Análise da Evolução de Engines de Jogos

Daniel Scherer, Daniele Ventura Batista, Aline de Cantalice Mendes

Departamento de Computação – Universidade Estadual da Paraiba (UEPB)
R. Baraúnas, 351 - Universitário, Campina Grande - PB, 58429-500

scherer@servidor.uepb.edu.br,{daniele.batista,aline.mendes}@aluno.uepb.edu.br

Abstract. *Software development, in particular, of games has always attracted attention as a gateway for new developers to enter the world of computing. Initially, it required knowledge of highly complex programming languages (eg. Assembler); after that it was necessary to have knowledge in a high level language such as C ++ and, finally, recently the space seems to have been dominated by Engines, like Unity and Unreal. It is interesting to note that there was a migration in the form of development, leaving demands for specific programming knowledge reaching situations in which the development is based almost exclusively on designing the desired game. In this way, this work seeks to analyze the game development engines, seeking to expose their advantages and disadvantages, what resources, community, support and documentation, as well as what knowledge requirements so that the current new developer can enter this world.*

Resumo. *O desenvolvimento de software, em particular, de jogos sempre atraiu a atenção como uma porta de entrada para novos desenvolvedores entrarem no mundo da computação. Inicialmente, era necessário o conhecimento de linguagens de programação altamente complexas (por exemplo, Assembly); depois disso, foi necessário ter conhecimento em uma linguagem de alto nível como C ++ e, finalmente, recentemente o espaço parece ter sido dominado por motores, como Unity e Unreal. É interessante notar que houve uma migração na forma de desenvolvimento, deixando demandas por conhecimentos específicos de programação atingindo situações nas quais o desenvolvimento se baseia quase que exclusivamente no design do jogo desejado. Dessa forma, este trabalho busca analisar os mecanismos de desenvolvimento de jogos, buscando expor suas vantagens e desvantagens, quais recursos, comunidade, suporte e documentação, além de quais requisitos de conhecimento para que o novo desenvolvedor atual possa entrar neste mundo.*

1. Introdução

Desenvolvimento de jogos normalmente foram portas de entrada para a definição pela carreira da computação. O desejo de criar um jogo melhor do que os existentes; em que não haveria as falhas de projeto; ou mesmo que contassem a estória imaginada. No princípio com as implementações em Assembly que exigiam conhecimentos do hardware, mas possibilitavam a obtenção de um resultado mais robusto; passando pelas implementações em linguagens intermediárias, tal como Basic, que eram mais simples de codificar, porém resultavam em softwares demasiadamente simples. Houve o avanço

com o uso de C e, mais tarde, com C++. Tais linguagens permitiam, inclusive, o uso de trechos de código Assembly. Com isto, há resultados mais robustos.

Porém estas linguagens, por mais completas que fossem, ainda exigiam grande conhecimento de desenvolvimento. Além disto, exigiam *know-how* sobre o desenvolvimento gráfico, uma vez que tudo deveria ser implementado na linguagem. Há uma alteração neste caminho com o surgimento das bibliotecas gráficas: DirectX (primeiro release em 1995 [Microsoft 2007]), OpenGL (primeiro release em 1992 [Group 2019]) e Mesa (primeiro release em 1995 [Mesa3D 2020]).

Estas bibliotecas permitiram fornecer aos desenvolvedores conjuntos de comandos que permitiam utilizar objetos e realizar operações, principalmente em aplicações 3D. O DirectX foi a resposta da Microsoft ao OpenGL. Já o Mesa era uma versão *open-source* do OpenGL. Assim, os jogos puderam evoluir significativamente; principalmente pois as bibliotecas permitiam facilitar o manuseio de multimídia nos jogos e aproveitar mais o potencial das placas de vídeo dos computadores; além de reduzir a complexidade para os desenvolvedores.

A demanda do mercado por jogos mais realistas (tanto em termos gráficos quanto em termos de conceitos de física), bem como a necessidade por lançamentos cada vez mais rápidos (muitos jogos no início dos anos 2000 levavam em média 5 anos para serem concluídos); colocam as Engines como possíveis soluções, pois são frameworks projetados para reduzir o custo, complexidade e tempo para entrega ao mercado; possibilita exigir menos conhecimentos de desenvolvimento, buscando deixar o desenvolvedor focado no *gameplay*[Halpern 2018].

Este trabalho apresenta um *review* sobre a evolução de *Game Engines*. Partindo On-line Systems de 1979, chegado a AppGameKit de 2016. Na próxima seção contém uma discussão sobre a evolução no desenvolvimento de Jogos e segue com a apresentação das *engines* analisadas. Após tem-se a discussão comparativa. Seguido das considerações finais e trabalhos futuros.

2. A evolução no desenvolvimento de Jogos

Em 1961 foi criado o jogo Spacewar, desenvolvido pelo engenheiro americano Steve Russel e Peter Samson. Spacewar foi o primeiro jogo distribuído comercialmente, mas poucas pessoas possuíam computadores para poder jogar. O jogo consumiu 200 horas de trabalho, mesmo sendo um jogo muito simples o desenvolvimento foi longo.

De acordo com Henry Lowood (2014), o jogo DOOM (1994) marcou o início dos jogos de computadores modernos, não apenas como uma conquista no mundo da programação, mas como um melhoramento para uma série de mudanças na percepção dos jogos. Por causa da popularidade de jogos como esse, houve mais comunidades, conteúdo modificável, fascínio pelas imagens e sons de jogos, bem como preocupações com representações hiper-realistas de violência [Lowood 2015].

Com a popularidade dos jogos DOOM (1994) e Quake (1996) ao invés dos novos desenvolvedores construírem um jogo do zero eles utilizaram os núcleos dos jogos como base de suas próprias criações.

Alguns dos primeiros Motores de jogos foram criados na década de 1980 como o Adventure Game Interpreter (AGI) de 1984 e o sistema SCUMM da LucasArts de 1987. Mas o termo Game Engine só começou a ser utilizado na década de 1990. Jason

Gregory (2015), expôs que, praticamente todos os motores de jogos, contêm um conjunto de componentes principais, tais como mecanismo de renderização, mecanismo de colisão e física, sistema de animação, sistema de áudio e sistema de inteligência artificial[Gregory 2015].

Segundo, Esdras Rocha (2013), *Game Engines* (Motores de jogos) permitem facilitar a reutilização de software, aumentando assim a praticidade, produtividade, mantendo o foco diretamente para as regras, funcionalidades, características e fundamentos do produto: o jogo [Oliveira 2013].

Antes das Game Engines os códigos dos jogos eram escritos á mão, assim como cada pixel que seria mostrado na tela. Era necessária toda uma equipe de programadores para desenvolver um jogo. A cada novo jogo desenvolvido era escrito um novo código do zero, levando muito tempo para ser finalizado. Os desenvolvedores tinham que considerar as limitações da época, manter o código o mais simples possível para fazer uso otimizado do hardware.

Segundo, Jason Gregory (2015), sobre o mercado de jogos nos dias de hoje, os jogos são uma indústria de vários bilhões de dólares que compete com Hollywood em tamanho e popularidade. Sendo que o software que impulsiona essa popularidade são os Motores de jogos, tais como Quake e Doomda id Software, EpicUnrealEngine 4, ValveSourceEngine e o Unity. Se tornaram kits de desenvolvimento de software reutilizáveis com todos os recursos que podem ser licenciados e usados para criar quase qualquer jogo imaginável [Gregory 2015].

A evolução contínua das *Game Engines* criou uma forte separação entre renderização, script, arte e design de níveis. Com o avanço das tecnologias e das Engines elas não são mais limitadas só para desenvolvimento de jogos, estão sendo usadas em outras áreas como na medicina em treinamento cirúrgico simulado [Marks et al. 2008]e também esta sendo inserida na área de arquitetura para simulações [Buyuksalih et al. 2017].

3. Engines de Jogos

Nesta seção são apresentadas as engines, com explicitação de suas vantagens e desvantagens.

3.1. RPG Maker [RPGMaker]

RPG Maker (1997) criada pelo grupo japonês ASCII e sucedido pela Enterbrain. Era específica para o desenvolvimento de jogos RPG 2D. Para a criação do cenário, o jogo oferece uma biblioteca de opções, onde são oferecidas imagens prontas demonstros e heróis[Cruz et al. 2010].Vários elementos do jogo já vêm pré-prontos, a criação de um jogo do gênero RPG é simples e pode ser mais facilmente aprendida em relação a outros Motores de jogos[Cruz et al. 2010]. Exporta para varias plataformas como Windows, MacOSX, Android, iOS e HTML 5. A RPG Maker possui tradução para português.

RPG Maker possui uma versão Trial, mas o preço da engine pode variar de R$ 19 a R$ 79 [RPGMaker]. A linguagem usada para a programação dos jogos é o JavaScript.

3.2. Unreal[Games 2004]

Unreal (1998) foi desenvolvida pela Epic Games e usada pela primeira vez em um jogo de tiro em primeira pessoa: o jogo Unreal, que utilizou pela primeira vez cores 16 bits, o programador de jogos John D. Carmack II comentou que duvidava que qualquer jogo importante seria projetado com cores de 8 bits a partir do uso que a Unreal fez com 16 bits. A Engine foi específica para desenvolvimento de jogos 3D de alta qualidade e possui uma interface intuitiva. A linguagem usada é o C++. Inicialmente era uma Engine paga, mas na sua última versão (Unrealengine 4) foi disponibilizada gratuitamente. Porém, se seu desejo é monetizar seus jogos, você terá que pagar 5% para a engine quando seu produto for bem sucedido [Games 2004]. Unreal permite a exportação dos seus projetos para Microsoft Windows, Linux, Mac OS, entre outros.

3.3. M.U.G.E.N [MUGEN 1999]

M.U.G.E.N (1999) foi desenvolvido por Elecbyte com o objetivo inicial de produzir um jogo de tiro mas, devido a carência de bons jogos de luta para PC, a Elecbyte mudou seu objetivo e desenvolveu a M.U.G.E.N. como engine específica para desenvolvimento de jogos de luta 2D[Elecbyte 1999]. A engine é gratuita para uso não comercial, porém teve sua última atualização disponibilizada em 2013. M.U.G.E.N foi desenvolvida para pessoas com pouca experiência em programação, mas com talento artístico e paciência para aprender[Elecbyte 1999]. Jogos criados na MUGEN podem ser exportados para Linux, DOS, Windows e OS X.

3.4. Game Maker [Games 2013] [Overmars 2006]

Game Maker (1999) criada por Mark Overmars. Possui programação intuitiva onde o desenvolvedor não precisa conhecer nenhuma linguagem de programação, pois poderá programar o jogo de forma visual, no estilo *DragandDrop*[Overmars 2006].Tem uma linguagem própria (Game MakerLanguage) [Overmars 2006] também permite a linguagem C. Jogos desenvolvidos pelo Game Maker poderão ser exportados para várias plataformas como iOS, Android, Windows Phone, entre outros. As versões pagas existem como: *Standart*, *Professional* ou *MastterCollection*. Possui a possibilidade de estender jogos usando DLLs, esta funcionalidade só está disponível na versão Pro do Game Maker. DLLS estendem o poder do Game Maker. Eles podem: Adicionar novas opções de códigos, ações etc.[Overmars 2006].

3.5.CryEngine [CryEngine 2019]

CryEngine (2001) foi desenvolvida pela empresa Crytek. Possui suporte para desenvolvimento de jogos 3D. Os jogos podem ser exportados para Windows, OS X, Linux, PlayStation, entre outros.ACryEngine gratuita possui limitações, mas permite comercializar seus jogos sem precisar pagar pelo uso. Caso deseje recursos extras, estes são pagos. Ela suporta somente a linguagem C#. A Engine possui iluminação global dinâmica totalmente em tempo real e permite criar ambientes ultra-realistas[CryEngine 2019],comoaguá 3D de alta qualidade que foi utilizado na criação do jogo FarCry.

3.6. Ren'Py [Ren'Py 2005]

Ren'Py (2004) foi criada por Tom Rothamel. É uma engine específica para jogos com histórias interativas 2D, como romances visuais e jogos de simulação de vida. A linguagem usada é o Python por meio de script. Ren'Py é de código aberto e gratuito para uso comercial. Ren'Py permite a exportação dos seus projetos para Android, Montagem em HTML5 / Web (Beta), Linux, Windows, Mac OS, iOS. A engine possui alguns efeitos poderosos como animação de imagem e o suporte a reprodução de arquivos de filme[Ren'Py 2005].

3.7. Unity 3D [Unity 2020]

Unity 3D (2005) foi desenvolvida pela Unidy Technologies. A engine possui suporte para desenvolvimento de jogos em 2D e 3D. A Unity 3D tem uma versão gratuita que possui quase todos os recursos da engine, podendo até comercializar seus jogos pela versão gratuita, só é necessário possuir uma versão paga se o desenvolvedor faturar US$100.000/ano [Unity 2020]. Unity 3D consta com um mecanismo para criação de terreno 3D detalhado de alta qualidade e sombras em tempo real. Unity 3D exporta seus projetos para 27 plataformas, algumas delas são: Android, iOS,Windows Phone entre outros [Creighton 2010].A linguagem principal usada na programação é C# ou JavaScript.

3.8. 001 Game Creator [001GameCreator 2006]

001 Game Creator (2006) foi desenvolvida por Engine001. Suporte para jogos 2D, não precisando de conhecimentos de programação. A engine contém programação por Script gráfico(Scripts de texto), com essa funcionalidade de script assistido por gráficos[001GameCreator 2006].001 Game Creator não é uma engine gratuita, está custando R$ 109,99, podendo ser comprada direto na Steam, mas possui uma versão de teste, está disponível para ser feito o download de graça no site oficial da engine. As plataformas que a engine suporta para exportar seus jogos são Windows, HTML5, Android e iOS. Possui tradução para o português.

3.9. Construct 2 [Scirra 2020]

Construct 2 (2007) foi desenvolvido pela ScirraLtd. Específica para desenvolvimento de jogos 2D. Não necessita que o desenvolvedor tenha conhecimento em linguagem de programação, pois a criação do jogo é feita toda de forma visual através de objetos já programados possuindo comportamentos e componentes já definidos [BarassuolandChicon 2017].Atualmente a Construct 2 tem uma versão gratuita com alguns recursos limitados (ex efeitos especiais entre outros), o usuário não poderá comercializar seus jogos [BarassuolandChicon 2017].O usuário que desejar possuir todos os recursos da engine deverá pagar uma quantia, podendo custar até R$2.249,99[Scirra 2020].Antigamente era utilizado o script Python na engine, mas foi retirado, citando complicações com a execução do Python nos navegadores e a complexidade geral de manter um sistema de script compatível. Um plug-in JavaScript SDK foi introduzido como um substituto.

3.10. 3D Game Builder [Builder 2009][de Lima 2008]

3D Game Builder (2008) foi criada pelo brasileiro Edirlei Everson Soares. A engine brasileira possui suporte para desenvolvimento de jogos 3D. Programação em forma de eventos, criando assim comportamentos e regras para o jogo. A linguagem utilizada na

engine para programação de scripts é Pascal sendo recomendada tanto para profissionais ou iniciantes. O 3D Game Build contém tutoriais em português para auxiliar na manipulação de ferramentas. A engine possuiu uma versão paga, as licenças variavam de R$ 114,00 a R$ 1.500,00, mas ultimamente ela está completamente gratuita.

3.11.Game Salad [GameSalad 2009]

Game Salad (2009) desenvolvida pela GameSalad. Engine indicada para iniciantes no mundo de desenvolvimento de jogos. Possuindo suporte para jogos 2D é uma engine sem programação, onde todos os objetos são arrastados e conectados, criando assim comportamentos e funcionalidades para o jogo. Game Salad possui duas versões disponíveis: Basic que custa US$ 199/ano e a versão Pro US$299/ano. Os jogos podem ser exportados para Windows, Mac, smartphones e podem rodar no navegador.

3.12. Cocos 2D-X [Cocos2d-X 2010]

Cocos 2D-X (2010) foi desenvolvida por Chukong Technologies. Engine de código aberto, com licença MIT, é totalmente gratuita [Andrade andKnop].Jogos criados nela podem ser comercializados sem o desenvolvedor se preocupar em pagar pela engine. Linguagens aceitas na sua programação são Lua, C++ e Javascript[Andrade andKnop].Plataformas para exportação de jogos: Android, iOS, Tizen, Windows, Linux e Mac.

3.13. Stencyl [Stencyl 2020]

Stencyl (2011) foi criado por Jonathan Chung. Engine com suporte para jogos 2D, onde se criam jogos sem programar, mas precisa entender de lógica de programação. A programação na Stencyl se resume em blocos, onde cada bloco representa um comando, arrastando e conectando esses blocos são criados os comportamentos e funcionalidades no jogo. A Stencyl possui três versões: a versão gratuita só permite a publicação em plataforma Web, é indicada para uso pessoal e individual; a versão Indie permite exportar para Web ou Desktop e custa US$ 99/ano; a versão Studio custa US$ 199/ano e permite também exportar para mobile tais como Android ou IOS [Stencyl 2020].

3.14. GDevelop [GDevelop 2013]

GDevelop (2013) foi desenvolvida por Florian Rival. Engine com suporte para desenvolvimento 2D não necessitando de nenhuma habilidade em programação pois a criação do jogo se resume a programar visualmente, por meio de eventos [Correa 2015]e assim criando comportamentos e regras para o jogo. GDevelop é uma engine de código aberto sendo possível comercializar os jogos de forma livre, não precisando pagar quantias para exportar e vender os jogos. Podem ser exportados para Web (HTML5) ou para Android, Windows, entre outros[Correa 2015]. Aceita a linguagem JavaScript.

3.15. Godot [Godot 2020]

Godot (2014) foi desenvolvida por Juan Linietsky e Ariel Manzur. Sendo 100% gratuita e de código aberto, a engine é excelente para criação de jogos 3D e 2D. Jogos feitos com a Godot podem ser exportados para múltiplas plataformas, tais como: Navegadores Web, Windows, Linux, Android, IOS. A engine conta com diversos recursos, como uma linguagem de programação própria GDScript (muito semelhante a Python). As linguagens suportadas oficialmente em Godot são GDScript, Visual Scripting, C++ e C#. O sistema de arquivos da engine armazena recursos no disco. Qualquer coisa, desde

um roteiro até uma cena ou uma imagem PNG, é um recurso para o motor [Godot 2020].

3.16. Xenko [Onyskiv 2016]

Xenko (2014), atualmente conhecida por Stride, foi desenvolvido originalmente pelo Silicon Studio. Engine recente, foi lançada como uma engine comercial, mas atualmente ficou gratuita e de código aberto. Desenvolvimento de jogos 2D e 3D. Utiliza a linguagem C#. Engine com suporte para as plataformas Xbox One, Android e iOS e Linux. Com a Xenco, não há necessidade de ser um especialista em renderização para criar jogos de ótima aparência. O mecanismo faz todo o trabalho, para que o usuário possa se concentrar nas coisas importantes [Stride 2020].

3.17. BuildBox [BuildBox 2020]

BuildBox (2014) foi criada por Trey Smith. Engine com suporte para desenvolvimento de jogos 2D e 3D não necessitando de nenhuma habilidade em programação pois a criação do jogo se resume a programar visualmente. Os jogos podem ser exportados para Android, Windows, IOS, Amazon, Google Play, tvOs, Steam, Windows Store[Audronis 2016].BuildBox possui uma versão gratuita, mas com algumas limitações (jogos compilados na versão gratuita possuirão um logotipo, que não pode ser removido). Além disso, a exportação é limitada apenas ao iOS e Android. Para acesso a todos os recursos da Engine deve-se pagar R$53.19/ano ou a versão mais completa por R$349.99/ano [BuildBox 2020].

3.18. AppGameKit [AppGameKit 2019]

AppGameKit (2016) foi desenvolvida pela empresa Dark Basic Software Limited. Engine com suporte para desenvolvimento de jogos 2D e 3D. A AppGameKit possui duas versões: Classic, que custa US$ 49,99, e a versão Studio que já vem com o editor visual e está custando US$ 99,00. Uma vantagem é que ela está disponível para avaliação gratuita, mas essa versão não possui alguns recursos como exportar projetos para Android, transmitir aplicativos diretamente para seus dispositivos e remover marca d'água de projetos compilados. A engine tem sua própria linguagem AppGameKit Script, mas os jogos também poderão ser programados em C++ [Peñalba et al. 2012].

4. Considerações

Com a evolução dos Motores de jogos (Figura 1) o desenvolvedor já não precisa se preocupar em como funcionará o mecanismo de mira, como programar o comportamento dos inimigos (ex. para um jogo de FPS). Mas sim, montar a história com as ferramentas disponíveis pois hoje as *Game Engines* já possuem inteligência artificial, ferramentas pré-programadas, mapas prontos e algumas, inclusive, possuem *templates* prontos (ex. Engine M.U.G.E.N que é especifica para jogos de lutas 2D e já vem com nove modos de jogos pré-programados para serem editados).

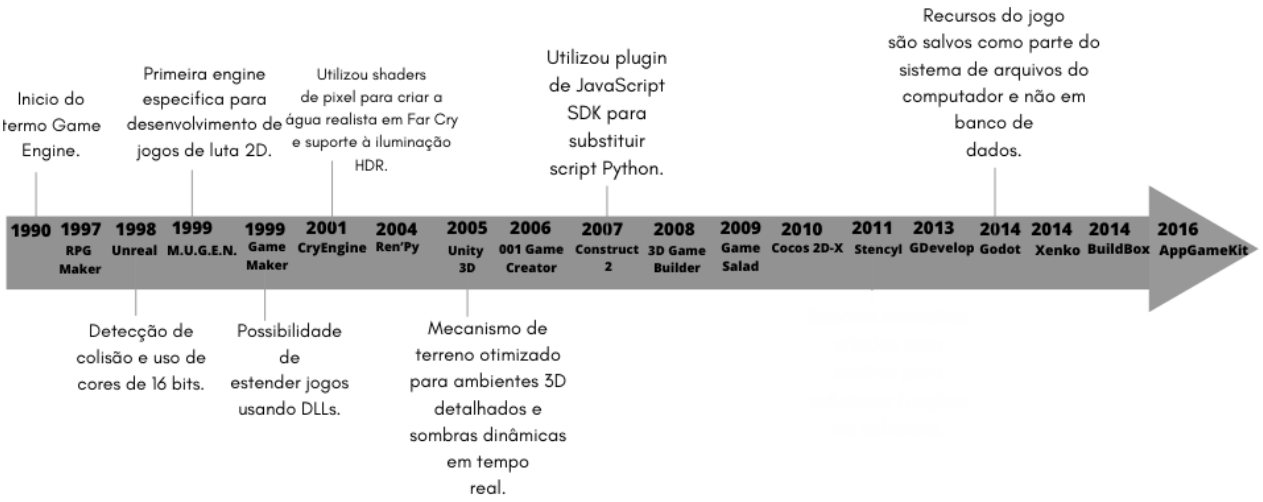

Figura 1. TimeLineEngines

Tabela 1. Tabela de comparativo entre engines

	Específico para jogos:	Possui tradução para PT?	Linguagem utilizada	Possui tutorial?	Possui documentação?	Exporta projetos para:							Versão gratuita	Versão paga	Suporte a VR
						Android	iOS	Windows Phone	HTML5	Linux	Windows	MacOS			
RPG Maker(1997)	RPG 2D	Sim	JavaScript	Sim	Não	Sim	Sim	Não	Sim	Não	Sim	Sim	Versão Trial	Sim	Sim
Unreal (1998)	3D	Sim	C++	Sim	Sim	Sim	Sim	Não	Sim	Sim	Sim	Sim	Sim	Sim	Sim
M.U.G.E.N (1999)	Luta 2D	Não	HTML	Sim	Sim	Não	Não	Não	Não	Sim	Sim	Sim	Sim	Não	Não
Game Maker (1999)	2D	Não	Game Maker Language e C.	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Limitada	Sim	Não
CryEngine (2001)	3D	Não	C++ e Lua.	Sim	Sim	Sim	Sim	Não	Não	Sim	Sim	Sim	Sim	Não	Sim
Ren'Py (2004)	2D	Não	Python	Sim	Sim	Sim	Sim	Não	Sim	Sim	Sim	Sim	Sim	Não	Não
Unity 3D (2005)	2D e 3D	Não	C# e JavaScript	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim
001 Game Creator (2006)	2D	Não	script gráfico	Sim	Sim	Sim	Sim	Não	Sim	Não	Sim	Sim	Versão Trial	Sim	Não
Construct 2 (2007)	2D	Não	HTML5 e Javascript	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Limitada	Sim	Não
3D Game Builder (2008)	3D	Sim	Pascal	Sim	Sim	Não	Não	Sim	Sim	Sim	Sim	Sim	Sim	Não	Não
Game Salad (2009)	2D	Não	Lua	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Versão Trial	Sim	Não
Cocos 2D-X (2010)	2D	Não	Lua, C++ e Javascript	Sim	Sim	Sim	Sim	Não	Sim	Sim	Sim	Sim	Sim	Não	Não
Stencyl (2011)	2D	Não	Haxe	Sim	Sim	Sim	Sim	Não	Sim	Sim	Sim	Sim	Limitada	Sim	Não
GDevelop (2013)	2D	Não	JavaScript	Sim	Sim	Sim	Sim	Não	Sim	Sim	Sim	Sim	Sim	Não	Não
Godot (2014)	2D e 3D	Sim	GDScript, Visual Scripting, C++ e C#	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Não	Sim
Xenko (2014)	2D e 3D	Não	C#	Sim	Sim	Sim	Sim	Não	Não	Sim	Sim	Não	Sim	Não	Sim
BuildBox (2014)	2D e 3D	Não	JavaScript	Sim	Sim	Sim	Sim	Não	Não	Sim	Sim	Sim	Limitada	Sim	Não
AppGameKit (2016)	2D e 3D	Não	AppGameKit Script e C++	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Versão Trial	Sim	Sim

Este trabalho teve por objetivo analisar Motores de Jogo (Tabela 1), mostrando seus potenciais, para discutir a facilidade de um não profissional na área conseguir desenvolver seu próprio jogo. Embora algumas das Engines mostradas necessitem de um nível intermediário de programação para serem usadas, há Engines que não necessitam nenhum conhecimento básico de programação. Muitas já possuem programação visual, sendo assim mais fácil e intuitivo um novo desenvolvedor conseguir iniciar e finalizar seu projeto. Próximos passos desta pesquisa envolvem o aprofundamento sobre engines gratuitas (ex. Godot) e possibilidade de aplicação tanto para aprendizagem de conceitos de computação quanto o desenvolvimento de aplicações não diretamente relacionadas a jogos.

5.Referências

001GameCreator (2006). “Make games. no coding.”, https://001gamecreator.com/, Acesso em: Maio de 2020.

Andrade, M. M. andKnop, I. O. “Projeto e desenvolvimento de jogos eletrônicos multiplataforma:um estudo de caso utilizando cocos2d-x”.

AppGameKit (2019). “Appgamekit.”, https://www.appgamekit.com/about-agk, Maio.

Audronis, T. (2016). *Buildbox 2.x Game Development*. Packt Publishing.

Barassuol, J. B. and Chicon, P. M. M. (2017). “O desenvolvimento do raciocínio lógico através da engineconstruct 2”.

BuildBox (2020). “Buildbox.”, https://www.buildbox.com//, Acesso em: Maio de 2020.

Builder,D.G.(2009).“3d game builder.” http://www.3dgamebuilder.com.br/en/3dgamebuilder/index.php, Acesso em: Maio de 2020.

Buyuksalih, I., Bayburt, S., Buyuksalih, G., Baskaraca, A., Karim, H., and Rahman, A. A.(2017). “3D Modelling and visualization based the Unity Engine-Advantages and Challenges”.

Cocos2d-X (2010). “Cocos2d-x framework.”, https://www.cocos.com/en/cocos2dx, Acesso em: Maio de 2020.

Correa, J. (2015). *Digitopolis II: Creation of video games GDevelop. Digitopolis*.

Creighton, R. H. (2010). *Unity 3D Game Development by Example*. PacktPublishing.

Cruz, D. M., de Albuquerque, R. M., and de Abreu Azevedo, V. (2010). “Rpgmaker como ferramenta pedagógica: produzindo jogoseletrônicos com crianças.”

CryEngine (2019). “Features.”, https://www.cryengine.com/features, Acesso em: Maio de 2020.

de Lima, E. S. (2008). “3d game builder”.

de Lima, J. E. S. and de Paula Filho, P. L. (2010). “3d game builder: Uma game engine para criação de ambientes tridimensionais”.

Elecbyte (1999). “M.u.g.e.n.”,http://www.elecbyte.com/mugendocs-11b1/readme.txt, Acesso em: Maio de 2020.

Games, E. (2004). “Unrealengine”, https://www.unrealengine.com/en-US/, Acesso em: Maio de 2020.

Games, Y.(2013). “Gamemakerstudio.”,https://www.yoyogames.com/gamemaker, Acesso em: Maio de 2020.

GameSalad (2009). “Game salad.”, https://gamesalad.com/, Acesso em: Maio de 2020.

GDevelop (2013). “The game engine for everyone.”, https://gdevelop-app.com/, Acesso em: Maio de 2020.

Godot (2020). “Godot.”, https://godotengine.org/, Acesso em: Maio de 2020.

Gregory, J. (2015). *Game Engine Architecture*. CRC Press.

Group,K.(2019).“History of opengl.”,https://www.khronos.org/opengl/wiki/History_of_OpenGL#OpenGL_1.0_.281992.29, Acesso em: Maio de 2020.

Halpern, J. (2018). *Developing 2D Games with Unity*. Apress Publishing.

Lowood, H. (2015). “Game Engines and Game History”,*History of Games InternationalConference Proceedings.*

Marks, S., Windsor, J., and W¨unsche, B. (2008). “Evaluation of Game Engines for SimulatedClinical Training”.

Mesa3D (2020). “The mesa 3d graphicslibrary.”, https://www.mesa3d.org/intro.html, Acesso em: Maio de 2020.

Microsoft (2007). “Directx.”, https://microsoft.fandom.com/wiki/DirectX, Acesso em: Maio de 2020.

MUGEN (1999). “M.u.g.e.n.”, https://mugen.fandom.com/wiki/M.U.G.E.N, Acesso em: Maio de 2020.

Oliveira, E. R. (2013). “*O uso de Engines para o desenvolvimentode jogos”*. PhD thesis, Universidade Estadual do Sudoeste ds Bahia, Vitória daConquista.

Onyskiv, T. (2016). “Introducing xenko: a game engine poised for vr.”, https://www.n-ix.com/introducing-xenko-game-engine-vr/, Acesso em: Maio de 2020.

Overmars, M. (2006). *Designing Games with Game Maker*. CRC Press.

Peñalba, O., Cerezo, A., Silos, A., and García-Tejedor, A. (2012). “Q-BAT: A customizable videogame for education.”*In Proceedings of the 4th International Conference onComputer Supported Education.*

Ren’Py (2005). “What is ren’py?”, https://www.renpy.org/, Acesso em: Maio de 2020.

RPGMaker. “Rpg maker.”, https://www.rpgmakerweb.com/, Acesso em: Maio de 2020.

Scirra (2020). “Make games with construct 2.”, https://www.scirra.com/store/construct-2, Acesso em: Maio de 2020.

Stencyl (2020). “Simple pricing.”, http://www.stencyl.com/ pricing/, Acesso em: Maio de 2020.

Stride (2020). “Stride”, https://stride3d.net/, Acesso em: Maio de 2020.

Unity (2020). “Game engines—how do they work?.”, https://unity3d.com/what-is-a-game-engine, Acesso em: Maio de 2020.