**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 Gráficos v](#_Toc499662867)

[CAPÍTULO 1: INTRODUÇÃO 1](#_Toc499662868)

[1.1 Contextualização e Motivação 1](#_Toc499662869)

[1.2 Objetivos 2](#_Toc499662870)

[1.3 Organização da Monografia 3](#_Toc499662871)

[CAPÍTULO 2: REVISÃO BIBLIOGRÁFICA 4](#_Toc499662872)

[2.1 Considerações Iniciais 4](#_Toc499662873)

[2.2 Conceitos e Técnicas Relevantes 5](#_Toc499662874)

[2.2.1 Jogos Eletrônicos 5](#_Toc499662875)

[2.2.2 Motores de Jogos 6](#_Toc499662876)

[2.2.3 Gêneros de Jogos 6](#_Toc499662877)

[2.3 Trabalhos Relacionados 7](#_Toc499662878)

[CAPÍTULO 3: DESENVOLVIMENTO DO TRABALHO 10](#_Toc499662879)

[3.1 Considerações Iniciais 10](#_Toc499662880)

[3.2 Descrição do Problema 10](#_Toc499662881)

[3.2.1 Poor Math Skills and Problem Solving Abilities 10](#_Toc499662882)

[3.2.2 Poorly Designed CS1 Lab Courses 11](#_Toc499662883)

[3.2.3 Lack of Practice / Feedback 11](#_Toc499662884)

[3.2.4 Graduate Student Teachers 11](#_Toc499662885)

[3.3 Descrição das Atividades Realizadas 12](#_Toc499662886)

[3.3.1 Disciplina de Introdução ao Desenvolvimento de Jogos Eletrônicos 12](#_Toc499662887)

[3.3.2 Introdução a Ciências de Computação I 14](#_Toc499662888)

[3.3.3 Programação Orientada a Objeto 16](#_Toc499662889)

[3.3.4 Computação Gráfica 16](#_Toc499662890)

[3.3.5 Multimídia 17](#_Toc499662891)

[3.3.6 Programação Concorrente 17](#_Toc499662892)

[3.3.7 Disciplinas Matemáticas 18](#_Toc499662893)

[3.3.8 Algoritmos e Estruturas de Dados e Introdução a Ciências de Computação II 18](#_Toc499662894)

[3.4 Dificuldades, Limitações e Trabalhos Futuros 18](#_Toc499662895)

[CAPÍTULO 4: CONCLUSÃO 19](#_Toc499662896)

[Referências 20](#_Toc499662897)

# 

# Lista de Gráficos

[Gráfico 1. Interesse de Alunos em Desenvolvimento de Jogos 2](#_Toc499482488)

[Gráfico 2. Perfil Final dos Alunos 13](#_Toc499482489)

[Gráfico 3. Frequência de Notas 13](#_Toc499482490)

# CAPÍTULO 1: INTRODUÇÃO

## 1.1 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.

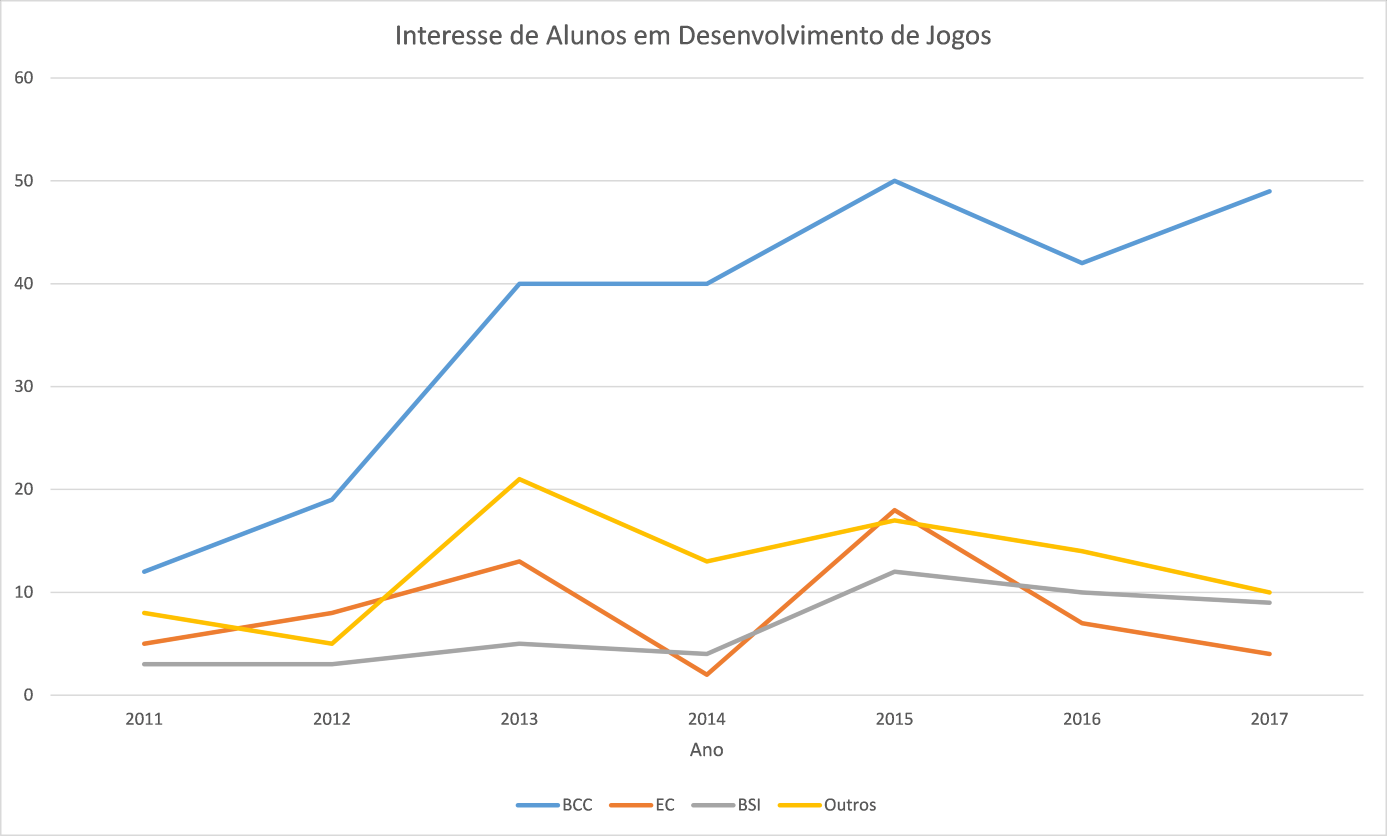


Gráfico 1. Interesse de Alunos em Desenvolvimento de Jogos

## 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

A monografia é voltada a desenvolvimento de jogos em matérias lecionadas no ICMC-USP. Além disso é abordado como foi a experiência de dar a primeira disciplina de Introdução a Desenvolvimento de Jogos Eletrônicos. São feitas revisões de bibliografias de artigos passados do mesmo assunto: desenvolvimento de jogos e educação.

# 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, 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.

### 2.2.2 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).

### 2.2.3 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 o trabalho de conclusão de curso (Pereira, 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

Existem alguns pontos que levam alunos da graduação a desistirem do curso, como apontado no capítulo 1.2 e no artigo (Beaubouef, et al., 2005). Esse trabalho propõe algumas abordagens para resolver alguns desses problemas frequentes. Existem três tipos de abordagem sugeridas:

* Usar ferramentas de visualização tridimensional e desenvolvimento de aplicações simples para maior entendimento de matemática e alguns tópicos de computação
* Usar desenvolvimento de jogos simples sem interface gráfica para introdução a programação imperativa
* Usar desenvolvimento de jogos de tempo real com interface gráfica bidimensional para tópicos mais avançados de programação imperativa
* Usar desenvolvimento de motor de jogos para programação orientada a objeto e tópicos avançados de computação

## 3.2 Descrição do Problema

### 3.2.1 Poor Math Skills and Problem Solving Abilities

Disciplinas de matemáticas são totalmente teóricas e fazem muito uso de exercícios feitos em lousa, além de provas matemáticas que muitas vezes não são triviais. Um dos motivos da desistências é a falta de uma base sólida em matemática (Beaubouef, et al., 2005), e exemplos em lousa geralmente não são intuitivos e não motivam os alunos. Onde é possível visualizar a matéria em alguma ferramenta ou aplicação, não é uma aproximação usada geralmente pelos professores do ICMC.[[14]](#footnote-14)

### 3.2.2 Poorly Designed CS1 Lab Courses

Atualmente no ICMC-USP é oferecida ICC (Introdução à Ciência de Computação) I para os calouros, onde é abordada a linguagem C e o paradigma imperativo. Também são passados conceitos como *hardware* e *software* entre outros. As aulas são teóricas e focadas em sintaxe e tecnologia, tendo em vista que aplicação dos conceitos é feita em Laboratório de ICC I. Atualmente a graduação é dividida nessas duas matérias introdutórias, sendo que a teórica tem o dobro de créditos que a de laboratório (4 e 2 créditos respectivamente).

A quantidade de tempo gasto em laboratório é muito pouca em relação à teoria passada, pode não ser possível passar o suficiente de algum tópico de linguagem imperativa. Além disso, as aulas precisam ter alguma ligação uma com as outras, ou seja, o aluno deve ser capaz de criar qualquer software com tudo o que foi aprendido, além de compreender o que foi passado e porque ou quando usar aquela ferramenta.

### 3.2.3 Lack of Practice / Feedback

Um dos problemas mais graves que acontece em qualquer um dos tópicos citados em (Beaubouef, et al., 2005). É necessário passar uma quantidade mínima de prática para os alunos, para o entendimento do assunto. Uma das grandes vantagens do curso de computação é aplicação de qualquer tópico abordado. Além disso é preciso ter um retorno dos erros e dos acertos do estudante em suas tarefas. Ou seja, correções automáticas e atribuição de nota apenas não acrescenta informação suficiente para os alunos.

### 3.2.4 Graduate Student Teachers

Não é criada uma proposta para resolver esse problema, a falta de treinamento e de didática por parte dos professores com certeza é um fator que influencia na desistência. Não é fácil treinar uma pessoa para instruir uma classe de alunos, mais difícil ainda é dar toda atenção necessária para todos os alunos, dado a quantidade de alunos por classe.

Porém nesse trabalho, como são descritas formas de ensino que nem sempre são usadas, é preciso que o professor tenha um contato maior com jogos para que seja possível ensinar através dessas atividades. Acredito que esse não será um problema para as próximas gerações, dado que jogos tem sido parte da infância da atual geração.[[15]](#footnote-15)

## 3.3 Descrição das Atividades Realizadas

Foram feitas análises sobre como melhorar o aprendizado e motivação dos alunos de Bacharelado de Ciências de Computação usando desenvolvimento de jogos. Foram pensadas propostas para o melhoramento do curso e criação de novas disciplinas.

### 3.3.1 Disciplina de Introdução ao Desenvolvimento de Jogos Eletrônicos

No segundo semestre de 2016 foi dado uma disciplina de Introdução ao Desenvolvimento de Jogos Eletrônicos, com o rótulo de SSC0147 - Tópicos Especiais em Sistemas de Computação I, com a turma de código 2016201. O curso abordou diversos tópicos, além de atividades práticas e um projeto final de jogo. A experiência foi um sucesso e é possível ver o *feedback* por parte dos alunos no trabalho de conclusão de curso (Pereira, 2016)

No Gráfico 2. É possível ver o perfil final é de alunos aprovados, onde os alunos reprovados por nota e falta são alunos que desistiram a matéria e deixaram de comparecer. São 5 alunos, dois alunos desistentes (não compareceram em nenhuma aula) e 3 alunos que abandonaram a matéria (com 31,25%, 37,5% e 62,5% de presença). Os mesmos alunos tiraram entre 0 e 5 na média final, como é possível ver no Gráfico 3. Além disso, o gráfico mostra a grande quantidade de notas altas, entre 7 e 10, dos alunos aprovados e reprovado por nota. Foi um aluno reprovado por nota, devido a falha de entrega de alguns trabalhos, o que acarretou a sua reprovação, apesar da nota final alta.

A disciplina está sendo ministrada novamente no segundo semestre de 2017 (atual semestre). Algumas alterações foram feitas em relação à forma de avaliação e ao material e ordem do material usados. Foi feita uma intercalação entre aulas teóricas e práticas, a fim de dar uma visão prática melhor da teoria. A avaliação agora é voltada ao projeto final e as documentações envolvidas, o que encaixa melhor na proposta da disciplina.

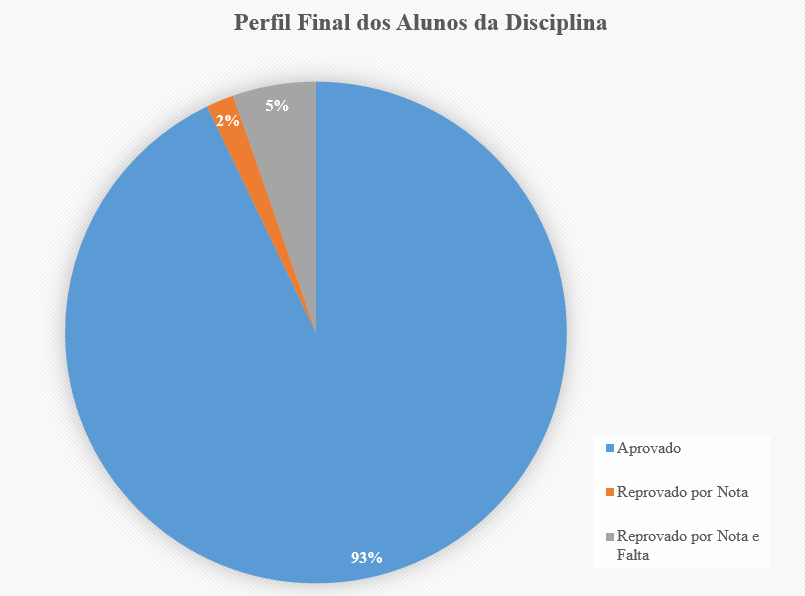


Gráfico 2. Perfil Final dos Alunos

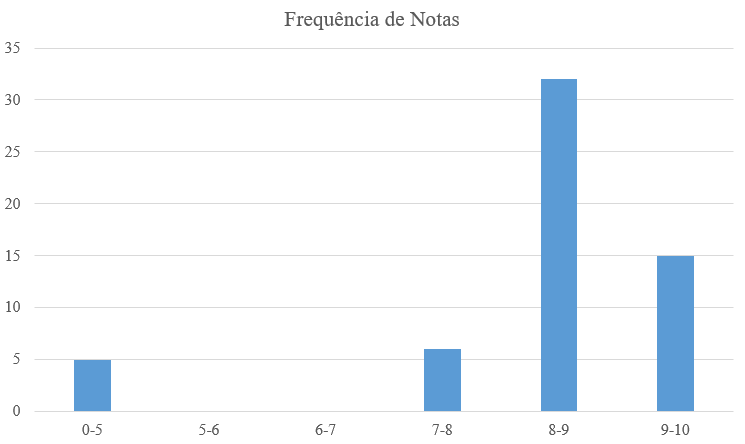


Gráfico 3. Frequência de Notas

### 3.3.2 Introdução a Ciências de Computação I

A ideia dos projetos dessa sessão são bem similares aos artigos que usam desenvolvimento de jogos simples para explicar conceitos. Destaques são esse (Leutenegger, et al., 2007) e esses projetos (Becker, 2001), que expõe a forma de avaliação de um dos jogos.

Existem diversos gêneros de jogos[[16]](#footnote-16), cada um tem sua caraterística e mecânicas (jogabilidade). O livro (Gregory, 2009) cita alguns gêneros e que módulos são necessários para implementar todas as mecânicas básicas. Cada gênero tem sua peculiaridade e diferentes ferramentas são necessárias para fazer cada um.

No tópico 2.2.1 é explicado o que é um jogo na visão de um programador, nesse caso o Jason Gregory. Na descrição, existe o termo *Real-Time* (tempo real), que indica que um jogo tem que ler a entrada, processar e responder o mais rápido possível para o jogador. Porém existem jogos que são mais reativos do que em tempo real, por exemplo, um TBS (*Turn Based Strategy*).

Nesse gênero do jogo, o jogador tem que preparar a sua estratégia durante o próprio turno (limitado ou não) e ao finalizar suas ações, é a vez do turno do inimigo. Esse tipo de mecânica pode ser misturada com diversas outras, como por exemplo de batalha. Alguns exemplos de TBS são: Rome: Total War[[17]](#footnote-17) e série Civilization[[18]](#footnote-18).

Outros jogos que são parcialmente reativos são alguns jogos de aventura e RPG (*Role Playing Game*). Algumas mecânicas de batalha como a de Pokémon[[19]](#footnote-19), onde é por turno e RPG de texto online[[20]](#footnote-20).

A ideia de projeto para ensinar programação imperativa é ir fazendo aos poucos um jogo baseado em batalha em turnos. Ou seja, o jogador tem que fazer escolhas como “Atacar”, “Defender”, “Usar Item” etc. No outro lado, seria feita inicialmente uma máquina “burra” (escolhas randômicas) para fazer sua ação no seu turno.

Algumas formas de introduzir elementos de programação usando esse jogo seriam:

* Entrada e Saída: entrada do nome do jogador, imprimir as possibilidades de ação, estado do jogador etc.
* Variáveis: nome, vida do jogador, mana[[21]](#footnote-21), força, destreza, inteligência[[22]](#footnote-22)
  + Ambos jogador e inimigo
* Aritmética: cálculo de ataque baseado nos atributos do jogador
* Condicionais: elementos de sorte, como acerto de ataque, dano, crítico
* Laços: múltiplos turnos
* Vetores: nome do jogador, inventário
* Funções: modularização do código
* Estruturas: jogador, inimigo, conceito de objeto de jogo

Existem muitas formas de explorar esse jogo, além de ser possível implementar as mecânicas básicas em laboratório. Uma das vantagens desse jogo é que ele é facilmente expansível: podem ser criados múltiplos jogadores, múltiplos ações e efeitos, múltiplos inimigos, jogador contra jogador etc. Podem ser listadas vários tópicos que seriam extra na forma de avaliação.

Acredito que isso pode ser apenas uma parte introdutória e que não levaria muitas aulas ou atividades para alcançar o objetivo de familiarizar o aluno com programação imperativa. O próximo passo seria montar um projeto final de jogo que seja tempo real, usando alguma biblioteca gráfica. Algumas vantagens:

* Elevado uso de ponteiros
* Familiarizar o aluno com bibliotecas externas
* Retorno visual além de texto
* Adição de conteúdo pessoal do aluno

O mesmo jogo de turno pode ser usado para esse projeto, a diferença são os aspectos visuais, além de que o aluno provavelmente precisará modularizar o código para reaproveita-lo. Animações e efeitos visuais, além de uma interface com o usuário podem ser requisitos desse trabalho. Algumas bibliotecas encaixam bem no projeto, como SDL[[23]](#footnote-23) (por causa de tutoriais[[24]](#footnote-24)) e Allegro[[25]](#footnote-25).

### 3.3.3 Programação Orientada a Objeto

O objetivo final do projeto orientado a objeto é criar uma *game engine* primitiva, usando alguma biblioteca voltada a jogo. Nesse caso, para programação Java, a possível melhor opção seria a Lightweight Java Game Library[[26]](#footnote-26). A intenção é realmente aumentar a dificuldade dos projetos. Caso seja possível, a série de tutoriais de OpenGL[[27]](#footnote-27) abrange muita programação de *rendering* tridimensional em C++.

Dois livros seriam essenciais para o professor e para o aluno: (Gregory, 2009) para arquitetura de jogos baixo nível, nos capítulos 5 e 7, essenciais para jogos. O capítulo 1 são bastante interessantes para iniciantes e a sessão 1.6 é essencial para *software engineering*. O livro (McShaffry, et al., 2013) é perfeito para iniciantes, pois explica passo a passo com bastante código. O único problema é que é feito em C++ com DirectX, como nas universidades o foco geralmente é sistemas Unix, fica complicado usá-lo.

Os principais tópicos que precisam ser abordados são:

* Objetos de Jogos e abstração de um elemento do jogo
* Herança e polimorfismo para criação de diferentes elementos e comportamentos
* Encapsulamento e módulos gráficos, entrada etc.
* Módulo de *render* tridimensional simples

### 3.3.4 Computação Gráfica

O projeto de computação gráfica seria uma plataforma de *rendering* completa, com todos tópicos importantes inclusos e de preferência usando a série de tutoriais de OpenGL27. As principais funções seriam de câmera e de *rendering* de malhas usando *shaders* usando uma mecânica para alterar e explorar as possibilidades com eles. É possível explorar:

* Câmera
* *Render* tridimensional
* Métodos de iluminação e *shaders*
* Mecânica de carregamento de *shaders*
* Mecânicas avançadas
  + *Normal Mapping*
  + *Post Processing*
  + *Anti Aliasing*
* Sombra

### 3.3.5 Multimídia

Um tópico de multimedia é compressão, que pode ser implementado voltado a jogos. No desenvolvimento de jogos, lidar com recursos é fundamental, porque existem muitos arquivos de imagem, modelos e áudio. Geralmente é feita alguma compressão ou compactação e pode ser um projeto específico da disciplina. O capítulo 8 do livro (McShaffry, et al., 2013) aborda melhor compressão e recursos, além de falar sobre compressão ZIP.

### 3.3.6 Programação Concorrente

Jogos só seriam *softwares* de tempo real se fosse possível processar em paralelo. Não é uma tarefa simples, o capítulo 20 do livro (McShaffry, et al., 2013) e parte do capítulo 7 do livro (Ericson, 2005) falam sobre o assunto. O foco é *game loop[[28]](#footnote-28)*, que é complicado, porém é muito mais simples paralelizar o módulo de física de um jogo. O livro (Ericson, 2005) aborda somente colisão física, que é altamente paralelizável

A ideia é implementar algum algoritmo de checagem de colisão usando programação paralela, é necessário criar formas de colisão, como caixa ou esfera, e implementar um algoritmo capaz de processar e criar uma lista de objetos que colidem. O algoritmo força bruta seria de complexidade n², portanto a otimização é essencial.

### 3.3.7 Disciplinas Matemáticas

Disciplinas matemáticas como Geometria Analítica são bastante visuais, e isso pode ser explorado através de alguma ferramenta tridimensional. A *game engine* Unity possui uma gama de funções de visualização, incluindo *gizmos[[29]](#footnote-29)* onde é possível desenhar vetores, formas simples etc. O restante pode ser feito através de malhas criadas em tempo real, para isso é preciso ter noção das fórmulas matemáticas e de visualização tridimensional, porém pode ser interessante para o desenvolvimento das habilidades matemáticas do aluno.

### 3.3.8 Algoritmos e Estruturas de Dados e Introdução a Ciências de Computação II

Novamente, Unity pode ser explorada nessas matérias. Algoritmos e Estruturas de Dados podem ser implementadas e depois usando as formas básicas da Unity, pode visualizar o resultado da interação com as estruturas. Similarmente para ICC II, onde algoritmos de ordenação e recursão são bastante explorados. Um artigo (Chaffin, et al., 2009) aplica visualização para recursão e a ideia é similar.

## 3.4 Dificuldades, Limitações e Trabalhos Futuros

A maior dificuldade e limitação da disciplina de Introdução a Desenvolvimento de Jogos Digitais é a falta de uma matéria só para laboratório. A pequena quantidade de créditos e horas aula não é suficiente para cobrir a programação baixo nível. Além disso falta tempo suficiente para aprofundar em outros tópicos como Game Design e Produção, que no modelo atual só tem uma aula para cada tópico.

# CAPÍTULO 4: CONCLUSÃO

Desenvolvimento de jogos é um tópico que foi bastante explorado mas ainda é pouco aplicado aqui no Brasil. Faculdades de Ciências da Computação não exploram o tema, que pode ser muito interessante para o aluno, ajudando a reduzir a quantidade de desistência do curso. No ICMC-USP ainda está sendo criada uma disciplina apenas sobre jogos eletrônicos que encaixe no curso de Bacharelado de Ciências de Computação.

# 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.

**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. 2005.** *Real-Time Collision Detection.* 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/.

**Pereira, Leonardo. 2016.** Materiais Didáticos para Jogos Eletrônicos. *Trabalho de Conclusão de Curso.* São Carlos, São Paulo, Brasil : Universidade de São Paulo, 2016.

**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.

**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)
14. Esses dados são meramente opinião e experiência do autor, sem base científica [↑](#footnote-ref-14)
15. Esses dados são meramente opinião e experiência do autor, sem base científica [↑](#footnote-ref-15)
16. <https://en.wikipedia.org/wiki/List_of_video_game_genres> [↑](#footnote-ref-16)
17. <https://en.wikipedia.org/wiki/Rome:_Total_War> [↑](#footnote-ref-17)
18. <https://en.wikipedia.org/wiki/Civilization_(series)> [↑](#footnote-ref-18)
19. <https://en.wikipedia.org/wiki/Pok%C3%A9mon_(video_game_series)> [↑](#footnote-ref-19)
20. <https://en.wikipedia.org/wiki/Online_text-based_role-playing_game> [↑](#footnote-ref-20)
21. Mana é um recurso usado pelo jogador para invocar feitiços [↑](#footnote-ref-21)
22. Força, destreza e inteligência são atributos geralmente usados em RPG para modificar os valores de dano de ataque, defesa e quantidade de mana [↑](#footnote-ref-22)
23. <https://www.libsdl.org/> [↑](#footnote-ref-23)
24. <http://lazyfoo.net/tutorials/SDL/> [↑](#footnote-ref-24)
25. <http://liballeg.org/> [↑](#footnote-ref-25)
26. <https://www.lwjgl.org/> [↑](#footnote-ref-26)
27. <https://learnopengl.com/> [↑](#footnote-ref-27)
28. Laço principal que controla todo o processo de leitura, processamento e *render* [↑](#footnote-ref-28)
29. <https://docs.unity3d.com/ScriptReference/Gizmos.html> [↑](#footnote-ref-29)