Departamento de Ciência da Computação
Instituto de Matemática e Estatística
Universidade de São Paulo

Desenvolvimento de Aplicação Web Voltada ao Ensino de Programação para o Público Infanto-Juvenil

Orientador
José Coelho de Pina

Novembro, 2015

Daniel Paulino Alves
Felipe Yamaguti
Rafael Batista Carmo

Resumo

O raciocínio computacional vem sendo cada vez mais divulgado e, ao longo do tempo, valorizado na sociedade moderna. Após a recente onda de inclusão digital, o uso da tecnologia tornou-se imprescindível para o profissional de hoje. Porém, ao que tudo indica, apenas saber utilizá-la não será mais suficiente neste cenário que está sendo moldado. O entendimento de seu funcionamento e a forma de raciocínio em si mostram-se cada vez mais necessários e úteis, não só para trabalhadores de áreas relacionadas com a tecnologia, mas para qualquer pessoa que queira destacar-se em seu campo de atuação.

O Brasil ainda se mostra atrás no cenário mundial quanto ao desenvolvimento desta nova forma de pensamento. O ensino de tecnologia em nossas escolas é superficial, visto que compreende apenas os primeiros contatos com o computador e os aplicativos mais usados, configurando, desta forma, apenas uma alfabetização digital. Além disso, poucas pessoas têm acesso aos meios de aprendizado necessários para o desenvolvimento dessa faculdade mental. Sob essas condições, a internet torna-se uma ferramenta de grande destaque para que essa situação possa ser alterada, ou seja, para que o raciocínio computacional seja apresentado de forma natural e introduzido cada vez mais no dia a dia do cidadão comum.

Baseado em pesquisas recentes e aplicações similares de sucesso comprovado, desenvolvemos uma plataforma online, livre, de ensino gratuito para jovens de doze a quinze anos que estejam interessados em descobrir essa nova forma de raciocínio e saber mais sobre como as atuais tecnologias funcionam. Utilizamos a *engine* para jogos *Unity 3D* para criar um sistema de aprendizado em módulos que possibilita ao usuário aprender um pouco do raciocínio computacional, tal como o funcionamento de comandos, laços e condições.

Este estudo descreve o processo de pesquisa, criação, design, teste e análise desta plataforma.

Palavras-chave: Raciocínio computacional, jovens, computação, ensino.

Abstract

Nowadays, there is a noticeable tendency for the promotion and appreciation of computational thinking in modern society. Following the recent trend in digital inclusion, the use of technology has become indispensable for the professional of today. Yet, to all appearances, just knowing how to use it will be no longer sufficient in this environment which is being sculpted. The understanding of the workings of technology and the way of thinking itself are increasingly necessary, not only to IT workers and employees in related domains, but to anyone who wants to be recognized in his own field.

Brazil is current falling behind in the world scene when it comes to the development of this new way of thinking. Technological education in our schools is superficial, insofar as it only comprehends basic computer notions and the most used applications, which comes out as just digital literacy. Besides, few people today have access to ways of learning what is necessary to the development of this mental faculty. Under those conditions, the Internet has become one tool of great importance to change this situation, that is to say, so that computational thinking can be introduced naturally and inserted more and more in the daily life of the common citizen.

Based on recent researches and similar applications of proven success, we have developed a free educational online platform for children between twelve and fifteen years old who are interested in deciphering this new way of thinking and in knowing a little more about how current technologies work. We have used the game engine *Unity 3D* to create a learning system in modules that enables the user to learn a bit more of computational thinking, and also commands, loops, and conditionals.

This paper describes the process of research, creation, design, trial and analysis of this platform.

Keywords: Computational thinking, teenagers, computer science, teaching.

Sumário

Parte Objetiva

Motivação

O século XXI foi marcado em seus primeiros anos pelo enraizamento da computação no dia a dia de quase todo cidadão. Seja morador da cidade ou do campo, jovem ou não, praticamente todo ser humano vivendo em sociedade tem presenciado a substituição cada vez mais corriqueira de uma ferramenta analógica por um computador, em maior ou menor escala.

A tecnologia não só mudou o jeito de se fazer as coisas, mas também criou novas oportunidades de resolução para problemas que antes eram insolúveis ou que nem sequer existiam. Isso ocorreu pois o modo de pensar da sociedade como um todo foi modificado ao longo do tempo.

Seguindo este rumo evolutivo, tornou-se interessante e, em alguns casos, desejável ou mesmo obrigatório a imersão do cidadão no mundo digital. Para muitos esta transição foi gradativa e natural ao longo dos últimos anos. Viver sem ferramentas que hoje estão sempre presentes e disponíveis para o uso tornou-se impensável.

Sendo assim, outra necessidade importante que surgiu foi o entendimento, pelo menos parcial, do funcionamento dessas ferramentas. Não só pelo simples fato de elas permearem a vida cotidiana, mas também por introduzirem uma nova forma de pensar.

O pensamento computacional, por sua vez, independente da tecnologia, pode ajudar na resolução de problemas cotidianos, tanto quanto a matemática o faz. Por isso, seria importante introduzi-lo já na infância e torná-lo algo natural, para que cada vez mais se aprimore esta habilidade de enxergar as coisas sob esta ótica em particular.

É com essa conjuntura em mente que nos propusemos a criar um sistema que possibilite o aprendizado a distância do raciocínio computacional. Esperamos, assim, trazer à tona o interesse de mais pessoas pela área de Ciência da Computação através de nossa plataforma, além de estimular o desenvolvimento dessa nova forma de pensar.

Introdução

Por que ensinar crianças a programar?

Com o advento da internet e o rápido desenvolvimento da tecnologia em geral, o mundo computacional vem adentrando a passos largos o dia a dia do cidadão comum. Ademais, com o discurso do presidente dos Estados Unidos, Barack Obama[1], que incentiva o aprendizado da ciência da computação desde cedo, e a decisão de outros países como Austrália, Reino Unido e Canadá em adicionar computação ao seu ensino para crianças a partir de cinco anos de idade, através da reformulação do currículo K-12[2][3], criou-se uma discussão sobre o porquê de se ensinar jovens a programar.

Mais do que isso, esta discussão estende-se também a adultos. Seria importante para uma pessoa aprender os conceitos básicos de programação, mesmo que ela nunca se torne programadora ou desenvolvedora de softwares? Mitchel Resnick, professor do MIT e principal responsável pelo projeto Scratch, responde esta pergunta da seguinte forma em seu artigo "Live to code, code to live"[4]:

> "É importante para todas as crianças aprender a escrever? Afinal, muito poucas se tornarão jornalistas, romancistas ou escritoras. Então, por que todos devem aprender a escrever?
>
> É claro que tal pergunta parece boba. As pessoas usam a escrita em diversas áreas de suas vidas: para enviar mensagens de aniversário para amigos, para fazer listas de compras, para anotar sentimentos pessoais em diários. O ato de escrever também leva as pessoas a pensar de maneiras diferentes. Quando alguém escreve, tal pessoa aprende a organizar, refinar e refletir sobre suas ideias. Claramente existem razões muito fortes para todos aprenderem a ler.
>
> Eu vejo a programação como uma extensão da escrita. A habilidade de programar permite a você 'escrever' novos tipos de coisas – histórias interativas, jogos, animações, e simulações. E, como na escrita tradicional, existem razões muito fortes para se aprender a programar."

O artigo do U.S. News, "STEM Education - It's Elementary"[5], discute a importância não só da computação, mas também da ciência, tecnologia, engenharia e matemática (science, technology, engineering and mathematics - STEM). E afirma que essas áreas devem ser o maior foco de investimento nos próximos anos devido à configuração que a economia atual vem tomando.

> "Se os Estados Unidos quiser manter seu poder econômico, então precisaremos de uma educação baseada na força de trabalho STEM, que possa suprir as demandas em uma economia com uma necessidade cada vez maior e mais complexa por tecnologia."

O autor do artigo, Anthony Murphy, professor da Universidade St. Catherine em St. Paul, afirma ainda que o ensino *STEM* deve iniciar-se cedo com as crianças, certamente na escola elementar (equivalente ao nosso ensino fundamental), e talvez até ainda mais cedo. Entretanto, tal ensino deve ser tratado com muito cuidado, e não de forma displicente ou sem a devida atenção para que ocorra um progresso, e não um retrocesso, nesta área. A falta de planejamento pode acarretar em uma educação defasada para toda uma geração, como o que temos vivido recentemente[6].

Um grande erro cometido ao longo dos últimos vinte anos, como cita John Naughton em seu artigo "Why all our kids should be taught how to code"[6] é, ao invés de se ensinar às crianças as mais novas tecnologias presentes no mercado, vem se ensinando-lhes a utilizar softwares obsoletos. Hong Kong, por exemplo, é uma das cidades mais conectadas do mundo; de acordo com o dados governamentais, 85% das casas possuem banda larga e cada pessoa tem em média dois aparelhos celulares. Entretanto, o sistema educacional não vem acompanhando a velocidade das mudanças do mundo digital; embora aulas de alfabetização digital sejam lecionadas, o ensino do raciocínio computacional recebe muito menos atenção nas escolas[7].

Isso é um caso do que o filósofo Gilbert Ryle chama de "erro de categoria": quando coisas de um tipo são apresentadas como se fossem de outro. Ainda segundo ele, nós caímos no conceito errôneo de que aprender sobre computadores é como aprender a dirigir um carro e, assim como não é necessário aprender sobre as engrenagens internas do automóvel, não seria preciso aprender detalhes do funcionamento do computador. Este é o maior problema; computadores não são automóveis, eles estão muito mais presentes em nossas vidas. Esta visão de Ryle também se aplica ao panorama brasileiro, haja vista a demanda crescente por celulares, tablets e outros dispositivos inteligentes. O país conta com mais de 300 milhões de equipamentos conectados à Internet e, com os avanços no campo da Internet das Coisas, este número tende a aumentar rapidamente.[8][9][10]

O mesmo artigo, publicado alguns anos atrás, ainda explicita tal problema relatando que carros não controlam o mundo, não monitoram nossas comunicações, não gerenciam nossas contas bancárias e muito menos selecionam nossos políticos em alguns países, ao passo que computadores são

responsáveis por isto e muito mais. Tratam-se de ferramentas completamente diferentes.

Deve-se focar, portanto, esta nova forma de aprendizado em duas principais áreas:

1. Primeiramente, nos vários conceitos-chave responsáveis pelo ambiente em que as crianças estão inseridas hoje. Como relata detalhadamente o artigo “Kindergarten is the model for lifelong learning”[11], o jardim de infância vem se tornando cada vez mais parecido com o resto da escola, quando o contrário deveria ser adotado. É lá que - através de contos infantis, pinturas e desenhos - desenvolvemos o pensamento criativo e o trabalho em equipe, tão importantes para a vida profissional em tempos modernos. Citando Mitchel Resnick mais uma vez:

> “Blocos de madeira e pinturas a dedo são ótimos para aprender-se os conceitos do jardim de infância (como números, formas, tamanhos e cores). Mas quando as crianças ficam um pouco mais velhas, elas querem e devem trabalhar em projetos mais avançados e aprender conceitos mais avançados. Para isso, elas precisam de novos tipos de ferramentas, mídias e materiais.
>
> É aí que eu acredito que a tecnologia digital pode ter o seu papel mais importante. Se desenvolvidas e usadas propriamente, as novas tecnologias podem estender a abordagem do jardim de infância, possibilitando aos 'alunos' de todas as idades continuar aprendendo como no jardim da infância e, através desse processo, continuar crescendo como pensadores criativos. ”

   Através do ensino prematuro da computação poderia-se, então, atingir bons resultados futuros no desenvolvimento da nova geração de profissionais, que estariam mais capacitados e mais adaptados a essa nova realidade.

2. E em segundo lugar, no ensino do pensamento computacional em si, que compreende o entendimento das diferenças entre a forma de agir humana e artificial, a aprendizagem da lógica recursiva, a conscientização quanto à necessidade de prevenção, detecção e proteção contra riscos e falhas, além da busca por resolução de problemas complexos. O artigo "Scratch: uma opção válida para desenvolver o pensamento computacional e a competência de resolução de problemas"[12], desenvolvido por alunos da Universidade do Minho em Portugal, expõe de maneira clara o que é e qual a importância do aprendizado dessa nova forma de raciocínio:

> “O pensamento computacional é a capacidade de desencadear o processo de formulação de problemas do mundo real e de solucioná-los (Cuny, Snyder, & Wing, 2010; Wing, 2007). Ao ser promovido o seu desenvolvimento, os indivíduos ficam um passo à frente da literacia tecnológica (Resnick, 2012; Phillips, 2009), deixando de ser meros utilizadores. Passam a ter, não só a capacidade de desenvolver os seus próprios sistemas, como reforçam competências adjacentes, sendo elas: o pensamento abstrato (utilização de diferentes níveis de abstração para perceber os problemas e, passo a passo, solucioná-los), o pensamento algorítmico (expressão de soluções em diferentes passos, de forma a encontrar a forma mais eficaz e eficiente de resolver um problema), o pensamento lógico (formulação e exclusão de hipóteses) e o pensamento dimensionável (decomposição de um grande problema em pequenas partes ou composição de pequenas partes para formular uma solução mais complexa) (Phillips, 2009; Resnick, 2007-2008). Tais capacidades, associadas por defeito à ciência da computação, transpõem-se para outras áreas de conhecimento e, consequentemente, para o dia a dia. ”

Entende-se, deste modo, que o pensamento computacional auxilia o desenvolvimento de diversas áreas de atuação humana, e não só aquelas onde um computador está envolvido, sendo essencial para adultos e jovens que buscam uma melhor condição de trabalho ou simplesmente querem resolver problemas cotidianos de forma mais eficaz.

Por estas razões, escolhemos a segunda área, pois no Brasil o estudo do pensamento computacional ainda não passa de um privilégio para poucos. Muitas crianças na escola ainda não sabem sequer dizer o que um profissional da ciência da computação faz. A matéria do Los Angeles Times "Want to prepare kids for the future? Teach them to code"[13] explicita o fato de que, com o avanço do ensino de computação nas escolas, mais jovens estarão expostos a carreiras de programação e, consequentemente, haverá mais profissionais de tais áreas disponíveis no mercado, que tem uma demanda cada vez maior.

Devido à atual conjuntura econômica, alguns poderiam alegar que devemos deixar de lado por enquanto o desenvolvimento dessas habilidades diferenciadas para focar em outras mais urgentes. Um ótimo exemplo de como tal decisão seria equivocada é a Estônia. Vivendo uma grande crise semelhante em 2007, a nação conseguiu recuperar-se com a ajuda da União Europeia e, principalmente, com o corte de gastos considerados supérfluos. Apesar disto, eles mantiveram a qualidade em sua educação, e mais, iniciaram em 2012 um projeto onde crianças de 7 anos aprendem o raciocínio computacional desde

cedo. A medida foi iniciada pela ONG Tiger Leap Foundation e logo obteve o apoio da iniciativa pública e privada[14]. Desta maneira, evidencia-se a relevância do desenvolvimento da área computacional, para que o ensino brasileiro não fique para trás em escala mundial.

É importante salientar que o objetivo de se ensinar às novas gerações os conceitos de programação não diz respeito principalmente à falta de profissionais nesta área, e sim, à falta de desenvolvimento deste modo de pensar nos currículos das escolas de hoje. O aumento na oferta de profissionais da área seria apenas uma consequência da implantação desse novo modelo de ensino. Estando tal conceito tão presente em nossas vidas quanto biologia, física ou matemática, por quê não estar também presente em nossas ementas escolares?

Tomando, então, como foco principal jovens e levando em consideração o ensino a distância, é de crucial importância uma plataforma que tenha certo apelo. O usuário em questão deve se sentir atraído, e mais que isso, não perder o interesse no estudo. Uma técnica que vem sendo difundida hoje é a ludificação (ou gamificação, do inglês, *gamification*), que se trata do processo de utilizar jogos para estimular diversas atividades, como por exemplo, o ensino.

> "Técnicas de jogos podem incentivar uma vida saudável, melhorar o ensino educacional, conscientizar e até promover produtos".

(Gabe Zichermann, consultor de empresas que planejam gamificar seus produtos e autor dos livros *Game-Based Marketing* e *Gamification By Design*[15])

Além de a gamificação poder ser utilizada como instrumento de incentivo, ela é uma porta para a tecnologia no espaço escolar, como aponta Lynn Alves, professora da UNEB (Universidade do Estado da Bahia)[16].

Sendo assim, levando em consideração o fato de que a programação pode ajudar o desenvolvimento do ser humano em diversas áreas de forma direta ou indireta, a realidade de a computação ser ainda uma área obscura para a maioria da população brasileira e a sua grande importância para toda a nova geração de profissionais que está por vir, resolvemos criar uma plataforma que facilitasse e incentivasse essa nova forma de expressão e raciocínio. Para tanto, aprovisionamo-nos da técnica de gamificação, para que pudéssemos atingir com mais sucesso nosso objetivo, dado que ela é considerada por si só um meio de inserção de tecnologia na vida infantil. Com isso, espera-se poder aos poucos inserir tal modo de pensamento na educação básica até que seja algo natural e tão questionável quanto a presença de história, biologia ou matemática em nossas instituições de ensino.

Neste trabalho, será exposto inicialmente, em sua parte objetiva, algumas informações que servirão de base para o entendimento geral da aplicação, como a pesquisa de campo para especificação do tema, nossas principais influências, tecnologias usadas e alguns conceitos importantes. Em seguida, haverá uma breve explicação de como os membros da equipe se organizaram para trabalhar. Adiante, apresentaremos os aspectos mais relevantes durante o processo de elaboração do jogo, abrangendo desde a definição da temática e a concepção do jogo, até os detalhes considerados no design da aplicação. Posteriormente, há uma seção dedicada a explicar o método de ensino e o contexto pedagógico adotados neste trabalho, justificando as decisões tomadas nesse âmbito. Por fim, apresentamos os resultados obtidos após análise de profissionais da área de jogos e as possíveis melhorias que poderiam ser implementadas futuramente.

Já na parte subjetiva, incluiu-se uma seção com as dificuldades e frustrações encontradas no decorrer do desenvolvimento da plataforma. Posteriormente, seguem as matérias de nossa graduação que tiveram maior relevância na elaboração deste projeto. Finalmente, um breve comentário pessoal de cada um dos integrantes sobre o trabalho em geral encerra esta monografia.

Objetivo

A meta principal deste trabalho é desenvolver uma plataforma de ensino de computação a distância para jovens baseada em gamificação com o propósito de difundir o raciocínio computacional e promover o seu ensino àqueles que não teriam essa oportunidade de outra maneira. Ao enfatizar a construção deste tipo de raciocínio e promover o ensino de programação, acreditamos que mais pessoas naturalmente terão interesse nesta área futuramente, onde a falta de indivíduos capacitados hoje é uma realidade.

Além disso, temos também como objetivo introduzir na vida do cidadão comum esta nova ferramenta, que é independente do desenvolvimento de softwares. Além de ajudar a compreender o funcionamento de tecnologias novas, também consiste em um instrumento para resoluções de problemas presentes no cotidiano de qualquer pessoa.

Pesquisa de Campo

Com o início das atividades de nosso trabalho de conclusão de curso, entramos em contato com diversos *web sites* e notícias relacionados à área de ensino da computação para crianças. Considerando a relevância das opiniões e sugestões de alguém que já trabalhasse em um campo semelhante, entramos em contato com Fábio Hirano, também aluno de Ciência da Computação pelo IME-USP, que trabalha atualmente na instituição de ensino MadCode.

A MadCode é um centro de tecnologia criativa para crianças e adolescentes de cinco a dezessete anos de idade. Através dos cursos oferecidos, os jovens que possuem interesse na área computacional podem adquirir novos conhecimentos em programação, robótica, criação de aplicativos, jogos e empreendedorismo.

Fizemos uma visita à unidade da escola localizada na região dos Jardins em São Paulo no dia dois de abril deste ano e além de conhecermos o ambiente onde as aulas são ministradas, conversamos com Fábio, que é um dos responsáveis pelo design de todos os cursos da instituição, com ênfase especial na programação dos cursos para crianças de oito a treze anos. Durante uma hora, fizemos uma série de perguntas que julgamos importantes e ouvimos as sugestões sobre qual caminho deveríamos trilhar de lá em diante. Além disso, fomos alertados sobre as possíveis dificuldades que iríamos encontrar no desenvolvimento e, principalmente, no design de nossa plataforma.

A primeira questão levantada foi sobre a idade do público-alvo de nossa plataforma. Nossa ideia é a de ensinar o pensamento computacional para jovens que tiveram pouco ou nenhum contato com programação. De acordo com os profissionais da MadCode, acredita-se que a idade de doze anos é ideal para o início de tal atividade. Como foi apontado, crianças mais novas ainda não têm a base matemática para tanto, e mais que isso, não possuem a capacidade de dedicar atenção por muito tempo à mesma atividade. Ademais, a partir desta idade, as dificuldades impostas pela nossa plataforma para que os conceitos sejam aprendidos corretamente terão maior probabilidade de atingir tal objetivo, e não atuarão como uma barreira em que a criança esbarraria e perderia o interesse.

O segundo tópico tratado foi o método de ensino. Uma das estratégias adotadas com sucesso pela instituição é o contato com jogos digitais. Muitos jovens acabam sendo atraídos para o mundo da computação através de jogos eletrônicos em videogames, computadores ou celulares. Em um dos módulos de ensino da empresa, por exemplo, as crianças criam a lógica de alguns jogos clássicos simples como Space Invaders[17] (lançado para fliperama em 1978) ou

MegaBall[18] (lançado para *Amiga* na década de 80). Sendo assim, ficou claro que essa era uma estratégia já previamente testada e com grande possibilidade de êxito.

O terceiro ponto abordado foi a grande diferença que teremos na interação com nosso público-alvo. Deveríamos atentar especialmente ao fato de que, em uma escola que dispõe de espaço físico, a perda da atenção ou interesse, por quaisquer que sejam os motivos, podem ser evitadas ou mitigadas pelo contato direto do professor com o aluno. No caso de uma ferramenta online, esse contato torna-se inviável. Desta forma, a retenção tornou-se uma das características mais delicadas de nosso design, pois não existe triunfo em criar um mecanismo de ensino perfeito, que não atraia e sustente o interesse dos indivíduos do grupo focado.

Por fim, ainda por não dispormos de um local físico onde os alunos possam aprender, já que o aprendizado se dá através de um ambiente virtual, precisaríamos implementar uma forma de continuidade e acompanhamento do progresso de cada usuário. Por isso, seria necessária a utilização de um banco de dados em um servidor online para que os avanços de cada usuário sejam armazenados.

Após nossa conversa, tivemos ainda a oportunidade de conhecer o ambiente e material de trabalho utilizados pelos profissionais daquela unidade. Além de servir como fonte de aprendizado e ideias, nossa visita também nos motivou ainda mais para o desenvolvimento de nosso trabalho por entrarmos em contato direto com o interesse dos jovens pela área de tecnologia.

Influências

Após determinarmos o foco da nossa aplicação, fizemos uma busca por outros sistemas que possuem uma proposta semelhante. Existem diversos exemplos distribuídos pela internet, alguns deles muito bem-sucedidos, e outros com propostas muito semelhantes à nossa.

Scratch

Desenvolvido no *Instituto de Tecnologia de Massachussets* pelo grupo *Lifelong Kindergarten*, Scratch[19] é comprovadamente uma ferramenta efetiva para o desenvolvimento do raciocínio computacional, como demonstrado no artigo da Universidade do Minho em Portugal, "*Scratch, uma opção válida para desenvolver o pensamento computacional e a competência para resolução de problemas*"[12]. Ele utiliza o método de programação em blocos. Estes, assim como em nosso projeto, representam comandos de ação ou controle, possibilitando o uso também de condicionais. Além das funcionalidades que foram aproveitadas por nós, existem diversas mais, como variáveis, sensores, entre outras.

Um ponto negativo que percebemos no Scratch é o fato de que não existem objetivos bem definidos, o que poderia significar a perda de usuários pela falta de um fator de engajamento maior. Um propósito mais claro e palpável pode incentivar o jovem a querer resolver os problemas, enquanto a falta dele pode acarretar em uma falta de interesse. Além disso, pelo fato de não possuirmos um local físico de ensino, professores que pudessem ser contatados e nem uma comunidade ativa de usuários pronta para estimular o conhecimento e sanar possíveis dificuldades, precisávamos de um apelo maior para garantir a retenção dos jogadores. Sendo assim, a aplicação do recurso de gamificação em nosso jogo acabou por suprir nossas necessidades neste quesito.

Lightbot

Mais voltado para o lado da gamificação, Lightbot[20] também utiliza blocos para programação, porém, diferentemente de *Scratch*, introduz um objetivo. Neste jogo, é preciso programar um robô e fazer com que ele passe por determinadas posições do cenário. Além disso, o jogo introduz a divisão do mapa em quadrantes e a movimentação do protagonista em passos discretos.

Apesar de não ser uma plataforma livre, ela oferece ao usuário algumas horas gratuitas para experimentação. Isso nos possibilitou uma maior visão dos pontos que poderiam ser utilizados também em nosso software.

Code Combat

Nossa última inspiração veio também de uma plataforma que é apenas parcialmente gratuita. Code Combat[21] possui uma abordagem um pouco diferente dos projetos anteriores. A mais importante é o fato de ele ser escrito, ou seja, utilizar o teclado como forma de entrada de comandos. O usuário pode escolher entre Python, Lua, Javascript, Coffeescript, Closure ou Io para escrever seu programa e levar o herói à vitória. Além disso, existe o modo de jogo simultâneo, para mais de um jogador.

Apesar de todas essas características que na verdade distanciam Code Combat de nossa aplicação, existe um detalhe que os aproxima. Apesar de o protagonista se movimentar de forma discreta, os inimigos não se comportam desta maneira, criando um desafio adicional para o usuário - analogamente à movimentação dos asteroides em nosso jogo.

Gamificação

Gamificação, ou ainda ludificação, é o uso de elementos de design de jogos aplicados a circunstâncias não necessariamente envolvidas no contexto de um jogo, como educação ou treinamento físico, com a finalidade de assegurar o engajamento do 'jogador' por mais tempo. O livro *Game Based Marketing*, de Gabe Zichermann, define o termo gamificação como o processo de utilizar o pensamento e a mecânica de um jogo para engajar o público e solucionar problemas. De modo geral, o processo de gamificação envolve atrair usuários utilizando-se de elementos típicos de jogos, como sistema de pontuações, moedas virtuais, ou atribuição de recompensas, de forma que outras atividades pareçam mais atraentes sob o pretexto do jogo.

O termo foi cunhado somente em 2002, mas o conceito de utilização de jogos em outras áreas já era empregado muito antes[22]. Somente em 2010 a expressão passou a ser utilizada em maior escala. Hoje diversas empresas utilizam gamificação em seus produtos, como a DevHub, uma empresa que fornece uma ferramenta online para criação de sites e páginas. Por meio da gamificação do processo de criação da página do usuário, eles aumentaram o índice de clientes que completam esse processo de criação em quase 8 vezes[23]. Uma plataforma de gamificação de vendas, chamada FantasySalesTeam, foi comprada pela Microsoft recentemente[24], com o objetivo de motivar e aumentar a produtividade de alguns times internos desta companhia.

O artigo científico "*Does Gamification Works?*"[25], feito em parceria pelas Universidades de Tampere e de Aalto, ambas na Finlândia, exibe um estudo sobre a literatura científica relacionada ao tema gamificação, compilando e interpretando os resultados de 24 artigos acadêmicos que, por sua vez, realizaram estudos empíricos independentes acerca do mesmo tema. Apesar do advento recente do objeto de estudo, o que restringe o número de trabalhos científicos na área atualmente, foi concluído a partir da análise estatística do resultado dos diversos outros trabalhos que, de fato, a gamificação é um processo eficaz.

Cada vez mais, a ludificação é utilizada com o propósito de incrementar produtividade em empresas, ou ainda, de favorecer o aprendizado. Citando o livro "The Gamification of Learning and Instructions"[26], temos:

> "Técnicas baseadas em jogos, ou gamificação, quando usadas de forma apropriada, têm o poder de engajar, informar e educar."

Foi com base nessa premissa que optamos pela utilização de gamificação em nossa plataforma. Assim, esperamos garantir por meio dos elementos do design de jogos uma maior retenção dos usuários de nossa aplicação.

Tecnologias

Unity3D

Unity3D, ou apenas Unity, é um motor de jogo (do inglês, *game engine*) que possui um ambiente de desenvolvimento integrado desenvolvido pela empresa *Unity Technologies*. Ele é muito utilizado nos dias de hoje para o desenvolvimento de jogos 3D, porém, não se restringe apenas a esses fins.

O grande diferencial desta ferramenta é sua portabilidade e facilidade de exportação para diversas plataformas, como *iOS*, *Android*, web, *XBox*, entre diversos outros. Isso torna o desenvolvimento mais centralizado e faz com que não seja necessário a elaboração de um código fonte diferente para cada tipo de dispositivo.

Por estar disponível em duas versões, as chamadas versão *Pro* (uma opção paga) e a versão *Comum* (livre para o uso individual e educacional), optamos pela segunda.

Por que Unity ?

Tendo em mente o aproveitamento da grande quantidade de documentação disponível, os diversos exemplos que conseguimos encontrar nos meios digitais e a possibilidade de exportarmos nosso projeto para plataformas acessíveis como navegadores de internet, decidimos em conjunto utilizar Unity para o desenvolvimento do núcleo do nosso projeto.

Além das diversas vantagens apresentadas acima, decidimos pela utilização da plataforma por ela ser largamente utilizada. De acordo com o próprio *web site* da empresa[27], já são mais de quatro milhões de desenvolvedores que utilizam Unity, o que representa 45% do mercado global de motores de jogos e, aproximadamente, o triplo do concorrente mais próximo. Isso proporciona uma enorme comunidade online que contribui ativamente para sua documentação, facilitando a resolução de problemas que podem aparecer no desenvolvimento.

Terminologias e Conceitos

Unity utiliza seu próprio vocabulário dentro de sua *engine* para que os desenvolvedores possam se entender e discutir os assuntos que julgam

pertinentes com facilidade. Alguns itens, que serão descritos a seguir, serão necessários para o melhor entendimento de nosso projeto e seu funcionamento.

Projetos

É o conjunto de elementos que compõe todo o software a ser desenvolvido, contendo cenas, imagens, sons, etc. Por padrão, um projeto Unity é inicializado com diversas pastas que, na verdade, são diretórios criados no sistema operacional do usuário. É uma boa prática organizar arquivos semelhantes em um mesmo diretório, mas esse não é um requisito da tecnologia Unity em si.

Cenas

Unity trabalha com o conceito de cenas, como pode ser observado em sua documentação[28]. Uma Cena é a maior subdivisão hierárquica dentro de um projeto de Unity, e cada cena pode ser vista individualmente como uma "*tela*" do jogo. Cada cena contém os objetos que lhe são pertinentes. Por exemplo, nossa tela principal de jogo contém a nave, os botões, os painéis com opções de comandos, entre outros.

Na prática, as cenas permitem o desenvolvimento do jogo em partes independentes, e o jogo em si pode ser descrito pelas cenas que o compõem. No nosso caso, temos a cena de escolha de nível, a cena de explicação no início de cada fase e a cena principal, onde ocorre o jogo em si, além de cenas de abertura e fim de jogo.

Assets

Os *Assets* são os recursos do projeto que podem ser referenciados dentro do mesmo, como por exemplo: scripts, sons, imagens, entre outros. Eles são associados à outros componentes, como um objeto do Unity, ou mesmo outro Asset.

Game Objects

Game Objects representam um dos conceitos chave do Unity. Eles não são os objetos referentes do paradigma da programação orientada a objetos, mas sim, objetos próprios do Unity, formando as bases de todas as entidades do programa[29]. São eles que compõem as cenas, podendo ser instanciados,

modificados, além de adquirirem comportamentos através de *scripts* associados.

Prefabs

Prefabs são moldes (do inglês, *templates*) de objetos do Unity. Na prática, são elementos que possuem características e comportamentos, podendo ser instanciados, ou seja, clonados. Apesar de já possuírem certas características determinadas na criação do molde, após serem instanciados, também podem ser modificados dinamicamente, sem que o *Prefab molde* seja alterado.

Eles são utilizados para a criação de objetos recorrentes no jogo; em nosso caso, os comandos, a nave, o planeta, etc.

Componentes

Um componente é o que dá uma característica a um *Game Object*. Por exemplo, um corpo físico que pode ser usado no tratamento de colisão com outros corpos. Outros exemplos recorrentes são: um *script* que atribua determinado comportamento, algum som, entre várias outras possibilidades.

Scripts

São os elementos que contêm código fonte e atribuem a algum *Game Object* determinada característica. Em Unity, são admitidos códigos escritos nas linguagens C# e javascript. Em nosso jogo, optamos pela primeira opção por preferência pessoal.

Monodevelop[30]

O *Monodevelop* é um editor de texto e um ambiente de desenvolvimento integrado à plataforma Unity. Ele já acompanha a instalação do próprio Unity e é a opção de editor de texto sugerida. Com suporte para várias linguagens, quando integrado ao Unity, ele é usado para o desenvolvimento em C# e *Javascript*.

Compilação

Uma vez salvo o arquivo de código e com a janela de desenvolvimento do Unity ativa, o código já é compilado e os erros são disponibilizados no

terminal automaticamente. Ou seja, Unity já acompanha um compilador C# totalmente integrado e de fácil utilização, uma vez que não é exigido nenhum conhecimento prévio ou ação do usuário.

Testes e simulações

Outra ferramenta de grande utilidade fornecida pelo Unity é seu console de testes, completamente integrado ao programa. Com apenas um botão é possível simular a cena que está sendo desenvolvida e, além disso, acompanhar em tempo real as propriedades dos objetos adicionados à Cena.

Com essa funcionalidade, evita-se ter que sempre exportar o projeto para alguma plataforma para que se possa testar o código, agilizando e facilitando o desenvolvimento e implementação de novas funcionalidades.

Exportando para plataformas

Como foi mencionado anteriormente, o que torna Unity muito popular é a facilidade no processo de exportação do seu projeto para diversas plataformas diferentes. Isso é disponibilizado de forma muito simples. Na janela do editor de projeto do Unity, basta selecionar a plataforma para qual o software deve ser exportado e selecionar a opção *Exportar*. Com isso o projeto atual é reconstruído especificamente sobre a plataforma desejada. Por exemplo, para iOS é criado um projeto *Xcode* em Objective-C, para *Android* é criado um arquivo de extensão *.apk*, e no nosso caso são criados arquivos com extensão javascript prontos para serem integrados a uma página HTML.

Git e Github

Git é um sistema de controle de versão distribuído, ou seja, uma forma de gerenciamento remoto de código fonte. É um software livre distribuído pela versão 2 da *GNU General Public License*[31]. Ele foi desenvolvido inicialmente por Linus Torvalds, criador do Linux, para a gestão do desenvolvimento do núcleo desse sistema operacional. É adotado hoje em dia em milhões de projetos. Esses projetos são armazenados remotamente nos chamados *repositórios*, que possuem ao menos duas cópias, uma local em cada máquina que trabalha sobre esse projeto e outra remota, onde o projeto de todas as máquinas é concentrado.

Esta ferramenta trabalha com o uso de *branches*, ou seja, ramos de desenvolvimento; e *merges*, união de *branches* diferentes. A implementação de novas funcionalidades é geralmente feita em um ramo diferente do projeto,

evitando a interferência dos outros desenvolvedores que trabalham de forma concomitante. Uma vez que a branch apresenta a funcionalidade completa, o desenvolvedor do projeto faz então um *merge*, uma união desse ramo secundário com o braço principal de desenvolvimento, minimizando possíveis danos no processo de integração de novas funcionalidades.

Github[32], por sua vez, é um sistema de hospedagem online do controle de versão *Git* desenvolvido em *Ruby on Rails*. Além disso, disponibiliza diversos serviços adicionais como estatísticas dos repositórios, serviços de redes sociais, entre outros.

Por que Git?

Sendo uma ferramenta de fácil manipulação, livre e gratuita que facilita o gerenciamento de projetos onde há um número grande de pessoas envolvidas, resolvemos adotar *Git* como parte do processo de desenvolvimento. Com ele, nosso grupo pôde trabalhar separadamente diminuindo o impacto deste paralelismo. Sendo assim, pudemos maximizar o desenvolvimento de nosso software, além de ter fácil acesso a versões anteriores do projeto, no caso de uma decisão errada ter sido tomada e implementada.

Trello

Trello*[33]* é uma ferramenta de gerenciamento de projetos. Uma de suas grandes vantagens é o fato de ser online. Inicialmente criado pela *Fog Creek Software Products*[34], em 2014 tornou-se uma empresa própria. Ele é largamente utilizado, chegando a mais de cinco milhões de usuários, conforme divulgado pela própria empresa. Além de auxiliar em projetos que envolvem a computação, ele também serve para diversas áreas, como para a preparação de aulas, por exemplo. Ele disponibiliza para cada projeto um quadro, onde é possível adicionar cartões e atividades, dividindo-as de forma pertinente a cada projeto. Além de estar disponível como web site, ele também pode ser utilizado em plataformas móveis android, iOS e Windows Phone.

Basicamente, Trello é um quadro online onde equipes podem praticar *Kanban*[35]. *Kanban* é um paradigma que estabelece uma abordagem visual para se executar a distribuição de tarefas em um projeto. Normalmente, essa abordagem é feita através da representação das tarefas a cumprir como cartões ou tíquetes fixados em um quadro, cuja posição e conteúdo orientam o desenvolvimento geral do projeto, ditando a urgência das tarefas e qual a próxima funcionalidade que deve ser produzida em seguida. Dentre as

aplicações disponíveis para a prática do Kanban, a que leva mais destaque é o Trello, pela gratuidade dos serviços básicos e pela interface amigável, requisito básico do método.

A adoção da metodologia de Kanban é de grande benefício aos projetos com poucos membros, pois essa técnica auxilia na estruturação do desenvolvimento de projetos de pequeno e médio porte, permitindo a todos os membros envolvidos verificar o progresso e o andamento da implementação das tarefas dos outros membros, o que efetivamente contribui para a sincronização do desenvolvimento como um todo. Kanban, pela simplicidade e representação visual do espaço de tarefas, é facilmente integrado aos processos de desenvolvimento ágil, sendo incorporado e adotado pela maioria dos métodos como *scrum* e programação extrema.

Em nosso projeto, o quadro de Kanban adotado era relativamente simples, dado a pequena quantidade de membros que compõem o time de desenvolvimento. O quadro é composto pelos *tickets* das tarefas cumpridas e das ainda por desenvolver, decompostas em listagens de tarefas menores a serem resolvidas dentro de cada cartão.

Por que Trello?

Optamos por esse serviço pois, apesar de não ser completamente gratuito, disponibiliza de forma aberta muitas ferramentas, as quais consideramos suficientes para o gerenciamento de nosso projeto em grupo. Trello funcionou para nós como um Kanban de fácil acesso para todos os membros do projeto.

Metodologia

Nesta seção, pretendemos explicitar nossa metodologia de pesquisa, ou seja, como e quais foram os procedimentos que realizamos ao longo de nossos estudos e o processo de tomada de decisões.

Inicialmente, tínhamos como objetivo a construção de um sistema onde fosse possível treinar o raciocínio lógico e aprender computação. Esse objetivo, um tanto vago e ambicioso, foi refinado ao longo de reuniões e encontros entre os membros do grupo e nosso orientador. Decidimos cercar o tema em torno do ensino de computação, com foco especial em crianças e adolescentes.

Consideramos prontamente o público infantil como alvo de nossos esforços porque acreditamos que um incentivo prematuro seria capaz de, não apenas melhorar o raciocínio cotidiano, como também aumentar a procura futura por nossa área, que hoje tanto carece de profissionais[36]. Esperamos assim levar adiante os conceitos que introduzimos ao usuário ao longo de sua vida, que podem ser úteis além do campo da informática.

Foi sob essas condições que começamos a realizar diversas pesquisas. Uma vez que decidimos qual seria o objeto de nosso estudo, fizemos sucessivas buscas sobre o modelo atual de ensino de computação voltado ao público infantil em diversos países. Para tal, utilizamos sites de notícias internacionais e uma coletânea de impressos e periódicos, parte dos quais foram cedidos gentilmente por nosso orientador.

Através dessas pesquisas, compreendemos que há atualmente no exterior uma crescente preocupação com o ensino de programação nas escolas infantis. Exemplos claros da convergência de opiniões a respeito do ensino de computação podem ser encontrados nos currículos escolares da Inglaterra[2] e da Austrália[37], onde recentemente o ensino de programação foi incluído como conteúdo obrigatório para crianças e adolescentes a partir dos 5 anos de idade.

Foi com base nessa conjuntura que decidimos implementar um sistema que ensinasse computação. Restava ainda saber o que, especificamente, seria essa aplicação e como nos proporíamos a ensinar efetivamente a distância com esta ferramenta. Considerando nosso público-alvo, imaginamos que seria difícil garantir a retenção de usuários apresentando um projeto final semelhante às ferramentas modernas de ensino digital que são facilmente encontradas na internet, como Code Academy[38] ou Programaê[39]. Acredita-se que crianças têm dificuldade em manter-se focadas por longos períodos de tempo[40] e que programas de estudo massantes podem desestimular o interesse dos usuários um pouco menos maduros. Foi por isso que decidimos por ensinar computação e o raciocínio lógico por meio de um jogo digital.

A ideia do aprendizado por meio de jogos é um conceito bem explorado em diversas áreas do conhecimento. Esse fenômeno moderno é também conhecido como gamificação. O artifício do jogo torna as atividades mais interessantes por meio de elementos de Design de Jogos, que propiciam ao usuário desfrutar melhor da atividade que está sendo proposta[41]. No nosso caso, pretendemos fixar o conteúdo ensinado por meio do jogo, prendendo a atenção do usuário enquanto este tem que desempenhar tarefas específicas para avançar ao próximo estágio do programa. Esperamos com isso poder segurar por mais tempo o usuário, de forma que os conceitos ensinados em nossa plataforma possam ser fixados.

A critério de confirmação da nossa linha de raciocínio, buscamos profissionais da área do ensino infantil e pré-adolescente. Fizemos uma rodada de entrevistas com professores da empresa MadCode[42], que se propõe a ensinar programação e robótica a jovens de cinco a dezessete anos. Nessa conversa sobre modelos de ensino nos foi levantado vários pontos de interesse que nortearam o desenvolvimento e o design de nossa aplicação, assim como nosso principal problema: Plateias infantis têm naturalmente mais dificuldade com respeito à concentração e à atenção, e por isso devemos ter uma consideração especial em relação à interface com o jogador.

Passamos então a um estudo um pouco mais aprofundado do meio pedagógico em que decidimos inserir nosso sistema e examinamos aplicações existentes que se propõem educar como nosso projeto. Analisamos a interface de usuário, a forma como o raciocínio computacional é inserido nas aplicações e como os jogos levam em conta a progressão do ensino. As aplicações mais consagradas na área, entre elas Scratch e Code Combat, foram estudadas a fim de definir qual seria o caráter de nossa aplicação. Essas referências foram ainda utilizadas na tomada de decisões de projeto como, por exemplo, a forma da interface com o usuário e os elementos principais dos jogos, os blocos de comandos. Pode-se observar mais detalhes desses dois softwares na seção *Influências* deste trabalho.

Desenvolvimento

Desde o princípio do desenvolvimento da nossa aplicação, durante as etapas de estimativa de escopo e elaboração do cronograma de tarefas, tivemos em mente a divisão harmoniosa dos encargos de nosso projeto. Tentamos, na medida do possível, dividir as tarefas em partes independentes, que pudessem ser atribuídas aos membros da equipe sem que uma comprometesse o andamento da outra.

Foram realizadas reuniões periódicas de alinhamento entre os membros da equipe, assim como reuniões de acompanhamento semanal com nosso orientador. Nestas, cada um de nós, ou todos coletivamente, nos propusemos a resolver algum problema que surgiu durante o desenvolvimento, ou a adicionar novo conteúdo ao projeto.

Introdução de Funcionalidades

Seguimos um processo de discussão interna para a aprovação de conteúdo novo a ser adicionado ao projeto. Esse processo envolve a explicação da funcionalidade a ser adicionada para os outros dois membros do grupo, e coletivamente decidimos a introdução dela no desenvolvimento do programa ou não. Após o processo de aprovação, ela é adicionada no Trello sob a lista das tarefas a serem implementadas no quadro de Kanban.

Desenvolvimento Individual

O desenvolvimento individual de funcionalidades é a forma pela qual o projeto mais cresceu. Grande parte do conteúdo que temos hoje foi concebido a partir das discussões de aprovação e *brainstorms* de implementação, sendo produzido individualmente por cada um dos membros do projeto. A divisão de tarefas era feita de acordo com a disposição desses encargos em nosso quadro visual de Kanban no Trello, e cada integrante era responsável por garantir que a branch de desenvolvimento principal do projeto estivesse sempre funcional, ou seja, sem bugs ou problemas que impedissem a jogabilidade da aplicação. Para tanto, fez-se necessário que, antes do processo de união de código-fonte feito através do Git, o responsável pela execução da tarefa testasse o código desenvolvido a fim de validar sua corretude, não somente do novo trecho de código, mas de como esse trecho se relacionaria com o restante do programa.

Depois que o desenvolvedor de uma funcionalidade considerasse que a testou satisfatoriamente, este então lidaria com as ferramentas de manuseio de código compartilhado, os chamados sistemas de versionamento. Como já dito em seções anteriores, versionamos o projeto por meio do Git, que apesar de ter sido suficiente para o manuseio de scripts e arquivos de código baseados em texto, deixou a desejar com os inúmeros arquivos binários existentes nos projetos de Unity.

Geralmente, cada membro era encarregado de solucionar qualquer problema ou bug relacionado aos trechos de implementação de si próprio, mas também tivemos sessões de discussão onde buscamos soluções para problemas de implementação geral do sistema. À medida que o escopo do projeto crescia, estas foram se tornando cada vez mais frequentes. Imaginamos que isso ocorra devido a complexidade e acoplamento crescentes das diferentes partes do projeto.

Desenvolvimento em Equipe

Parte do projeto também foi desenvolvida em conjunto por todos os integrantes da equipe. Por vezes, tínhamos reuniões de implementação onde nos propusemos a implementar uma funcionalidade nova, ou ainda consertar um bug mais trabalhoso. Também nos reunimos algumas vezes na fase de implementação dos diversos níveis que existem no jogo, tanto para discussões de concepção e design quanto para implementação dos scripts de nível que gerenciam as instâncias de Game Object.

Essas reuniões de implementação geralmente foram empregadas em discussões sobre o problema corrente a ser resolvido ou o propósito da funcionalidade a ser adicionada. Consideramos essas reuniões de grande importância ao projeto porque elas também serviram de alinhamento dos integrantes da equipe com relação aos objetivos do programa.

Concepção do Jogo

Em design de jogos, os processos de concepção e criação podem ser agrupados basicamente em duas linhas de pensamento, que diferem quanto ao aspecto do jogo a ser priorizado durante este processo. São o desenvolvimento orientado à narrativa e o desenvolvimento orientado à mecânica de jogo.

Narração de histórias (do inglês, *storytelling*) é um recurso comumente utilizado em marketing e que consiste em atribuir uma identidade ou uma história a uma empresa ou produto. O objetivo da narração é tornar um produto mais impactante e formar uma conexão com o público-alvo através de um forte apelo emocional. Quanto mais criativa, autêntica e envolvente a história, maior será o grau de identificação e engajamento experimentados pelo público[43]. De forma similar, tais técnicas são aplicadas no domínio dos jogos a fim de contar uma história interessante para prender e instigar o usuário, de forma a motivá-lo a progredir no jogo como um leitor curioso em desvendar o enredo de um livro. Portanto, a narrativa de um jogo diz respeito aos elementos que, quando reunidos, contam uma história: as personagens envolvidas e suas personalidades, os conteúdo dos diálogos, os avanços e as reviravoltas durante a trajetória do protagonista, os diferentes itens manipuláveis, dentre outros.

Desta forma, o enredo de um jogo pode influenciar diretamente no envolvimento e retenção de seus usuários. Além disso, uma história com um final memorável pode render desde comentários positivos e recomendações entre amigos, até uma legião de fãs que podem garantir o sucesso de continuações e expansões futuras. Isto explica, em partes, a incrível marca de um milhão de cópias vendidas atingida recentemente, pelo jogo Final Fantasy XIII em seu primeiro fim de semana à venda na América do Norte[44], por exemplo. A franquia, pertencente a empresa Square-Enix, também figura entre as dez mais bem-sucedidas da história dos jogos e reúne fãs ao redor do mundo inteiro[45].

Já a mecânica do jogo concerne os meios disponíveis de interação entre usuário e jogo, e os resultados obtidos através destas relações. Jogos cuja mecânica é bem desenvolvida provém diversão e uma boa experiência aos jogadores ao disponibilizar possibilidades diversas de interação; seja uma coleção de habilidades dominadas pelo protagonista, seja um arsenal de itens com efeitos variados, ou até mesmo um conjunto de tarefas a serem cumpridas que, embora repetitivas, sejam desafiadoras. Nestes casos, a narrativa presente costuma surgir como mero pretexto para as tarefas a serem executadas e que consistem, de fato, o componente intrigante do jogo. Exemplos disso são as

célebres séries Guitar Hero e Rock Band, onde o jogador basicamente deve tocar músicas famosas, dispondo de aparelhos simplificados que simulam instrumentos musicais reais para isto. Conforme afirma Pedersen[46], o sucesso dessas franquias, que já venderam milhões de cópias ao redor do mundo, decorre de sua interface amigável, inovadora e, até então, inexplorada.

Naturalmente, esses dois princípios de desenvolvimento de jogos não são excludentes; na realidade, um designer de jogos deve se ater aos diferentes benefícios e limitações que ambos os métodos podem trazer para o seu projeto. Afinal, é evidente que um jogo terá mais chances de sucesso se investir em ambos os aspectos. Por exemplo, o jogo Assassin's Creed IV: Black Flag vendeu mais de 11 milhões de cópias em menos de um ano após seu lançamento em 2013[47]. Décimo jogo da franquia da empresa Ubisoft, é conhecido por apostar tanto em seu enredo - onde o jogador é imerso em um contexto histórico e vive um assassino -, quanto na mecânica de jogo que permite explorar o ambiente como um todo e fazer uso de estratégias variadas, efetivamente convencendo que se está em outro universo. Entretanto, é importante salientar que, normalmente, a ideia inicial de jogo é aperfeiçoada priorizando-se um dos aspectos citados: enredo ou interação.

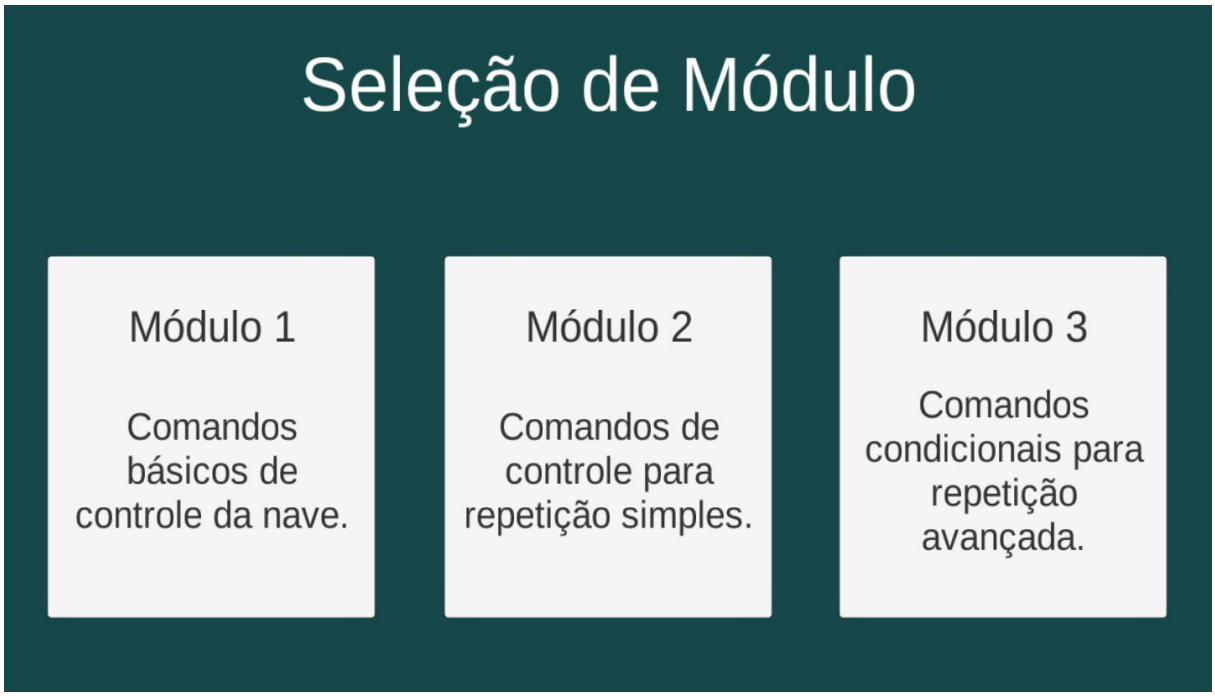
Em nosso jogo, nomeado Torre de Controle[48], optamos pela abordagem focada na mecânica do jogo, e isto se deu por duas razões. Vale ressaltar que nossa principal meta era apresentar os conceitos essenciais e comuns às linguagens de programação através do recurso da gamificação, mascarando, assim, conceitos complexos e abstratos da programação por meio de uma interface responsiva e uma mecânica intuitiva (veja seção *Introdução*). Logo, os primeiros esboços produzidos durante a fase de concepção do projeto já refletiam esta tendência em se abstrair o conteúdo do jogo a fim de se concentrar esforços na adoção de uma metodologia pedagógica acertada.

Em segundo lugar, é bem mais fácil de se produzir posteriormente um enredo rico que se encaixe de maneira convincente no modelo de jogo pensado do que o contrário. Arquitetar todos os elementos, recursos e objetivos do jogo após a elaboração da história e das personagens envolvidas pode ser um grande desafio. As limitações impostas pelo contexto do jogo previamente definido e consolidado podem não permitir a inclusão futura de alguns recursos. Caso estas restrições sejam contrariadas, corre-se o risco de gerar situações forçadas e aparentemente desnecessárias, que soariam falsas ou inverossímeis para o jogador. Assim, se tivéssemos seguido por este caminho, talvez o produto final obtido não atendesse às necessidades do usuário e nem correspondesse às nossas expectativas quanto à usabilidade.

Temática do Jogo

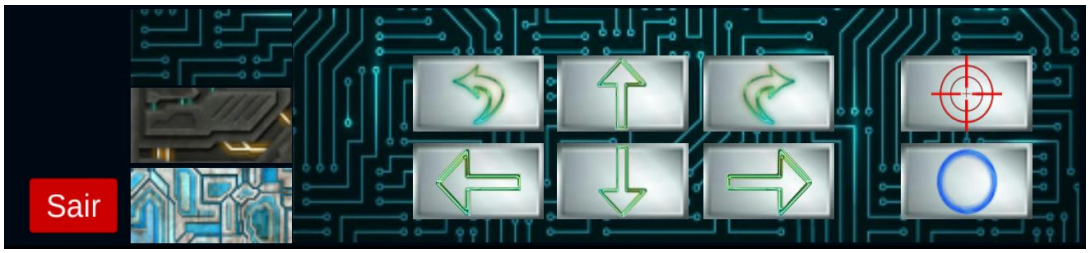
O jogo consiste em uma expedição intergaláctica, onde o jogador é convidado a programar uma nave espacial a distância a fim de explorar o universo, recolher rochas e cristais raros para estudo, ao mesmo tempo que enfrenta diversos perigos e tenta regressar ileso com a nave para o seu planeta natal. Devido à imensa distância, a sequência de comandos que a nave seguirá deve ser submetida integralmente de uma única vez para evitar custos e problemas de atraso na transmissão. Além disto, os comandos armazenados são simples, previamente bem definidos e não podem ser numerosos. Logo, posicionado em uma **torre de controle**, o jogador deve se empenhar e fazer bom uso de seu arsenal limitado para cumprir a missão espacial.

Inicialmente, somos apresentados à Tião, mascote do jogo e comandante da equipe responsável por transmitir a missão, que fornece as explicações iniciais a respeito do funcionamento e modo de interação com o jogo. Sempre que necessário, este porquinho amigável surge para dar dicas e introduzir novos elementos ao jogo, à medida que os níveis e módulos são superados. Ao todo, o jogo divide-se em três módulos, contendo dez níveis cada.

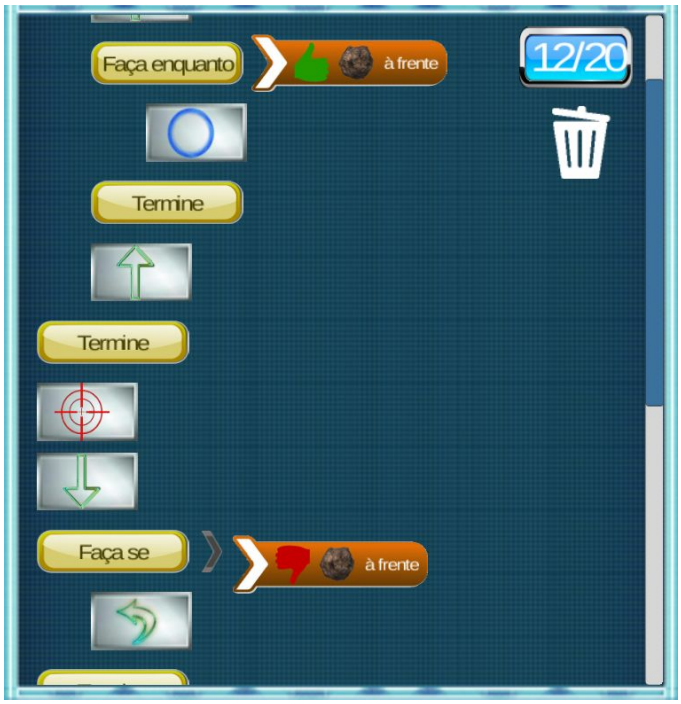


A interface é composta essencialmente por um conjunto de menus interativos e pode ser destrinchada em três partes principais.

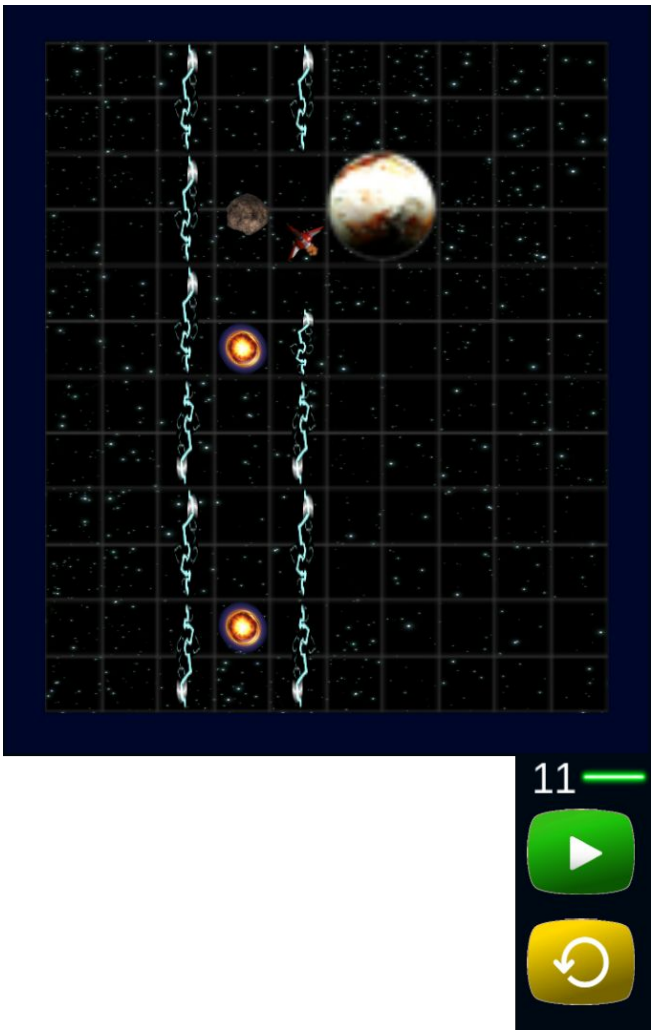
- Menu: é onde o usuário encontra à sua disposição todas as funcionalidades que permitem operar a nave espacial. O menu dividi-se em três abas, com o intuito de categorizar os comandos de acordo com a sua natureza. A primeira aba guarda os comandos básicos de manipulação da nave. Já a segunda aba armazena comandos de controle, como comandos de repetição simples e condicional. Por fim, na terceira aba localizam-se as condições a serem anexadas aos comandos de controle pertinentes. O menu permanece ativo somente enquanto o jogo está no modo interativo. Também contém um botão 'Sair' que permite abandonar o nível corrente.



- Painel de comandos: é o painel onde os comandos são depositados e armazenados em ordem, que pode ser modificada. A ordenação ocorre no sentido vertical, de maneira que os primeiros comandos são mantidos sempre no topo. Analogamente ao menu, o usuário pode interagir com o painel apenas no modo interativo; durante o modo de execução, os comandos conservam-se estáticos e, na medida em que vão sendo executados pelo simulador, mudam de cores. Possui um número máximo de comandos que podem ser armazenados e que não pode ser excedido. Comandos depositados no painel são apagados ao serem arrastados para a lixeira. Acima desta, encontra-se um mostrador, responsável por indicar o número de comandos em uso e o limite máximo (que pode variar segundo o nível). Caso o número de comandos extrapole o limite da tela, uma barra de rolagem é disponibilizada para que possam ser visualizados.



- Simulador: trata-se do interpretador gráfico dos itens depositados no painel de comandos. Esta tela é responsável por apresentar sequencialmente todas as animações e efeitos resultantes dos comandos escolhidos e dispostos segundo o usuário. Também inclui um botão encarregado de reger a alternância entre o modo interativo e o modo de execução do jogo, e outro capaz de restaurar o nível em questão. Acima destes botões, um contador indica quantos disparos de raios laser podem ser efetuados pela nave. Contrariamente às outras duas partes, o simulador segue estático durante o modo interativos, em que tão somente apresenta o cenário espacial da missão corrente. Ao iniciar o modo de execução, o simulador efetua a interpretação dos comandos, que também é sinalizada pelo painel de comandos, conforme já descrito.



Cenário

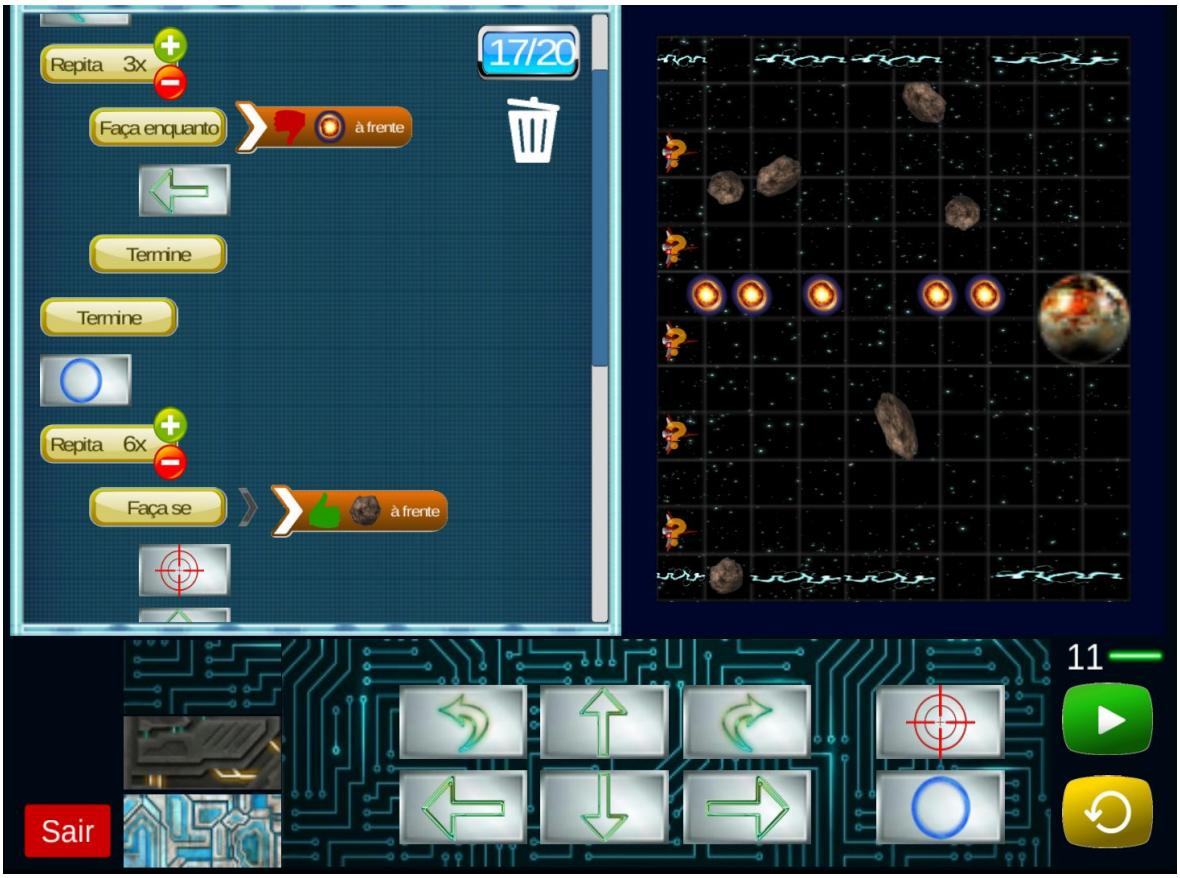
Através do simulador, obtemos informações a respeito do paradeiro da nave e das coordenadas específicas de elementos espaciais, tais como: o planeta natal, nuvens de asteroides, campos de força e minérios raros. Estas coordenadas são interpretadas pelo simulador e apresentadas através de um sistema de divisão do mapa da galáxia em quadrantes, simplificando, assim, o entendimento do processo de movimentação da nave.

Dentre os perigos enfrentados pela nave, constam os campos de força e os asteroides; se a nave encostar em algum deles, ela explodirá imediatamente. Enquanto os campos de força são imóveis, os asteroides podem transladar pelo universo, embora alguns permaneçam em movimento periódico de rotação. Além disso, asteroides podem ser destruídos pela arma de raios laser acoplada à nave, ao passo que campos de força são indestrutíveis.

Uma vez que a meta primordial desta exploração é encontrar minérios raros, é essencial que a nave retorne ao seu planeta de destino somente após

ter coletado todos os minérios disponíveis. Caso contrário, a missão incumbida ao jogador não pode ser considerada como cumprida.

Enfim, por vezes, a localização exata da nave não pode ser determinada devido a problemas na transmissão. Neste caso, o simulador é capaz de oferecer apenas uma aproximação das possíveis coordenadas em que a nave possa estar; cabe ao jogador submeter um programa que resolva a missão independentemente das coordenadas originais. Desta maneira, é de vital importância que comandos condicionais sejam aplicados a fim de definir precisamente qual será a posição da nave ao longo da missão.

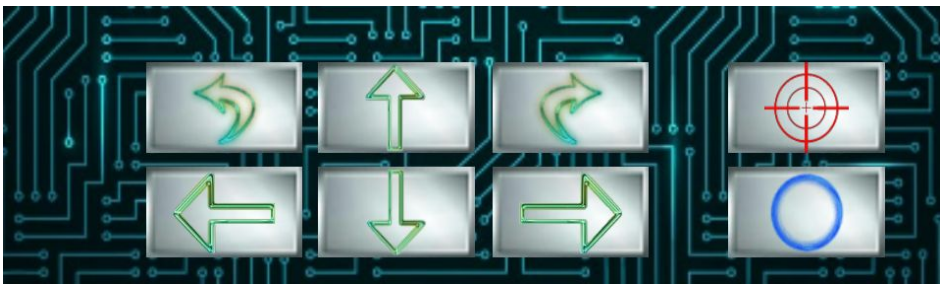


Comandos

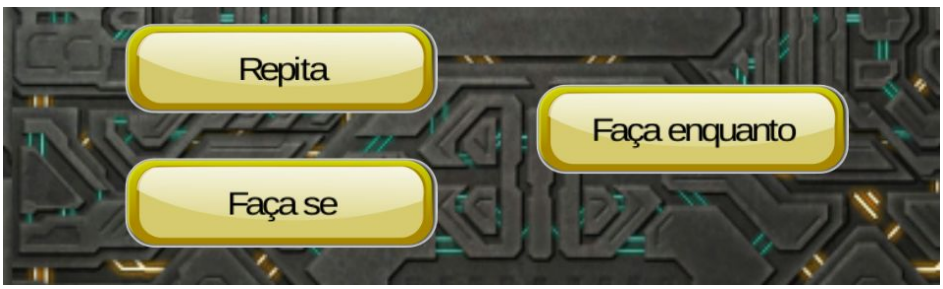
Para completar os níveis de Torre de Controle, o jogador dispõe ao todo de catorze comandos diferentes, organizados em categorias. São oito comandos básicos da nave, três comandos de controle de fluxo e outros três comandos condicionais. No primeiro nível, apenas um comando está disponível, de modo que os outros restantes vão sendo desbloqueados no decorrer dos níveis e no

cumprimento dos módulos. Em certas ocasiões, comandos desbloqueados previamente podem não estar disponíveis para um nível futuro; esta medida tem por finalidade aumentar a dificuldade em níveis avançados ou diminuí-la para ensinar um conceito novo, por intermédio da eliminação de programas indesejados.

Os comandos básicos de manipulação da nave permitem a movimentação nas quatros direções triviais (relativas à orientação da nave), giros de 45° em ambos os sentidos, além de disparos de raios laser e ativação do escudo de energia, úteis para confrontar ameaças espaciais. Cada uma dessas ações ocupa apenas um espaço no painel de comandos.



Os comandos de controle de fluxo são mecanismos de repetição e/ou seleção e atuam como blocos de código. Comandos inseridos no interior do bloco tornam-se parte do bloco que poderá ser executado novamente. Comandos dispostos dentro do bloco são deslocados horizontalmente, com o intuito de simular a identação realizada por editores de texto ao interpretar trechos de código. Assim, esta categoria de comandos é indispensável na medida em que proporciona o reaproveitamento de comandos, ocupando somente dois espaços no painel em troca deste efeito. Podem ser de dois tipos: repetição simples ou repetição condicional. Comandos do primeiro tipo (`Repita`) executam o bloco exatamente o número de vezes delimitado anteriormente. Já para os comandos do segundo tipo, a execução do bloco acontece em função do comando condicional anexado, que pode atuar como critério de execução (`Faça se`) ou de parada (`Faça enquanto`).



Finalmente, os comandos condicionais conectam-se aos comandos de controle de repetição condicional e funcionam como parâmetros que definem o critério de execução ou parada do fluxo. Por funcionarem como complementos, não ocupam espaço adicional no painel. São convenientes para detectar a presença ou ausência de asteroides, campos de força ou minérios raros no quadrante exatamente à frente da nave.

Design do Jogo

Torre de Controle é uma aplicação disponível livremente na Internet[48] e é autossuficiente como método de ensino para introdução à programação. Isto é, embora possua algumas limitações devido ao seu escopo reduzido, não requer qualquer tipo de tutor ou acompanhamento presencial ao longo das aulas. Por um lado, isto aumenta a divulgação e facilita o acesso ao projeto, bastando para isso um computador dispondo de conexão à Internet; por outro lado, assim como ocorre com outros métodos de ensino a distância, isto também prejudica o comprometimento e a retenção de usuários.

Uma das principais preocupações durante o desenvolvimento do jogo foi garantir uma boa experiência do usuário a fim de suprir esta carência em incentivos que um professor presente proveria. Afinal, uma experiência ruim vivenciada pelo aluno no jogo poderia afetar diretamente o seu envolvimento ou até mesmo aumentar a taxa de evasão das aulas. O estresse e frustrações provocados são capazes de ocasionar um abandono prematuro, antes mesmo que o usuário experimente um grau de imersão que o prenda por tempo suficiente para descobrir e usufruir dos demais recursos e funcionalidades que o jogo possa oferecer. Logo, a fim de evitar estas e outras possíveis frustrações, a interface de Torre de Controle foi concebida para ser intuitiva, amigável e responsiva, mantendo-se sempre em foco a simplicidade do sistema como um todo, em conformidade com as principais recomendações difundidas na área de design de interfaces para o usuário.

Já há algum tempo, designers têm tentado facilitar o trabalho de futuros profissionais com a criação de compilados das principais sugestões para se desenvolver interfaces acertadas. Embora algumas recomendações possam ser muito específicas, incompletas, incompatíveis ou até mesmo obsoletas, é importante ter em mente que configuram apenas diretivas, em vez de regras rígidas a serem seguidas. Devem ser interpretadas e adequadas ao projeto em questão, visto que é responsabilidade do designer determinar precisamente as necessidades de seus usuários e os contextos em que seus produtos serão usados[49]. Do contrário, seria possível haver um design único e que satisfizesse as necessidades de todo e qualquer usuário.

Sejam os critérios de avaliação ergonômica de Bastien-Scapin[50] ou o conjunto de regras de ouro[49] de Shneiderman e Plaisant, inúmeros modelos para concepção de interfaces estão disponíveis na literatura especializada. Estes guias práticos têm por objetivo alertar sobre eventuais perigos, erros de concepção e problemas comuns de design, assim como disseminar soluções efetivas, baseadas em conhecimento acumulado ao longo das últimas décadas.

Ainda que estes modelos possam diferir, isto ocorre principalmente por questões de categorização do conteúdo, e não por presença de discrepâncias elementares.

A seguir, listamos os principais princípios de design de interfaces adotados neste projeto, juntamente com exemplos de aplicações destes princípios em Torre de Controle.

Consistência e Homogeneidade

Trata da compatibilidade entre os aspectos visuais da interface e suas funcionalidades, bem como da existência de uma padronização entre elementos similares.

No jogo, isto se traduz pela organização dos comandos em abas no menu; cada aba representa uma categoria e agrupa comandos que ocupam o mesmo espaço no painel e realizam operações de natureza semelhante. Tanto as abas quanto seus respectivos comandos possuem cores características e formatos marcantes, evitando possíveis confusões.

Redução de Carga

Tem por objetivo avaliar o nível de sobrecarga mental sofrida pelo usuário ao manipular a interface e a facilidade de aprendizado do usuário em aprender a utilizá-la. Também mede o tempo e o número mínimo de ações necessários para se cumprir uma tarefa.

Para atender a este critério, optamos por uma interface baseada puramente no cursor, isto é, onde o manuseio de todos os elementos ocorre por meio do arraste de blocos e do pressionamento de botões. Por dispensar o uso do teclado e adotar botões variados no lugar de formulários, esta mecânica é mais intuitiva e fácil de operar.

Segundo afirmam Shneiderman e Plaisant[49],

> "Esta abordagem baseada em manipulação direta é atrativa pois os usuários podem evitar memorizar comandos novos, a chance de erros tipográficos devido ao uso do teclado são reduzidas, e a atenção pode ser mantida no monitor. Frequentemente, os resultados são desempenho acelerado, menor taxa de erros, aprendizado facilitado e maior satisfação. Cursores também são importantes para dispositivos pequenos ou telas imensas fixadas à parede, que fazem dos teclados dispositivos de entrada impraticáveis."

Logo, esta escolha de mecânica também facilita o lançamento futuro de Torre de Controle para sistemas embarcados. Ainda segundo eles, cursores são úteis para desempenhar tarefas tais como seleção, posicionamento, orientação e quantificação linear. Estas funções correspondem às funções presentes em nosso projeto, desde o arraste e posicionamento dos blocos, até a escolha e configuração dos comandos de controle de fluxo.

Ademais, outras plataformas destinadas ao ensino de programação para jovens também se apoiam no emprego de cursores. Além de nossas principais influências, Scratch e Lightbot, First Code Academy - escola particular de ensino de programação para crianças a partir dos seis anos localizada em Hong Kong - adota esta mesma medida. Em aulas semanais de noventa minutos, as crianças aprendem a criar jogos para tecnologias móveis, usando programas baseados em cursores. De acordo com Michelle Sun[7], fundadora da escola, trata-se de uma forma de programação; desta maneira, as crianças não precisam digitar muito, sendo mais intuitivo para que os pequenos aprendam os conceitos-chave e formulem um embasamento teórico.

Do mesmo modo, demais funcionalidades foram implementadas para reduzir a sobrecarga do usuário; missões são reiniciadas automaticamente em caso de falhas, tutoriais e diálogos estão presentes somente quando necessário e a linguagem escolhida é adequada ao público-alvo (infantil e adolescente).

Controle Explícito

Preza pela relação clara entre ações tomadas pelo usuário e o funcionamento e as consequências decorrentes de tais ações. Da mesma forma que o sistema deve satisfazer as exigências do usuário, deve ser possível interromper qualquer sequência ou ação indesejada em curso. Resultados inesperados, dificuldades em obter uma informação e lentidão nos processamentos são exemplos de falhas para este princípio.

No jogo, é o jogador quem possui controle absoluto sobre a nave espacial. O simulador deve ser acionado para que a execução do programa se inicie, podendo ser interrompido a qualquer instante. Botões para abandonar o nível atual também estão sempre presentes, para que a troca de missão seja facilitada. Além disto, todos os tutoriais e diálogos presentes no jogo são elementos adicionais, de maneira que podem ser completamente ignorados, se o usuário assim quiser.

Gestão de Erros

Diz respeito à qualidade dos métodos implementados de detecção, prevenção, correção e recuperação de erros durante a execução do sistema e também à gravidade das consequências plausíveis resultantes desses erros.

Com o intuito de evitar a submissão de programas inválidos à nave, não é possível iniciar uma simulação caso algum comando de controle de fluxo condicional esteja incompleto; analogamente, desde que a simulação tem início, o menu e o painel de comandos tornam-se inativos, inviabilizando a interação com elementos que possam alterar os rumos do jogo. Enfim, os níveis são todos independentes e reinicializados após cada simulação, dificultando que eventuais falhas possam se perpetuar, causando situações desastrosas.

Método de Ensino

Em Torre de Controle, o aprendizado ocorre em ciclos, em que cada elemento novo desbloqueado é apresentado brevemente por nosso mascote-tutor, através de um conjunto de mensagens sucintas. A preocupação aqui foi passar o novo conteúdo de forma resumida, sem sobrecarregar o jogador.

Esta é a técnica sustentada por George Fan em 'How I got my mom to play *Plants vs. Zombies*', palestra apresentada na 22ª edição da Game Developers Conference (GDC). Trata-se da maior e há mais tempo em vigor feira mundial dedicada inteiramente a profissionais da indústria dos jogos. Criada em 1987, acontece todos os anos em São Francisco, onde reúne programadores, ilustradores, designers de jogos e outros para discutir o futuro da indústria.

Fan é designer e criador do jogo *Plants vs. Zombies* e, em seu discurso, explica as razões do imenso sucesso atingido e, em particular, como conseguiram cativar jogadores casuais com um jogo de estratégia, considerando que este gênero costuma ser reservado ao público mais comprometido com este universo. Ele credita a maior parte do sucesso do jogo aos tutoriais e garante que o uso de frases esparsas para comunicar as ideias principais pode aumentar as chances de o jogador lê-las, processá-las e se interessar pelo jogo[51].

> "Usem menos palavras; nos meus jogos, eu sigo a regra:
>
> Deveria haver, no máximo, oito palavras na tela em um dado momento.
>
> Naturalmente, isto é impossível de se atingir, e eu mesmo quebro esta regra às vezes. Mas é uma boa ideia a qual se ater.
>
> Também é importante manter um baixo número de frases durante o tutorial. É necessário descobrir o que precisa ser dito ao jogador e, então, dizer apenas isso."

Torre de Controle é divido em módulos, que por sua vez, são compostos por dez níveis de dificuldade variável e incremental. Em cada ciclo, o primeiro nível é propositalmente fácil, de forma que o jogador possa resolvê-lo rapidamente e demonstrar entendimento. Os níveis seguintes - mais difíceis - misturam o novo elemento com outros elementos do jogo para aumentar o grau de fixação tanto do novo conceito aprendido quanto dos já estudados anteriormente.

A quantidade de níveis que compõem um ciclo oscila de acordo com a complexidade do material em estudo. Tendo acabado um ciclo, o processo é, então, recomeçado, isto é, um novo elemento é incluído e apresentado.

Quando conceitos mais inovadores e complexos são introduzidos, um vídeo tutorial complementar ilustra as explicações de Tião. Assim como ocorre com as mensagens, os vídeos são elucidativos e curtos, para não serem tediosos.

Novamente, Fan afirma que todo texto do enredo deveria ser esclarecedor ou envolvente, e o conteúdo excedente deveria ser removido.

Bombardear o jogador com mensagens irrelevantes, uma após a outra, é arriscar-se a perdê-lo de uma vez por todas[51].

Por outro lado, também empregamos elementos pedagógicos clássicos no jogo Torre de Controle. A formação do conhecimento do usuário através da superação dos desafios propostos a cada nível representa o conceito-chave em que nossa aplicação se baseia para ensinar o raciocínio computacional. É o aluno quem descobre, por meio de experimentação no ambiente que oferecemos, a melhor forma de usar cada comando. O aluno constrói seu aprendizado à medida que utiliza os comandos disponíveis para completar o objetivo do nível corrente, e é a forma como são empregados que, de fato, desenvolve o raciocínio do aluno. Este método de ensino, dito construtivista, foi o que mais se adequou aos nossos propósitos.

Piaget e o Construtivismo

O construtivismo é uma das principais teorias modernas cognitivas e afirma que é a partir de interações entre humanos, suas ideias e suas experiências prévias, que conhecimento é gerado[52]. A teoria construtivista foi embasada nas ideias do psicólogo suíço Jean Piaget, cujos estudos fomentaram grande impacto nas correntes de ensino e pensamento em meados do século XX.

Parte deste impacto se deve à sua insurgência como contraponto ao modelo tradicional de ensino, onde um professor, visto como detentor absoluto do conhecimento, distribui-o aos seus alunos. Essa forma de ensino se assemelha fortemente ao padrão de ensino atual que temos na grande maioria das escolas brasileiras. Fernando Becker, Professor de Psicologia da Educação da Faculdade de Educação da Universidade Federal do Rio Grande do Sul, declara que a escola, como instituição generalizada, é motivo de insatisfação justamente pela persistência neste modelo antiquado:

> "[...] insatisfação com um sistema educacional que teima em continuar essa forma particular de transmissão que é a Escola, que consiste em fazer repetir, recitar, aprender, ensinar o que já está pronto, em vez de fazer, agir, operar, criar, construir a partir da realidade vivida por alunos e professores, isto é, pela sociedade..."[53]

Segundo Piaget, o conhecimento não é inerente ao sujeito aprendiz, ou seja, não nasce pronto com o sujeito e simplesmente precisa ser despertado; e tampouco resulta completamente de um processo passivo de observação e absorção do meio, como dita o modelo tradicional. Para ele, o conhecimento

surge a partir da interação das estruturas cognitivas inerentes ao sujeito com o meio em que se está inserido[54].

Piaget argumenta ainda que um esquema mental não é uma mera cópia de carbono de algo exterior à mente, mas uma interpretação mental de informações recebidas e que foi construída ativamente. O desenvolvimento cognitivo envolve a adaptação do conhecimento atual a novas informações, o que implica ser impossível transmitir conhecimento para estudantes de forma direta, através da simples memorização de fatos e procedimentos.[2]

O papel fundamental do professor (ou plataforma de ensino) deve ser o de dialogar com o aluno, para que ideias sejam debatidas e reformuladas, construindo juntos o conhecimento, até que atenda aos objetivos pedagógicos definidos. Assim, para que o aprendizado seja incentivado, deve-se estimular a construção do conhecimento não apenas através de explicações, mas também por meio da exibição de problemas instigantes e atividades desafiantes.

Para cumprir esta proposta, Torre de Controle foi desenvolvida recorrendo-se à inclusão de elementos diversos, ao recurso de gamificação e tomando como base a forma como aplicações construtivistas ensinam crianças. *Scratch*, por exemplo, é um ambiente de aprendizado computacional fundamentado na teoria construtivista; já foi testado e é amplamente utilizado no ensino infantil com graus variados de sucesso, como vimos na seção *Influências*.

No decorrer deste processo, saber como encarar as falhas é indispensável. Conforme adverte Luis de França Ferreira, em seu texto "Ambiente de Aprendizagem Construtivista"[54]:

> “Em uma abordagem construtivista, o erro é uma importante fonte de aprendizagem, o aprendiz deve sempre questionar-se sobre as consequências de suas atitudes e, a partir de seus erros ou acertos, ir construindo seus conceitos, ao invés de servir apenas para verificar o quanto do que foi repassado para o aluno foi realmente assimilado, como é comum nas práticas empiristas. Neste contexto, a forma e a importância da avaliação mudam completamente, em relação às práticas convencionais.”

Em Torre de Controle, essa abordagem é garantida na medida em que não há limites para número de erros por nível, nem por módulo, como costuma ocorrer em outros jogos. Neste ambiente educacional, o aluno é livre para elaborar e experimentar novas combinações de comandos, e aprimorar suas técnicas, sem correr riscos de ser penalizado por isto.

Teoria do Desenvolvimento Cognitivo

Após conduzir estudos com crianças, baseado em suas observações, Piaget desenvolveu uma teoria do desenvolvimento intelectual, particularmente voltado à progressão infantil. Em sua visão, crianças são como cientistas mirins, uma vez que exploram o mundo ao seu redor com vivacidade, em busca de respostas. Entretanto, acreditava que a mente infantil não se resumia tão somente a uma versão menor da mente adulta, uma tábula rasa pronta para ser preenchida com os tópicos que quisermos ensinar. Seus estudos mostraram que crianças não são menos inteligentes que adultos, elas apenas pensam de maneira diferente. Enquanto crescem e interagem com o mundo, crianças reorganizam e modelam seus pensamentos até começarem a raciocinar como adultos[55].

Segundo ele, toda criança atravessa uma série de quatro estágios críticos à medida que desenvolve seus processos cognitivos. Todas passam pelos mesmos estágio, na mesma ordem - ainda que algumas possam nunca atingir os últimos estágios[52]. Cada estágio caracteriza-se por alterações cognitivas e o aprendizado de novas habilidades, que explicam a forma como as crianças entendem o mundo. Desta forma, ao progredir de um estágio para outro, a criança não apenas acumula mais conhecimento, como também ocorrem alterações fundamentais no modo de ela enxergar e interpretar o mundo.

Do nascimento até os dois anos, o bebê se encontra no estágio sensório-motor. Neste estágio, desenvolve-se o domínio do próprio corpo e do sistema sensorial, que é amplamente usado como ferramenta de manipulação e experimentação do mundo. Nesta fase, o bebê cria o senso de identidade; aprende a diferenciar sua existência da das outras pessoas e dos objetos. Todavia, o sistema sensorial é tão determinante nesta idade, que a própria existência dos objetos é determinada por sua capacidade de detecção. O entendimento que objetos continuam existindo ainda que não estejam ao seu alcance é a principal conquista desta fase.

Durante o estágio pré-operatório, de dois até seis ou setes anos, crianças aprendem a linguagem e a se comunicar. Crianças neste ponto ainda não dominam a lógica e têm dificuldades em lidar com propriedades constantes e pensar sob outras perspectivas. Ao final deste estágio, elas compreendem que seus próprios pensamentos podem não ser de conhecimento comum.

Crianças no estágio operatório-concreto (dos 7 aos 11 anos) pensam com uma lógica mais aguçada, porém, ainda de uma forma bastante concreta. Lidar com conceitos muito abstratos ou hipotéticos pode ser um desafio. Aqui, desenvolvem uma maior preocupação com os sentimentos e pensamentos

alheios e passam a entender que ideias e opiniões pessoais podem divergir das dos demais.

O último estágio, denominado operatório-formal, se inicia a partir dos onze anos. Caracteriza-se pela consolidação do pensamento lógico e a compreensão de ideias abstratas. Crianças nesta fase também são capazes de pensar em várias possibilidades para um mesmo cenário, raciocinar a respeito do futuro e preocupar-se com questões filosóficas.

Por fim, é importante frisar que as faixas etárias de cada estágio podem variar de um indivíduo para outro, de modo que as idades indicadas são estimativas representando o intervalo que uma criança, em média, prevalece no respectivo estágio. Devido a isto, outros estudiosos preferem não falar em estágios bem definidos[52]; sendo o desenvolvimento cognitivo um processo contínuo, aproximativas ao mundo discreto sempre podem levar a erros.

Ainda assim, a teoria piagetiana do desenvolvimento infantil teve um grande impacto para a educação e para as técnicas pedagógicas de ensino. Embora Piaget tenha se dedicado a desvendar o processo de aprendizagem, em vez de criar formas de aprimorá-lo, seu trabalho sugere que crianças deveriam estudar um conteúdo compatível com sua idade, ou melhor, com o estágio cognitivo em que se encontram. Logo, considerando que é no último estágio que a criança média possui suas faculdades lógicas consolidadas e os mecanismos de abstração em evolução, a escolha de faixa etária para uso de nossa aplicação é justificada.

O outro lado

Em seu artigo, intitulado "An alternative view on why, when and how computers should be used in education"[56], Setzer e Monke se propõem a analisar a questão do uso de computadores como ferramenta educacional para jovens e as consequências desta medida. Segundo eles, grande parcela dos trabalhos anteriores preferiram focar em temas como a eficácia da tecnologia na educação e o grau de profundidade do currículo escolar com respeito à computação. Fatores como a idade apropriada para o primeiro contato com computadores e os efeitos benéficos e maléficos decorrentes de seu uso para o indivíduo e a sociedade costumam ser negligenciados. Motivados pela indispensabilidade do ensino do funcionamento de máquinas e fenômenos como meio de instigar e satisfazer a curiosidade humana por compreensão (que curiosamente coincidem com parte das motivações expostas neste trabalho), os autores denunciam como o estudo da computação nas escolas não cumpre o seu papel; é superficial, e limita-se apenas ao emprego de softwares e aplicações para desempenhar certas tarefas mais eficientemente.

Posteriormente, enaltecem no mesmo artigo a natureza do computador como máquina abstrata para manipulação de dados; em sua visão, trata-se de uma máquina matemática pois opera sob linguagens e formalismos matemáticos. Em seguida, começam a indagar a respeito da complexidade em se dominar uma máquina nessas condições.

> "Já que o computador exige um tipo de pensamento e linguagem formais, matemáticos, como caracterizamos anteriormente, poderíamos formular a seguinte pergunta: quando crianças ou jovens devem começar a exercer esse tipo de pensamento e linguagem? Se lembrarmos o que foi dito, isto é, que a atividade de programar um computador ou usar um programa qualquer (necessariamente, por meio de uma linguagem de comandos) é análoga à prova de teoremas da matemática, poderemos afirmar o seguinte: a idade adequada é a mesma em que se deseja que os jovens comecem a provar teoremas."

E concluem que "qualquer uso de computadores antes do ensino médio, isto é, mais ou menos aos quinze anos, é prejudicial à criança ou jovem". A definição desta idade fundamenta-se nas ideias da chamada Pegadogia Walfdorf, criada em 1919 pelo pensador austríaco Rudol Steiner. Steiner acreditava que os jovens passam por três etapas de setes anos cada ao longo de seu desenvolvimento, ao invés de quatros estágios de duração variável. Neste ponto, é importante salientar que as ideias de Steiner e Piaget diferem somente quanto aos seus propósitos, já que ambas as categorizações são bem semelhantes. Embora Steiner também tenha dividido o desenvolvimento cognitivo humano em estágios, diferentemente de Piaget, não concentrou seus estudos somente nos processos psicológicos compreendidos, mas também na formulação e aplicação de técnicas pedagógicas condizentes.

Desta forma, Setzer e Monke acreditam que, ao usar um computador, a criança seria obrigada a exercer um tipo de pensamento que deveria empregar somente em idade bem mais avançada. E é neste ponto que este trabalho discorda do primeiro por uma série de razões.

Primeiramente, porque o emprego de ferramentas tecnológicas não pressupõe e nem requer necessariamente o conhecimento do funcionamento interno das máquinas e fenômenos envolvidos. Ainda que a estruturação dos currículos escolares deva se adequar de fato à capacidade cognitiva da criança[2][55] e considerar o ensino e desenvolvimento do raciocínio computacional[4][6][13], é totalmente possível desassociar os mecanismos e operações internas de um computador de seus efeitos produzidos. Afinal, para um expectador, qual é a diferença entre ouvir uma música no *Spotify*, ver um vídeo no *Youtube,* assistir a um filme no *Netflix,* ler notícias no site da *BBC* ou consumir estes mesmos conteúdos através das mídias tradicionais: rádio,

jornal, televisão, etc.? Esta perspectiva fica ainda mais notória quando consideramos plataformas especializadas em tarefas deste tipo, tais como leitores digitais, dedicados a simular livros.

Ao longo do artigo, Setzer e Monke criticam as principais empreitadas relativas ao emprego de computadores na educação. Segundo eles, lidar com uma máquina matemática requer a compreensão de linguagens formais, forçando os usuários a exercitarem um tipo de raciocínio lógico-simbólico impróprio para menores de quinze anos, aproximadamente. Entretanto, jovens nesta faixa etária já encontram-se no estágio operatório-formal, mesmo se desconsiderarmos a crítica por parte de trabalhos recentes de que Piaget tenha subestimado a capacidade infantil em sua classificação[52]. Alguns testes da época foram apontados, por vezes, como sendo muito confusos ou difíceis de assimilar.

Em terceiro lugar, a área de design tem por principal objetivo a concepção de produtos usáveis de alta qualidade. Preza pela redução de carga mental com o objetivo de facilitar o aprendizado por parte de usuários novos e pela simplicidade na mecânica para facilitar a manipulação por parte de usuários habituais. Todavia, este processo é mais complexo do que pode parecer, como confirmam Shneiderman e Plaisant[49]:

> "Usuários não são sobrecarregados pela interface [eficaz] e são capazes de predizer o que acontecerá em resposta a cada uma de suas ações. Quando um sistema interativo é bem desenvolvido, a interface quase desaparece, permitindo aos usuários concentrar-se em seu trabalho, pesquisa ou entretenimento. Criar um ambiente em que tarefas sejam desempenhadas quase sem esforço e usuários fiquem 'imersos' requer um alto nível de trabalho duro por parte do designer."

Se o fato de lidar com computadores gere uma intelectualização forçada e prejudicial, como apontado por Setzer e Monke, talvez isto se deva a falhas na concepção e no design da aplicação em questão. Além disso, o desenvolvimento de novas tecnologias tende a ampliar a experiência do usuário com produtos mais realistas, gerando novas possibilidades de interação, mais ricas e mais naturais. Celulares e outros dispositivos inteligentes produzidos atualmente costumam ser sensíveis ao toque enquanto outras tecnologias - para rastreamento ocular, por exemplo - estão sendo pesquisadas com o propósito de possibilitar a construção de interfaces muito mais simples e intuitivas futuramente.

Enfim, com respeito ao contexto de jogos eletrônicos, é curioso notar que, em alguns casos, a simulação do jogo pode ser feita sem meios digitais, assim como muitas vezes é possível replicar o processamento de informações feito por um computador mentalmente ou usando apenas lápis e papel[56].

Embora este método possa significar a perda dos benefícios da gamificação[26], comprometa a retenção e, consequentemente, a aprendizagem do usuário, ele poderia ser aplicado a Torre de Controle, por exemplo. Bastaria, para tal, representar o nível e os elementos de jogo através de um tabuleiro divido em quadrantes e um conjunto de peças previamente posicionadas, semelhante ao jogo Batalha Naval[57]. Os comandos poderiam ser igualmente representados por peças, a aleatorização de certos elementos poderia ser calculada com a ajuda de dados e a simulação em si seguiria o conjunto de regras bem definidas exposto no jogo. E este jogo de tabuleiro seria equivalente e tão prejudicial quanto o jogo eletrônico original.

De fato, esta abordagem já existe e é amplamente utilizada mediante testes de usabilidade via protótipos. Aplicados em laboratórios específicos, onde todas as ações e reações efetuadas são registradas para futura análise, usuários são incumbidos de cumprir uma lista de tarefas a que o programa se destina[49]. Em fases iniciais, o sistema final almejado pode ser simulado com desenhos em papel, papelões moldados e outros artigos de escritório. Enquanto o usuário reage frente ao design testado, o condutor do teste simula as operações da máquina em função do comportamento observado. Assim, testes de usabilidade provém recomendações para mudanças, confirmações quanto a erros e acertos do projeto e descomplicações em realocação de recursos. São úteis para validar projetos de grande porte e alto risco ou aplicações ousadas, que não se assemelhem a outros produtos existentes, que poderiam ser usados como referência.

Ainda que não concordemos com a visão destes autores, frisamos a importância destes trabalhos, na medida em que põem em voga a questão do uso de dispositivos tecnológicos por crianças. Pois, conforme evidenciam Setzer e Monk, crianças não possuem maturidade nem grau de autocontrole suficientes para regular o tempo de exposição a computadores, por exemplo. Ademais, com o fenômeno da globalização acelerada e, em particular após o surgimento e a popularização das redes sociais, diminuíram-se as distâncias, facilitou-se o acesso à informação e a capacidade de comunicação, tornando a vida muito mais prática. Entretanto, no atual mundo digital, a falsa sensação de impunidade e o anonimato relativo propiciado por estas plataformas tecnológicas, também potencializaram os perigos no âmbito virtual, dando origem à nova categoria de crimes digitais.

Finalmente, é importante desmistificar a crença de que a brutalidade presente em jogos e filmes possa aumentar as taxas de violência na sociedade. De fato, estudos recentes mostraram que não existe relação causal entre a violência da ficção com a violência da vida real[58]. Embora o conteúdo de determinadas mídias possa não ser adequado para crianças - ou mesmo adolescentes - em certas idades, esta questão já é contemplada pelo recurso da

classificação indicativa. Assim, buscando o emprego seguro da tecnologia, desencorajamos o uso de dispositivos inteligentes por crianças, especialmente quando conectados a Internet, sem a devida supervisão de seus pais ou responsáveis. Da mesma forma como ocorre com outras mídias, são eles os responsáveis pelo material e pelas informações consumidas pelos pequenos, e por isso acreditamos que, tratando-se de conteúdos digitais, não deveria ser diferente.

Resultados Obtidos

Na concepção do projeto, havíamos idealizado uma etapa conclusiva de testes com crianças da faixa etária a que se destina Torre de Controle, ou seja, de doze a quinze anos. Entretanto, não foi possível encontrar um contingente significativo, com essas características, a tempo de organizar uma rodada de avaliação. Por isso, alteramos a proposta do teste empírico de nossa aplicação e convidamos especialistas em design de jogos a darem suas críticas e sugestões sobre nossa plataforma pessoalmente.

Por volta do mês de outubro, estabelecemos que Torre de Controle atingira um estágio de desenvolvimento satisfatório para a realização dessa rodada de avaliações. Assim, por precaução, preparamos nosso sistema e realizamos uma nova bateria de testes de todos os tipos em busca de quaisquer erros e imperfeições que comprometessem a jogabilidade, assegurando-nos de que o sistema encontrava-se mesmo em um estágio estável.

Exibimos Torre de Controle a cinco designers de jogos, em momentos separados. Nenhum deles teve algum tipo de experiência prévia com o jogo, nem tampouco uma explicação ou interferência de nossa parte antes ou durante o processo de teste. Esta abordagem teve o intuito de preservar o primeiro contato do designer com nossa aplicação, sem que nossas opiniões ou expectativas corrompessem a avaliação deles. A observação do comportamento natural dos especialistas sem nossa intervenção nos ajudou a entender quais pontos de nossa interface são intuitivos o bastante para serem considerados autossuficientes - característica bastante desejada devido às nossas preocupações em promover o ensino a distância.

A seguir descrevemos as opiniões dos designers convidados para a rodada de testes.

Primeiramente, foi destacado que Torre de Controle é louvável em sua iniciativa educacional, e que o jogo acerta no apelo temático ao público-alvo que escolhemos. Teve destaque especial, nesta questão, o instrutor e mascote do jogo, o porquinho *Tião*. Este ponto foi levantado por diversos designers, separadamente.

Também foi denotado que o balanceamento do jogo foi bem elaborado no primeiro dos módulos de Torre de Controle. Os designers afirmaram que a introdução de funcionalidades da nave a intervalos regulares neste módulo é indicador de qualidade em design de jogos. Por motivo semelhante, foi dito que o segundo módulo é tido como fonte de dispersão de público. Escolhemos, por motivos de reforço pedagógico, apresentar diversos desafios referentes ao uso

do comando `Repita` neste módulo, em detrimento da introdução de funcionalidades novas, as quais entram em cena somente no terceiro módulo. Entretanto, o resultado final foi considerado parcialmente maçante pelo caráter repetitivo dos desafios.

A principal funcionalidade introduzida no terceiro módulo, a aleatorização da posição inicial da nave, foi considerada pela maioria dos designers como fundamental para a aplicação. Segundo eles, essa funcionalidade é a garantia que o jogo não é somente sobre repetição e movimentação, mas sobre a concepção e abstração de soluções para problemas de natureza genérica. Essa foi, de fato, nossa motivação principal na adição dessa funcionalidade, e acreditamos que ela cumpre bem seu papel em Torre de Controle: assegurar que o usuário deva generalizar seus programas e, assim, compreender melhor como solucionar problemas.

Também foi levantado que não é possível enxergar muito bem a grade que divide o universo em quadrantes, e que a princípio não se compreende bem que a nave segue unidades discretas de movimentação. Acreditamos que isso ocorre principalmente em monitores de baixa resolução, e a resposta a esse problema seria simplesmente engrossar as linhas que compõem essa grade, de forma a facilitar a visualização desse *asset*.

Alguns designers também mencionaram o fato de que a arte de Torre de Controle é um tanto desencontrada e sem padronização. Cada comando, aba ou mesmo o mascote mistura elementos de escolas diferentes, com estilos desiguais, o que provoca uma pequena sensação de estranhamento. Sabemos, é claro, que apesar de ser fundamental em aplicações comerciais, o estilo visual do jogo não é o ponto que consideramos mais importante, principalmente por não possuirmos formação artística. Já tínhamos imaginado que críticas dessa natureza surgiriam mais cedo ou mais tarde, devido à decisão de focar nossos esforços na melhoria da mecânica do jogo, assim como em seu conteúdo pedagógico, e não em aspectos visuais.

Foi dito que os vídeo-tutoriais, apresentados juntamente com as novas funcionalidades do programa, garantiram que o usuário não tivesse que se ater aos textos introdutórios nas telas anteriores ao jogo, evitando que o jogador considere esses momentos de aprofundamento pedagógico monótonos. Era esse, de fato, o objetivo que tínhamos em mente quando decidimos incluir vídeos curtos juntamente com a presença do mascote.

Entre os comentários que recebemos dos designers, acreditamos que o que mais fez diferença foi a menção a um pequeno contador que indicasse quantos comandos ainda podem ser adicionados ao painel. Acreditamos que essa funcionalidade somente tem a acrescentar à experiência do usuário final, por deixar claro que existe e especificar, em cada nível, o número de comandos que ainda podem ser utilizados. Implementamos essa funcionalidade depois

dessa rodada de entrevistas, e ela já pode ser encontrada na versão final de Torre de Controle[48].

Melhorias Futuras

Torre de Controle, como ferramenta educativa, ainda tem espaço para a a inserção de melhorias, como qualquer outro projeto de grande porte. Nesta seção, explicamos alguns pontos em aberto; funcionalidades que não estavam previstas dentro das especificações iniciais do projeto, mas que consideramos interessantes.

Em primeiro lugar, gostaríamos de ter desenvolvido uma ferramenta de depuração para a lista de comandos que é elaborada pelo usuário. Seria interessante exibir um mecanismo onde o jogador conseguisse avançar a simulação passo a passo, conseguindo assim simular o programa na medida em que o mesmo é executado. Acreditamos que esta funcionalidade pode ajudar a compreender melhor o funcionamento da simulação e, por extensão, da programação e do raciocínio computacional. Esta ferramenta foi pensada somente nos estágios mais avançados de implementação do jogo, e acreditamos que seria dispendioso alterar a estrutura interna do código para acomodá-la na fase de desenvolvimento em que nos encontrávamos. Por ser muito significativa e atender aos interesses pedagógicos do usuário, esta é uma das ferramentas que certamente merece atenção e foco de desenvolvimento em versões futuras de Torre de Controle.

Ademais, acreditamos que a interface de arraste dos comandos em nosso jogo é muito compatível com dispositivos móveis modernos, como *tablets* e *smartphones*, que adotam esse padrão como forma de interação com o usuário. O formato da tela é propício à jogabilidade de nossa aplicação e acreditamos que a mobilidade e grande presença desses aparelhos pode favorecer a retenção de usuários a médio prazo. Escolhemos *Unity* como framework de desenvolvimento principalmente por fatores já citados em seções anteriores, mas vale lembrar que essa ferramenta é extremamente versátil em sua capacidade de gerar arquivos executáveis, e portanto não é complicado exportar Torre de Controle para outras plataformas.

Tambpem gostaríamos de lançar futuramente nosso jogo nas três maiores lojas de aplicativos modernas, *AppStore*[59], *Google Play*[60] e *MicrosoftStore*[61], além da versão atualmente disponível para navegador. Para isso, acreditamos que o jogo precise ser repensado em termos de interface de usuário, de forma que ele continue intuitivo mesmo com essa mudança de paradigma.

Não tivemos desde o princípio a intenção de transformar nossa aplicação em um serviço multiplataformas e acreditamos que pequenos inconvenientes precisem ser ajustados antes de exportar o jogo para outros

meios; por exemplo, a capacidade de multi-toque dos smartphones certamente precisa de algum tratamento especial antes de interagir com os blocos de comandos de nosso jogo. Apesar desses pequenos consertos, em sua maioria cosméticos, acreditamos que é possível transformar nosso jogo para navegador em uma aplicação para dispositivos móveis de forma bem-sucedida.

Parte Subjetiva

Desafios e Frustrações

Implementar projetos de grande porte é sempre uma tarefa desafiadora. Desde a organização interna do código e das classes do programa, que necessitaram de muitas refatorações ao longo do desenvolvimento, até os problemas que encontramos na implementação das funcionalidades mais complexas, sempre nos deparamos com desafios que nos estimularam. Vários desse desafios foram superados com esforço e inventividade, além do uso apropriado das ferramentas que adquirimos ao longo de nossa formação acadêmica.

Alguns dos desafios que encontramos, entretanto, fugiam das nossas áreas de experiência pessoal. É isso que ocorreu no quesito artístico do jogo, tanto sonoro quanto visual. Não temos nenhum tipo de educação formal quanto à produção de recursos visuais, e decidimos desde o princípio investir nas questões de programação e lógica do projeto ao invés de trabalhar na apresentação do jogo. É claro, essa decisão tem suas inconveniências. A aplicação final não ficou à altura de nossas expectativas quanto ao seu aspecto visual.

Acreditamos que a maior frustração que sofremos se deve a um problema conhecido no arcabouço do *Unity*, para o qual não encontramos solução satisfatória. Uma das funcionalidades do projeto é a deleção de comandos por meio da lixeira, que detecta colisão através de um elemento *collider*, componente do *Unity* que define uma caixa ao redor de um *Game Object*. Descobrimos um problema do Unity relativo ao posicionamento do *collider* da lixeira que, entre duas execuções do programa, muda de lugar de forma automática, se não for reposicionado manualmente antes da execução do programa. A mudança de posicionamento do colisor da lixeira acaba inutilizando a funcionalidade de deleção, porque não é possível saber em que lugar o colisor se encontra depois que o bug ocorre.

Esse problema foi resolvido de forma não satisfatória em relação aos padrões de código que esperamos manter no desenvolvimento do projeto, mas uma correção melhor não pôde ser apresentada por nós, uma vez que a introdução do bug se deve ao *Unity* em si. O que fizemos para solucionar o problema imediato é reposicionar via script a posição do colisor para o mesmo lugar que a imagem da lixeira, depois de ele ser automaticamente reposicionado de forma errônea. Isso garante que a caixa de colisão e a sua respectiva imagem casem perfeitamente na mesma posição, independente do reposicionamento problemático do arcabouço.

Também nos deparamos com um problema no sistema de versionamento *Git* que utilizamos para desenvolver Torre de Controle, como explicado brevemente na seção de *Tecnologias*. Os recursos que a engine *Unity* gera para o desenvolvimento de jogos são, por padrão, binários, e o *Git* não lida bem quando versões diferentes dos arquivos binários conflitam. Os conflitos nos arquivos de texto são resolvidos manualmente, mas essa tarefa não é possível e nem praticável quando somente se dispõe de editores de texto para resolução de conflitos em binários.

Nossa grande frustração, nesse caso, foi descobrir nos estágios finais da implementação de Torre de Controle que *Unity* oferece suporte a serialização dos arquivos binários, ou seja, conversão para texto, de seus recursos internos. Entretanto, esse suporte não vem por padrão na criação de um projeto, sendo ainda necessário uma versão específica do Unity para que esse suporte seja oferecido.

Os arquivos serializados de Unity são dispostos no formato *.yaml*, um formato de serialização de dados que é legível para humanos e semelhante a XML[62]. Dada a apresentação dos dados no formato texto, seria possível resolver os conflitos nos recursos manualmente e, portanto, acelerar o processo de desenvolvimento da aplicação por meio da manipulação paralela das cenas do projeto, já que nos revezávamos na edição de certos arquivos do Unity.

Citando agora os desafios encontrados, acreditamos que um deles foi a implementação do algoritmo de interpretação da lista de comandos. Grande parte de nossos esforços foram focados no desenvolvimento dos algoritmos que são executados no interpretador do jogo. Uma parte considerável do tempo foi dispendido nas refatorações que garantiram melhoria da legibilidade de código.

Outro desafio que pôs à prova nossa capacidade de solução de problemas em conjunto foi o arrasto dos comandos que compõem os programas dentro de nossa aplicação. *Unity*, por ser um motor de jogos 3D, não apresenta um gerenciador de cliques como outros motores de jogos bidimensionais, e grande parte do tratamento do arraste dos comandos foi feito manualmente, através da inserção de gatilhos e variáveis booleanas de controle nas classes que representam os diversos comandos do jogo.

Os comandos de controle, como `Repita` e `Faça se`, que movimentam todo o seu escopo em conjunto quando arrastados, foram especialmente trabalhosos durante sua implementação. Muitos testes e retoques foram feitos antes que ficássemos satisfeitos com o resultado final.

Pretendíamos, desde a fase de concepção do jogo, adicionar uma base de dados online que incrementasse a jogabilidade de Torre de Controle. De acordo

com nossas expectativas, essa base armazenaria o progresso do jogador à medida que ele avançasse pelos níveis e módulos do jogo. Além disso, poderíamos observar com atenção quais fases foram completadas, ou ainda, descobrir quantas tentativas foram necessárias para vencer os desafios que planejamos. Assim, poderíamos obter uma noção mais aguçada sobre o balanceamento dos níveis elaborados ao extrair estatísticas relevantes para o design do jogo, provenientes dos dados armazenados neste banco. Entretanto, principalmente pela ausência de tempo e pela complexidade da inserção do banco de dados online dentro de nossa aplicação, essa ideia foi abandonada.

Disciplinas Relevantes

Ao longo do desenvolvimento desse projeto, percebemos a grande influência de certas matérias da grade curricular do Bacharelado em Ciência da Computação. Antes de listá-las, é importante ressaltar que todas tiveram seus variados graus de importância em nossa formação acadêmica, e que o amadurecimento intelectual necessário para a composição deste trabalho vem não somente da completude de uma grade de ensino, mas de todo o caminho que trilhamos até completá-la.

Abaixo estão listadas as matérias que consideramos mais pertinentes ao desenvolvimento do projeto, com uma explicação sucinta, ressaltando as relevâncias de cada uma.

MAC0332 Engenharia de Software

Além de introduzir princípios de desenvolvimento importantes, como os padrões de projeto que utilizamos, Engenharia de Software também nos ensinou a conduzir projetos de grande porte de maneira organizada e eficiente, a partir dos fundamentos de Kanban e metodologias de acompanhamento de projeto. Outro conceito utilizado, que foi aprendido nesta matéria, foi o desenvolvimento incremental, onde novas funcionalidades vão sendo adicionadas aos poucos e, preferencialmente, de acordo com a sua urgência, isto é, o valor que agregam ao projeto e a importância que têm para a aplicação final.

MAC0316 Conceitos Fundamentais de Linguagens de Programação

Conceitos Fundamentais de Linguagens de Programação nos ajudou ativamente a definir os comandos que o usuário poderia utilizar no jogo e quais seriam suficientes para expressar toda a riqueza de instruções que temos em linguagens de programação sofisticadas. Além disso, moldar nosso interpretador de comandos foi uma tarefa que tomou grande parte do desenvolvimento, e que só foi possível com os ensinamentos que tivemos nesta disciplina.

MAC0446 Princípios de Interação Humano-Computador

Disciplina que nos orientou tanto no processo de elaboração da interface gráfica, disposição e concepção dos menus e painéis quanto no design dos níveis e o balanceamento da dificuldade destes. Suas diretivas serviram de ferramenta para o desenvolvimento de uma aplicação mais simples, concisa, intuitiva, e portanto mais eficiente para o usuário do programa final.

MAC0327 Desafios de Programação

A matéria de desafios de programação inspirou o design das fases de nosso jogo, onde gostaríamos de introduzir conceitos novos a partir de um problema que o usuário deveria resolver com os comandos que a nave pudesse executar. Este curso ainda foi responsável por nos ensinar o valor da persistência no desenvolvimento de um algoritmo (pois, raramente se consegue um resultado positivo na primeira tentativa), além de métodos alternativos na forma de atacar os problemas que tivemos de enfrentar.

MAC0211 e **MAC0242** Laboratório de Programação I e II

Laboratório de Programação I foi importante para o aprendizado de desenvolvimento em equipe; foi a primeira matéria a trabalhar esse aspecto de nossa formação. Já em Laboratório de Programação II, tivemos uma ideia mais refinada sobre desenvolvimento de projetos de grande porte, sobre o uso do paradigma de programação Orientada a Objetos - fundamental no decorrer de nosso projeto - e, mais especificamente, no desenvolvimento de um jogo. Ademais, tal projeto pode ser considerado uma prévia em pequena escala que tivemos do que seria o Trabalho de Formatura Supervisionado.

MAC0121 Princípios de Desenvolvimento de Algoritmos

Princípios de Desenvolvimento de Algoritmos toma um papel fundamental em qualquer projeto que possua alguma estrutura mais complexa do que uma pilha. De fato, essa matéria foi relevante para o desenvolvimento do nosso interpretador de comandos, assim como foi bastante utilizada na concepção do sistema como um todo.

Apreciação Pessoal e Crítica

Esta seção contém a visão pessoal de cada membro da equipe sobre a produção deste trabalho como um todo, o impacto da aplicação final e as experiências de se trabalhar em grupo utilizando os conhecimentos adquiridos ao longo de todo o curso.

Daniel Paulino Alves

Quem lê a versão final desta monografia sequer imagina que grande parte do trabalho não foi documentada. Muitas das reuniões se deveram simplesmente à definição do tema. O trabalho, que originalmente era para ser de ensino de geometria devido à nossa grande empolgação com a disciplina *Geometria Computacional*, mudou bastante de lá para cá. Ainda assim, fico muito contente pela escolha de nosso tema. Hoje vejo como foi acertada.

Torre de Controle me permitiu conhecer um pouco mais das áreas de ensino e psicologia, que particularmente me interessam. Por ser estudante em um instituto de matemática, as influências acadêmicas recebidas de ciências tipicamente humanas, como essas, infelizmente costumam ocorrer apenas por causa das disciplinas optativas livres, que são poucas. A interdisciplinaridade tão prezada pelos vestibulares dificilmente ultrapassa as paredes dos institutos.

Também confesso que termos abandonado a proposta inicial dos testes com crianças em ambiente pedagógico, por dificuldades burocráticas e temporais, foi bem decepcionante. Afinal de contas, seria bastante gratificante observar nosso trabalho sendo validado e efetivamente usado para fins educativos, conforme visualizamos inicialmente. E honrar um dos três pilares que sustentam a universidade: a extensão. Embora seja o mais esquecido, é sua presença que enfim nos lembra porquê e para quê a universidade existe. E que nem todo conhecimento se faz em sala de aula.

Pessoalmente, vejo este trabalho justamente como uma tentativa de devolver à sociedade o que me foi oferecido e contribuir de alguma forma para o seu avanço. Um esforço em mudar a realidade de alguém, ainda que só uma parcela de como a Universidade de São Paulo mudou a minha.

Felipe Yamaguti

O desenvolvimento e concepção de Torre de Controle foi, para mim, uma experiência única dentro do ambiente acadêmico. Tive a oportunidade de, trabalhando em equipe, desenvolver um software de grande porte cujo objetivo é, segundo minha própria opinião, de grande relevância atualmente. Observei em primeira mão como o desenvolvimento gradual e iterativo deste projeto aconteceu, quais foram as decisões de projeto que tomamos, e de que forma esse processo afetou o produto final. Eu acredito que essa vivência desenvolve uma habilidade pouco estimulada dentro da graduação em Ciência da Computação, o trabalho prático e em equipe.

Achei muito interessante examinar como as frequentes reuniões que fazíamos entre os membros da equipe e o orientador norteavam o desenvolvimento das funcionalidades e encaminhavam o processo de idealização do jogo. Essas reuniões foram, no meu entendimento, fundamentais para a caracterização deste projeto como hoje ele é.

Tenho confiança que o jogo, como produto final, serve ao propósito pelo qual o idealizamos, e espero, em seu devido tempo, que a área de Ciência da Computação possa se beneficiar de muitos outros projetos como este.

Rafael Batista Carmo

Foi muito interessante desenvolver esta plataforma, primeiramente por ser algo que, na minha opinião, pode ajudar a difundir a comunidade da Ciência da Computação, na qual estou hoje em dia inserido. Além disso, trabalhar em grupo é sempre uma experiência que agrega valores e conhecimentos quando bem efetuada. Com certeza aprendemos a trabalhar com uma equipe distribuída, através de reuniões e com a ajuda do *Trello*, o que já é uma realidade na minha vida de trabalho atual, ou seja, pude trazer o aprendizado para meu dia-a-dia.

Nem tudo acabou saindo como planejávamos e tivemos que lidar com o prazo para decidir o que deveríamos implementar e o que não era de tanta importância, além disso não foi possível realizar todos os testes que desejávamos. Mesmo assim, completamos boa parte do que nos propusemos a fazer, o que na minha opinião, faz com que nossa empreitada tenha sido bem-sucedida.

Como um balanço geral, os conhecimentos adquiridos ao longo da graduação foram de fato, alguns muito úteis, outros nem tanto. Porém, de

forma mais genérica, pode-se dizer que aprendemos realmente conceitos importantes nesses seis anos que passamos estudando no IME e fora dele, no meu caso, no KIT de Karlsruhe na Alemanha.

Agradecimentos

Gostaríamos de dedicar esta seção a agradecer àqueles que foram, de certa forma, especiais no decorrer do desenvolvimento e concepção desta obra.

Primeiramente, a nossos pais, que sempre nos ofereceram suporte e compreensão durante o longo período de nossos estudos.

Ao nosso orientador, o professor doutor Coelho, pelas nossas reuniões frequentes e por todos os seus conselhos. Esperamos que este trabalho tenha sido tão divertido para o senhor quanto foi para nós.

Nossa gratidão àqueles que voluntariamente cederam seu tempo na realização das entrevistas, tanto no começo quanto ao término de Torre de Controle. Um abraço especial ao Fábio, Heloisa, Maíra, Diego, Antoine e Anderson, verdadeiros conhecedores de suas áreas.

A conclusão desse trabalho também não seria possível sem a grande dose de compreensão e tolerância dos pais do Rafael, que cederam muitas de suas noites de sono em prol dessa empreitada.

E por fim, a nossos colegas de curso, tanto os que já terminaram quanto os que ainda estão por concluir o bacharelado em Ciência da Computação. Permaneçam fortes!

Muito obrigado a todos vocês.

Referências

[1]: Discurso do presidente Barack Obama. Acessado em 26 de novembro de 2015. URL:https://www.youtube.com/watch?v=6XvmhE1J9PY

[2]: “Addressing the Full Range of Students: Challenges in K-12 Computer Science Education”, revista Computer, edição de setembro de 2013.

[3]: Wikipedia: K-12. Acessado em 26 de novembro de 2015. URL:https://en.wikipedia.org/wiki/K%E2%80%9312

[4]: "Learn to Code, Code to Learn", artigo de Michel Resnik. Acessado em 26 de novembro de 2015. URL:https://www.edsurge.com/n/2013-05-08-learn-to-code-code-to-learn

[5]: “STEM Education - It’s elementary”. Artigo do US News. Acessado em 26 de novembro de 2015. URL:http://www.usnews.com/news/articles/2011/08/29/stem-education--its-elementary

[6]: “Why all our kids should be taught how to code”, artigo de John Naughton. Acessado em 26 de novembro de 2015. URL:http://www.theguardian.com/education/2012/mar/31/why-kids-should-be-taught-code

[7]: "Hong Kong children learn to code after school", Reportagem da BBC. Acessado em 15 de outubro de 2015. URL:http://www.bbc.com/news/business-32880185

[8]: “Número de smartphones supera o de computadores no Brasil”. Reportagem da Exame. Acessado em 26 de novembro de 2015. URL:http://info.abril.com.br/noticias/mercado/2015/04/numero-de-smartphones-supera-o-de-computadores-no-brasil.shtml

[9]: “Brasil tem 41,5 milhões de veículos”. Reportagem do UOL. Acessado em 26 de novembro de 2015. URL:http://omundoemmovimento.blogosfera.uol.com.br/2015/05/08/brasil-tem-415-milhoes-de-veiculos

[10]: “The Internet of Things is far bigger than anyone realizes”. Artigo de Daniel Burrus. Acessado em 26 de novembro de 2015. URL:http://www.wired.com/insights/2014/11/the-internet-of-things-bigger/

[11]: "Kindergarten is the model for lifelong learning". Artigo de Mitchel Resnick. Acessado em 26 de novembro de 2015. URL:http://www.edutopia.org/kindergarten-creativity-collaboration-lifelong-learning

[12]: Sousa, Rui Miguel & Lencastre, José Alberto (2014). "Scratch: uma opção válida para desenvolver o pensamento computacional e a competência de resolução de problemas". Acessado em 26 de novembro de 2015, URL:https://repositorium.sdum.uminho.pt/bitstream/1822/29944/1/RuiSousa%26JALencastre_EJML_2014.pdf

[13]: "Want to prepare kids for the future? Teach them to code". Reportagem do Los Angeles Times. Acessado em 26 de novembro de 2015. URL:http://articles.latimes.com/2014/apr/07/news/la-ol-teach-students-code-computer-science-20140406

[14]: "Escolas estonianas incluem programação no currículo do ensino fundamental". Reportagem da Folha de São Paulo. Acessado em 26 de novembro de 2015. URL:http://blogdetec.blogfolha.uol.com.br/2012/09/07/escolas-estonianas-incluem-programacao-no-curriculo-do-ensino-fundamental/#

[15]: “Conheça a gamificação, que transforma suas tarefas cotidianas em games”. Revista Galileu. Acessado em 26 de novembro de 2015. URL:http://revistagalileu.globo.com/Revista/Common/0,,EMI291109-17773,00-CONHECA+A+GAMIFICACAO+QUE+TRANSFORMA+SUAS+TAREFAS+COTIDIANAS+EM+GAMES.html

[16]: “Com desafios, missões e rankings, ‘gamificação’ pode turbinar EAD”. Reportagem do UOL Educação, com citação de Lynn Alves, Universidade do Estado da Bahia. Acessado em 26 de novembro de 2015. URL:http://educacao.uol.com.br/noticias/2014/02/21/com-desafios-missoes-e-rankings-gamificacao-pode-turbinar-ead.htm).

[17]: Wikipedia: Space Invaders. Acessado em 26 de novembro de 2015. URL:https://en.wikipedia.org/wiki/Space_Invaders

[18]: *Web site* do Lemon Amiga: Mega Ball. Acessado em 26 de novembro de 2015.
URL:http://www.lemonamiga.com/games/details.php?id=1582

[19]: *Web site* do Scratch. Acessado em 26 de novembro de 2015.
URL:https://scratch.mit.edu/

[20]: *Web site* do Lightbot. Acessado em 26 de novembro de 2015.
URL:https://lightbot.com/

[21]: *Web site* do Code Combat. Acessado em 26 de novembro de 2015.
URL:http://br.codecombat.com/

[22]: Wikipedia: Gamificação. Acessado em 21 de novembro de 2015.
URL:https://en.wikipedia.org/wiki/Gamification#cite_note-hamarireview-3

[23]: *Web site* builder DevHub gets users hooked by ‘gamifying’ its service”. Artigo de Dean Takahashi. Acessado em 20 de novembro de 2015.
URL:http://venturebeat.com/2010/08/25/devhub-scores-engagement-increase-by-gamifying-its-web-site-creation-tools/

[24]: Microsoft: Blog Oficial. Acessado em 21 de novembro de 2015
URL:http://blogs.microsoft.com/blog/2015/08/03/microsoft-acquires-fantasysalesteam-an-innovative-sales-gamification-platform-to-help-organizations-increase-productivity/

[25]: Hamari, Juho; Koivisto, Jonna & Sarsa, Harri (2014). "Does Gamification Works? - A Literature Review of Empirical Studies on Gamification". Acessado em 20 de novembro de 2015.
URL:http://people.uta.fi/~kljuham/2014-hamari_et_al-does_gamification_work.pdf

[26]: Kapp, Karl M. (2012). "The Gamification of Learning and Instruction", Livro Online. Acessado em 20 de Novembro de 2015.
URL:https://books.google.com.br/books?hl=en&lr=&id=M2Rb9ZtFxccC&oi=fnd&pg=PR12&dq=gamification+works&ots=JwLj_69F3H&sig=x3a9XH8aMWnMeFH5gd-Wmvsv6aA#v=onepage&q&f=false

[27]: *Web site* do *Unity*: Estatísticas. Acessado em 26 de novembro de 2015.
URL:http://unity3d.com/pt/public-relations

[28]: Documentação do Unity: Cenas. Acessado em 26 de novembro de 2015. URL:http://docs.unity3d.com/Manual/CreatingScenes.html

[29]: Documentação do Unity: Game Objects. Acessado em 26 de novembro de 2015.
URL:http://docs.unity3d.com/ScriptReference/GameObject.html

[30]: *Web site* do software Monodevelop. Acessado em 26 de novembro de 2015. URL:*http://www.monodevelop.com/*

[31]: Wikipedia: GNU - Licença pública geral. Acessado em 26 de novembro de 2015.
URL: https://pt.wikipedia.org/wiki/GNU_General_Public_License

[32]: *Web site* do Github. Acessado em 26 de novembro de 2015.
URL:https://github.com/

[33]: *Web site* do Trello. Acessado em 26 de novembro de 2015.
URL:https://trello.com/

[34]: *Web site* do Fog Creek. Acessado em 26 de novembro de 2015.
URL:https://www.fogcreek.com/

[35]: “O que é kanban e como usá-lo?”. Acessado em 26 de novembro de 2015.
URL:http://www.informant.com.br/blog/2013/11/06/o-que-e-kanban-e-como-usa-lo/

[36]: “Setor da tecnologia da informação oferece 276 mil vagas em todo o país”. Reportagem do Jornal Hoje. Acessado em 17 de novembro de 2015.
URL:http://g1.globo.com/jornal-hoje/noticia/2013/06/setor-da-tecnologia-da-informacao-oferece-276-mil-vagas-em-todo-o-pais.html

[37]: “Coding to be taught in Australian schools from primary age”. Artigo de Denham Sadler. Acessado em 26 de novembro de 2015.
URL:http://mashable.com/2015/09/21/coding-schools-australia/#8lH4lcvbiGkx

[38]: *Web site* do Code Academy. Acessado em 26 de novembro de 2015.
URL:https://www.codecademy.com/

[39]: *Web site* do Programaê. Acessado em 26 de novembro de 2015.
URL:http://programae.org.br/

[40]: *Web site* do Kids Grow: aconselhamento e tutelagem de crianças. Acessado em 26 de novembro de 2015.
URL:http://www.kidsgrowth.com/resources/articledetail.cfm?id=1007

[41]: “Você já sabe o que é gamificação”. Acessado em 26 de novembro de 2015.
URL:http://opusphere.com/voce-ja-sabe-o-que-e-gamificacao/

[42]: *Web site* da Mad Code: empresa de ensino infanto-juvenil. Acessado em 26 de novembro de 2015.
URL:http://madcode.com.br/

[43]: "Using storytelling to strengthen your brand". Acessado em 26 de novembro de 2015.
URL:http://www.i-scoop.eu/using-storytelling-strengthen-brand

[44]: "Final Fantasy XIII achieves 1 million-unit-five-day sales". Acessado em 26 de novembro de 2015.
URL:http://www.gamasutra.com/view/news/27750/FFXIII_Achieves_1_MillionUnit_FiveDay_Sales.php

[45]: “Top 10 franquias de games mais populares de todos os tempos”. Acessado em 26 de novembro de 2015.
URL:http://top10mais.org/top-10-franquias-de-games-mais-populares-de-todos-os-tempos

[46]: Pedersen, Roger E. (2009). Game design foundations. 2ª edição.

[47]: "Ubisoft sells 11 million copies of Assassin's Creed IV - and looks to release franchises ‘more regularly’". Acessado em 26 de novembro de 2015.
URL:http://venturebeat.com/2014/05/15/ubisoft-sells-11-million-copies-of-assassins-creed-iv-and-looks-to-release-franchises-more-regularly/

[48]: Jogo Torre de Controle. Acessado em 14 de novembro de 2015.
URL:http://www.ime.usp.br/~danielpa/torre_de_controle

[49]: Shneiderman, Ben & Plaisant, Catherine (2005). Designing the user interface: strategies for effective human-computer interaction. 4ª edição.

[50]: Bastien, J. M. C., & Scapin, D. L. (2001). Évaluation des systèmes d'information et Critères Ergonomiques. In C. Kolski (Ed.), *Systèmes d'information et interactions homme-machine. Environnement évolués et évaluation de l'IHM. Interaction homme-machine pour les SI*(Vol. 2, pp. 53-79). Paris : Hermes.

[51]: “How I Got My Mom to Play Through Plants vs. Zombies”. Palestra de George Fan na GDC 2012. Acessado em 26 de novembro de 2015. URL: http://www.gdcvault.com/play/1015541/How-I-Got-My-Mom

[52]: McLeod, S. A. (2015). Jean Piaget. Acessado em 26 de Novembro de 2015. URL:http://www.simplypsychology.org/piaget.html

[53]: Becker, Fernando (1994). “O que é Construtivismo?”. Acessado em 26 de novembro de 2015.
URL:http://www.crmariocovas.sp.gov.br/pdf/ideias_20_p087-093_c.pdf

[54]: “Ambiente de Aprendizagem Construtivista”. Artigo de Luis de França Ferreira. Acessado em 26 de novembro de 2015.
URL:http://penta.ufrgs.br/~luis/Ativ1/Construt.html

[55]: Breedlove, S. Marc (2015). Principles of Psychology.

[56]: Setzer, Valdemar W. & Monke, Lowell (2001). An alternative view on why, when and how computers should be used in education.

[57]: Jogos com Caneta e Papel: Batalha Naval.
URL:http://zamorim.com/jogos/papel/batalha-naval-regras.html

[58]: “Video games are not making us more violent, study shows”. Reportagem do The Guardian. Acessado em 26 de novembro de 2015.
URL:http://www.theguardian.com/technology/2014/nov/10/video-games-violent-study-finds.

[59]: *Web site* da AppStore. Acessado em 26 de novembro de 2015.
URL:https://itunes.apple.com/en/genre/ios/id36?mt=8

[60]: *Web site* daGoogle Play. Acessado em 26 de novembro de 2015.
URL:https://play.google.com/store?hl=en

[61]: *Web site* da MicrosoftStore. Acessado em 26 de novembro de 2015. URL:http://www.microsoftstore.com/store/msbr/pt_BR/home

[62]: Especificação da Linguagem YAML. Acessado em 26 de novembro de 2015.
URL:http://www.yaml.org/spec/1.2/spec.html