Proposta de uma Ferramenta Focada no Ensino do Desenvolvimento de Jogos Eletrônicos

Max Ricardo Benin[1] Saulo Popov Zambiasi[2]

Faculdades Barddal, Curso de Sistemas de Informação (SC)[1]
Universidade Federal de Santa Catarina, Depto de Automação e Sistemas, Florianópolis (SC)[2]

Resumo

O desenvolvimento de jogos independente tem se tornado uma realidade devido às novas ferramentas (tais como motores de jogos) com o fim de simplificar a criação de jogos digitais. Muitas dessas ferramentas visam trabalhar melhor com o hardware, fornecer formas facilitadas para acesso aos recursos gráficos, disponibilizar gerenciamento de física do jogo, Inteligência Artificial, etc. Contudo, devido ao grande impacto das ferramentas para desenvolvimento de jogos 3D, a maioria dos motores têm sido desenvolvidos para esse fim. A problemática está em que essas ferramentas são, em geral, complexas, necessitam um grande esforço para a produção de um game e são caras, e quando gratuitas, falta documentação necessária para seu aprendizado. Recentemente, jogos em 2D têm retomado sua popularidade e retornado aos computadores e consoles. Além de serem alternativas divertidas para os jogadores, essa forma de jogo facilita o desenvolvimento. O presente trabalho tem como objetivo, além de discutir a utilização de motores de jogos 2D e 3D, a produção de material didático para o ensino do desenvolvimento de jogos eletrônicos, utilizando como base o desenvolvimento de um Motor de Jogo 2D multiplataforma com a utilização de bibliotecas gratuitas.

Palavras-chave: Motores de Jogos, ferramenta de aprendizagem, desenvolvedores independentes.

Abstract

The independent development of games have become a reality due to the new tools (such as game engines) in order to simplify creation of digital games. Many of these tools are designed to work best with the hardware, providing easy ways to access graphics capabilities, providing management game physics, artificial intelligence, etc.. However, due to the great impact of tools for developing 3D games, most engines have been developed for this purpose. The problematic is that these tools are usually complex, require a lot of effort to produce a game and are expensive, and when free, have no necessary documentation for their learning. Recently, 2D games have resumed their popularity and returned to the computers and consoles. Besides being fun alternative for players, it is a easiest way of game development. Besides discussing the use of engines 2D and 3D games, this work aims production of educational materials for teaching the development of electronic games, using as basis the development of a multiplatform 2D game engine with the use of free libraries.

Keywords: Game Engine, learning tools, independent developers.

Authors' contact:

mbenin@gmail.com[1]
saulopz@gmail.com[2]

1. Introdução

Diversas tecnologias para o desenvolvimento de jogos digitais têm sido geradas para fornecer uma base para desenvolvedores produzirem seus games com maior rapidez e eficiência, sem a preocupação com detalhes como aceleração de hardware, física, inteligência artificial, gerenciamento do som, etc. Um ponto importante a se ressaltar está diretamente ligado a crescente ascensão atual do desenvolvimento de jogos independentes.

Nesse cenário, novos desenvolvedores vêm procurando alternativas baratas ou gratuitas para criar seus próprios jogos. Desta forma, o desenvolvimento independente trouxe à tona o saudosismo do desenvolvimento de jogos 2D com aplicativos retrôs ou até mesmo 2D, mas sob os paradigmas da tecnologia atual. Contudo, essas tecnologias disponíveis atualmente são, em sua maioria, ferramentas cujo foco principal é o 3D, cabendo ao desenvolvedor a tarefa de adaptar o ambiente para que seja possível desenvolver seus jogos.

Uma justificativa para a utilização de motores 2D específicos envolve os motores atuais que, quando pagos, possuem custos iniciais caros para um desenvolvedor independente. Também, tais ferramentas podem ser complexas demais para o escopo do projeto e limitadas com relação aos recursos. Mesmo com a utilização de ferramentas gratuitas, essas ainda requerem um grande esforço inicial para o aprendizado de sua utilização, considerando também a dificuldade de encontrar documentação para muitos dos esforços *opensources*. Dessa forma, o presente trabalho tem como metodologia apresentar uma proposta de desenvolvimento de uma *suite opensource*,

multiplataforma e focada no desenvolvimento de jogos em 2D. O propósito disso está na produção de material didático que possa servir de auxílio para o ensino das técnicas de desenvolvimento de jogos eletrônicos nas instituições de ensino, bem como no fornecimento da *suíte opensource* como uma alternativa gratuita e de documentação intuitiva para o desenvolvimento de jogos eletrônicos em 2D.

2. Desenvolvimento Independente

O avanço da computação nas décadas de 70 e 80, possibilitou trazer aos lares os computadores pessoais (Benin, 2007). O consumidor assumiu, desde então, uma postura técnica e entusiasta. Nesse sentido, a computação pessoal levou à possibilidade de desenvolvimento de jogos eletrônicos que poderiam ser compartilhados e produzidos entre os usuários desta tecnologia recém-introduzida, de forma *hobbista.* O desenvolvimento de jogos independentes ascende nessa época.

Segundo (Lemes, 2009) o desenvolvimento independente de jogos pode ser caracterizado por pequenos projetos, individuais ou constituído por equipes reduzidas de pessoas, cujo foco é o desenvolvimento por pura paixão ou com pretensões de se seguir a carreira no desenvolvimento de jogos a longo prazo. Outra característica é o desenvolvimento com baixo ou nenhum orçamento e até mesmo sem aportes financeiros de grandes empresas.

Da mesma forma que os filmes de classe independente, o desenvolvedor independente hoje, preocupa-se menos com os aspectos tecnológicos, que encarece a produção e mais com os aspectos criativos do game. (Holmes, 2012, apud Novak). Isso gera uma liberdade artística e criativa que em muitos casos falta em empresas que criam jogos com alto orçamento. A exemplo disso, o desenvolvimento independente trouxe à tona o saudosismo dos jogos 2D com aplicativos retrôs sob os paradigmas da tecnologia atual, ato que permitiu explorar novas formas criativas de se desenvolver jogos que por muito tempo não houve tal prática em empresas de porte maior.

3. Motores de Jogos

Um motor de jogo é caracterizado pela reutilização dos seus componentes com a finalidade de se construir diversos tipos de jogos com um mesmo conjunto de códigos. Tais componentes fazem uma separação distinta entre o código específico do jogo e o código específico do motor. Segundo Becker (2011, p.29) as finalidades como o processamento da inteligência artificial, o gerenciamento de som e música, a renderização de objetos na tela e a simulação de física em jogos, podem ser considerados motores uma vez que realizam tarefas especialistas que podem ser utilizas para diversos fins.

Quando um jogo possui os seus componentes *hard-coded,* ou seja, sem abstração alguma, como por exemplo a lógica, as regras ou a forma de renderizar um tipo específico de objeto na tela, se torna difícil de ocorrer a reutilização de componentes. Desta forma, fazer um jogo diferente com uma mesma peça de código se torna inviável. (Gregory, 2012).

Os motores de jogos para computador mais modernos intensificam a sua finalidade, em sua maioria, em torno do universo tridimensional ou 3D. Isso se dá ao fato do legado adquirido quando o termo "motor de jogo" foi popularizado. O termo surgiu na década de 90 através do código base para construção de jogos de tiro em primeira pessoa, vendidos pelos próprios desenvolvedores para gerar receita extra para as empresas da época. (Gregory, 2012). As empresas seguiram este modelo ao longo dos anos embutindo nas licenças de seus jogos, ferramentas com as quais era possível o jogador ou estúdios independentes realizar modificações do jogo original que eram, em sua maioria, jogos de tiro em primeira pessoa.

Em tempo, o mercado acabou por gerar uma necessidade mais específica. O custo para se desenvolver o próprio motor de jogo se tornou relativamente alto, possibilitando assim que empresas especializadas surgissem no mercado para suprir tal demanda. Contudo, o fato de um motor de jogo admitir reusabilidade como ponto principal que o caracteriza, pode implicar muitas vezes em um conceito errado de que o motor de jogo é uma série de "blocos de código" capaz de se adaptar para a criação de qualquer jogo ainda não imaginado (Gregory, 2012).

O desenvolvimento de um motor de jogo varia muito e depende principalmente com a forma com a qual é concebida por parte da equipe que a desenvolve. Sendo assim, tornam-se extremamente subjetivas as maneiras de apresentar a forma ideal para se desenvolver um motor. Gregory (2011, p.29) propõe uma estrutura complexa que abrange diversas camadas em um modelo *top-down* com a finalidade de evitar referências cíclicas na arquitetura proposta.

Entretanto, é possível definir o funcionamento base de um motor de jogos apenas descrevendo suas características principais a fim de atingir o nível de abstração que todos os motores desenvolvidos até então garantem, independente da sua finalidade. Para Becker (2011, p.27) as características principais que um motor de jogo deve assumir são (i) Gerenciador de Recursos, (ii) Renderização, (iii) Colisão e Física e (iv) Interfaces de Comunicação.

Assim como os jogos eletrônicos, os motores de jogos também podem possuir códigos reaproveitáveis de outras aplicações para o mesmo fim, chamados de softwares terceiros ou *third-party*. Assim, a implementação de determinados componentes pode ser

reaproveitada pelo motor de jogos através de softwares terceiros. Becker (2011, p.29) explica que finalidades como o processamento da inteligência artificial, o gerenciamento de som e música, a renderização de objetos na tela e a simulação de física em jogos, podem ser considerados motores uma vez que realizam tarefas especialistas que podem ser usadas para diversos fins, não somente para jogos eletrônicos.

4. Estado da Arte dos Motores de Jogos

Com a crescente evolução da tecnologia, o desenvolvimento de motores de jogos e softwares terceiros especializados, tornaram-se uma indústria em paralelo ao próprio desenvolvimento de jogos. Em sua maioria, os motores de jogos ainda são comercializados ou fornecidos através de versões gratuitas porém com limitações, estes desenvolvedores são ainda os líderes de mercado (Becker, 2011). Contudo, existem os que aderiram ao desenvolvimento de motores de código aberto, permitindo a participação da comunidade a fim de contribuir com ferramentas gratuitas e que possam, em alguns casos, serem modificadas de acordo com a necessidade.

Uma ferramenta de código aberto que vêm se destacado é a OGRE 3D[1] (*Object-Oriented Graphics Rendering Engine*). (Gregory, 2012 p.28) cita a OGRE 3D como um motor especializado, bem arquitetado, de fácil aprendizado, de fácil uso, escrito em C++ e voltado para aplicações 3D aceleradas via hardware. Uma característica desse motor é a sua abstração, podendo utilizar Direct3D, para aplicações específicas para o sistema operacional Windows, ou OpenGL para as demais plataformas. Ela é especificamente um motor de renderização 3D ou seja, não compete a ela tratar outras particularidades que um motor completo faria, sendo recomendada não somente para jogos, mas sim para diversas aplicações envolvendo simulação.

Outro motor semelhante e que atinge os mesmos resultados em termos de funcionalidade é o motor Irrlicht[2], cujo foco também é fornecer renderização multiplataforma inclusive usando um renderizador próprio via software caso o dispositivo utilizado não possua placa de vídeo, fundamental para renderização gráfica via hardware.

O motor Panda3D[3] é visto como um motor completo, de código aberto e que vem sido muito bem-aceito na comunidade. Isso se deve ao fato de que ela possui todos os requisitos básicos para o desenvolvimento de jogos abrangendo Motor de Renderização Avançado, Gerenciador de Som, Motor de Física (quatro opções disponíveis), Interface Gráfica do Usuário (GUI), biblioteca própria para inteligência artificial, monitor de performance e suporte total ao Python, que é uma linguagem *opensource* de script muito utilizada.

No que tange aos motores de jogos comerciais com versões gratuitas é possível destacar o *Unreal Development* Kit ou UDK[4]. A UDK é uma derivação da *Unreal Game Engine*, criada pela empresa Epic, e trata-se de uma versão gratuita do motor que deu origem aos jogos *Unreal Tournament, Gears of War* e mais recentemente ao jogo *Dishonored.* Uma característica interessante e que deve ser salientada como ponto forte é que a UDK é uma suíte completa para o desenvolvimento de jogos. Ela possui um editor visual completo que permite ao desenvolvedor concentrar todo o processo produtivo em uma única ferramenta. Além disso, a UDK introduz um sistema de programação baseado em um *script* próprio chamado *Unreal Script*. Dessa maneira, boa parte da lógica da aplicação a ser desenvolvida, e que em uma eventualidade não possa ser coberta pelo editor, pode ser facilmente programada por qualquer desenvolvedor com conhecimentos em lógica de programação, uma vez que o *Unreal Script* permite delegar funcionalidades sem que haja a necessidade do uso extensivo da linguagem de programação nativa que deu origem ao motor. Em tempo, os termos de uso da ferramenta se tornam gratuitos para fins didáticos e para o desenvolvimento de aplicações sem fins lucrativos.

Semelhante à UDK, a *CryEngine*[5] se apresenta também como uma ferramenta comercial cuja licença pode ser oferecida gratuitamente para fins educacionais e para estúdios independentes. Dessa forma, o uso integral da ferramenta é gratuito para jogos sem fins lucrativos caso contrário, uma determinada porcentagem é exigida para cada cópia do jogo vendida desenvolvido por ela.

Contudo, um motor que merece destaque é o motor Unity 3D[6]. Segundo (Blackman, 2011 p.5) A ferramenta em questão é responsável por fornecer um ambiente voltado para o desenvolvimento de jogos por pequenos estúdios, desenvolvedores independentes e entusiastas que possuem o desejo de desenvolver o seu próprio jogo em 3D, cuja possibilidade de expansão para o desenvolvimento de jogos em 2D se torna factível mediante configuração do ambiente através da compra de *plugins* de desenvolvimento. Uma característica dessa ferramenta é a sua capacidade de distribuição multiplataforma, o que torna possível a execução final de um jogo, desenvolvido por ela, em plataformas móveis, consoles de última geração e plataformas cujos sistemas operacionais variam entre Linux, Windows, e o Sistema Operacional MacOS X.

1 http://www.ogre3d.org
2 http://irrlicht.sourceforge.net
3 http://www.panda3d.org
4 http://www.unrealengine.com/udk
5 http://www.mycryengine.com
6 http://www.unity3d.com

Entretanto, a Unity 3D possui foco em sua versão comercial, limitando sua versão gratuita em diversos pontos. Essas limitações vão desde a limitação de recursos até a limitação de sua distribuição. Entretanto, ainda é uma ferramenta interessante para ser utilizada para fins acadêmicos ou mesmo comercial. Ainda citando as limitações da ferramenta, dependendo da aplicação a que se propõe, por exemplo uma aplicação com iluminação em tempo real, não é possível em sua versão gratuita.

Em tempo, vale ressaltar que o esforço para se adaptar as ferramentas 3D ora descritas para um ambiente de produção em 2D, além de utilizar recursos desnecessários, pode ser grande e impactar no tempo do desenvolvimento entre outros fatores.

Motores 2D que se destacam são os chamados "*Code-Free*" e se caracterizam por não exigir a necessidade do desenvolvimento usando linguagens de programação. Estes dão prioridade no desenvolvimento ágil, criativo e produtivo. Inspirados em ferramentas de ensino para a criação de *Story Telling* e programação de computador como a *Scratch,*[7] do Instituto de Tecnologia de Massassuchets, e a *Snap!*[8] da Universidade de Berkeley, Motores como o *Stencyl*[9] (Figura 1), *Construct* 2[10] e *Game Salad*[11] possuem características que diferem de outros motores 2D. Suas ferramentas dão a possibilidade de se programar através de blocos, formando diagramas de fluxo que dão origem às iterações dos elementos do jogo. Além disso, é muito comum encontrar nessas ferramentas outros componentes que seguem a mesma linha de produtividade, como gerenciador de atores, gerenciador de *assets*, editor de mapas embutidos na ferramenta entre outros.

Porém, essas ferramentas seguem o modelo gratuito com limitações, como por exemplo a impossibilidade de se publicar na plataforma Android, como é o caso da Game Salad, ou a impossibilidade de se publicar jogos para a plataforma Iphone, no caso da *Stencyl*.

Em uma abordagem diretamente voltada para o desenvolvimento de jogos 2D, é possível fazer destaque à ferramenta *Game Maker* [12] cujo propósito apesar de comercial, possui um apelo ao desenvolvimento através da prototipação rápida e multiplataforma. Através de uma versão gratuita com funcionalidades limitadas, é possível, segundo Nakamura et. al. (2006), usar a ferramenta *Game Maker* como uma ferramenta didática no ensino do desenvolvimento de jogos.

7 http://scratch.mit.edu
8 http://snap.berkeley.edu
9 http://www.stencyl.com
10 http://www.scirra.com/construct2
11 http://www.gamesalad.com
12 http://www.yoyogames.com/gamemaker/studio

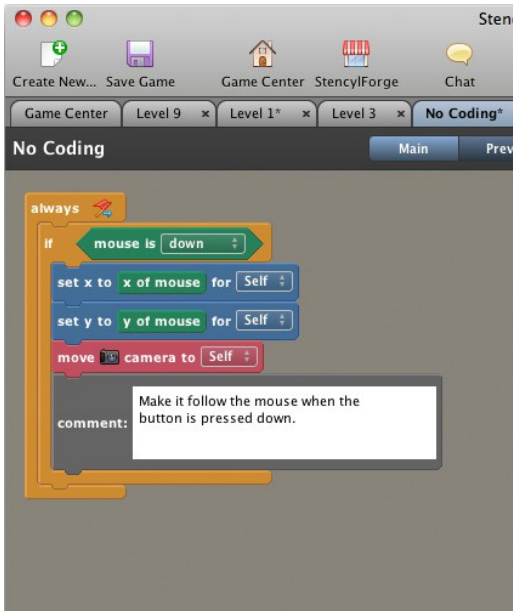

Figura 1 - Editor de blocos da ferramenta Stencyl

Itterheim et. al. (2001) cita que o motor de jogos Cocos 2D para Iphone se trata de uma solução focada no desenvolvimento de jogos em 2D para as plataformas móveis da Apple, com uma filosofia menos técnica. Embora seja um motor que não possua nenhuma ferramenta que diminua a necessidade de programação direta usando uma linguagem, a Cocos 2D evita que o programador tenha contato direto com funcionalidades de baixo nível, como por exemplo o OpenGL ES. Ao invés disso, (Itterheim et. al., 2001) explica que a maioria dos gráficos são desenhados por classes simples de *sprites*, a física é controlada por dois motores inclusos na forma de *plugin* (sob escolha do desenvolvedor em qual utilizar) e toda a lógica básica de um jogo já possui uma série de classes pré-programadas para facilitar o seu uso. A Cocos 2D é um motor gratuito, de código aberto e possui versões em diversas outras plataformas como computadores pessoais e outros dispositivos móveis como Android, Windows Phone e Bada.

5. Ferramenta Focada no Ensino de Motores 2D

A liberdade artística e criativa trouxe a possibilidade dos desenvolvedores independentes poderem se expressar de forma simples através do desenvolvimento de jogos com baixo custo operacional. Contudo, as tecnologias disponíveis atualmente, são, em sua maioria ferramentas cujo foco principal é o 3D, cabendo ao desenvolvedor a tarefa de adaptar o ambiente, ou ferramentas cujo teor de complexidade acaba impactando no escopo simples do projeto, podendo até interferir no processo criativo do mesmo.

Outro ponto a ser ressaltado é que as constantes atualizações nas grades curriculares de cursos de graduação tecnológica, tendem a fazer uso de material didático que foque no ensino do desenvolvimento de jogos eletrônicos. Tal prática pode ser tanto em cursos especializados, como em cursos direcionados a aplicações práticas que possam servir como elo para o aprendizado de disciplinas que exijam mais foco do aluno.

Dessa forma, essa sessão apresenta uma proposta de desenvolvimento de uma suíte *opensource* completa, multiplataforma e focada ao desenvolvimento de jogos em 2D. Em tempo, o seu foco está também na produção material didático para o ensino de desenvolvimento de jogos eletrônicos bem como o ensino de disciplinas correlatas em cursos de graduação de tecnologia atuando também de uma forma multidisciplinar.

Para atingir os objetivos, propõe-se uma arquitetura inicial com elementos principais e indispensáveis em motores de jogos e alguns subsistemas especializados usados para realizar trocas de informações entre os componentes do motor e o jogo construído sob essa arquitetura. A proposta pode ser analisada conforme a Figura 2. Esta é composta pelos seguintes módulos:

Gerenciador de Recursos: possui um papel fundamental na organização de todos os recursos que envolvem o jogo. Provê uma interface unificada para acesso a todo e qualquer *game asset* e outros dados de entrada do motor. Contempla também uma alternativa para o gerenciamento automático, ajudando a descarregar conteúdos já utilizados pelo sistema, a fim de economizar memória.

Renderização: Possui um papel muito importante no desenvolvimento de motores de jogos, pois é dela a responsabilidade de exibição de todo o conteúdo gráfico na tela. Envolve deste a exibição de textos, objetos primitivos até vídeos e efeitos especiais. É um dos maiores e mais complexos componentes de qualquer motor de jogo.

Front-End: Responsável por exibir os conteúdos fixos inerentes à camada da frontal como textos, menus, barras de status, pontuações e componentes que não sofrem tanta iteração.

Camada de Colisão e Física: Componente que trata de todas as simulações realistas ou semi-realistas. Sem um sistema de colisão, é impossível haver interação entre os objetos do jogo.

Interfaces de Comunicação (HID): Responsáveis por processar as ações de entrada do jogador e converter em ações internas no jogo. Tais ações são executadas por dispositivos de entrada como mouse, teclado, *joystick* e etc.

Subsistemas Especializados: São componentes inerentes ao comportamento especialista do jogo e podem variar de acordo com o estilo do jogo que se propõe. Algumas características são implementações de componentes de gerenciamento de atores, sistemas de câmera, inteligência artificial, gerenciamento de veículos etc.

Componentes Lógicos: São gerenciadores da lógica do sistema. Coordenam especificamente tudo o que tange o tempo *(timers)* e as chamadas de recursos lógicos.

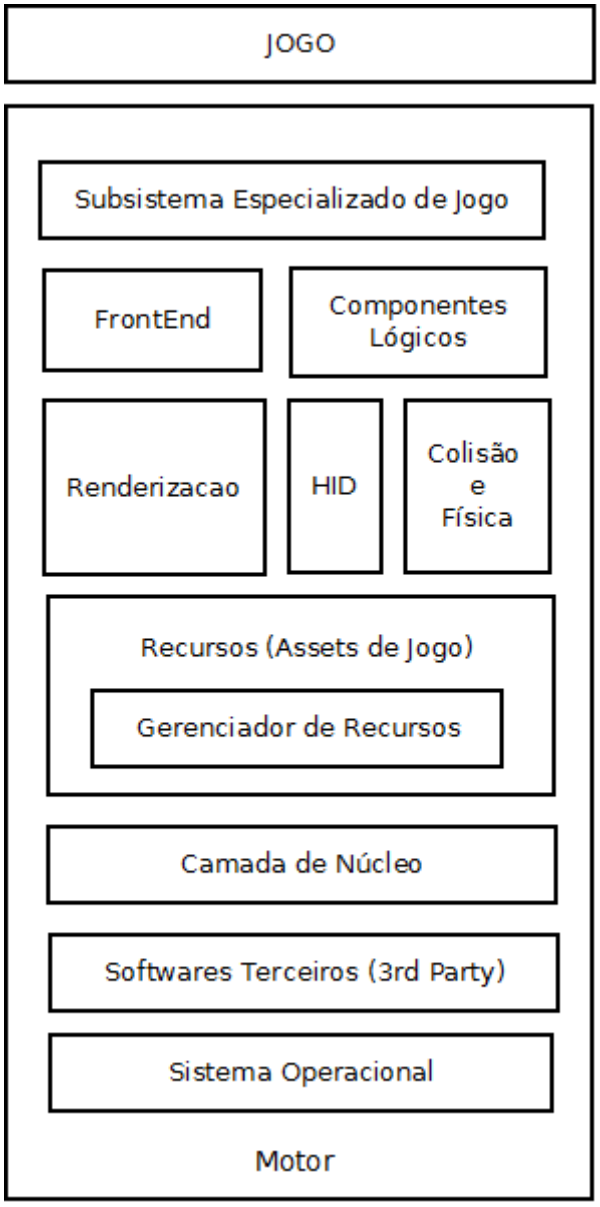

Figura 2 - Arquitetura em camada proposta para um motor 2D.

Cada componente proposto será desenvolvido em módulo de acordo com as suas dependências diretas. Leva-se em consideração que as camadas de Sistema Operacional, Softwares Terceiros e Camada de Núcleo conterão algoritmos de terceiros desde que gratuito e em uma versão estável.

Em seu estado inicial, para fins de desenvolvimento ágil e para iniciar a geração de material didático, os módulos terão uma única dependência, a da biblioteca *3rd Party* chamada SFML (*Simple and Fast Multimedia Library)* que fornece a abstração entre os Sistemas Operacionais Linux, Windows e MacOs X para o gerenciamento de eventos de sistema, janelas, som, rede e dispositivos de entrada de dados. Isso se tornará extremamente útil uma vez que o desenvolvimento de componentes específicos para cada sistema operacional demanda tempo, desta forma o foco é mantido somente na criação do motor em si.

A geração de material didático se dará de duas formas. 1) Cada módulo desenvolvido servirá como caso de teste para disciplinas específicas de programação que não necessariamente podem estar envolvidas com o desenvolvimento de jogos eletrônicos uma vez que é conteúdo relacionado a algoritmos e técnicas de programação; 2) Cada *release* estável do motor será utilizado como ferramenta para o ensino das técnicas especialistas do desenvolvimento de jogos eletrônicos. Isso também gerará material didático para o ensino em sala de aula.

6. Considerações Finais

Este trabalho apresentou uma relação fundamental que o desenvolvimento independente tem com a criatividade e com o estado da arte no desenvolvimento de jogos com baixo orçamento. Por meio dessa premissa, propôs que o desenvolvedor independente necessita de uma ferramenta que não tenha limitações tecnológicas e que seja o suficientemente gratuita e igualmente funcional em relação às soluções comerciais. Outro ponto ressaltado foi a necessidade de uma ferramenta específicas para os jogos atualmente produzidos pelos desenvolvedores independente que, em sua maioria, são jogos 2D.

Fez-se necessário realizar uma abordagem sobre o conceito dos motores de jogos e o seu estado da arte atualmente, abrangendo as principais soluções tanto gratuitas quanto comerciais. Isso serviu para salientar que muitas soluções, embora profissionais, são em sua maioria limitadas ou mal documentadas.

Com base nos motores atuais, foi proposto a criação de um motor 2D de código aberto e gratuito que atenda a todos os requisitos para o desenvolvimento de aplicações interativas independentes. A arquitetura inicial foi concebida de acordo com os elementos principais que um motor de jogos deve possuir. A proposta também deve gerar material didático para o ensino do desenvolvimento de jogos eletrônicos em sala de aula e para o auxílio de disciplinas da área de tecnologia.

Como propostas futuras, se tem planejada a geração de material de pesquisa específico para eventos onde a discussão do tema seja conveniente, aliado à evolução do desenvolvimento do motor em si. A revisão bem como a adição de novos componentes na arquitetura proposta é outra proposta futura uma vez que o desenvolvimento de um protótipo funcional irá ajudar a definir os componentes necessários.

Agradecimentos

Agradecimentos ao apoio do Grupo de Pesquisa Subverse - grupo de pesquisa em ciberarte.

Referências

GREGORY, J. **Game Engine Architecture**. Florida: Taylor and Francis Group,LLC, 2009.

NOVAK, J. **Desenvolvimento de games**. Tradução da 2ª. Edição Norte Americana ; tradução Pedro Cesar de Conti ; revisão técnica Paulo Marcos Figueiredo de Andrade. São Paulo : Cengage Learning, 2010.

LEMES, D. O. **Games Independentes: Fundamentos metodológicos para criação, planejamento e desenvolvimento de jogos digitais**. Programa de Pós-Graduação em Tecnologias da Inteligência e Design Digital. Dissertação de Mestrado. Orientação: Luís Carlos Petry. Pontifícia Universidade Católica de São Paulo - PUC: SP, 2009.

BECKER, H. O. **Desenvolvimento de Motores de Jogos 2D: Um Estudo de Caso**. Monografia. Faculdades Barddal. Florianópolis, 2011.

BLACKMAN, S. **Beginning 3D Game Development with Unity: World´s most widely used multi-platform game engine.** New York: Apress, 2011.

NAKAMURA, R., BERNARDES J. TORI R. **enJine: Architecture and Application of an Open-Source Didactic Game Engine** In: Simpósio Brasileiro de Jogos para Computador e Entretenimento Digital, 2006, Recife. Anais do V Simpósio Brasileiro de Jogos para Computador e Entretenimento Digital, 2006.

ITTERHEIM, S., LÖW A. **Learn cocos2d Game Development with IOS 5.** USA: Learn Apress, 2011