará o seu modo de ataque que consistem em tentar atingir Marshall com o seu afiado cutelo. Por ser um monstro grande, com movimentação lenta e ângulo de ataque reduzido, ele precisará sempre estar a curta distância do Marshall para atingí-lo e, como consequência, sua resistência deverá ser alta, para que situações de combate sejam possíveis.

Para não torná-lo um alvo tão vulnerável, sempre que ele for atingido por algum dos disparos de Marshall, ele irá entrar em um modo especial de defesa/ataque que consiste em, por um curto intervalo de tempo, dar a este monstro mobilidade cerca de três vezes maior do que a sua mobilidade normal, este é o efeito chamado *Revenge Timing*.

### **9.3.2 Evento de Acionamento**

Butcher é o guardião da porta que separa o *Corridor do Main Hall* e, em sendo assim, ele a princípio deve ficar em seu posto. No entanto, ele iniciará a sua caminhada em direção a Marshall quando o último monstro presente no corredor tiver sido abatido, quando ele tomar ao menos um tiro de Marshall ou quando Marshall estiver a menos de 1 metro de distância dele.

### **9.3.3 Controle do Nível de dificuldade**

De forma intuitiva, os níveis de dano causado/sofrido durante os combates serão ajustados juntamente com a escolha da dificuldade do jogo. O outro ponto a ser levado em consideração será a velocidade de reação que o Butcher terá durante o *Revenge Timing*. Caso tenha sido escolhido o nível de menor dificuldade, a amplitude da movimentação lateral (ou em direção a Marshall) será reduzida.

## **9.4 Crusher**

### **9.4.1 Movimentação de Ataque/Defesa**

O seu modo de ataque será semelhante ao do Butcher pois este monstro possui mobilidade ainda mais reduzida. No entanto, por ser maior e o seu tacape possuir um alcance maior do que o cutelo do Butcher, a sua distância mínima de ataque será maior, assim como o seu ângulo de ataque, que será mais amplo, em uma comparação direta com o Butcher.

Entretanto, também devido à sua força descomunal e ao incrível peso de seu tacape, o Crusher, no momento de seu ataque, acaba se deslocando para a frente, como resultado da inércia do movimento do golpe. Este efeito pode ser muito perigoso, uma vez que pode encurtar ainda mais a distância dos seus oponentes.

Uma outra diferença sutil com relação aos métodos de ataque do Butcher diz respeito ao seu Revenge Timing: haverá um balanço entre ataque e defesa onde o movimento de

esquiva será buscando mobilidade lateral e, movimento de ataque, o confronto direto com Marshall.

#### **9.4.2 Evento de Acionamento**

A tarefa que Diabolus deu ao Crusher foi proteger com sua própria vida o local onde ele, Diabolus, está repousando. Por esta razão, o Crusher estará a princípio plantado próximo da porta que divide o *Main Hall* e a *Library*, onde desenrola-se a terceira e última fase do jogo.

Mas, quando Marshall derrotar pelo menos metade dos inimigos alocados dentro do saguão, ou seja, quando ele eliminar metade do sub-exército montado pelo Crusher, ele iniciará sua própria caçada.

Se por outro lado, Marshall ousar chegar a menos de 2 metros do Crusher ou atirar nele, sua rotina de ataques também será iniciada.

#### **9.4.3 Controle do Nível de dificuldade**

Novamente, os níveis de dano causado/sofrido durante os combates serão ajustados juntamente com a escolha da dificuldade do jogo além do que haverá de acordo com o nível de dificuldade escolhido, uma variação na proporção dos movimentos de ataque/esquiva durante o *Revenge Timing*. Quanto maior for o nível de dificuldade, maior será a quantidade de investidas do Crusher para cima de Marshall.

Uma vez iniciado este movimento de ataque, ele apenas será interrompido quando:

- Marshall Gory for eliminado.
- Quando o Crusher for atingido novamente (e, neste último caso, um novo round do Revenge Timing será acionado)

### **9.5 Chefe final**

#### **9.5.1 Movimentação de Ataque/Esquiva**

O chefe final possui dois tipos de ataque: ele pode golpear com suas garras (ataque de curto alcance) ou lançar imensas bolas de fogo de sua boca.

##### **9.5.1.1 Golpes com garras**

Desde que Marshall Gory esteja dentro do alcance das garras, em intervalos de tempo periódicos (sugestão 1 segundo/0,8 segundos - de acordo com o nível de dificuldade), Diabolus deverá iniciar uma sequência de dois ataques, começando pelo braço mais próximo de Marshall.

##### **9.5.1.2 Bolas de Fogo**

Quando Marshall Gory estiver à média/longa distância de Diabolus, este deverá identificar o ponto onde Marshall encontra-se e disparar uma sequência de 3 bolas de fogo contra ele, sendo que cada bola de fogo terá uma trajetória fixa, mas calculada de acordo com a movimentação de Marshall. Por exemplo, se Marshall estiver caminhando para a direita no momento do início do ataque, percorrendo um trajeto A1, A2 e A3, a primeira bola terá como alvo o local onde Marshall estava no início do ataque (A1), a bola posterior corrigirá a trajetória, tentando acompanhar a movimentação de Marshall (A2), e o mesmo se repetirá

para a terceira e última bola (A3). O intervalo de tempo entre as sequências de ataques deverá ser o mesmo daquele descrito para os golpes com garras.

Assim sendo, a luta contra o chefe final será dinâmica e a estratégia usada para vencê-lo deverá obrigatoriamente combinar agilidade e precisão.

### **9.5.2 Evento de Acionamento**

Automático. Assim que Marshall Gory entrar pela porta da biblioteca, e a Cena apresentando o Chefe Final terminar, sua rotina de ataques será iniciada.

Como os dois tipos de ataque disponíveis para este monstro são distintos e não simultâneos, deve existir um mecanismo que alterne entre estes dois modos. Este mecanismo será baseado na distância entre o monstro e Marshall Gory (valor padrão 4 metros).

Assim sendo, enquanto Marshall Gory estiver ao alcance de suas garras (4 metros), o modo de ataque das garras será ativado e ele será automaticamente modificado para o modo de ataque das bolas de fogo se Marshall Gory afastar-se.

É importante mencionar que Diabolus, após ser atingido um determinado número de vezes no peito (número que depende do nível de dificuldade), entrará num estado de tontura, no qual não poderá atacar nem reagir a ataques de Marshall. São em momentos assim que Marshall poderá atingir o coração de Diabolus a fim de acabar com a vida deste. Após o momento de tontura passar, o escudo de proteção de seu coração se regenerará e a rotina de ataques recomeçará.

### **9.5.3 Controle do Nível de dificuldade**

Uma vez que a sua mobilidade é limitada, a modificação do nível de dificuldade não implica em mudança nos seus padrões de ataque.

No entanto, é possível variar a frequência dos seus ataques (tempo entre um ataque e outro). Seus ataques serão mais constantes (intervalo de tempo 20% menor até o próximo ataque) no nível mais difícil.

# 10 Detalhamento Técnico

Este capítulo descreve informações técnicas sobre o jogo, tratando principalmente de definir os requisitos mínimos para jogar e de mencionar as tecnologias utilizadas no desenvolvimento.

## 10.1 Requisitos mínimos para execução do jogo

Os requisitos mínimos para executar o jogo Late Redemption são:

- Windows XP SP2 or Windows Vista
- Processador de 2.0+ GHz
- 2 GB de memória RAM
- Placa de vídeo NVIDIA 8000 series ou maior
- 3 GB de espaço livre no disco rígido

Os requisitos recomendados são:

- Windows Vista 64 SP2 ou Windows 7
- Processador multi-core de 2.0+ GHz
- 8 GB de memória RAM
- Placa de vídeo NVIDIA 8000 series ou maior
- 3 GB de espaço livre no disco rígido

## 10.2 Tecnologias Utilizadas

Para realizar a criação do jogo será utilizada a engine UDK - Unreal Development Kit. Com releases mensais, cada versão apresenta funcionalidades novas que não são necessariamente compatíveis, portanto recomenda-se usar apenas uma versão específica. A versão adotada pelo grupo é a 7876, change list 776445 que se encontra neste link: <http://download.udk.com/UDKInstall-2011-02-BETA.exe>. Para verificar se a versão instalada é a correta, escolha a opção **Help → About Unreal Development Kit....** Lá, deverão constar: Version: 7876 Change List: 776445.

Para realizar a modelagem dos personagens, será utilizada a ferramenta Blender, versão 2.49b que se encontra neste link:

<http://www.blender.org/dl/http://download.blender.org/release//Blender2.49b/blender-2.49b-windows.exe>

## 10.3 Engine

A UDK é a engine da empresa Epic Games, responsável pelo desenvolvimento de diversos jogos famosos de mercado, como Gears of War (Epic Games - 2006), BioShock (2K Boston - 2006), Mass Effect 1 e 2 (Bioware - 2007, 2010), entre outros.

É uma engine livre para uso não comercial. Possui uma interface de desenvolvimento muito amigável e a comunidade de desenvolvedores é muito grande. Diversos tutoriais podem ser encontrados no site da Epic Games como em fóruns. No Brasil o fórum UDK Brasil,

<http://udkbrasil.blogspot.com/> é o ponto onde desenvolvedores brasileiros se encontram para trocarem experiências e compartilharem conhecimentos através de tutoriais.

A oportunidade de poder estudar uma engine de mercado é um grande motivador para aqueles que desejam ingressar no mercado de desenvolvimento de jogos que a cada dia vem crescendo no Brasil.

# 11 Testes

Testes serão necessários para validar todas as definições deste documento. O objetivo desta seção não é descrever os planos de testes, mas dizer em linhas gerais quais as características que devem ser testadas e qual deverá ser o processo de teste. O documento com a especificação dos testes a serem executados será elaborado em momento oportuno.

## 11.1 Estratégia de Testes

Haverá uma lista de atividades que devem ser desenvolvidas. Cada versão liberada do jogo deverá conter também uma lista de funcionalidades implementadas naquela versão.

As atividades de teste serão então focadas no teste das funcionalidades disponibilizadas, juntamente com testes de regressão nas funcionalidades da versão anterior que tenham relacionamento com as funcionalidades que acabaram de ser liberadas.

Os testes serão baseados na especificação do GDD, e deverão contemplar todas as características do jogo. O seguintes testes serão efetuados:

- Testes funcionais. Deverão avaliar o funcionamento segundo as especificações dos personagens e fases. Os diagramas de fluxo de fases, transições de cenário, comportamentos dos inimigos e comportamento do personagem principal devem ser exercitados.
- Testes de desempenho. Deverão levar em consideração os requisitos mínimos para funcionamento do jogo. Deverão ser testados itens como desempenho do jogo quando existem muitos inimigos no mesmo cenário. Testes em ambientes especificados devem ser exercitados.
- Testes de usabilidade. Deverão levar em consideração as características das fases, cenas e personagens, respondendo o seguinte checklist:
  - O jogo é fácil de jogar?
  - O jogo é muito difícil de jogar?
  - O jogo é fácil de aprender?
  - Os controles são intuitivos?
  - A tela do usuário é limpa e fácil de navegar?
  - O jogo é divertido?
  - O jogo está de acordo com o enredo e filosofia?

## 11.2 Processo

O processo de teste será dividido em 3 etapas:

- Identificação dos erros. Poderá ser executada por qualquer membro da equipe, preferencialmente os testadores da versão recém liberada.
- Correção dos erros. Deverá ser realizada por um desenvolvedor, preferencialmente aquele que implementou a funcionalidade com defeito.
- Re-teste de erros corrigidos. Poderá ser executado por qualquer membro da equipe, preferencialmente pelo próprio testador que identificou a falha.

### **11.2.1 Identificação de erros**

1. Identificar um eventual problema.
2. Preencher uma entrada no bugtracking, indicando: descrição detalhada para simulação do erro.
3. Neste passo, o responsável pelos testes deve priorizar as entradas do bugtracking.

### **11.2.2 Correção de erros**

1. Obter uma entrada, por prioridade, do bugtracking.
2. Atuar na reprodução e investigação do problema.
3. Se o problema pôde ser reproduzido, realizar correção do mesmo.
4. Se o problema foi corrigido, encaminhar para novos testes no bugtracking, identificando eventuais impactos da correção em outras funcionalidades. Senão, encaminhar para o responsável pelos testes.

### **11.2.3 Re-teste de erros corrigidos**

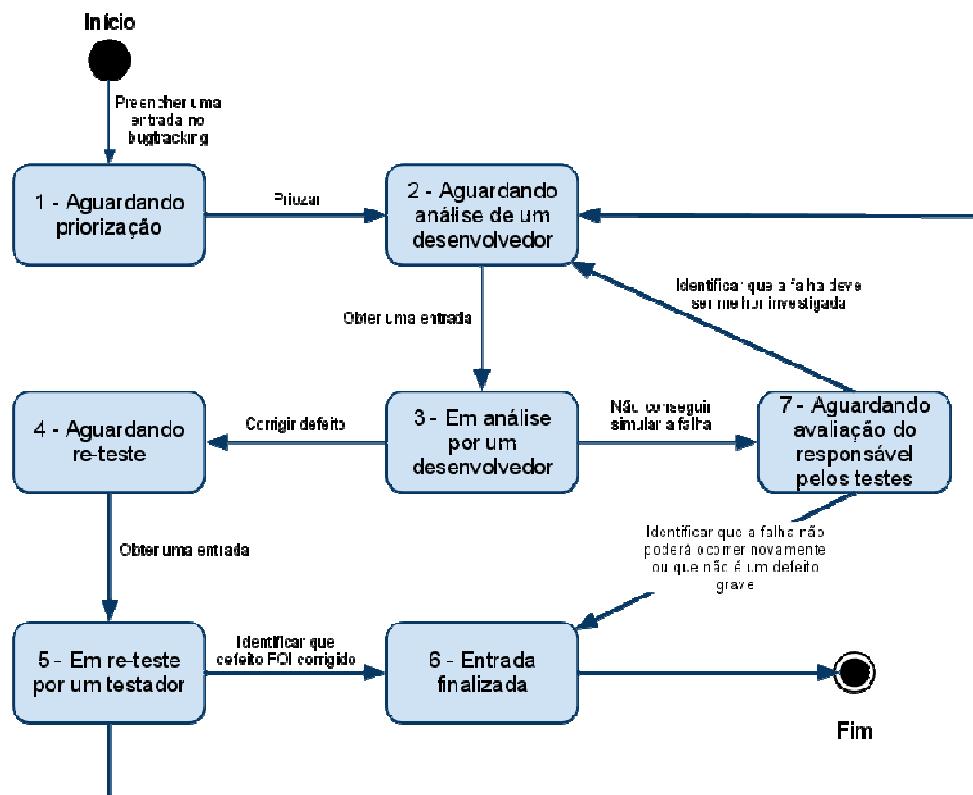
1. Receber descrição do erro corrigido.
2. Definir o que precisa ser re-testado.
3. Alterar o plano de testes existente, ou identificar quais passos de um plano de testes existente serão repetidos.
4. Executar os testes.
5. Caso os testes sejam bem-sucedidos, atualizar bugtracking, finalizando a entrada.
6. Caso os testes não sejam bem-sucedidos, atualizar a entrada no bugtracking e devolver para o desenvolvedor.

Cada entrada no bugtracking deve possuir as seguintes informações:

- Funcionalidade que apresentou falha
- Data do teste
- Falha identificada
- Descrição em passos detalhados para simulação da falha
- Prioridade para correção
- Situação da entrada

A **Figura 11.1** mostra os estados que uma entrada, reportando uma falha, pode assumir.

## Diagrama de estados das entradas do bugtracking



**Figura 11.1 - Estados das entradas do bugtracking**

### 11.3 Definições

- Testador. Papel desempenhado por aquele que identificou uma falha no jogo.
- Desenvolvedor. Papel desempenhado por aquele que implementou uma funcionalidade ou corrigiu um defeito.
- Responsável pelos testes. Papel desempenhado por uma pessoa, responsável por gerenciar o bugtracking e os documentos contendo as especificações de testes.
- Bugtracking. Lista de todas as falhas encontradas e a situação de cada uma dentro do processo de testes.
- Plano de testes. Sequência de passos definida pelos testadores para realizar o teste de alguma funcionalidade ou característica do jogo.

### 11.4 Fontes

- [http://www.lucattelli.com/downloads/Estrategia\\_de\\_Teste.pdf](http://www.lucattelli.com/downloads/Estrategia_de_Teste.pdf)
- [http://www.wthree.com/rup/process/workflow/test/co\\_testr.htm](http://www.wthree.com/rup/process/workflow/test/co_testr.htm)
- <http://web.niaccist.niacc.edu/~milleste/classroom/testingconcepts/section8/index.html>
- <http://www.pontov.com.br/site/index.php/arquitetura/60-testes-em-jogos/200-tecnicas-de-teste-em-jogos-digitais>

## 12 Produção

Neste capítulos serão abordados alguns itens importantes para o gerenciamento do projeto, principalmente úteis na fase de produção do jogo.

### 12.1 Milestones do Projeto

A tabela abaixo exibe os Milestones do projeto durante a produção do jogo:

Id	Data	Descrição
#1	03/03/2011	Entrega e apresentação da proposta do jogo
#2	24/03/2011	Entrega e apresentação do GDD
#3	26/04/2011	Entrega e apresentação do jogo versão 1 (documentação e executável)
#4	26/05/2011	Entrega e apresentação do jogo versão 2 (documentação e executável)
#5	28/06/2011	Entrega e apresentação do jogo versão final (documentação e executável)

*Tabela 12.1 - Milestones do projeto*

### 12.2 Listas de Tarefas e Cronograma

A tabela abaixo exibe um cronograma preliminar para o desenvolvimento do jogo. Durante as próximas semanas, uma análise mais intensa será realizada para avaliar se as atividades e datas propostas estão coerentes com o o que é esperado no desenvolvimento do jogo.

Id Tarefa	26/abr 26/mai 28/jun
1 Planta do jogo completa (3 fases) (paredes/portas/janelas) (textura mock) 2 Sala da Mansão - Disposição de todos os itens (mock) na sala 3 Marshall - Implem. do comportamento (atirar, andar, reload, trocar de munição, pegar itens). 4 Sala da Mansão - Implementação do quadro/cofre 5 Corridor - Disposição dos itens e textura (mock) 6 Main Hall - Disposição dos itens (mock) 7 Library - Disposição dos itens (mock) 8 Screamer - Implementação do comportamento/IA 9 Butcher - Implementação do comportamento/IA 10 Screamer - Modelagem 11 Marshall - Modelagem 12 Implementação do HUD 13 Implementação do inventário 14 Implementação de controles que não estejam no padrão da UDK (inventário, por ex)	
15 Sala da Mansão - Implementação do quadro de energia 16 Sala da Mansão - Implementação das regras (fluxo das atividades) 17 Sala da Mansão - Modelagem dos itens da sala 18 Shorties - Modelagem 19 Butcher - Modelagem 20 Shorties - Implementação do comportamento/IA 21 Diabolus - Modelagem 22 Diabolus - Implementação do comportamento/IA 23 Planta do jogo - modelagem dos elementos comuns (fase2/fase3) de cenário 24 Implementação das regras básicas do jogo (fluxo principal do jogo, condição de vitória, etc) 25 Implementar diferenças entre modo normal e difícil 26 Implementação de menu	
27 Corridor - Modelagem dos itens 28 Main Hall - Modelagem dos itens 29 Library - Modelagem dos itens 30 Integração de sons 31 Integração de músicas 32 Integração de cutscenes 33 Implementação das estatísticas de jogo / ranking 34 Crusher - Implementação do comportamento/IA 35 Crusher - Modelagem	
36 Atualização do GDD 37 Testes - Definição dos casos de teste	

**Figura 12.1 - Cronograma preliminar da fase de desenvolvimento do jogo**

As premissas a seguir foram consideradas para elaborar o cronograma:

- No milestone #3, serão entregues
  - Planta completa das fases, porém sem aplicação de texturas ou com texturas simples, e objetos usando modelos prontos nos cenários;
  - Puzzle do quadro (pintura) com dispositivo de segurança da fase 1;
  - Modelagem do personagem principal e de um inimigo;
  - Programação da IA/comportamento de dois inimigos na fase 2;
  - Os itens que não possuirem modelagem serão representados através de modelos previamente fornecidos pela engine ou buscados de outras fontes, como a internet.
- No milestone #4, será entregue:
  - Modelagem e IA/comportamento do chefe final;
  - Implementação das regras básicas do jogo (fluxo, condição de vitória, etc);
  - Entrega da fase 1 na versão final;
  - Jogo completo em funcionalidades, porém pouco refinado em acabamento.
- No milestone #5 será entregue:
  - Implementação e modelagem do Crusher;

- Modelagem de itens de decoração;
- Integração de músicas, sons e cutscenes;
- Implementação do ranking;
- Jogo completo em sua versão final.

## 13 Histórico de modificação do documento

A tabela abaixo apresenta as modificações efetuadas no documento desde sua criação:

Versão	Descrição	Data
1.0	Primeira versão do documento. Definição da estrutura inicial do documento. Transposição das informações contidas no documento inicial de escopo do jogo.	08/Mar/2011
1.1	Versão inicial dos Sistemas de Combate e de Dano, Fases, Referências.	09/Mar/2011
1.2	Atualização da seção Referências, nos tópicos Diablo, Hotéis e Mansões, Armas para Personagens.	10/Mar/2011
1.3	Atualização da seção de Jogabilidade.	13/Mar/2011
1.4	Adicionando descrição de inimigos.	14/Mar/2011
1.5	Adicionando descrição dos cenários da fase 2.	16/Mar/2011
1.6	Adicionando versão inicial da descrição do personagem principal, adicionando a arte final do personagem principal e as referências utilizadas na sua concepção. Incluído os tipos de som envolvendo porta na seção de efeitos sonoros.	16/Mar/2011
1.7	Adicionado a descrição da fase 1.	16/Mar/2011
1.8	Alterando planta baixa da fase 2.	17/Mar/2011
1.9	Adicionando texto sobre Horas de Jogo, Condições de Vitória e Salvar e Carregar. Revisando nível de dificuldade, adicionando a diferença para o puzzle. Adicionando texto sobre Estatísticas de Jogo / Ranking.	19/Mar/2011
2.0	Revisão e atualização dos sistemas de combate e dano; revisão das fases 1 e 2; definição de pontos de vida e dano para arma do personagem principal, inimigos e itens do puzzle; Alteração da formatação do documento	20/Mar/2011
2.1	Removendo capítulo de Referências. Agora, as referências estarão espalhadas pelos demais capítulos do documento. Adicionado seções de Controles e Filosofia.	20/Mar/2011
2.2	Adicionando o capítulo referente ao armamento do personagem principal. Adicionado o capítulo referente à IA dos inimigos	20/Mar/2011
2.3	Adicionando texto e diagrama sobre a progressão do jogo e texto sobre os menus. Para a progressão/menu ser escrita, montei uma lista de cenas que o jogo teria. Em algumas coloquei uma descrição breve.	20/Mar/2011

2.4	Adicionando descrição da personagem Samara	21/Mar/2011
2.5	Adicionando capítulo sobre testes.	21/Mar/2011
2.6	Adicionado Arte do Chefe final, arte e descrição do HUD, descrição de como se dá a morte do personagem, requisitos para executar o jogo, tecnologias utilizadas e motivação no uso da engine UDK.	21/Mar/2011
2.7	Adicionando descrição do inventário	21/Mar/2011
2.8	Adicionado Planta Baixa Fase 3 - Library, Concept Art fase 3, Concept Art Silmara, Concept Art Screamer, adicionando referências para Shortie	22/Mar/2011
2.9	Adicionando seção Motivação	22/Mar/2011
3.0	Reescritas as seções de enredo e cenas do jogo.	23/Mar/2011
3.1	Adicionando Modelos Artísticos do Inimigos	23/Mar/2011
3.2	Revisão e formatação final para entrega	23/Mar/2011
3.3	Criação da documentação técnica (Anexo A), contendo a documentação da versão 1 do jogo Late Redemption.	23/Abr/2011
3.4	Adição da seção sobre criação de cenários no Anexo A.	25/Abr/2011
3.5	Atualização da seção sobre IA.	25/Abr/2011

*Tabela 13.1 – Histórico de modificações do documento*

# ANEXO A – DOCUMENTAÇÃO DO PROJETO

Esta é a documentação técnica do jogo Late Redemption, desenvolvido na disciplina IA369-A. A disciplina compõe parte da integralização do programa de pós-graduação da Faculdade de Engenharia Elétrica e de Computação – FEEC – UNICAMP.

## 1 INTRODUÇÃO

O jogo Late Redemption foi desenvolvido durante o primeiro semestre de 2011 utilizando a plataforma UDK - Unreal Development Kit, Version: 7876 Change List: 776445. Todo o desenvolvimento foi baseado no *Game Design Document* (GDD) desenvolvido anteriormente.

Neste documento é descrito detalhes técnicos necessários para a implementação do jogo na *engine* escolhida. Caso seja necessário realizar a re-implementação do jogo, este documento servirá como guia.

Desta forma, este documento complementa o GDD, detalhando as construções e decisões técnicas tomadas pela equipe de desenvolvimento durante a implementação do jogo.

## 2 CONTROLE DE VERSÕES

Para o controle de versões, foi utilizado o Google Code (<http://code.google.com>). Para controlar o acesso aos arquivos versionados foi utilizado o programa TortoiseSVN (<http://tortois svn.tigris.org/>). Assim, foi criada uma estrutura para o trabalho a distância dos integrantes do grupo. A desvantagem desta abordagem é que o Google Code não permite o travamento de forma exclusiva de documentos. Portanto, foi definido que cada vez que alguém do grupo desejasse alterar algo, deveria enviar um e-mail para o grupo avisando.

## 3 PACOTES

Os elementos que compõem um jogo são organizados pela engine em pacotes. Para agilizar o salvamento das informações no repositório de código, os elementos do projeto foi dividido em diversos pacotes, conforme listados abaixo:

- **LateRedemptionMarshall** – Contém os itens relacionados com o personagem principal.
- **LateRedemptionDoorTextures** – Contém as texturas usadas nas portas da mansão.
- **LateRedemptionFloorTextures** – Contém as texturas usadas no chão da mansão.
- **LateRedemptionWallTextures** – Contém as texturas usadas nas paredes.
- **LateRedemptionMarshall** – Contém as texturas, materials e o mesh do Marshall.
- **LateRedemptionPackageSounds** – Contém todos os arquivos de som.
- **LateRedemptionTextures** – Contém o restante das texturas usadas no jogo.
- **LateRedemptionPackage** – Contém o restante dos elementos usados no jogo, incluindo, efeitos de explosão, matérias usados no quadro de distribuição e no quadro do cofre.

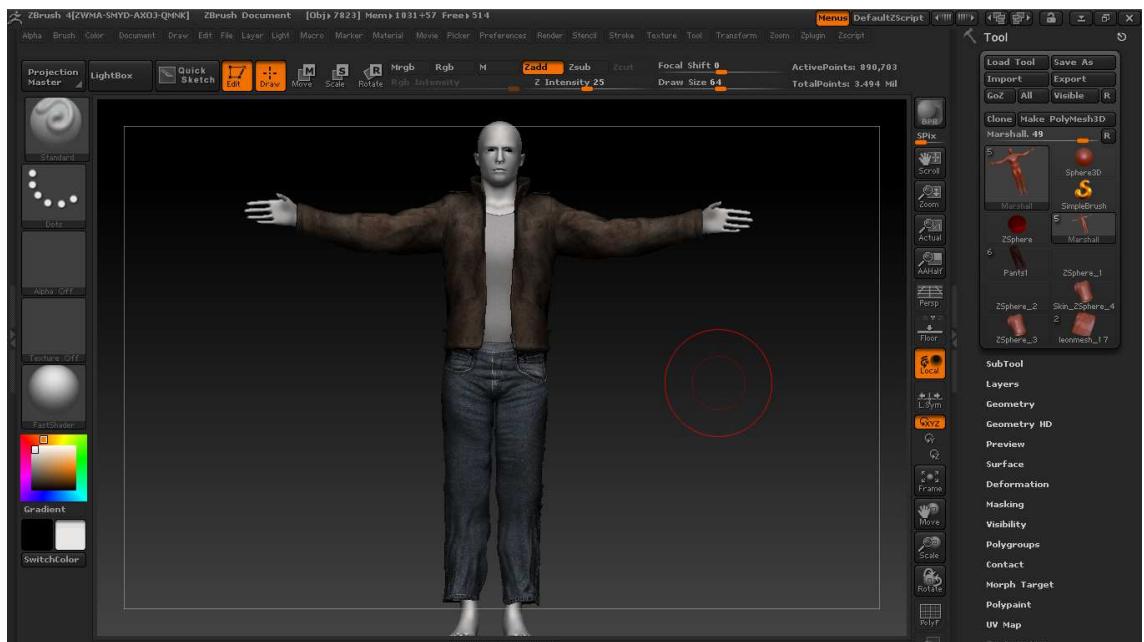
- **CH\_Zombie** – Contém os itens utilizados na implementação da primeira versão do Screamer e do Butcher.

## 4 ASPECTOS GERAIS

### 4.1 Modelagem e animação do Marshall

A modelagem do personagem Marshall Gory foi realizado utilizando o conjunto três programas ZBrush 4.0 (<http://www.pixologic.com/home.php>) , Maya 2011 (<http://usa.autodesk.com/maya/>) e o 3D Studio Max 9 (<http://usa.autodesk.com/3ds-max/>).

O processo se iniciou no ZBrush, onde este fornecia um modelo do corpo humano masculino para ser modelado. Através desse corpo foi-se criando os acessórios que iriam compor o personagem. Esses acessórios eram compostos por camiseta preta, jaqueta de couro marron, calça jeans, luvas pretas, cinto e cortuno preto. O ZBrush utiliza uma abordagem diferente para modelagem 3D. Enquanto programas como Maya e 3D Studio utilizando-se de vértices e arestas para construir um modelo, o ZBrush utiliza-se de massa de esculturismo onde deve-se modelar, através de pincéis, o objeto até a forma desejada. A figura 5.1 mostra o modelo do corpo humano fornecido pelo ZBrush já com alguns dos acessórios.

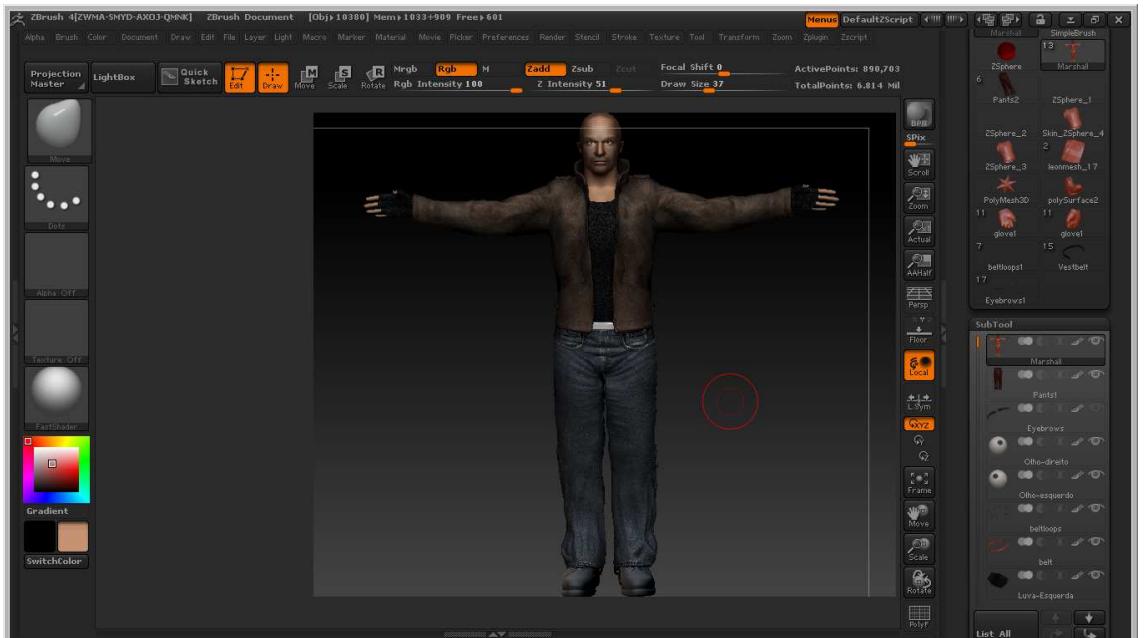


*Figura 4.1 – Modelo humano fornecido pelo ZBrush.*

Os acessórios calça jeans, jaqueta e corturno foram cobertos por texturas. Já as luvas e a camiseta foram pintados no próprio ZBrush. A pintura da pele humana foi feita através da técnica Spotlight ([http://www.pixologic.com/docs/index.php/Working\\_with\\_Spotlight](http://www.pixologic.com/docs/index.php/Working_with_Spotlight)) na qual através da foto do rosto de uma pessoa, o programa obtém todos os detalhes da imagem e transfere ao rosto do modelo 3D. A figura 5.2 mostra como foi o processo e as figuras 5.3 a 5.6 mostra o modelo finalizado.



**Figura 4.2 – Técnica Spotlight fornecida pelo ZBrush para pintar rostos.**



**Figura 4.3 – Modelo de perfil do Marshall Gory finalizado.**

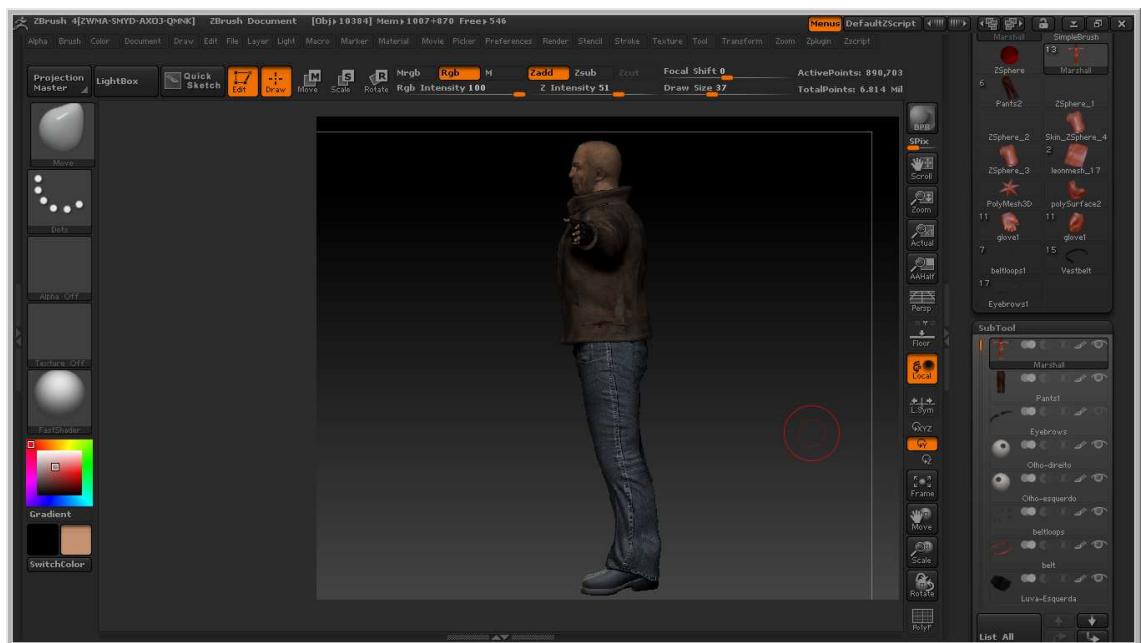


Figura 4.4 – Modelo de lado do Marshall Gory finalizado.

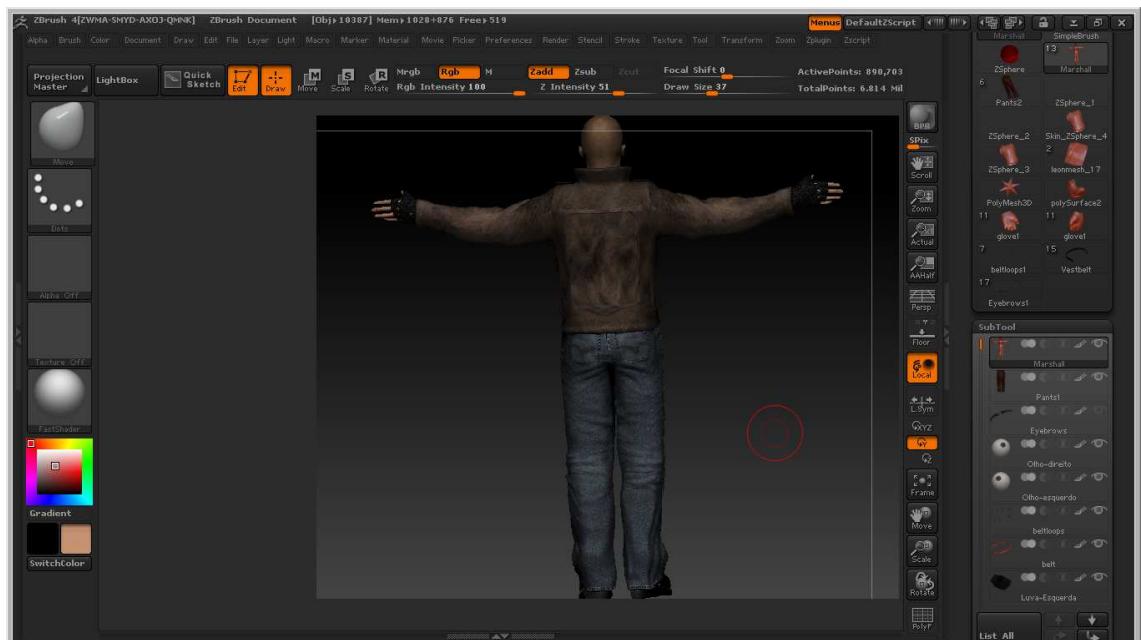
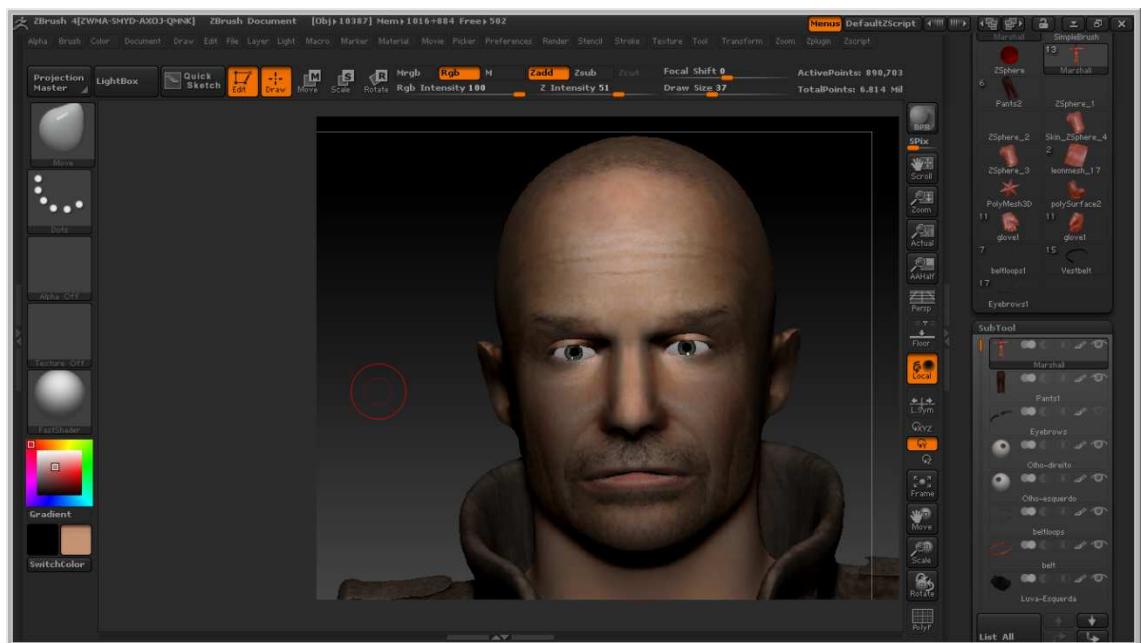


Figura 4.5 – Modelo de costas do Marshall Gory finalizado.



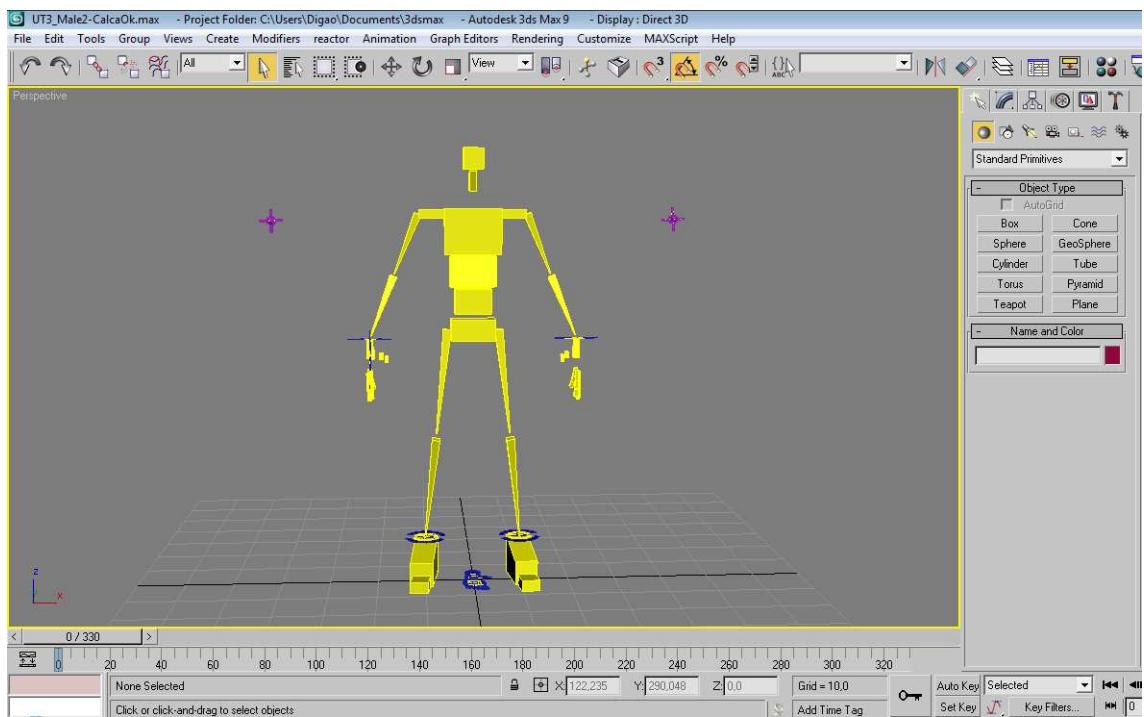
*Figura 4.6 – Modelo do rosto do Marshall Gory finalizado.*

No final, o modelo do Marshall Gory feito no ZBrush ficou com 890.703 polígonos, o que para um jogo é algo relativamente alto. Para realizar o processo de diminuição de polígonos e, ao mesmo tempo, não perder tanto a qualidade, foi utilizado o processo UV Layout, para salvar todas as texturas em alta qualidade e depois a diminuição dos polígonos utilizando o plugin Decimation Master (<http://www.pixologic.com/zbrush/features/decimation/>) chegando a 30.000 polígonos.

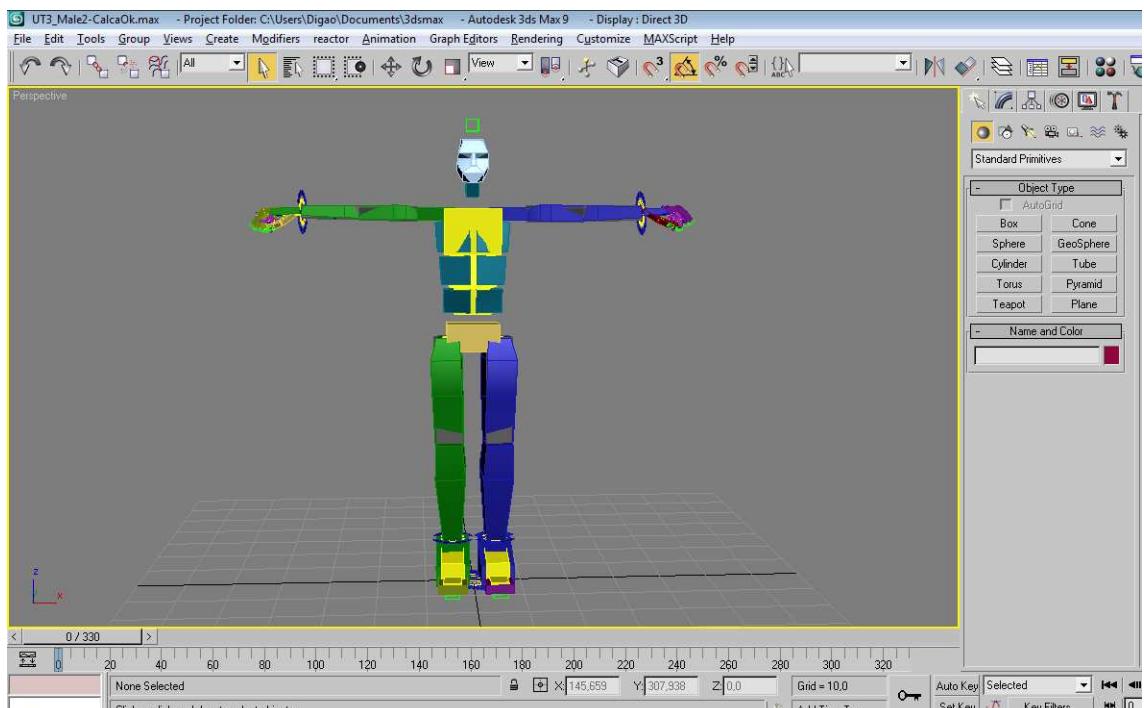
Com o término da modelagem, o modelo 3D foi exportado utilizando a extensão .OBJ para que o 3D Studio Max pode importar. Agora, iniciou-se o processo conhecido como Rigging que é a criação de um esqueleto para que as animações possam ocorrer.

No site do UDK (<http://udn.epicgames.com/Three/UDKCustomCharacters.html>) é fornecido um esqueleto pronto, com as nomeações corretas de cada osso para ser utilizado no 3D Studio Max. Utilizando esse esqueleto, o UDK fornece todas as animações de um personagem quer necessite correr, pular e atirar e esse é o caso do Marshall Gory. Dessa forma não precisaremos criar as animações e utilizaremos as do próprio UDK.

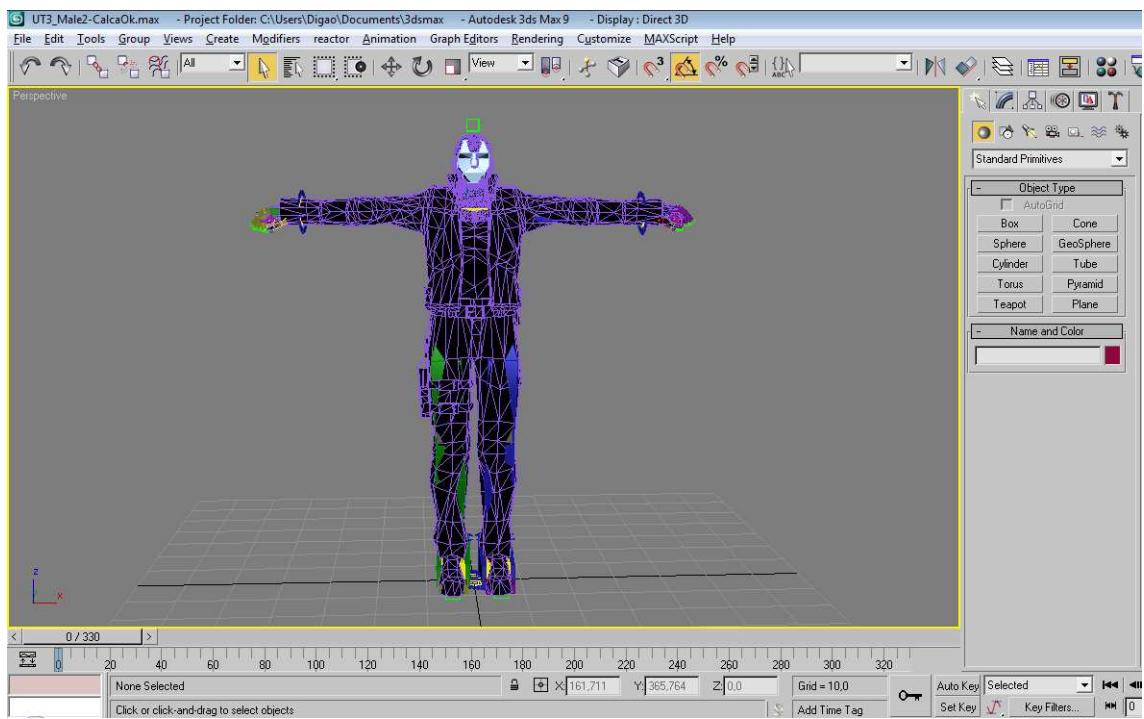
Porém, para isso devemos encaixar o modelo 3D do Marshall nesse esqueleto. As figuras 5.7 a 5.9 mostram o esqueleto e o modelo juntos.



**Figura 4.7 – Esqueleto fornecido pelo site do UDK para ser utilizado em jogo.**



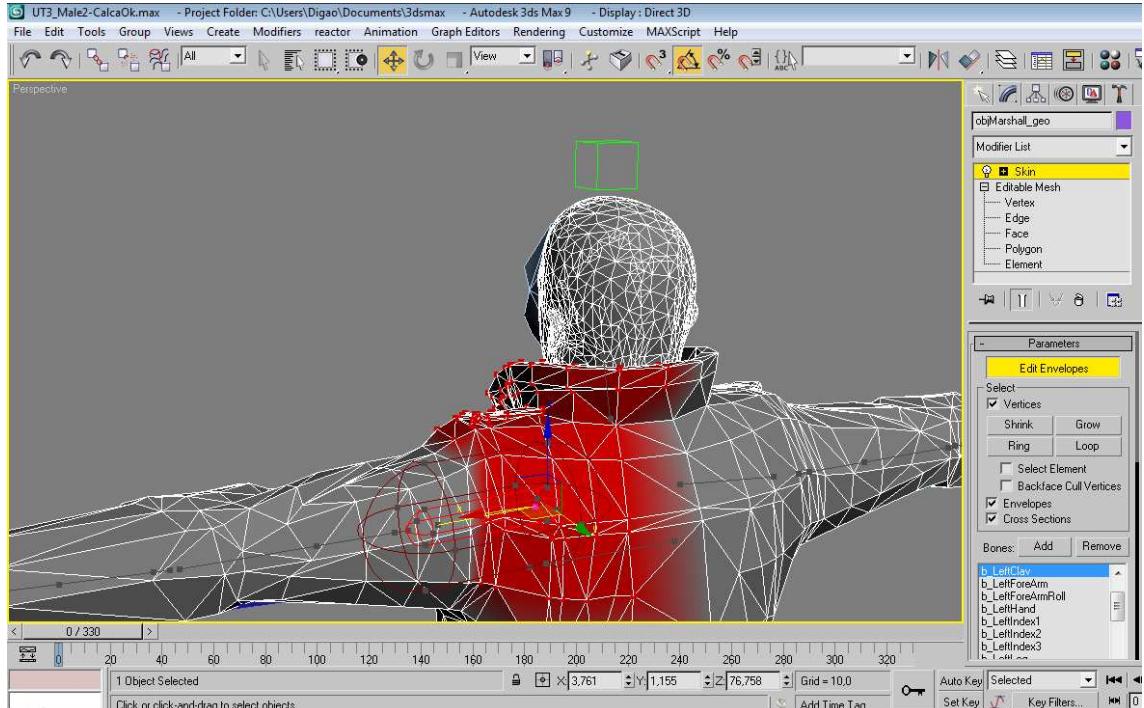
**Figura 4.8 – Esqueleto fornecido pelo site do UDK com um modelo de auxílio para ser utilizado em jogo.**



**Figura 4.9 – Esqueleto junto com modelo do Marshall para ser utilizado em jogo.**

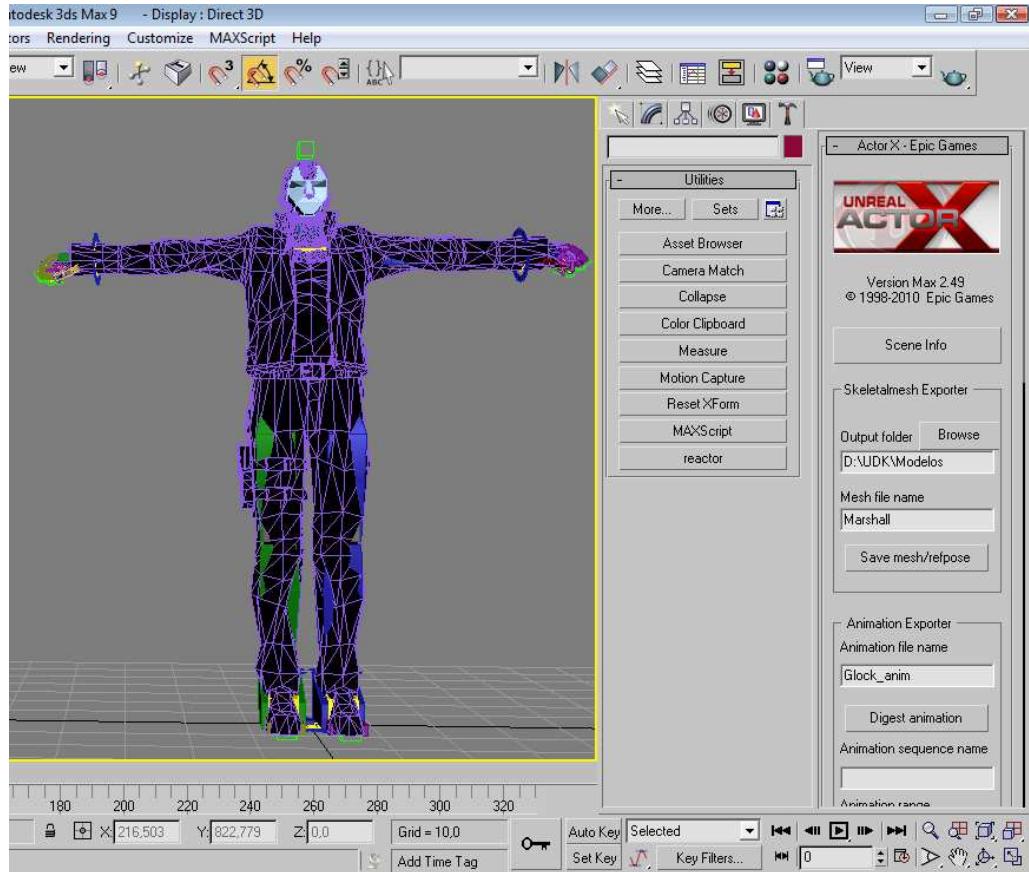
Após juntar o esqueleto ao modelo foi realizado o processo de Skinning que é o processo que define quais partes do modelo se movimentarão e como movimentarão quando os ossos se movimentarem.

Para isso, foi utilizada a ferramenta Weight Tool do próprio 3D Studio Max onde ao clicar em um osso, definem-se quais vértices fará parte do skinning do osso selecionado. A figura 5.10 mostra o processo de skinning.

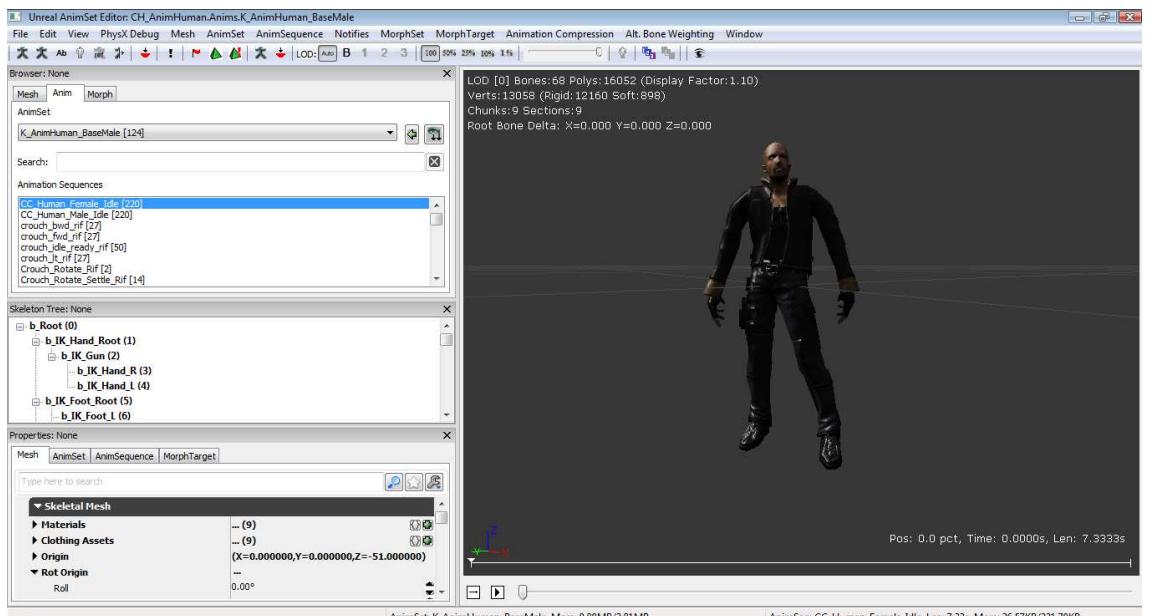


**Figura 4.10 – Processo de skinning sendo realizado na jaqueta.**

Quando processo de skinning foi finalizado, o modelo foi exportando para o UDK utilizando o plugin ActorX (<http://udn.epicgames.com/Three/ActorX.html>). Esse plugin faz a conversão do modelo 3D em um modelo 3D que o UDK interprete e aplique as animações. A figura 7.11 mostra a utilização do plugin e a figura 7.12 mostra o modelo exportado no UDK.



**Figura 4.11 – Plugin ActorX para exportar o modelo do Marshall para o UDK.**

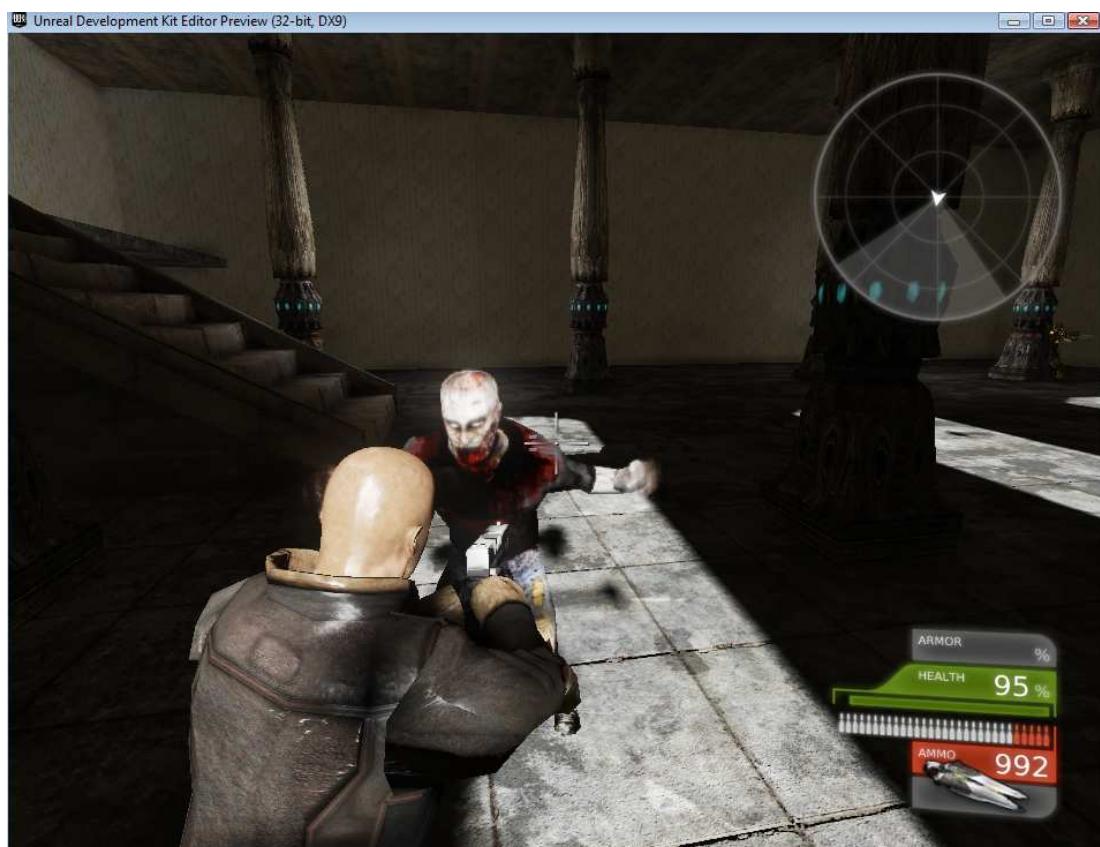


**Figura 4.12 – Modelo do Marshall no UDK com as animações.**

As figuras 5.13 a 5.15 mostram Marshall em jogo.



*Figura 4.13 – Marshall na versão inicial do Main Hall.*



*Figura 4.14 – Marshall na versão inicial do Main Hall lutando contra um inimigo de testes.*

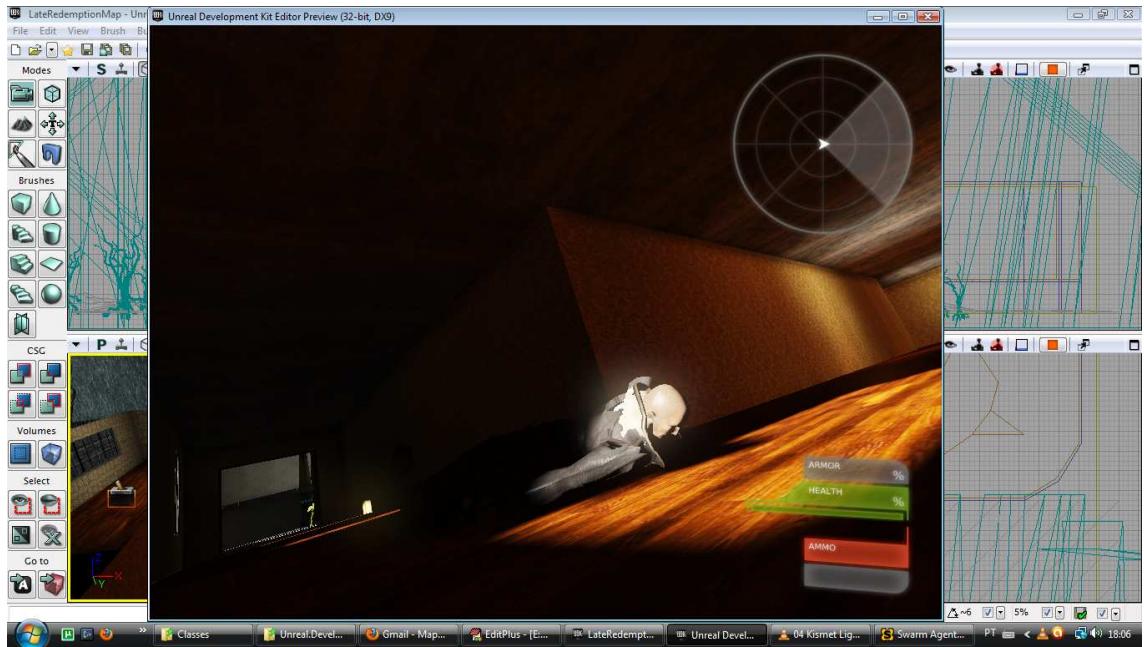


Figura 4.15 – Marshall morto na versão inicial do Corridor.

## 5 Inteligência Artificial

### 5.1 Programando no UDK

A inteligência artificial que dá vida aos NPCs (*Non Playable Characters*) dentro do UDK pode ser implementada utilizando-se dois pontos de partida:

Programação Orientada a Objetos : Este modelo utiliza a linguagem de programação própria do UDK chamada de *Unreal Script*, muito semelhante a C++ mas, com características próprias voltadas ao desenvolvimento de jogos como um suporte muito forte à implementação de máquinas de estado. Sendo uma linguagem orientada a objetos, o conceito de herança é amplamente utilizado a fim de promover reuso de código (e assim, várias características, comuns a todos os NPCs, estão automaticamente disponíveis para os novos NPCs a partir de herança).

Programação Gráfica: A segunda opção é utilizar o editor gráfico de programação do UDK chamado Kismet, uma ferramenta muito versátil (detalhes adicionais estão descritos no capítulo a seguir, dedicado exclusivamente ao Kismet).

Uma alternativa muito comum durante o desenvolvimento de projetos em UDK é a de utilizar o Unreal Script para moldar os principais elementos do comportamento dos NPCs (por exemplo, dentro das classes de Unreal Script estão descritas as ações dos NPCs em cada um dos seus estados bem como quais eventos podem ser ativados a partir destes estados) e, utilizar o Kismet para atividades adicionais (como aplicação de sons em determinados momentos). Um ótimo exemplo presente na primeira entrega diz respeito à implementação da IA do NPC *Butcher*: Suas principais ações estão descritas dentro da sua classe de controle enquanto que via Kismet, as ações pertinentes à sua arma (Ferir o Marshall, valor do dano causado ao Marshall) foram implementadas.

## 5.2 Preparando o Ambiente de Programação em Unreal Script.

Para iniciar a programação com o Unreal Script é necessário pré-configurar o sistema a fim de que o compilador seja capaz de enxergar o diretório onde as novas classes (já que o Unreal Script é uma linguagem orientada a objetos) estão presentes (além de obviamente, criar estes diretórios).

Portanto, para o jogo Late Redemption, as novas classes foram adicionadas em:

...\\UDK-2011-02\\Development\\Src\\LateRedemptionTest\\Classes

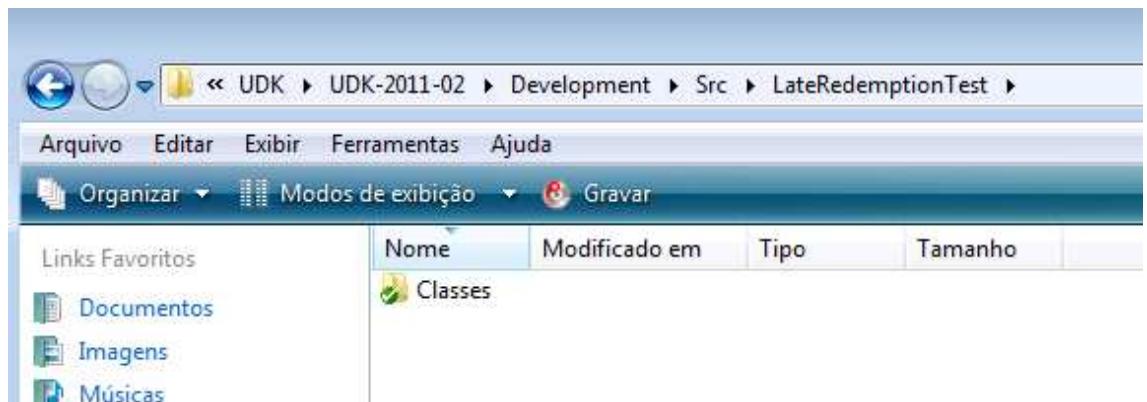


Figura 5.1 – Diretório para armazenamento dos Unreal Scripts.

E, para que o compilador do UDK seja capaz de encontrar estas classes, o arquivo

...\\UDK-2011-02\\UDKGame\\Config\\DefaultEngineUDK.ini

Também teve que ser modificado a fim de incluir as linhas marcadas abaixo:

```
[UnrealEd.EditorEngine]
EditPackagesOutPath=..\\..\\UDKGame\\Script
FRScriptOutputPath=..\\..\\UDKGame\\ScriptFinalRelease
+EditPackages=UDKBase
+EditPackages=UTEdition
+EditPackages=UTGame
+EditPackages=LateRedemptionTest
;ModEditPackages=MyMod
AutoSaveDir=..\\..\\UDKGame\\Autosaves
InEditorGameURLOptions=?quickstart=1?numplay=1

[Engine.Client]
MinDesiredFrameRate=35.000000
```

Figura 5.2 – Modificações necessárias no arquivo DefaultEngineUDK.ini.

A primeira linha foi necessária, pois algumas classes criadas para o jogo *LateRedemption* herdaram comportamentos e funções descritas no pacote UTGame. Já a

segunda linha foi necessária para que o programa seja capaz de encontrar as novas classes, específicas para o jogo LateRedemption. As únicas restrições aqui são:

- O nome do Edit Package apontando para os scripts específicos do jogo LateRedemption aponte para o diretório correto
- A ordem da disposição dos pacotes apresentada acima deve ser respeitada.

Após efetuadas estas modificações, ao adicionar (ou substituir) uma classe dentro do diretório contendo as classes do jogo LateRedemptionCom o Editor do UDK irá notar esta mudança e, antes de iniciar a Engine (ao clicar no ícone do UDK Editor), o UDK irá oferecer ao usuário a possibilidade de recompilar os scripts modificados e, caso o usuário aceite, as classes modificadas já estarão disponíveis para serem usadas na próxima vez que o editor for inicializado.

## 5.3 Criando a inteligência para um novo personagem:

Sempre que um novo NPC é criado utilizando-se o Unreal Script para moldar o seu comportamento, duas classes obrigatoriamente deverão ser manipuladas:

### A classe Pawn:

Esta classe contém as características físicas do personagem, e a sua função principal é apontar quais os arquivos contém o modelo do seu esqueleto, as suas animações, suas características físicas. Portanto, pontos de interesse desta classe são:

- A instrução **placeable** torna esta classe “adicionável” a partir do *content browser*.
- Os parênteses imediatamente seguintes à declaração de uma variável – ex: var () float – tornam esta variável editável a partir do editor do UDK (selecionando-se o objeto – por exemplo o personagem *Butcher* – e, apertando-se a tecla F4, surge uma listagem com todas as propriedades modificáveis dentro de todas as classes herdadas pelo *Butcher*)
- DefaultProperties – lista quais os valores iniciais para as propriedades desta classe – note que, estes valores podem sobrescrever aqueles previamente determinados em classes “pais” na hierarquia de herança. Aqui normalmente são listados valores iniciais para as principais características físicas do personagem como o seu esqueleto, sua aparência, suas animações.
- Notificações de eventos que interfiram em suas características físicas – Por exemplo: Como o personagem é afetado quando leva um dano; em que ponto de seu corpo este dano foi recebido; informação de que o personagem esteja caindo.

### A classe Controller:

É o cérebro deste novo personagem. Essencialmente contém a sua máquina de estados e as funções capazes de modificar o estado de um determinado personagem. Portanto, as principais características destas classes são:

- Descrever detalhadamente todas as ações a serem desenvolvidas em um determinado estado (a diretiva **auto** indica qual estado é automaticamente carregado quando a classe *Controller* toma posse – ativa – uma determinada classe *Pawn*).
- Funções que podem ser ativadas em qualquer estado. Estas funções estão definidas fora do escopo reservado para cada um dos estados e, quando chamadas, serão

executadas independente do estado em que o personagem encontre-se (ou seja, são funções globais).

- Funções que possuem comportamentos diferenciados em determinados estados: Dentro do escopo de um determinado estado, funções podem ser definidas e estas podem ser versões de funções globais ou novas funções particulares de um determinado estado.

Além destas características básicas, as funções descritas a seguir desempenham papel fundamental no protocolo de comunicação entre a classe *Pawn* e a sua classe *Controller*:

- *PostBeginPlay*: Existente tanto na classe *Controller* quanto na classe *Pawn*, descreve quais ações devem ser tomadas quando o objeto em questão (*Controller* ou *Pawn*) for criado. Normalmente é neste momento que a associação entre o *Pawn* e o seu *Controller* é feita (nas classes implementadas para o jogo LateRedemption, duas funções auxiliares *SetPawn* e *Possess*, complementam este processo de associação entre o personagem e o seu controlador).
- *Tick*: É uma interrupção periódica disponível nas classes *Pawn* (por analogia é como se fosse a respiração do *Pawn*). A cada ciclo, novas ações podem ser tomadas ou notificações para o *Controller* (cérebro) podem ser enviadas.

Para a entrega do primeiro demo, foram criadas classes *Controller* e *Pawn* tanto para o *Butcher* quanto para o *Screamer* e, as novas classes (bem como a hierarquia existente por detrás de cada uma destas classes) podem ser observadas nas figuras a seguir:

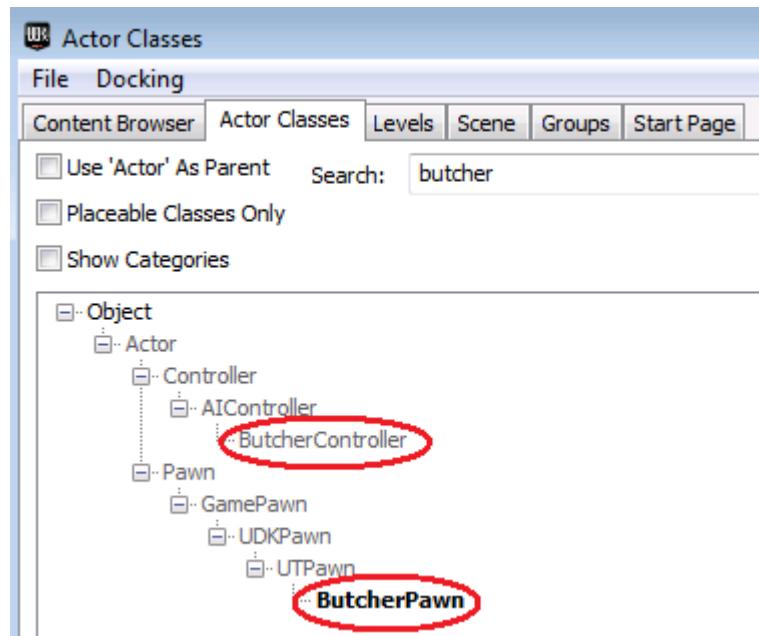
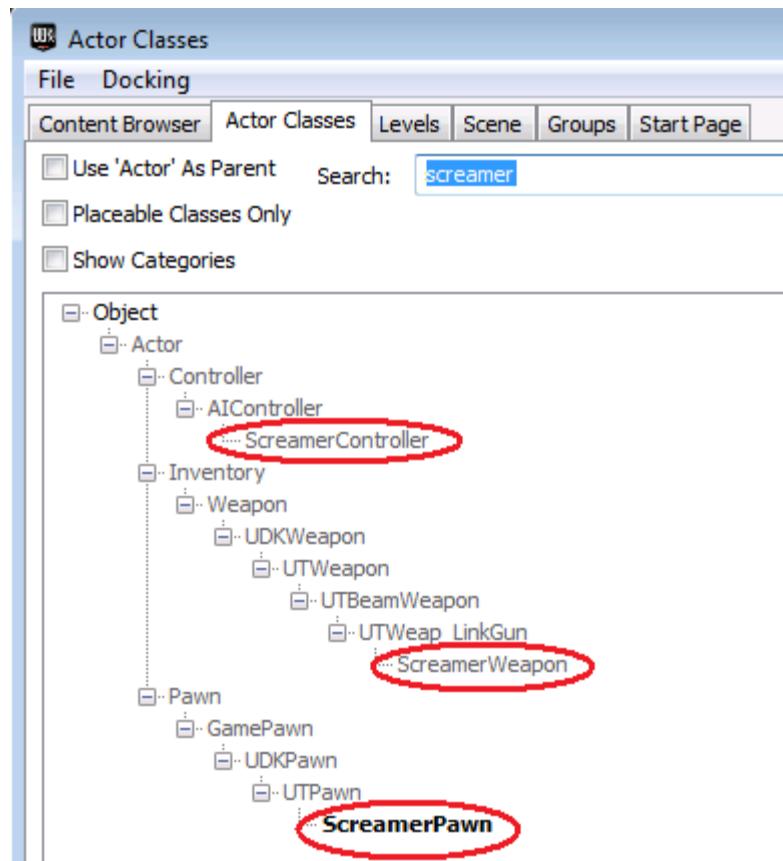


Figura 5.3 – Novas classes criadas para o *Butcher* e hierarquia adotada.



**Figura 5.4 – Novas classes criadas para o Screamer e hierarquia adotada.**

*Nota:* Por ser extremamente básica, a classe *ScreamWeapon* não será mais discutida neste documento. Sua função no primeiro demo do jogo foi apenas modificar a freqüência de disparos provenientes do *Screamer* (modificando o valor inicial estipulado pela arma *UTWeap\_LinkGun* que foi o ponto de partida para a arma utilizada pelo *Screamer* neste Demo).

As figuras anteriores foram obtidas a partir do Content Browser (é um mecanismo existente no UDK capaz de listar todos os componentes que podem ser adicionados aos jogos) e, as classes descritas em Negrito (*ScreamPawn* e *ButcherPawn*) indicam que estas classes podem ser carregadas diretamente no Editor do UDK (ou seja, elas podem ser arrastadas para o Editor, incluindo-se automaticamente um *Screamer* ou um *Butcher* ao jogo, graças à instrução **placeable** mencionada anteriormente).

## 5.4 Características básicas implementadas nas Classes (Primeiro Demo).

A preocupação inicial, no que diz respeito à IA dos inimigos para a entrega do primeiro demo foi a de desenvolver os principais conceitos que fazem parte do corpo das máquinas de estados. Dentre os conceitos desenvolvidos e, implementados nas classes implementadas, destacam-se:

- Trigger de ativação dos monstros a partir da proximidade com o Player: O inimigo entrará em ação quando avistar o player, sendo que a distância (fator de miopia...) que determina se a perseguição ao Player será iniciada ou não pode ser ajustado individualmente (ou seja, cada monstro pode possuir o seu valor próprio). Para isto, a

função *SeePlayer*, dentro da classe de controle foi utilizada e, ela basicamente seta o Player como alvo de perseguição.

- Trigger de ativação dos monstros a partir de um dano tomado: A função primordial desta opção é forçar o inimigo começar a perseguir o player assim que ele for atingido. Dentro da classe *Pawn*, existe a função *TakeDamage* (chamada automaticamente sempre que o monstro levar um dano a partir dos disparos do Marshall) e, em ambos os casos (*Screamer* e *Butcher*), esta função foi reescrita a fim de enviar uma notificação para a classe *Controller* (chamando a função *NotifyTakeHit*). Portanto, é dentro da função *NotifyTakeHit* que o Player será apontado como objeto alvo da perseguição dos inimigos.
- Mudança de atitude do monstro a partir de um dano tomado: Durante um determinado intervalo de tempo depois de levar um tiro, cuja notificação é feita a partir das funções *NotifyTakeHit* e *TakeDamage* monstro irá mudar o seu estado (na verdade, o que ocorre aqui é que o monstro estará em um sub-estado do estado de Ataque padrão, caracterizado por um aumento da agressividade, e, agressividade sendo um fator preponderantemente ligado à classe *Pawn*, as ações descritas a partir da classe *Controller* não serão modificadas – ou seja, o estado propriamente dito não será modificado).

O *Screamer* possui um estágio de fúria, enquanto que o *Butcher* possui dois estágios de fúria. Estes estágios são na verdade sub-estados do estado de ataque e, caracterizam-se pelo aumento da velocidade e, por consequência aumento das investidas dos monstros contra o Marshall.

- Revenge Timer: É um timer interno do jogo foi criado a partir da função *Tick* reescrita na classe *Pawn* que serve para controlar as transições dos sub-estados de ataque.
- Distinguir o Player dos outros bots dentro do jogo e persegui-lo: Os monstros podem ver uns aos outros mas não irão atacar-se mutuamente (eles podem até ser feridos involuntariamente por algum outro monstro tentando atingir o Marshall) nem devem iniciar a sua movimentação quando detectam a presença de um outro bot.

Usando como critério o tipo da classe de controle de um personagem, os bots são capazes de identificar o personagem sendo controlado pelo player e persegui-lo (Se um personagem estiver sendo controlado a partir de um *PlayerController* (ao invés de um *AIController*), ele não é um NPC).

Na primeira versão, a perseguição é linear ou seja, o monstro irá em intervalos periódicos de aproximadamente meio segundo, traçar uma reta entre o seu ponto atual e o ponto no mapa onde encontra-se o Player e tentar percorrer esta reta. Se o caminho até o Player estiver obstruído, nesta primeira versão, o monstro irá parar.

Para a entrega da Demo-2, será implementado um mecanismo auxiliar de navegação para que, nos casos descritos acima (caminho interrompido), o monstro seja capaz de encontrar um caminho alternativo e, assim, continuar a perseguição ao Marshall.

- Elaboração da primeira versão da máquina de estados dos bots e assim, entender como funciona o mecanismo e a transição entre os estados, como fechar o loop de execução das máquinas de estado. Resumidamente para esta primeira versão, os estados para os dois monstros serão:

- Idle: O monstro ainda não está ativo. Estado padrão. Futuramente, o Screamer irá iniciar uma patrulha próximo ao seu ponto de Origem depois de um intervalo de tempo no estado idle.
  - Pursuit: Ou o Player foi avistado dentro do range de visão do monstro ou o monstro foi atacado pelo Marshall para que este estado seja alcançado. Neste estado, os monstros começam a perseguir o Marshall até que eles cheguem na distância de ataque.  
Caso Marshall consiga despistar os monstros (fique a uma distância maior do que a distância de percepção do campo de visão dos monstros), eles voltarão para o estado Idle.
  - Attack: Marshall está dentro do campo de ataque do Monstro. O campo de ataque do Butcher é bem menor do que o campo de ataque do Screamer.  
Neste estado, o Butcher tentará atingir o Marshall com a sua arma enquanto que o Screamer irá disparar suas bolas de fogo.  
Caso o Marshal consiga escapar (saindo do range de ataque dos monstros), eles voltarão para o estado Pursuit.  
Caso Marshall consiga despistar os monstros (fique a uma distância maior do que a distância de percepção do campo de visão dos monstros), eles voltarão para o estado Idle.
  - Revenge e Insane: (este último válido apenas para o Butcher) são sub-estados de ataque, que caracterizam-se pelo aumento da velocidade de ataque enquanto o Revenge Timer estiver ativo.  
É importante salientar que o revenge timer será resetado sempre que um novo dano for recebido pelos monstros.  
Quando o revenge timer vence (ele é configurável), o monstro retornará ao seu sub-estado de ataque com menor grau de agressividade.
  - Death: O Player está morto (estado presente nas classes Parent).
- Dano causado por arma física: Aqui foi usado o Kismet para anexar uma arma ao *Butcher*. O código do Kismet assegura que, assim que o *Butcher* morrer, a arma deixe de ser letal. Mais detalhes, no capítulo a seguir.
- Dano a partir de projéteis: É o dano sendo causado pelo *Screamer* que na primeira versão Demo está usando como mecanismo de ataque uma das armas pré-existentes no UDK, sendo que pequenas modificações no que diz respeito à taxa de tiros foram feitas.
- Dano causado por contato: Se durante os testes o Marshall tocar tanto no *Screamer* quanto no *Butcher*, sua energia lentamente será "drenada". Quando causado por estes dois monstros, este dano é praticamente desprezível para o Marshall mas, esta função será empregada futuramente na classe do Shortie para que ele torne-se letal.

Portanto, com as classes desenvolvidas para a entrega da primeira versão Demo, dois dos inimigos de Marshall (*Screamer* e *Butcher*) ganharam vida e, as classes responsáveis por esta implementação estão listadas abaixo:

## 5.5 Classes Desenvolvidas para a entrega da primeira Demo

Como descrito anteriormente, sempre que um novo personagem (inimigo) é adicionado ao jogo, duas classes obrigatoriamente precisam ser reescritas. Aqui, estão listadas as classes relacionadas com os personagem Butcher e Screamer desenvolvidos para a primeira entrega. A finalidade de cada uma das funções bem como os detalhes a respeito dos estados implementados em cada um dos bots está contida na própria classe na forma de comentários.

### 5.5.1 Butcher Controller

```
class ButcherController extends AIController;

//=====
//-----Variables-----
//=====

var ButcherPawn myButcher1Pawn;           // Butcher controlled by this IA.
var Pawn thePlayer;                      // Player - must die...

var int actual_node;                     // Used during navigation
var int last_node;                       // Used during navigation

var Name AnimSetName;

var bool followingPath;                  // Movement is ongoing
var float distanceToPlayer;             // No comments...
var Float IdleInterval;
const ADDSPEEDONHIT=350;                 // Increase in speed at hit.

//-----
// Variables you get directly from the ButcherPawn Class
//-----

var array<NavigationPoint> navigationPointsButcher;
var float perceptionDistance;           // Myopia factor :)
var float attackDistance;               // Anything within this radio dies

//=====
//-----Just in case they are not initialized,-----
//-----let's give them a default value-----
//=====

defaultproperties
{
    actual_node = 0
    last_node = 0
    perceptionDistance = 10000
    attackDistance = 50
    AnimSetName = "ATTACK"
    followingPath = true
    IdleInterval = 2.5f
}

//-----
//-----Global Functions/Events-----
//-----They can be unleashed no matter the state we are-----
//=====

//-
// This function is only used for debug purpose. It simply unleashes
// internal messages in case the logactive flag is turned on.
//-
```

```

function LogMessage(String texto)
{
    if (myButcher1Pawn.logactive)
    {
        Worldinfo.Game.Broadcast(self, texto);
    }
}

//-----
// Function triggered by the ButcherPawn Class when a new butcher
// is created (it associates an Pawn object with its controller
// object. Some important variable values are sync at this moment.
//-----
function SetPawn(ButcherPawn NewPawn)
{
    LogMessage("Function ButcherController SetPawn");
    myButcher1Pawn = NewPawn;
    Possess(myButcher1Pawn, false);
    myButcher1Pawn.SetAttacking(false);
    navigationPointsButcher = myButcher1Pawn.navigationPointsButcher;
    perceptionDistance = myButcher1Pawn.perceptionDistance;
    attackDistance = myButcher1Pawn.attackDistance;
}

//-----
// This function is called when the Pawn being controlled by this
// controller gets alive (it is called from within above function
// which is triggered when the ButcherPawn is Spawned).
//-----
function Possess(Pawn aPawn, bool bVehicleTransition)
{
    LogMessage("Function ButcherController Possess");
    if (aPawn.bDeleteMe)
    {
        LogMessage("Function ButcherController Possess Fail");
        ScriptTrace();
        GotoState('Dead');
    }
    else
    {
        LogMessage("Function ButcherController Possess Success");
        Super.Possess(aPawn, bVehicleTransition);
        Pawn.SetMovementPhysics();
        if (Pawn.Physics == PHYS_Walking)
        {
            Pawn.SetPhysics(PHYS_Falling);
        }
    }
}

//-----
// Whenever the Pawn is hit, it will inform its controller by means
// of this function.
// The action being taken is to lock target on player, send a
// request towards the Pawn Class to increase its speed (revenge
// timer and attack sub-states are triggered here) and change
// the internal state value.
//-----
function NotifyTakeHit1()
{
    LogMessage("Event ButcherController NotifyTakeHit");
    thePlayer = GetALocalPlayerController().Pawn;
    myButcher1Pawn.ChangeSpeed(ADDSPEEDONHIT); // Revenge Timer...
    distanceToPlayer = VSize(thePlayer.Location - Pawn.Location);
    GotoState('Pursuit'); // Go after Player
}

```

```

}

//-----
//-----State = IDLE-----
// In this state, the Butcher will wait for a while (IdleInterval)
// and in case Player is not visible, it will start a patrol.
// (To be implemented in phase-2).
// This is the default state - the one triggered after the
// Butcher is possessed.
//-----
auto state Idle
{
    event SeePlayer(Pawn seenPlayer)
    {
        LogMessage("Event ButcherController SeePlayer Idle");
        thePlayer = seenPlayer;
        if( PlayerController(thePlayer.Controller) != none )
        {
            distanceToPlayer = VSize(thePlayer.Location - Pawn.Location);
            if (distanceToPlayer < perceptionDistance)
            {
                GotoState('Pursuit');
            }
        }
    }

    Begin:
    LogMessage("State ButcherController Idle");
    Pawn.Acceleration = vect(0,0,0);
    Sleep(IdleInterval);
}

//-----
//-----State = Pursuit-----
// Either when the Player is detected or when the Butcher is hit
// by a player projectile, this state will be unleashed.
// The action sequence within this state ensures that wherever the
// player goes, the Butcher will be following him.
// - If the Butcher loose contact with Player, it will go back to
//   the idle State.
// - In case player is not reachable (e.g.: protected by a wall),
//   the Butcher will try to reach him by going through some of its
//   known anchor places (to be implemented in phase-2, as this
//   is part of the patrol mechanism)).
// - If the player is close enough (i.e.: it is within the Attack
//   distance value), the state will be changed to Attack.
//-----
state Pursuit
{
    Begin:
    LogMessage("State ButcherController Pursuit");
    Pawn.Acceleration = vect(0,0,1);

    while (Pawn != none && thePlayer.Health > 0)
    {
        if (ActorReachable(thePlayer))
        {
            distanceToPlayer = VSize(thePlayer.Location - Pawn.Location);
            if (distanceToPlayer < attackDistance)
            {
                GotoState('Attack');
                break;
            }
        }
    }
}

```

```

        {
            MoveToward(thePlayer, thePlayer, 20.0f);
            if(Pawn.ReachedDestination(thePlayer))
            {
                GotoState('Attack');
                break;
            }
        }
    else
    {
        MoveTarget = FindPathToward(thePlayer);
        if (MoveTarget != none)
        {
            LogMessage("Moving Towards Player");
            distanceToPlayer = VSize(MoveTarget.Location - Pawn.Location);
            if (distanceToPlayer < 100)
                MoveToward(MoveTarget, thePlayer, 20.0f);
            else
                MoveToward(MoveTarget, MoveTarget, 20.0f);
        }
        else
        {
            GotoState('Idle');
            break;
        }
    }
    Sleep(1);
}
GotoState('Idle');
}

//-----
//-----State = Attack
// If we reached this state (by the way, global state Attack is kept
// when revenge timer is active as well - i.e.: states Revenge and
// insane are in fact Attack Sub-states) it means that poor Player
// is within the Butcher's cleaver range attack.
// This state is mortal, especially if player is trapped into
// a corner hehehe...
// If the player is able to escape from Butcher attacks (either
// by hiding himself or by fleeing), Butcher state will be moved
// back to Pursuit.
//-----
state Attack
{
    Begin:
    LogMessage("State ButcherController Attack");
    myButcher1Pawn.SetAttacking(true);
    Pawn.Acceleration = vect(0,0,0);

    while(true && thePlayer.Health > 0)
    {
        if (!ActorReachable(thePlayer))
        {
            myButcher1Pawn.SetAttacking(false);
            myButcher1Pawn.StopFire(0);
            GotoState('Pursuit');
            break;
        }

        distanceToPlayer = VSize(thePlayer.Location - Pawn.Location);
        if (distanceToPlayer > attackDistance * 2)
        {
            myButcher1Pawn.SetAttacking(false);
        }
    }
}

```

```

        GotoState('Pursuit');
        break;
    }
    Sleep(1);
}
myButcher1Pawn.SetAttacking(false);
GotoState('Idle');
}

```

### 5.5.2 Butcher Pawn

```

class ButcherPawn extends UTPawn
placeable;                                // Available in Content Browser

//=====
//-----Variables-----
//=====

var SkeletalMesh defaultMesh;                // Custom Mesh member
var AnimTree defaultAnimTree;                // Custom Anim member
var array<AnimSet> defaultAnimSet;          // Custom Anim member
var PhysicsAsset defaultPhysicsAsset;         // Custom Mesh member
var MaterialInterface defaultMaterial0;      // Custom Mesh member
var Name AnimSetName;

const MAXGROUNDSPEED=800;
const MINGROUNDSPEED=100;
const MAXTICKERCOUNTER=20;
const REDUCESPEEDONTIMEOUT= -350;

var ButcherController myController;           // Butcher IA Controller
var float tickCounter;                      // Timing variable
var int reduceSpeedTimer;                   // Timing variable
var bool AttAcking;                        // Flag to indicate state
var () bool logactive;                      // Turn debug on-off

//-----
// Variables to be used by the Controller Class
//-----

var () float perceptionDistance;           // Myopia factor :)
var () float attackDistance;                // Death distance.
var () array<NavigationPoint> navigationPointsButcher;

//=====
//-----Just in case they are not initialized,-----
//-----let's give them a default value-----
//=====

defaultproperties
{
    GroundSpeed = MINGROUNDSPEED
    AnimSetName="ATTACK"
    AttAcking=false
    logactive = false;
    perceptionDistance = 10000
    attackDistance = 50

    defaultMesh=SkeletalMesh'CH_Zombie.Mesh.SK_Zombie'
    defaultAnimTree=AnimTree'CH_Zombie.Anims.Zombie_AnimTree'
    defaultAnimSet(0)=AnimSet'CH_Zombie.Anims.Zombie_AnimSet'
    defaultPhysicsAsset=PhysicsAsset'CH_Zombie.Mesh.SK_Zombie_Physics'

    Begin Object Name=WPawnSkeletalMeshComponent
        SkeletalMesh=SkeletalMesh'CH_Zombie.Mesh.SK_Zombie'
        AnimSets(0)=AnimSet'CH_Zombie.Anims.Zombie_AnimSet'
        AnimTreeTemplate=AnimTree'CH_Zombie.Anims.Zombie_AnimTree'
}

```

```

bOwnerNoSee=false
CastShadow=true
BlockRigidBody=true
BlockActors=true
BlockZeroExtent=true
BlockNonZeroExtent=true
bAllowApproximateOcclusion=true
bForceDirectLightMap=true
bUsePrecomputedShadows=false
LightEnvironment=MyLightEnvironment
End Object
mesh = WPawnSkeletalMeshComponent

Begin Object Name=CollisionCylinder
CollisionRadius=+0041.000000
CollisionHeight=+0044.000000
BlockZeroExtent=false
End Object
CylinderComponent=CollisionCylinder
CollisionComponent=CollisionCylinder

bCollideActors=true
bPushesRigidBodies=true
bStatic=False
bMovable=True
bAvoidLedges=true
bStopAtLedges=true
LedgeCheckThreshold=0.5f
}

//=====
//----- Functions/Events -----
//=====

//-----
// The main purpose of this function is to establish the link
// between this Pawn and its controller Class.
//-----
simulated function PostBeginPlay()
{
    super.PostBeginPlay();
    SetPhysics(PHYS_Walking);
    if (myController == none)
    {
        myController = Spawn(class'ButcherController', self);
        myController.SetPawn(self);
    }
}

//-----
// This function is only used for debug purpose. It simply unleashes
// internal messages in case the logactive flag is turned on.
//-----
function LogMessage(String texto)
{
    if (logactive)
    {
        Worldinfo.Game.Broadcast(self, texto);
    }
}

```

```

//-----
// Whenever the Attack State is changed within the Controller class,
// (i.e.: either entering or leaving the Attack state) this function
// is called so that the Attacking variable can be properly updated.
// This variable can be useful to trigger future Kismet Sequences.
//-----

function SetAttacking(bool atacar)
{
    Attacking = atacar;
}

//-----
// This function is linked to the revenge timer and with the Attack
// substate Revenge.
// It will be called by the Controller class whenever it receives
// the notification that the Butcher has been hit requesting to
// increase its speed velocity.
// Before changing the speed, the Butcher must check whether the
// new requested value is within acceptable limits.
// On the opposite direction, when revenge timer runs out, this
// function is internally called in order to decrease the Butcher
// speed, changing in this way its Attack sub-state
// It means that in case the ChangeSpeed succeeds in changing the
// Butcher speed, it is going from Attack to Revenge or from Revenge
// to Insane states.
// If the ChangeSpeed succeeds in decreasing its speed, it changes
// its state on the opposite direction Insane -> Revenge or
// Revenge -> Attack.
//-----
function ChangeSpeed(int speed)
{
    reduceSpeedTimer = 0;                                // Reset revenge timer
    groundspeed = groundspeed + speed;
    if (groundspeed > MAXGROUNDSPEED)
    {
        groundspeed = MAXGROUNDSPEED;
    }
    if (groundspeed < MINGROUNDSPEED)
    {
        groundspeed = MINGROUNDSPEED;
    }
}

//-----
// This event is notified whenever the Butcher is hit. It has been
// overwritten here just to send a notification for the controller
// class by calling the NotifyTakeHit1 function.
//-----
event TakeDamage (int Damage, Controller EventInstigator, Object.Vector
HitLocation, Object.Vector Momentum, class<DamageType> DamageType, optional
Actor.TraceHitInfo HitInfo, optional Actor DamageCauser)
{
    LogMessage("Event Pawn TakeDamage");

super.TakeDamage(Damage,EventInstigator,HitLocation,Momentum,DamageType,HitInf
o,DamageCauser);
    myController.NotifyTakeHit1();
}

//-----

```

```

// This event is from utmost importance to every Pawn.
// At some pre-determined intervals (usually between 0.05 and 0.1
// seconds) this event will be triggered and for instance, the Pawn
// can use these notifications in order to take timing based actions.
// In this case, a function to drain the Player life is implemented
// in order to reduce the Player health whereas he is in contact
// with the Butcher.
// Here, revenge timer is also implemented, with the aid of
// reduceSpeedTimer variable so that when a predetermined timeout
// (equal to 100 Ticks) is reached, a request to reduce the
// Butcher speed will be carried out by calling the ChangeSpeed
// function with a negative value (in this case, if the Attack
// sub-state is Revenge, it will be set back to Normal Attack).
//-----
simulated event Tick(float DeltaTime)
{
    local UTPawn gv;
    super.Tick(DeltaTime);
    if (tickCounter < MAXTICKERCOUNTER)
    {
        tickCounter +=1;
    }
    else                                // Wait 10 primary timer intervals
    {
        reduceSpeedtimer = reduceSpeedtimer + 1;
        tickCounter = 0;
        foreach VisibleCollidingActors(class'UTPawn', gv, 100)
        {
            if(Attacking && gv != none)
            {
                if(gv.Health > 10)      // Contact will not kill player.
                {
                    gv.Health -= 1;
                    gv.IsInPain();
                }
            }
        }
    }

    if (reduceSpeedTimer == 10)      // Wait 100 primary timer intervals
    {
        reduceSpeedTimer = 0;
        ChangeSpeed(REDUCESPEEDONTIMEOUT);
    }
}

//-----
// This is the function responsible for changing the character when
// new version of the Mesh, AnimSet or Physical Asset are available.
//-----
simulated function SetCharacterClassFromInfo(class<UTFamilyInfo> Info)
{
    Mesh.SetSkeletalMesh(defaultMesh);
    Mesh.SetMaterial(0,defaultMaterial0);
    Mesh.SetPhysicsAsset(defaultPhysicsAsset);
    Mesh.AnimSets=defaultAnimSet;
    Mesh.SetAnimTreeTemplate(defaultAnimTree);
}

```

### 5.5.3 Screamer Controller

```
class ScreamerController extends AIController;
```

```

//=====
//-----Variables-----
//=====

var ScreamerPawn myScreamer1Pawn; // Screamer controlled by this IA.
var Pawn thePlayer; // Player - must die...

var int actual_node; // Used during navigation
var int last_node; // Used during navigation

var Name AnimSetName; // Just overwrite parent value.

var bool followingPath; // Movement is ongoing
var float distanceToPlayer; // No comments...
var Float IdleInterval;
const ADDSPEEDONHIT=350; // Increase in speed at hit.

//-----
// Variables you get directly from the ScreamerPawn Class
//-----

var array<NavigationPoint> navigationPointsScreamer;
var float perceptionDistance; // Myopia factor : )
var float attackDistance; // Anything within this radio dies

//=====
//-----Just in case they are not initialized,-----
//-----let's give them a default value-----
//=====

defaultproperties
{
    actual_node = 0
    last_node = 0
    perceptionDistance = 10000
    attackDistance = 300.0f
    AnimSetName ="ATTACK"
    followingPath = true
    IdleInterval = 10.0f
}

//=====
//-----Global Functions/Events-----
//-----They can be unleashed no matter the state we are-----
//=====

//----- This function is only used for debug purpose. It simply unleashes
// internal messages in case the logactive flag is turned on.
//-----
function LogMessage(String texto)
{
    if (myScreamer1Pawn.logactive)
    {
        Worldinfo.Game.Broadcast(self, texto);
    }
}

//-----
// Function triggered by the ScreamerPawn Class when a new Screamer
// is created (it associates an Pawn object with its controller
// object). Some important variable values are sync at this moment.
//-----
function SetPawn(ScreamerPawn NewPawn)

```

```

{
    LogMessage("Function ScreamerController SetPawn");
    myScreamer1Pawn = NewPawn;
    Possess(myScreamer1Pawn, false);
    myScreamer1Pawn.SetAttacking(false);
    navigationPointsScreamer = myScreamer1Pawn.navigationPointsScreamer;
    perceptionDistance = myScreamer1Pawn.perceptionDistance;
    attackDistance = myScreamer1Pawn.attackDistance;
}

//-----
// This function is called when the Pawn being controlled by this
// controller gets alive (it is called from within above function
// which is triggered when the ScreamerPawn is Spawed).
//-----
function Possess(Pawn aPawn, bool bVehicleTransition)
{
    LogMessage("Function ScreamerController Possess");
    if (aPawn.bDeleteMe)
    {
        LogMessage("Function ScreamerController Possess Fail");
        ScriptTrace();
        GotoState('Dead');
    }
    else
    {
        LogMessage("Function ScreamerController Possess Success");
        Super.Possess(aPawn, bVehicleTransition);
        Pawn.SetMovementPhysics();
    }
}

//-----
// Whenever the Pawn is hit, it will inform its controller by means
// of this function.
// The action being taken is to lock target on player, send a
// request towards the Pawn Class to increase its speed (revenge
// timer and attack sub-states are triggered here) and change
// the internal state value.
// If the player is too far from the Screamer, the Screamer will
// start a Patrol mechanism, searching for him (Patrol to be fully
// implemented in the Demo-2 version).
//-----
function NotifyTakeHit1()
{
    LogMessage("Event ScreamerController NotifyTakeHit");
    thePlayer = GetALocalPlayerController().Pawn;
    myScreamer1Pawn.ChangeSpeed(ADDSPEEDONHIT); // Revenge Timer...
    distanceToPlayer = VSize(thePlayer.Location - Pawn.Location);
    if (distanceToPlayer < perceptionDistance)
    {
        GotoState('Pursuit'); // Go after Player
    }
    else
    {
        GotoState('FollowPath'); // Search the Player
    }
}

//-----
//-----State = IDLE-----
// In this state, the Screamer will wait for a while (IdleInterval)
// and in case Player is not visible, it will start a patrol.
// (The Patrol mechanism will be fully implemented in demo-2).

```

```

// This is the default state - the one triggered after the
// Screammer is possessed.
//-----
auto state Idle
{
    event SeePlayer(Pawn seenPlayer)
    {
        LogMessage("Event ScreammerController SeePlayer Idle");
        thePlayer = seenPlayer;
        if( PlayerController(thePlayer.Controller) != none )
        {
            distanceToPlayer = VSize(thePlayer.Location - Pawn.Location);
            if (distanceToPlayer < perceptionDistance)
            {
                GotoState('Pursuit');
            }
        }
    }

    Begin:
        LogMessage("State ScreammerController Idle");

        Pawn.Acceleration = vect(0,0,0);
        Sleep(IdleInterval);

        actual_node = last_node;
        GotoState('FollowPath');                                // Start Patrol
    }

//-----
//-----State = Pursuit-----
// Either when the Player is detected or when the Screammer is hit
// by a player projectile and the distance between the player and the
// Screammer is within the perception distance limit, this state
// will be unleashed. The action sequence within this state ensures
// that wherever the player goes, the Screammer will be following him.
// - If the Screammer loose contact with Player, it will go back to
//   the idle State (and in sequence, will restart the Patrol
//   mechanism - to be implemented in phase-2).
// - In case player is not reachable (e.g.: protected by a wall),
//   the Screammer will try to reach him by going through some of its
//   known anchor places (to be implemented in phase-2, as this
//   is part of the patrol mechanism)).
// - If the player is close enough (i.e.: it is within the Attack
//   distance value), the state will be changed to Attack.
//-----
state Pursuit
{
    Begin:
        LogMessage("State ScreammerController Pursuit");
        Pawn.Acceleration = vect(0,0,1);

        while (Pawn != none && thePlayer.Health > 0)
        {
            if (ActorReachable(thePlayer))
            {
                distanceToPlayer = VSize(thePlayer.Location - Pawn.Location);
                if (distanceToPlayer < attackDistance)
                {
                    GotoState('Attack');
                    break;
                }
                else
                {
                    MoveToward(thePlayer, thePlayer, attackDistance);
                    if(Pawn.ReachedDestination(thePlayer))

```

```

        {
            GotoState('Attack');
            break;
        }
    }
else
{
    MoveTarget = FindPathToward(thePlayer, ,perceptionDistance);
    if (MoveTarget != none)
    {
        LogMessage("ScreamerController Moving Towards Player");
        distanceToPlayer = VSize(MoveTarget.Location - Pawn.Location);
        if (distanceToPlayer < 100)
            MoveToward(MoveTarget, thePlayer, 20.0f);
        else
            MoveToward(MoveTarget, MoveTarget, 20.0f);
    }
    else
    {
        GotoState('Idle');
        break;
    }
}
Sleep(1);
}
GotoState('Idle');
}

//-----
//-----State = Attack-----
// If we reached this state (by the way, global state Attack is kept
// when revenge timer is active as well - i.e.: state Revenge is
// in fact a Attack Sub-state) it means that Player is under
// Screamer's fireballs range. Hunting season has just begun...
// If the player is able to escape from Screamer's fire sight (either
// by hiding himself or by fleeing), Screamer state will be moved
// back to Pursuit.
//-----
state Attack
{
    Begin:
        LogMessage("State ScreamerController Attack");
        myScreamer1Pawn.ZeroMovementVariables();
        myScreamer1Pawn.SetAttacking(true);
        myScreamer1Pawn.StartFire(0);
        MoveToward(thePlayer, thePlayer, attackDistance);

        while(true && thePlayer.Health > 0)
        {
            if (!ActorReachable(thePlayer))
            {
                myScreamer1Pawn.SetAttacking(false);
                myScreamer1Pawn.StopFire(0);
                GotoState('Pursuit');
                break;
            }

            distanceToPlayer = VSize(thePlayer.Location - Pawn.Location);
            if (distanceToPlayer > attackDistance * 2)
            {
                myScreamer1Pawn.SetAttacking(false);
                myScreamer1Pawn.StopFire(0);
                GotoState('Pursuit');
                break;
            }
        }
}

```

```

        }
        Sleep(1);

    }
    myScreamer1Pawn.StopFire(0);
    myScreamer1Pawn.SetAttacking(false);
    GotoState('Idle');

}

//-----
//-----State = FollowPath-----
// When the Screamer is hit by a bullet but the Player is too far,
// rather than initiating an offensive against the player, it will
// start a patrol process trying to move towards reference points
// whereas searching for the player.
// This Patrol is also triggered by default after some time without
// action (timer defined by IdleInterval parameter) and as mentioned
// earlier, it will be fully implemented at Demo-2 game version.
// It is important to notice that in this state, the Screamer will
// keep searching for the Player and in case it is visible, it will
// start its hunting sequence by going to Pursuit state.
//-----
state FollowPath
{
    event SeePlayer(Pawn seenPlayer)
    {
        LogMessage("Event ScreamerController SeePlayer FollowPath");
        thePlayer = seenPlayer;
        distanceToPlayer = VSize(thePlayer.Location - Pawn.Location);
        if (distanceToPlayer < perceptionDistance)
        {
            followingPath = false;
            GotoState('Pursuit');
        }
    }
}

Begin:

LogMessage("State FollowPath");
followingPath = true;

while(followingPath)
{
    MoveTarget = navigationPointsScreamer[actual_node];

    if(Pawn.ReachedDestination(MoveTarget))
    {
        actual_node++;

        if (actual_node >= navigationPointsScreamer.Length)
        {
            actual_node = 0;
        }
        last_node = actual_node;

        MoveTarget = navigationPointsScreamer[actual_node];
    }
    if (ActorReachable(MoveTarget))
    {
        MoveToward(MoveTarget, MoveTarget);
    }
    else
    {
}

```

```

        MoveTarget
FindPathToward(navigationPointsScreamer[actual_node]);
        if (MoveTarget != none)
        {
            MoveToward(MoveTarget, MoveTarget);
        }
    Sleep(1);
}
}

```

#### 5.5.4 Screamer Pawn

```

class ScreamerPawn extends UTPawn
placeable;                                     // Available in Content Browser

//=====
//-----Variables-----
//=====

var SkeletalMesh defaultMesh;                  // Custom Mesh member
var AnimTree defaultAnimTree;                 // Custom Anim member
var array<AnimSet> defaultAnimSet;           // Custom Anim member
var PhysicsAsset defaultPhysicsAsset;         // Custom Mesh member
var MaterialInterface defaultMaterial0;       // Custom Mesh member
var Name AnimSetName;

const MAXGROUNDSPEED=800;
const MINGROUNDSPEED=300;
const MAXTICKERCOUNTER=20;
const REDUCESPEEDONTIMEOUT= -350;

var ScreamerController myController;           // Screamer IA Controller
var float tickCounter;                       // Timing variable
var int reduceSpeedTimer;                   // Timing variable
var bool Attacking;                         // Flag to indicate state
var () bool logactive;                      // Turn debug on-off

//-----
// Variables to be used by the Controller Class
//-----

var () float perceptionDistance;           // Myopia factor : )
var () float attackDistance;                // Death distance.
var () array<NavigationPoint> navigationPointsScreamer;

//=====
//-----Just in case they are not initialized,-----
//-----let's give them a default value-----
//=====

defaultproperties
{
    GroundSpeed = MINGROUNDSPEED
    AnimSetName="ATTACK"
    Attacking=false
    logactive = false;
    perceptionDistance = 10000
    attackDistance = 300.0f

    defaultMesh=SkeletalMesh'CH_IronGuard_Male.Mesh.SK_CH_IronGuard_MaleA'
    defaultAnimTree=AnimTree'CH_AnimHuman_Tree.AT_CH_Human'
    defaultAnimSet(0)=AnimSet'CH_AnimHuman.Anims.K_AnimHuman_BaseMale'

    defaultPhysicsAsset=PhysicsAsset'CH_AnimCorrupt.Mesh.SK_CH_Corrupt_Male_Physic
s'

    Begin Object Name=WPawnSkeletalMeshComponent

```

```

SkeletalMesh=SkeletalMesh'CH_IronGuard_Male.Mesh.SK_CH_IronGuard_MaleA'
    AnimTreeTemplate=AnimTree'CH_AnimHuman_Tree.AT_CH_Human'
    AnimSets(0)=AnimSet'CH_AnimHuman.Anims.K_AnimHuman_BaseMale'

    bOwnerNoSee=false
    CastShadow=true
    BlockRigidBody=true
    BlockActors=true
    BlockZeroExtent=true
    BlockNonZeroExtent=true
    bAllowApproximateOcclusion=true
    bForceDirectLightMap=true
    bUsePrecomputedShadows=false
    LightEnvironment=MyLightEnvironment
End Object
mesh = WPawnSkeletalMeshComponent

Begin Object Name=CollisionCylinder
    CollisionRadius=+0041.000000
    CollisionHeight=+0044.000000
    BlockZeroExtent=false
End Object
CylinderComponent=CollisionCylinder
CollisionComponent=CollisionCylinder

bCollideActors=true
bPushesRigidBodies=true
bStatic=False
bMovable=True
bAvoidLedges=true
bStopAtLedges=true
LedgeCheckThreshold=0.5f
}

//=====
//----- Functions/Events -----
//=====

//-----
// This function is only used to add the weapon, defined within
// internal messages in case the logactive flag is turned on.
// This weapon represents the mechanism used to toss fireballs.
//-----
function AddDefaultInventory()
{
    InvManager.CreateInventory(class'ScreamerWeapon');
}

//-----
// The main purpose of this function is to establish the link
// between this Pawn and its controller Class.
//-----
simulated function PostBeginPlay()
{
    super.PostBeginPlay();
    SetPhysics(PHYS_Walking);
    AddDefaultInventory(); //Attach the weapon
    if (myController == none)
    {
        myController = Spawn(class'ScreamerController', self);
        myController.SetPawn(self);
    }
}

```

```

//-----
// This function is only used for debug purpose. It simply unleashes
// internal messages in case the logactive flag is turned on.
//-----
function LogMessage(String texto)
{
    if (logactive)
    {
        Worldinfo.Game.Broadcast(self, texto);
    }
}

//-----
// Whenever the Attack State is changed within the Controller class,
// (i.e.: either entering or leaving the Attack state) this function
// is called so that the Attacking variable can be properly updated.
// This variable can be useful to trigger future Kismet Sequences.
//-----
function SetAttacking(bool atacar)
{
    Attacking = atacar;
}

//-----
// This function is linked to the revenge timer and with the Attack
// substate Revenge.
// It will be called by the Controller class whenever it receives
// the notification that the Screamer has been hit requesting to
// increase its speed velocity and also its speed to toss fireballs.
// (Note: Changes in the toss fireball speed will be implemented
// in Demo-2 version)
// Before changing these values, the Butcher must check whether the
// new requested values are within acceptable limits.
// On the opposite direction, when revenge timer runs out, this
// function is internally called in order to decrease these values.
// to their original ones.
//-----
function ChangeSpeed(int speed)
{
    reduceSpeedTimer = 0;                                // Reset revenge timer
    groundspeed = groundspeed + speed;
    if (groundspeed > MAXGROUNDSPEED)
    {
        groundspeed = MAXGROUNDSPEED;
    }
    if (groundspeed < MINGROUNDSPEED)
    {
        groundspeed = MINGROUNDSPEED;
    }
}

//-----
// This event is notified whenever the Screamer is hit. It has been
// overwritten here just to send a notification for the controller
// class by calling the NotifyTakeHit1 function.
//-----
event TakeDamage (int Damage, Controller EventInstigator, Object.Vector
HitLocation, Object.Vector Momentum, class<DamageType> DamageType, optional
Actor.TraceHitInfo HitInfo, optional Actor DamageCauser)

```

```

{
    LogMessage("Event Pawn TakeDamage");

super.TakeDamage(Damage,EventInstigator,HitLocation,Momentum,DamageType,HitInfo,DamageCauser);
    myController.NotifyTakeHit1();
}

-----  

// This event is from utmost importance to every Pawn.  

// At some pre-determined intervals (usually between 0.05 and 0.1  

// seconds) this event will be triggered and for instance, the Pawn  

// can use these notifications in order to take timing based actions.  

// In this case, a function to drain the Player life is implemented  

// in order to reduce the Player health whereas he is in contact  

// with the Screamer.  

// Here, revenge timer is also implemented, with the aid of  

// reduceSpeedTimer variable so that when a predetermined timeout  

// (equal to 100 Ticks) is reached, a request to reduce the  

// Screamer speed will be carried out by calling the ChangeSpeed  

// function with a negative value (in this case, if the Attack  

// sub-state is Revenge, it will be set back to Normal Attack).
-----  

simulated event Tick(float DeltaTime)
{
    local UTPawn gv;
super.Tick(DeltaTime);
if (tickCounter < MAXTICKERCOUNTER)
{
    tickCounter +=1;
}
else                                // Wait 10 primary timer intervals
{
    reduceSpeedtimer = reduceSpeedtimer + 1;
    tickCounter = 0;
foreach VisibleCollidingActors(class'UTPawn', gv, 100)
{
    if(Attacking && gv != none)
    {
        if(gv.Health > 10)      // Contact will not kill player.
        {
            gv.Health -= 1;
            gv.IsInPain();
        }
    }
}
if (reduceSpeedTimer == 10)      // Wait 100 primary timer intervals
{
    reduceSpeedTimer = 0;
    ChangeSpeed(REDUCESPEEDONTIMEOUT);
}

}

-----  

// This is the function responsible for changing the character when  

// new version of the Mesh, AnimSet or Physical Asset are available.
-----  

simulated function SetCharacterClassFromInfo(class<UTFamilyInfo> Info)
{
    Mesh.SetSkeletalMesh(defaultMesh);
    Mesh.SetMaterial(0,defaultMaterial0);
}

```

```

    Mesh.SetPhysicsAsset(defaultPhysicsAsset);
    Mesh.AnimSets=defaultAnimSet;
    Mesh.SetAnimTreeTemplate(defaultAnimTree);

}

```

## 6 Kismet

Nesta seção, todas as imagens dos Kismets utilizados na implementação são listadas. A figura 1 apresenta a herarquia das sequências desenvolvidas até agora. Para cada sequência uma imagem demonstrando a implementação será apresentada.

### 6.1 Contagem de tempo

A contagem de tempo para o jogo foi desenvolvida usando a própria estrutura dada pelo mapa que usamos como base para criação do nosso jogo. Neste mapa, o tempo para finalização já era de 20 minutos. Foi desenvolvida em Kismet<sup>1</sup> (Figura 6.6) uma sequência que monitora o tempo do jogo e só inicia a contagem no momento que o papel com a dica de senha do cofre é encontrada. De forma resumida, foram implementados os seguintes itens:

- Tempo do jogo só inicia ao encontrar o papel com dica da senha (Figura 6.7).
- Badaladas de sino à meia-noite, uma, duas e três da manhã (Figura 6.7).
- Morte de Marshal se o jogo não termina até às 3h33 (Figura 6.8).
- Conversão dos 20 minutos (tempo real) para as três horas e trinta e três minutos (tempo do jogo) (Figura 6.9, Figura 6.10, Figura 6.11 e Figura 6.12).

### 6.2 Sistema de danos – Regeneração

O sistema de danos foi implementado conforme as especificações neste documento. Após algum dano tomado, um timer aguarda por 5 segundos e começa a regeneração do personagem (Figura 6.28).

### 6.3 Outros elementos

Como sequências auxiliares, foram criadas as seguintes funcionalidades:

- Raios e trovões (Figura 6.34).
- Configuração para a shoulder câmera (Figura 6.29).
- Obter uma referência para o jogo (Figura 6.33).
- Obter uma referência para o jogador (Figura 6.32).

---

<sup>1</sup> Kismet é uma linguagem visual de *script* para a plataforma UDK. Ela permite aos desenvolvedores criarem programas em *UnrealScript* para eventos do jogo usando uma interface visual.

## 6.4 Fase 1 – Living Room (Sala do Puzzle)

Foram implementados nesta etapa o quadro de distribuição de energia e o quadro (pintura) que esconde o cofre com a chave que abre a porta.

Para o quadro de distribuição de energia, foram implementados os seguintes itens:

- Ao iniciar a fase, uma explosão abre a porta do quadro de distribuição de energia as luzes se apagam (Figura 6.25).
- Quatro chaves de liga-desliga foram criadas. Elas variam do valor 0 (desligada) para o valor 1 (ligada). A implementação do visual e do valor que cada chave possui foi implementado na mesma sequência (Figura 6.24).
- Após escolher a ligação de chaves, o jogador deve tentar ligar as luzes da sala acionando a alavanca da chave geral. Nestemo momento, uma sequência valida se as chaves estão ligadas corretamente (Figura 6.27) e, dependendo da ligação, o jogador toma dano ou as luzes se acendem (Figura 6.26).

Para o cofre, foram implementados os seguintes itens:

- Ao iniciar o jogo, uma entre 5 opções de senha do cofre é sorteada (Figura 6.20).
- Se o jogador andar no canto esquedo da sala, a dica referente a senha sorteada é apresentada (Figura 6.22).
- Porta da sala está trancada. Se tentar abrir toca som de porta trancada. Se possuir a chave consegue abrir a porta (Figura 6.14).
- Uma pintura esconde o cofre. É necessário abrir esta pintura para ver o cofre atrás dela (Figura 6.18).
- Três dígitos para o cofre. Quando se interage com um dígito, ele mostra o número que está selecionado. Para isso existe uma sequência que controla a parte visual (Figura 6.17) e uma sequência que controla o contador (Figura 6.16).
- Quando a senha informada nos três dígitos do cobre estão corretas, o cofre abre (Figura 6.21).
- Badaladas de sino à meia-noite, uma, duas e três da manhã (Figura 6.7).
- Após a abertura, pode-se obter a chave dentro do cofre (Figura 6.19).

## 6.5 Fase 2 – Corridor e Main Hall

Foram implementados nesta etapa a porta que separa o corredor do Main Hall e a porta que separa o Main Hall da Library (Figura 6.4).

A porta que separa o corredor do Main Hall está trancada e só poderá ser aberta após matar o Butcher que defende a entrada e pegar a chave que está com ele. Uma sequência (Figura 6.3) foi criada para controlar a abertura da porta e outra sequência foi criada para mostrar a chave e o jogador conseguir pegá-la quando ela for tocada (Figura 6.31).

Para esta fase também foi usado uma sequência para vincular a arma à cada Butcher (Figura 6.30) e configurar o dano que o toque da arma causaria no jogador (Figura 6.31).

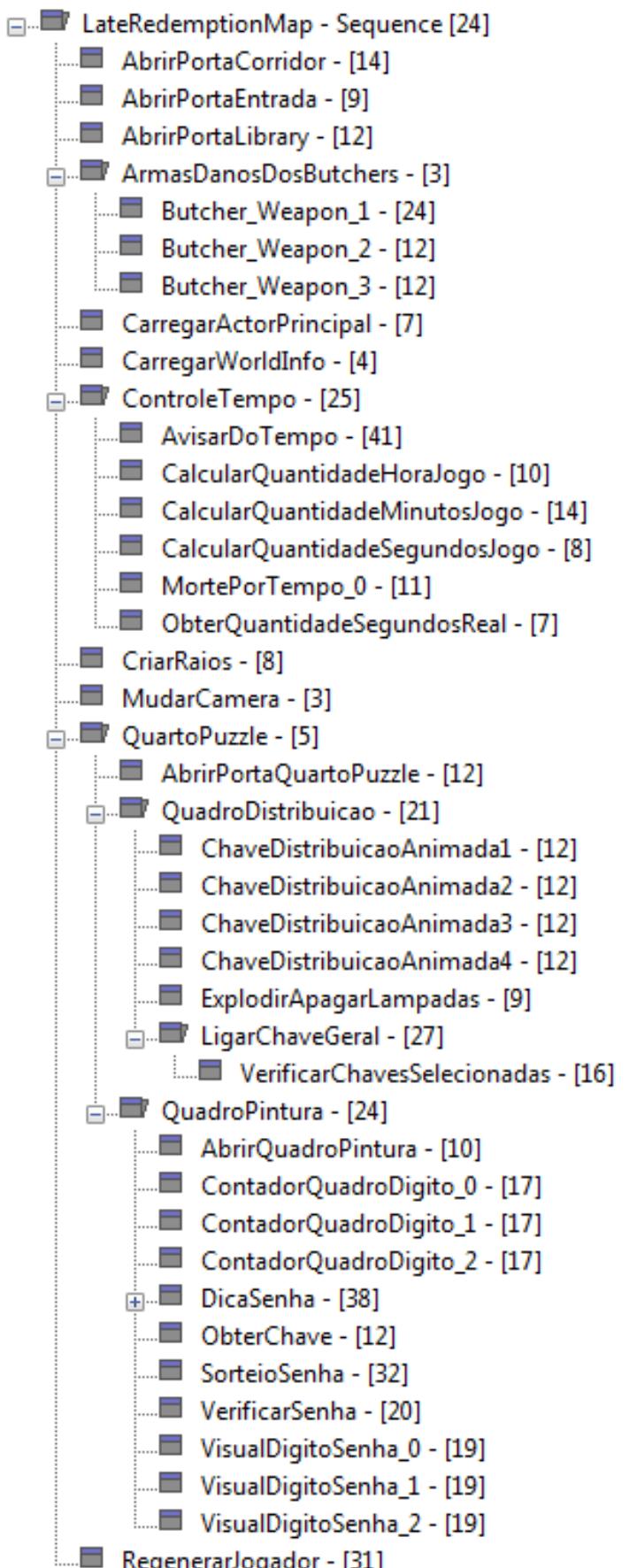


Figura 6.1 – Visão geral das sequências implementadas

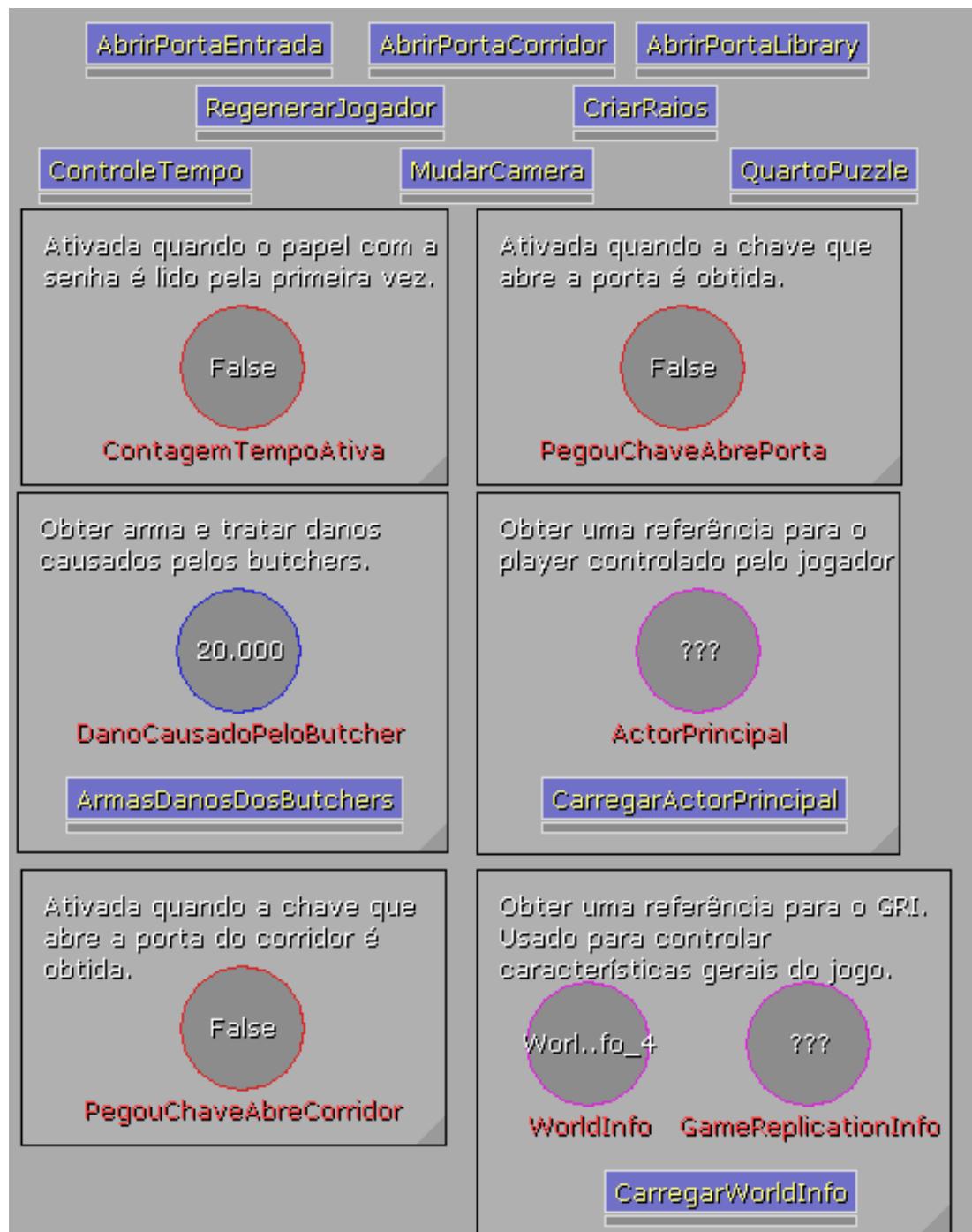


Figura 6.2 – LateRedemptionMap

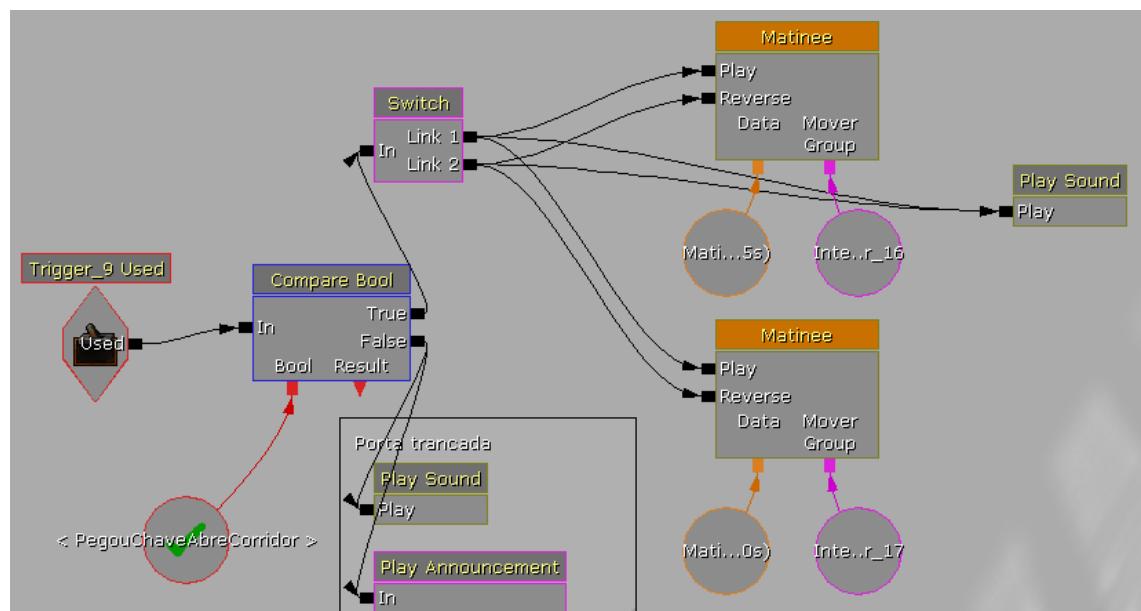


Figura 6.3 – AbrirPortaCorridor

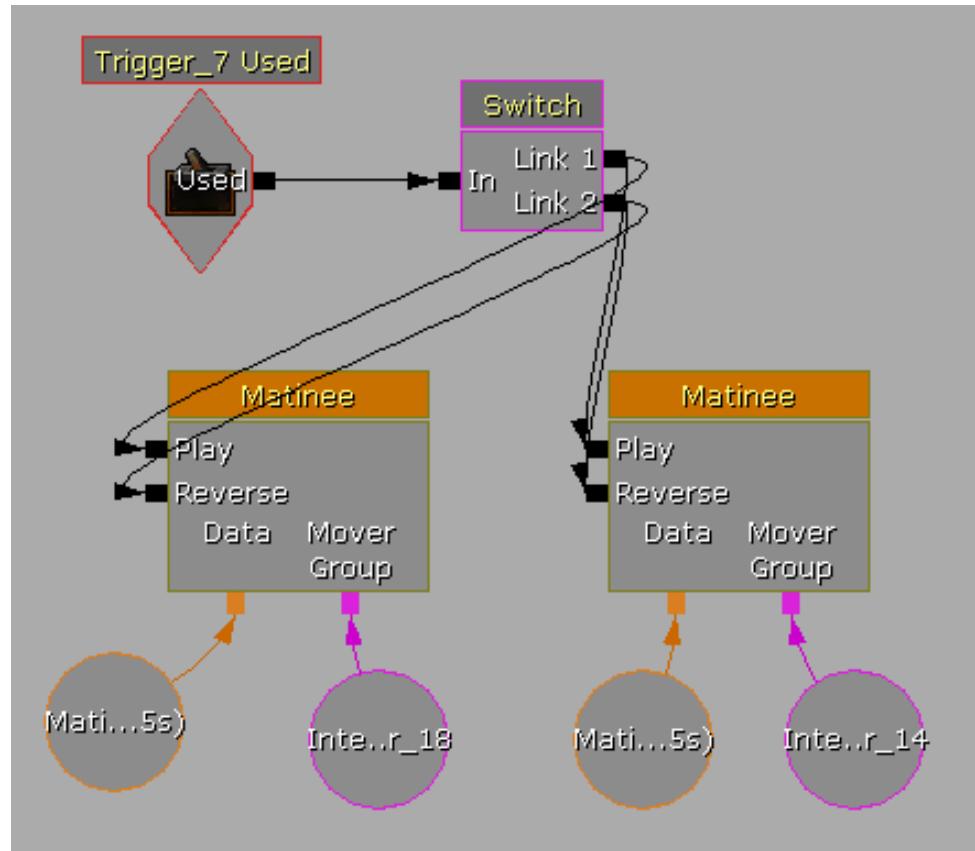
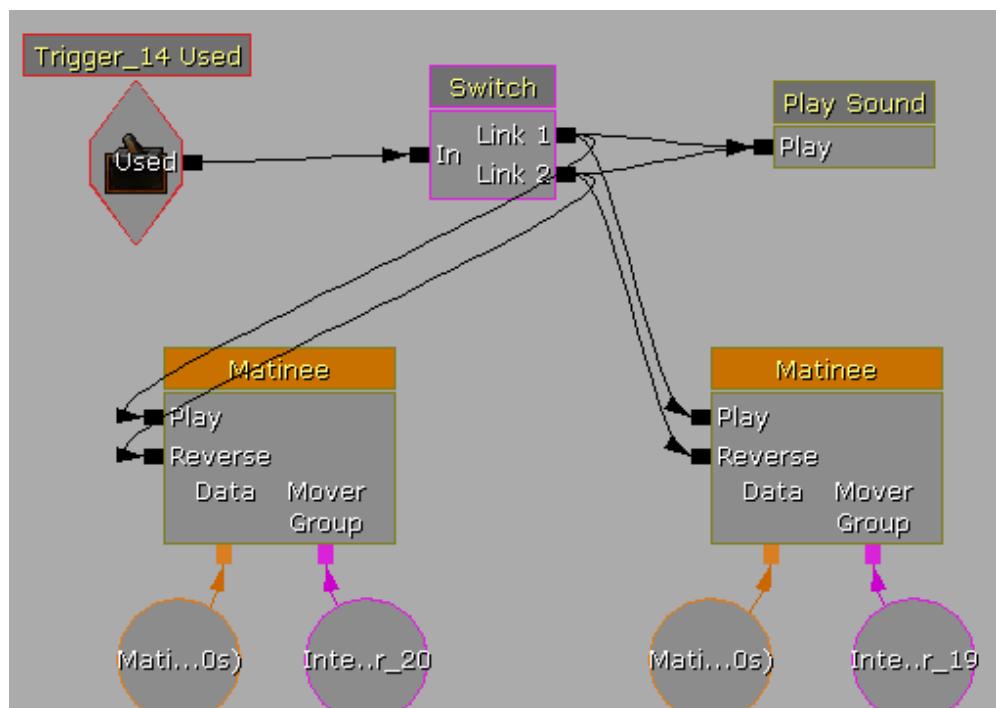


Figura 6.4 – AbrirPortaLibrary



**Figura 6.5 – AbrirPortaEntrada**

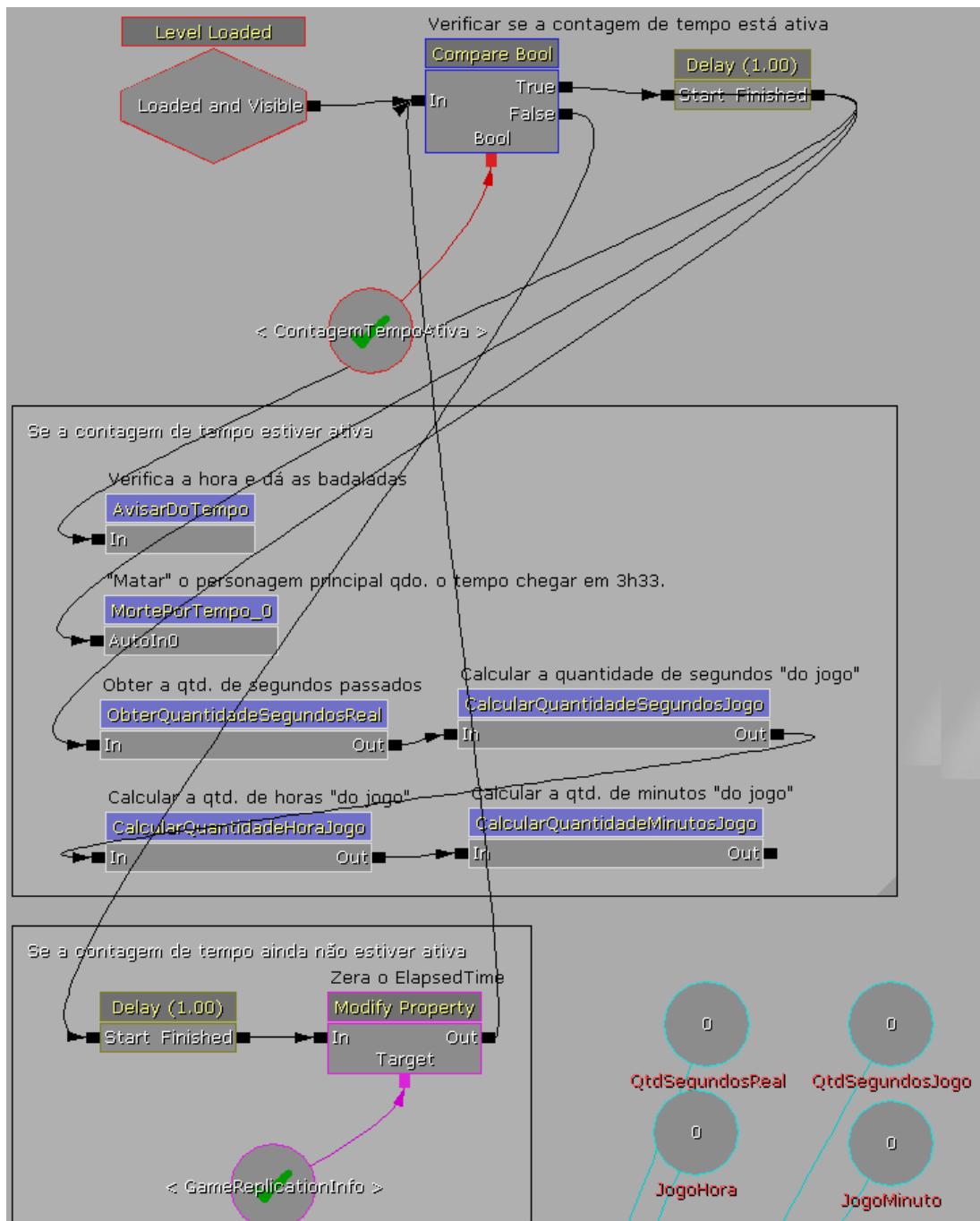
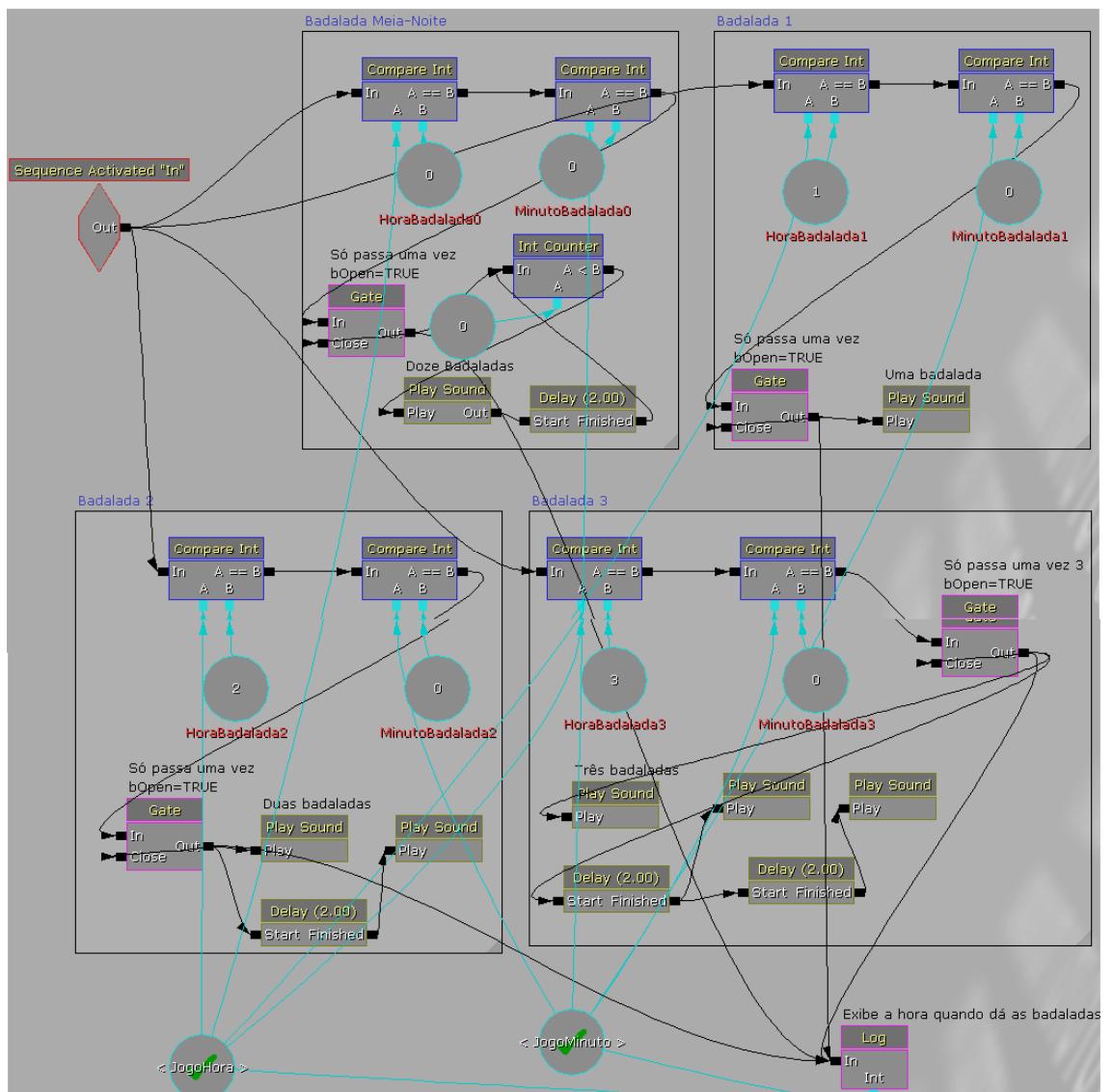


Figura 6.6 – ControleTempo



**Figura 6.7 – ControleTempoAvisarDoTempo**

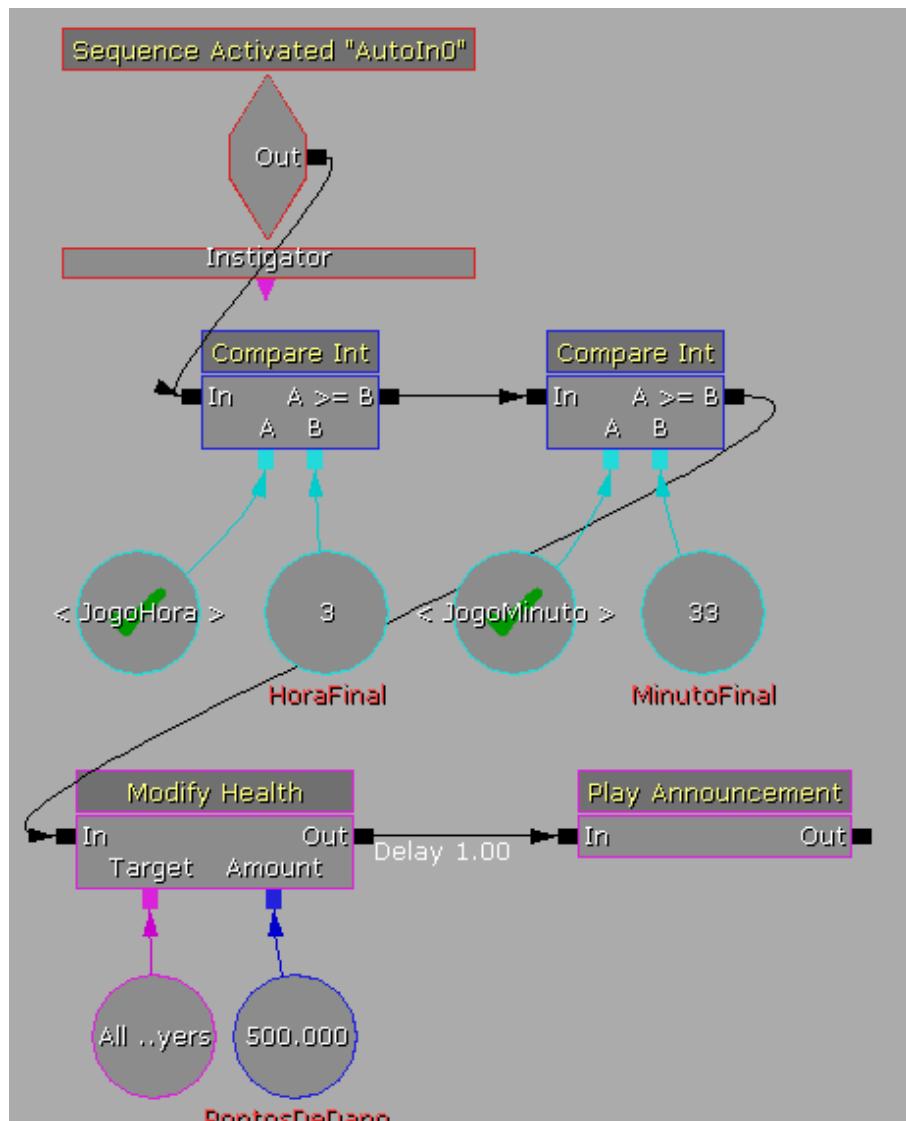
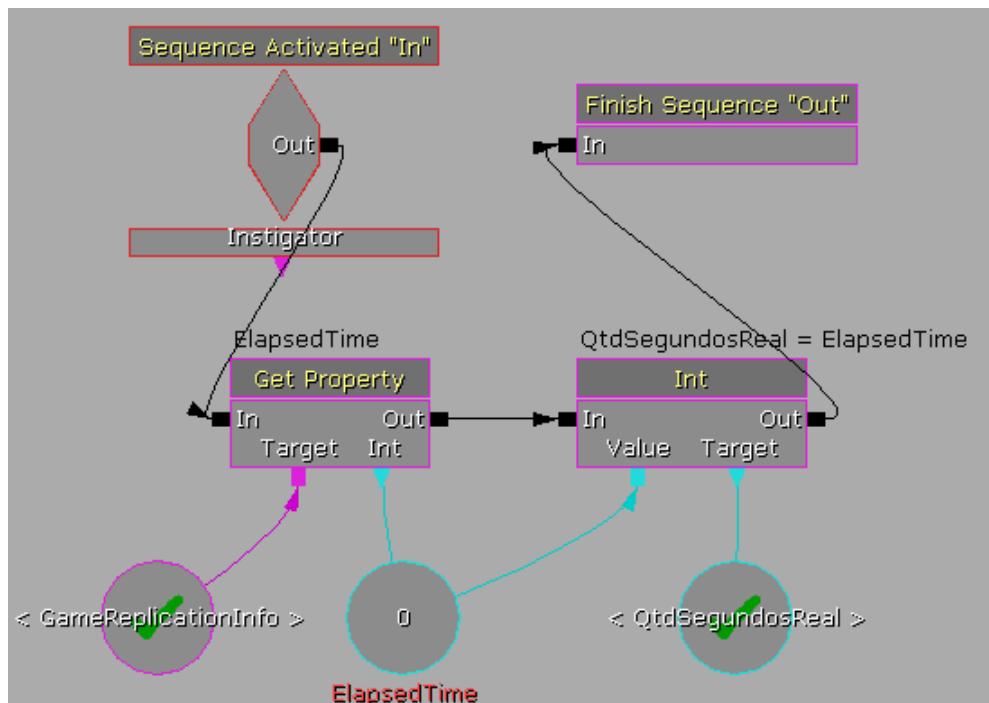
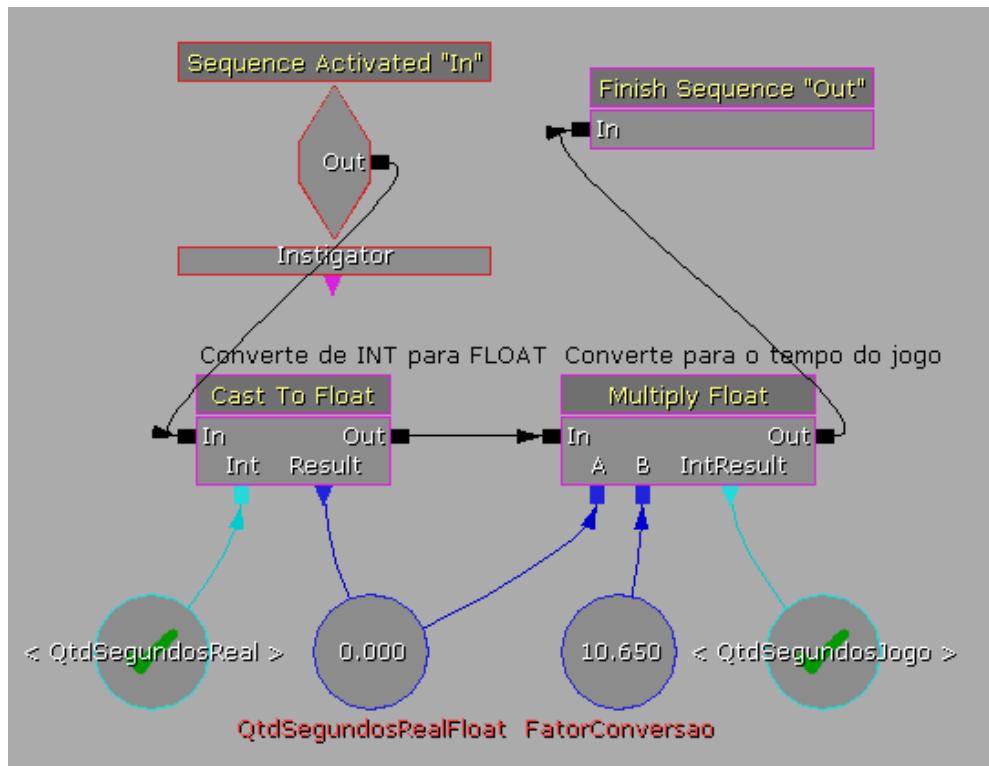


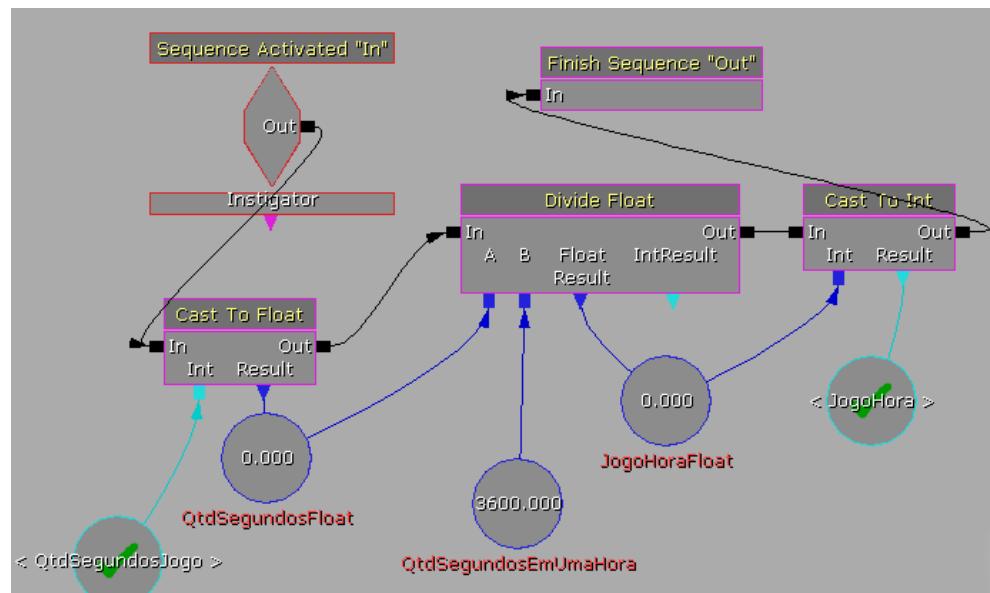
Figura 6.8 – ControleTempoMortePorTempo



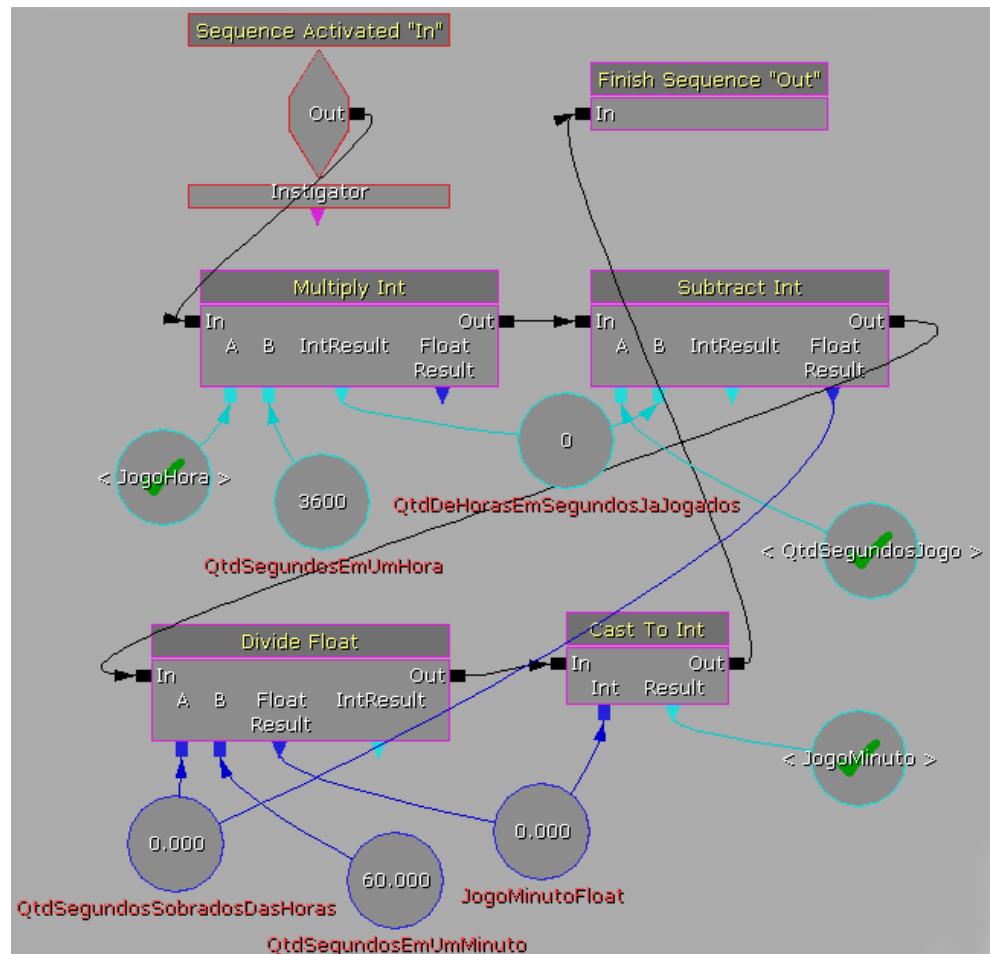
**Figura 6.9 – ControleTempo.ObterQuantidadeSegundoReal**



**Figura 6.10 – ControleTempo.ObterQuantidadeSegundosJogo**



**Figura 6.11 – ControleTempo.CalcularQuantidadeHoraJogo**



**Figura 6.12 – ControleTempo.CalcularQuantidadeMinutosJogo**

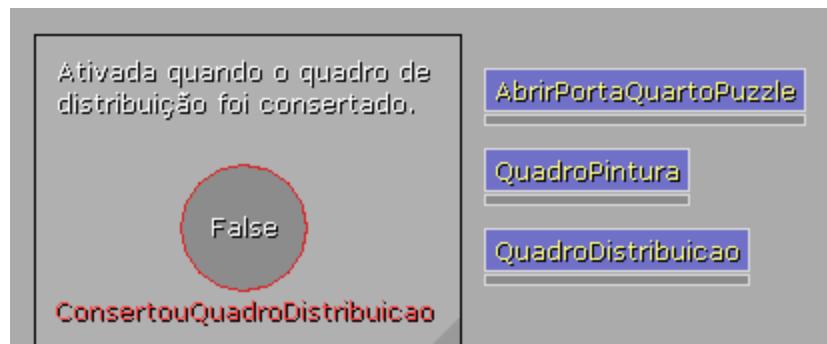


Figura 6.13 – QuartoPuzzle

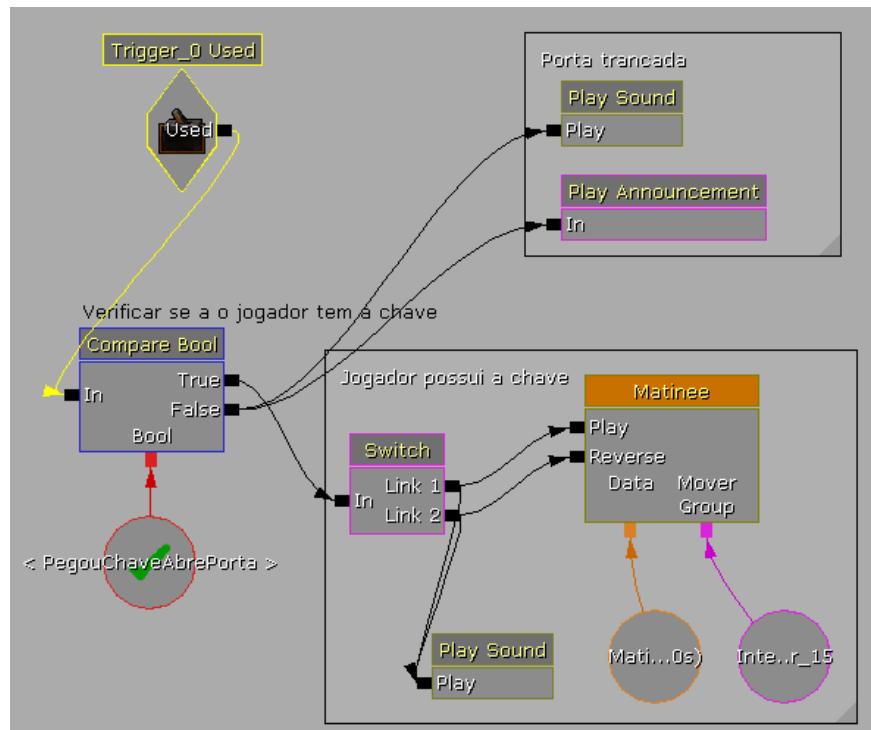
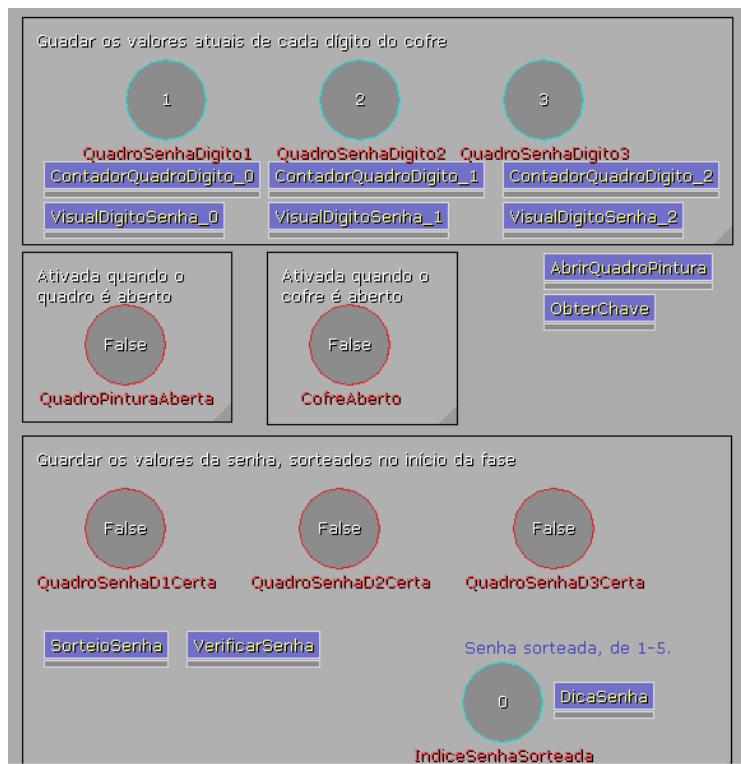
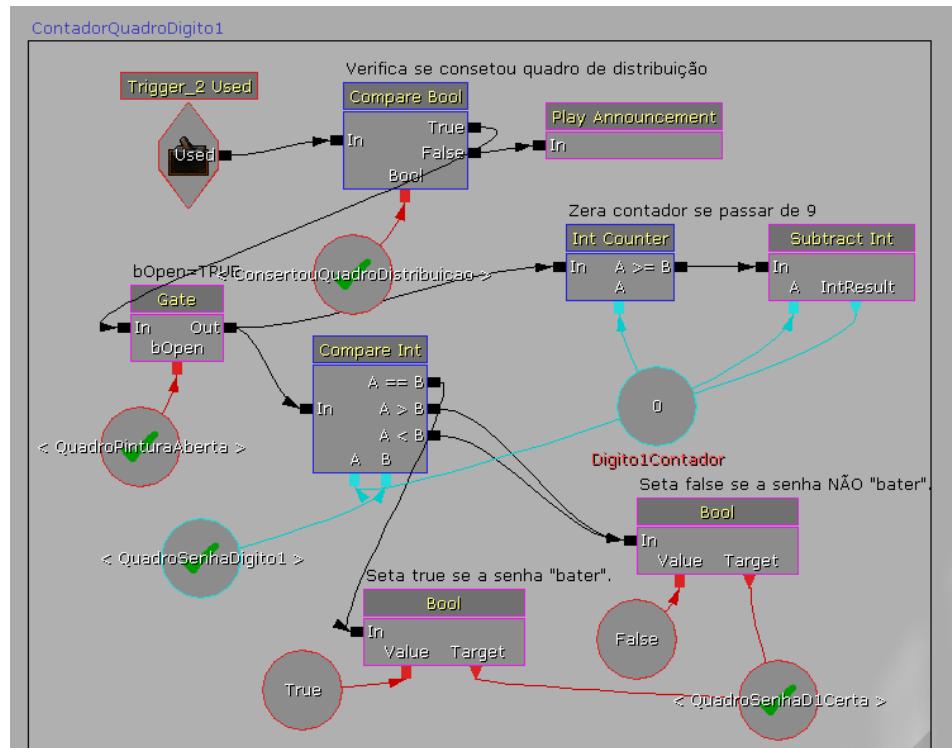


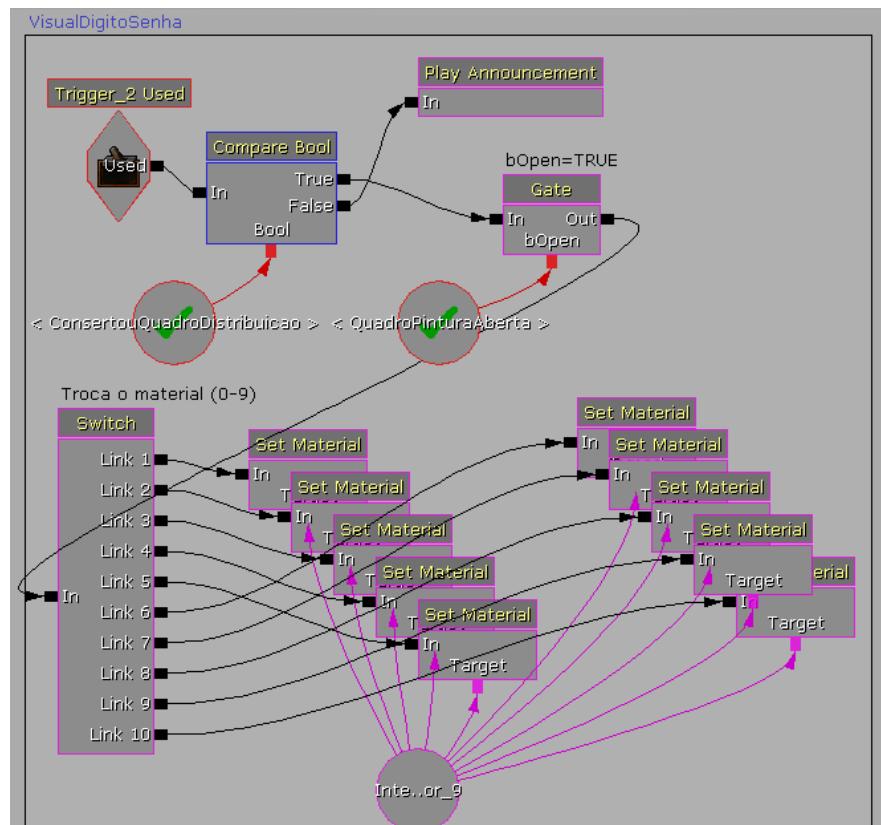
Figura 6.14 – QuartoPuzzleAbrirPortaQuartoPuzzle



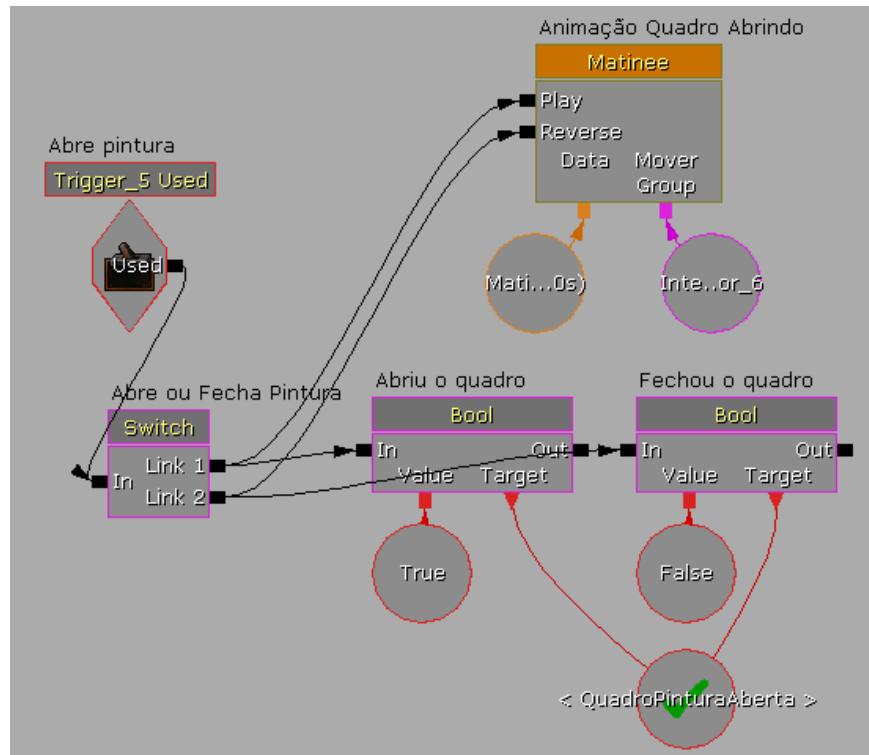
**Figura 6.15 – QuartoPuzzle.QuadroPintura**



**Figura 6.16 – QuartoPuzzle.QuadroPintura.ContadorQuadroDigito**



**Figura 6.17 – QuartoPuzzle.QuadroPintura.VisualDigitoSenha**



**Figura 6.18 – QuartoPuzzle.QuadroPintura.AbrirQuadroPintura**

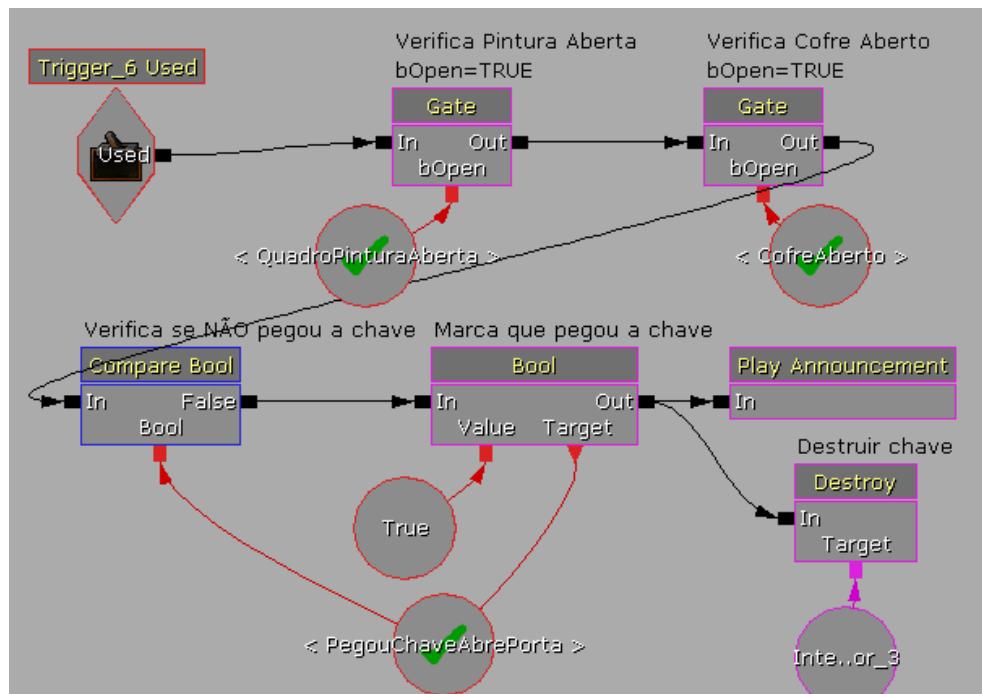


Figura 6.19 – QuartoPuzzle.QuadroPintura.ObterChave

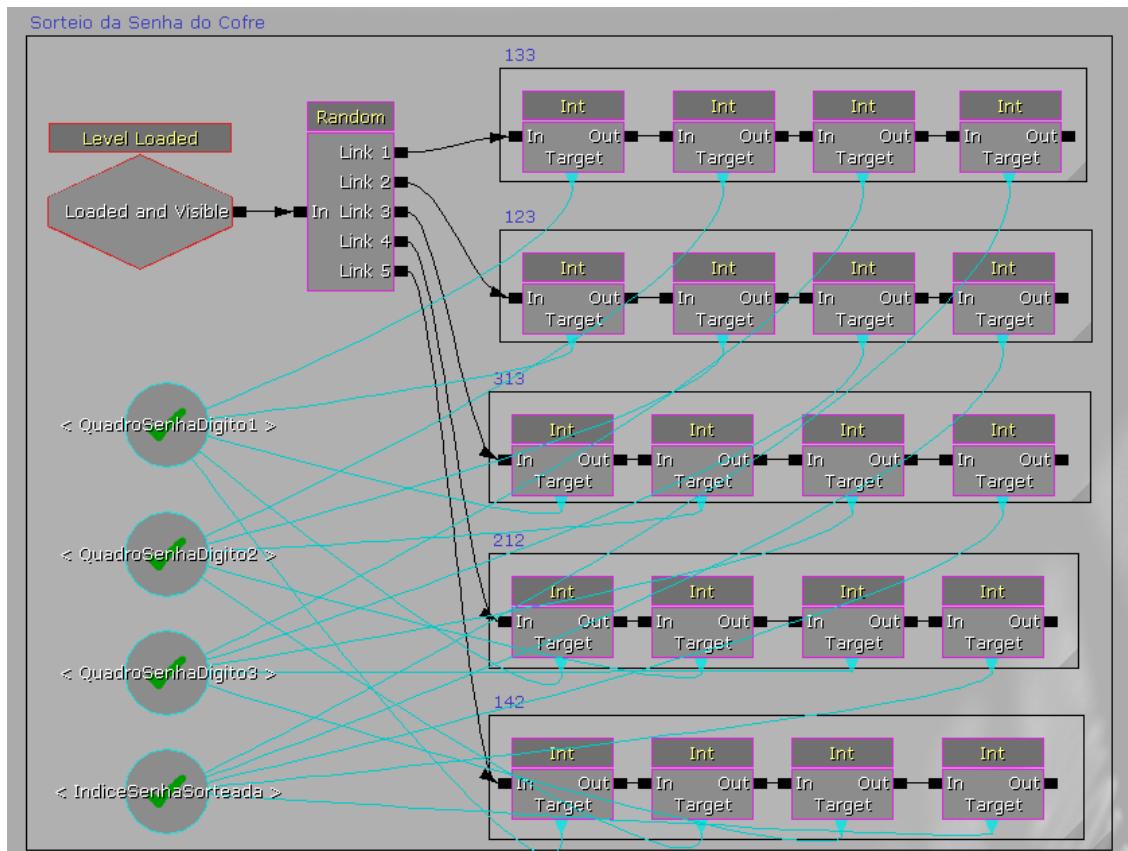
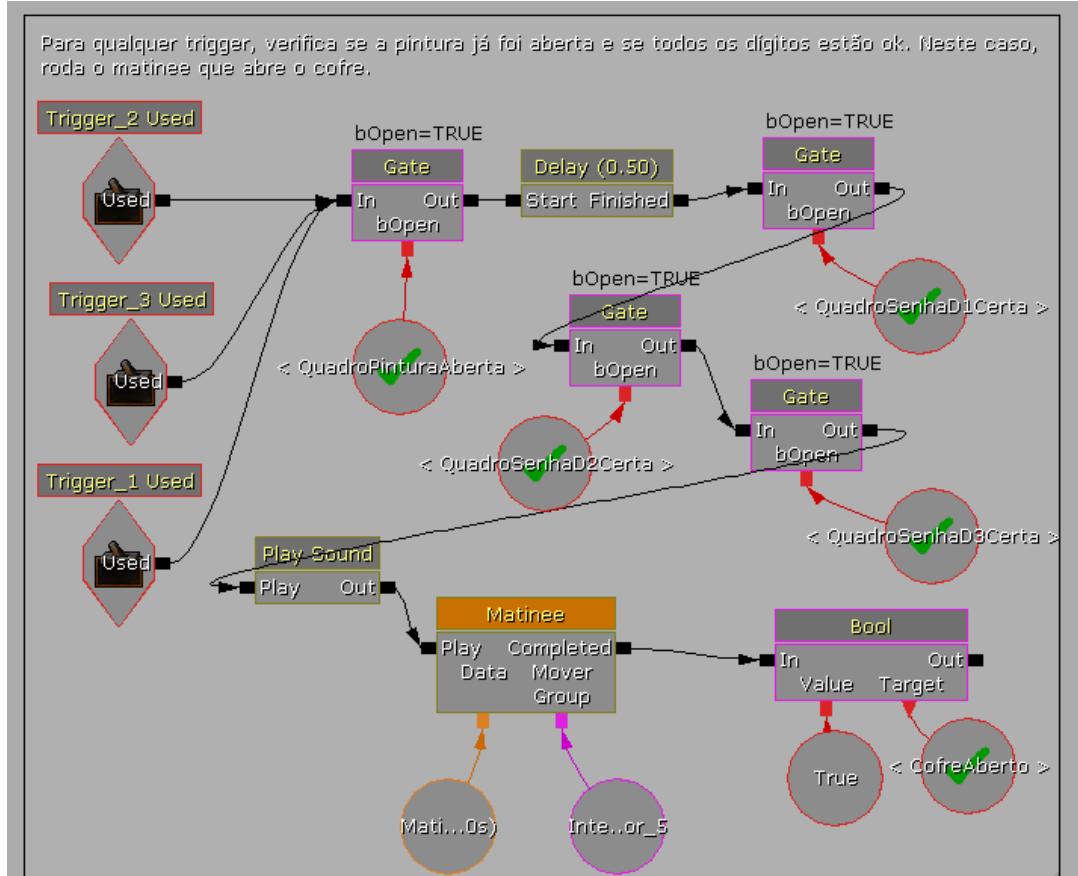
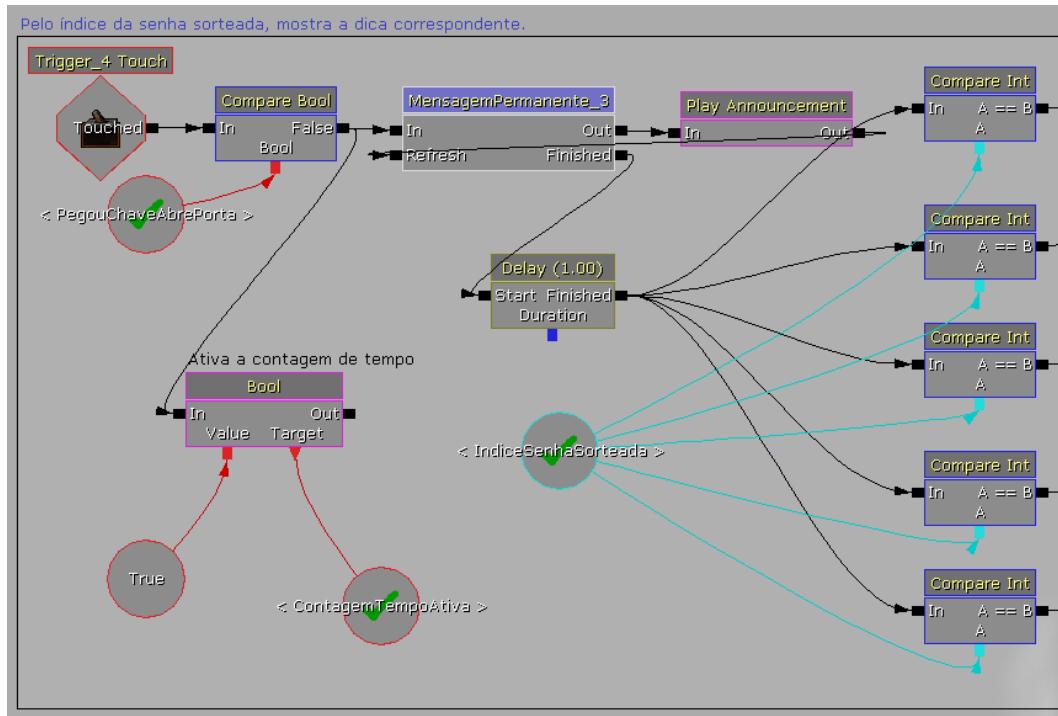


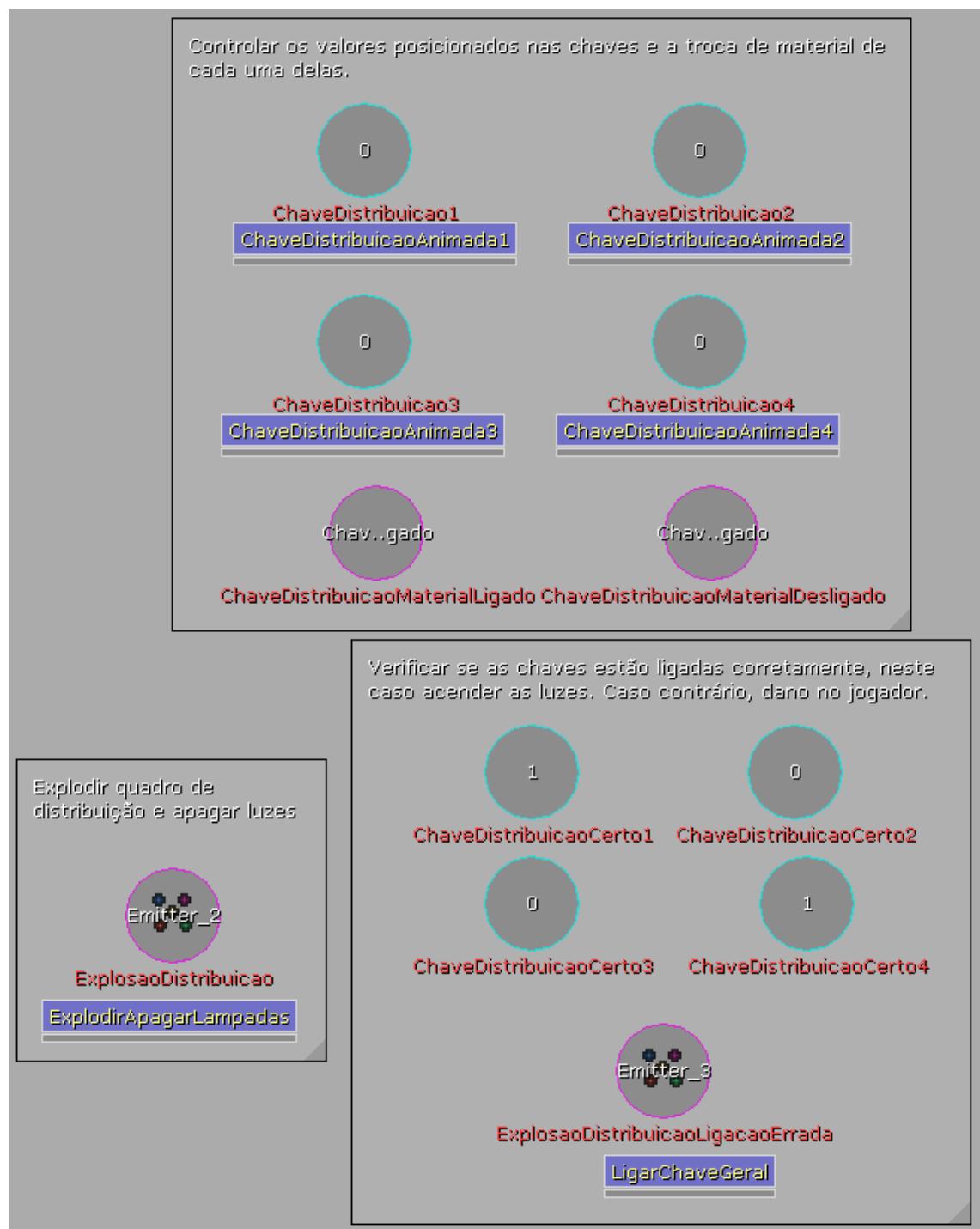
Figura 6.20 – QuartoPuzzle.QuadroPintura.SorteioSenha



**Figura 6.21 – QuartoPuzzle.QuadroPintura.VerificarSenha**



**Figura 6.22 – QuartoPuzzle.QuadroPintura.DicaSenha**



*Figura 6.23 – QuartoPuzzle.QuadroDistribuicao*

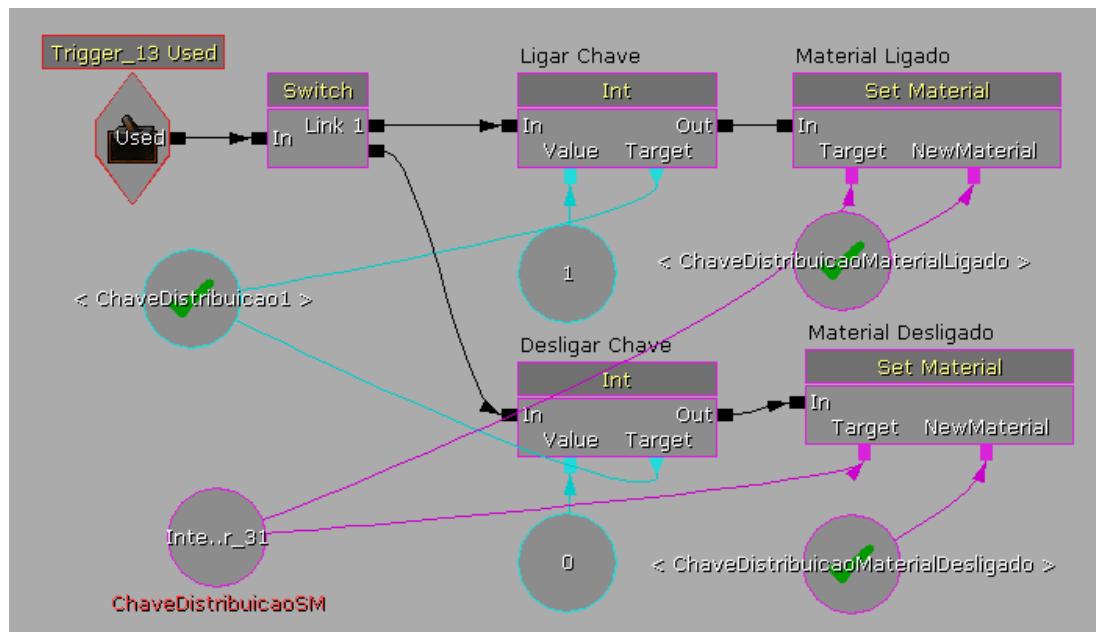


Figura 6.24 – QuartoPuzzle.QuadroDistribuicao.ChaveDistribuicaoAnimada

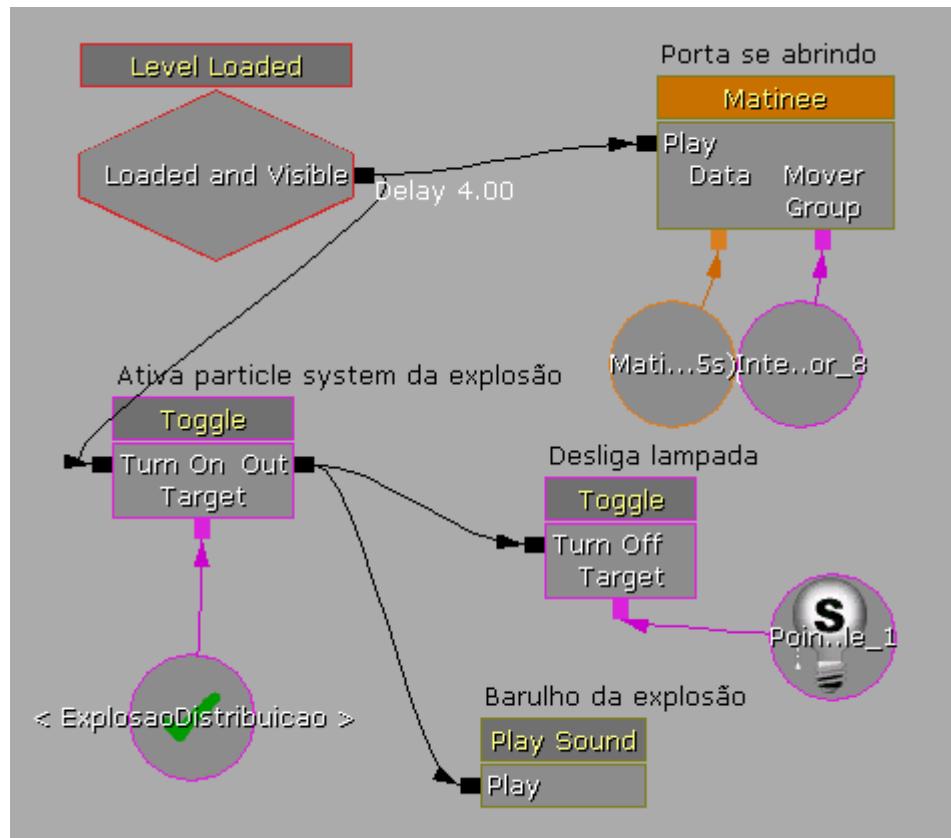


Figura 6.25 – QuartoPuzzle.QuadroDistribuicao.ExplodirApagarLampadas

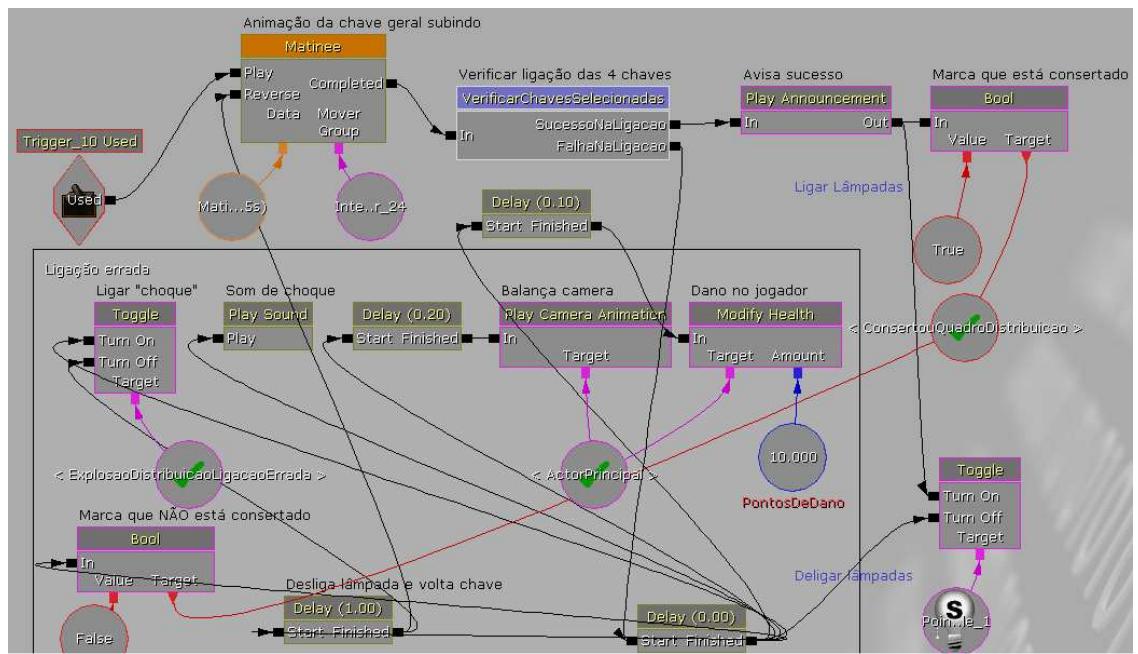


Figura 6.26 – QuartoPuzzle.QuadroDistribuicao.LigarChaveGeral

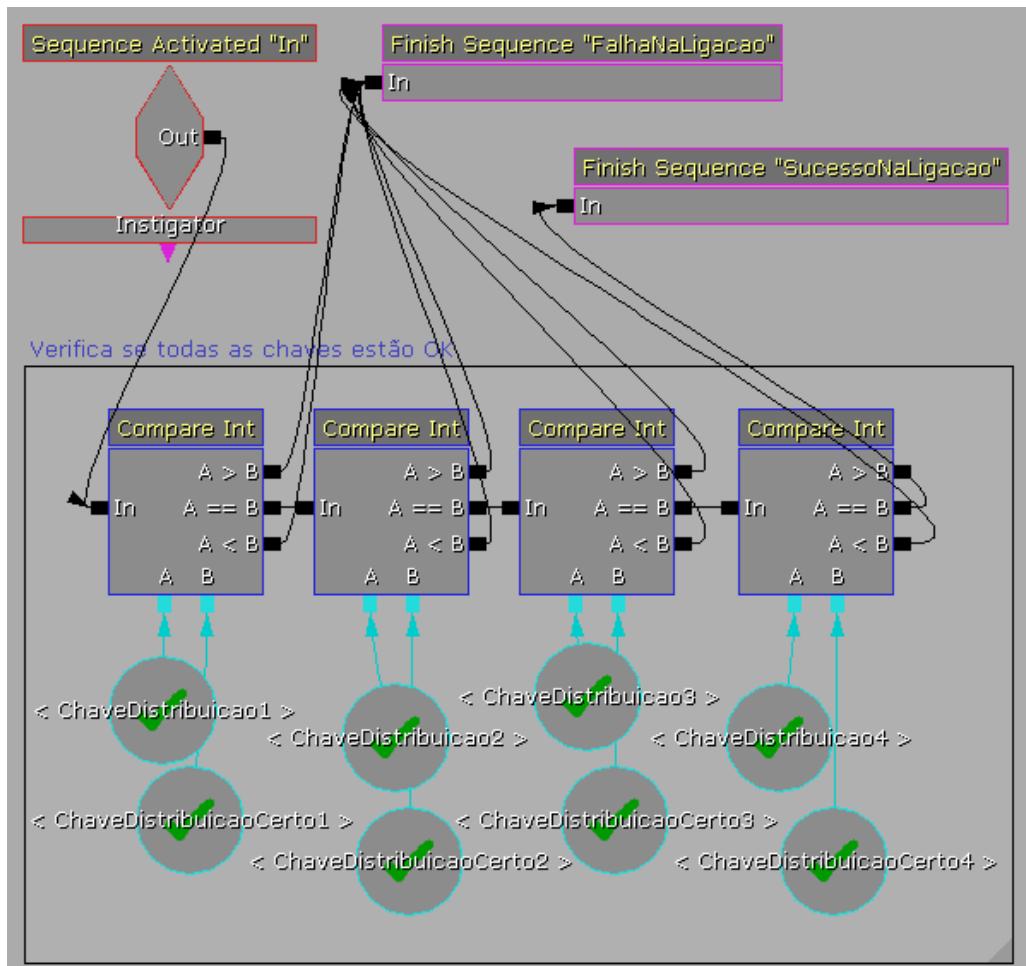


Figura 6.27 – QuartoPuzzle.QuadroDistribuicao.LigarChaveGeral.VerificarChavesSelecionadas

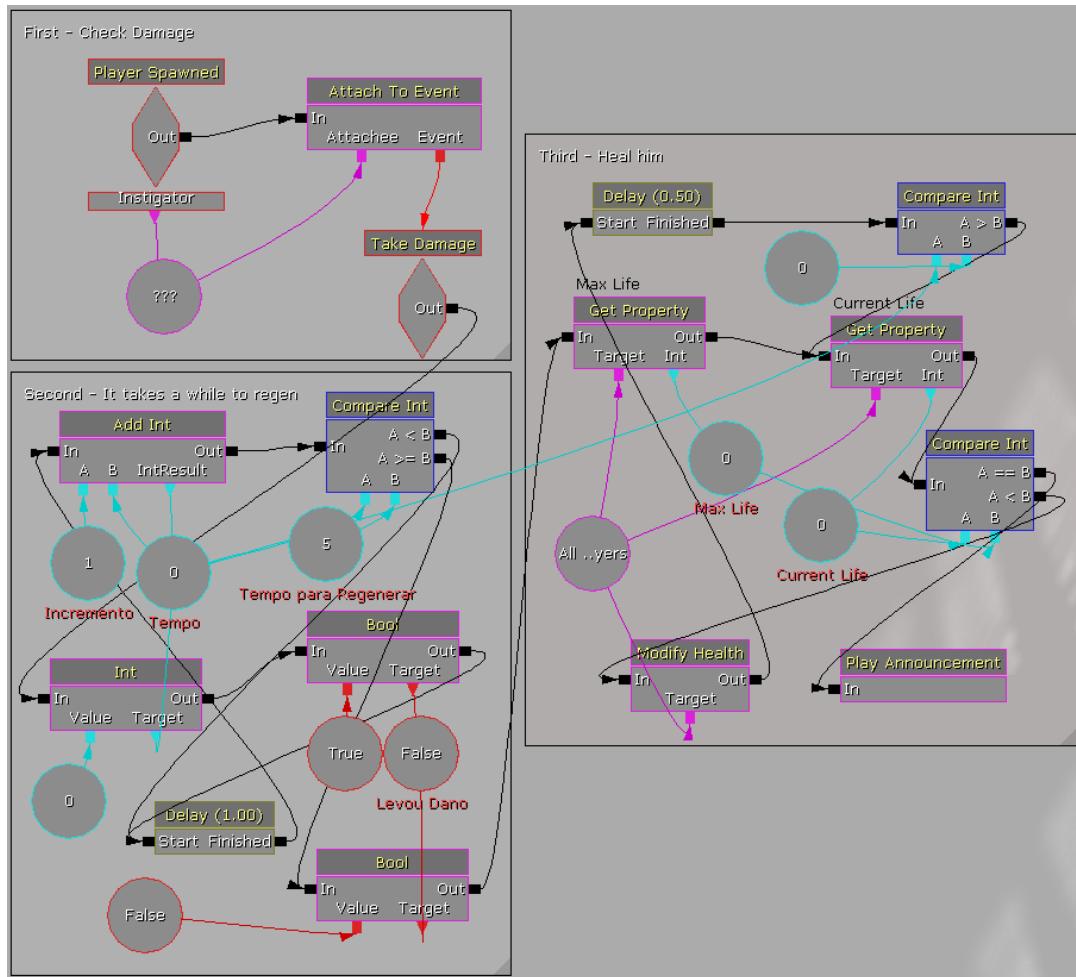


Figura 6.28 – Regenerar Jogador

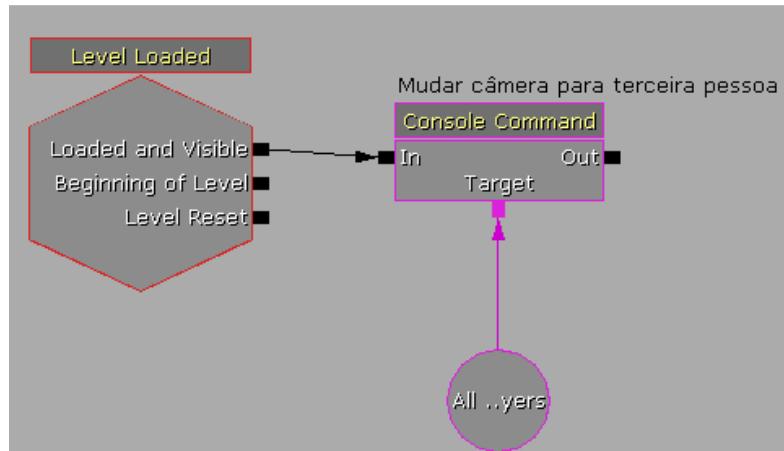


Figura 6.29 – MudarCamera



Figura 6.30 – ArmasDosButchers

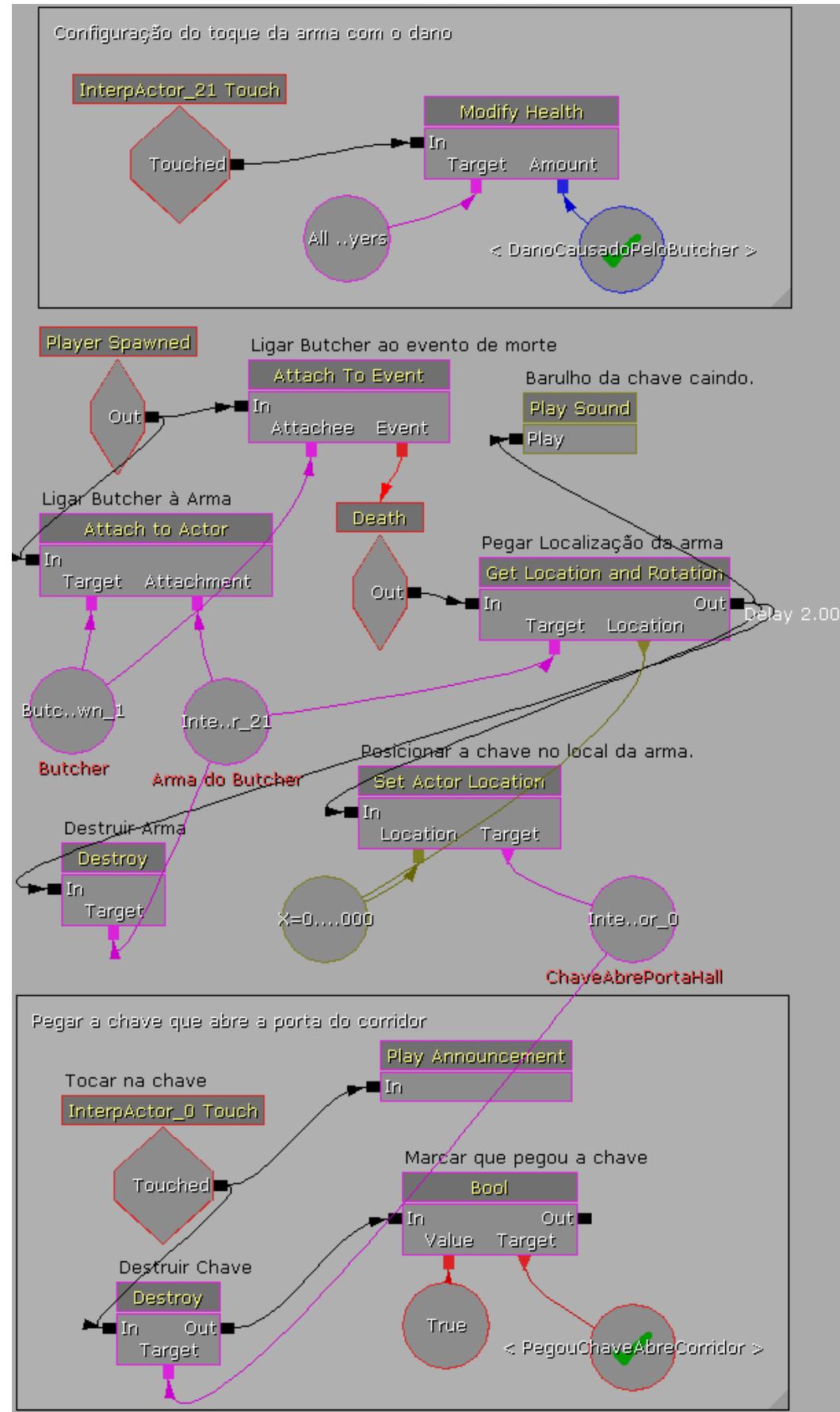
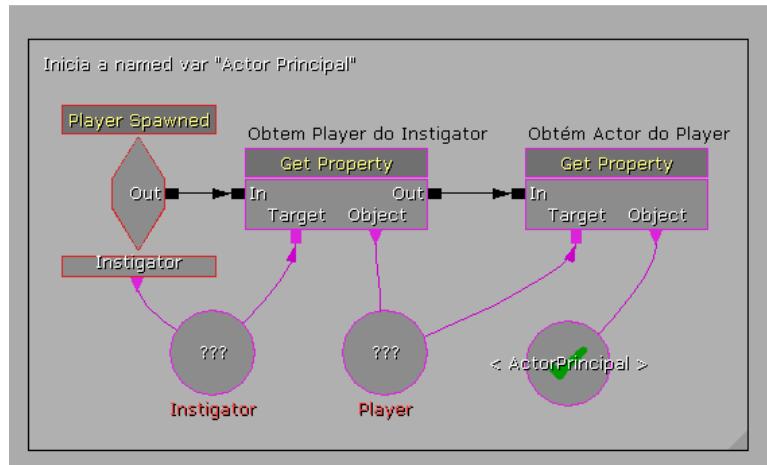
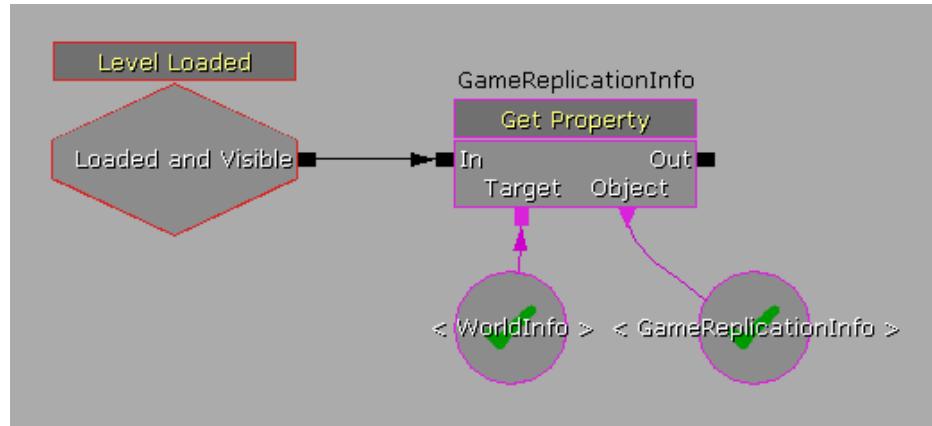


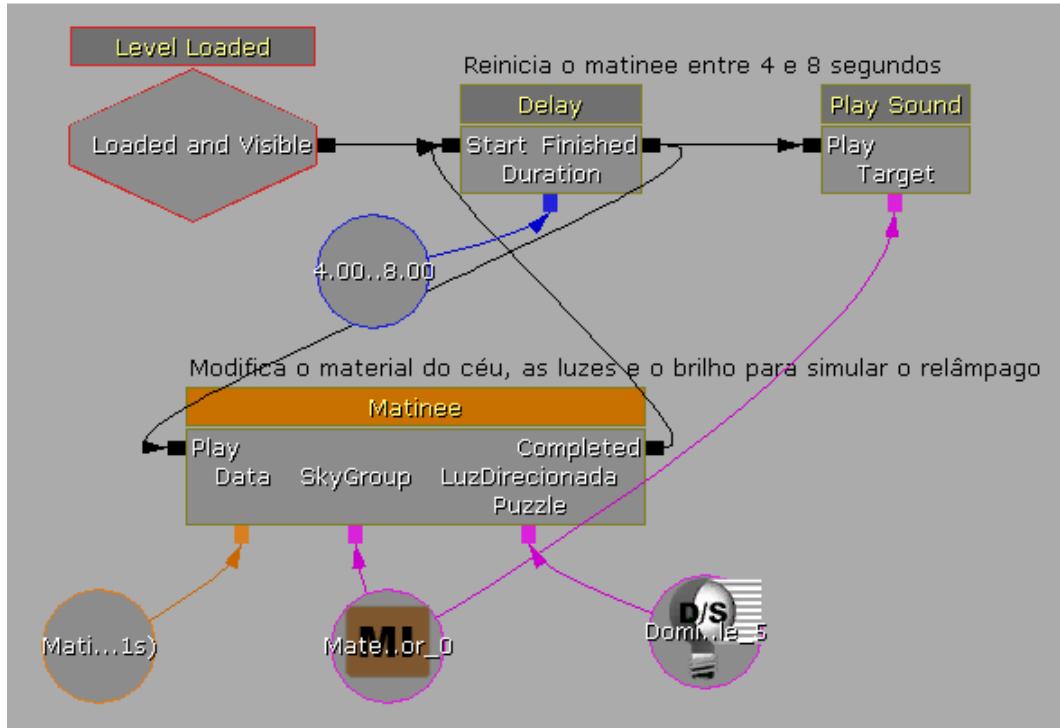
Figura 6.31 – Butcher Weapon



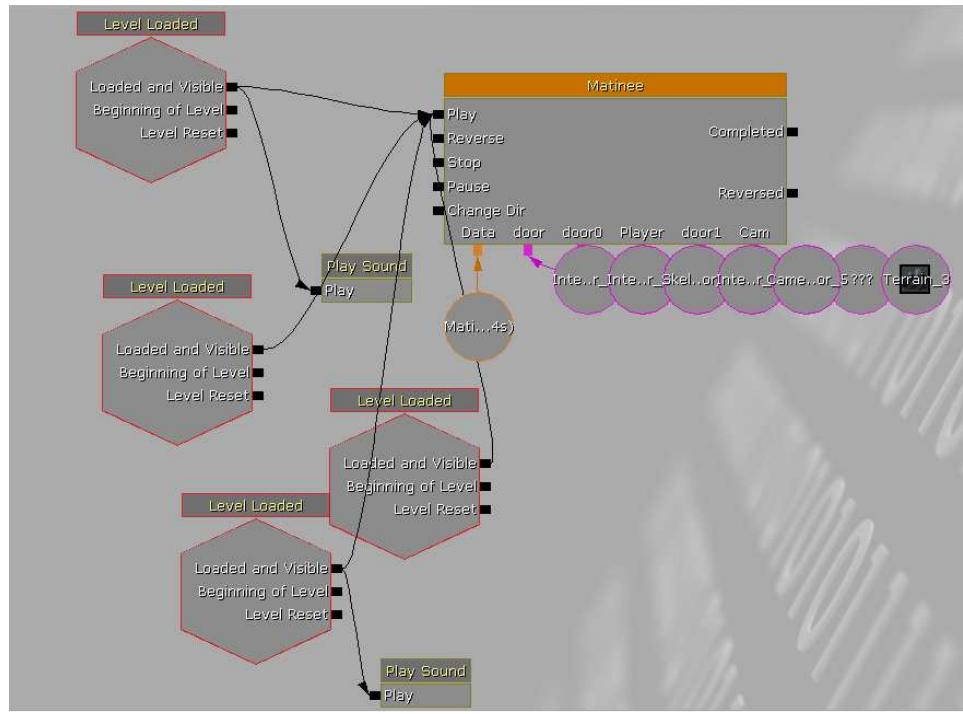
**Figura 6.32 – CarregarActorPrincipal**



**Figura 6.33 – CarregarWorldInfo**



**Figura 6.34 – CriarRaios**



**Figura 6.35 – Cinemática Matinee**

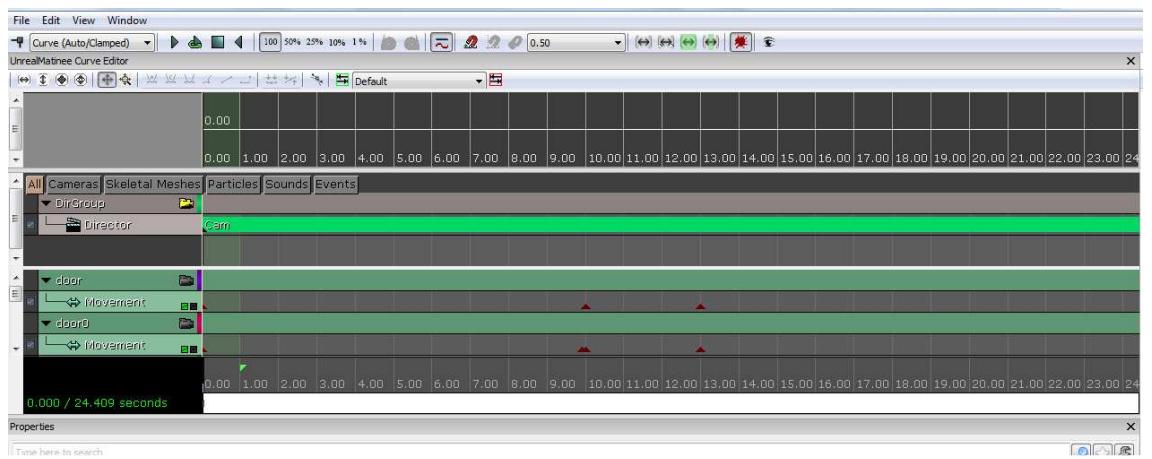
## 7 Cinemática

Para a entrega da primeira demo, foi desenvolvida uma cinematática onde o personagem visualiza a porta principal da mansão abrindo sozinha, em seguida o personagem percorre do corredor da mansão até o quarto do puzzle, chegando ao quarto a porta se tranca e há uma explosão no quadro de distribuição de energia, a luz se apaga. A partir deste momento o jogo se inicia.

Para a realização da cinematática foi desenvolvida a animação do player, porta principal e porta do quarto e em seguida foi inserido uma câmera para gravar a ação das cenas.

### 7.1 Animação

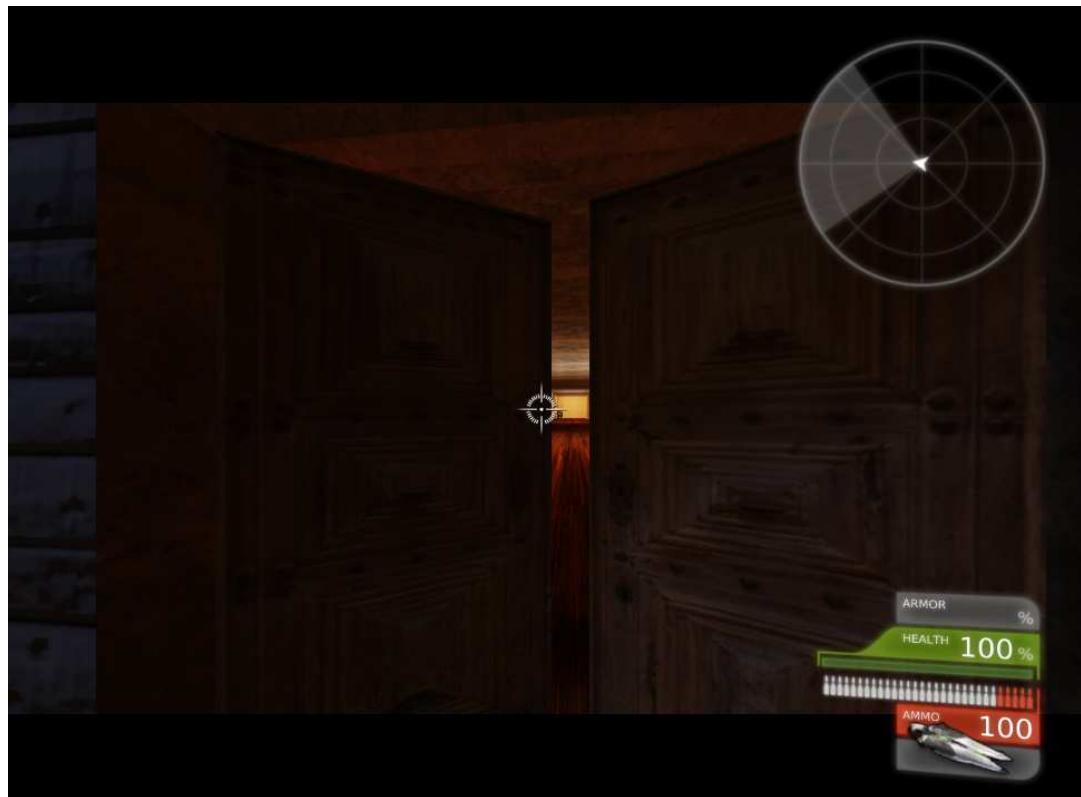
Para isso foi criado um Kismet integrando todos esses elementos (**vide seção Kismet figura 6.35**). Após isso foi implementada a matinee de animação e por fim inserida uma câmera para mostrar cada cena mencionada.



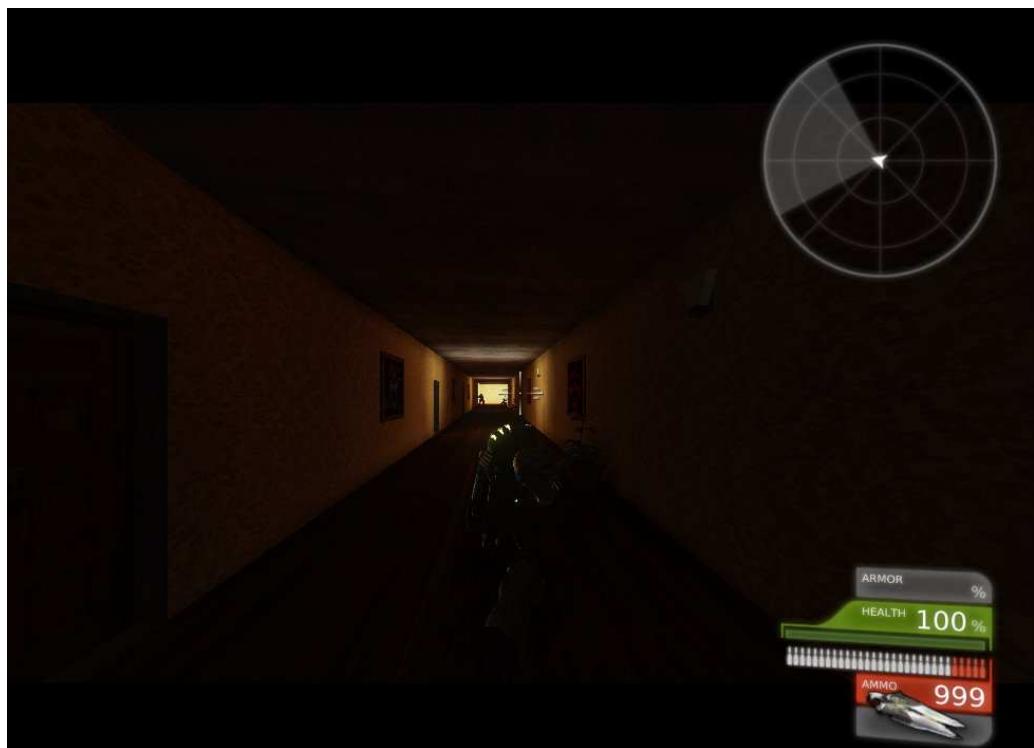
**Figura 7.1 – Cinematica Matinee**

## 7.2 Cenas

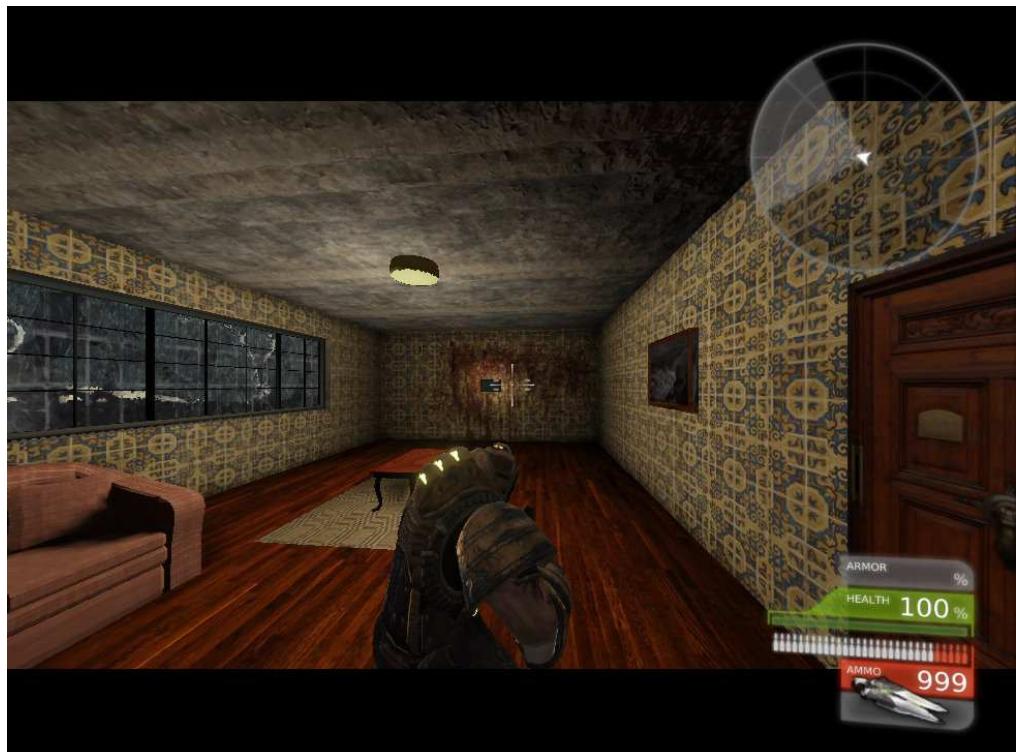
A seguir a sequência das imagens da cinematográfica do game. Note que as cenas mostradas refletem o conceito das sequências de animação, e podem não conter as últimas versões dos cenários e dos personagens do jogo.



**Figura 7.2 – Momento que a porta da mansão se abre**



*Figura 7.3 – Personagem percorrendo o corredor*



*Figura 7.4 – Dentro da sala do puzzle*



Figura 7.5 – O quadro de energia explode e a luz se apaga

## 8 Criação dos Cenários

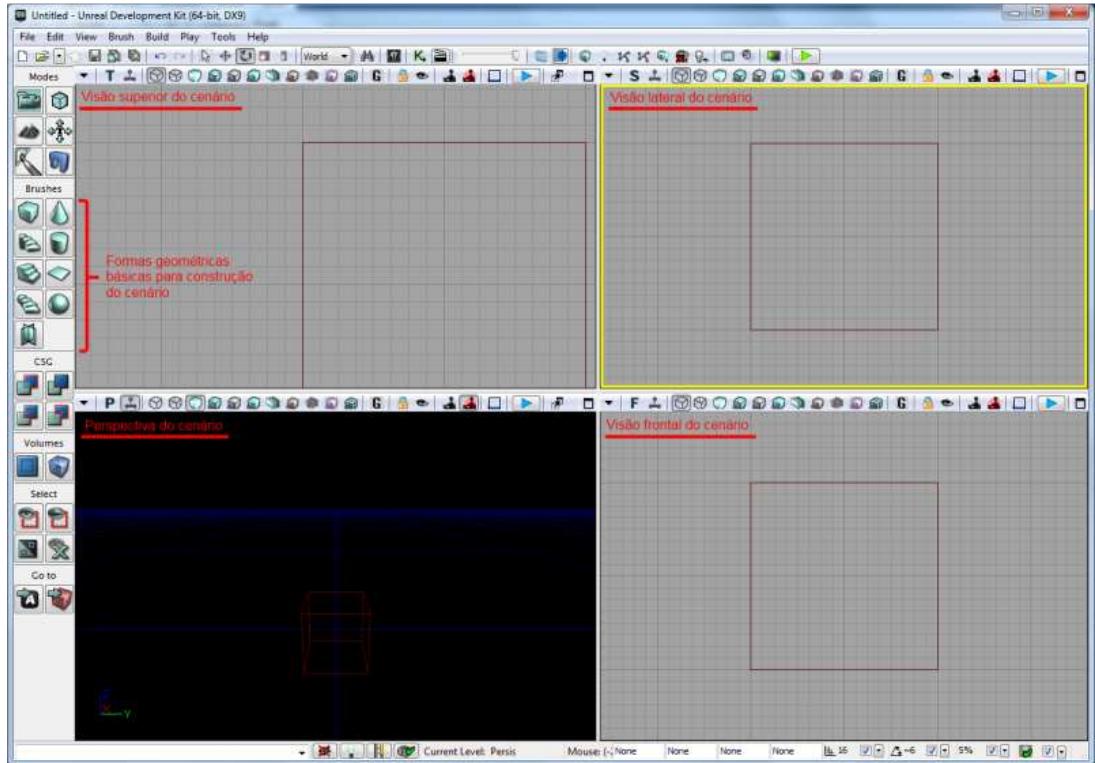
O cenário básico do jogo foi criado com base nas plantas baixas e descrições disponíveis no GDD. O cenário é constituído da sala onde o jogador começa a controlar o personagem principal, que é Fase 1, o Corridor e Main Hall que constituem a Fase 2, a Library que é o cenário para a Fase 3, e um ambiente externo à mansão onde o personagem encontra-se durante a animação inicial do jogo e que também pode visualizar ao olhar por algumas janelas. Os tópicos abaixo descrevem como estes ambientes foram criados.

### 8.1 Cenário base

A definição de cenário base neste tópico trata-se do cenário ainda sem objetos de decoração ou interação com o personagem, porém já com o acabamento em paredes, chão e teto. Veja a seguir como este cenário foi criado.

### 8.1.1 Criação do cenário básico no UDK

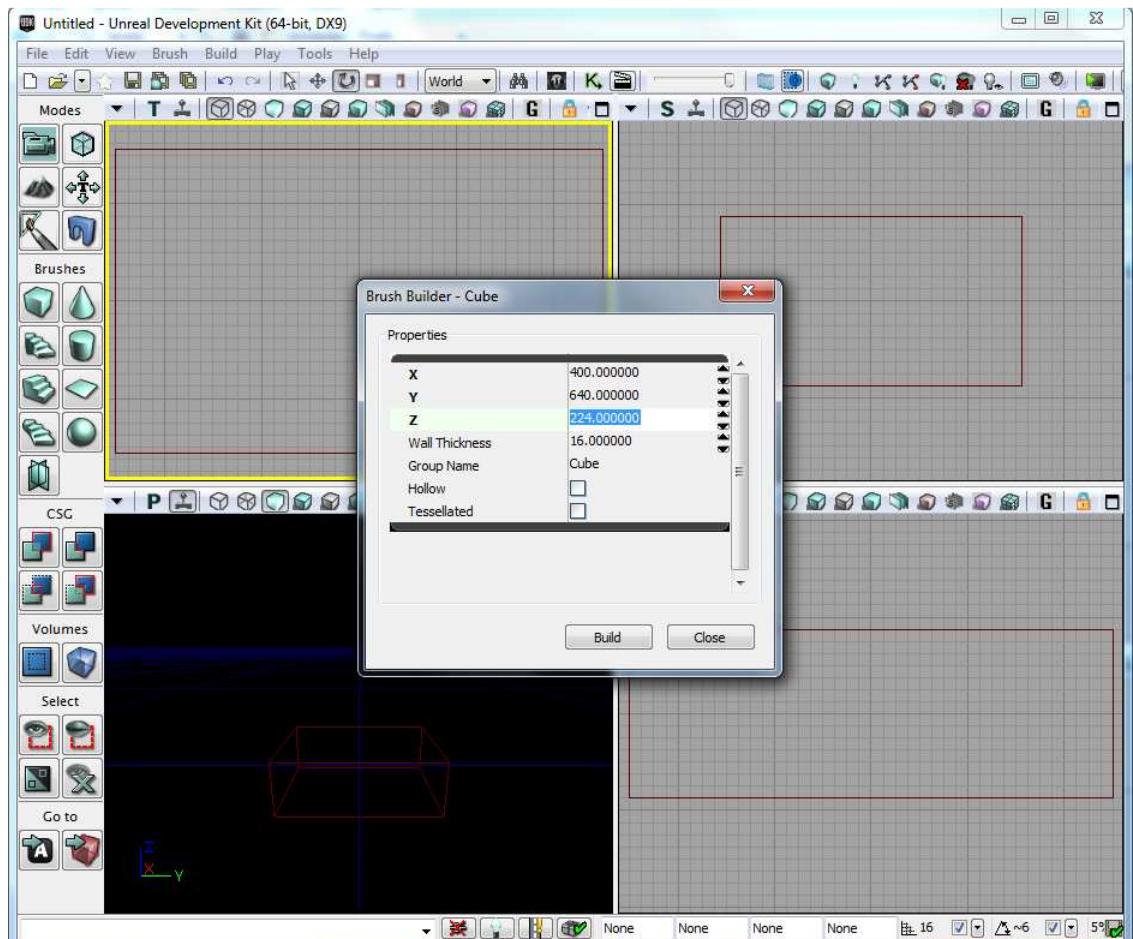
Ao abrir o UDK, é possível visualizar ao lado esquerdo da Engine algumas formas geométricas básicas que podem ser utilizadas para construir o cenário, conforme demonstrado na figura abaixo:



**Figura 8.1 – Interface do UDK e formas geométricas básicas para construção de cenário**

Tais formas geométricas são os “pincéis” que podem ser utilizados para então criar as formas desejadas. A grande maioria do cenário foi construída utilizando cubos, tendo suas formas alteradas conforme a necessidade. Na criação do quarto, por exemplo, o procedimento foi o seguinte:

1. Ao clicar com o botão direito sobre o pincel cubo, é possível determinar a dimensão do objeto. Cada unidade do UDK correponde a cerca de 2 cm do mundo real, e para representar a sala foi definido um tamanho próximo a 13 m de comprimento por 8 m de largura e 4,5 m de altura. No UDK, foi decidido utilizar unidades sempre múltiplas de 16, pois isto facilita a edição, já que o UDK disponibiliza uma grade para os objetos adicionados e um alinhamento a esta grade que permite manipular o posicionamento e tamanho dos objetos de forma mais simples. Portanto, as unidades do UDK para o quarto foram de 640 por 400 por 224, conforme ilustrado abaixo:



*Figura 8.2 – Definição do cubo que representará a sala*

2. Após o posicionamento do objeto no cenário, foi utilizada então a ferramenta de adição de objetos ao cenário, para que o pincel propriamente adicione a forma geométrica, conforme imagem abaixo:

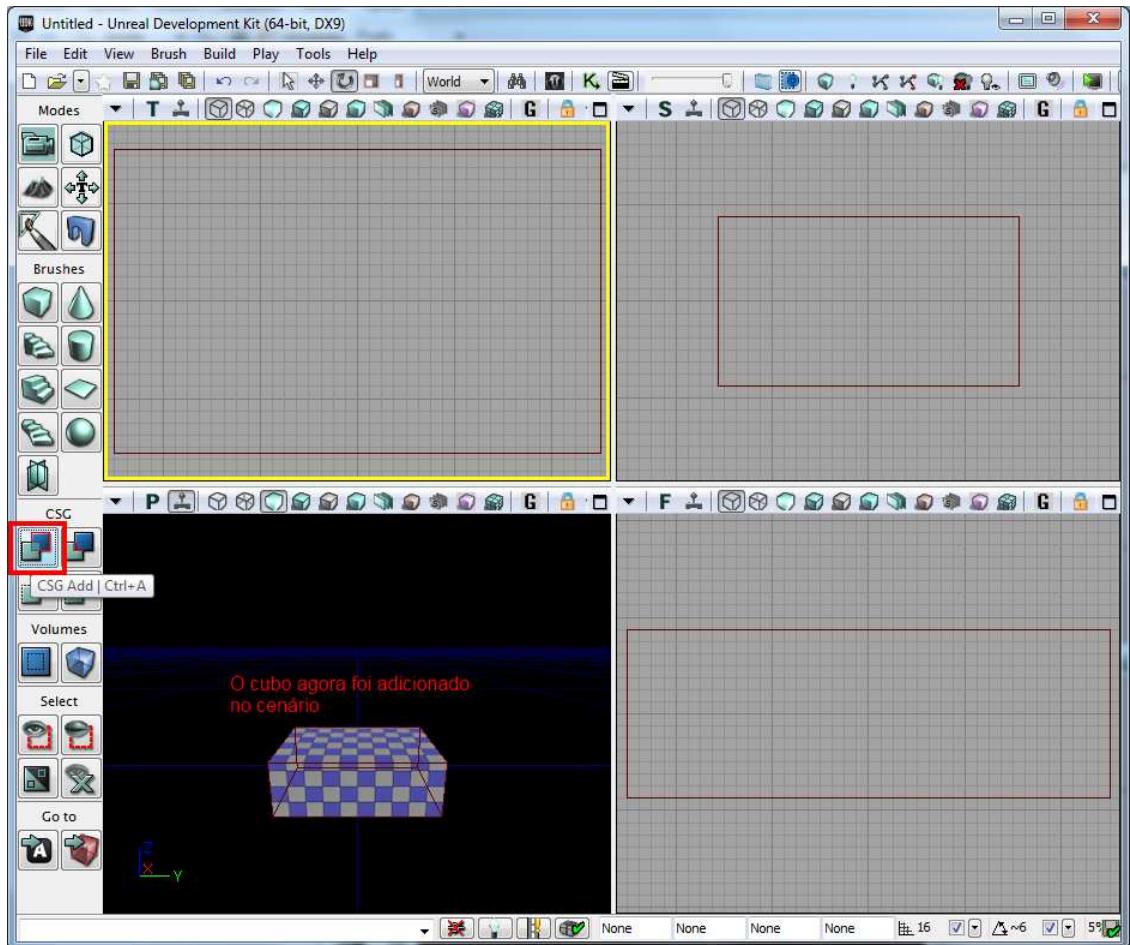
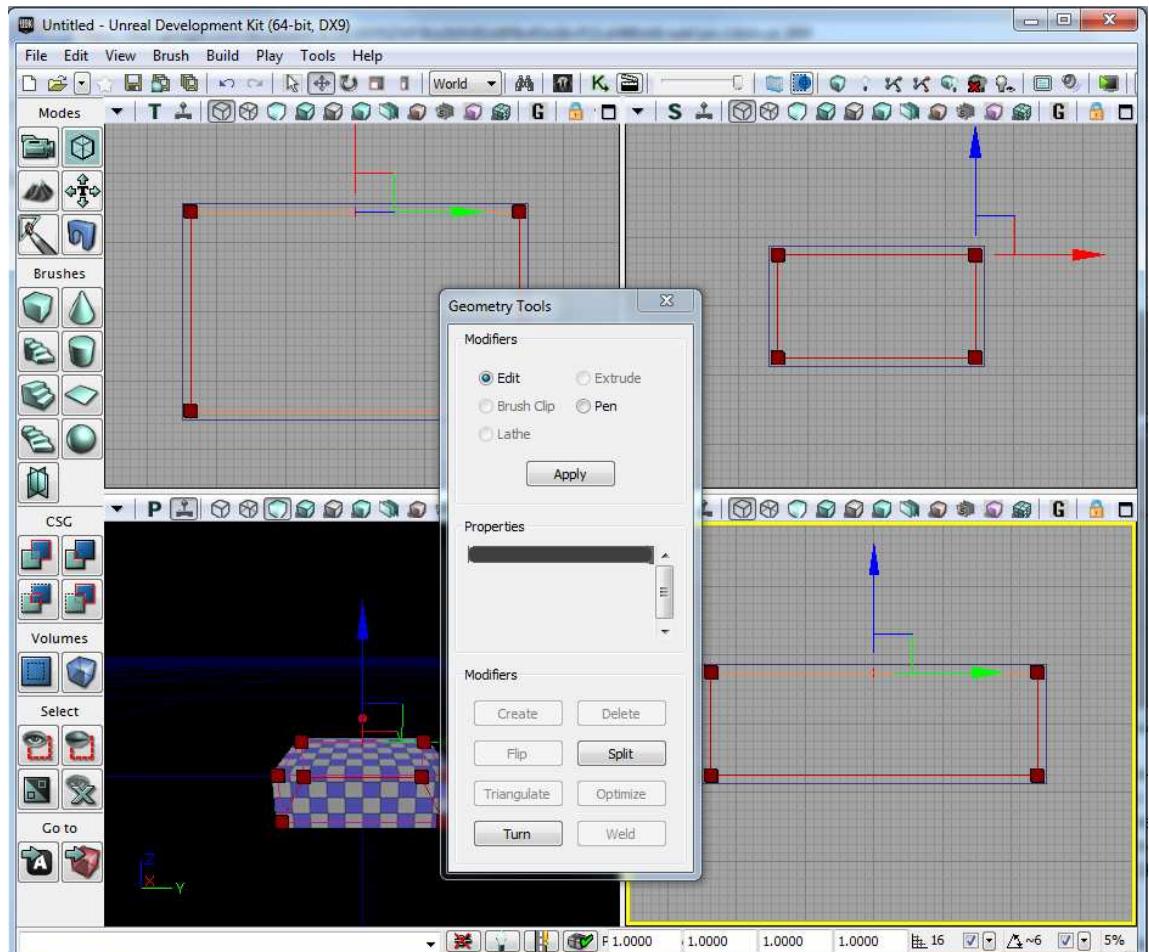


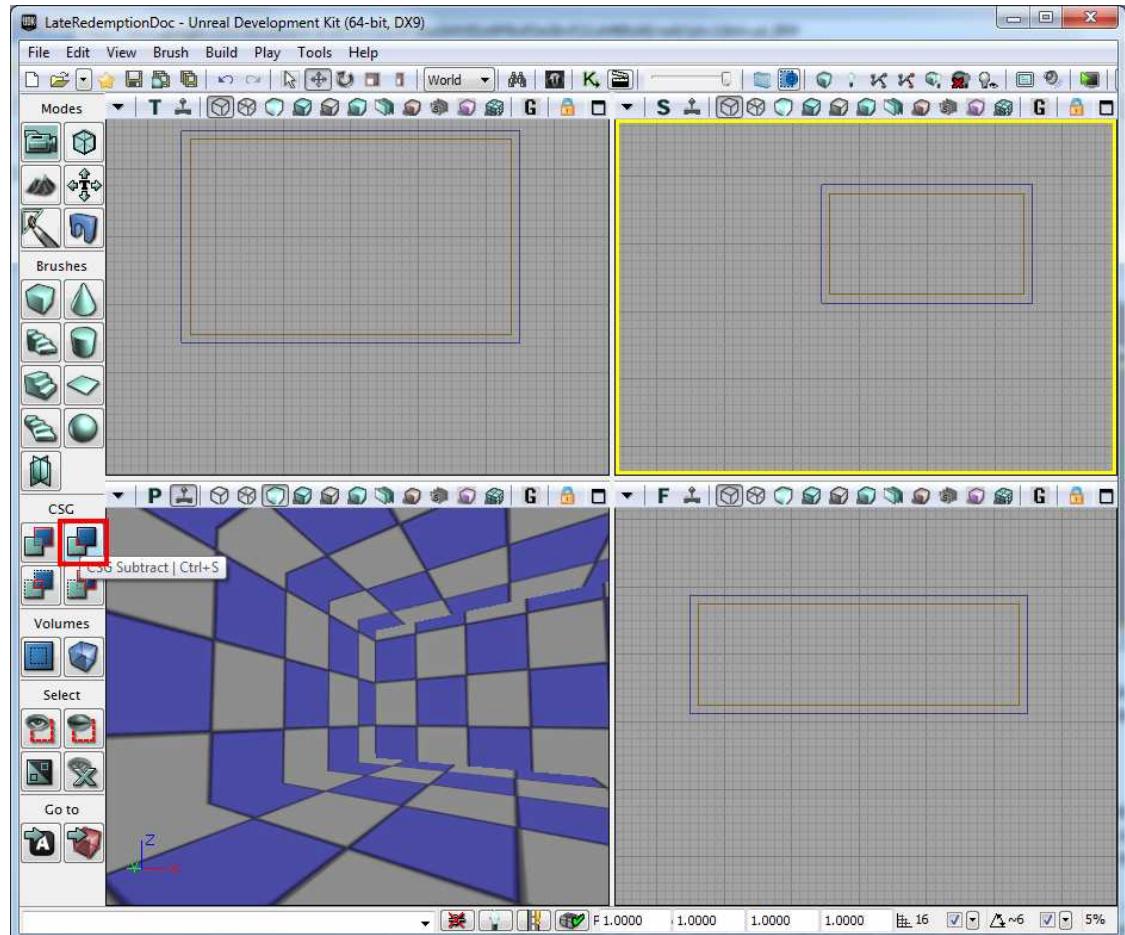
Figura 8.3 – Adicionando o cubo que representa a sala ao cenário

3. Neste momento, o objeto adicionado é um cubo totalmente sólido, e torna-se necessário deixá-lo oco por dentro, já que o espaço interno é o interior da sala por onde o personagem poderá andar. Para esta operação o pincel de cubo que já está na tela é redimensionado para que suas unidades X, Y e Z fiquem 16 unidades menores. Utilizando a ferramenta de edição geométrica e as visões Front e Top do cenário disponibilizadas no UDK, o objeto é redimensionado e posicionado dentro do cubo maior, conforme a imagem abaixo. Note que o cubo vermelho está posicionado dentro do cubo azul em todas as perspectivas:



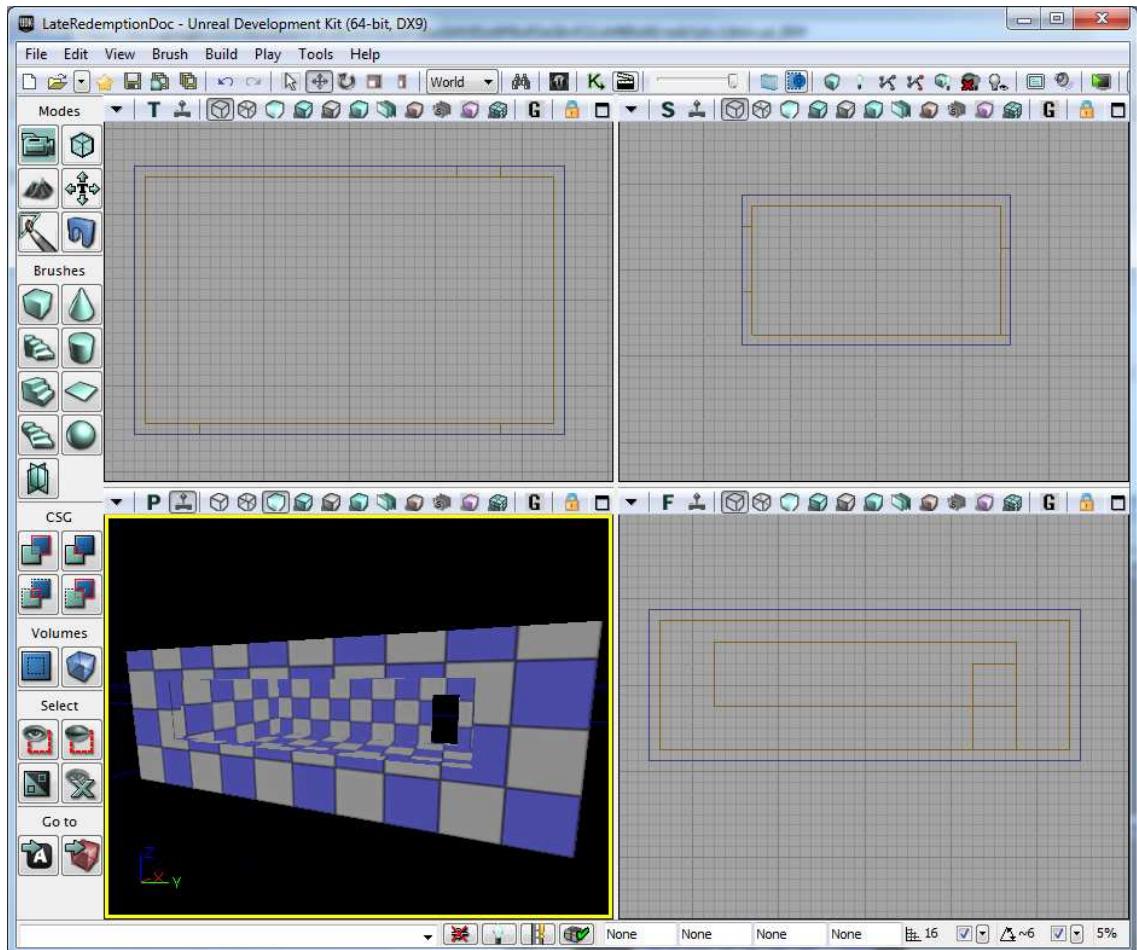
**Figura 8.4 – Utilização das Geometry Tools para criar as paredes internas da sala**

4. Desta vez, a ferramenta de subtração de objetos será utilizada ao invés da ferramenta de adição, já que o objetivo é remover a área interna do cubo. O resultado é demonstrado na imagem abaixo. O traço amarelo nas visões Top, Side e Front representam uma subtração, e a visão Perspective está posicionada dentro do cubo:



**Figura 8.5 – Visualização das paredes internas da sala nas 4 perspectivas**

5. Os últimos elementos faltantes no cenário então são as cavidades nas janelas para定位 a porta e a janela. Para criar estas cavidades, também foi utilizada a ferramenta de subtração, antes redimensionando os objetos para que eles tenham o tamanho desejado:

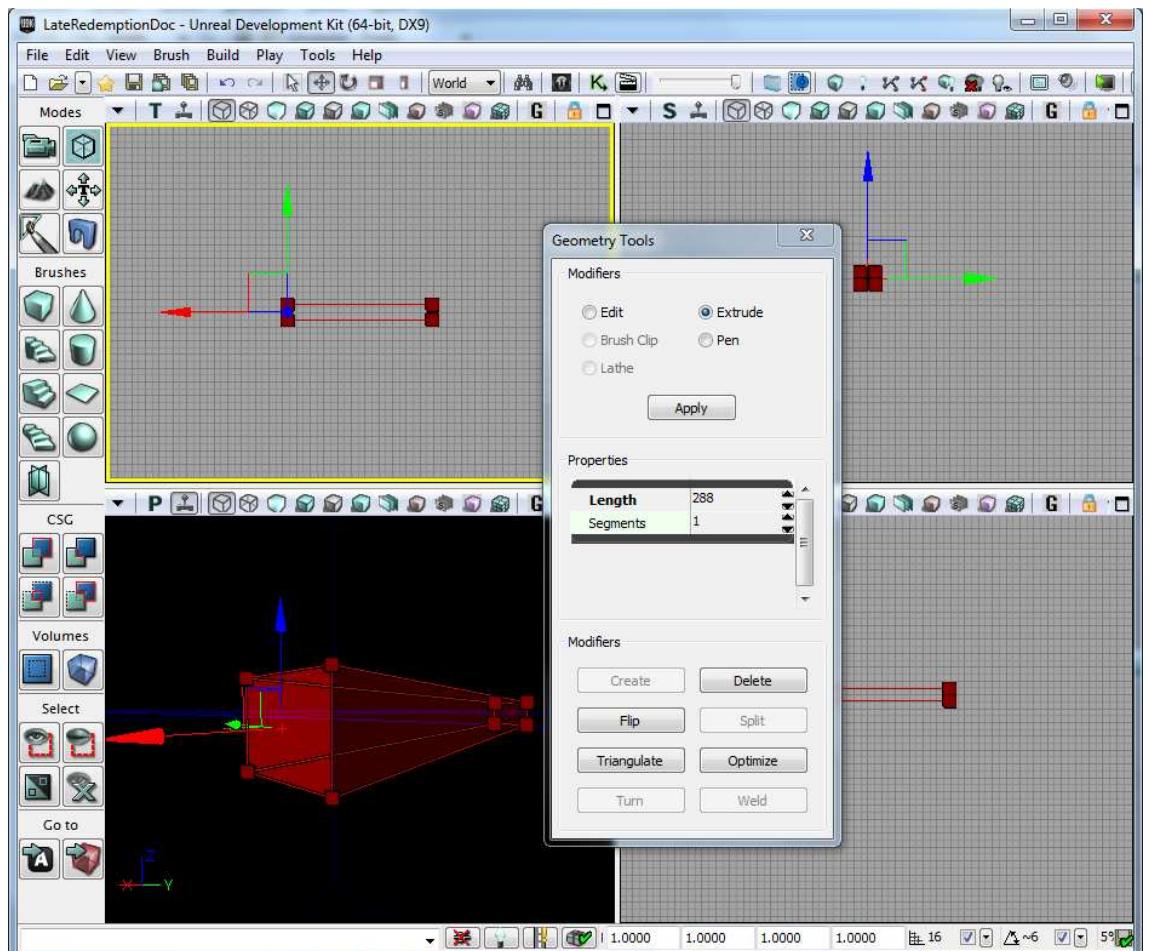


*Figura 8.6 – Sala contendo agora as cavidades para janela e porta*

O cenário então possui todas as suas formas geométricas criadas de acordo com a especificação do GDD. No próximo tópico, este mesmo exemplo do quarto será utilizado para entender como é feita a aplicação de texturas, e mais em seguida, também a criação da janela e da porta.

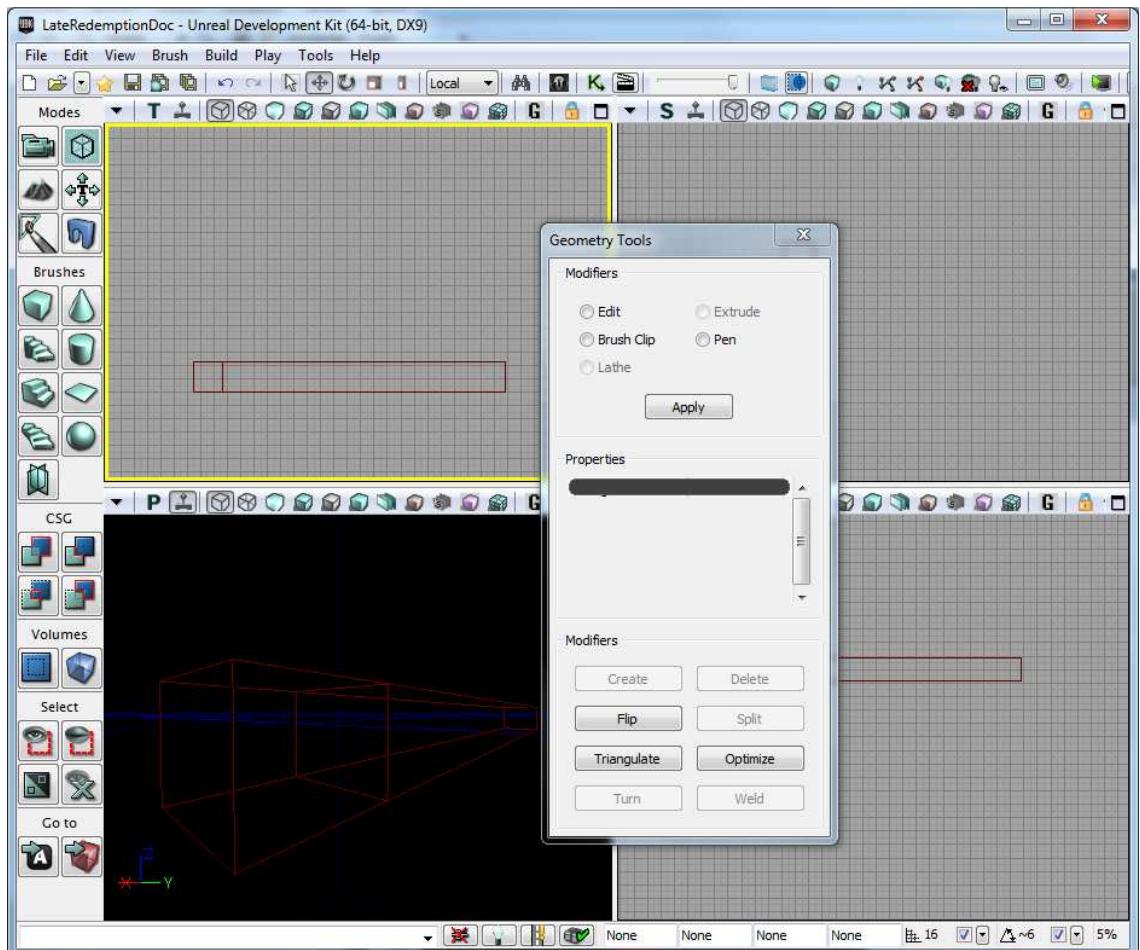
Tanto o Corridor, como o Main Hall e a Library tiveram um processo semelhante de criação. Algumas diferenças no processo de criação são citadas abaixo.

O Corridor é constituído por um único objeto geométrico em forma de “U”. Para criar este objeto, foi utilizada a ferramenta geométrica Extrude. Para utilizar o Extrude, o procedimento a seguir pode ser feito. Com o pincel de cubo na tela e a dimensão de uma dos lados já definidos, a ferramenta de edição geométrica é selecionada. Em seguida, a face do polígono que deve ser estendida é selecionada, conforme a imagem abaixo:



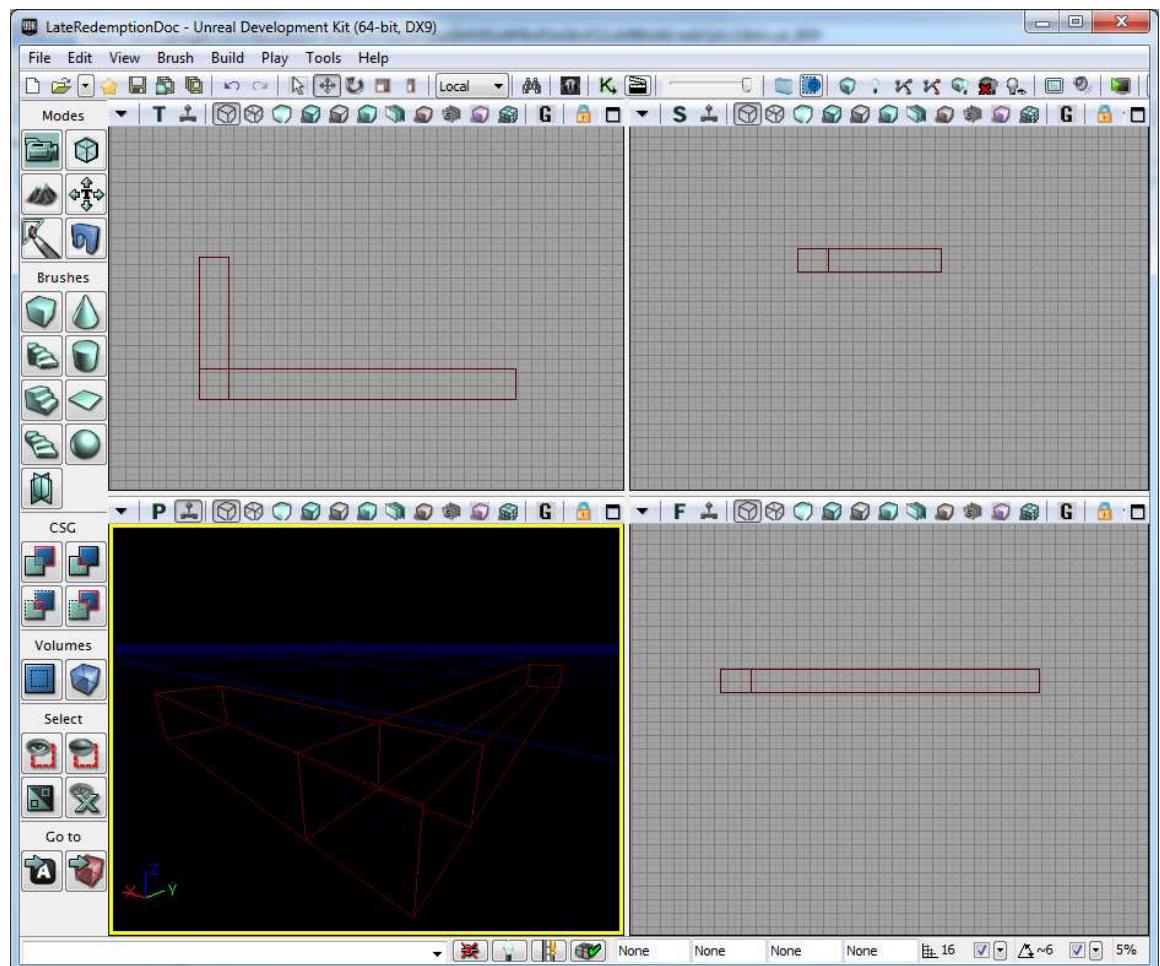
**Figura 8.7 – Utilização de Extrude para fazer as curvas do corredor**

Na visão Perspective da imagem acima, note que uma das faces está com a cor mais destacada, sendo esta a fase selecionada. Além disso, a opção Extrude está marcada em Geometry Tools. O próximo passo então é definir a quantidade de segmentos de objetos que será criada e o tamanho do segmento. O objetivo do Extrude neste momento é criar a curva do corredor, então apenas um segmento será criado e que tenha a forma de um cubo da mesma largura do corredor, sendo que o resultado é ilustrado na figura a seguir:



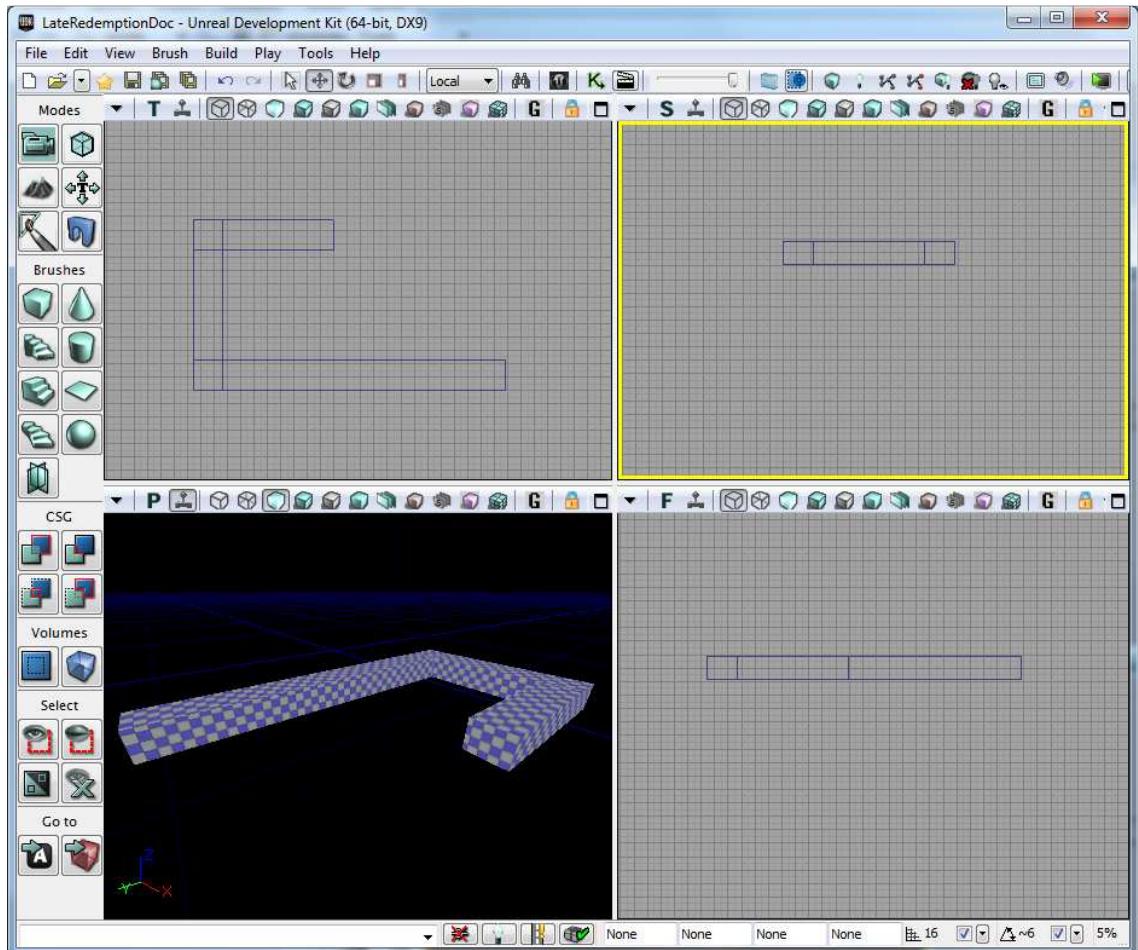
*Figura 8.8 – Utilização de Extrude para fazer as curvas do corredor*

Um novo Extrude é então executado para criar a próxima parte do corredor:



*Figura 8.9 – Utilização de Extrude para fazer as curvas do corredor*

O mesmo processo é feito então para a última parte. A ferramenta de adição então é utilizada para adicionar a forma geométrica básica na tela:



*Figura 8.10 – Objeto que representa o corredor adicionado ao cenário*

Para criar as paredes, é feito o mesmo processo da sala inicial, ou seja, o pincel é redimensionado para ficar um pouco menor em todas as direções, posicionado dentro do objeto maior que representa o corredor e então sua forma é substraída da forma existente.

A forma da Library é semelhante a um semi-círculo. Para criar este objeto, foi utilizado um pincel de cilindro a a ferramenta de edição geométrica Delete. Primeiramente, utilizando o pincel de cilindro redimensionando-o na largura da Library. No Geometry Mode, todos os vértices de um lado do cilindro são selecionados pela visão Top, como demonstrado abaixo:

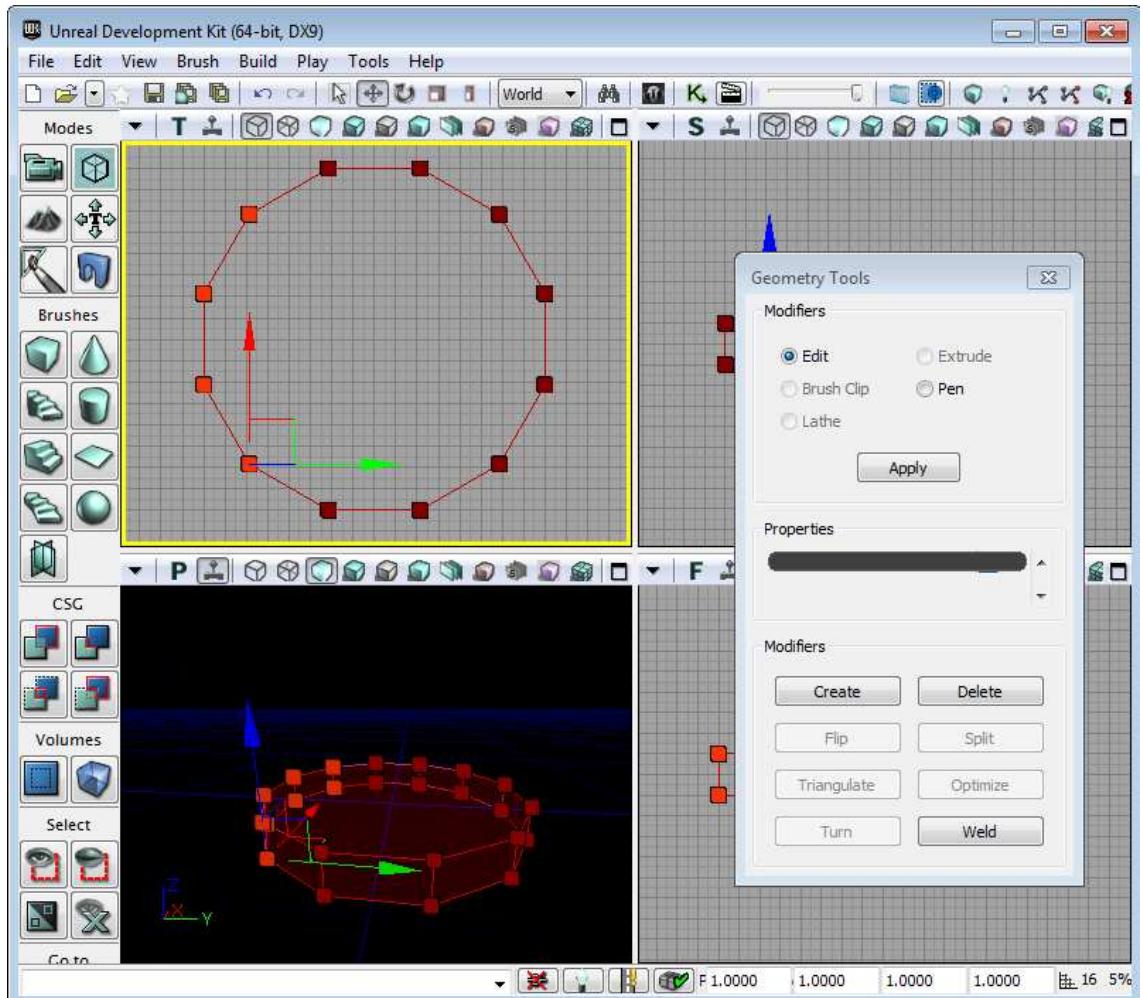
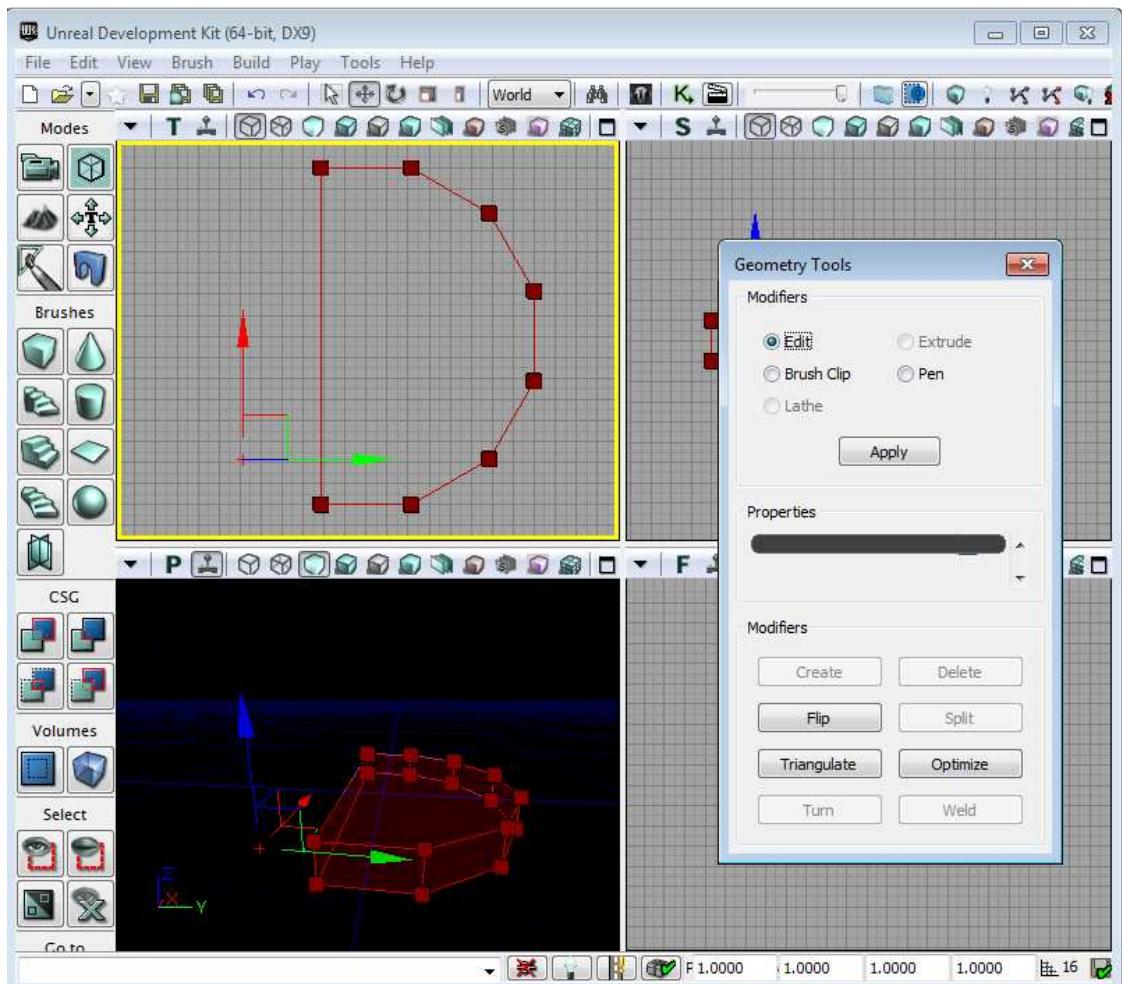


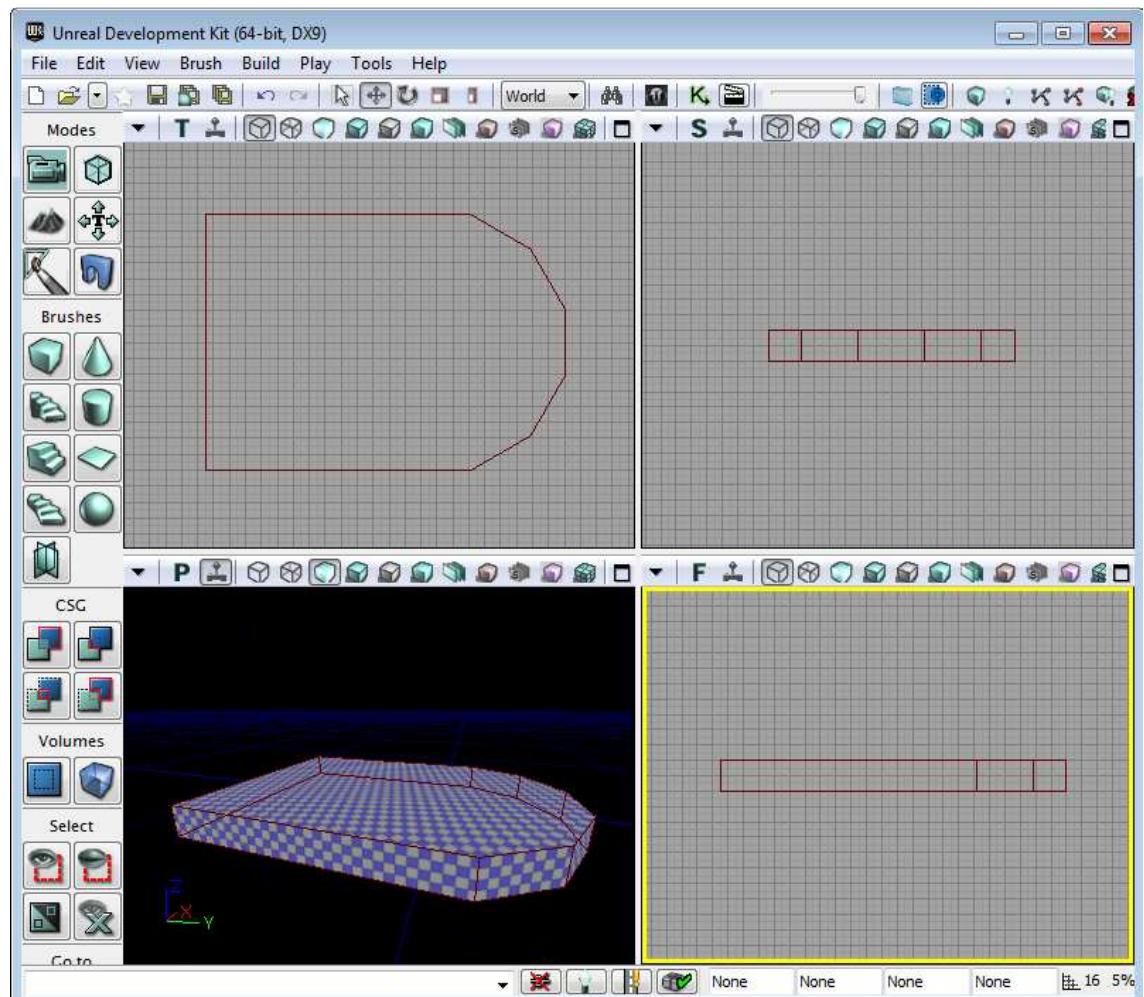
Figura 8.11 – Utilização de cilindro para criar a Library

Em seguida opção Delete é utilizada para excluir estes vértices e então criar um semi-círculo, conforme as imagens abaixo:



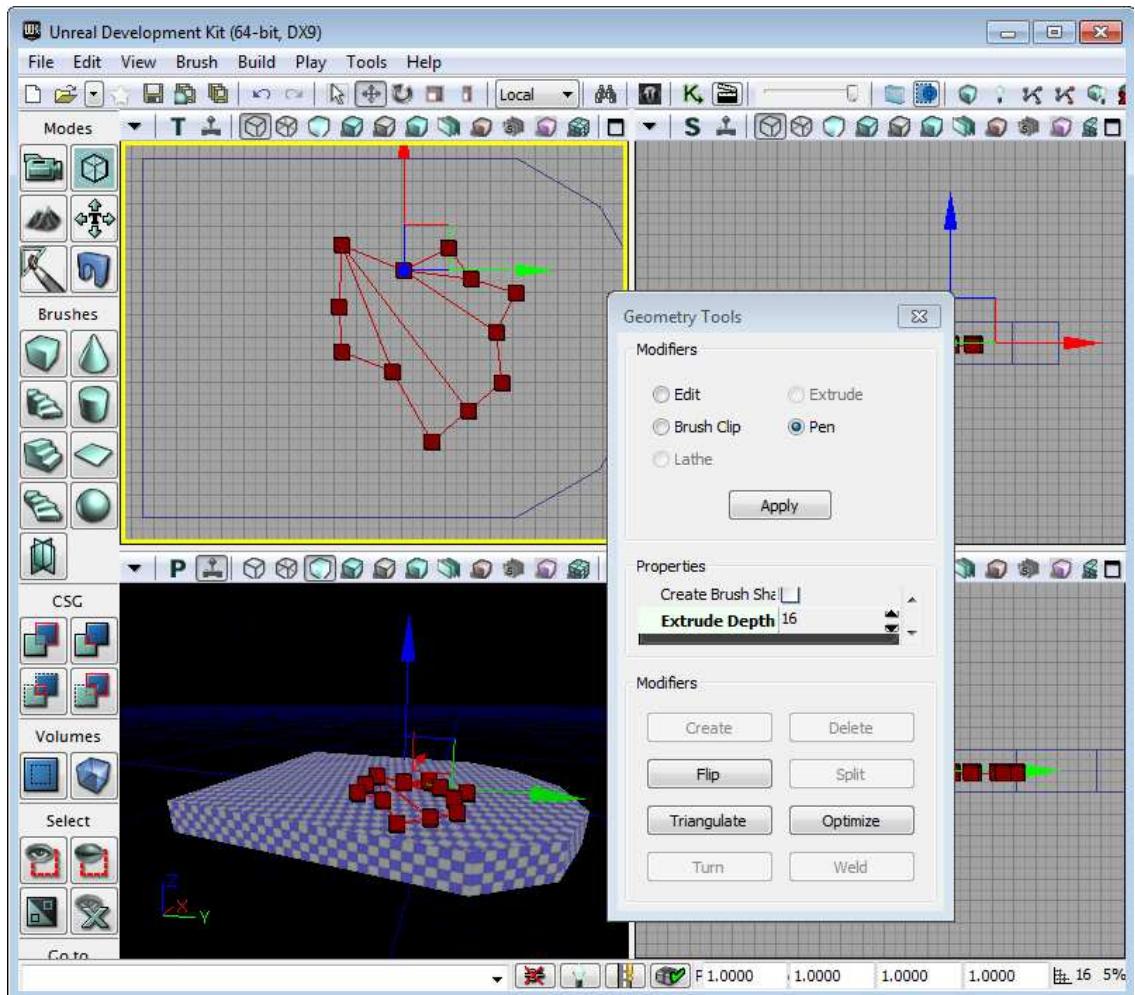
**Figura 8.12 – Cilindro com alguns vértices excluídos para representar a Library**

A extremidade reta do semi-círculo é então aumentada para obter a forma geométrica esperada para a Library:



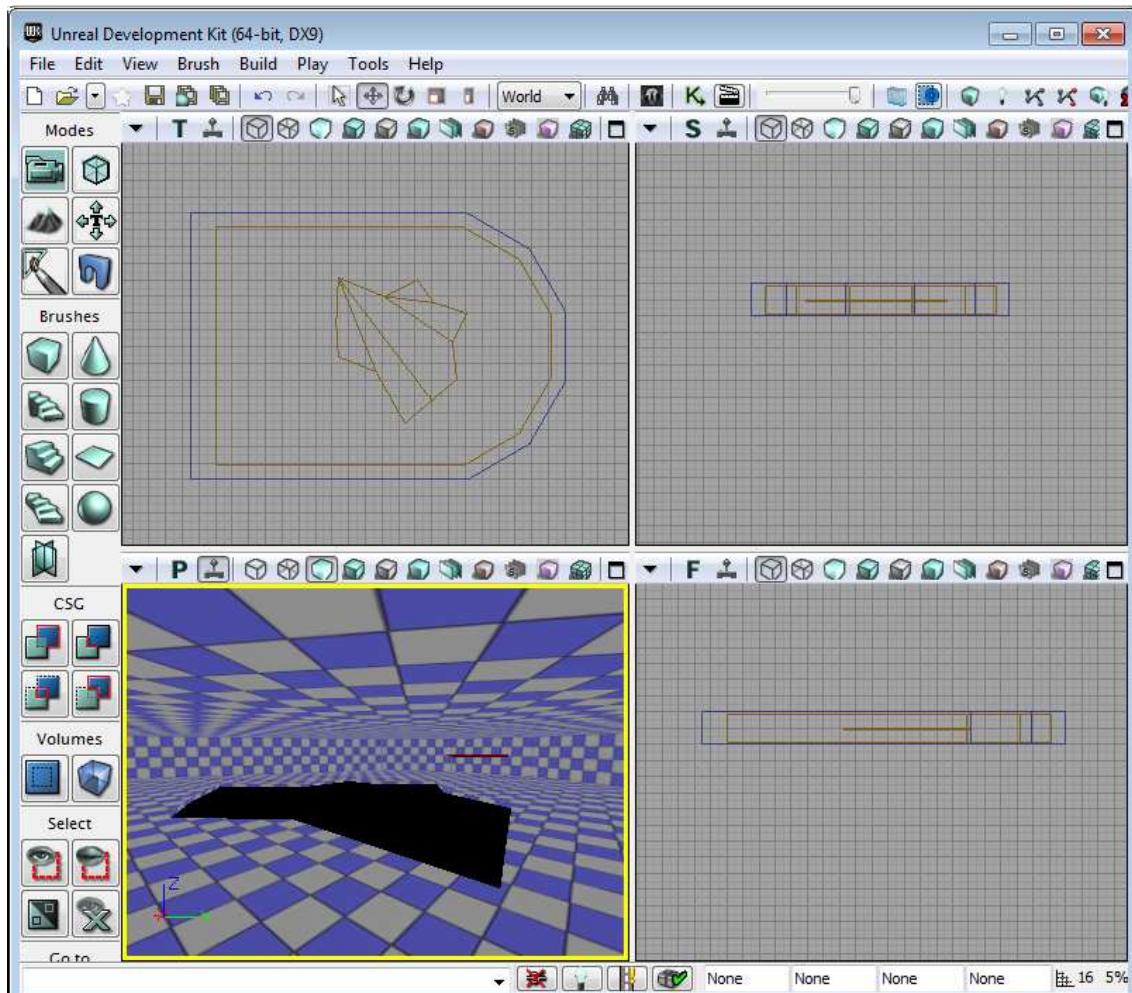
**Figura 8.12 – Objeto no formato da Library**

Ainda na library, há um buraco no chão que é onde Diabolus, o chefe final do jogo estará. Para criar este buraco, foi utilizada a ferramenta Pen, que permite criar um objeto formas geométricas irregulares como, por exemplo, na figura abaixo:



**Figura 8.13 – Criação do buraco onde o Diabolus estará situado**

O objeto desenhado com a ferramenta Pen então é posicionado na altura do chão e subtraído do cenário para desenhar o buraco:



**Figura 8.14 – Buraco onde o Diabolus estará situado**

Uma vez que a estrutura base do cenário está pronta, é hora de aplicar texturas no cenário, e também adicionar elementos a ele, como portas, cadeiras, mesas e quadros. Para isso, seguem-se os seguintes passos de maneira geral:

- 1 - Seleção de textura
- 2 - Criação de textura no UDK
- 3 - Criação de material no UDK
- 4 - Aplicação de material nos objetos criados / importados
- 5 - Criação / importação de objetos / static meshes no UDK
- 6 - Adição dos objetos no cenário

### 8.1.2 Seleção de texturas

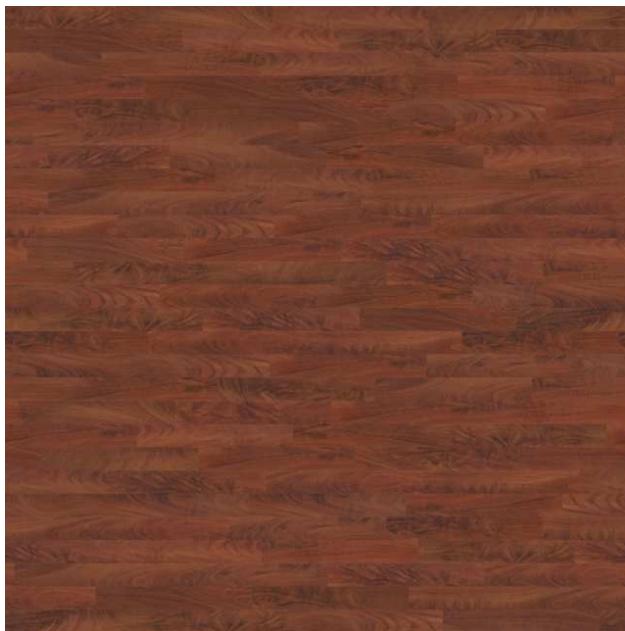
Durante o desenvolvimento do cenário, a opção foi de selecionar texturas existentes ao invés de desenhar as próprias texturas. Alguns sites foram utilizados para buscar texturas de uso livre, sendo o principal desses o site <http://www.cgtextures.com/>. O site classifica texturas por categorias tais como paredes, chão, telhado, portas, etc. Após escolher uma

textura pelo site, basta clicar na opção de download para baixar o arquivo, normalmente no formato JPEG.

### 8.1.3 Criação de texturas para o UDK

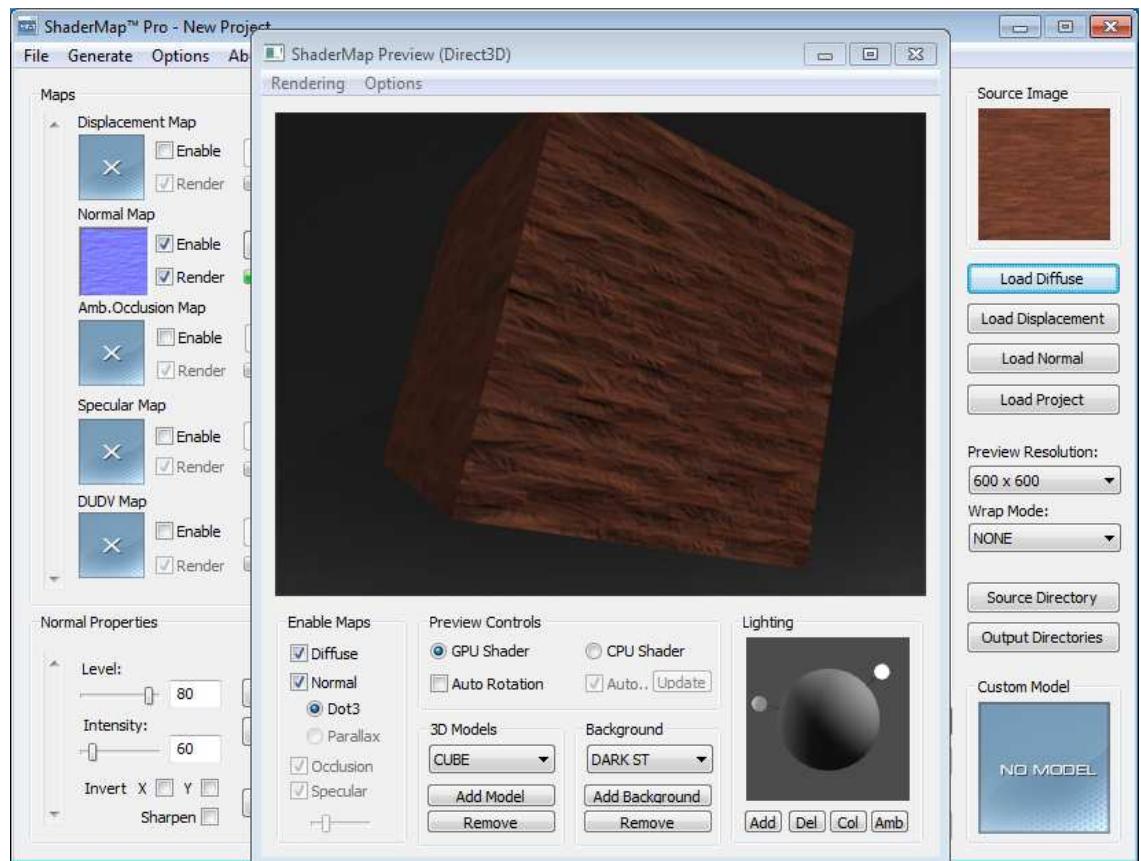
O UDK utiliza um conceito chamado Material, que nada mais é que um material que pode ser aplicado sobre uma determinada superfície para dar a esta uma determinada textura e alguns efeitos como a simulação de relevo (Normal Map) ou brilho (Specular Map). Por isso, um material depende que uma textura tenha sido criada na UDK engine, para depois ser associada a este material. A partir de agora descreveremos todos os passos envolvidos na criação do material utilizado no assoalho da sala onde Marshall Gory inicia o jogo.

1. Após a seleção da textura, encontrada no link <http://www.cgtextures.com/texview.php?id=31730>, é necessário baixar o arquivo e convertê-lo para um formato entendido pelo UDK. O formato selecionado foi o TGA. A conversão do arquivo JPEG ou PNG para TGA pode ser feita por softwares como o Adobe Photoshop ou Paint.NET. É importante também que o arquivo tenha tanto a largura como o comprimento iguais e que o valor em pixels seja uma potência de 2 (exemplo: 512 x 512), pois este é o formato aceito pelo UDK. Portanto, caso o arquivo não esteja nestas proporções, será necessário redimensioná-lo, utilizando também ferramentas como o Adobe Photoshop ou o Paint.NET. A textura do assolhado é esta demonstrada abaixo:



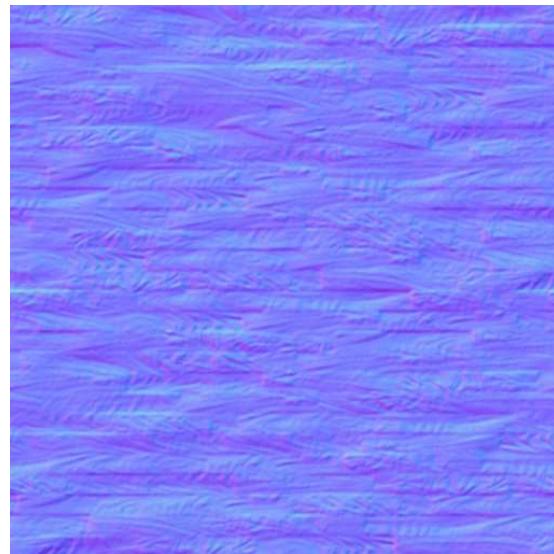
*Figura 8.15 – Textura para o assoalho da sala*

2. O material do assoalho tem um leve efeito de relevo. Para aplicar este efeito, deve ser criado um Normal Map para a textura. A criação do Normal Map pode ser feita na ferramenta ShaderMap. No ShaderMap, a imagem original pode ser carregada na opção Load Diffuse e então, após selecionar a opção Normal Map, basta acertar os parâmetro de intensidade do Normal Map. É possível solicitar um Preview 3D da imagem, onde é possível visualizar a tela a seguir:



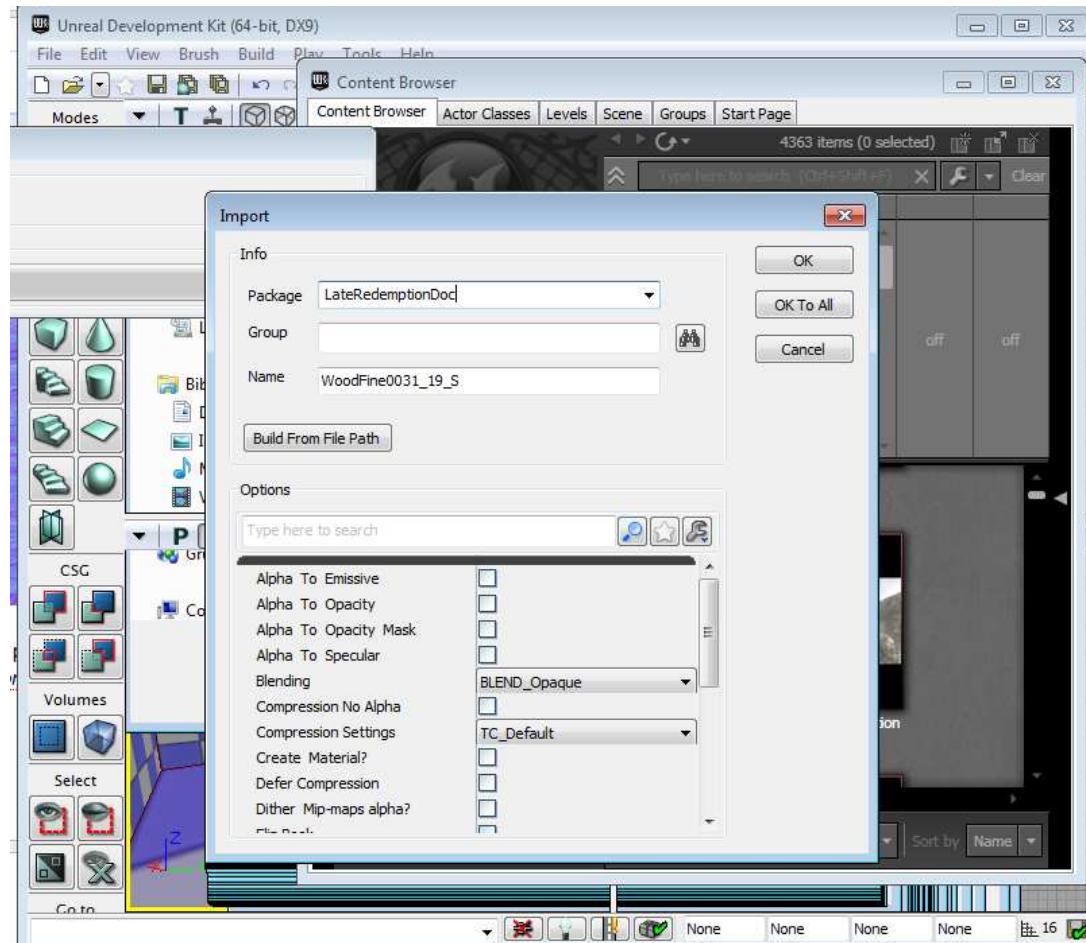
**Figura 8.16 – Visualização 3D no ShaderMap após aplicação de Normal Map**

3. Em seguida, basta clicar em Generate ao lado da opção Normal Map para salvar o arquivo no mesmo diretório da textura original. O Normal Map fica como na imagem a seguir:

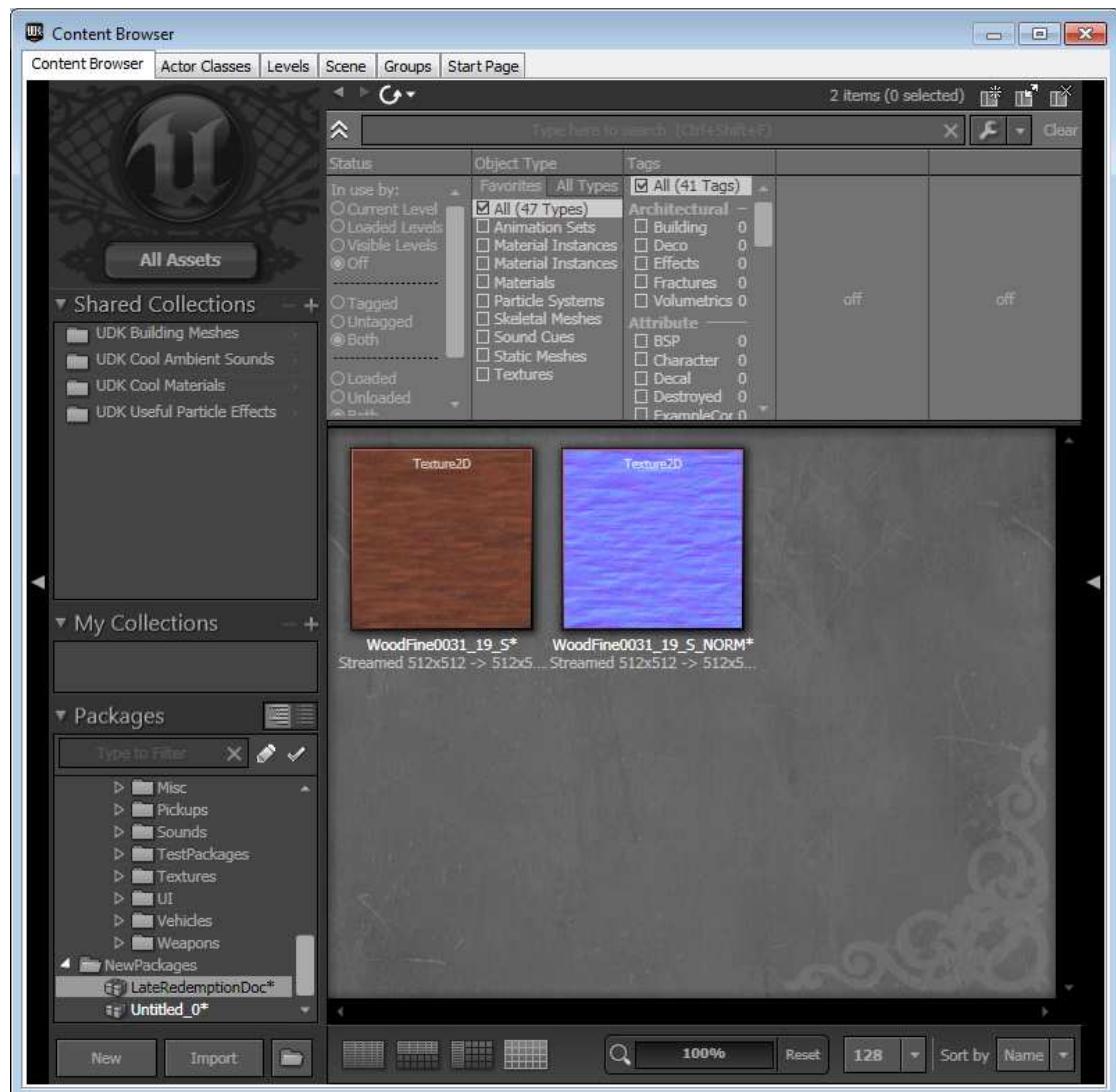


**Figura 8.17 – Normal Map do assoalho criado no ShaderMap**

4. O próximo passo é importar as texturas no UDK. Abra o Content Browser e clique na opção Import. Selecione então o arquivo da textura e do Normal Map, grave-os no package que desejar e os arquivos estarão no UDK:



**Figura 8.18 – Importação dos arquivos de textura no UDK**



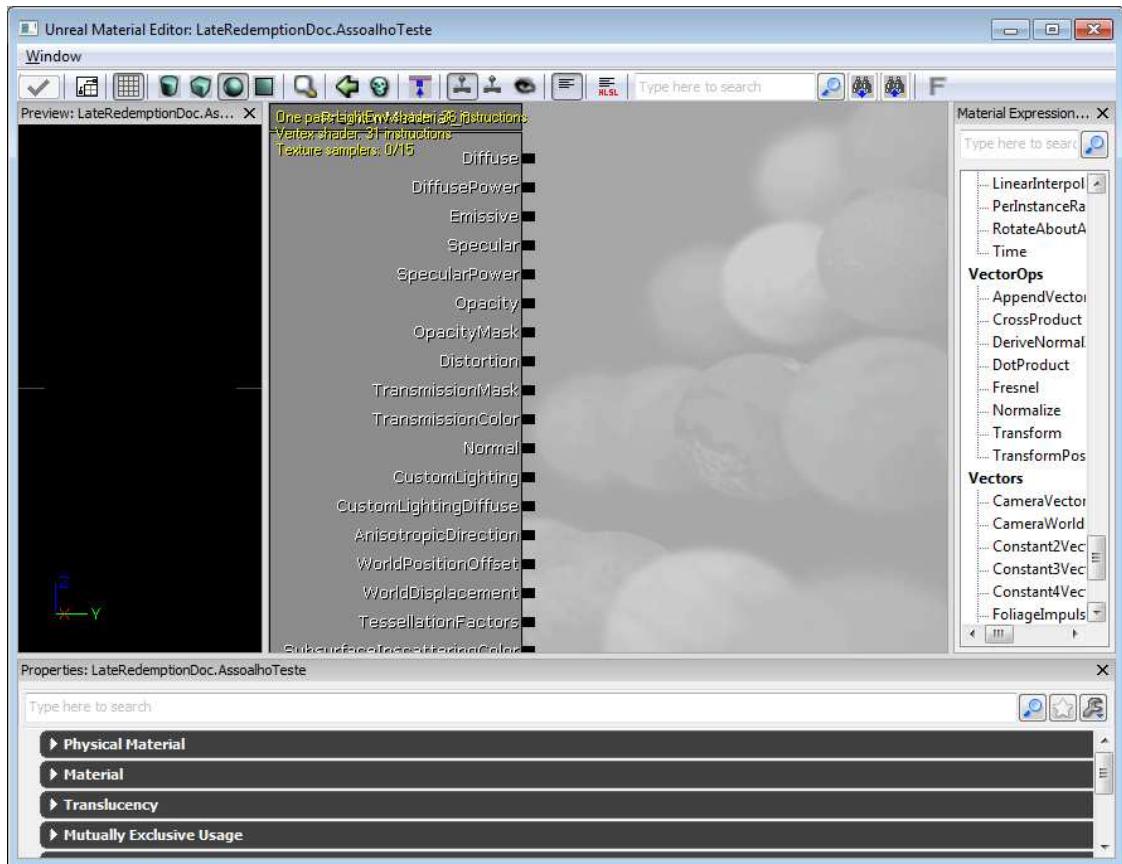
**Figura 8.19 – Texturas importadas no UDK**

5. Salve o package criado.

#### 8.1.4 Criação de um material para o UDK

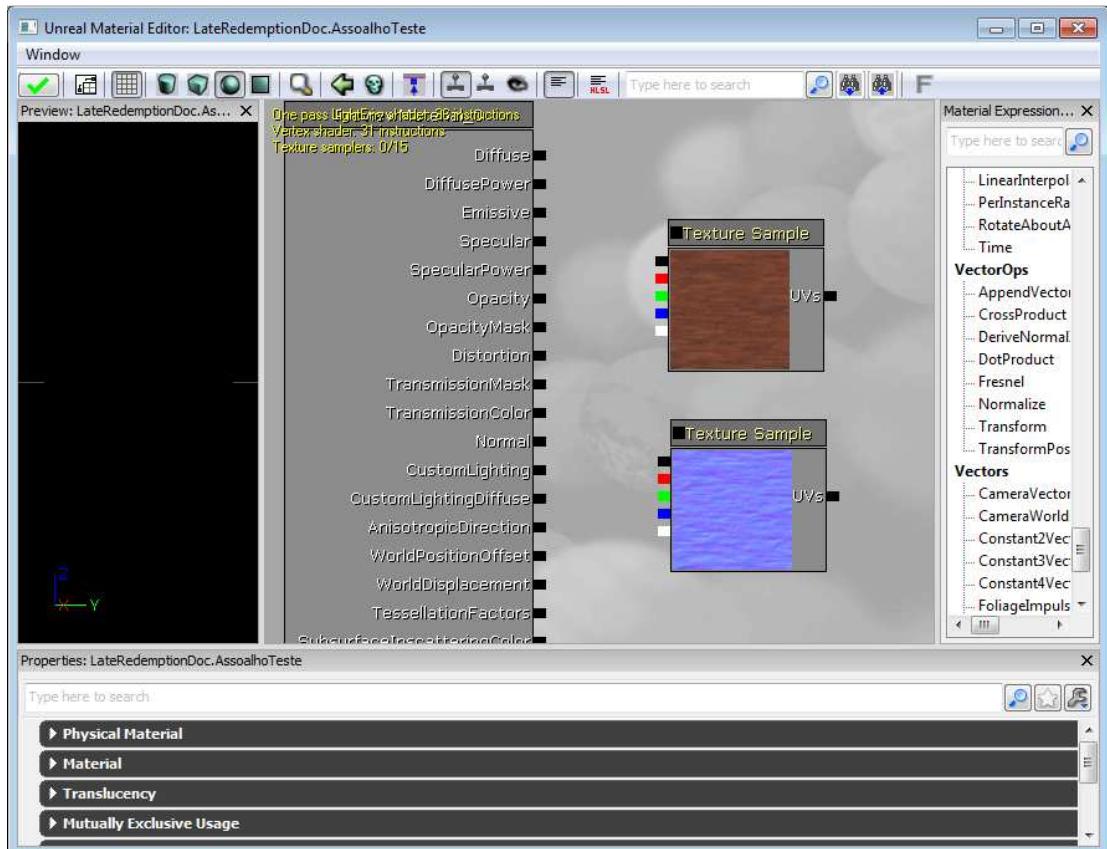
Com a textura criada no UDK, agora precisamos criar um material e associá-lo a esta textura.

6. Crie dentro de algum package um novo material. Para isso, clique com o botão direito na região central do Content Browser e selecione a opção New Material. Coloque o nome do material, clique em OK e salve o package. A tela de criação de material será aberta:



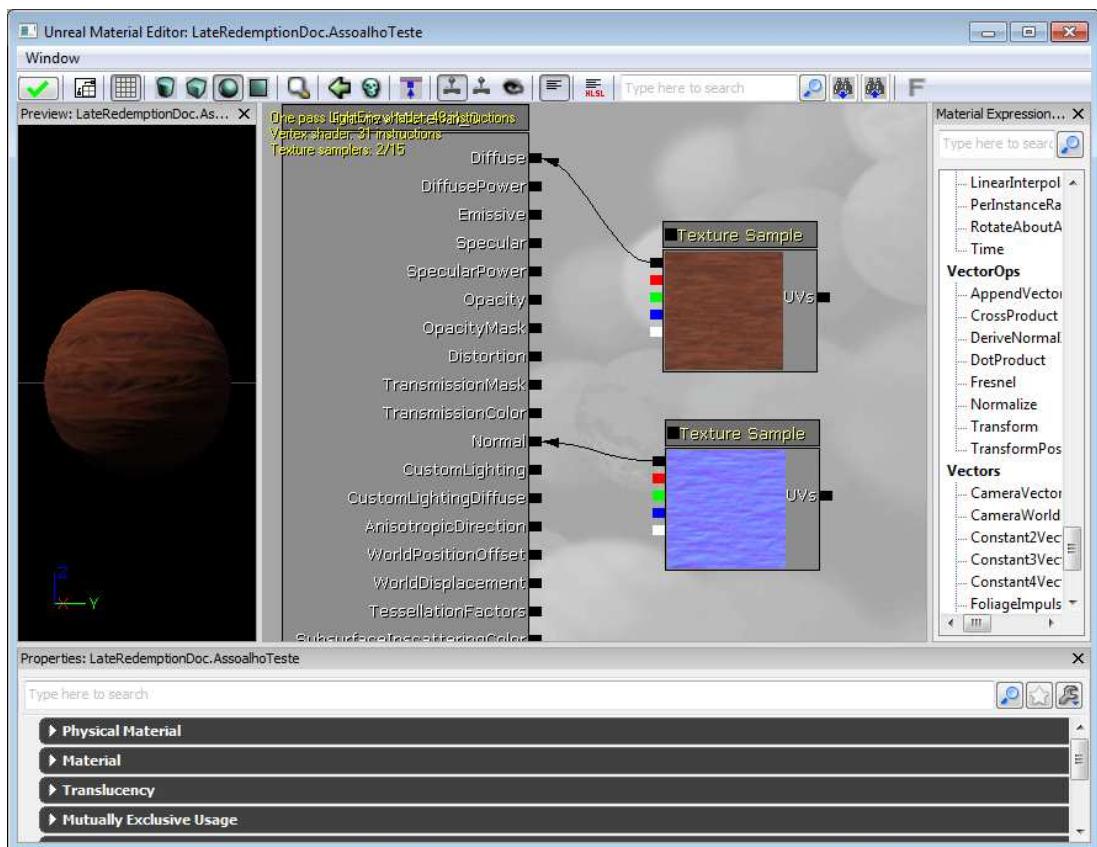
**Figura 8.20 – Tela de edição de material no UDK (Unreal Material Editor)**

7. Selecione as texturas no Content Browser e arraste-as para a área central da tela de criação de material:



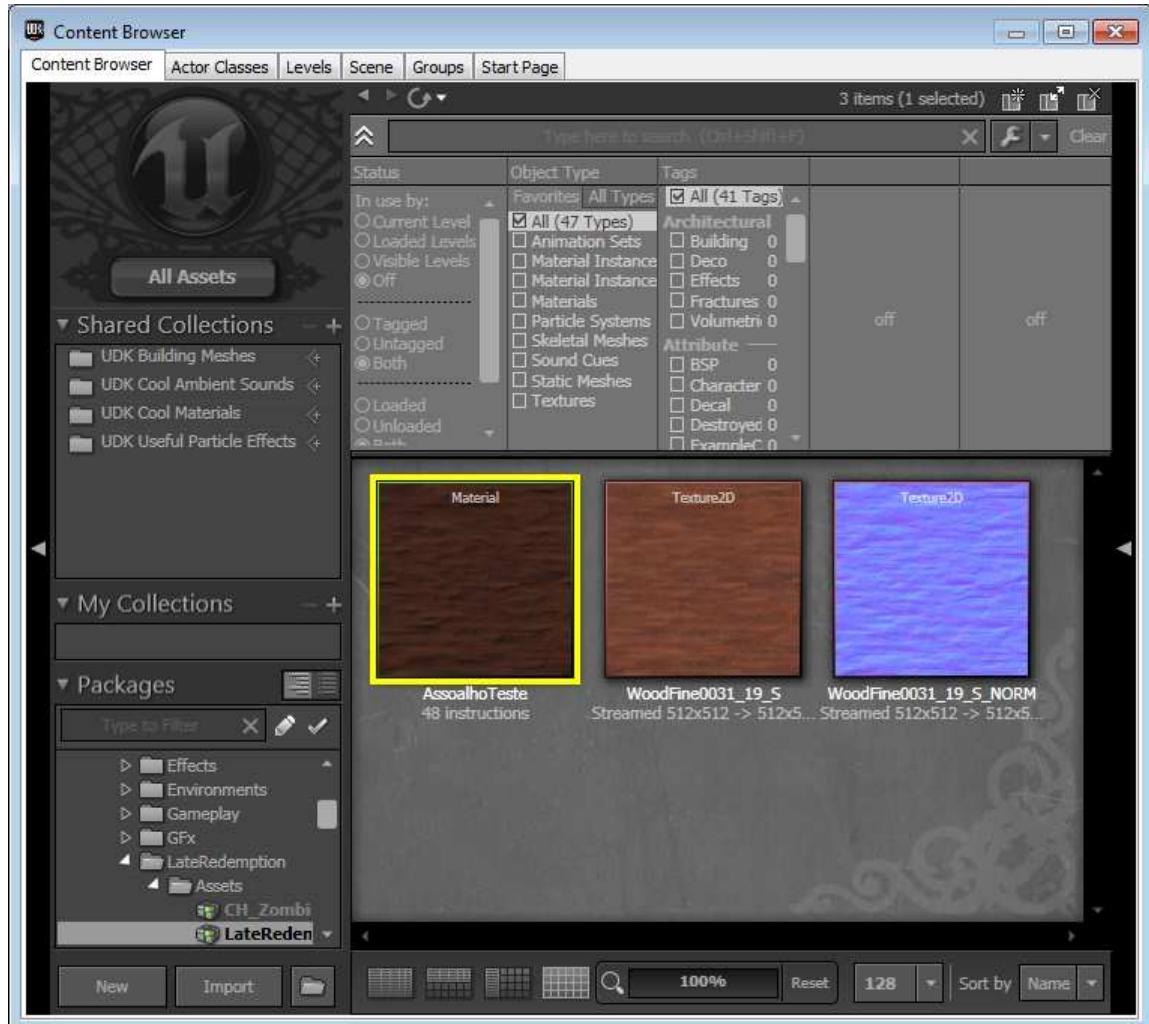
**Figura 8.21 – Texturas adicionadas no novo material criado**

8. Agora basta ligar a opção Diffuse com a textura e a opção Normal com o Normal Map:



**Figura 8.22 – Relacionamento das texturas com as propriedades do material**

9. Clique no ícone de compilação do material (o ícone verde no canto superior esquerdo da tela) e o material será alterado. Salve o package e o material estará criado. Na imagem abaixo, material criado está marcado com uma borda amarela em sua volta:



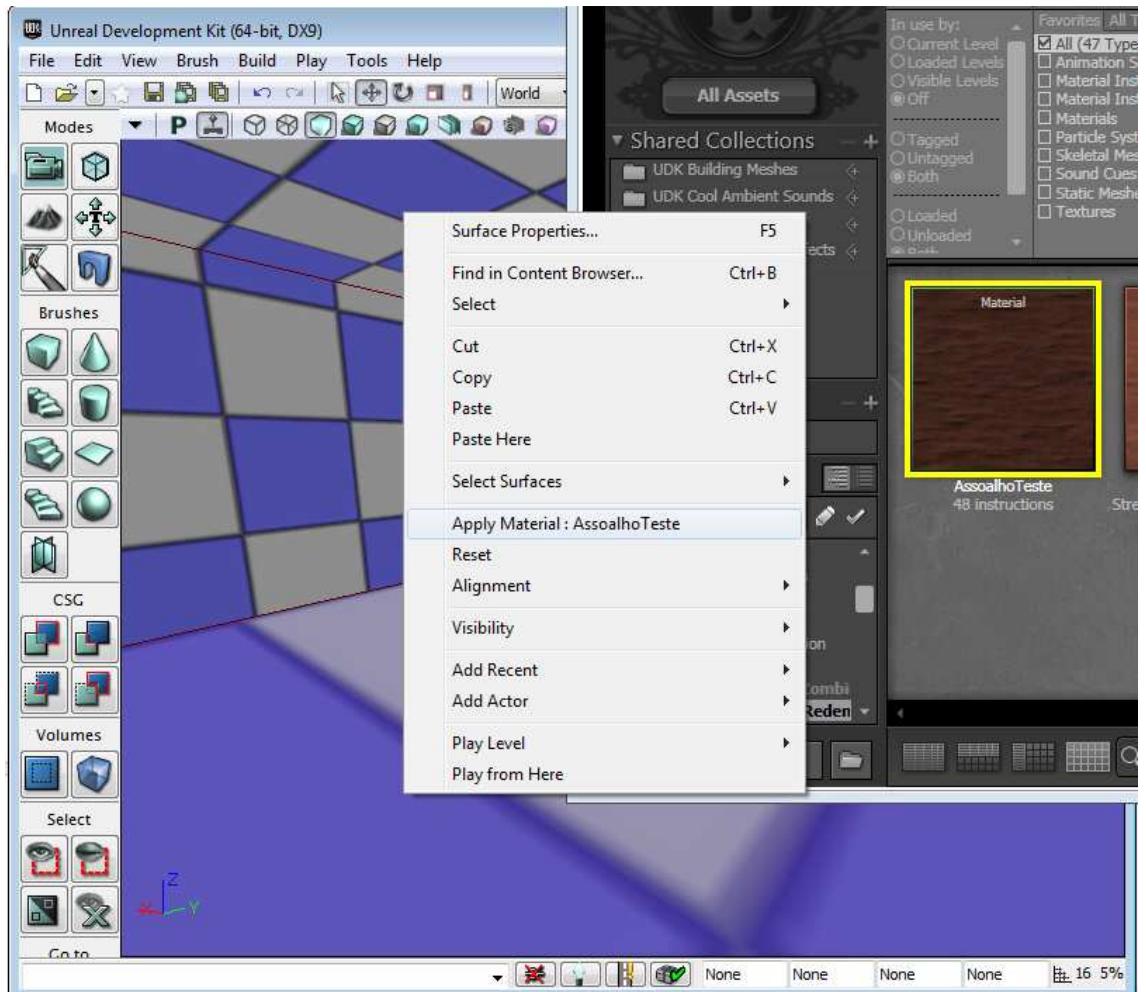
*Figura 8.23 – Visualização do material criado no Content Browser*

### 8.1.5 Aplicação de material nos objetos criados / importados

Como explicado na seção anterior, o UDK utiliza o conceito de Material para aplicar texturas aos objetos existentes. A maioria dos materials utilizados na criação do cenário foram criados seguindo o procedimento descrito na seção anterior. Alguns outros foram materials já fornecidos pelo próprio UDK, situação em que tais materials devem ser copiados para dentro dos packages do jogo, como boa prática para evitar problemas de referências quebradas. Em qualquer um dos casos, o procedimento para aplicar o material é o mesmo. Seguindo o exemplo do assoalho da sala, será demonstrado a seguir como aplicar o material.

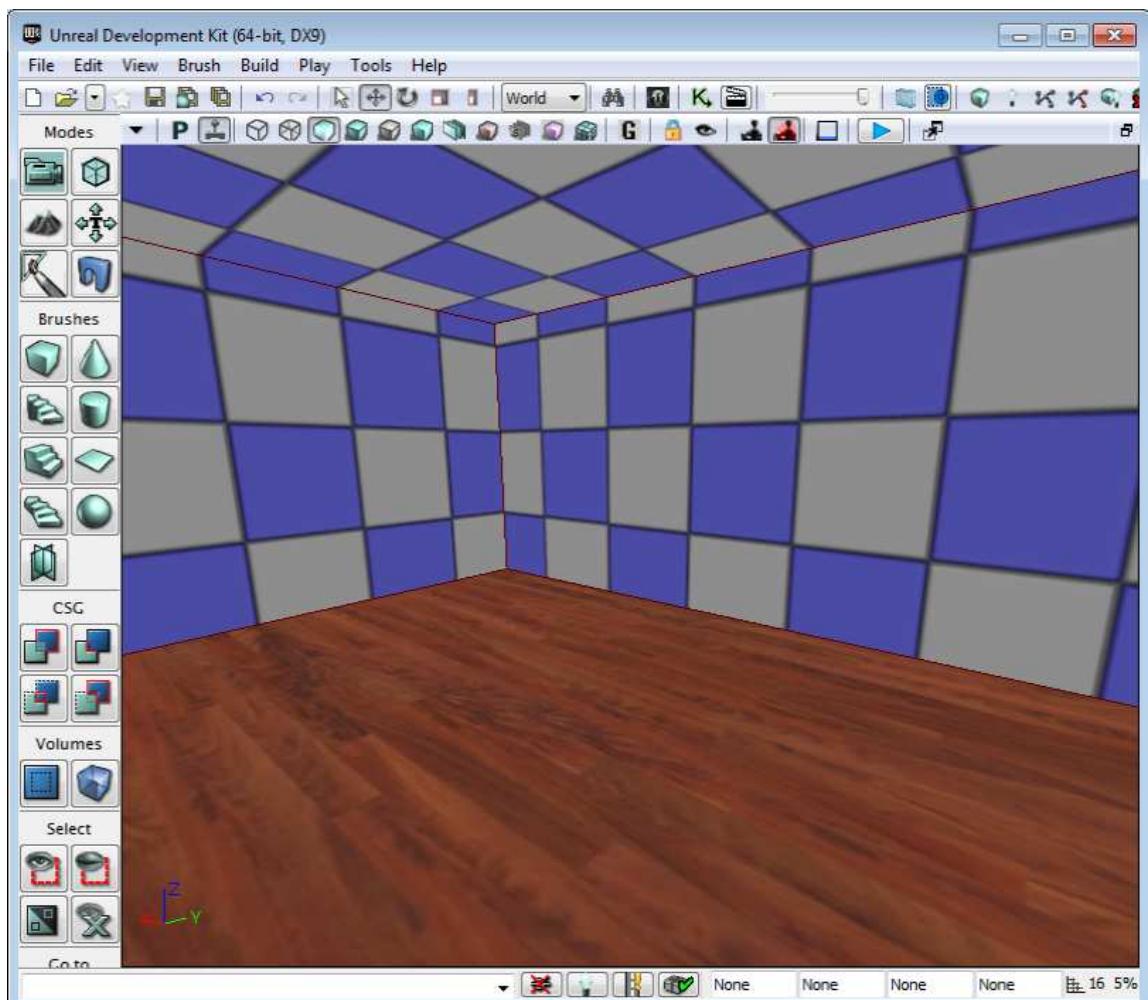
1. Selecione no Content Browser o material que deseja aplicar. Para isto basta clicar sobre o material, e ele ficará com uma borda amarela, como demonstrado na última figura da seção anterior.

2. Após selecionar o material, selecione na visão Perspective a face de um objeto no qual deseja aplicar o material, sendo neste caso o chão da sala. Clique com seguida com o botão direito do mouse e selecione a opção Apply Material:



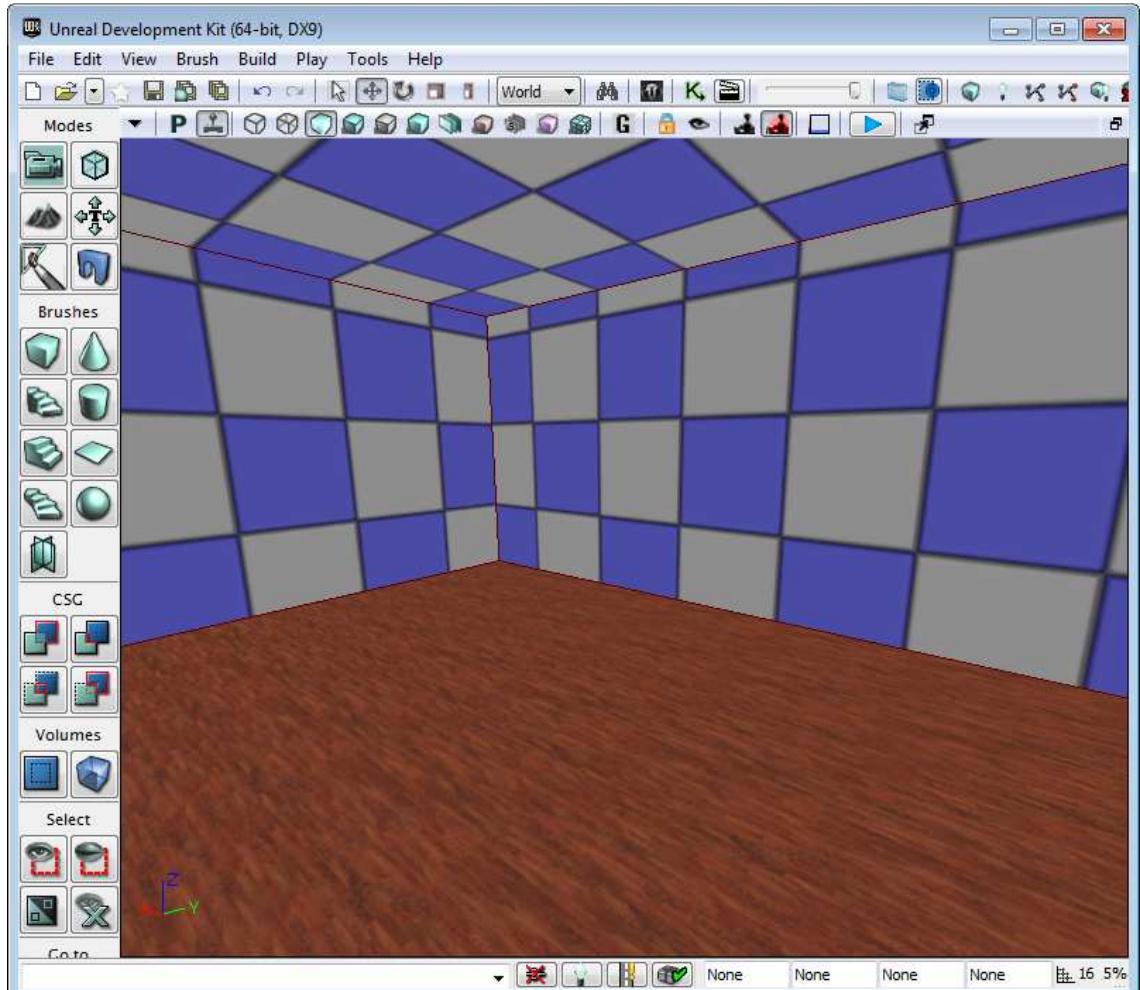
**Figura 8.24 – Aplicação do material do assoalho da sala**

3. O resultado da aplicação do material pode ser visto abaixo:



*Figura 8.25 – Material do assoalho agora aplicado*

4. Ao clicar com o botão direito na superfície onde foi aplicado o material, e selecionar a opção Surface Properties, é possível ainda alterar algumas propriedades da textura tais como seu alinhamento e escala. O exemplo abaixo mostra a mesma imagem, porém com uma redução na escala da textura:



*Figura 8.26 – Textura do assoalho redimensionada para um tamanho menor*

### **8.1.6 Criação / importação de objetos / static meshes no UDK**

Para objetos de formas simples, como caixas, o UDK provê recursos suficientes para sua criação. No caso das portas e quadros dos cenários, bastou criar formas geométricas simples para então aplicar materials a elas, processo este descrito nas seções anteriores.

Para a criação / modelagem de objetos mais complexos ou com mais detalhes de acabamento, o UDK não provê um suporte muito bom. Nesses casos, indica-se a criação desses objetos em programas de modelagem 3D, como Autodesk Maya ou 3DS Studio Max. Uma vez que os objetos tenham sido criados, basta importá-los para o UDK. Este processo será descrito a seguir.

### 8.1.6.1 Criação / importação de objetos / static meshes

Para a criação de objetos mais complexos, o Autodesk Maya 2011 foi utilizado. Tomando o sofá da sala inicial como exemplo, o modelo 3D no Maya era:

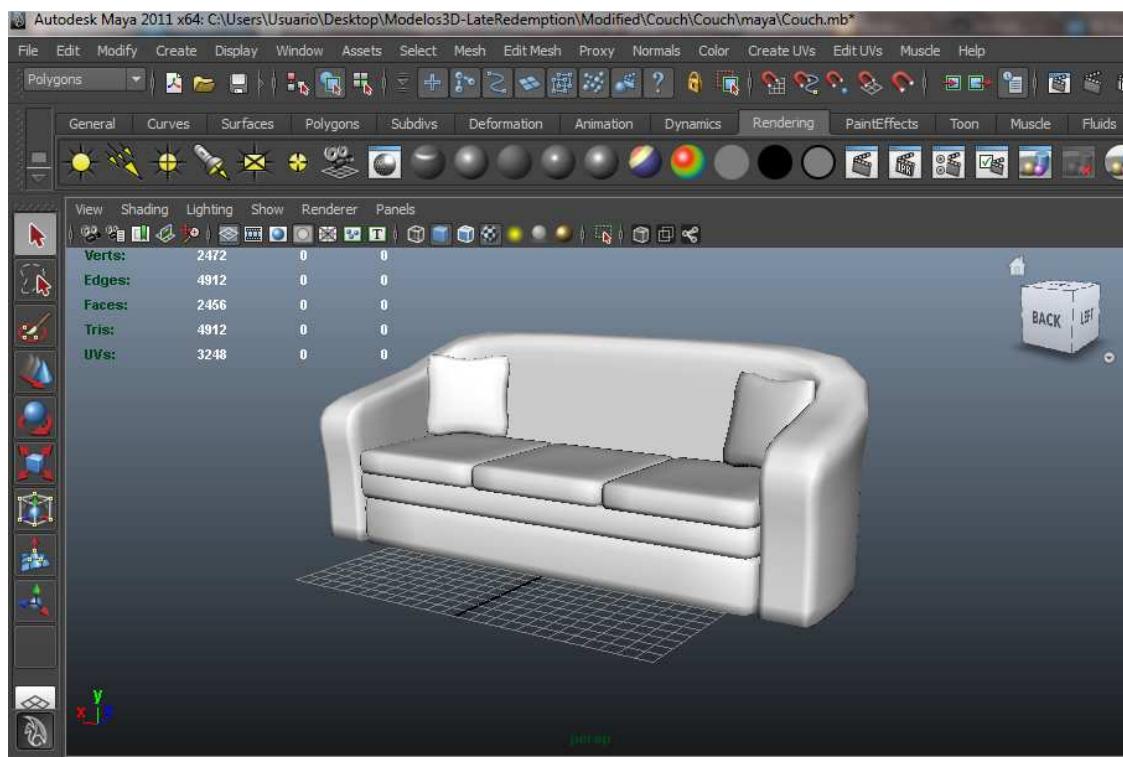


Figura 8.27 – Modelo do sofá no Autodesk Maya

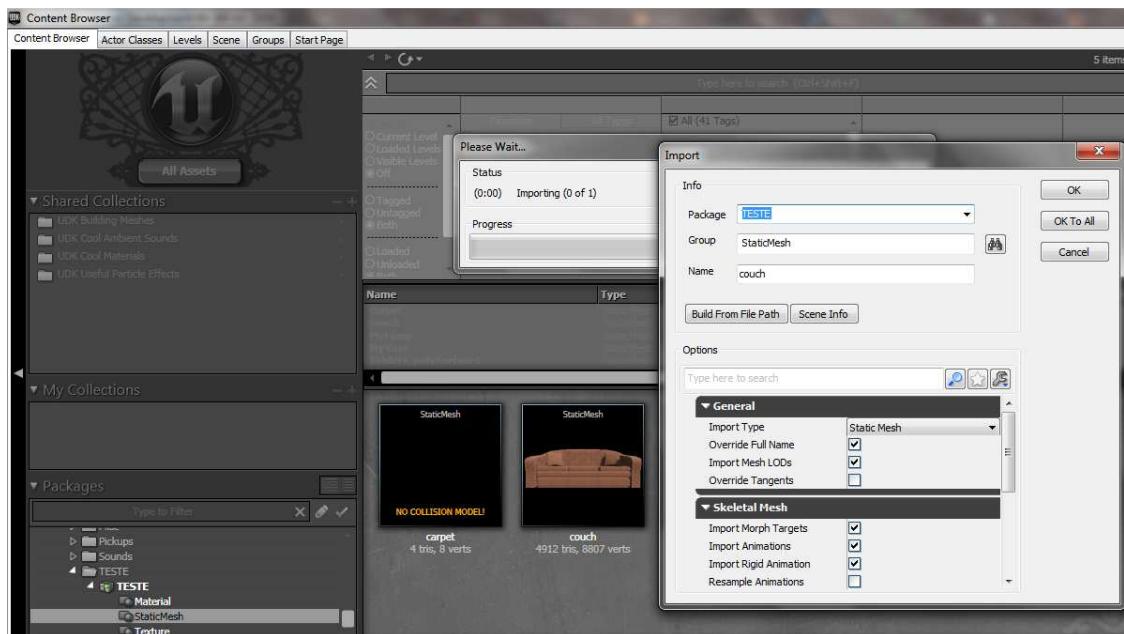
Ao modelar um objeto no Maya que será depois importado no UDK, uma série de detalhes importantes devem ser considerados, caso contrário, o resultado do processo não será satisfatório:

- 1 - Como a escala de tamanho no Maya é diferente da escala do UDK, deve-se sempre tentar achar uma proporção razoável entre os dois programas. O modelo a ser exportado no Maya sempre deve ser maior do que o de costume, como pode-se ver na figura.
- 2 - Como resultado do escalonamento do objeto (e de outras operações no Maya), o pivot (indicador das coordenadas dos 3 eixos do objeto em questão) acaba se deslocando de forma errônea na tela. É importante que o objeto seja exportado com o pivot centralizado. Para isso, deve-se usar a opção **Modify → Center Pivot**.
- 3 - Recomenda-se ainda posicionar o objeto modelado no centro do grid, para evitar problemas de referências dos 3 eixos cartesianos depois.
- 2 - Deve-se determinar quais são as áreas com texturas diferentes no objeto. Como o UDK não consegue importar o objeto criado no Maya já com a textura aplicada, esse passo de mapeamento de texturas será muito importante mais adiante, quando o material será aplicado no objeto dentro do UDK.
- 3 - Deve-se remover as entradas de histórico do objeto no Maya, caso contrário, todos os passos intermediários da modelagem também serão importados no UDK. Para a remoção das

entradas do histórico no Maya, basta escolher a opção de menu **Edit → Delete by Type → History**.

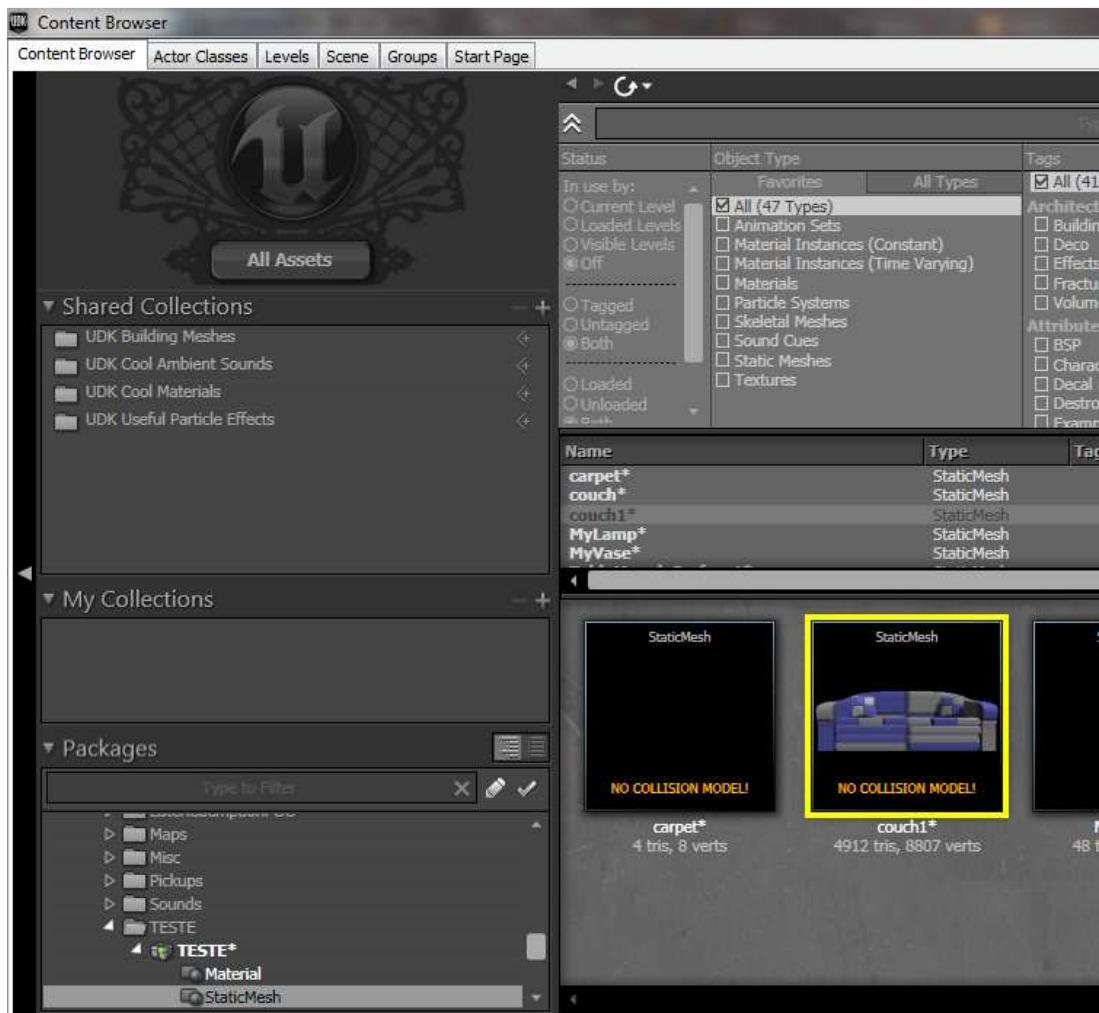
4 - Para exportar um objeto, basta escolher a opção **File → Export All**, e em seguida salvar o arquivo com uma extensão **.fbx**. Essa extensão é uma das opções reconhecidas pelo UDK a importação de um static mesh. A outra opção é usar o plugin Actor X do Maya, que gera arquivos com extensão **.ase**, porém essa alternativa não foi utilizada.

Com o modelo exportado, no Content Browser, devemos criar um novo *Static Mesh* para ele. Dentro do *Content Browser*, selecione o pacote para o qual o objeto será importado e escolha a opção Import. Selecione então o objeto **.fbx** criado e informe o grupo, pacote e nome a ser usado pelo objeto.



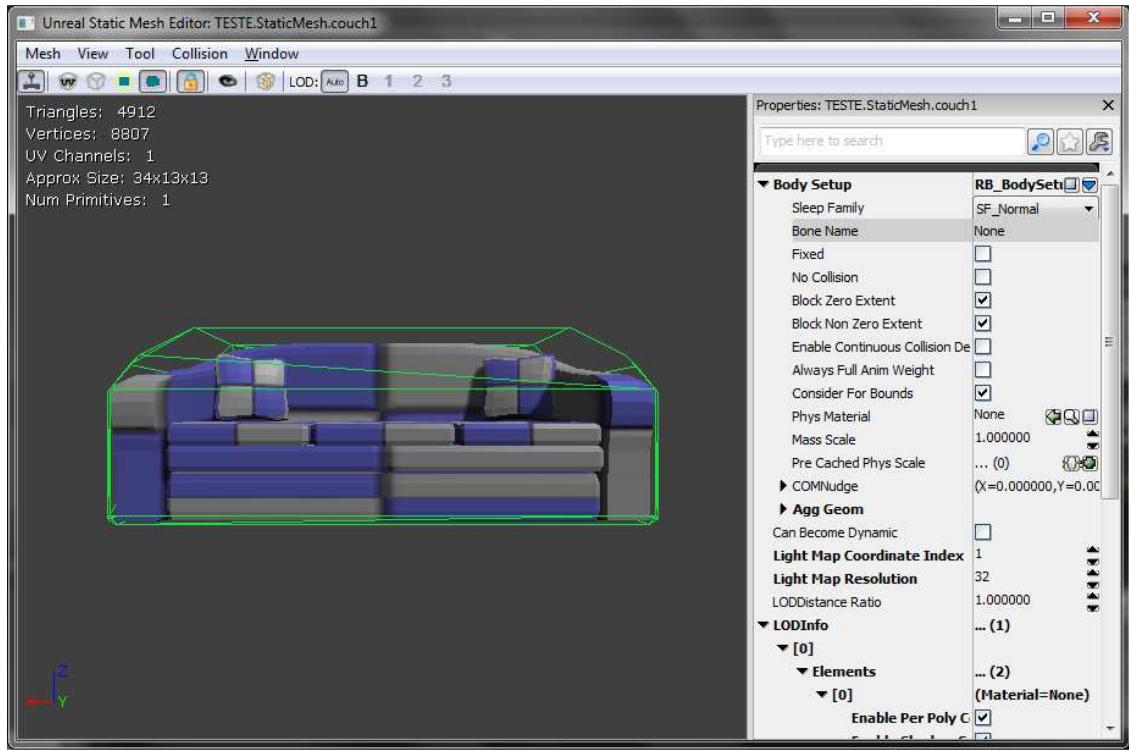
**Figura 8.28 – Criando um Static Mesh para o sofá no UDK**

O resultado do processo é um objeto sem textura e sem modelo de colisão:



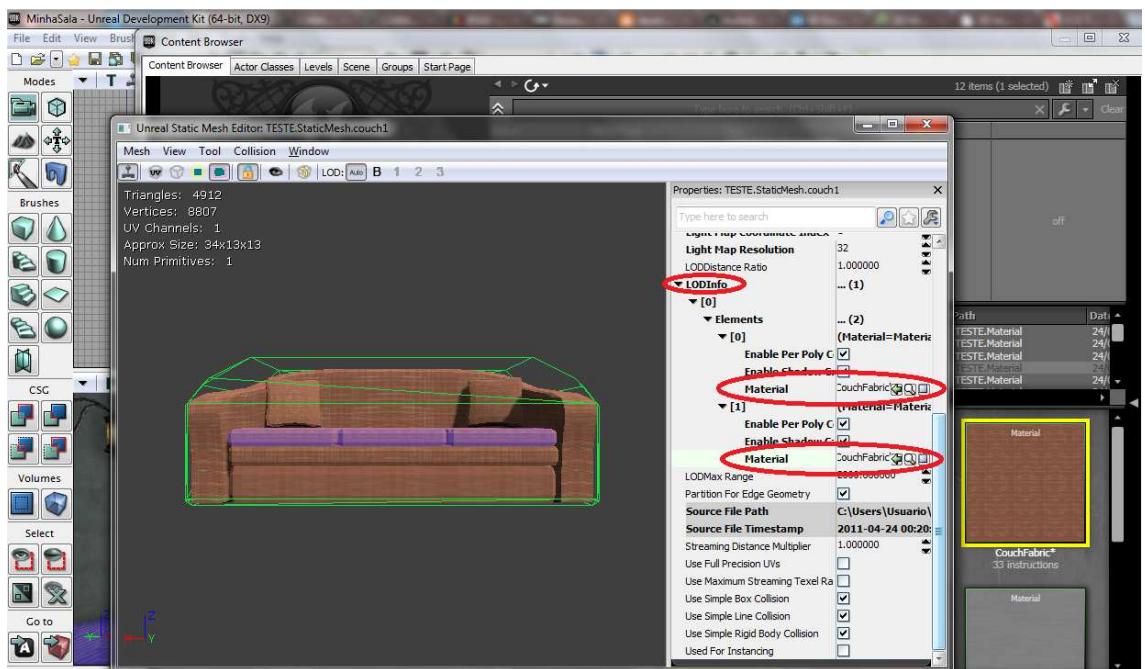
*Figura 8.29 – Sofá importado, mas sem modelo de colisão ou textura*

Sem um modelo de colisão, o objeto poderá ser atravessado pelos personagens no cenário. Na grande maioria dos casos, esse não é um resultado esperado, logo, para adicionar um modelo de colisão no objeto, basta clicar duas vezes no mesmo, ainda dentro do *Content Browser*. A janela do *Static Mesh Editor* será aberta, e nela poderemos indicar o modelo de colisão a ser utilizado. Dentro do menu *Collision*, escolha a melhor opção para o objeto em questão, e também selecione a opção *Show Collision* na janela do *Static Mesh Editor*.



**Figura 8.30 – Adicionando um modelo de colisão ao sofá**

Para acrescentar a textura a este objeto, ainda com a janela do *Static Mesh Editor* aberta, selecione no *Content Browser* o material a ser usado no objeto (com a seleção em amarelo na figura abaixo), e então, adicione este material aos elementos do nó *LODInfo* do objeto (conforme ressaltado em vermelho):



**Figura 8.31 – Adicionando textura ao sofá**

Com estes passos, o objeto está pronto para ser adicionado no cenário.

### 8.1.7 Animação das portas

Para que o personagem principal possa abrir e fechar as portas existentes no cenário, é necessário adicionar um evento de interação com a porta e uma animação para fazer com que ela abra e feche. Como exemplo, será demonstrado abaixo como foi feita a interação com a porta da sala onde o personagem inicia o jogo.

1. Este exemplo está considerando que o *Static Mesh* que representa a porta já está posicionada no cenário como um objeto do tipo *InterpActor* (caso seja um *Static Mesh*, basta convertê-lo para *InterpActor*). O próximo passo então é criar um *Trigger* próximo a porta que quando acionado uma vez irá ativar uma animação de abertura da porta e quando acionado novamente acionará um evento de fechamento da porta. Para criar o *Trigger*, basta clicar próximo à porta com o botão direito do mouse, acessar o menu *Add Actor* e clicar em *Add Trigger*. Um objeto será criado como na figura abaixo:

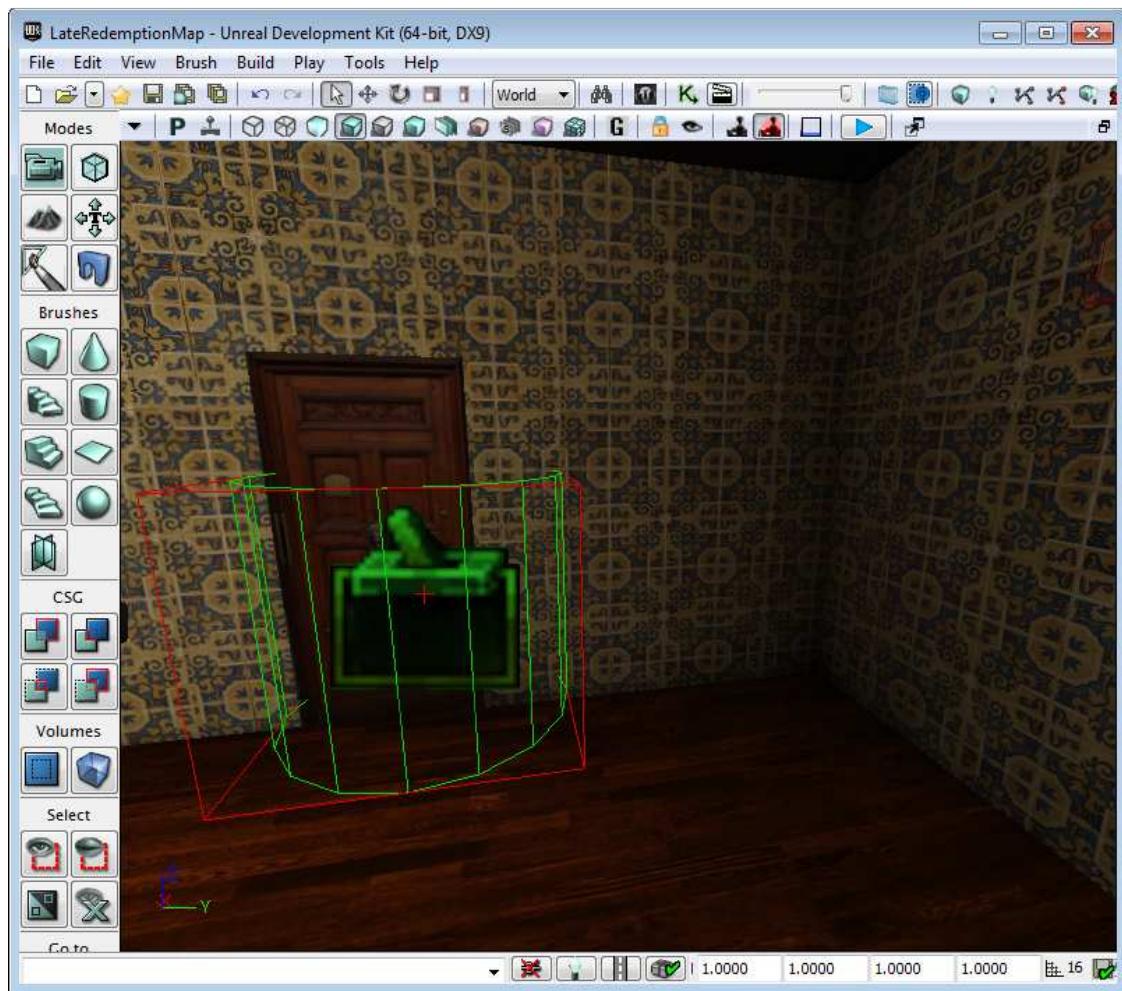
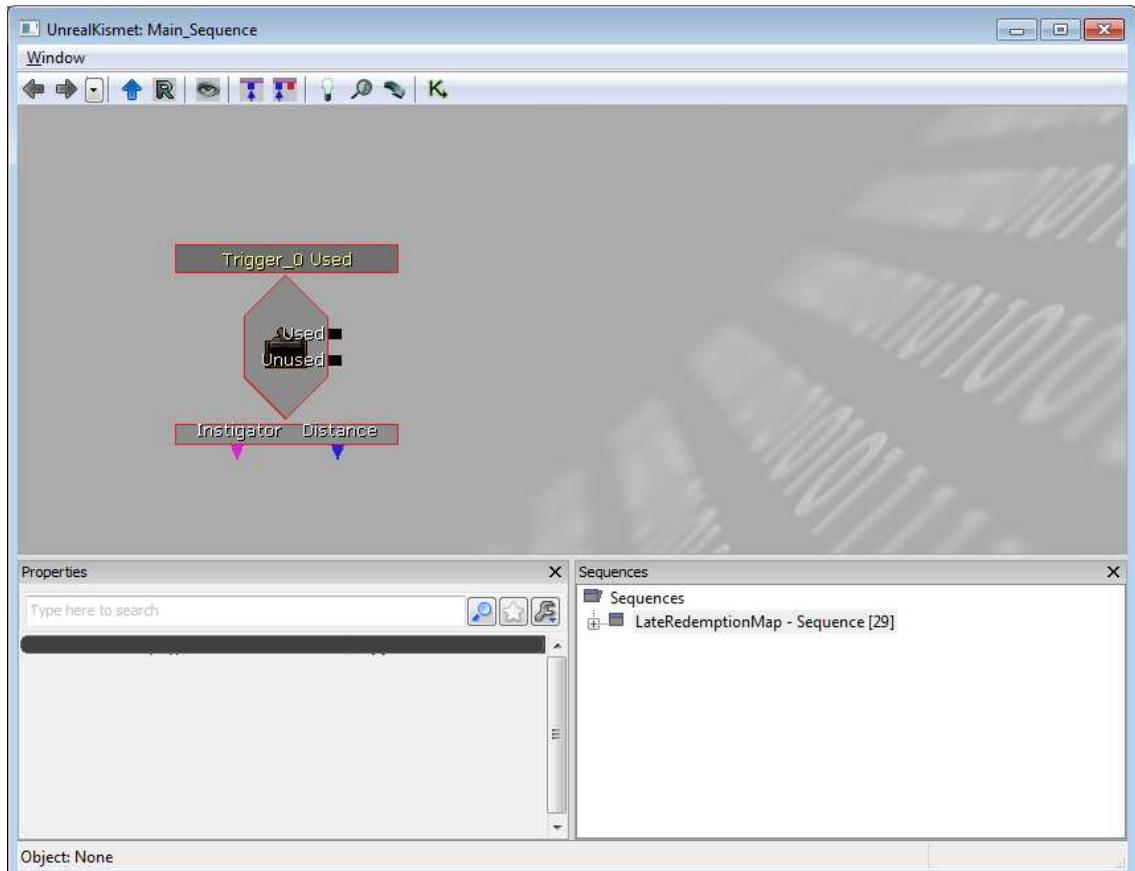


Figura 8.32 – Trigger para abrir e fechar a porta da sala

2. O *Trigger* então deve ser posicionado no centro da porta, e sua área de atuação, representada por um cilindro verde, deve abranger a área onde a *Trigger* pode ser ativada. O Trigger da imagem acima já está com estas características.

3. Em seguida, selecione o *Trigger* clicando sobre ele e abra o *UnrealKismet* para adicionar o evento de abertura e fechamento da porta. Clicando com o botão direito na área central do *Kismet*, selecione a opção *New Event Using Trigger - Used*. Isto significa que o *Trigger* será acionado ao ser usado, ou seja, quando o jogador estiver em sua área de atuação e pressionar a tecla “E”. Um objeto será adicionado na tela do *Kismet* como na imagem abaixo:



**Figura 8.33 – Trigger para abrir e fechar a porta no UnrealKismet**

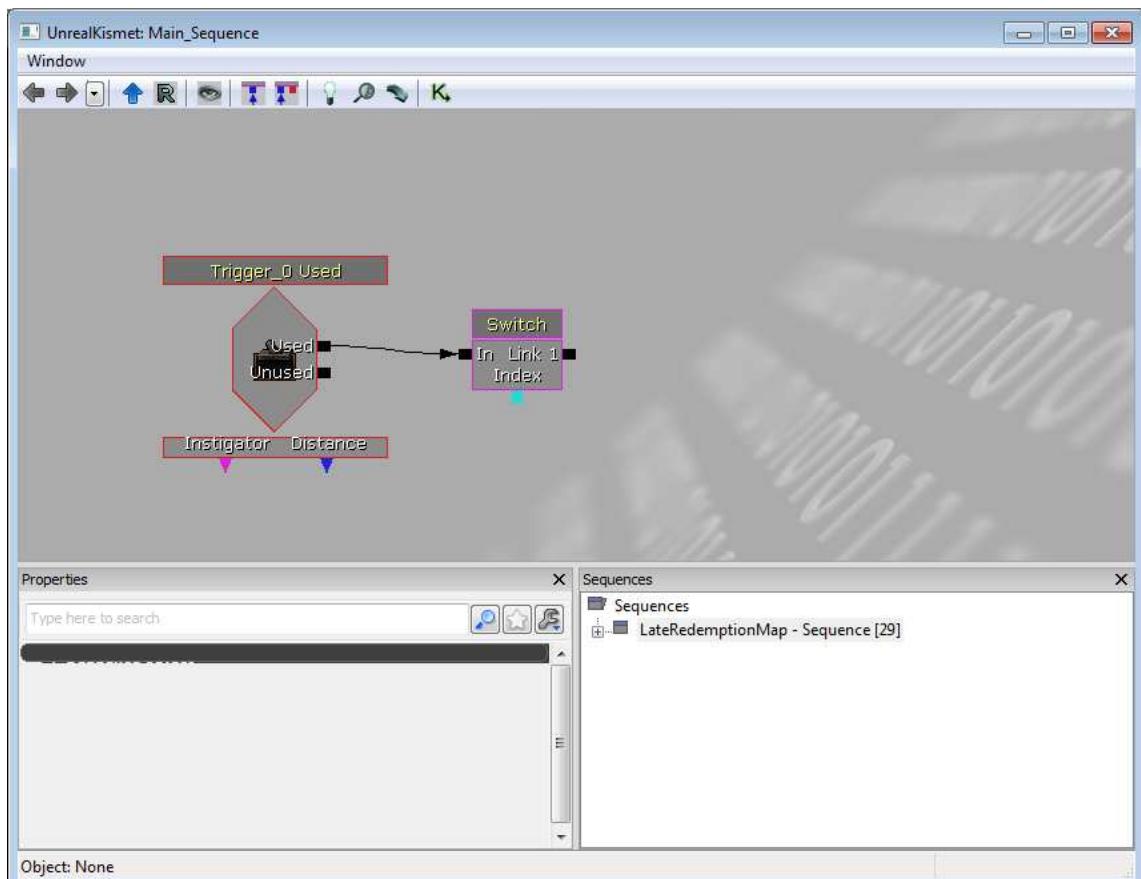
4. Ainda no *Kismet*, é importante alterar algumas propriedades do *Trigger* na parte inferior da tela. Acesse a aba *Seq Event\_Used* e desabilite a opção *Aim to Interact*. Isto garante que a mira do jogador não precisará estar na direção do *Trigger* para que este possa ser utilizado. Na aba *Sequence Event*, altere a propriedade *Max Trigger Count* para 0, garantindo assim que o *Trigger* possa ser utilizado infinitas vezes ao invés de apenas 1 permitida por padrão. Assim o jogador poderá abrir e fechar a porta quantas vezes quiser.

5. Ao ser usado, um *Trigger* permite apenas uma mesma sequência de ações a ser executada. No caso da porta, é necessário que no primeiro uso a porta seja aberta, e no segundo uso ela seja fechada. Para isso é necessário que o objeto de interação com o *Trigger* seja um *Switch*. Para adicionar um *Switch*, clique com o botão direito na área central do *Kismet* e selecione *New Action - Switch - Switch*.

6. Selecione o *Switch*, e na área de propriedades na parte inferior do *Kismet*, abra a aba *Seq Act\_Switch* e altere a propriedade *Link Count* para 2, permitindo assim a execução de duas

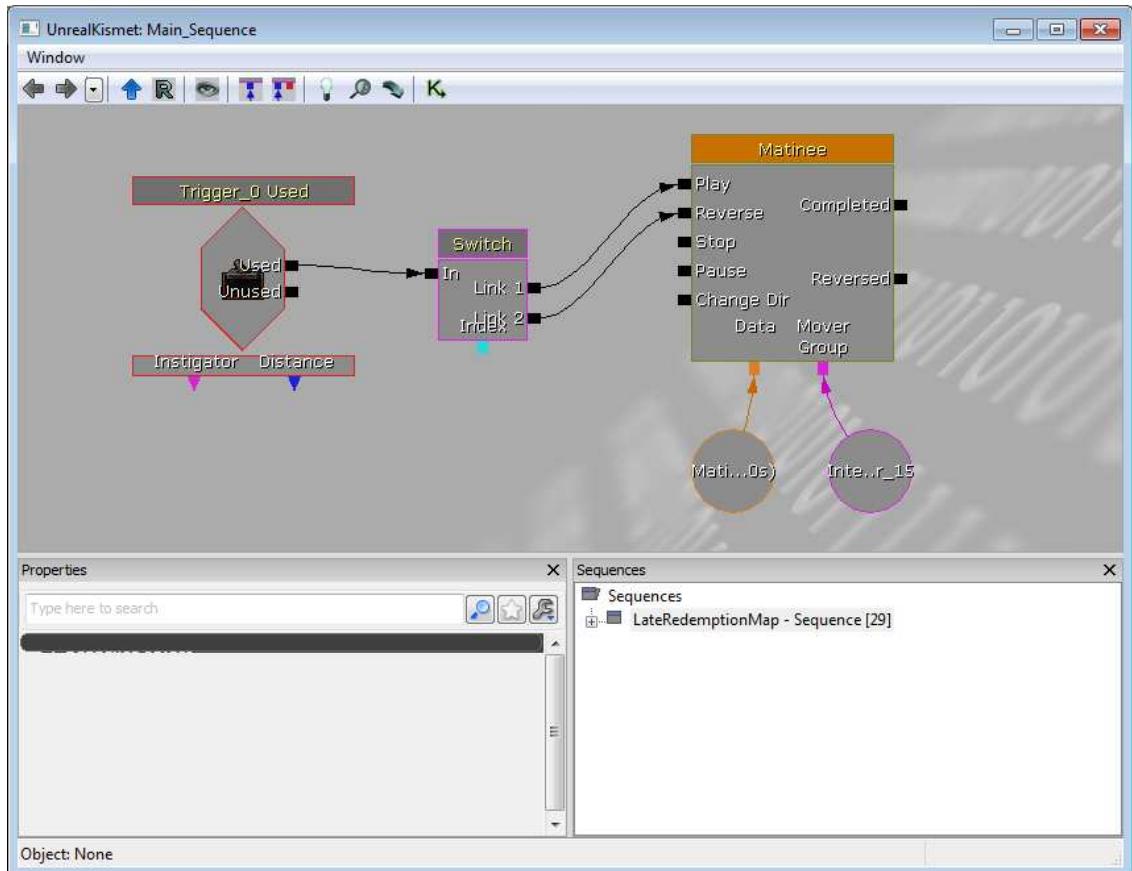
ações diferentes intercaladamente. Marque também a opção *Looping* para que o *Switch* possa ser acionado infinitamente.

7. Ligue a opção *Used* do *Trigger* com a opção *In* do *Switch*. Assim, o acionamento do *Trigger* irá acionar o *Switch*. O resultado no *Kismet* fica igual ao da imagem abaixo:



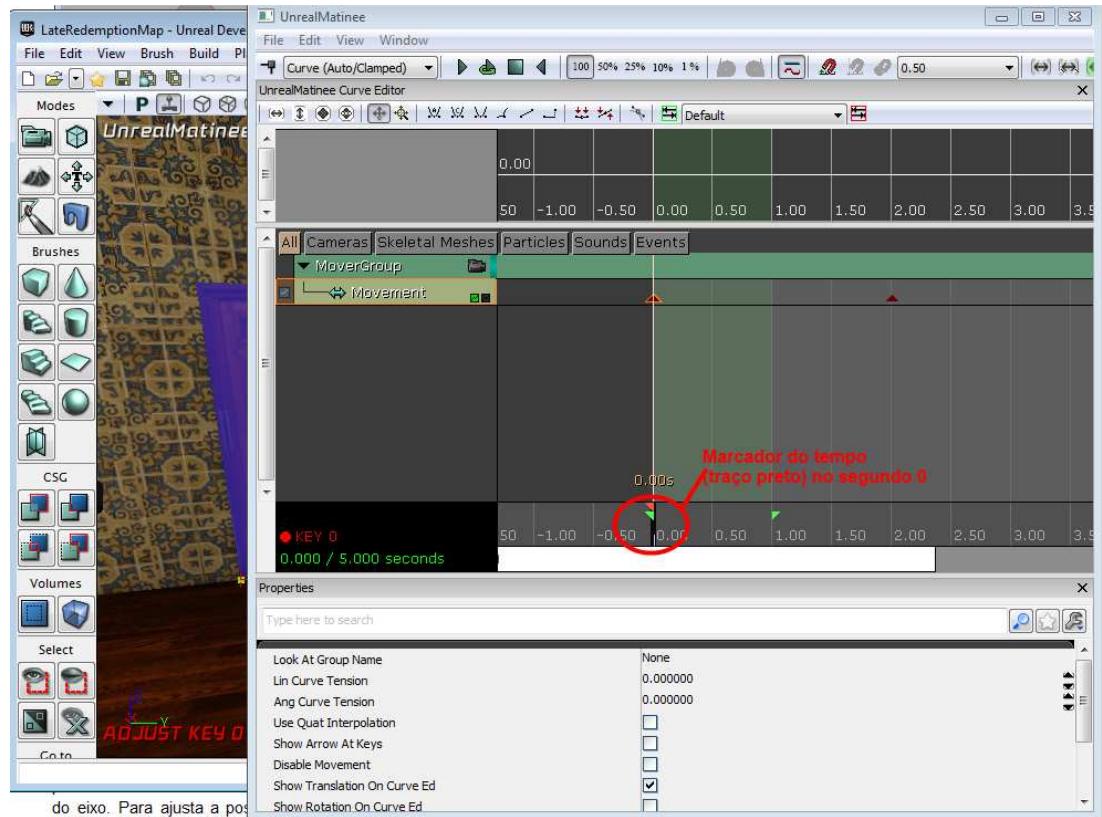
**Figura 8.34 – Relacionamento entre Trigger e Switch no UnrealKismet**

8. O próximo passo é fazer com que o *Switch* acione a animação de abrir e fechar a porta. Para isso, selecione a porta no UDK, volte ao *Kismet*, clique com o botão direito na área central e selecione a opção *New Event Using InterpActor - Mover*. Serão criado no *Kismet* um *Matinee* contendo uma faixa de *Mover* para a porta, e uma referência à própria porta. O objeto que representa a porta pode ser excluído do *Kismet*, pois não será utilizado. Conecte o *Link 1* do *Kismet* na opção *Play* do *Matinee* e o *Link 2* na opção *Reverse*. Ou seja, ao acionar o *Trigger* uma vez a animação que faremos para abrir a porta irá ser exibida, e ao acioná-lo novamente, a animação será revertida, fazendo com que a porta se feche. A figura no *Kismet* fica conforme abaixo:



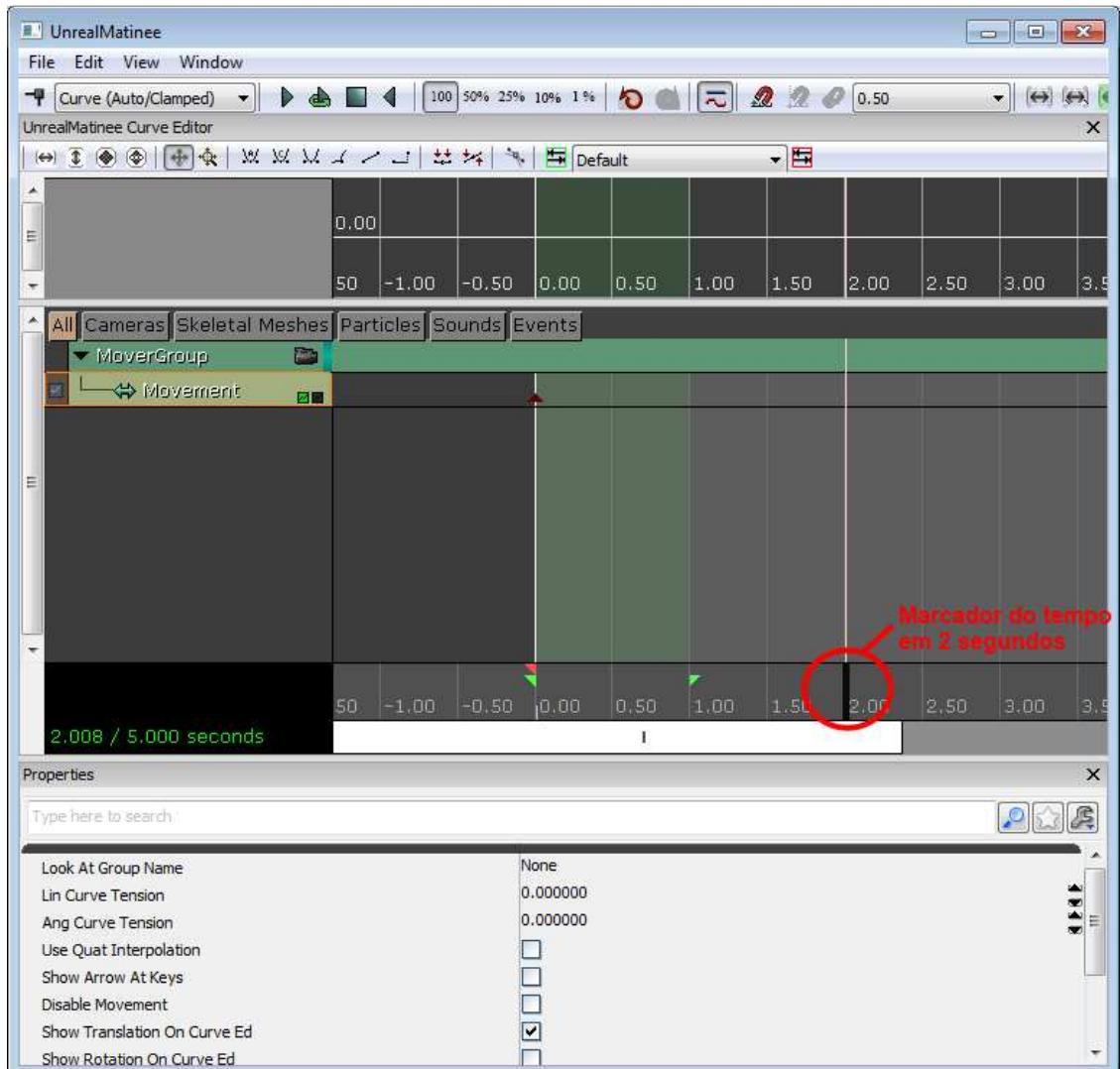
**Figura 8.35 – Visualização no UnrealKismet para Trigger, Switch e Matinee para abrir e fechar a porta**

9. Feito isso, basta fazermos a animação no *Matinee*. Dê um duplo clique no objeto *Matinee* no *Kismet* para abrir a tela de edição de animações.
10. Com o *Matinee* aberto, selecione a porta na visão *Perspective* do UDK. No *Matinee*, mantenha o marcador de tempo no segundo 0 conforme na imagem abaixo e então pressione Enter para adicionar um *Key Frame*. Isto indica que no início da animação a porta estará fechada.



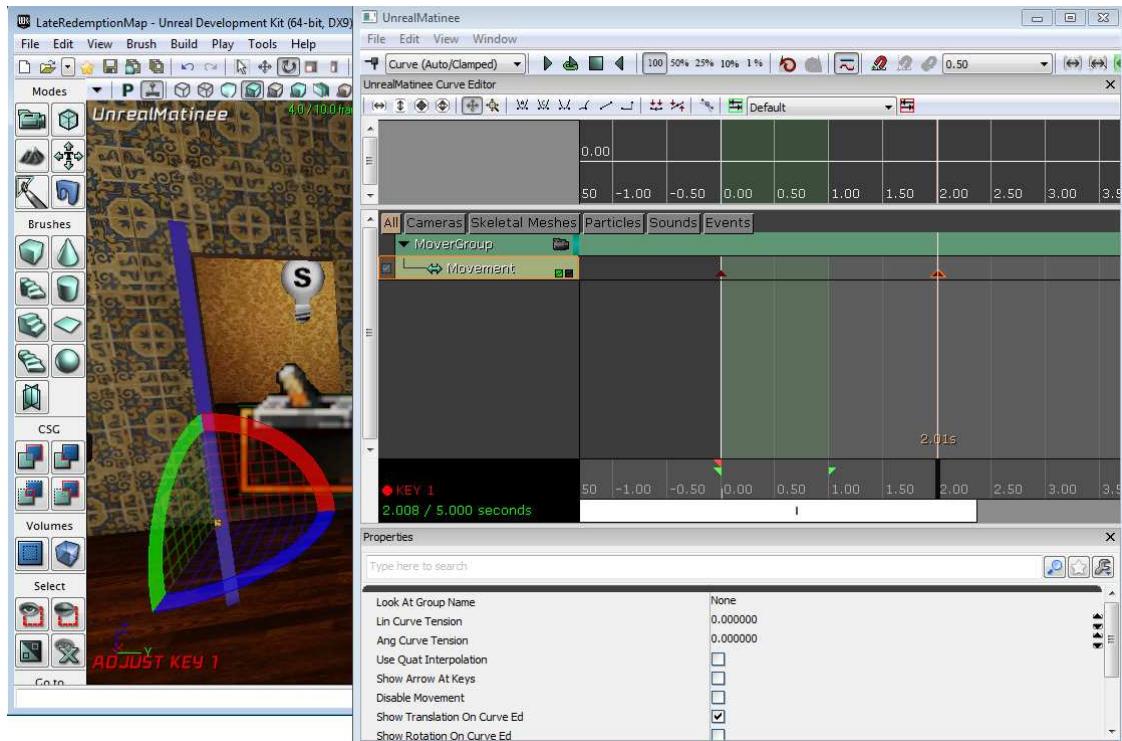
**Figura 8.36 – Matinee da abertura da porta configurado no momento inicial (0 segundos) da animação**

11. Mova o marcador para o tempo necessário para abrir a porta na animação (no exemplo 2 segundos), conforme a imagem abaixo:



*Figura 8.37 – Matinee da abertura da porta configurado no momento final (2 segundos) da animação*

12. Mantenha o *Matinee* aberto, e na visão *Perspective* do UDK, utilize a ferramenta de rotação para girar a porta em 90 graus para o lado que ela deve ser aberta. Em seguida pressione Enter para gravar um novo *Key Frame* que representa o fim da animação.



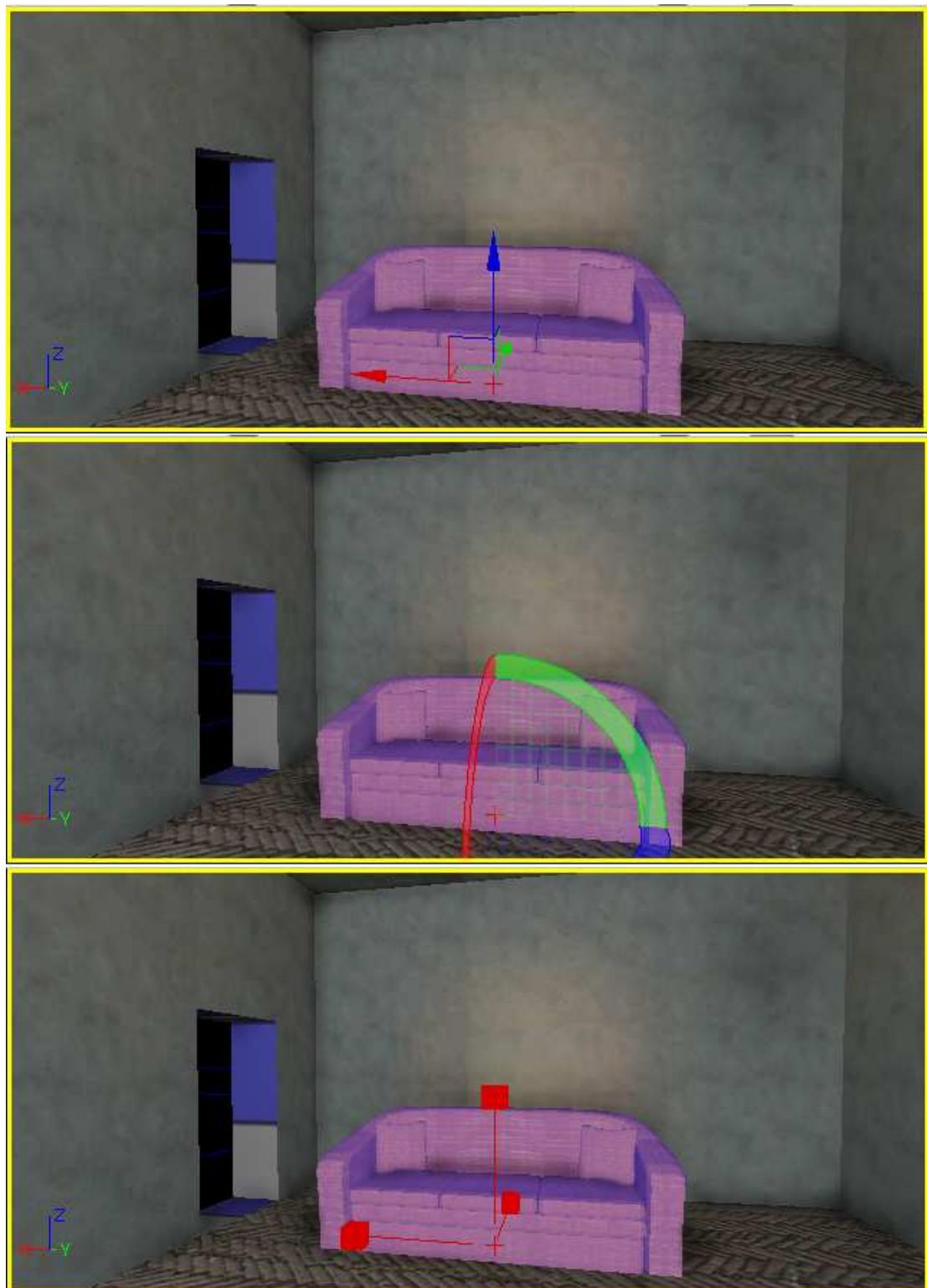
*Figura 8.38 – Rotação da porta para simular sua abertura na animação*

Nota sobre o passo 12: Antes de rotacionar a porta, é importante que o eixo do objeto esteja posicionado em um dos cantos inferiores do objetos, pois a rotação terá como base a posição do eixo. Para ajustar a posição do eixo, mantenha a porta selecionada e então clique com o botão direito no local onde o eixo deve ser posicionado. Acesse então a opção *Pivot - Move Here*. Acesse então a opção *Pivot - Save Pivot to PrePivot*.

13. A animação está concluída e pode ser testado no próprio *Matinee* utilizando as teclas *Play* e *Stop* na parte superior da tela.

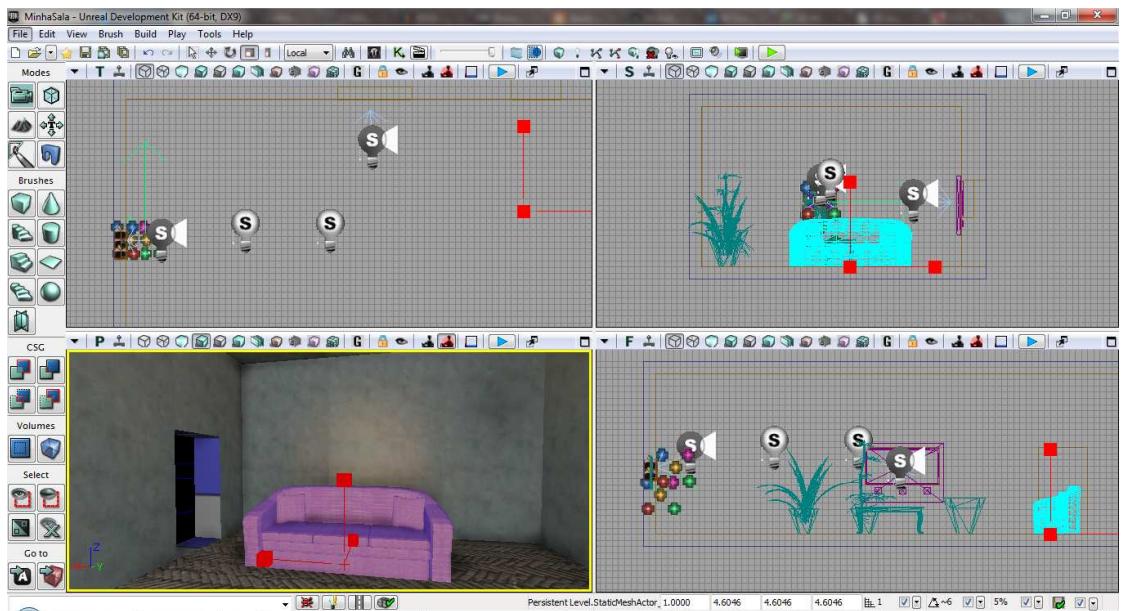
### 8.1.8 Adição dos objetos no cenário

Com o objeto importado para o UDK, basta agora adicioná-lo no cenário. Para isso, no *Content Browser*, selecione o objeto. Sem fechar o *Content Browser*, segure a tecla “S” e clique no local do cenário onde deseja adicionar o objeto. Neste momento, o objeto será adicionado, já com a textura correta. A partir de então, ele pode ser manipulado como qualquer outro elemento do cenário, com as 3 operações básicas de deslocamento, escalonamento e rotação, conforme a figura abaixo:



**Figura 8.39 – As 3 operações básicas para fazer em um objeto de cenário**

Dessa forma, é possível colocar o objeto em qualquer local do cenário, em qualquer tamanho. Vale lembrar que os painéis com as outras visões *Top*, *Side* e *Front* do *UDK Editor* podem e devem ser utilizadas para esse propósito, conforme pode ser visto na figura abaixo.



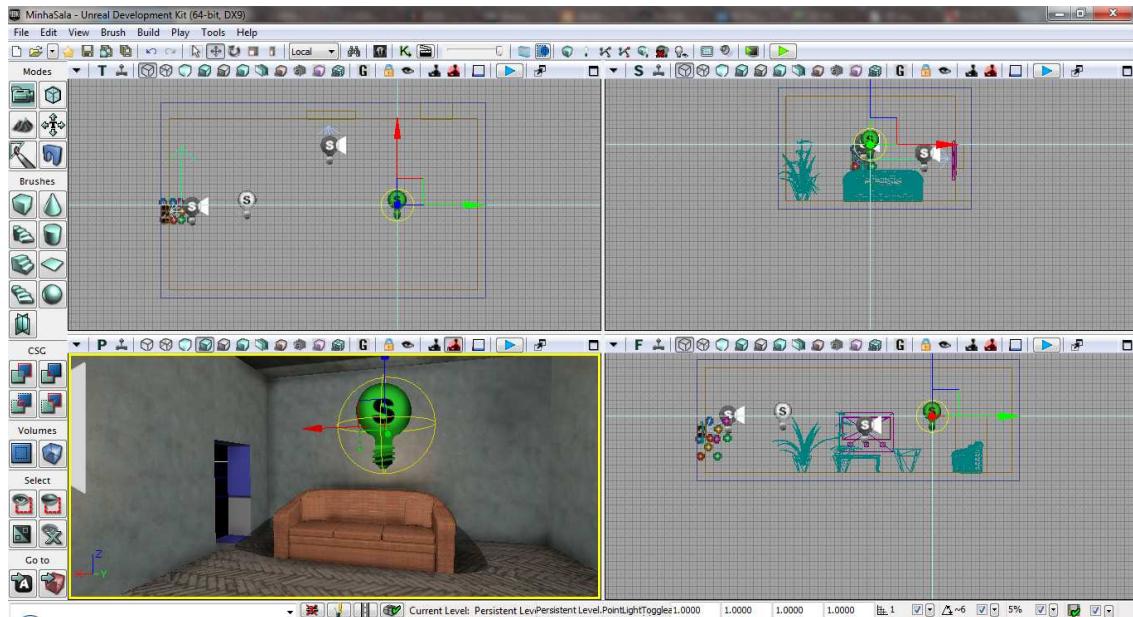
**Figura 8.40 – As 3 views do Unreal Editor: top, side e front**

### 8.1.9 Adição de luzes no cenário

Sem a presença de luzes no cenário, este não é renderizado corretamente. Por isso, é essencial adicionar luzes em pontos específicos, de acordo com o que foi pensado para o cenário.

Para adicionar uma luz ao cenário, basta segurar pressionada a tecla “L” e clicar com o botão esquerdo do mouse no cenário. Dessa maneira, a forma mais comum de luz no UDK, a *Point Light*, será adicionada no cenário no local onde ocorreu o clique. A *Point Light* tem propriedades genéricas, para comportamentos mais específicos, como uma luz iluminando um ponto específico, outros tipos de luzes podem ser utilizados, como a *Spot Light* (para o caso em questão). Existem duas maneiras para adicionar outros tipos de luzes: a primeira seria clicando com o botão direito no cenário, e escolhendo a opção *Add Actor* → *All Templates* → *Add Light <type>*, onde <type> é o tipo específico de luz requerida. Outra alternativa possível é converter uma luz já existente no cenário em outro tipo de luz. Para isso, clique com o botão direito na luz a ser convertida e escolha a opção *Convert Light*, e então escolha a conversão a ser feita, indicando luz existente e luz a ser obtida.

Uma vez presente no cenário, uma luz também se comporta como qualquer elemento de cenário, podendo sofrer a ação das três operações básicas: deslocamento, escalonamento e rotação, conforme ilustrado na figura abaixo.



**Figura 8.41 – Adição de luzes ao cenário**

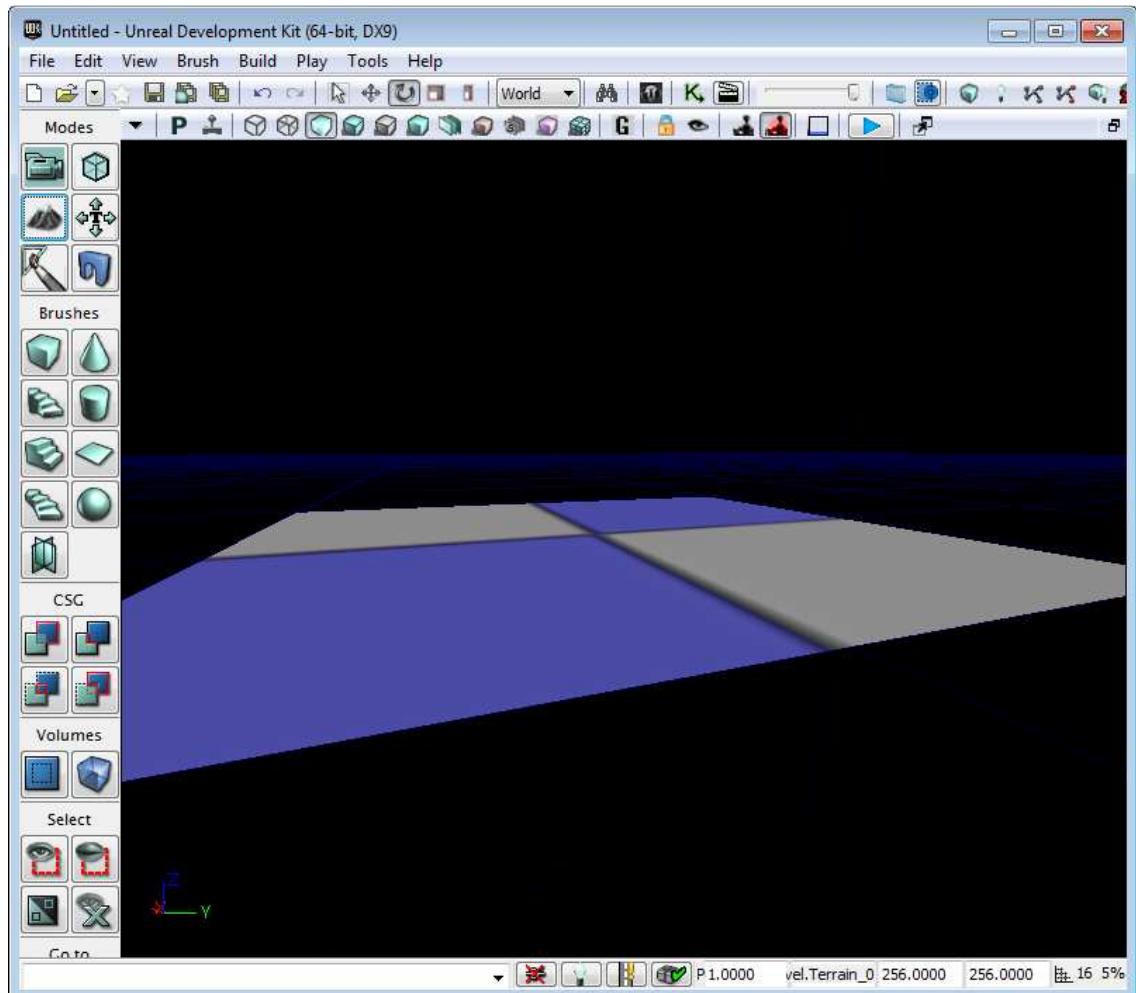
## 8.2 Ambiente externo

O ambiente externo onde Marshall começa na animação inicial do jogo e também onde pode ser visto através das janelas é constituído de dois elementos principais: um Terreno (*Terrain*) com textura de grama seca e contendo algumas árvores queimadas e também uma cúpula representando o céu. Esta seção irá explicar brevemente como estes objetos foram adicionados.

### 8.2.1 Terreno

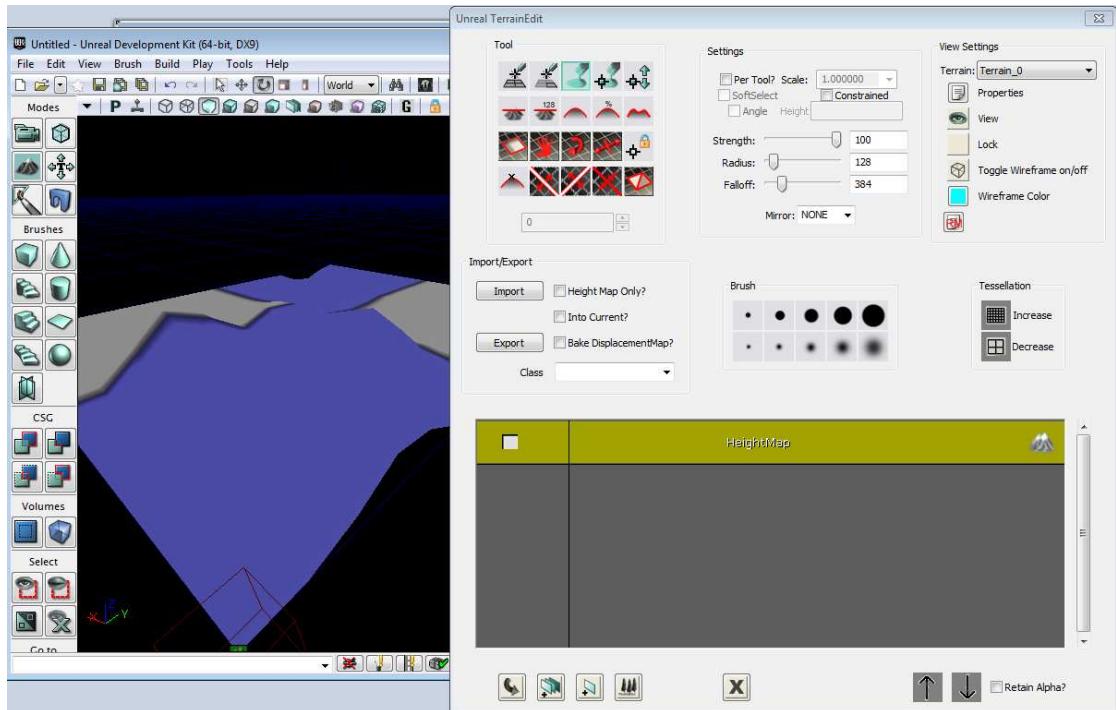
O UDK possui um recurso chamado *Terrain* que é frequentemente utilizado para criação de ambientes externos. Um *Terrain* é um tipo de objeto adicionado como uma folha, ou seja, ele não possui uma medida de altura Z (possui apenas as dimensões X e Y), sendo que o terreno pode ser esculpido pelo desenvolvedor utilizando a ferramenta de edição de terreno. É possível adicionar elevações e depressões e também aplicar mais do que um tipo de material em diferentes partes do terreno. A criação de um terreno semelhante ao feito na área externa do Late Redemption será demonstrada abaixo.

1. Acesse o menu *Tools* do *UDK - New Terrain*. Na opção Patches da tela que surgir, determine o valor de X e Y para dimensionar o terreno. Como exemplo, mantenha as duas opções como 16. Um terreno é então criado:



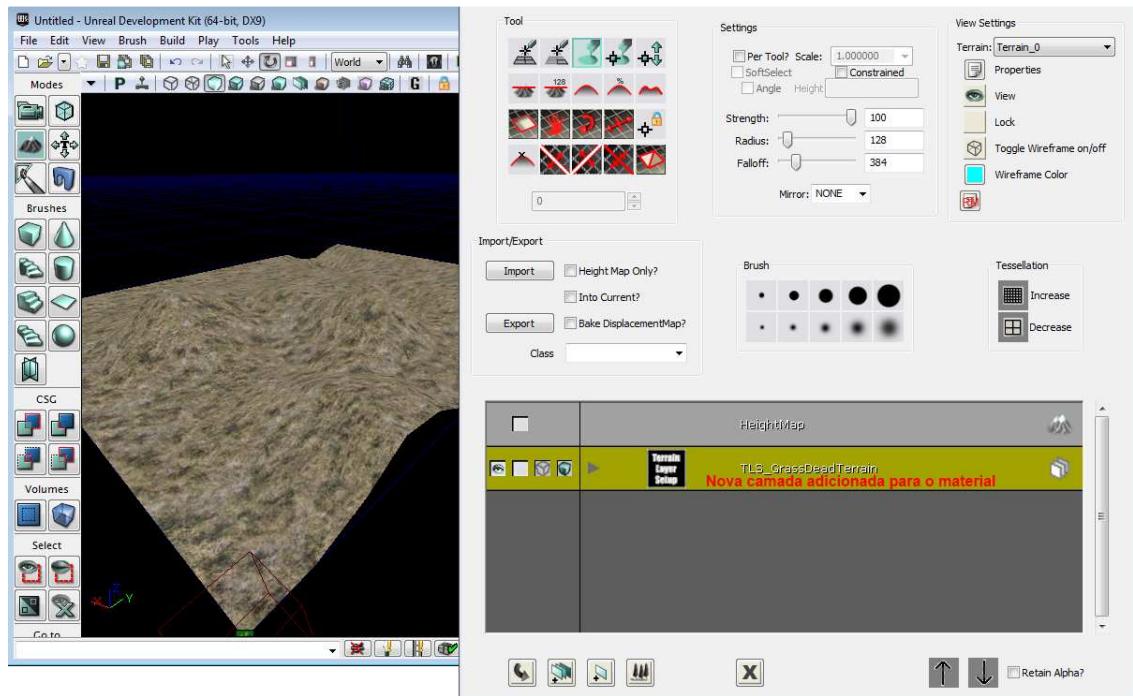
*Figura 8.42 – Terreno no tamanho 16 por 16 logo após ser adicionado em um cenário vazio*

2. Acesse a ferramenta de edição de terreno na barra de ferramentas à esquerda no UDK. Na tela que irá aparecer, a ferramenta *Paint* e outras ferramentas como *Smooth* e *Noise* podem ser utilizadas para esculpir o terreno. No exemplo abaixo, a ferramenta *Paint* foi utilizada para criar alguns relevos:



**Figura 8.43 – Terreno após a aplicação de elevações e depressões e ferramenta de edição de terreno**

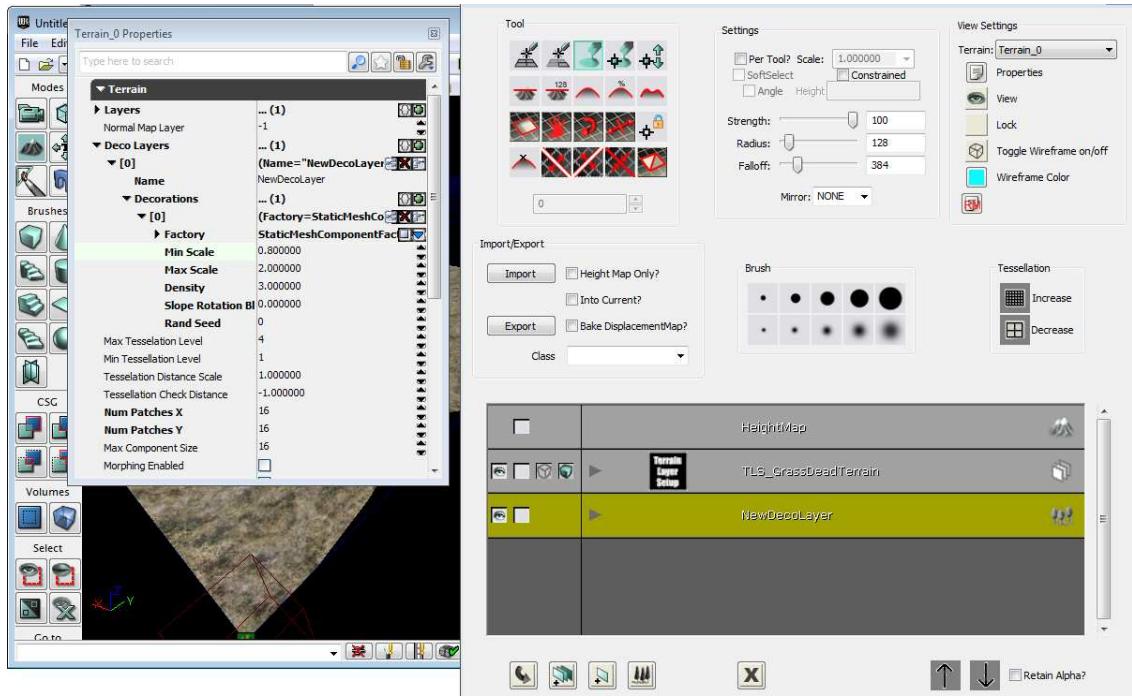
3. É possível aplicar um material no terreno para colocar uma textura. Basta selecionar o material no *Content Browser*, e na parte inferior cinza escura da tela de edição de terreno, clique com o botão direito e selecione a opção *New Terrain Setup Layer from material* (auto-create). O resultado aplicando o mesmo material do terreno do Late Redemption é o demonstrador abaixo. Note que uma nova camada pode ser visualizada na ferramenta de edição de terreno.



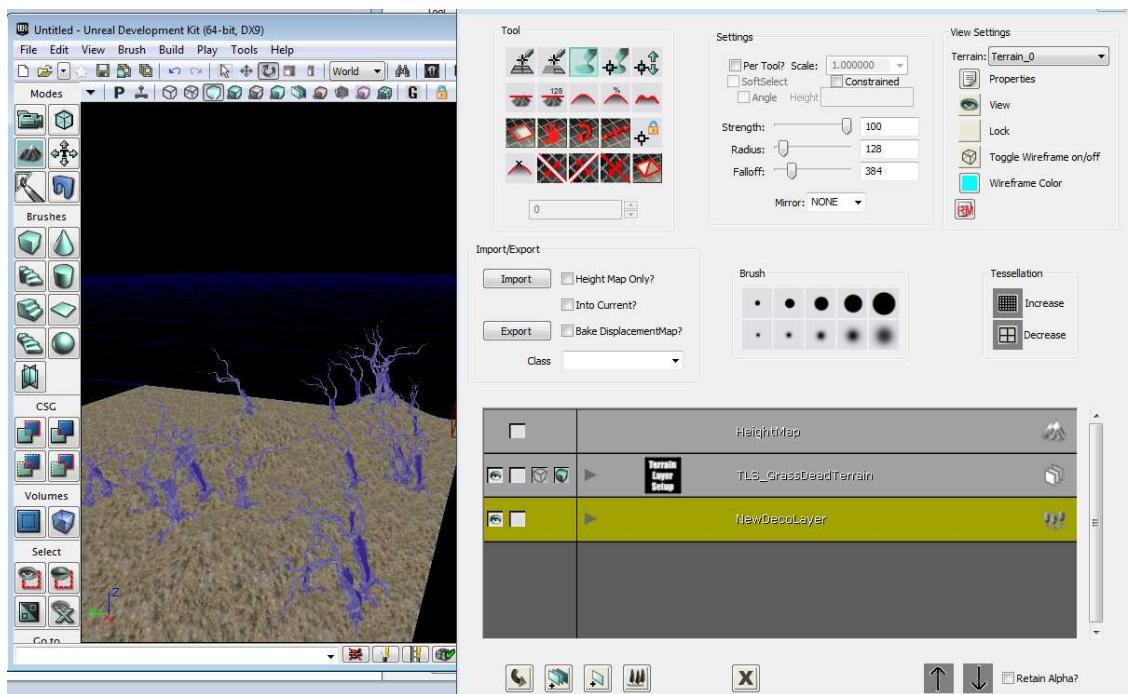
**Figura 8.44 – Terreno após aplicação de material**

4. É possível ainda criar uma camada de objetos de decoração como as árvores queimadas que podem ser vistas pelas janelas no jogo. Para isso, é necessário criar uma *Deco Layer* (com o botão direito na parte inferior cinza escura da tela de edição de terreno selecione a opção *New Deco Layer*). Após criar a camada, selecione o *Static Mesh* que representa a árvore no *Content Browser* e clique com o botão direito sobre a *Deco Layer* na ferramenta de edição e selecione a opção *Add Selected Decoration*.

5. Após alterar algumas propriedades de escala e densidade da *Deco Layer* na tela de propriedades do terreno (acessada pressionando F4 quando o terreno estiver selecionado), é possível adicionar as árvores sobre o terreno:



**Figura 8.45 – Propriedades de escala e densidade dos objetos de decoração da Deco Layer**



**Figura 8.46 – Árvores adicionadas na Deco Layer**

## 8.2.2 Céu

O céu é um objeto em forma de cúpula com uma textura de céu. O UDK já fornece alguns modelos prontos de céu, e esses modelos nada mais são do que *Static Meshes*. Para adicionar o céu, selecione-o no *Content Browser*, clique com o botão direito na visão *Perspective* e selecione a opção *Add Static Mesh*, conforme ilustrado nas imagens abaixo:

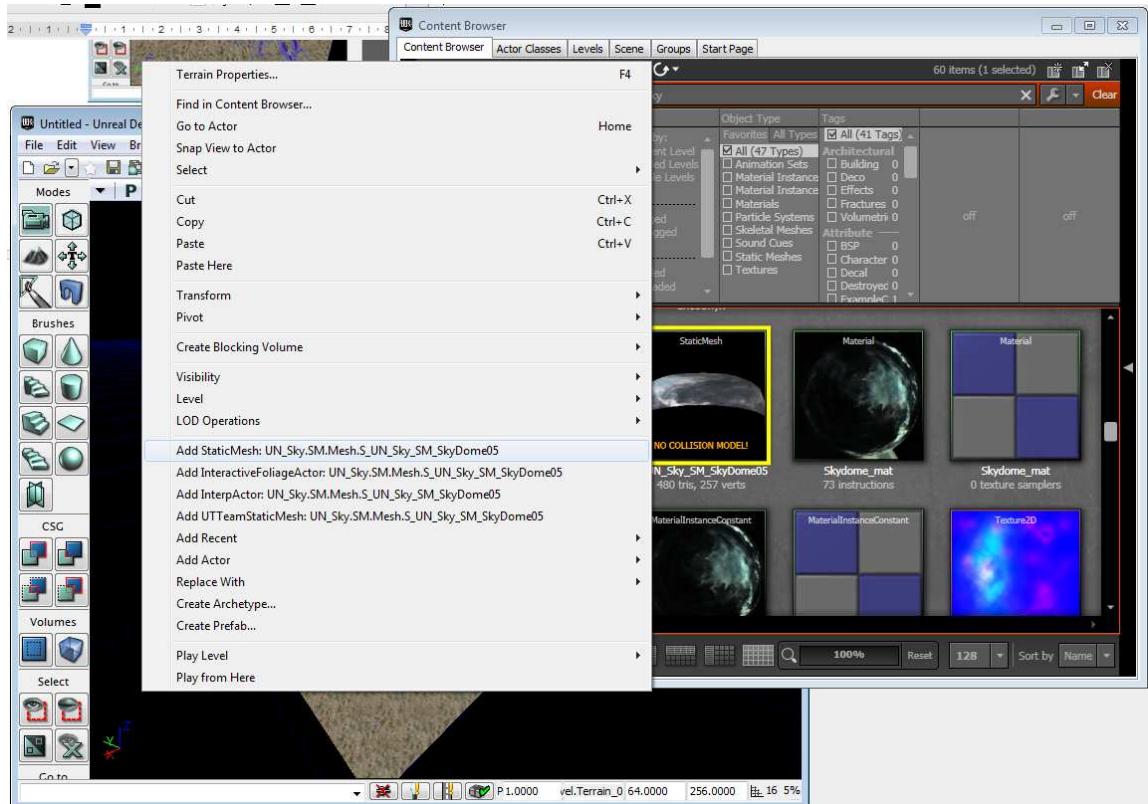


Figura 8.47 – Adição do objeto que representa o céu no cenário

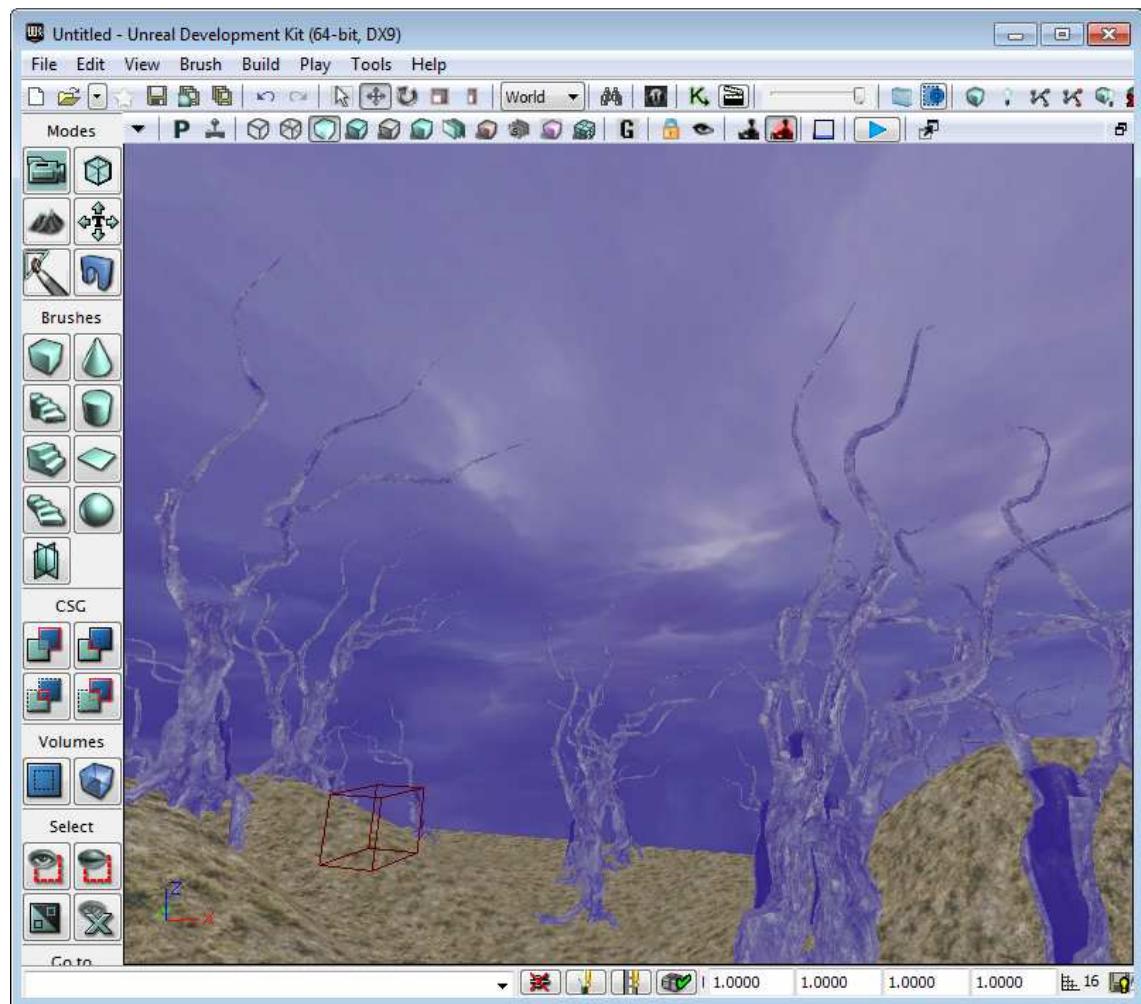
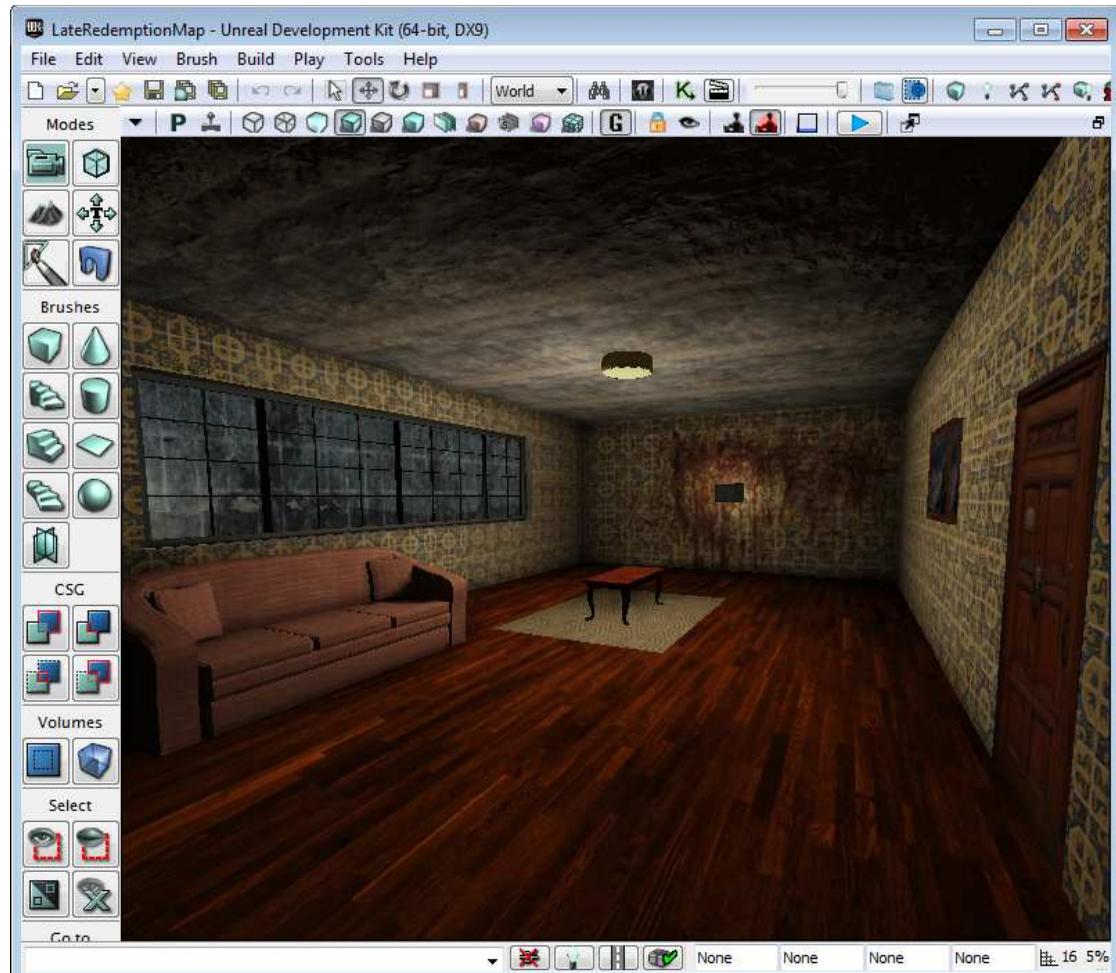


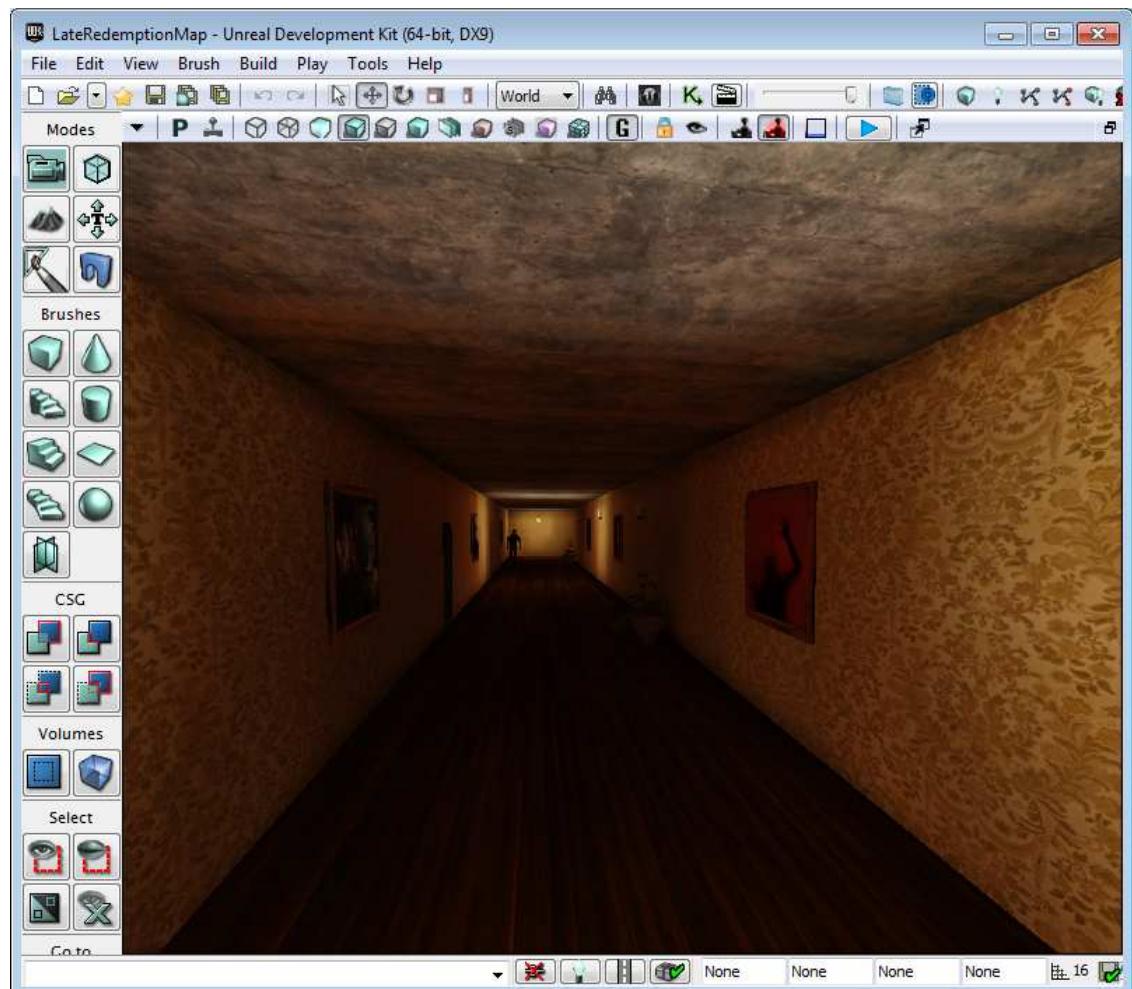
Figura 8.48 – Ambiente com o céu já adicionado

### 8.2.3 Exibição do cenário concluído

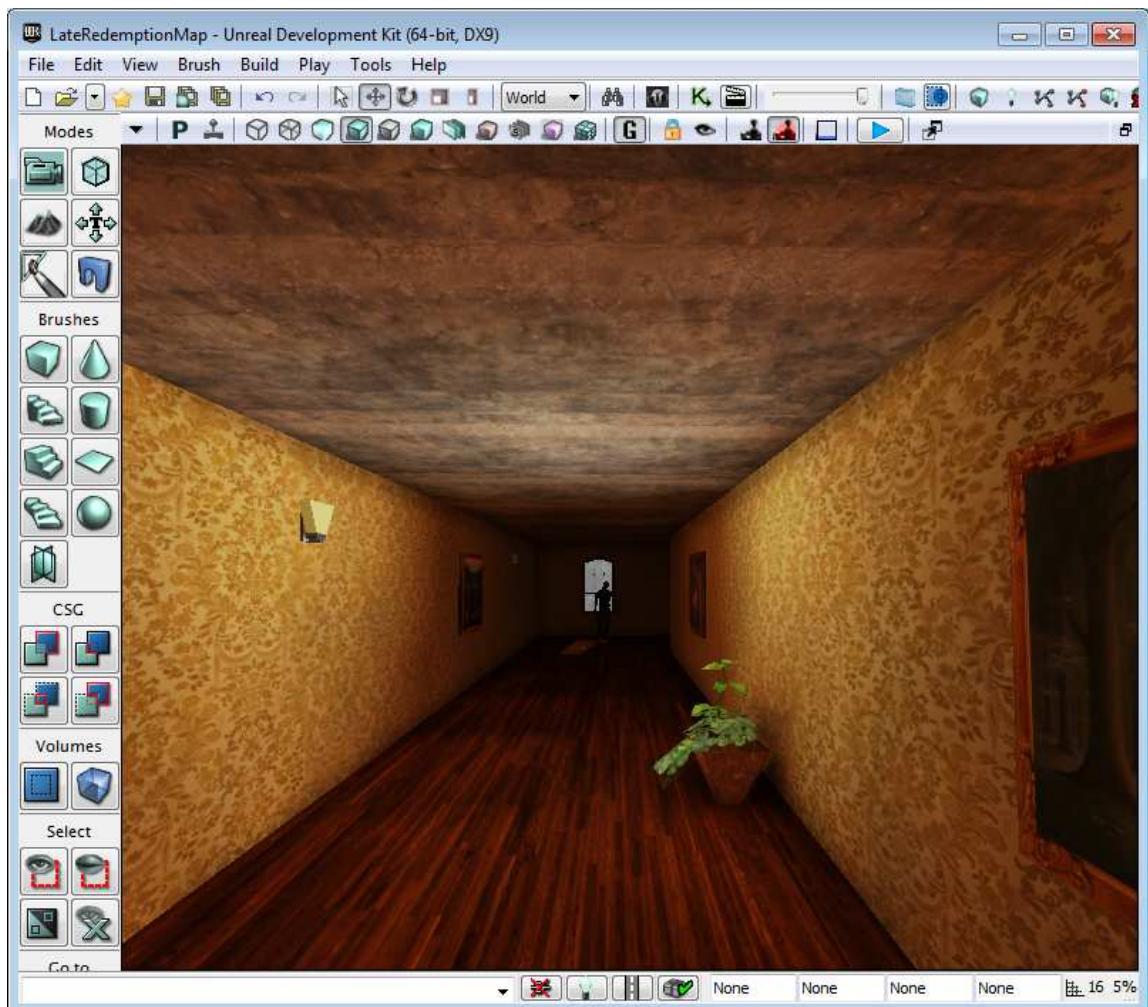
As imagens neste tópico exibem algumas partes do cenário em sua versão atual para a primeira demo.



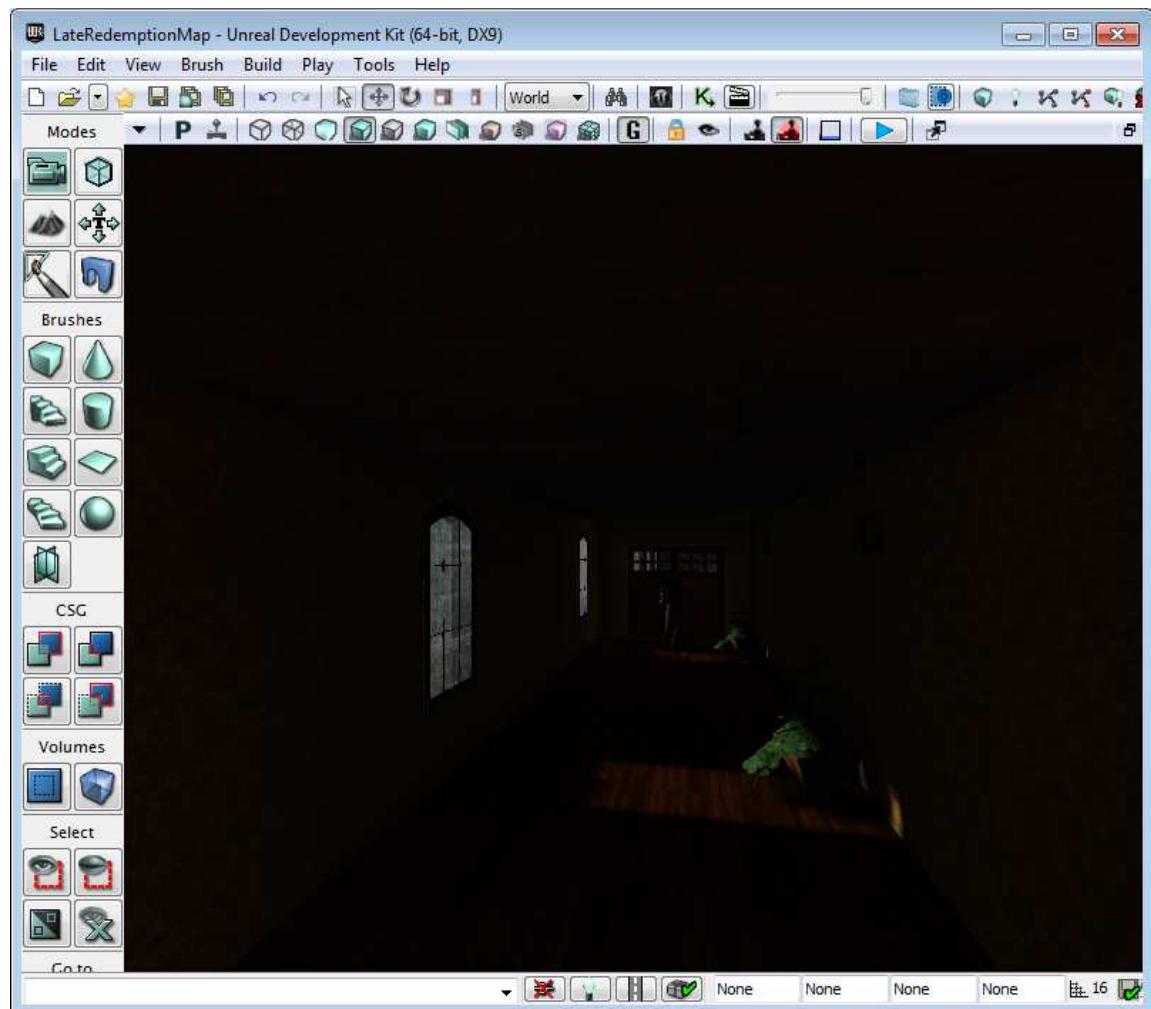
**Figura 8.49 – Visualização da sala onde Marsall deverá resolver alguns puzzles para conseguir sair**



**Figura 8.50** – Visualização da parte inicial do cenário Corridor



**Figura 8.51 – Visualização da segunda parte do cenário Corridor**



**Figura 8.52 – Visualização da última parte do cenário Corridor**

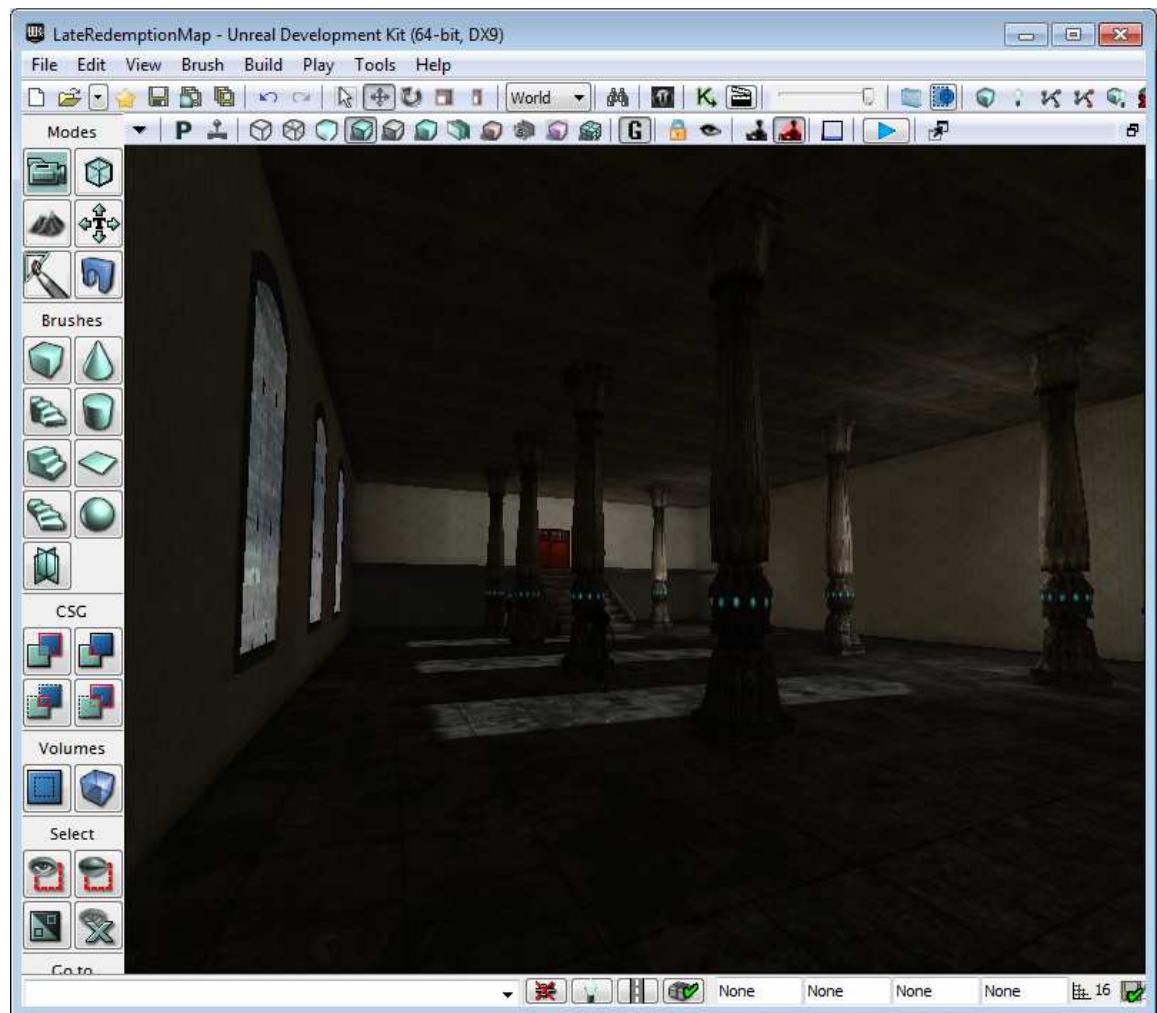


Figura 8.53 – Visualização do cenário Main Hall

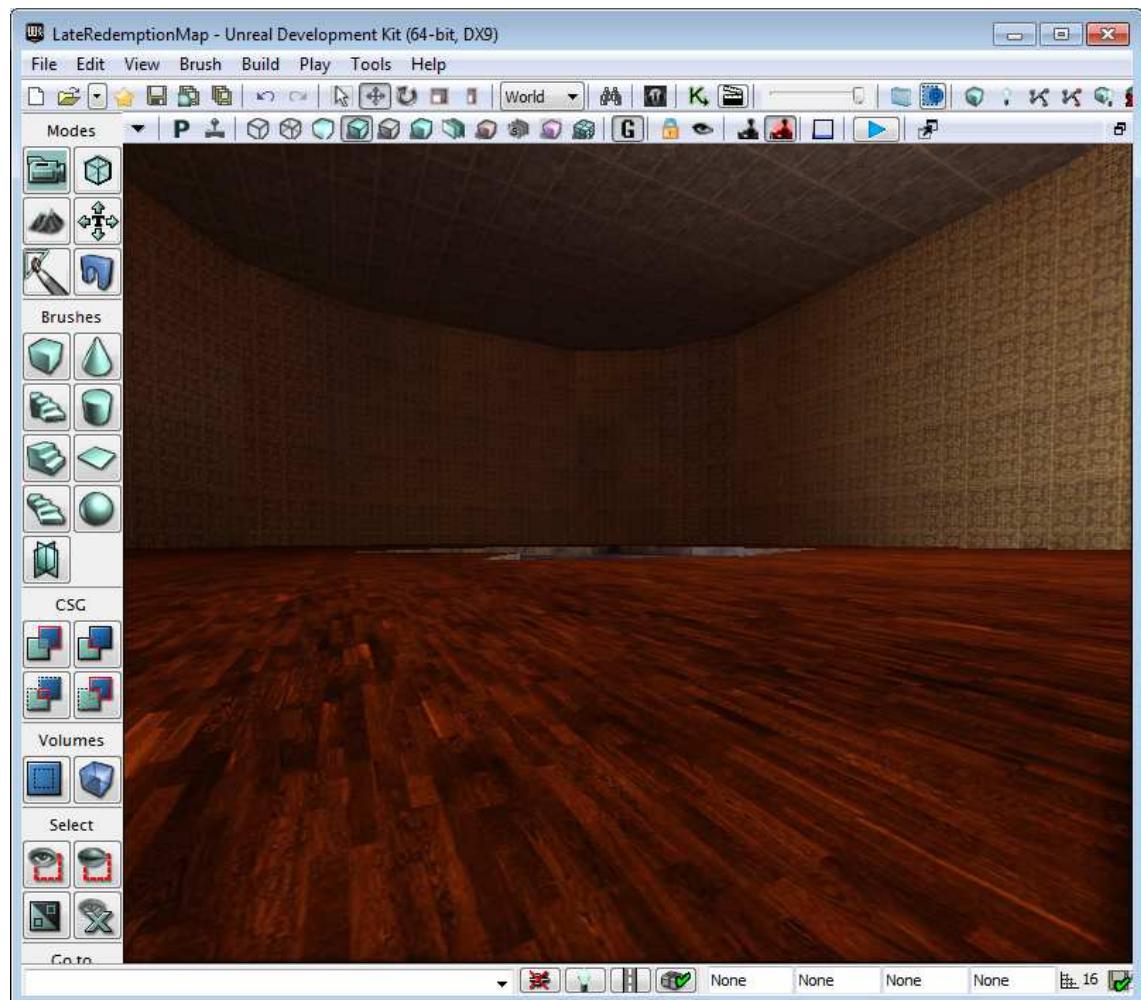
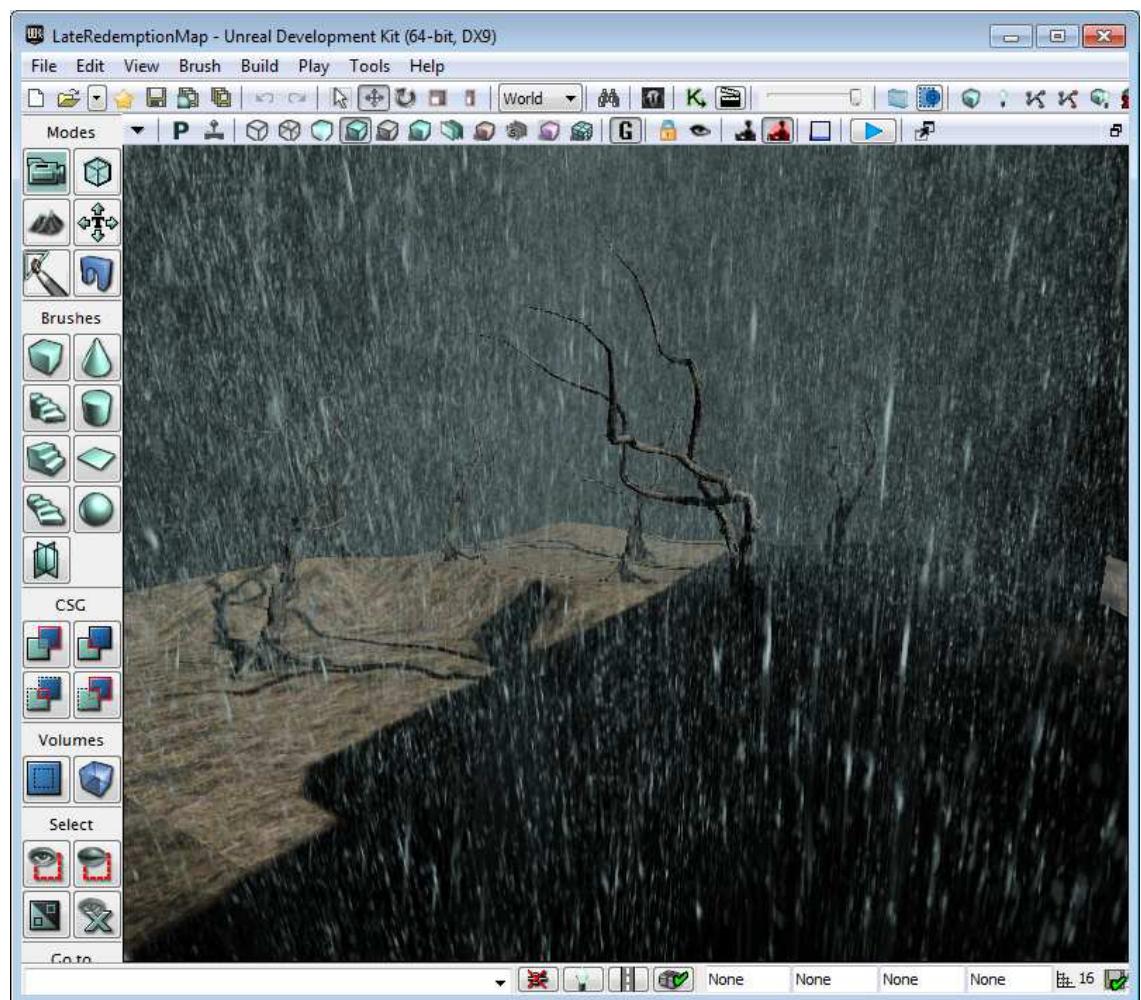


Figura 8.54 – Visualização do cenário Library



**Figura 8.55 – Visualização do ambiente externo da mansão**