**UNIVERSIDADE DE SÃO PAULO**

**Instituto de Ciências Matemáticas e de Computação**

Utilizando Desenvolvimento de Jogos Eletrônicos para o Ensino de Computação

***Gustavo Ferreira Ceccon***

***[Nome do Aluno]***



Utilizando Desenvolvimento de Jogos Eletrônicos para o Ensino de Computação

***Gustavo Ferreira Ceccon***

##### Orientador: Claudio Fabiano Motta Toledo

|  |
| --- |
| Monografia de conclusão de curso apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP - para obtenção do título de Bacharel em Ciências de Computação. |
| Área de Concentração: Ensino, Desenvolvimento de Jogos, Ciências de Computação |

**USP – São Carlos**

Novembro **de 2015**

*A good teacher can inspire hope, ignite the imagination,*

*and instill a love of learning..*

Resumo

O projeto tem como objetivo incentivar professores de diversas disciplinas a usar jogos eletrônicos como forma a engajar o aluno em matérias de computação. Essas diretrizes são voltada para áreas como Introdução à Ciência da Computação I e II (SCC221 e SCC201), Geometria Analítica (SMA300), Algoritmos e Estruturas de Dados (SCC202), Programação Orientada a Objetos (SCC204), Programação Concorrente (SSC143), Computação Gráfica (SSC250) e Multimídia (SCC261). O desenvolvimento de jogos abrange muitas áreas e pode ser explorado para envolver o aluno em uma aplicação real e num contexto interessante. A monografia enumera atividades e projetos de desenvolvimento de jogos para cativar o aluno e dar uma visão de como aplicar o conteúdo aprendido. Um estudo foi feito sobre projetos similares de engajamento de alunos de Ciências de Computação em outras universidades.

Sumário

[Lista de Figuras v](#_Toc499132194)

[Lista de Tabelas vi](#_Toc499132195)

[Lista de Gráficos vii](#_Toc499132196)

[CAPÍTULO 1: INTRODUÇÃO 1](#_Toc499132197)

[1.1. Contextualização e Motivação 1](#_Toc499132198)

[1.2. Objetivos 1](#_Toc499132199)

[1.3. Organização da Monografia 1](#_Toc499132200)

[CAPÍTULO 2: REVISÃO BIBLIOGRÁFICA 2](#_Toc499132201)

[2.1. Considerações Iniciais 2](#_Toc499132202)

[2.2. Conceitos e Técnicas Relevantes 2](#_Toc499132203)

[2.3. Trabalhos Relacionados 2](#_Toc499132204)

[2.4. Considerações Finais 2](#_Toc499132205)

[CAPÍTULO 3: DESENVOLVIMENTO DO TRABALHO 2](#_Toc499132206)

[3.1. Considerações Iniciais 3](#_Toc499132207)

[3.2. Descrição do Problema 3](#_Toc499132208)

[3.3. Descrição das Atividades Realizadas 3](#_Toc499132209)

[3.4. Resultados Obtidos 3](#_Toc499132210)

[3.5. Dificuldades, Limitações e Trabalhos Futuros 4](#_Toc499132211)

[3.6. Considerações Finais 4](#_Toc499132212)

[CAPÍTULO 4: CONCLUSÃO 5](#_Toc499132213)

[4.1. Contribuições 5](#_Toc499132214)

[4.2. Considerações sobre o Curso de Graduação 5](#_Toc499132215)

[REFERÊNCIAS 6](#_Toc499132216)

[APÊNDICE A – Título deste apêndice 7](#_Toc499132217)

[ANEXO A – Título deste anexo 8](#_Toc499132218)

# Lista de Figuras

Nenhuma entrada de índice de ilustrações foi encontrada.

# Lista de Tabelas

Nenhuma entrada de índice de ilustrações foi encontrada.

# Lista de Gráficos

Nenhuma entrada de índice de ilustrações foi encontrada.

# CAPÍTULO 1: INTRODUÇÃO

## Contextualização e Motivação

Ensinar computação não é uma tarefa fácil, de acordo com (Beaubouef, et al., 2005) existem 7 motivos principais para que 30% a 40% dos estudantes de Ciências de Computação larguem o curso no primeiro ou no segundo ano. Outro artigo (Biggers, et al., 2008) faz um estudo sobre o porquê dos estudantes largarem o curso no Geórgia Instituto de Tecnologia em Atlanta.

Um artigo (Bennedsen, et al., 2007) faz uma pesquisa de diversas faculdades para ver a porcentagem de reprovações em disciplinas introdutórias de programação. Apesar de ter um baixa quantidade de resposta dos formulários por parte das universidades (12.7%), ele verifica o alto índice de reprovações dessas matérias (33%). Além disso, o mesmo estudo avalia o que é dado nessas disciplinas, por exemplo paradigma de programação, o que indicou que quase metade (49%) dos institutos usam orientação a objeto como introdutória, apesar que todos os paradigmas tem índices similares.

O mercado de jogos eletrônicos tem uma alta demanda e tem uma alta renda, de aproximadamente US$108,9 bilhões (McDonald, 2017). Enquanto isso, o Brasil está em 13º lugar em mercado de jogos, com 66,3 milhões de jogadores e US$1,3 bilhão gasto em jogo (Newzoo, 2017).

Além disso, os estudantes estão interessados em desenvolvimento de jogos, como indica a Figura 1 feita sobre os inscritos para o processo seletivo do grupo de desenvolvimento de jogos do ICMC-USP, o FoG (Fellowship of the Game). Não apenas alunos de Bacharelado em Ciências de Computação (BCC), mas alunos de Engenharia de Computação (EC), Bacharelado em Sistemas de Informação (BSI) etc. também tem interesse na área.

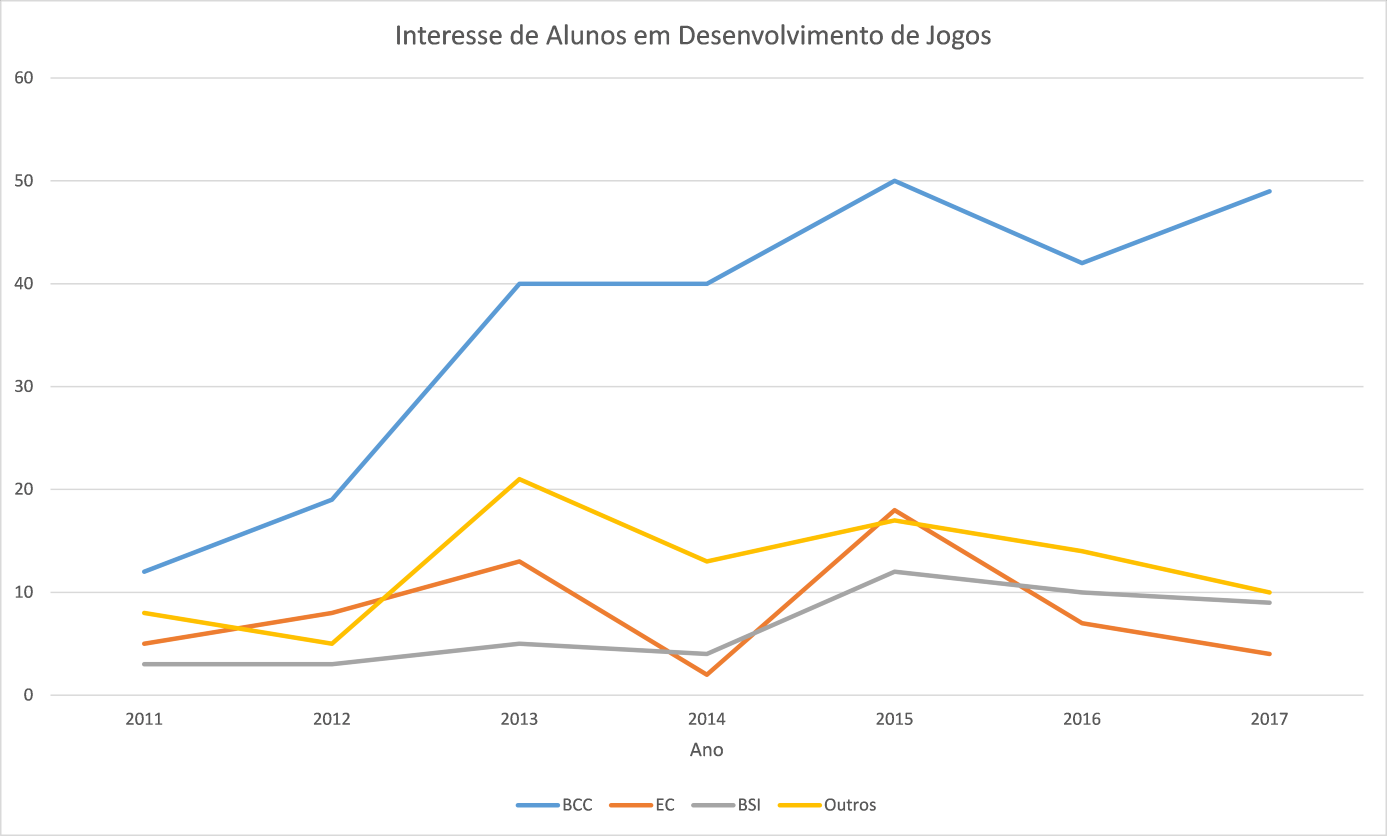


Figura . Interesse de Alunos em Desenvolvimento de Jogos

## 1.2. Objetivos

Alguns dos motivos (Beaubouef, et al., 2005) podem ser lidados usando desenvolvimento de jogos:

* *Poor Math Skills and Problem Solving Abilities*
* *Poorly Designed CS1 Lab Courses*
* *Lack of Practice / Feedback*

A escolha entre programação orientada a objeto no início ou no meio da graduação parece ser um debate sem uma resolução certa. Há pontos positivos e negativos em ambos lados e não será enfoque desse trabalho. Jogos eletrônicos podem ser feitos em ambos paradigmas e, dependendo do gênero, um pode servir melhor que o outro.

Jogos envolvem muita matemática (principalmente em 3D com computação gráfica) e problemas de decisão. Quando apresentado de uma forma concisa, problemas matemáticos são fáceis de implementar e ajudam a visualizar melhor o que é aprendido em matérias como Geometria Analítica. Uma visão tridimensional de vetores, superfícies, volume etc. pode desenvolver a compreensão matemática. Quando não for possível desenvolver algo, seja por limitação de tempo ou dificuldade, uma ferramenta auxiliar pode ser usada.

Um ponto positivo de jogos no aprendizado é que eles são incrementais, ou seja, é possível desenvolver um jogo e conseguir um *feedback* das mudanças e alterações durante o processo. Isso é importante porque é possível dar laboratórios de desenvolvimento onde o professor consegue ir adicionando conteúdo e explicando algoritmos durante a aula. Além disso o aluno pode desenvolver em casa as atividades passadas em aula, incrementando aos poucos e adicionar o próprio conteúdo.

Ter um conteúdo pessoal a mais num projeto, um *flavor,* é importante para um aluno que gosta de jogos. Uma atividade pode ter uma avaliação mínima e adicionais podem fazer parte de pontos extra, estimulando o interesse do aluno.

## 1.3. Organização da Monografia

TODO: Capítulos do livro, exemplos e projetos

# CAPÍTULO 2: REVISÃO BIBLIOGRÁFICA

## 2.1. Considerações Iniciais

Jogos eletrônicos envolvem muito mais que programação apenas, envolve matemática, física, artistas, marketing etc. Reunir todos esses tópicos em apenas um trabalho é impossível, por isso essa monografia cobre apenas algumas partes de todo esse escopo, mesmo no escopo de programação.

Jogos ultimamente não são feitos do zero, eles não almejam apenas uma plataforma e com certeza levam tempo, por exemplo o jogo Antichamber[[1]](#footnote-1), feito por Alexander Bruce, levou aproximadamente 7 anos de desenvolvimento[[2]](#footnote-2). Escolhas de projeto altamente influenciam no tempo de desenvolvimento, custo e riscos.

Para amenizar esses problemas de desenvolvimento de jogos, hoje em dia muitos jogos fazem uso de *game engines* (motores de jogos), que nada mais são que uma ferramenta auxiliar que facilita o desenvolvimento oferecendo suporte para funções e arquitetura básica para se criar qualquer tipo de jogos, em alguns casos, ou tipos específicos de jogos, em outros. Existem grandes *game engines*, como Unity[[3]](#footnote-3), Unreal Engine[[4]](#footnote-4), CryEngine[[5]](#footnote-5) e GameMaker Studio[[6]](#footnote-6).

Existem diversos livros que ensinam como montar um jogo do zero, passando pelo processo de criação de uma *game engine*, como (Ericson, et al., 2005), (Gregory, 2009) e (Eberly, 2005). Um tem mais enfoque em montar um jogo, como o primeiro, do que montar um motor de jogo, como o segundo e terceiro livro.

Porém jogos não necessariamente são voltados para entretenimento, existem os chamados “jogos sérios”, voltados para educação, áreas da saúde etc. Existem diversos estudos (Michael, et al., 2005 ) e (Ritterfeld, et al., 2009) sobre jogos sérios.

Além de livros sobre jogos sérios, existe uma vasta coleção bibliográfica sobre gêneros (tipos) de jogos, como (Apperley, 2006) e sobre como desenvolver jogos (game design), como (Rogers, 2014).

## 2.2. Conceitos e Técnicas Relevantes

## 2.2.1. Jogos Eletrônicos

Jogar não é algo novo e vem das raízes humanas e da sociedade em si, ou seja, jogar é simplesmente instintivo humano (Huizinga, 2012). Computadores são apenas um meio novo para criar uma plataforma onde os seres humanos possam jogar e suprir esse instinto competitivo. Jogos eletrônicos envolvem artifícios artísticos, tantos visuais quanto sonoros, envolvem competividade ou apenas uma mensagem passada pelo desenvolvedor. É difícil explicar o que é bom num jogo, ou o que transforma jogo em um jogo, é subjetivo. Porém uma coisa é certa: jogos tem algo a adicionar para o jogador, pode ser um sentimento, um conhecimento novo ou desenvolver as suas capacidades.

Mas em geral, jogos eletrônicos podem ser descritos da seguinte forma (Gregory, 2009):

*Soft Real-Time Interactive Agent-Based Computer Simulations*

Em outras palavras, jogos são programas de simulação de um mundo virtual, rodando em tempo-real. Não são sistemas críticos, por isso *soft*, são altamente dependendo da interação do usuário, na maior parte do tempo o usuário precisa fazer alguma ação para ter retorno. *Agent-Based* está diretamente relacionado a como jogos são desenvolvidos, pois todos os elementos de um jogo podem ser abstraídos como objetos, nesse caso chamado “agente”. Isso reflete o paradigma geralmente usado para desenvolver jogos: orientação a objeto. Seja Unity usando C# ou Unreal Engine, usando C++, as duas linguagens são modernas e orientada a objetos. Isso não significa que não é possível criar um jogo usando paradigma procedural ou até mesmo paradigma funcional. Um dos jogos antigos mais famosos Doom (1993), tem código aberto[[7]](#footnote-7) e é feito principalmente em C (e um pouco de Assembly). O interesse em paradigma funcional tem aumentado, e existe uma subpágina[[8]](#footnote-8) na própria página oficial da linguagem Haskell para falar só disso.

## 2.2.1. Motores de Jogos

Como já foi dito, motores de jogos aceleram o desenvolvimento de jogos, possibilitaram exportar jogos para múltiplas plataformas com menos esforço. Motores de jogos começaram a tomar lugar no cenário na década de 90, com o jogo Castle Wolfenstein 3D, Quake e Doom, as três sendo da id Software e voltadas para PC (*Personal Computer*). Como os jogos eram similares em arquitetura, eles compartilhavam código, o que é ideia de um motor de jogos.

Esses motores de jogos ainda eram primitivos, eram específico para aquele gênero e eram de proprietárias. Outras empresas como Bethesda, Ubisoft, Blizzard etc. tem suas próprias *engines* de código fechado usadas para propósito dos próprios jogos. Por outro lado existem *engines* de código aberto, como Allegro[[9]](#footnote-9), Cocos2d-x[[10]](#footnote-10) entre outras.

Mas as maiores *game engines* com impacto no cenário atual de jogos são provavelmente a Unity e a Unreal Engine. As licenças comerciais funcionam de formas diferentes:

* Unity é necessário pagar pela inscrição da Unity Pro caso ultrapasse um valor de renda máximo
* Unreal Engine é necessário pagar uma taxa proporcional a sua renda

Muitas das informações sobre história de *game engines* pode ser encontrada no capítulo 1 do livro (Gregory, 2009).

## 2.2.1. Gêneros de Jogos

Gêneros de jogos tem sua própria classificação e delas derivam vários subgêneros e assim por diante. O tipo de jogo está relacionado com as mecânicas principais, a jogabilidade, a arte, câmera e diversos fatores. O primeiro capítulo do livro (Gregory, 2009) dá uma ideia de quais são os principais gêneros e o necessário para desenvolvê-los. Para mais gêneros e subgêneros, outro livro (Crawford, 2011) e (Apperley, 2006) que tem uma visão mais aprofundada e crítica do assunto.

## 2.3 Trabalhos Relacionados

Existem diversas formas de você abordar desenvolvimento de jogos no Bacharelado em Ciência da Computação. A maioria dos artigos usam algum tipo de ferramenta ou motor de jogos para criar um curso voltado ao desenvolvimento de um mais jogos durante o curso.

Existem dois artigos similares (Barnes, et al., 2007) e (Barnes, et al., 2008) que fazem uso de dois jogos educacionais criados pelos orientados para ensinar básico de programação. O projeto, chamado Game2Learn, tem como objetivo criar jogos educacionais e plataformas de aprendizado focados em *game design* e apresentar para alunos de disciplinas do primeiro ano da Universidade do Norte da Califórnia em Charlotte.

A mesma abordagem foi usada no Prog&Play (Muratet, et al., 2010), um jogo sério de ensino baseado em um jogo de código aberto chamado Kernel Panic[[11]](#footnote-11). Em cima desse jogo foi feita uma API onde os alunos podiam usar elementos e memória compartilhada do jogo. O interessante desse artigo é que apesar de ter um feedback bom na primeira iteração da disciplina, a frustração por causa da dificuldade da segunda fase fez com que os alunos sentissem menos motivados. A dificuldade reflete bem o *gameplay*, que por si só reflete na motivação dos alunos.

Na Universidade de Denver, esse artigo (Leutenegger, et al., 2007) descreve o processo de ensino de uma matéria introdutória durante um ano usando jogos. Nesse exemplo, o autor faz uso de Flash e ActionScript, que atualmente estão em desuso, pula para programação orientada a objeto (C++) e introduz conceitos de computação gráfica (OpenGL) aos alunos.

Não só em Denver, mas nesse artigo (Jones, 2000) o autor faz uso de Java 1.2 e não só aplica conceitos básicos de desenvolvimento de jogos e motores, como *game loop, input* e gráficos bidimensionais, como aborda conceitos mais avançados como física, *3D pipeline, binary space partition tree* entre outros tópicos. Esse trabalho é extremamente desatualizado, porém é de quando o ensino usando jogos estava no ínicio. No artigo, o autor cita 4 outros cursos oferecidos em universidades diferentes na mesma época (ver 1. Background). Infelizmente não foi possível recuperar as referências. Da mesma época temos (Becker, 2001) que conta a experiência de atribuição de duas tarefas (um Minesweeper[[12]](#footnote-12) e um Asteroids[[13]](#footnote-13)), com método de avaliação do segundo.

Outra abordagem seria usar uma *game engine* para ensinar básico de programação, como foi feito em (Hernandez, et al., 2010) usando o motor de jogos GameMaker. Esse estudo teve sucesso (no ano de 2009) em comparação ao ano anterior (2008). De acordo com o autor: “This experience allowed the initial presentation of the concepts of visual programming, event-oriented programming and object-oriented programming, without formalizations or specifications.”, e também que “[...]GameMaker engine allows teachers to introduce to freshmen the basic principles of programming logic, without dealing with paradigms’ idiosyncrasies or in programming languages’ details of syntax.”. Isso indica que o formalismo que era lecionado no ano de 2008 usando Java, de alguma forma criava uma barreira no aprendizado do aluno, ao contrário de 2009, onde foi usado interface e elementos simples e interface usando GameMaker. Outro artigo (Chaffin, et al., 2009) usa interface e jogos para ensinar recursão de uma forma interativa, um foco específico que trouxe bons resultados.

Alguns trabalhos, como (Kelleher, et al., 2007), são focados em desenvolvimento de jogos para incentivar mulheres à computação, já que a porcentagem de mulheres no curso é muito baixa, de acordo com o artigo. Porém a porcentagem de jogadoras é quase, se não maior que 50%, como indica (Yee, 2017).

Por último, existem três trabalhos muito importantes sobre o relacionamento de desenvolvimento de jogos e cursos introdutórios de programação. Esse artigo (Bayliss, et al., 2006) sugere como ensinar tópicos de programação, tanto paradigma procedural quanto orientado a objeto. O segundo artigo (Zhang, et al., 2014) resume as metodologias aplicadas e as vantagens, assim como os resultados de usar desenvolvimento de jogos e jogos em geral em uma disciplina introdutória. Por último (Coleman, et al., 2005) cria um currículo voltado a programação, além disso (Pereira L., " Materiais Didáticos para Jogos Eletrônicos" Trabalho de Conclusão de Curso, Universidade de São Paulo, São Carlos, São Paulo, Brasil, 2016) faz uma avaliação de um curso de 33,75 horas, sendo 27 horas de aula e conteúdo e o restante de avaliação e conteúdo extra.

# CAPÍTULO 3: DESENVOLVIMENTO DO TRABALHO

## 3.1. Considerações Iniciais

TODO: como icc1 e disciplinas são dadas, falhas de projetos, apresentação, aulas, avaliação, usar ferramentas, criar algo visual

TODO: criar novos capitulos

## 3.2. Descrição do Problema

TODO: falar da situação atual das aulas

## 3.3. Descrição das Atividades Realizadas

TODO: criação da disciplina, proposta de modelo de graduação

## 3.5. Dificuldades, Limitações e Trabalhos Futuros

TODO: limitação de créditos, criação de novas disciplinas, reorganização do curso, melhor didática dos professores e nova metodologia

# CAPÍTULO 4: CONCLUSÃO

TODO: falar do tamanho da área, oportunidades, inspiração, procura, cursos de jogos e relação com computação

TODO: criar novos capitulos

# Referências

**Apperley, Thomas. 2006.** Genre and game studies: Toward a critical approach to video game genres. *Sage Publications.* Março de 2006.

**Barnes, Tiffany, et al. 2007.** Game2Learn: Building CS1 Learning Games for Retention. *ITiCSE.* Junho de 2007.

**Barnes, Tiffany, et al. 2008.** Game2Learn: Improving the motivation of CS1 students. *GDCSE.* Fevereiro de 2008.

**Bayliss, Jessica e Strout, Sean. 2006.** Games as a “Flavor” of CS1. *SIGCSE.* Março de 2006.

**Beaubouef, Theresa e Mason, John. 2005.** Why the High Attrition Rate for Computer Science Students: Some Thoughts and Observations. *SIGCSE.* Junho de 2005.

**Becker, Karin. 2001.** Teaching with Games: The Minesweeper and Asteroids Experience. *Consortium for Computing in Small Colleges.* Dezembro de 2001.

**Bennedsen, Jens e Caspersen, Michael. 2007.** Failure Rates in Introductory Programming. *SIGCSE.* Junho de 2007.

**Biggers, Maureen, Brauer, Anne e Yilmaz, Tuba. 2008.** Student perceptions of computer science: a retention study comparing graduating seniors with cs leavers. *SIGCSE.* Março de 2008.

**Chaffin, Amanda, et al. 2009.** Experimental Evaluation of Teaching Recursion in a Video Game. *Sandbox.* Agosto de 2009.

**Coleman, Ron, et al. 2005.** Game Design & Programming Concentration Within the Computer Science Curriculum. *SIGCSE.* Fevereiro de 2005.

**Convery, Stephanie. 2017.** https://www.theguardian.com/technology/2017/jan/03/women-make-living-gaming-twitch. *The Guardian.* [Online] 3 de Janeiro de 2017. https://www.theguardian.com/technology/2017/jan/03/women-make-living-gaming-twitch.

**Crawford, Chris. 2011.** *The art of computer game design.* s.l. : McGraw-Hill Osborne Media, 2011.

**Eberly, David H. 2005.** *3D Game Engine Architecture: Engineering Real-Time Applications with Wild Magic.* s.l. : Elsevier, 2005.

**Ericson, Christer e Graham, David. 2005.** *Game Coding Complete.* IV. s.l. : Elsevier, 2005.

**Gregory, Jason. 2009.** *Game Engine Architecture.* s.l. : Taylor and Francis Group, 2009.

**Hernandez, Cristiane Camilo, et al. 2010.** Teaching Programming Principles through a Game Engine. *CLEI Eletronic Journal.* Abril de 2010.

**Huizinga, Johan. 2012.** *Homo ludens.* s.l. : Perspectiva, 2012.

**Jones, Randolph. 2000.** Design and implementation of computer games: A capstone course for undergraduate computer. *SIGCSE.* Janeiro de 2000.

**Kelleher, Caitlin, Pausch, Randy e Kiesler, Sara. 2007.** Storytelling Alice Motivates Middle School Girls to Learn Computer Programming. *CHI.* Abril de 2007.

**Leutenegger, Scott e Edgington, Jeffrey. 2007.** A Games First Approach to Teaching Introductory Programming. *SIGCSE.* Março de 2007.

**McDonald, Emma. 2017.** The Global Games Market Will Reach US$108.9 Billion in 2017 With Mobile Taking 42%. *Newzoo.* [Online] 20 de Abril de 2017. https://newzoo.com/insights/articles/the-global-games-market-will-reach-108-9-billion-in-2017-with-mobile-taking-42/.

**McShaffry, Mike e Graham, David. 2013.** *Game Coding Complete.* s.l. : Course Technology, 2013. Vol. IV.

**Michael, David e Chen, Sandra. 2005 .** *Serious Games: Games That Educate, Train, and Inform.* s.l. : Muska & Lipman, 2005 .

**Muratet, M, et al. 2010.** Experimental Feedback on Prog&Play: A Serious Game. *Computer Graphics Forum.* Agosto de 2010.

**Newzoo. 2017.** The Brazilian Gamer | 2017. *Newzoo.* [Online] 15 de Junho de 2017. https://newzoo.com/insights/infographics/the-brazilian-gamer-2017/.

**Ritterfeld, Ute, Cody, Michael e Vorderer, Peter. 2009.** *Serious Games: Mechanisms and Effects.* s.l. : Routledge, 2009.

**Rogers, Scott. 2014.** *Level up!: the guide to great video game design.* s.l. : Wiley, 2014.

**Takanashi, Fábio. 2009.** Folha de São Paulo. *Folha UOL.* [Online] 06 de Abril de 2009. http://www1.folha.uol.com.br/educacao/2009/04/546576-matematica-e-ciencias-da-computacao-tem-alta-taxa-de-abandono.shtml.

**Walker, Robert. 2011.** Computer Science Vs. Game Development (or Which Degree Should I Get?). *Gamasutra.* [Online] 13 de Fevereiro de 2011. https://www.gamasutra.com/blogs/RobertWalker/20110213/88888/Computer\_Science\_Vs\_Game\_Development\_or\_Which\_Degree\_Should\_I\_Get.php.

**Yee, Nick. 2017.** Beyond 50/50: Breaking Down The Percentage of Female Gamers by Genre. *Quantic Foundry.* [Online] 19 de Janeiro de 2017. https://quanticfoundry.com/2017/01/19/female-gamers-by-genre/.

**Zhang, Fan, Kaufman, David e Fraser, Simon. 2014.** Using Video Games In Computer Science Education. *European Scientific Journal.* Agosto de 2014.

1. <http://www.antichamber-game.com/> [↑](#footnote-ref-1)
2. <https://www.youtube.com/watch?v=wOlcB-JxkFw> [↑](#footnote-ref-2)
3. <https://unity3d.com/> [↑](#footnote-ref-3)
4. <https://www.unrealengine.com/> [↑](#footnote-ref-4)
5. <https://www.cryengine.com/> [↑](#footnote-ref-5)
6. <https://www.yoyogames.com/gamemaker> [↑](#footnote-ref-6)
7. <https://github.com/id-Software/DOOM> [↑](#footnote-ref-7)
8. <https://wiki.haskell.org/Game_Development> [↑](#footnote-ref-8)
9. <http://liballeg.org/> [↑](#footnote-ref-9)
10. <https://github.com/cocos2d/cocos2d-x> [↑](#footnote-ref-10)
11. <https://springrts.com/wiki/Kernel_Panic> [↑](#footnote-ref-11)
12. <https://en.wikipedia.org/wiki/Minesweeper_(video_game)> [↑](#footnote-ref-12)
13. <https://en.wikipedia.org/wiki/Asteroids_(video_game)> [↑](#footnote-ref-13)