Um Estudo Sobre os Processos de Desenvolvimento de Jogos Eletrônicos (Games)

ADEMAR DE SOUZA REIS JUNIOR[1]
BOGDAN T. NASSU[2]
MARCO ANTONIO JONACK[3]

UFPR - Universidade Federal do Paraná
DInf - Departamento de Informática
Curitiba, setembro de 2002

[1]ademar@ademar.org [2]nassu@matrix.com.br [3]marcojk@yahoo.com

Resumo: o desenvolvimento de jogos eletrônicos é uma área onde os processos, quando existem, são muito variados e extremamente flexíveis. Isso se dá devido à mistura de criação artística com a produção de software e o grande número de desafios e novidades tecnológicas a serem vencidos. Neste trabalho, são feitas uma contextualização e análise dos processos e conceitos existentes nesse desenvolvimento, trazendo uma introdução aos conceitos e uma visão geral sobre o assunto. São mostrados ainda estudos de caso do desenvolvimento de jogos consagrados onde é possível verificar de maneira prática como as idéias e conceitos introduzidos se aplicam.

Palavras-chave: jogos eletrônicos; games; processo de desenvolvimento.

1 Introdução

O processo de desenvolvimento de jogos eletrônicos, ou *games*, costuma ser bastante caótico e complicado. A inconstância de diversos aspectos inerentes ao desenvolvimento, como os requisitos a se satisfazer e as tecnologias disponíveis, faz com que a adoção de processos rígidos e bem definidos seja bastante dificultada, muitas vezes levando equipes inteiras a seguir processos "artesanais" ou "artísticos".

Este artigo tem como objetivo apresentar e discorrer sobre estes processos. As diferentes fases deste desenvolvimento são enfocadas, desde a concepção inicial até o produto final.

A abordagem utilizada tem como ênfase apenas os processos em si, evitando entrar em detalhes de tarefas burocráticas ou administrativas. Desta forma, são traçados paralelos diversos entre as fases do desenvolvimento, assim como os processos e técnicas adotados nelas, com as metodologias clássicas da Engenharia de Software. Todos os passos do desenvolvimento são apresentados de forma simplificada, trazendo uma visão genérica de como funcionam os processos mais utilizados, sem detalhes muito profundos e sem favorecer uma metodologia específica.

O restante do artigo é organizado como segue. A Seção 2 descreve o processo de desenvolvimento de forma ampla e global, apresentando diversos tópicos inerentes a este tema, como informações importantes que rodeiam o desenvolvimento de games, o papel e a importância do *game designer*, as fases de *design*, projeto, cronograma/orçamento, implementação , testes e manutenção. A Seção 3 traz um resumo de análises *postmortem* de quatro games conhecidos: *Thief: The Dark Project*, *Half-Life*, *Black and White* e *Baldur's Gate II: Shadows of Amn*. Finalmente, a Seção 4 conclui o artigo.

2 Processo de Desenvolvimento

Um jogo eletrônico, ou game, é uma aplicação interativa voltada ao entretenimento. É um produto de software, e como tal passa por um certo número de fases durante o seu desenvolvimento.

É possível segmentar os processo de criação de um game de forma similar à que se faz com um outro software qualquer, tendo as fases típicas da engenharia de software, como análise, projeto, implementação e teste, ou fases análogas a estas. Porém, dada a natureza subjetiva de um game, em especial dos requisitos a se satisfazer, processos rígidos e bem definidos dificilmente são utilizados. Geralmente as fases não seguem uma ordem única, entrelaçando-se durante todo desenvolvimento do produto, com algumas assumindo um papel mais significativo em certos momentos.

Pode-se afirmar que a criação de um game é um processo quase artesanal ou até mesmo artístico. Desta forma, este artigo busca apresentar algumas das práticas mais difundidas durante a produção de um game de forma geral, mas sempre tendo em vista a grande variedade de hábitos e processos existentes e a grande flexibilidade das abordagens utilizadas.

2.1 Questões Relacionadas

Antes de definir o processo de criação do game propriamente dito, alguns pontos exteriores ao desenvolvimento, mas de fundamental importância, devem ser definidos. Entre eles estão a concepção inicial, o papel dos publicadores e distribuidores, a procura por investidores, a análise do público-alvo e a escolha da plataforma sobre a qual o game irá ser executado.

2.1.1 Concepção Inicial

O primeiro passo que se dá quando na criação de um game é a concepção. Esta é a idéia básica por trás do game, a sua premissa inicial, que pode levar ao seu real desenvolvimento. Esta primeira concepção pode vir de diversas fontes: da necessidade que uma empresa possui de desenvolver algum produto que traga lucros a um sonho de realização pessoal de um desenvolvedor visionário.

Independente da motivação, o conceito do game deve então ser trabalhado até que este se torne um projeto real. A idéia pode ser levada adiante por um time de desenvolvimento, por uma empresa ou mesmo por um "pequeno grupo de amigos na garagem da casa de um deles". Deve-se ter em mente que nos dias atuais a produção de um game pode custar milhões de dólares e exigir uma equipe bastante numerosa ou talentosa. Desta forma, empreitadas individuais ou

amadoras podem exigir uma dedicação extrema, talvez muito maior que aquela que se pode ou se deseja despender à produção do game.

2.1.2 Publicadores e Distribuidores

Uma vez concluído o processo de desenvolvimento do game, o mesmo ainda não está pronto como produto. Para isto, é necessário o trabalho de um publicador, assim como o de uma distribuidora. O publicador é o responsável pela replicação em larga escala do software, pela impressão dos manuais e caixas, por parte da publicidade e às vezes pela sua localização (adaptação para comercialização no exterior), em outras palavras, pela transformação do game em um produto. A distribuidora é responsável pela comercialização e distribuição do game, e também por parte da publicidade.

Existem empresas especializadas neste tipo de atividade, e a escolha correta de um publicador ou distribuidor pode ser crucial para o sucesso comercial de um game. Em geral, os contratos com publicadores e distribuidoras são feitos já durante os estágios iniciais de seu processo de desenvolvimento. É comum que estes contratos pareçam muito desvantajosos para o estúdio desenvolvedor, com ganhos médios de algo em torno de 10% dos lucros totais.

Muitas vezes, uma mesma empresa desempenha o papel de publicador e distribuidor simultaneamente, oferecendo apenas propostas que englobem estas duas atividades. Alguns publicadores contratam estúdios desenvolvedores para criar games específicos, enquanto outros possuem estúdios internos.

2.1.3 Investidores

Devido a uma série de fatores, entre eles os já citados contratos com margens de lucro proporcionalmente pequenas, é comum que exista entre os desenvolvedores a necessidade de investimentos vindos de outras empresas ou fontes. As negociações feitas para obtenção destes recursos envolvem reuniões com executivos, onde se apresentam conceitos, idéias e versões prototipais ou demonstrativas do game. Na maior parte das vezes, os publicadores e as distribuidoras estão entre estes investidores.

A realidade no Brasil é bastante dura neste ponto. Existe uma grande dificuldade em se obter apoio financeiro por parte das empresas, que se mostram muito céticas e cautelosas a respeito da produção de games, uma atividade que não possui tradição no país. Existem louváveis iniciativas de incentivo que podem ser verificadas em alguns lugares, como aquelas tomadas pelas incubadoras tecnológicas, mas os desenvolvedores ainda são responsáveis pela injeção de uma quase totalidade dos recursos necessários à sua atividade.

2.1.4 Público-Alvo

Quando se desenvolve um game, deve-se fixar um público-alvo para o mesmo. A idade média do consumidor de jogos eletrônicos mudou bastante nos últimos tempos, e atualmente a maioria dos usuários tem idade entre 16 e 25 anos. A maior parte dos compradores efetivos possuem idade superior a 18 anos. Estas estatísticas são contrárias à idéia popular de que games são direcionados ao público infantil, o que era verdade há alguns anos. Também é verificado que nesta faixa etária se encontram os jogadores mais dedicados, e que os gostos deste tipo de

jogador (dito *hardcore*) em geral são bastante diferentes dos jogadores mais "casuais". Em geral, os primeiros possuem uma preferência por games complexos e desafiadores, enquanto os demais priorizam diversões mais leves e simples.

2.1.5 Plataforma

A plataforma de hardware sobre a qual o game será executado é bastante importante. Embora seja comum que muitos games estejam disponíveis para mais de uma plataforma, em geral eles são desenvolvidos tendo como alvo apenas a uma delas, sendo adaptados posteriormente. As plataformas possíveis são muitas, englobado os diversos modelos de consoles de *videogame*, computadores pessoais ou placas de arcade (estas últimas cada vez mais raras).

No caso de consoles de videogame ou placas de arcade, o desenvolvimento é feito com ferramentas especializadas, disponibilizadas em geral pelo fabricante do aparelho. Licenças especiais muitas vezes são necessárias, e o conhecimento do hardware é imprescindível. Mesmo que em todos os casos o game seja um produto *off-the-shelf*, fechado, não direcionado a um único cliente específico, games para consoles de videogame não podem ser atualizados com o uso de correções (*patches*), e só podem ser lançados após um extensivo processo de testes que garanta sua qualidade, evitando assim a descoberta de erros (*bugs*) após o seu lançamento. O desenvolvimento de games para PC é mais flexível, não estando "preso" às limitações de um hardware específico, mas este fato acaba por dificultar e estender o processo de testes, muitas vezes levando ao lançamento de diversos patches após o lançamento do game.

2.2 O Game Designer

Se existe um papel de fundamental importância no desenvolvimento de um game, é o desempenhado pelo assim chamado *game designer* [6]. Este profissional está presente na grande maioria das equipes de desenvolvimento de games, e mesmo quando não há um game designer específico, sua função é executada de alguma forma pela equipe.

O game designer possui diversas funções: redigir a documentação de design, desenhar a interface do usuário, designar objetivos da lógica e do game, escrever textos de diálogos ou da estória do mesmo, pesquisar e desenvolver algoritmos e tabelas e participar dos processos de refinamento, teste e integração das partes do software [1, 2, 3]. O papel de criar e ter novas idéias que façam parte da concepção inicial do game não é necessariamente uma tarefa para o game designer, embora este muitas vezes seja responsável por idéias que levem o game a atingir seus objetivos iniciais. O seu papel pode ser visto como uma soma de gerente técnico do projeto, de roteirista e diretor de cinema e produtor musical.

2.2.1 Habilidade de Escrita

Segundo [6], a habilidade mais importante que um game designer deve possuir é a de escrever muito bem. Ele deve ser capaz de comunicar idéias complexas entre pessoas de diferentes formações de forma clara e objetiva. Os textos que surgem no decorrer do game e a estória contada no mesmo, quando presentes, devem ser consistentes e lógicos. Além disso, para se escrever bem é necessária uma certa dose de inteligência, organização e clareza de pensamento, e estas são características bastante desejáveis para um game designer.

2.2.2 Conhecimentos Gerais

O game designer deve possuir conhecimentos amplos em diversas áreas. Quando desenvolvendo um game com um determinado tema, o seu conhecimento sobre este deve ser o suficiente para que não sejam cometidos erros históricos ou de consistência. Por exemplo: ele deve ter um conhecimento razoável sobre história, táticas e estratégias militares para poder desenvolver um game que tenha como tema a Segunda Guerra Mundial. Do contrário, podem surgir situações absurdas ou inconsistentes que poderiam impedir ou dificultar a imersão do jogador.

Além disso, o game designer deve possuir conhecimentos gerais em áreas comuns que são encontradas na maioria dos games, como artes gráficas, música, cinema, história e outras áreas diversas; dado o seu papel como integrador dos componentes que fazem parte do game. Os melhores game designers são como grandes pensadores renascentistas, com inúmeros interesses e habilidades.

2.2.3 Outras Características

Existem ainda outras características importantes que são muito desejáveis em um bom game designer.

Ele deve conhecer e dominar uma grande variedade de games e tecnologias. Seu conhecimento sobre games deve extrapolar suas preferências pessoais: ele deve ter uma visão clara de como os produtos atuais funcionam, quais os fatores responsáveis por seu sucesso ou fracasso, e qual é o estado da arte em games de estilos diversos. Mais que conhecer os games que lhe agradam, um bom game designer deve procurar conhecer aquilo que agrada ao consumidor.

O game designer também deve saber, mais que qualquer outro profissional envolvido no processo de desenvolvimento, trabalhar em equipe. Ele não deve apenas cooperar com outros profissionais, mas servir como elo de ligação entre pessoas com conhecimentos e maneiras de pensar diferentes.

Um grande conhecimento técnico na área de programação também é fundamental, pois muitas vezes o game designer deve apresentar idéias inovadoras e maneiras de implementá-las. Além disso, ele deve estar ciente das capacidades e propriedades dos equipamentos (hardware) e softwares atuais, de forma a aproveitá-los da melhor maneira possível.

Por fim, saber aceitar o mercado é preciso. Grandes idéias e apostas nem sempre são bem sucedidas. O game designer deve saber quando "jogar" com as oportunidades, admitir o abandono de idéias que lhe agradam mas que são muito incertas, e muitas vezes esquecer a originalidade em favor de uma idéia mais segura. As grandes companhias em geral gastam algo em torno de 90% dos seus recursos replicando idéias bem sucedidas e apenas 10% em idéias inovadoras. Muitas vezes, acrescentar pequenas inovações a um modelo já existente é mais seguro que criar um modelo "revolucionário". Porém, apostar em novas idéias na hora certa pode ser exatamente o caminho para o sucesso, e saber quando e no que apostar é algo que o game designer deve fazer com habilidade.

2.3 Design (Análise)

O design de um game é equivalente à fase de análise durante o desenvolvimento de um sistema de software qualquer, quando são definidas as características e requisitos do produto. A maior

diferença que existe entre uma fase de análise "clássica" e o design de um game é a imensa dificuldade que se tem em coletar os requisitos do sistema neste último, em especial nos estágios iniciais do desenvolvimento, dada a sua subjetividade e as constantes mudanças e adaptações causadas pelo avanço tecnológico ou pelo lançamento de outros games. Desta forma, processos rígidos e bem-definidos são abandonados, dando lugar a técnicas e diretivas flexíveis e abertos a adaptações. Em geral, o design é testado e os requisitos são coletados durante todo o processo de desenvolvimento do game, permitindo que mudanças sejam feitas durante o tempo em que se está trabalhando sobre o mesmo. Porém, deve-se lembrar que em geral mudanças profundas ou complexas devem ser detectadas e executadas logo nos estágios iniciais da produção, ou o desenvolvimento pode ser prejudicado por atrasos e bugs.

Independente do processo adotado, a fase de design deve elucidar algumas questões [7], como por exemplo:

- Quais são as tecnologias atualmente em uso?
- O que o mercado está buscando?
- Quais são as ferramentas acessíveis?
- Qual é o estilo visual e musical da atualidade?
- O que já foi feito e como foi feito?
- Como é a interface com o jogador?
- Qual é o ritmo do game?
- Qual a dificuldade do game?
- O que pode ser reaproveitado?
- Como é a estória do game? Como são suas fases e estágios?
- Quais são as características mais importantes do game?

A união da concepção inicial do game com as respostas para perguntas como estas é capaz de definir, em um nível mais alto, a maior parte das características que se espera que o game tenha, ou seja, como é de fato o game que se está desenvolvendo. Durante o processo de produção, outras questões surgirão, assim como a necessidade de se detalhar certos requisitos [8, 9].

2.3.1 Proposta

O primeiro tipo de documentação que normalmente é escrita é a proposta do game. Esta proposta serve como guia para a criação de outros tipos de documentação e para uma primeira análise por parte de investidores e executivos. Normalmente, a proposta de um game apresenta:

Introdução

Estória/motivação

Descrição

Características chave
Gênero (tipo de game)
Plataforma(s)
Arte conceitual

2.3.2 Documentação Funcional

Geralmente, a descrição do game é documentada de forma rica e detalhada, num documento denominado documentação funcional [17, 18]. Este documento costuma ser usado tanto para descrever os requisitos que devem ser satisfeitos pelo projeto do game quanto para apresentar o trabalho que se está desenvolvendo a investidores e diretores executivos. Uma boa documentação funcional de design deve ser simples de entender, clara e direta, possuindo uma abordagem descritiva, dizendo o que o game faz, mas evitando detalhes técnicos de implementação. As seguintes características do game costumam ser descritas (as informações entre parênteses exemplificam o tipo de informação que pode ser encontrado quando tratando da característica apresentada):

- Funcionamento do Game (jogabilidade, fluxo de game, personagens, elementos de interação, física e estatísticas, comportamento da inteligência artificial e as regras gerais).
- Interface com o Usuário (requisitos funcionais, menus, janelas e opções).
- Arte e Vídeo (objetivos e estilo, arte, animação, vídeo e cenas pré- renderizadas).
- Som e Música (objetivos e estilo, efeitos sonoros, situações, músicas, temas e trilhas sonoras).
- Estória (sinopse, *storyboards* e roteiro).
- Fases e Estágios (ligação entre as diversas áreas do game, objetivos e características).

Vale ressaltar que estas são apenas linhas genéricas, que dão uma idéia do conteúdo que normalmente é encontrado na documentação de design . A estrutura, o formato e o tamanho da documentação varia de acordo com o projeto e o desenvolvedor. De fato, alguns fazem uso apenas de documentos mais simplificados, e alguns até mesmo desenvolvem games de grande sucesso comercial sem redigir documentação alguma. Porém, como ensina a engenharia de software, uma boa documentação pode ser uma ferramenta poderosa e até mesmo necessária, em especial ao se levar em conta o nível de complexidade atingido pelos games atualmente.

2.3.3 Linhas Gerais para um Bom Design

Em [16, 19] são descritos diversos princípios que podem ser seguidos para que se tenha um bom design em um game. Entre estes, o mais importante é: “Não existe uma fórmula mágica para o sucesso”. Os elementos que fazem um game ser bem sucedido são muitos e variados, e muitas vezes até mesmo uma questão de “sorte”. Porém, existem algumas idéias que podem ser seguidas de forma a maximizar as chances de sucesso de um game. Estas idéias não são leis, mas conselhos, muitas vezes sendo compartilhadas por diversos game designers experientes. A maioria delas diz respeito ao cliente ou ao chamado *Wow Factor*, descritos adiante.

O Cliente

Além de conhecer bem o tipo de público alvo para o game, os desenvolvedores devem ter sempre em mente o fato de que o comprador deve ser atraído para o produto, agradado, "mimado".

A interface e a jogabilidade de um game são muito importantes. Não se deve frustrar ou entediar os jogadores com dificuldades extremas (exceto quando existe uma boa recompensa por trás delas), oponentes que quebram as regras, controles confusos ou sem resposta, enigmas indecifráveis, repetições constantes, situações absurdas ou outras falhas que façam com que a diversão do jogador se perca.

Os desenvolvedores devem ouvir os clientes, conhecer os seus anseios e desejos. Como ocorre com outros produtos *off-the-shelf*, não existe um único cliente, mas um grande grupo; e os desenvolvedores devem estar cientes de que nem todos os possíveis compradores do seu produto utilizarão os mesmos canais de comunicação. Por exemplo, a maioria dos opinadores em fóruns e salas de bate-papo na Internet é formada por jovens e jogadores assíduos (*hardcore*), com opiniões que muitas vezes não representam as do público em geral. Além disso, um bom game designer deve saber quando ouvir e quando ignorar as opiniões de seu público, muitas vezes reservando surpresas agradáveis e bem-vindas. Ou seja, um bom game designer muitas vezes deve saber mais sobre o seu cliente que o próprio cliente, e confiar neste conhecimento.

Um bom exemplo da ocorrência deste tipo de situação pode ser visto na maior feira de games do mundo, a E3[1], em 2002. Shigeru Miyamoto, game designer de grande renome, responsável por games considerados "clássicos", como a série Super Mario Bros., vinha recebendo diversas críticas por adotar uma nova tecnologia gráfica no game em que vinha trabalhando, um novo capítulo para a série Zelda. Esta tecnologia, baseada no algoritmo chamado *cell shading*, faz com que os gráficos se pareçam com um desenho animado, e muitos consumidores exigiam gráficos de aspecto mais realista, afirmando que o novo game em desenvolvimento parecia "infantil" e "estranho". Após a apresentação de uma versão demonstrativa jogável durante a feira, a opinião dos jogadores mudou radicalmente, tendo estes então admitido que a adoção da nova tecnologia era realmente vantajosa e fazia com que o game tivesse um efeito "fantástico".

O "Wow Factor"

O termo *Wow Factor*, traduzido livremente como "Efeito Oh!", surgiu durante uma entrevista dada pelo famoso game designer John Carmack, criador de games de grande sucesso comercial como Doom e Quake. Quando perguntado sobre a característica mais importante num game, aquela que todos deveriam possuir para ter chances de se tornar um sucesso, ele mencionou um certo Wow Factor.

Esta característica poderia ser descrita como algo que é capaz de prender a atenção de qualquer jogador por um período de tempo relativamente curto, algo entre dez e trinta minutos, logo na primeira vez em que este tenha contato com o game. O objetivo deste Wow Factor é fazer com que o jogador se sinta atraído ou tenha uma boa primeira impressão do game, efetivamente fazendo com que o game se destaque entre os demais, quando este jogador o vê em feiras, versões demonstrativas, lojas ou até mesmo quando assiste a alguma outra pessoa jogando o game. Após o término deste período, a qualidade do game como um todo e o gosto pessoal do

[1]Exposição de Entretenimento Eletrônico (E3). Mostra internacional realizada todos os anos dirigida inteiramente a entretenimento e mídias interativas, incluindo games para computadores, game para consoles (*videogames*) e hadware.

jogador passam a determinar a continuidade ou não do interesse do mesmo pelo game.

O Wow Factor pode ser obtido através de gráficos de grande beleza, da interatividade, de idéias inéditas e inovadoras ou simplesmente do estilo do game, mas em geral de uma combinação destes fatores. Entrando em concordância com esta idéia, [12] afirma que a maneira mais rápida de se perder o interesse de um comprador é através da aparência e da sensação causada por um game. Em especial, quando o comprador tem poucas informações sobre um game, ele tende a ser atraído mais por aqueles que lhe parecem mais sofisticados e bonitos.

Ao se buscar o Wow Factor, deve-se levar em conta a constante evolução tecnológica que ocorre nos equipamentos usados para a execução de games. Tecnologias que sejam recentes ou desconhecidas muitas vezes se tornam mais baratas e acessíveis com o passar do tempo. Os requisitos mínimos de hardware para a execução do game evoluem rapidamente, e muitas vezes o seu design deve prever qual será o tipo hardware que será acessível após o término do seu desenvolvimento, e quais serão os recursos que poderão ser utilizados para a obtenção do "Wow Factor".

2.4 Projeto

Assim como ocorre em todos os aspectos da produção de games, na fase de projeto são adotadas técnicas e processos diversos, que variam de equipe para equipe. Na maior parte das vezes, entretanto, o projeto do game não segue qualquer padrão rígido que vise definir todos os pontos do software de forma a se obter uma implementação mais automática. Isto se deve a uma série de fatores, e entre eles pode-se destacar a busca constante por tecnologias novas e em geral desconhecidas, cuja viabilidade de uso muitas vezes só pode ser detectada após a sua implementação e integração ao ambiente do game, e a grande quantidade de alterações de design, aspecto já abordado na Seção 2.3.

Desta forma, a maior parte dos modelos de projeto usados na produção de games vai sendo construída conforme os mesmos são implementados, normalmente na forma de uma "documentação técnica", que aborda exatamente os mesmos tópicos da documentação funcional (descrita na Seção 2.3.2), mas de uma forma mais detalhada e focada na explicação dos algoritmos, estruturas de dados e detalhes de implementação. As mesmas observações feitas anteriormente sobre a documentação funcional também valem aqui: é muito comum encontrar desenvolvedores que descrevam apenas uma arquitetura geral e pouco detalhada, com explicações sobre algumas soluções e algoritmos, ou até mesmo alguns que usem apenas o "código como projeto".

2.4.1 Escolha das Tecnologias

Uma peculiaridade que se destaca no projeto de um game de forma mais notável que na maioria dos softwares é a necessidade de escolha das tecnologias. Todo software faz uso de tecnologias que buscam satisfazer da melhor forma possível os seus requisitos, mas o tipo de requisito que os games geralmente têm costuma fazer com que os desenvolvedores destes busquem constantemente usar tecnologias muito novas, muitas vezes não disponíveis comercialmente, que precisam ser implementadas pela própria equipe. Além disso, os games em geral tentam fazer o melhor uso dos recursos de hardware e muitas vezes envolvem cálculos e algoritmos complexos, o que faz com que a imensa maioria dos games seja escrita em linguagens de nível considerado mais baixo (quase sempre C, C++ e Assembler), talvez com alguma outra linguagem leve de

scripts para descrever pontos onde o desempenho não é tão crítico ou onde a complexidade excessiva pode inviabilizar o projeto.

Todos estes fatores fazem com que alguns desenvolvedores licenciem o *engine* de alguns de seus games, o que facilita o acesso a algoritmos e tecnologias recentes aos desenvolvedores que por algum motivo demonstrem interesse. O licenciamento ou não de engines já existentes é uma decisão tomada em outros níveis, mas depende profundamente da fase de projeto, pois a avaliação feita deve levar em conta o custo, o tempo, e a dificuldade de produção de um engine próprio.

2.5 Cronograma e Orçamento

Quando se desenvolve um game, a criação de um cronograma e de um orçamento muitas vezes é necessária [4]. Mesmo quando isto não ocorre, o planejamento pode vir a ser bastante útil. Além de servir como um plano a ser seguido pelo próprio estúdio desenvolvedor, publicadores e investidores podem exigir que se apresentem cronogramas detalhados que possam ajudá-los a tomar decisões importantes e a ter uma melhor idéia do escopo do projeto. As pessoas responsáveis pela produção destes planos são em geral o produtor (que pode ser visto como um gerente administrativo) e o game designer.

Deve-se notar que muitas das técnicas clássicas da engenharia de software simplesmente não podem ser aproveitadas aqui. O desenvolvimento de games envolve a criação de estórias, arte gráfica e música, assim como sua integração ao ambiente do game, atividades que não possuem relevância para a maioria dos projetos de software, mas que demandam boa parte do tempo e dos recursos de um estúdio que desenvolve games.

2.5.1 Cronograma

Apesar da grande variedade de possíveis técnicas que podem ser utilizadas para se criar um cronograma, este deve responder a algumas questões básicas, como:

- O que deve ser feito? Para responder a esta questão, é preciso que já se tenha uma boa idéia sobre os componentes e o funcionamento do game, provavelmente com uma versão da documentação funcional já disponível, assim como as tecnologias que serão usadas. O número de fases do game, efeitos sonoros, menus, bibliotecas e outros aspectos também são importantes. Além disso, é necessário que se saiba quais os equipamentos que estão disponíveis e quais se deve adquirir, onde será localizado o escritório e o laboratório, qual a disponibilidade de linhas telefônicas e outras questões ligadas à infra- estrutura da empresa.

- Quem irá fazê-lo? Esta decisão é tomada com base na questão anterior. Por exemplo, se o game irá precisar de uma grande quantidade de desenhos feitos a mão, profissionais especializados devem estar disponíveis em uma proporção adequada.

- Quais são os recursos necessários ao trabalho? Esta questão envolve o custo de cada profissional envolvido no projeto (salário, espaço físico, equipamentos, etc.), os gastos com advogados, contratos e processos legais e o custo de licenças e compras de bibliotecas, ferramentas e pacotes de software que devem ser adquiridos.

- Quando deve ser feito? Cada profissional deve seguir um cronograma específico. As datas de conclusão e entrega de trabalhos devem ser seguidos. O atraso de uma única pessoa pode afetar todo o time.

2.5.2 Orçamento

Utilizando-se o cronograma como base, deve-se gerar um orçamento, prevendo os valores em moeda que se farão necessários e as datas em que eles devem estar disponíveis. Novamente, existem muitas abordagens possíveis para o problema. Seja qual for a adotada, ao fazer um orçamento, o desenvolvedor deve buscar uma medida que equilibre a qualidade do trabalho, o tempo necessário, o escopo do projeto e o custo total, dadas as condições do estúdio e do ambiente.

2.6 Implementação

Assim como ocorre nas demais fases, a implementação de um game tem algumas características particulares que a diferem do desenvolvimento de software tradicional. Por uma questão de organização, o processo de implementação será divido em três partes: codificação, testes e manutenção. Uma vez que a criação de um game muito se aproxima de uma produção artística, tal divisão pode não existir em alguns projetos.

2.7 Codificação

O processo de codificação de um game em geral exige grandes conhecimentos e capacidade técnica por parte do programador. Algumas características específicas exigidas de um programador de games são: capacidade de se adaptar facilmente a novas tecnologias, trabalhar em projetos não detalhados, otimizar rotinas e, principalmente, lidar bem com outros profissionais envolvidos no projeto, como artistas, designers e escritores.

Não se pode dizer que a programação de games é um processo “automatizável”. Os algoritmos em geral apresentam grandes desafios, a otimização de rotinas complexas sempre se mostra necessária e a utilização de linguagens com baixo nível de abstração é comum (na maioria absoluta dos casos o desenvolvimento é feito em linguagens como C, C++ e Assembler). Além disso, o uso de tecnologias de ponta faz da área de desenvolvimento de jogos um verdadeiro laboratório, onde os programadores freqüentemente são obrigados a descartar boa parte de seu trabalho por não conseguirem alcançar um resultado satisfatório.

Pode-se listar algumas características da equipe de programadores como sendo bastante relevantes para o sucesso de um game:

- Integração com artistas e designers;
- Conhecimento tecnológico;
- Criatividade;
- Grande capacidade de adaptação;

- Gosto pelo que se faz.

2.8 Testes

O processo de testes de um game envolve não apenas a procura por falhas, mas também a jogabilidade e aceitação do jogo por parte de usuários. Por esse motivo, os testes geralmente são distribuídos entre todas as fases de desenvolvimento.

Os testes de funcionalidade e *bugfix* em geral possuem as características das abordagens clássicas da Engenharia de Software. Existe porém uma classe de testes que é particular ao desenvolvimento de games: o chamado *playtest*.

É no playtest que os programadores analisam a aceitação do game e a reação dos jogadores. Um playtest pode ser aberto (disponibiliza-se uma versão do jogo para download ou marca-se um dia e local para testes) ou fechado, onde testadores previamente selecionados (existem os chamados "testadores profissionais") são utilizados. O diferencial do uso destes profissionais é que além de simplesmente apontar o problema, eles são capazes de dar informações mais detalhadas e sugestões que auxiliam em muito a correção.

As abordagens de teste variam conforme a equipe e fase de desenvolvimento, sendo que as mais conhecidas e utilizadas são:

Usuários jogam enquanto programador apenas observa. Nessa abordagem, o programador não tem contato algum com o jogador. Ele apenas mantém-se nas proximidades fazendo anotações das reações e dificuldades que o jogador encontra durante a seção de testes. Essa metodologia se mostra eficiente na descoberta de dificuldades de jogabilidade e de partes tediosas de um game.

Usuários jogam enquanto programador faz perguntas. Conforme o jogador vai avançando no jogo, o programador vai lhe questionando sobre aspectos do jogo, dificuldades e emoção. Funciona bem para jogos onde a ação seja constante e dinâmica, uma vez que o programador não teria tempo de captar todas as reações do jogador se não fizesse perguntas.

Usuários jogam e fazem relatório. Muito utilizada em playtests de grande porte (como os via internet), essa abordagem exige que o usuário jogue e faça um relatório do que achou do jogo. Nesse modelo se encaixam bem os "testadores profissionais", que já estão acostumados a avaliar jogos. Durante o processo de testes, os desenvolvedores devem estar a par do problema de vazamento de informações. Afinal de contas, o "efeito oh!", "aquela" surpresa, o "segredo do sucesso" e outros detalhes nem sempre devem ser divulgados com antecedência. É comum acontecer de versões piratas "não oficiais" de games serem disponibilizados na Internet meses antes de seu lançamento, fruto de cópias de seções de testes. Tal ocorrência pode trazer grandes problemas para o sucesso de lançamento do game e gerar transtornos com incompatibilidades nos jogos *on-line*.

2.9 Manutenção

Um game não faz sucesso se seus jogadores não o puderem jogar. Por esse motivo é necessário que, além de ter boa jogabilidade e trazer entretenimento, um jogo seja simples de instalar,

compatível com os ambientes utilizados pelo público alvo e funcione sem problemas. É extremamente frustrante para um jogador ter horas de jogo perdido devido a um problema bug no software, assim como adquirir o jogo e não conseguir fazê-lo funcionar (ou mesmo instalar) adequadamente. No caso de games para consoles, não existem muitas alternativas para a atualização. Felizmente essas plataformas dificilmente apresentam problemas de compatibilidade e uma boa e extensiva fase de testes geralmente é suficiente para evitar a frustração do lançamento de um game com problemas. Já com os games para computadores pessoais, ocorre de outra forma. Por mais que se façam testes, problemas acabam por aparecer dada a variedade do hardware e software em utilização. O lançamento de atualizações com correções é uma prática extremamente comum neste ambiente. É difícil encontrar um jogo de sucesso que não tenha passado por alguns pares de atualizações (geralmente entre 5 e 10 atualizações são o suficiente para corrigir problemas de compatibilidade e correção de bugs gerais). A criação de FAQs e a disponibilização de recomendações e *drivers* é uma prática que ajuda muito no processo de suporte a usuários com problemas e tem sido utilizada extensamente.

3 Estudos de Caso

De forma a demonstrar o funcionamento do processo de criação de games como ocorre no "mundo real", esta seção apresenta um resumo de análises postmortem (análise feita após o lançamento de um produto) de quatro games conhecidos: *Thief: The Dark Project*, *Half-Life*, *Black and White* e *Baldur's Gate II - Shadows of Amn*. O objetivo não é avaliar a qualidade do game como produto de entretenimento, mas sim abordar a história da produção do mesmo.

Todos os games aqui apresentados demonstraram um razoável sucesso comercial e de crítica. Um bom complemento a estas análises seria o estudo de casos de fracasso comercial, de previsões frustradas ou de games que simplesmente não atingiram todos os seus objetivos. Compreender os motivos do fracasso pode ser tão importante quanto compreender os motivos do sucesso.

3.1 Thief: The Dark Project

Produzido pela Looking Glass Studios, *Thief* foi criado com a proposta de ser um game de ação em primeira pessoa [13], provendo uma experiência totalmente diferente nesta categoria de games, já bastante conhecida na época.

A forma com que o jogador desempenha seu papel em Thief desafia a tradicional forma de jogo dos games de ação 3D, onde os jogadores detém velocidade e desejo de conflito fora do comum. Um jogador experiente de Thief move-se vagarosamente, evita conflitos, sofre penalização ao matar pessoas e é totalmente mortal.

Este estilo de jogo não foi bem visto por muitos "observadores", que estavam preocupados com a não aceitação do game por parte do público, causando dúvidas até mesmo sobre as pessoas intimamente ligadas ao projeto.

3.1.1 O Período de Desenvolvimento

O projeto do game Thief começou em março de 1996 como *Dark Camelot*, um game de combate de espadas com elementos de RPG (*Role-Playing Game*) e aventura, baseado em uma inversão da lenda do rei Arthur. Em 1997 o game foi reposicionado como um jogo de ação/aventura dentro de um cenário sombrio e de fantasia. Até este momento, já existia uma pequena porção de material produzido que poderia tornar o game comercializável.

O desenvolvimento real sobre Thief começou em maio de 1997 com um time em grande parte diferente do time original. Durante o ano seguinte, foi criada uma grande quantidade de código, arte e design de qualidade.

Em meados de junho de 1998, vários problemas começaram a aparecer. O time, já exausto, achava que o game não podia ser chamado de "divertido" e ainda enfrentava o ceticismo do publicador do game.

Segundo a filosofia de desenvolvimento da produtora do game Thief, games imersivos vêm de um mundo rico em objetos regidos por alta qualidade e sistemas de simulação alto-consistentes. Tais sistemas tomam um considerável tempo de desenvolvimento e requerem muitos ajustes pontuais. O sistema de simulação para o game Thief ficou pronto quinze meses depois do início do desenvolvimento, a três meses antes da data prevista para a comercialização do game.

Quando Thief finalmente ficou pronto, o time percebeu que tinha produzido um bom game e que realmente poderia ser divertido. Também havia a motivação criada pelo lançamento de outros games com estilos similares ao de Thief (tais como Metal Gear Solid e Commandos) e jogos de ação em primeira pessoa mais ricos em conteúdo (por exemplo Half-Life).

Em muitos aspectos, Thief foi um projeto típico: teve problemas desafiadores, uma equipe talentosa, espaço para expressão criativa e algumas vezes bugs hilários. Thief também teve problemas comuns a projetos de software: subestimação de tarefas, surtos de baixa moral, "fluxos" de demos para o lixo, um cronograma absolutamente não realístico, documentação pobre e especificação insuficiente.

Embora muitos dos riscos assumidos pelo projeto de Thief (como a natureza experimental assumida pelo game e a proposta de criar componentes reutilizáveis) pudessem tê-lo levado ao não lançamento, a dedicação e sacrifício de todas as pessoas envolvidas fizeram com que o projeto chegasse ao seu final.

3.1.2 Casos de Sucesso no Desenvolvimento de Thief

1) O desenvolvimento de ferramentas orientadas a dados

A experiência dos líderes do projeto de Thief com outros games lhes dizia que um dos problema com o desenvolvimento de games que não seguem o cronograma previsto é a dependência mútua de artistas, designers e programadores nos vários estágios do desenvolvimento. Logo um dos objetivos de desenvolvimento da *Dark Engine*, sobre a qual Thief foi construído, foi criar um conjunto de ferramentas que permitisse aos diversos profissionais envolvidos trabalhar de maneira mais efetiva e independente. O foco deste esforço era fazer um game altamente orientado a dados e dar aos integrantes sem conhecimentos de programação um grande controle sobre a integração de seu trabalho, permitindo que elementos do game fossem facilmente conectados sem o envolvimento direto de programadores.

A abordagem da ferramentas orientadas a dados funcionou de maneira tão eficaz no desenvolvimento de Thief que algumas das ferramentas criadas foram utilizadas em outros games produzidos pela Looking Glass.

2) O som como um dos focos do projeto

O som tem um papel central no estilo de jogo de Thief, tanto enriquecendo o ambiente como sendo parte integral do próprio game.

Como definido no design de Thief, o som tem duas funções principais. Na primeira, o som é um meio através do qual os personagens do game comunicam sua localização e estado ao jogador. A segunda função do som é informar aos personagens do game, através dos ruídos gerados por objetos, detalhes sobre o que os cerca.

Tudo isso exigiu muito trabalho à equipe de Thief, que necessitava implementar um sistema de som significativamente mais sofisticado que muitos outros games da época. No final da construção do game, o som se tornou uma interface até mais importante que os gráficos produzido em Thief.

3) O foco do game

No início do design de Thief, o game deveria ter uma séria de características e elementos, como várias ferramentas para o jogador e muitos modos de jogo, que foram todos descartados. O resultado disto foi a produção de um game sem suporte a múltiplos jogadores e com missões lineares baseadas apenas em torno da "ladroagem".

O time todo concordou que estas decisão foram as melhores para manter a coerência com o estilo do game.

4) Os objetivos e dificuldades das fases do game

Impressionados com a forma de desenvolvimento dos níveis de dificuldade no game Goldeneye para Nintendo 64, a equipe de Thief decidiu estender estas idéias para o próprio Thief. Eles adicionaram a noção de quando a dificuldade aumenta, a tolerância pelo "assassinato" de humanos diminuía. O time também modificou o game para permitir que o jogador altere o nível de dificuldade no início de cada missão.

Estas características foram bem sucedidas por duas razões. Primeiro, o game deixa claro ao jogador o significado de dificuldade. Segundo, elas permitiram aos designers criar diferentes experiências em cada nível de dificuldade sem alterar a geometria e estrutura geral da missão. Isto deu ao game um alto nível de jogabilidade a um custo mínimo de desenvolvimento.

5) As soluções por ferramentas de script específicas

Para ter uma maior flexibilidade, o game Thief necessitava de uma linguagem de *scripting* para definir completamente o comportamento dos objetos. Ao invés de criar um sistema de descrição único, o time de Thief criou uma série de ferramentas específicas para o processamento de scripts.

O desenvolvimento de tais ferramentas permitiu aos designers criar vários ambientes e comportamento de objetos sem alterar o código do game.

3.1.3 Problemas no Desenvolvimento de Thief

1) A Inteligência Artificial

Um dos motivos para a demora na finalização do game foi uma série de problemas com a inteligência artificial (AI) presente em Thief.

A AI original de Thief foi desenvolvida por um outro programador antes do game ter os seus novos requisitos baseando em "ladroagem" completamente especificados. Apenas seis meses antes da data prevista para a comercialização do game e é que o trabalho sobre a nova AI tomou tempo integral, ainda com poucos elementos da antiga AI, ficando pronta para testes vinte semanas depois. Antes disso, o time acreditava que teria a AI terminada para uso real entre três a cinco meses antes da data de lançamento do game.

2) Um renderizador desatualizado

O núcleo básico do renderizador utilizado em Thief foi desenvolvido em 1995 por Sean Barrett. No final de 1996, Barret decidiu deixar a produtora Looking Glass. Embora o time do game Thief estivesse apto a fazer pequenas modificação no renderizador, a falta de um programador com conhecimento específico para para este componente fez com que o game não atingisse o estado da arte dos gráficos de outros games lançados no ano de 1998. Ainda assim, o renderizador se mostrou apropriado ao estilo de jogo de Thief.

3) A perda de pessoal

Quando o projeto de Thief ficou realmente definido na metade do ano de 1997, o estúdio produtor Looking Glass estava enfrentando uma crise financeira. O time de Thief foi tomado pelo desânimo e incerteza da continuidade de seu trabalho. Sobre estas condições, alguns membros importantes da equipe abandonaram o projeto. Alguns meses se passaram até que o time restante recuperasse o ânimo e a companhia se estabilizasse. A lição que o time aprendeu de tudo isso foi que por mais desencorajadora que possa parecer uma situação, você já viu piores.

4) Problemas como o programa de edição

Dromed foi o programa da edição utilizado no projeto de Thief. Embora Dromed fornecesse as funcionalidades essenciais que o time precisava, ele era pouco documentado e como o próprio time dizia, "um editor desagradável". Dromed foi primariamente desenvolvido como um editor de demonstração quando a plataforma alvo de Thief era DOS. Como uma aplicação DOS, Dromed não tinha nenhuma das ferramentas e interfaces encontradas nas aplicações Windows, tendo que ser reconstruído para se adaptar as novas características do projeto. Esta adaptação tomou um tempo precioso do projeto e poderia ter sido substituída pelo desenvolvimento de um novo "framework" próprio para edição.

5) Planejamento inadequado

Mesmo sendo um clichê da indústria de software, o projeto foi seriamente afetado pelo inadequado planejamento de cronograma e orçamento.

Vários elementos contribuíram para as deficiências do planejamento. Durante o desenvolvimento de Dark Camelot e durante a primeira metade do projeto de Thief, foram alocadas pessoas no time do projeto antes que o design e tecnologias as serem utilizadas estivessem prontas, levando o time a apressar a finalização destes componentes sem ter a sua completa especificação. Isto acarretou na produção de muito conteúdo que era essencialmente inútil para o game que estava seria realmente produzido.

Ainda afetado por cronogramas errôneos, o time não deixou tempo suficiente para o tipo de experimentação, diálogo e prototipação que um game como Thief necessitava. No início de 1998, muitos dos erros no cronograma foram corridos e durante o restante do projeto, a herança deixada pelos erros anteriores fez com que as missões que não contavam com a tecnologia prevista ou que não estavam dentro do foco do projeto fossem refeitas e até mesmo excluídas.

3.1.4 Informações Adicionais

Data do lançamento: dezembro de 1998.

Tamanho da equipe: 19 programadores em tempo integral e alguns contratados.

Orçamento do projeto: aproximadamente 3 milhões de dólares.

Duração do projeto: 2,5 anos.

Plataformas: Windows 95/98.

Software utilizado: Microsoft Visual C++ 5.0, Watcom C++ 10.6, Opus Make, PowerAnimator, 3D Studio Max, Adobe Photoshop, AnimatorPro, Debabilizer e After Effects.

3.2 Half-Life

Half-Life (HL) é um game de ação em primeira pessoa desenvolvido pela Valve e distribuído pela Sierra Inc [5]. Embora esse tipo de game já fosse bastante comum, HL alcançou um sucesso tremendo, tendo vendido milhões de cópias e sendo considerado “game do ano” por mais de 50 publicações nos anos de 1998 e 1999[10]. Além disso, foram lançadas três continuações para o game[11]: *Half-Life: Opposing Force*, *Half-Life: Blue Shift* e o consagrado *Half-Life: Counter-Strike*, para ser jogado em rede

O lançamento do game HL estava originalmente agendado para novembro de 1997, mas essa data foi prorrogada em mais de um ano, pois em setembro desse mesmo ano, os autores tinham um parecer sobre o game: ele era ruim. Havia uma tecnologia interessante (o game é baseado no *engine* do Quake2 da ID) e algumas fases divertidas, mas num contexto geral, o game era considerado descepcionante.

Em uma situação como esta, estender um pouco o cronograma e fazer algumas alterações que melhorem o game é o caminho geralmente seguido, mas na Valve a escolha foi diferente: resolveu-se por retrabalhar-se cada estágio do game, partindo-se praticamente do início.

A primeira coisa feita foi colecionar tudo o que havia de bom e interessante no game até então e criar uma “super-fase”. A idéia dessa fase era ser um protótipo que agradasse a todos e que tivesse as características desejáveis para que o game obtivesse sucesso. A criação dessa

fase levou cerca de um mês e foi o suficiente para empolgar os desenvolvedores e mostrar o potencial do game. Bastava então multiplicar o que havia para criar-se mais uma centena de fases e o game estaria pronto.

Antes do início do processo de design do game, foram definidos rumos e estratégias para isso. Entre estas, estavam algumas teorias como a de que o conteúdo das fases devia ser baseado em distâncias e não tempo, que o "mundo" do game devia apresentar interação com o jogador e que o jogador nunca devia culpar o game por não conseguir alcançar seus objetivos (ou seja, perder). Com as diretrizes e a fase modelo criadas, bastava agora criar o restante do design do game, e para isso esperava-se ter um bom game designer que comandasse o processo.

3.2.1 A Questão do Game Designer

Após uma longa procura por um game designer que suprisse todas as expectativas da equipe do HL, chegou-se a conclusão que tal pessoa não existia. Foi quando surgiu a idéia de que ao invés de um *"super"* game designer, o trabalho poderia ser realizado por um grupo de pessoas de áreas distintas do desenvolvimento, grupo esse a que se deu o nome de *Cabal*. O grupo tornou-se responsável por criar um documento detalhado com a descrição de todo o design do game, incluindo fases, monstros, efeitos especiais, personagens e estória.

O *Cabal* inicialmente era consistido de três programadores, um designer de fases, um escritor e um animador. Os participantes eram freqüentemente rotacionados com outro membros da equipe de maneira que todos os envolvidos no projeto do HL pudessem ter a oportunidade de participar do *Cabal*. As reuniões do grupo eram realizadas quatro vezes por semana e tinham duração média de seis horas cada uma. A freqüência das reuniões e duração foi diminuída com o andamento do projeto, uma vez que havia menos trabalho de design a fazer.

O sucesso do *Cabal* superou as expectativas da equipe do HL. Onde se esperava que conflitos de ego e problemas com gostos heterogêneos fossem gerar resultados ruins, alcançou-se o contrário: o design foi se tornando limpo e empolgante, e o time trabalhando de maneira extremamente colaborativa.

O resultado final do trabalho do *Cabal* foi um documento com mais de 200 páginas detalhando desde onde cada interruptor de luz devia estar até a hora do dia em que cada fase se passava. O documento incluía ainda desenhos, animações e listas de tecnologias a serem desenvolvidas.

Um dos pontos ressaltados no design do HL foi a contratação de um escritor profissional (Marc Laidlaw) para dar consistência à estória, vida aos personagens e manter uma estrutura temática no game, que tinha conteúdo para um filme de mais de 30 horas de duração.

3.2.2 Pontos Fortes na Utilização do *Cabal*

Na avaliação do time de desenvolvimento da Valve, a utilização do *Cabal* foi um sucesso. Entre os principais pontos fortes de tal abordagem, podemos citar:

- Praticamente todos os níveis do game tiveram a participação de mais de uma pessoa da equipe. Com isso, tornou-se mais fácil resolver problemas e manter a consistência entre os níveis.
- Falta de hierarquia. Haviam poucas desavenças e conflitos de interesse dentro da equipe uma vez que não havia uma hierarquia definida dentro do *Cabal*.

- Todas as idéias podiam ser ouvidas, e idéias ruins eram descartadas pelo grupo (é comum um game designer achar que sua idéia é boa quando não é).
- Mesmo com um documento de 200 páginas, muitos detalhes ainda precisavam ser resgatados da memória dos participantes do designer. Com um grupo, essa tarefa era facilitada.
- Ao final do projeto, o trabalho podia ser dividido entre vários membros da equipe, uma vez que a experiência nas fases e tecnologias era compartilhada.

3.2.3 Dicas para a Criação de um *Cabal*

A Valve vem utilizando a abordagem do *Cabal* no desenvolvimento de novos games com sucesso. A recomendação principal para um time que pretenda fazer uso dessa abordagem é iniciar com um grupo relativamente pequeno e alterar o processo conforme as necessidades do projeto.

Algumas dicas importantes:

- Inclua um especialista de cada área do games (programação, arte, som, animação, roteiro/estória, etc). Argumentar a respeito de algo que ninguém entenda é uma total perda de tempo.
- Escreva tudo o que for decidido. É interessante ter seções de *BrainStorming* durante as reuniões, mas se os detalhes não forem escritos, eles serão esquecidos. O mais importante do documento gerado é que ele responda a perguntas futuras dos desenvolvedores do game.
- Nem todas as idéias são boas. As vezes é difícil convencer alguém do grupo que sua idéia é ruim. Em casos onde um conflito de idéias apareça, é interessante que esta seja guardada para o futuro. Se ela for boa, acabará se encaixando em algum ponto do game.
- Não adianta sonhar com tecnologias que podem não existir a tempo. É interessante que o game faça uso de tecnologias de ponta, mas deve-se ter bom senso e evitar um *design* irreal. Se a tecnologia aparecer, ela pode ser incorporada futuramente.
- Evite elementos técnicos que sejam utilizados apenas uma vez. Tudo o que exigir trabalho de programação deve ser utilizado em vários pontos do game. O trabalho de um programador é lento, e deve ser utilizado para tornar a vida do pessoal da equipe mais produtiva. Gastar tempo em tecnologias utilizadas em poucos pontos é uma perda de tempo e recursos.

3.2.4 Informações Adicionais

Data de lançamento: Novembro 1998

Estúdio Desenvolvedor: Valve Software

Publicador: Sierra Games

Tamanho da equipe: 20 desenvolvedores

Plataformas: Windows 95/98/2000/ME

3.3 Black and White

O game *Black and White* (BW), desenvolvido pela Lionhead Studios, é em sua essência um game que mistura conceitos de simulação de vida e estratégia, onde o jogador assumi a personalidade de um deus que pode ser bom, mau ou ambos, usando estas características para regrar e mudar todo o ambiente do game [14]. O jogador também deve escolher uma entre três criaturas pré-determinadas que irá ajudá-lo a completar os desafios do jogo, que incluem converter vilarejos e suprir os anseios de seus moradores.

O autor de BW, Peter Molyneux, sempre foi fascinado por games que permitiam que o jogador controlasse e influenciasse pessoas, mas ele queria um game aberto, flexível e atrativo diferente de tudo que já havia sido criado. Tecnicamente, Molyneux não queria um painel de ícones ou um conjunto de opções na tela que lembrassem ao jogador que ele estava apenas jogando um videogame. Molyneux também queria introduzir em BW a habilidade de selecionar uma criatura (originalmente qualquer criatura do mundo do game) e torná-la em um ser grande e inteligente que pudesse aprender, operar independentemente e cumprir as ordens do jogador a qualquer momento. Tal característica iria exigir uma estrutura de inteligência artificial diferente tudo que havia sido escrito.

3.3.1 O Período de Desenvolvimento

A primeira tarefa do desenvolvimento de BW foi a construção de um biblioteca de ferramentas que desse suporte ao game, o que foi feito por um time de seis pessoas.

O início oficial do desenvolvimento de BW foi em Fevereiro de 1998, já com um grupo de nove pessoas. Nesse momento, o time começava a pensar no game em termos gerais (o que ele poderia ter, o que deveria ter e como o time gostaria de vê-lo em uma situação perfeita), embora não tivessem certeza de que todas as características vislumbradas pudessem realmente ser feitas.

Durante o primeiro ano de trabalho sobre o BW, algumas pessoas foram adicionadas gradualmente ao time. A política de recrutamento utilizada foi apenas contratar pessoas de talento notável e cuja maneira de trabalho fosse adequada a dos membros já pertencentes ao projeto, uma vez que o time do BW já possuía suas características e jeito próprio de trabalho.

O time tinha convicção de que o jogo BW seria algo especial, fato que os inspirava a cada passo do desenvolvimento. Mesmo assim, muito trabalho foi necessário para terminar e lançar o jogo.

3.3.2 Casos de Sucesso no Desenvolvimento de BW

1) O projeto foi terminado

Por ser uma nova companhia, grande parte das ferramentas e bibliotecas utilizadas no game tiveram que ser construídas a partir do zero, obrigando o time a experimentar soluções e mudá-las caso não funcionassem, isto é, empregar o método conhecido como “tentava e erro”. Dessa forma o desenvolvimento de BW foi marcado por grandes problemas e seu lançamento posto em dúvida muitas vezes. Ainda assim, a união e perseverança da equipe fez com que o projeto fosse completado.

2) Todos os riscos foram compensados

Muitas das características incluídas no BW nunca tinham sido utilizadas anteriormente e era um risco apostar no seu desenvolvimento e sucesso. Alguns exemplos podem ser citados:

- Utilização de um sistema de gestos para ativar comandos no game no lugar de um painel de controle;
- Integração de uma linha de estória dentro de um game de estratégia de fluxo livre;
- Criação de uma criatura inteligente que fosse capaz de aprender coisas novas, levando a tecnologia ao seu extremo;
- As mudança do mundo do game, incluindo a atmosfera, construções, criatura e interface, dependendo se o jogador está atuando como um deus bom ou mau;
- Mecanismo de envio e recebimento de mensagens eletrônicas através da web e importação de nomes do catálogo de e-mails, permitindo colocá-los em aldeões das vilas do game.

3) A beleza do game

Quando algumas prévias do game foram apresentadas no encontro E3* na cidade de Atlanta em 1998, muitas pessoas que qualquer um poderia desenhar aquelas imagens em um pacote artístico e que o resultado final não seria tão bom. Mesmo assim, BW não apenas atingiu a aparência que os desenvolvedores pretendiam como a excedeu. Tal resultado exigiu muita personalização de software e também a aposta de que a capacidade mínima do PCs domésticos fosse compatível com as necessidades do game (por exemplo, na época muitas pessoas tinham 32 MB de RAM e o game necessitava de 64 MB).

4) A inteligência artificial

O resultado da inteligência artificial no game BW, com sua flexibilidade e poder, surpreendeu até mesmo o seu criador, Richard Evans. Cada vila tem seus próprios desejos, motivações. A criatura forma sua personalidade de maneira fascinante, de forma que dois jogadores nunca terão a mesma criatura. Tudo o que o jogador faz dentro do mundo do game pode ser ensinado a criatura. O grande desafio foi deixar a complexidade dessas características num ponto mínimo o suficiente para manter o game rápido.

5) A maneira como o time desenvolveu BW

Como BW foi o primeiro projeto do Lionhead Studios, as pessoas, o software, e o ambiente de trabalho eram todos novos. Embora isto fosse adequado para produção de um game diverso e inovador, também trouxe alguns problemas que foram bem enfrentados por Molyneux. Apesar do time ser composto por pessoas muito boas, o elemento de sorte foi que eles puderam trabalhar todos juntos muito bem entrosados.

3.3.3 Problemas no Desenvolvimento de BW

1) O planejamento da estória

O time de desenvolvimento subestimou quanto tempo seria preciso para construir e escrever a estória de BW. A natureza livre do game necessitava de uma narrativa para lhe dar estrutura e levá-lo a uma conclusão. O time começou a trabalhar na estória do game em Outubro de 1999, estimando não mais que dois meses para finalizá-la, mas algum tempo depois o time compreendeu que não teria as habilidades necessárias para cuidar de tal aspecto do game. Tal fato os levou a contratar James Leach, que já trabalhara como escritor de roteiros em outros games. Leach reuniu todas as idéias já feitas pelo time do BW em uma trama, escrevendo centenas de desafios e expedições, bem como todos os diálogos do game.

As vantagens de trabalhar com um escritor de roteiros foram inúmeras para a equipe de BW. Segundo Molyneux, Leach deu continuidade, consistência e estilo a todo o game, facilitando muitas tarefas.

2) A correção dos bugs

A correção de bugs no BW enfrentou os mesmos problemas da maioria dos projetos de software: existiam muitos bugs (aproximadamente 3 mil) para serem corrigidos em pouco tempo (seis semanas) e para cada bug arrumado, apareciam alguns outros. Foram criadas listas de bugs que circulavam por todos os membros da equipe. Após três semanas de trabalho o número de bugs se estabilizou e apenas restaram os dez últimos e mais difíceis para serem resolvidos. “Era como se jogo não quisesse ser finalizado e estar perfeito”, disse Molyneux.

3) O tamanho do projeto

O projeto de BW ficou tão extenso que os programadores sentiam-se perdidos dentro do código. A carga da menor ferramenta levava muitos minutos, e a compilação todo o game exigia mais de uma hora. Isto significava que no final do fase desenvolvimento, mesmo uma pequena mudança poderia levar um dia todo para ser implementada.

Outro problema como relação ao tamanho do projeto, e que não tinha sido antecipado, era que o game deveria caber em apenas um CD. O tamanho dos arquivos, como por exemplo áudio e música, levaram a equipe a comprimi-los e, em vista disso, ter a perda da alta qualidade.

Por fim, o game deveria ser lançado em 15 línguas diferentes, o que se tornou, segundo Molyneux, o maior trabalho de localização que ele jamais tinha participado.

4) A exclusão de características do game

Embora as características inicialmente pensadas para o BW não tenham mudado muito no curso da criação do game, algumas delas se mostraram inviáveis. Molyneux achava que os problemas seriam causados por restrições de software ou hardware, mas o que realmente limitou algumas características foram “questões emocionais”.

Por exemplo, a idéia original era que o jogador poderia escolher qualquer ser vivente do mundo do game para ser a sua criatura, desde formigas até seres humanos. Foi gasto muito tempo em concepção de arte neste sentido, mostrando os vários estágios que a criatura teria

durante o seu desenvolvimento. O time de BW logo percebeu que as pessoas teriam certas expectativas quanto a humanos como criaturas. Obviamente uma tartaruga não deveria aprender tão rapidamente quanto um humano. Também, disciplinar a criatura envolve dar-lhe palmadas, e certamente os desenvolvedores não queriam que o jogar fizesse isso em uma criança ou mesmo em um homem adulto. Grande parte do trabalho de concepção artística das criaturas foi abandonado, bem como a idéia de tornar qualquer coisa vivente em uma criatura que pudesse ser treinada.

Outro caso foi o uso de cores como conceitos dinâmicos e de "guerra de cores". A princípio, tais recursos seriam usados em grande escala, mas segundo o próprio Molyneux, a idéia foi descartada quando o time pensou em como o ambiente do jogo poderia se parecer com "um desenho feito por uma criança", fato os levou a restringir a utilização destes recursos.

5) As Datas de Lançamento

Um problema confesso de Molyneux e que ele não consegue deixar de falar a respeito de seus trabalhos atuais, o que o fez anunciar antecipadamente a data de lançamento de BW. Com todos os problemas enfrentados pela equipe, correção de bugs e testes, o resultado foi que o lançamento oficial de BW apresentou uma defasagem de alguns meses sobre data prevista. O próprio Molyneux fala a respeito da solução deste problema: "A melhor coisa a fazer é pegar a primeira data de lançamento e movê-la duas vazes à frente, e não anunciá-la antes de se chegar a sua metade".

3.3.4 Informações Adicionais

Data de lançamento: 30 de março de 2001.

Publicador: Eletronic Arts

Tamanho da equipe: 25 programadores em tempo integral e 3 contratados.

Tamanho do projeto: aproximadamente 2 milhões de linhas de código.

Orçamento do projeto: aproximadamente 5.7 milhões de dólares.

Duração do projeto: 3 anos, 1 mês e 10 dias.

Plataformas: Windows 95/98/2000/ME.

Software usado: Microsoft Dev Studio e 3D Studio Max.

Hardware utilizado: 800Mhz Pentium III com 256MB de memória RAM, discos rígidos de 30GB e placa de vídeo Nvidia GeForce.

3.4 Baldur's Gate II: Shadows of Amn

Baldur's Gate II: Shadows of Amn (BG2) é um game do tipo RPG, baseado na ambientação de fantasia medieval do mundo de Forgotten Realms e nas regras da segunda edição do RPG de mesa *Advanced Dungeons & Dragons* (AD&D) [15]. Inteiramente escrito nas linguagens

C++ e CLua (uma versão alterada da linguagem de scripting Lua, desenvolvida no Brasil, na PUC-RJ), ele foi lançado no ano de 2001 e obteve notável êxito comercial, assim como o seu antecessor, lançado 2 anos antes. Tanto o original quanto a seqüência foram desenvolvidos pela empresa canadense BioWare, e foram publicados pela Black Isle Studios, a divisão de RPGs da Interplay Productions.

O desenvolvimento do game original, Baldur's Gate (BG), exigiu esforços de 90 homens-ano, levados adiante por uma equipe de indivíduos inexperientes, mas talentosos. A BioWare possuía até então apenas um único título a se creditar, um game chamado Shattered Steel. Baldur's Gate se tornou um grande sucesso comercial, tendo ganhado diversos prêmios e vendido em torno de 1,5 milhões de cópias em todo o mundo. A BioWare então iniciou o desenvolvimento do BG2, tendo como objetivo não apenas repetir o sucesso do original, mas também melhorar aquele que já era um grande game.

Construir uma seqüência de qualidade não é um trabalho fácil. O engine do game original, o Infinity Engine, já havia sido licenciado e utilizado em outros dois games: Icewind Dale e Planescape: Torment, ambos desenvolvidos pela Black Isle Studios, o publicador do BG. O novo game seria comparado aos padrões de qualidade estabelecidos por estes títulos, e teria que superá-los. Portanto, foi necessário que se iniciasse o trabalho com a filosofia correta: o objetivo devia ser fazer algo melhor, e não o mesmo game novamente. Além disso, devia-se adotar um mecanismo que permitisse a quantificação dos erros anteriores e posterior aprendizado sobre estes: se os erros não fossem identificados, dificilmente seria possível consertá-los.

Na BioWare, criou-se uma cultura de se fazer análises pós-projeto detalhadas para se buscar as áreas fortes e fracas no desenvolvimento de seus produtos. No caso do BG original, a empresa concluiu que não houve tempo suficiente para que os objetivos do design fossem alcançados, devido ao desenvolvimento simultâneo do engine e do conteúdo do game. Desta forma, houveram esforços para que houvesse uma simplificação de algumas características. Quando desenvolvendo o BG2, portanto, um tempo maior foi dado aos desenvolvedores, para que estes pudessem fazer uso de todo o potencial apresentado pelo engine já existente.

3.4.1 A Lista de Características

No início do projeto, a equipe criou uma lista de características que o jogo deveria ter. A licença do AD&D trazia junto consigo uma infinidade de opções e características que poderiam ser acrescentadas ao game. O desafio, então, era definir quais destas características seriam utilizadas. Foram tomadas duas rotas: a primeira foi criar uma lista interna, gerada pela BioWare e pela Black Isle, do que seria racional e lógico levando-se em conta o engine. A segunda foi perguntar aos fãs do game original, com o uso da Internet, o que eles gostariam de ver. Isto deu à equipe uma boa idéia sobre o que os fãs mais assíduos esperavam, e qual a direção a se tomar. A lista de características resultante possuía diversas opções, de diversas naturezas: do aumento da resolução gráfica ao uso de duas armas simultaneamente pelos personagens; de alterações na interface ao desenvolvimento das relações entre os personagens. Muitas características foram acrescentadas ao game mais tarde, mas poucas não foram implementadas ou funcionaram pior do que o esperado.

Algo que não foi feito, e que mais tarde verificou-se que teria sido uma boa idéia, foi a atribuição de prioridades às características da lista. A equipe não possuía qualquer mecanismo para decidir quais delas seriam mais ou menos importantes, tentando acomodar tantas quanto

possível. Por exemplo, o modo *deathmatch*[2] deveria ter sido eliminado no início do projeto, mas foi desenvolvido até os estágios finais da produção. Ele só foi eliminado quando a equipe percebeu que o tempo não era suficiente, e que o modo não estava sendo bem recebido nos testes realizados. Um mecanismo que apontasse quais eram as prioridades poderia ter indicado que esta não era uma característica crítica, economizando tempo dos programadores, que poderia ter sido usado em outras áreas.

3.4.2 Diretivas de Design

Algo que os desenvolvedores queriam definitivamente evitar era repetir quaisquer falhas de design que existiram no game original. Já que muitos dos membros da equipe haviam sido contratados após o desenvolvimento do BG, e levando em conta que não se poderia confiar na memória dos demais, decidiu-se criar um conjunto de diretivas que cada departamento deveria seguir. Alguns exemplos de diretivas são:

Regras básicas de design

- O jogador deve sempre sentir que são as suas ações que trazem o seu sucesso.
- O jogador deve ter a sensação de afetar o seu ambiente, trazendo conseqüências visíveis no mundo do game.
- O design deve considerar um caminho “bom” e um caminho “mau”, com sub-tramas mudando e surgindo de acordo com o comportamento do jogador.

Design da estória/roteiro

- O foco deve ser sempre o jogador: todos os eventos devem ocorrer à sua volta.
- A estória deve ser dinâmica, apresentando viradas e surpresas envolvendo a estória do personagem do jogador.
- O final da estória deve ser mantido em aberto, possibilitando continuações.

  Design do Ambiente:

- Certas áreas centrais, às quais o jogador retorna constantemente, devem mudar conforme o tempo passe. Ações do jogador em outras áreas devem afetar estas áreas centrais.
- As tramas devem evitar o deslocamento do jogador entre muitas áreas, pois isto pode se tornar entediante.

Design do sistema de jogo

- O jogador deve ser recompensado constantemente através do ganho de poderes, ítens, desenvolvimento da trama, monstros, arte, romances, etc.

[2]Modo de jogo criado pela Id Software onde o objetivo da partida é eliminar o maior número de oponentes no menor tempo possível.

- O jogador deve ser capaz de criar e configurar de maneira única o seu personagem.
- O mundo do game deve refletir o modo como o jogador criou o seu personagem.

  Diretivas dos Textos:
- O texto não deve ter palavrões ou expressões modernas.
- Cada pedaço de diálogo não deve exceder duas linhas, exceto em circunstâncias muito especiais.
- Sotaques devem ser evitados.
- Quando há uma lista de opções de respostas para o jogador, seu tamanho deve ser sempre três. Em situações específicas, este número pode ser de dois ou quatro.

Alguns comentários sobre estas diretivas devem ser feitos. Esta lista não é exaustiva, trazendo apenas uma parte da lista completa. As diretivas foram escritas durante todo o desenvolvimento do jogo, e não refletem a lista que existia no princípio do desenvolvimento. Deve-se também lembrar que elas são diretivas, e não leis que não podem ser quebradas.

Ao final do desenvolvimento, concluiu-se que estabelecer diretivas logo no início do projeto é algo bastante útil, mesmo que as mesmas precisem de revisões e atualizações.

3.4.3 Pipeline

Um dos elementos mais importantes do processo de desenvolvimento do game foi o *pipeline* de arte e conteúdo. O pipeline, que já havia sido usado no desenvolvimento do BG original, foi dividido em quatro partes: programação, filmes, animações do game e níveis (ou áreas). O maior e mais complexo destes era o pipeline usado para criar os níveis, cuja seqüência é mostrada abaixo como exemplo:

1. Os designers mapeiam uma área e a descrevem;
2. Um artista cria um desenho isométrico conceitual do nível;
3. Modelos são criados para o nível;
4. Os modelos são posicionados e texturizados;
5. A área é preenchida com objetos menores, como barris e cadeiras. São aplicados efeitos de iluminação, e os últimos detalhes são desenhados;
6. O trabalho artístico é enviado aos desenvolvedores para que sejam acrescentados os detalhes técnicos, como altitude, *clipping* e mapa de busca;
7. São inseridos os ítens, criaturas, gatilhos e armadilhas da área;
8. Os scripts do nível são completados.

Ao final do projeto, diversas fraquezas do processo foram detectadas. A documentação das alterações feitas durante o desenvolvimento foi ineficiente, por vezes ficando incompleta ou desatualizada. Algumas partes do time trabalharam independentemente, e a falta de comunicação entre os programadores e os artistas muitas vezes acabaram por fazer com que certos elementos em algumas áreas do game ficassem faltando, e algumas convenções e nomes fossem diferentes entre as equipes. Levando em conta que o game possui centenas de áreas e milhares de peças individuais de arte, este pode ser um problema muito grande. A empresa pretende melhorar a integração da equipe em projetos futuros.

3.4.4 Gerenciando o Período Intermediário do Projeto

Durante o desenvolvimento de qualquer game, chega um momento onde as pessoas já trabalharam tanto sobre o mesmo tema que começam a se cansar dele. Este período intermediário, no caso do BG2, foi complicado pela existência de projetos novos e atraentes dentro da empresa sendo desenvolvidos ao mesmo tempo.

Para resolver este problema, alguns funcionários foram deslocados entre os projetos. A empresa buscou respeitar suas vontades, levando em conta seu perfil de cada um, e a existência de algumas pessoas que gostam mais de iniciar os projetos, enquanto outras preferem concluí-los.

3.4.5 Testes

Por causa do seu tamanho imenso, os testes do BG2 foram imensamente complicados, em especial porque as áreas individuais não possuíam uma fase específica de testes durante o desenvolvimento. O game possuía em torno de 290 sub-tramas distintas, com tamanhos variados (uma sub-trama pode durar de 20 minutos a 2 horas). Cada uma destas sub-trama precisava ser testada nos modos para um único jogador e para jogadores múltiplos.

O processo usado para os testes consistia na colocação de diversos quadros brancos nos salões onde os testes eram realizados, listando cada sub-trama. Ao lado de cada sub-trama, um "X" era colocado. Os desenvolvedores e os membros da equipe de garantia de qualidade, ou *Quality Assurance* (QA), foram divididos em duplas. Cada dupla assumia o teste de uma sub-trama específica, testando-a exaustivamente, e removendo o "X" referente àquela sub-trama quando se certificava que esta estava estável. Em duas semanas, o quadro foi limpo (numa primeira passada). Ao mesmo tempo, uma outra equipe de QA, com o auxílio de mais alguns desenvolvedores e programadores, trabalhou testando o game no seu modo para múltiplos jogadores.

Ao final do processo, foram encontrados e corrigidos um total de 15000 bugs. Certos ajustes e alterações não puderam ser feitos, devido aos esforços que as mudanças exigiriam num estágio tão avançado da produção. No final, a empresa concluiu que mais testes deveriam ser feitos durante o desenvolvimento.

3.4.6 Sumário

O que funcionou:

- Tecnologia do engine estável;
- Time dedicado ao projeto;
- Veteranos retornando para incrementar um sistema que eles mesmos haviam criado, já estando portanto familiarizados com o pipeline e com o engine;
- Boa disciplina de projeto;
- O processo de controle de qualidade adotado no final do desenvolvimento se mostrou bastante eficiente.

O que poderia ser melhorado:

- Fragmentação da comunicação do time;
- O game era grande demais;
- Falta de um estágio prematuro de QA.

4 Conclusão

Uma vez que o desenvolvimento de um game é um processo criativo e em vários pontos artístico, a definição de um modelo que funcione na maioria dos casos é uma tarefa difícil, se não impossível. Contudo, através de uma contextualização do assunto e um estudo de modelos que têm funcionado é possível gerar algumas diretrizes que auxiliam o processo.

A escolha de um bom game designer pode ser crucial para o sucesso de um game. Por sua vez, profissionais com o perfil ideal de um game designer são muito raros, e processos como o *Cabal* criado pela Valve no desenvolvimento do game *Half-Life* são uma alternativa interessante.

Embora seja difícil fazer um paralelo com os processos conhecidos da *Engenharia de Software* tradicional, várias fases e técnicas no desenvolvimento podem ser identificadas, como por exemplo modelos de documentação, implementação, testes, orçamento, cronograma e manutenção. Entretanto, como um processo acima de tudo criativo, o desenvolvimento de games tem espaço para a inovação, improvisação e adaptação, se caracterizando como um modelo totalmente flexível.

Referências

[1] Ernest Adams. The Designer's Notebook: Bad Game Designer, No Twinkie. http://www.gamasutra.com, March 1998.

[2] Ernest Adams. The Designer's Notebook: Bad Game Designer, No Twinkie II. http://www.gamasutra.com, March 2000.

[3] Ernest Adams. The Designer's Notebook: Bad Game Designer, No Twinkie III. http://www.gamasutra.com, February 2002.

[4] Luke Ahearn. Budgeting and Scheduling Your Game. http://www.gamasutra.com, May 2001.

[5] Ken Birdwell. The Cabal: Valve's Design Proccess For Creating Half-Life. http://www.gamasutra.com, December 1999.

[6] Arnold Hendrick. Hiring Game Designers. *Game Developer Magazine*, February 1998.

[7] Tim Huntsman. A Primer for the Design Process, Part 1: What to Do. http://www.gamasutra.com, June 2000.

[8] Tim Huntsman. A Primer for the Design Process, Part 2: What to Do. http://www.gamasutra.com, July 2000.

[9] Tim Huntsman. A Primer for the Design Process, Part 3: What to Do. http://www.gamasutra.com, July 2000.

[10] Sierra Entertainment Inc. Official Half Life Web Site – Awards section. http://www.sierra.com/games/half-life/awards.html.

[11] Sierra Entertainment Inc. Official Half Life Web Site – Order section. http://www.sierra.com/games/half-life.

[12] Rob Irving. Lost Along The Way: Design Pitfalls on the Road from Concept to Completion. http://www.gamasutra.com, April 2002.

[13] Tom Leonard. Postmortem: Looking Glass's Studios Thief: The Dark Project. http://www.gamasutra.com.

[14] Peter Molyneux. Postmortem: Lionhead Studio's Black & White. http://www.gamasutra.com, June 2001.

[15] Gavin Moore. Baldur's Gate II: The Anatomy of a Sequel. http://www.gamasutra.com, May 2001.

[16] Roger Pedersen. Pedersen Principles of Game Design and Production. http://www.gamasutra.com, March 1999.

[17] Tim Ryan. The Anatomy of a Design Document, Part 1. http://www.gamasutra.com, October 1999.

[18] Tim Ryan. The Anatomy of a Design Document, Part 2: Documentation Guidelines for the Functional and Technical Specifications. http://www.gamasutra.com, December 1999.

[19] Marc Salttzman. Game Design: Secrets of the Sages. http://www.gamasutra.com, March 2002.