O Paradigma do Projeto Baseado em Plataformas Aplicado ao Game Design

Dino L. Figueirôa Fábio Campos André M. Neves

Universidade Federal de Pernambuco, Dept. de Design, Brasil

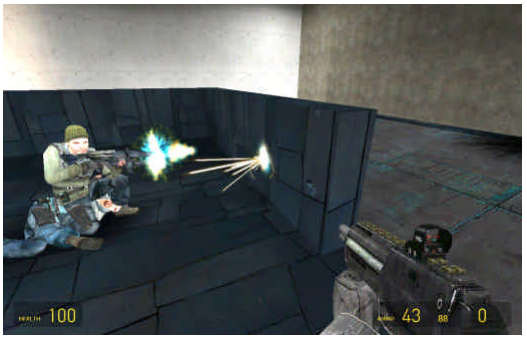

Figura 1: Desenvolvimento do jogo SBGames MOD a partir da plataforma Half-Life 2

Resumo

Os modernos jogos para PC vêm sofrendo uma mudança de paradigma com o advento do projeto baseado em plataforma (*platform-based design* – PBD), emergida nos ambientes industriais. Através dessa mudança o jogador também é convidado a ser um game designer, gerando jogos personalizados. Este documento expõe como esta técnica vem sendo absorvida pelos *game designers*, resolvendo problemas de design de software. Esse artigo apresenta conceitos como *MOD*s, *add-on*s e pacotes de expansão e os posiciona como resultados diretos do emprego da técnica em questão. Por fim, é apresentado um modelo de plataforma com o qual é construído um jogo ilustrativo dentro desta proposta, o "SBGames MOD".

Palavras-chave: plataformas, design baseado em plataformas, game design, jogos para PC, reuso de código, MODs

Abstract

The modern PC games are changing with the platform-based design - PBD - advent, born at industrial environment. Thru this change the player is invited to be a game designer, creating his own customized games. This document expose how the technique is being inserted into the game designing community, resolving software design problems. This article also presents concepts as MODs, add-ons and expansion packs as direct results of the use of the mentioned technique. Finally, we introduce a platform model, constructing an illustrative game under this proposal, the "SBGames MOD".

Keywords: platforms, platform-based design, game design, PC games, code reuse, MODs

Contato dos autores:
{dinolincoln,fc2005,andremneves}@gmail.com

1. Introdução

Principalmente a partir dos anos 80, outorgando grandes contribuições científicas providas pela HfG de Ulm na década antecedente, o design consolida-se como ciência. Novas metodologias de projeto, catalizadas pela globalização e pelo crescimento industrial delineado desde o pós-guerra, são categorizadas como "mudanças de paradigma de projeto" [Bürdek 2006]. As linhas industriais deparam-se com novas demandas metodológicas uma vez que os projetos tornaram-se por demais complexos, em grande volume e difíceis de serem reusados [Alexander 1964]. Recaindo sobre os designers a responsabilidade de aperfeiçoar o processo industrial.

Nestas circunstâncias, nasce a "plataforma", um design reutilizável mediante customização de alguns componentes intencionalmente dispostos para este fim [Goering 2002].

A técnica do design baseado em plataforma, também conhecida por *"platform-based design"*, emergiu nos ambientes industriais como alternativa modularizar e reaproveitar projetos eficientemente [De Marinis et al. 2003]. Através da mesma uma única plataforma de chassis poderia servir para uma inteira linha de carros, com a modificação de apenas algumas seções configuráveis, intencionalmente dispostas para este fim.

A engenharia eletrônica absorveu fortemente esta técnica, principalmente no design de componentes de hardware que, atuando como plataformas, atendiam a um amplo número de projetos [Bouldin 2003]; dessa maneira evitando que fosse desenvolvido um chip para cada atuação diferente [Sangiovanni-Vincentelli 2002].

Outra área onde as plataformas são aceitas com grande sucesso é a engenharia de software [Graaf et al. 2003] (sobre esse campo nos aprofundaremos no decorrer desse trabalho), seguindo o mesmo conceito predominante nas outras áreas, onde uma única plataforma de software pode ser produzida para servir a uma grande variedade de aplicações, proporcionando aos desenvolvedores reutilização de código e redução do *time-to-market* (tempo até o mercado).

Chegamos então ao projeto de jogos digitais. Nesse ínterim, se observarmos os jogos para PC, veremos que se tratam de sofisticados programas de computador que herdam as mesmas demandas e problemas de projeto encontrados na engenharia de software [Claypool e Claypool]. Esse segmento está rapidamente incorporando a técnica da plataforma, produzindo jogos configuráveis (muitas vezes pelo próprio usuário), e gerando a partir de um mesmo jogo (ou seja de uma "plataforma") uma série de continuações do jogo original e/ou outros jogos completamente diferentes [El-Nasr e Smith].

Nesse trabalho ilustraremos a modelagem dos componentes de uma plataforma de games e exemplos de como manuseá-los para gerar novos jogos.

2. Design Baseado em Plataforma

Considerando que existem muitas conceituações para o "design baseado em plataforma", Goering [2002], adotaremos o conceito de Sangiovanni-Vicentelli [2002], o qual conceitua o design baseado em plataformas como um projeto baseado em uma camada de abstração com duas visões. Visões essas que representam a seção com componentes fixos e a seção com componentes configuráveis. Nesse paradigma, quem desenvolve/configura os componentes da seção configurável se abstrai da complexidade da seção fixa; sendo essa abstração uma das maiores vantagens do uso dessa técnica.

O Grupo de Desenvolvimento em Projeto Baseado em Plataforma (Platform-based Design Development Working Group – PBD DWG) sugere que esse paradigma seja utilizado na forma de "uma abordagem de design integrado enfatizando reuso sistemático para desenvolvimento de produtos complexos sob Plataformas... ...com a intenção de reduzir custos de desenvolvimento, riscos do projeto e tempo até o mercado" [Goering 2002].

O paradigma de projeto baseado em plataforma ganha consistência por apresentar características constantes ao serem utilizadas para gerar diversos objetos, muitas vezes de áreas distintas (vide Figura 2). Algumas dessas características são:

- Componentes fixos, onde se encontram a maior parte dos recursos de disponibilizados e onde se concentra o maior esforço de desenvolvimento inicial, também definido como "sessão de arquitetura" [Chen et al. 2003]
- Componentes variáveis, através das modificações dos quais é possível gerar produtos distintos (mas pertencentes a uma mesma plataforma), a chamada "sessão de aplicações" [Chen et al. 2003]
- Capacidade de abstração, pois os desenvolvedores dos componentes variáveis não precisam se aprofundar na complexidade de desenvolvimento dos componentes fixos, mas sim em como interagir com os mesmos [Sangiovanni-Vincentelli 2002]

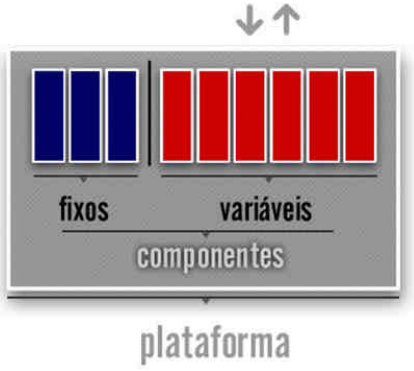

Figura 2: Estrutura de uma plataforma

2.1 Plataformas na Indústria Automobilística

A indústria automobilística FIAT vem utilizando a técnica das plataformas em diversas divisões dos projetos de seus carros. Tanto motores e chassis quanto em sistemas eletrônicos. O projeto V.E.N.I.C.E. é uma resposta baseada em plataforma diante uma problemática internacional. Um mesmo modelo de automóvel tem configuração diferente do sistema de luzes em cada país comercializado, de acordo com a legislação local. Entretanto para evitar contratempos tais como produzir diferentes sistemas, componentes e instruções de instalação para as fábricas, além de dificuldades em vender carros fabricados para um país em outro, a FIAT produziu uma única plataforma do sistema eletrônico que administra as luzes dos veículos para diversas configurações. Como toda plataforma, esse sistema possui a sessão de componentes fixos, tais como cabos, lâmpadas, circuitos eletrônicos, códigos de software e microprocessador. E possui a sessão de componentes variáveis, que, nesse caso particular, trata-se apenas de um elemento: a informação proveniente da leitura de um código ou chip de memória. Dependendo da informação inserida nesta área, o sistema se informa acerca de qual configuração de luzes deve ser assumida, modificando suas características para atender a legislação de um determinado país (vide Figura 3).

Caso pretenda-se registrar o veículo em outra jurisdição, basta encaminhar-se a uma manutenção autorizada FIAT para, novamente, ajustar qual configuração o sistema deve assumir.

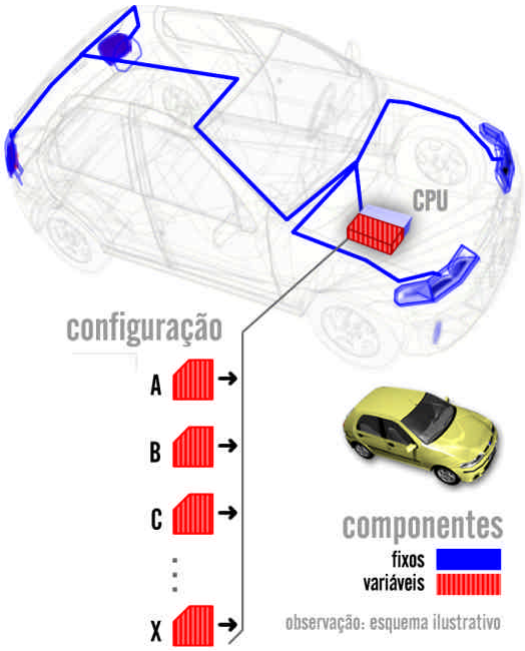

Figura 3: Plataforma V.E.N.I.C.E.

2.2 Plataformas na Engenharia de Software

Na engenharia de software, onde a abstração exerce uma forte demanda não só no produto final, mas também no seu desenvolvimento, as plataformas são usadas em diversas vertentes. Podemos destacar como exemplo a programação orientada a objetos [Douglass 2003] que emprega o paradigma em questão utilizando os argumentos, ou variáveis de entrada, como componentes variáveis; e toda a estrutura restante do objeto, como componentes fixos. Visualizando os "métodos" ou "classes" como pequenas plataformas dentro do programa [Graaf et al. 2003].

É importante destacar que podemos ter objetos dentro de objetos, como métodos que possuem outros métodos dentro de sua estrutura. Além da recursão, onde uma função computacional pode ter ela mesma como componente interno. Desdobrando-se em diversos encapsulamentos de objetos e demonstrando assim a capacidade das plataformas de ter outras plataformas dentro de sua estrutura. Atuando como diferentes tipos de componentes. Em outras palavras, uma plataforma pode ser um componente, fixo ou variável, de outra plataforma.

2.3 Benefícios do Projeto Baseado em Plataforma

A já citada abstração e o reuso são benefícios primários que acarretam vantagens indiretas no uso do projeto baseado em plataforma [Chen et al. 2003]. De acordo com Augusto de Oliveira, arquiteto chefe da Philips, alguns projetos de chips caminham para 50% de economia em tempo no desenvolvimento, utilizando esse paradigma. Área que por sofrer bastante pressão para reduzir o tempo até o mercado tem se tornado uma dos principais catalisadoras do desenvolvimento do projeto baseado em plataforma [Nsame et al. 2001].

3. Projeto Baseado em Plataforma aplicado ao Game Design

A técnica do projeto baseado em plataforma vem ganhando espaço no game design por atender a fortes demandas do setor, como reutilização eficiente de código, modularização, redução dos riscos e rápida reconfiguração [Rucker 2002].

Não é necessário fazer uma analogia à engenharia de software para indicar de onde o game design (para PCs, por exemplo) herdou estes problemas. Uma vez que os jogos modernos para PC são softwares entre os mais complexos**,** envolvendo recursos avançados (da modelagem 3D à inteligência artificial), técnicas que otimizem sua produção abstraindo diversas etapas de desenvolvimento são rapidamente absorvidas. Bethke [2003] chega a afirmar que: "Desenvolvimento de jogos é desenvolvimento de software".

Aos problemas próprios de software o universo do game design acrescenta uma forte tendência de jogos expansíveis [Claypool e Claypool], sendo esta expansão feita muitas vezes pelo próprio usuário que utiliza a web como ferramenta de divulgação e compartilhamento.

O conceito de plataforma adotado neste documento comunica um de método de design. Portanto, não devemos confundir com o termo "Jogos de Plataforma", "*Platform Games*". Este refere-se a um estilo de jogo onde um dos objetivos é saltar, evitando obstáculos, de "plataforma" em "plataforma" [Lecky-Thompson 2002].

4. Modelo Proposto

Assim como no projeto V.E.N.I.C.E., os games atuais estão assumindo uma estrutura de plataforma, com uma sessão de componentes fixos e outra de componentes variáveis. Através da modificação dessas variáveis pode-se produzir uma ampla variedade de jogos personalizados, às vezes sem saber programar sequer uma linha de código, uma vez que na configuração destes componentes variáveis abstraímo-nos da complexidade dos componentes fixos e suas aplicações.

Sugerimos uma estrutura de plataforma para jogos para PC como a ilustrada na Figura 4; onde cada componente será descrito a seguir.

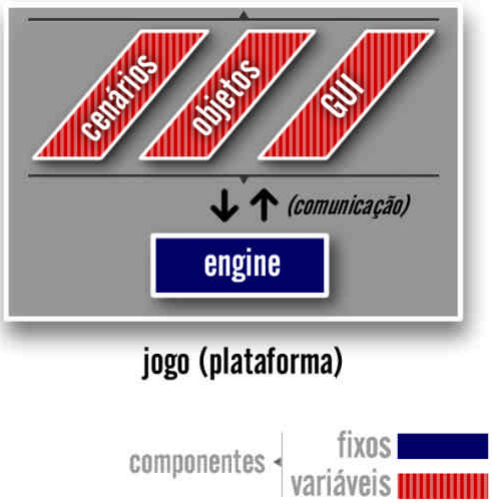

Figura 4: Plataforma de jogos

4.1 Engine

É a central nervosa do programa do jogo, sendo responsável pela geração de gráficos 2D e 3D (*rendering*), aplicações de rede, animações, detecção de colisões, inteligência artificial, sons, geração de partículas, dentre outros [Rouse III 2001]. Chegando muitas vezes a ser modularizado, por tarefa, tais como "*render engine*" para gráficos, "*physics engine*" para cálculos de animação de objetos [Eberly 2001; 2003] e assim por diante. Em outras palavras, o engine principal pode ser entendido como o encapsulamento de vários engines.

Um engine completo pode ser desenvolvido na forma de plataforma, para atender linhas inteiras de jogos. A Valve Corporation, fabricante da série de jogos de tiro em primeira pessoa mais jogados do mundo, o *Half-Life*, desenvolveu recentemente o "*Source Engine*", para o *Half-Life 2*. Utilizando Visual Studio 7.1 e linguagem de programação C/C++. O "*Source*", como é conhecido, trata de recursos extremamente avançados como o inovador "*digital muscles*", tecnologia de reprodução de expressão facial humana e não-humana em personagens 3D. Capaz de rodar em diferentes ambientes (Microsoft Windows, Xbox 360 e Playstation 3) este mesmo engine é utilizado para os jogos *Counter Strike: Source, Team Fortress: Source, Day of Defeat: Source*, e centenas de outros. Justificando assim o seu custo de desenvolvimento, uma vez que terá as vendas multiplicadas pelos diversos segmentos de jogos que utilizam seus recursos. Este custo de desenvolvimento varia bastante, dependendo obviamente dos requisitos que o projeto visa atender. Se o jogo vai possuir multiplayer tornando necessário desenvolvimento de aplicações de rede, ou se disponibilizará campanha, propiciando uma necessidade de level design. Contudo, para se ter uma idéia da dimensão deste setor, o desenvolvimento do engine do bem sucedido "*Warcraft III*", da Blizzard Entertainment, chegou à marca de 3,7 milhões de dólares.

4.2 Objetos

São modelos, geralmente 3D, que interagem com o usuário e o cenário [Fullerton et al. 2004]. Atuados automaticamente pelo engine, algumas vezes articulado com IA (inteligência artificial), outras vezes com movimentos programados, ou ainda pela intervenção do usuário. Compõem os personagens, armas, veículos e etc.

Enquanto que em alguns jogos pode-se trocar a textura dos modelos 3D apenas, em outros é possível inserir objetos completamente novos, assim como suas características de comportamento, modelados em programas de edição tridimensionais externamente (vide Figura 5). Tudo dependendo da flexibilidade da plataforma em questão.

Figura 5: Objetos em desenvolvimento para o jogo Ghost Recon Advanced Warfighter

4.3 Cenários

Ambientes virtuais onde atuam os objetos em jogo. Pode ser encarado como um "objeto grande" que interage com os demais objetos, porém com algumas características bastante particulares, como tecnologia para geração de céu virtual ("*skybox*"), ou efeitos de reflexo na água [Bethke 2003]. O design de cenários vem ganhando aperfeiçoamentos nos últimos anos com o progresso das tecnologias de gráficos vetoriais 3D, que possibilitam a "renderização" de cenários com mais objetos e detalhes.

A maioria dos SDKs para jogos possuem apenas construtores de cenários, os quais são, na verdade, um administrador de um conjunto de objetos posicionados sobre um terreno atuando com um seletor de texturas (paredes, terrenos, águas, etc) para superfície de objetos e terrenos.

4.4 GUI

A interface gráfica ("*Graphical User Interface*") é a principal responsável por imergir o usuário na temática que os designers projetaram para o jogo. Segue a idéia do desenho de conceito, transmitindo informações que cognitivamente estimulem o jogador a sentir-se

envolvido pelo ambiente proposto. Conceitualmente, podemos subdividir o sistema em duas interfaces.

Na primeira situam-se as configurações preliminares ao jogo, tais como: ajustes gráficos, opções de dificuldade, seleção de estágios e etc. Exemplificando, um jogo de guerra pode ter como tela de fundo imagens de armas, enquanto um simulador de cidades reproduz um centro comercial. Em alguns casos o usuário interage com uma interface tridimensional, como em *"Star Wars Battlefront"*, onde é passada a impressão de alta tecnologia (veja Figura 6).

Figura 6: GUI do Star Wars Battlefront

A segunda é a interface de jogo, onde adota-se o termo técnico *"HUD"* (*Head Up Display*). Trata-se da mira, quantidade de munição, radares, tempo restante do estágio, etc. Estes elementos dinâmicos são conhecidos como *"widgets"* ou *"gauges"*. Por serem efetivamente interligados ao engine, tornam esta interface mais trabalhosa para manuseio. Podendo vir a exigir alguma intervenção em linguagem de programação.

4.5 SDKs

Muitos jogos acompanham um SDK (*"Software Development Kit"* - Conjunto de Desenvolvimento de Software) para desenvolver recursos adicionais [Nieborg 2005], como o *"World Builder"* do *"Command & Conquer: Generals"*, EA Games, onde o usuário pode criar novos campos de batalha e adicionar à biblioteca de mapas do jogo (vide Figura 7).

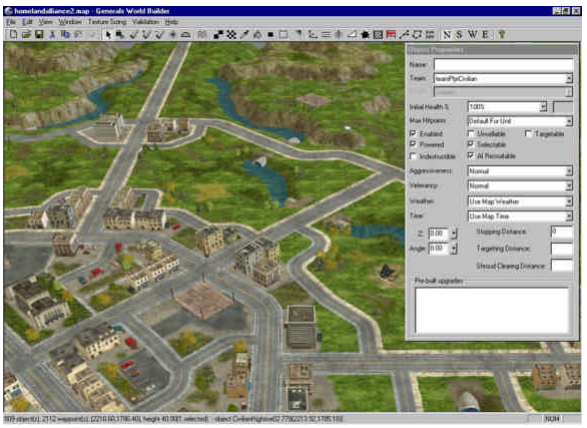
Figura 7: World Builder do Command & Conquer Generals

Algumas empresas disponibilizam programas para desenvolvimento de games acompanhados de um engine para que o desenvolvedor usufrua de seus recursos se abstraindo da complexidade das aplicações que rodam no mesmo. O *"Torque Game Engine"* da "Garage Games" é um exemplo bastante difundido. Utilizado para produzir, dentre outros, o jogo *"Wildlife Tycoon: Venture África"*. Ele constitue-se num engine repleto de recursos acompanhado de uma API (*Aplication Programing Interface* – Interface de Programação de Aplicações) para administrar os componentes variáveis da plataforma.

5. Utilização Prática das Plataformas nos Games

As alterações dos componentes variáveis dos games em plataformas são tão comuns hoje em dia que já ganharam até nomes. "MOD"s, "add-ons", e pacotes de expansão são, na prática, quase tão conhecidos quanto os próprios jogos para os quais eles são produzidos. Sites, grupos e até empresas utilizam a estrutura de plataforma para criá-los, possibilitando a implementação novos mapas, armas, cenários, interface e até narrativas, criando um jogo totalmente novo a partir da plataforma principal.

A seguir descreveremos os conceitos dos principais tipos de artefatos gerados em sobre plataformas de games:

5.1 "MOD"s

Como o próprio nome sugere, "MOD" é o termo abreviado para "modificação". Os componentes variáveis, de acordo com o modelo de plataforma, são alterados para transmitir a idéia de que se está em outro jogo. Atividade também conhecida por *"modding"* [Salen e Zimmerman].

Podem ser produzidos tanto por empresas, com fins comerciais, quanto pelos usuários sem fins lucrativos. Mesmo sendo *"freewares"* em sua maioria, certas vezes causam um verdadeiro *frenesi* no mercado. Pois para utilizar um "MOD", é necessário ter a plataforma

principal comprada, para então modificá-la [Bethke 2003]. Por exemplo, muitos fãs da trilogia cinematográfica *"Star Wars"* utilizam "MOD"s "*freeware*" para transformar diferentes jogos no ambiente *"Star Wars"* (veja Figura 8).

Figura 8: O MOD futurista Galatic Conquest (Star Wars) para Battlefield 1942, um jogo que reproduz a II Guerra Mundial

O "MOD" mais famoso até então é o *"Counter-Strike"* (CS), que reproduz batalhas entre policiais e terroristas. Muitos dos milhões de jogadores deste jogo ao redor do mundo, que atinge a marca de mais de 94 mil servidores online, nunca ouviram falar do *"Half-Life :Deathmatch"*, que acumula cerca de 800 servidores [Valve 2007]. Enquanto que na verdade, o CS é um "MOD" do *"Half-Life"*. O "MOD" fez mais sucesso do que a própria plataforma em que foi criado, tornando-se o *"First Person Shooter"* multiplayer mais jogado do mundo.

5.2 "Add-on"s

Consistem em pequenos "MOD"s, com objetos ou cenários adicionais , mas sem alterar o drasticamente a temática do jogo nem a GUI [Rouse III 2001] (vide Figura 9). É uma categoria amplamente empregada em simuladores. O *"Microsoft Flight Simulator X"*, também conhecido por "FS", na versão mais completa (Deluxe Edition) acompanha 24 aeronaves disponíveis para pilotagem. Cada um destes objetos são compostos de modelo 3D externo, painel 2D, painel 3D (cockpit), texturas, sons e realismo de vôo (arquivos de extensão **.cfg*). Entretanto um dos fatores associados ao grande sucesso do "FS" é a possibilidade de adicionar, literalmente, milhares de novas aeronaves disponibilizadas pela internet.

Apesar da maioria ser *"freeware"*, algumas empresas especializadas em "add-on"s atendem a demanda pelos mesmos.

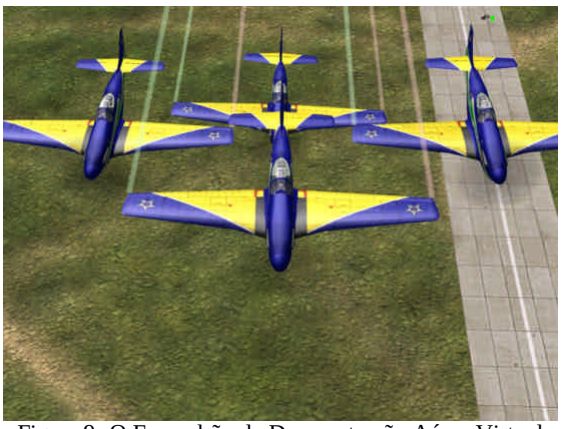

Figura 9: O Esquadrão de Demonstração Aérea Virtual implementa add-ons no IL-2 Sturmovik para simular a Esquadrilha da Fumaça

5.3 Pacotes de Expansão

Constituem-se de *"paywares"* (softwares pagos) lançados pelo próprio fabricante da plataforma. São uma continuação do jogo sem alterar sua temática, particularmente comuns em jogos de campanha, estendendo a mesma. Pode ser considerado um super pacote de "add-on"s, mas que altera drasticamente o tamanho do jogo [Salen e Zimmerman].

Comercialmente falando, servem de manutenção do ciclo de vida do produto, após sua fase de maturação. Permitindo que o mesmo continue sendo atrativo no mercado quando começaria a ter um declínio natural de suas vendas, inerente a qualquer produto. Depois de um tempo de depreciação, torna-se comum a venda do jogo e a expansão, ou expansões, num único pacote com o preço reduzido.

5.4 "Third Party"

Iniciar um projeto da estaca zero é um dos problemas que a técnica das plataformas se propõe a resolver e o faz com grande eficácia, conforme ilustramos nesse trabalho. A criação de muitos jogos é provida pela cópia autorizada de componentes de outras plataformas, reaproveitando-os num novo projeto. Por exemplo o engine do *"Star Wars Gallatic Battlegrounds"* da LucasArts declara na caixa do produto: *"Rodando com o engine do Age of Empires II"*.

O *"Genie Engine"*, desenvolvido pela Ensemble Studios para o *Age of Empires II*, foi comprado e aplicado em outro jogo com todos os seus recursos; como as aplicações multiplayer, capacidade de níveis (level design), e música de fundo, dentre outros. Outro caso é o *"The Battle for Middle-earth II"*, lançado em 2006 que utiliza o *"Sage Engine"* desenvolvido para o *"Command & Conquer: Generals"*. Um reproduz batalhas épicas da trilogia cinematográfica Senhor dos Anéis, enquanto o outro simula guerra moderna com direitos a tanques, caças e até bombas atômicas. Contudo, ambos os jogos são bastante parecidos

quanto à jogabilidade. No primeiro, conforme o jogador enfrenta inimigos, recebe pontos de comando que lhe permitem chamar reforços como as águias ou "exércitos dos mortos". Já no segundo o general ganha patente, permitindo-lhe chamar apoio aéreo de pára-quedistas ou lançar minas terrestres na base inimiga, dentre outros. Entretanto, tecnicamente falando, são jogos iguais com interfaces diferentes.

6. Estudo de Caso da Utilização de uma Plataforma na Criação de um Novo Game

Neste experimento utilizamos o "*Source SDK*" e o programa "*VTFEdit*" para modificar a plataforma "*Source*", produzida inicialmente para o jogo "*Half-Life 2*", um dos maiores sucessos do mundo em seu segmento. Conforme o modelo de plataforma proposto, não modificaremos nada do engine, o qual representa o componente fixo. Entretanto seus recursos, tais como aplicações de rede e física dos objetos serão explorados, constituindo-se nos componentes variáveis: objetos, cenários e GUI. Assim geramos um novo jogo de categoria "MOD" a partir dessa plataforma.

6.1 Introduzindo a Plataforma Escolhida: HL2

O "*Half-life 2: Source*" (HL2) é um clássico da indústria de jogos para PC [Nieborg 2005]. Como já foi mencionado neste artigo, essa plataforma foi desenvolvida pela Valve Corporation, por cerca de $40 milhões de dólares [Bramwell 2007]. Contudo, atualmente ela pode ser comprada por $54 dólares. É um sucesso desde sua primeira versão, "*Half-Life*", que vendeu cerca de 8 milhões de cópias desde seu lançamento em 1998 [Musgrove 2007]. Travou algumas batalhas judiciais contra hackers que roubaram o código fonte de seções restritas do programa, com direito a intervenção do FBI sobre os infratores. Dando continuidade à saga, o HL2 vendeu 1.7 milhão de cópias em menos de 2 meses de lançamento. É possível jogar numa campanha, atuando como o cientista Gordon Freeman, que se envolve num acidente em que criaturas alienígenas invadem a terra. Ou ainda, na versão *Deathmatch*, onde se enfrenta outras pessoas jogando em rede. Assumindo a estrutura de plataforma, tanto o HL quanto o HL2 fazem grande sucesso por gerar diversos jogos diferentes pela alteração dos componentes variáveis disponibilizados.

Pra se ter uma idéia da grande variedade de "MOD"s gerados a partir da plataforma "*Half-life 2*", estaremos apresentando sinteticamente alguns dos mais famosos nas Figuras 10, 11, 12 e 13:

Figura 10: Insurgency (http://www.insmod.net), batalhas das forças da coalizão liderada pelos EUA contra insurgentes iraquianos.

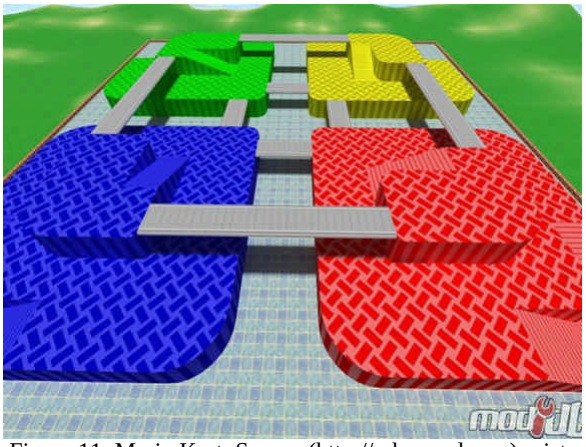

Figura 11: Mario Kart: Source (http://mks-mod.com), pista de corrida divertida revivendo o jogo Mario Kart da Nitendo

Figura 12: Dragonball: Source, baseado na série de desenho animado Dragonball Z

Figura 13: Startrek (http://www.enterprise-tcw.com), jogo que simula as operações da tripulação do filme.

Com esses poucos exemplos fica explícita a grande flexibilidade da plataforma *"Source"*, bem como os benefícios da técnica das plataformas. Chega-se o ponto em que um FPS é transformado num jogo de corrida, manuseando os componentes variáveis intencionalmente disponibilizados pela plataforma.

Dezenas de outros "MOD"s "*freeware*" estão disponibilizados, ou em desenvolvimento, sendo compartilhados em sites especializados na internet.

6.2 SBGames MOD

É o nome do jogo "criado" por nós (vide Figura 1). Por questões de limitação de tempo de desenvolvimento, ele não possui tantos detalhes e refinamentos quanto os exemplos mostrados (que demandaram meses de desenvolvimento de uma equipe competente).

Iniciando o desenvolvimento, acessamos o *"Source SDK"* (vide Figura 14) e utilizamos o programa "*Hammer Editor*", no qual é possível construir cenários e importar objetos para plataforma.

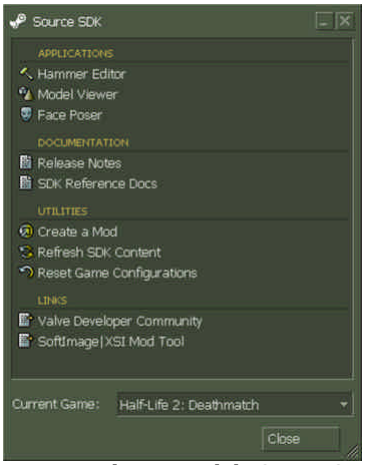

Figura 14: Tela principal do Source SDK

Construímos alguns cenários utilizando a biblioteca de texturas já presente no programa, conforme ilustrado na Figura 15.

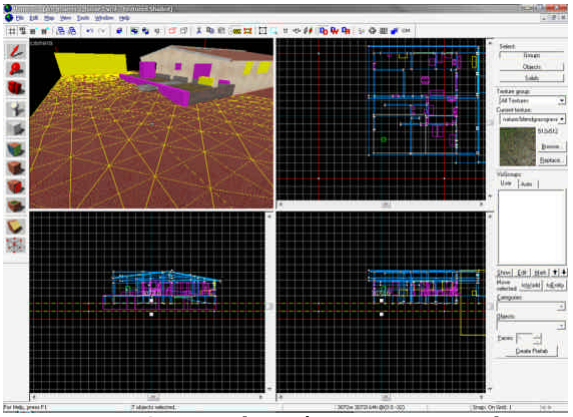
Figura 15: Construindo cenários no Hammer Editor

Quanto aos objetos, temos duas categorias: objetos espalhados pelo cenário e personagens. São construídos em softwares 3D externos, como *o "3D Studio Max"* ou *"Maya"*, e então importados pelo *"Hammer Editor"*. Contudo, pegamos emprestada a riquíssima biblioteca de objetos do *"Half-Life 2"* (Figura 16), posicionando os mesmos pelo cenário com relativa facilidade.

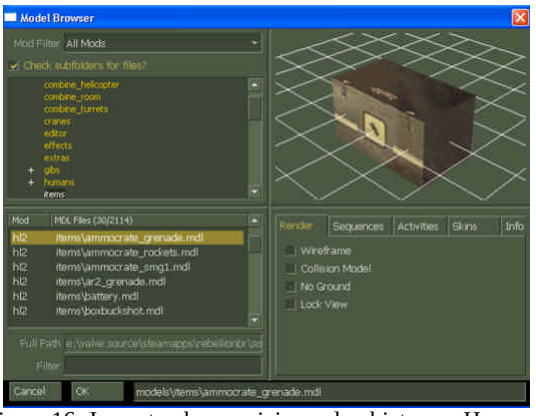

Figura 16: Importando e posicionando objetos no Hammer Editor

Para edição do comportamento dos personagens, utilizamos o *"Face Poser"*, incluso no SDK. Este programa também é empregado como visualizador de objetos e seus movimentos.

Recorremos ao software *"VTFEdit"*, não incluso no citado SDK, para alterações diversas na GUI, como plano de fundo. Acessamos o arquivo *"gameinfo.txt"*, presente no pacote de aplicações, para ajustar os itens do menu do jogo, apresentado na Figura 17.

Figura 17: GUI inicial do SBGames MOD

Finalmente, entramos na opção *"Create a MOD"* do *"Source SDK"* para criar o SBGames MOD, definindo sua pasta de arquivos e posicionando os arquivos acima desenvolvidos nela. Nas Figuras 18 e 19 ilustramos os testes do jogo concluído.

Figura 18: Jogando o SBGames MOD em rede

Figura 19: Efeitos de explosão do MOD produzidos pelo Source Engine

7. Conclusões

Com a adoção do paradigma do projeto baseado em plataforma os games atingiram uma nova dimensão. Existem inúmeros sites que compartilham alterações para jogos, além dos pacotes de expansão frequentemente lançados.

O modelo de plataforma sugerido nesse artigo, composto pelos componentes engine, objetos, cenários e GUI, mostrou-se de fácil utilização, conforme ilustramos com diversos exemplos de terceiros e com o nosso estudo de caso.

Apesar das alterações para jogos possuírem diversos grupos especializados de desenvolvimento, dos quais muitos possuem fins comerciais, elas são acessíveis também aos próprios jogadores, permitindo que eles gerem seus próprios jogos derivados da plataforma.

Em nosso estudo de caso, ilustramos como em menos de 48 horas de trabalho é possível produzir-se um jogo de relativa sofisticação utilizando o paradigma de projeto baseado em plataforma, como mostramos com a criação do SBGames MOD.

Os *"stakeholders"* diretamente envolvidos são beneficiados de diferentes formas. Os usuários usufruem de grande variedade de jogos, os desenvolvedores ganham uma reutilização eficiente de código, e os comerciantes (*"publishers"*) fermentam as vendas de suas plataformas de jogos com os novos "MOD"s, "add-on"s ou pacotes de expansão disponibilizados.

Consideramos que conseguimos ilustrar a viabilidade e utilidade do paradigma do projeto baseado em plataforma quando aplicado ao game design.

Agradecimentos

Os autores gostariam de agradecer a Williams Artimã Chaves pelo efetivo suporte no desenvolvimento do SBGames MOD e a Felipe Breyer pelos esclarecimentos diversos.

Referências

ALEXANDER, C., 1964. Notes on the Synthesis of Form. Cambridge: Harvard University Press.

BETHKE, E., 2003. Game Development and Production. Plano: Wordware Publishing.

BOULDIN, D., 2003. Platform-based System-on-Chip Design, *Proceedings of 2003 NASA Symposium on VLSI Design, 28-29 Maio*. Cour d'Alene: ID.

BRAMWELL, T., 2007. *Half-Life 2: $40 milion in, still Steaming foward* [online] Eurogamer. Disponível em: http://www.eurogamer.net/article.php?article_id=54977 [Acessado 25 julho 2007].

BÜRDEK, B., 2006. Design: História, Teoria e Prática do Design de Produtos. São Paulo: Edgard Blücher.

CHEN, R., SGROI, M., LAVAGNO, L., MARTIN, G., SANGIOVANNI-VINCENTELLI, A. E RABAEY, J., 2003. *UML and platform-based design*. UML for real: design of embedded real-time systems. Norwell: Kluwer Academic Publishers.

CLAYPOOL, K. E CLAYPOOL, M., 2005. Software engineering design: teaching software engineering through game design. *In*: *Proceedings of 10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education, Junho Capprica.* Nova Iorque: CM Press.

DE MARINIS, M., FANUCCI, L., GIAMBASTIANI, A., RENIERI, A., ROCCHI, A., ROSADINI, C., SICILIA, C., SICILIA, D., 2003. Sensor Platform Design for Automotive Applications. *Euromicro Symposium on Digital Systems Design (DSD'03), 01-06 Setembro.* Washington: IEEE Computer Society.

DOUGLASS, B., 2003. Real-Time Design Patterns: Robust Scalable Architecture for Real-Time Systems; Boston: Addison Wesley.

EBERLY, D., 2003. Game Physics. San Francisco: Morgan Kaufmann.

EBERLY, D., 2001. 3D Game Engine Design. San Francisco: Morgan Kauffman.

EL-NASR, M. E SMITH, B., 2006. Learning through game modding., *Computers in Entertainment.* Nova Iorque: ACM Press.

FULLERTON, T., SWAIN, C., HOFFMAN, S., 2004. Game Design Workshop: Designing, Prototyping, and Playtesting Games. Lawrence: CMP Books.

GOERING, R., 2002. Platform-based Design: A Choice, Not a Panacea. Washington: EETimes.

GRAAF, B., LORMANS, M., TOETENEL, H., 2003. Embedded software engineering: the state of the practice. *IEEE Software.* Los Alamitos: IEEE Computer Society Press.

LECKY-THOMPSON, G., 2002. Infinite Game Universe: Level Design, Terrain, and Sound. Rockland: Charles River Media, Inc.

MUSGROVE, M., 2007. *Half-Life 2's real battle* [online] Washington Post. Disponível em: http://www.washingtonpost.com/wp-dyn/articles/A52849-2004Nov15.html [Acessado 11 agosto 2007].

NIEBORG, D., 2005. *Am I Mod or Not? – An analysis of First Person Shooter modification culture* [online] Creative Gamers Seminar - Exploring Participatory Culture in Gaming University of Tampere. Disponível em: http://gamespace.nl/research [Acessado em 11 agosto 2007].

NSAME, P., LINZER, H., KAUTZMAN, M., LAVERE, J., KLING, G., MCGRODDY, K. E GUERRIERO, T., 2001. IBM Platform-Based Design Methodology Enablement. Micronews, Essex Junction: IBM Microeletronics, Novembro, vol. 7, n. 4, 14.

ROUSE III, R., 2001. Game Design Theory and Practice. Wordware Publishing.

RUCKER, R., 2002. Software Engineering and Computer Games. Boston: Addison-Wesley Longman Publishing

SALEN, K. E ZIMMERMAN, E., 2004. Rules of Play – Game Design Fundamentals. Cambridge: MIT Press.

SANGIOVANNI-VINCENTELLI, A., 2002. Defining Platform-based Design. *EEDesign, fevereiro*. Washington: EETimes.

VALVE, 2007. *Steam Network Status* [online] Disponível em: http://www.steampowered.com/status/game_stats.html [Acessado 20 julho 2007].