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
Professor Doutor 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 tounou-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 TI e áreas relacionadas, mas para qualquer pessoa que queira destacar-se em seu campo de atuação.

O Brasil ainda se mostra muito atrás do 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 tem 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 realizado por alunos do curso de Ciência da Computação da Universidade de São Paulo sob a supervisão do professor doutor José Coelho de Pina.

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

Abstract

TODO: Traduzir o resumo

Sumário

Motivação

O século XXI foi marcado em seu primeiro décimo pelo enraizamento da computação no dia a dia de todo cidadão. Seja morador da cidade ou do campo, jovem ou não, praticamente todo ser humano vivendo em sociedade presenciou 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, como 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, 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 elas 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 à 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 do Reino Unido em adicionar computação ao seu currículo para crianças a partir de cinco anos de idade através do programa K-12[2], 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 seja um programador ou desenvolvedor de software? 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":

> "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 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." [3]

O artigo do U.S. News, "STEM Education--It's Elementary"[4] , discute a importância não só da computação, mas também da ciência, tecnologia, engenharia e matemática (science, technology, engineering and mathematics - STEM). Ele afirma que essas áreas devem ser o maior foco de investimento nos próximos anos devido a configuração que a economia atual vem tomando.

"Se os Estados Unidos quer 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 de tecnologia"

O autor do artigo, Anthony Murphy, professor da Universidade St. Catherine em St. Paul, ainda afirma 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 regresso nesta área. A falta de planejamento pode acarretar em uma educação defasada para toda uma geração, como o que temos vivido recentemente[5].

Um grande erro que vem se cometendo através dos últimos vinte anos, como cita John Naughton em seu artigo "Why all our kids should be taught how to code"[5] é, ao invés de se ensinar às crianças as mais novas tecnologias presentes no mercado, vem se ensinando-lhes a utilizar softwares obsoletos. Isso é um caso do que o filósofo Gilbert Ryle chama de "erro de categoria": um erro em que 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 que 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 a Internet e, com os avanços no campo da Internet das Coisas, este número tende a aumentar rapidamente.[31][32][33]

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 e, no quesito tecnologia, é impossível a comparação com carros ou qualquer outro meio de transporte.

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

1) Primeiramente, vários conceitos-chave responsáveis pelo ambiente em que esta criança está inserida hoje. Como relata detalhadamente o artigo “Kindergarten is the model for lifelong learning” [6], o jardim de infância vem se tornando cada vez mais como 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, ela 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 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, o 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"[7] 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" [8] 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 por novos cientistas da computação.

Pode-se dizer que, devido à atual conjuntura econômica, 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[9]. 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 as 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 à 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 (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* [10])

Como disse Zichermann, a gamificação pode ser utilizada como instrumento de incentivo. Além disso ela é uma porta para a tecnologia no espaço escolar, como aponta Lynn Alves, professora da Uneb (Universidade do Estado da Bahia) no artigo "Com desafios, missões e rankings, "gamificação" pode turbinar EAD"[11].

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, além da realidade de que computação é ainda uma área obscura para a maioria da população brasileira e será de 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.

Objetivo

A meta principal deste trabalho é desenvolver uma plataforma de ensino de computação à distância para jovens baseada em ludificação 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 desse 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 software. 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.

Influências

Logo

TODO: esta seção

Scratch

TODO: esta seção

Light Bot

TODO: esta seção

Tecnologias

1. Unity 3D

O Unity3D, ou apenas Unity, é um motor de jogo (do inglês, *game engine*) que possiu 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 apenas restrito a esses fins. Também é empregado frequentemente na indústria de aplicativos e jogos móveis, ou seja, softwares produzidos para celulares e tablets, um mercado em crescente espansão.

O grande diferencial desta ferramenta é sua facilidade no porte para diversas plataformas, como *iOS*, *Android*, web, *XBox*, entre diversos outros. Isso torna o desenvolvimento mais centralizado e faz com que não seja necessário o desenvolvimento de um código fonte diferente para cada tipo de dispositivo.

Por estar disponível em duas versões, as chamadas versão *Pro* e a versão *Comum*, sendo a Pro uma opção paga e a Comum livre para o uso individual e educacional, optamos pela segunda. As funcionalidades extras presentes na versão Pro são necessárias apenas para desenvolvedores que pretendem colocar suas criações a venda ou no mercado aberto para download, não sendo este o cenário em que nosso software se enquadra.

1.2 Por que Unity ?

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

Além das diversas vantagens apresentadas acima decidimos pela utilização da plataforma pois ela é largamente utilizada. De acordo com o próprio website da empresa[12] já são mais de quatro milhões de desenvolvedores que utilizam o Unity, o que representa 45% no mercado global de motores de jogos, 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.

1.3 Terminologias e Conceitos

O 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 como ele funciona.

Projetos

É o conjunto de elementos que compõe todo o software a ser desenvolvido, contendo cenas, imagens, sons, e 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 requerimento da tecnologia Unity em sí.

Cada jogo criado é na verdade um novo projeto dentro do software Unity. No nosso caso utilizamos apenas um projeto.

Cenas

O Unity trabalha com o conceito de cenas, como pode ser observado em sua documentação[13]. Uma Cena é a maior subdivisão hierárquica dentro de um projeto de Unity, e cada cena podem ser vistas individualmente como uma "*tela*" do jogo. Cada cena contém os objetos que a ela são pertinentes. Por exemplo, nossa tela principal de jogo contém a nave, os botões, os painéis com opções de comando, entre outros.

Na prática, as cenas permitem o desenvolvimento do jogo em partes independentes, e o jogo em sí pode ser descrito pelas cenas que o compõe. 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 sí.

Assets

Os *Assets* são elementos 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, sendo estes as bases de todas as entidades do programa[14]. São eles que compõem as cenas, podendo ser instanciados, modificados, e adquirir 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 eles podem ser modificados dinamicamente, sem que o *Prefab template* seja alterado.

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

Componente

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 recorentes são: Um *script* que atribui 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 admitidas as linguagens C# e javascript.

Monodevelop[15]

O *Monodevelop* é um editor de texto e um ambiente de desenvolvimento integrado à plataforma Unity. Ele é a solução sugerida e que já acompanha a instalação do próprio Unity. Nele é criado o código de nosso software. Com suporte para várias linguagens, quando integrado ao Unity ele é usado para o desenvolvimento em C# e *Javascript*, sendo que optamos pela primeira opção por preferência pessoal.

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 Console automaticamente. Ou seja, o 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 também já integrado à *IDE*. 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 o 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 em uma página HTML.

2. Git

O *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*[16]. Ele foi desenvolvido inicialmente por Linus Torvalds, criador do Linux, para a gestão do desenvolvimento do *kernel* do *Linux*. É adotado hoje em dia por 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*, 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 na integração de novas funcionalidades.

2.1 Por que Git?

Sendo uma ferramenta de fácil manipulação, livre, gratuita que facilita o gerenciamento de projetos onde há um número grande de pessoas envolvidas, resolvemos adotar o *Git* como parte do processo de desenvolvimento. Com ele, nosso grupo pode 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.

2.2 Github

O *Github* é um sistema de hospedagem online do controle de versão *Git* disponível em https://github.com/. Ele foi desenvolvido em *Ruby on Rails*. Além disso ele disponibiliza diversos serviços adicionais como estatísticas e serviços de redes sociais entre outros. Optamos por ele pela necessidade de um servidor que estivesse online 24 horas por dia, e podendo utilizar-se da versão gratuita, concluímos que seria a melhor solução para nosso problema de hospedagem.

3. Trello

O Trello é uma ferramenta de gerenciamento de projetos disponível em https://trello.com/. Uma de suas grandes vantagens é o fato de ser online. Inicialmente criado pela *Fog Creek Software Products*[17], em 2014 tornou-se uma empresa própria. Ele é largamente utilizado, chegando a mais de cinco milhões de usuários como 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. Através do paradigma de *Kanban*, 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 website, ele também pode ser utilizado em plataformas móveis android, iOS e Windows Phone.

3.1 Por que Trello?

Optamos por esse serviço pois, apesar de não ser completamente gratuito, disponibiliza muitas ferramentas, que consideramos suficientes, para o gerenciamento de nosso projeto em grupo. Ele funcionou para nós como um Kanban de fácil acesso para todos.

3.2 Kanban

O paradigma Kanban[18] 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, que cuja posição e conteúdo orientam o desenvolvimento geral do projeto, ditando a urgência das tarefas e qual a próxima funcionalidade que deve ser produzida sem seguida. Dentre os softwares disponíveis para a prática do Kanban, o que mais toma destaque é o Trello, pela gratuidade dos serviços básicos e pela interface amigável, requisito básico do método.

A introdução do método de Kanban é de grande benefício aos projetos com poucos membros pois esse método auxilia na estruturação do desenvolvimento de projetos de pequeno e médio porte, permitindo a todos os membros envolvidos a 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 xp.

Em nosso projeto, o quadro de Kanban que adotamos era relativamente simples, dado a pequena quantidade de membros que compõe nosso time de desenvolvimento. Ele é composto basicamente das tarefas cumpridas e das ainda por desenvolver, com listagens das sub-tarefas a serem resolvidas dentro de cada tíquete.

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 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, o Professor Doutor José Coelho de Pina. 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. 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 se tem atualmente uma crescente preocupação no exterior com o ensino de programação nas escolas infantis. Exemplos claros da convergência de opiniões a respeito do ensino de computação pode ser encontrado nos currículos escolares da Inglaterra[2] e da Austrália[19], onde recentemente o ensino de programação foi incluído como conteúdo obrigatório para crianças e adolescentes a partir dos 7 anos.

Foi com base nessa conjuntura que decidimos implementar um sistema que ensinasse computação. Restava ainda saber o que, especificamente, seria esse essa aplicação e como nos proporíamos a ensinar efetivamente à distância com esta ferramenta. Considerando nosso público alvo, imaginamos que seria difícil garantir a retenção de usuários apresentando um projeto final como as ferramentas modernas de ensino eletrônico que são facilmente encontradas na internet, como Code Academy ou Programaê. Acredita-se que crianças tem dificuldade em manter-se focadas

por grandes quantiades de tempo[20], 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 idéia 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 Game Design, que propiciam ao usuário desfrutar melhor da atividade que está sendo proposta[21]. 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[22], que se propõe a ensinar programação e robótica a jovens de 5 a 17 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 tem naturalmente mais dificuldade na retenção da atenção, e por isso devemos ter uma consideração especial à interface com o jogador.

Passamos então a um estudo um pouco mais aprofundado do meio em que decidimos inserir nosso sistema e fizemos observações sobre as aplicações existentes que se propõe a fazer a mesma coisa que nosso projeto. Analisamos a interface de usuário, a forma como o raciocínio computacional é inserido na aplicação, e como os jogos levam em conta a progressão do ensino. As aplicações mais consagradas na área, o Scratch e o Code Combat foram estudadas a fim de definir qual seria o estilo de nossa aplicação. Essas duas referências foram ainda utilizadas na tomada de decisões de projeto como por exemplo a forma de input de usuário e os elementos principais dos jogos, os blocos de comando. Podemos observar com mais detalhes esses dois softwares na seção Influências deste trabalho.

Desenvolvimento

Desde o princípio do desenvolvimento da nossa aplicação, nas 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 Features

Seguimos um processo de discussão interna para a aprovação de conteúdo novo a ser adicionado ao projeto. Esse processo envolve a explicação da feature a ser adicionada para os outros dois membros do grupo, e coletivamente decidimos a introdução dessa funcionalidade no desenvolvimento do programa ou não. Após o processo de aprovação, a feature aprovada é adicionada no Trello sob a lista das funcionalidades 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 deses 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, é 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 relaciona com o restante do programa.

Depois que o desenvolvedor de uma feature considera que a testou o suficiente, este então lida com as ferramentas de manuseio de código compartilhado, os chamados sistema de versionamento. Como já foi dito nas seções anterires, versionamos o projeto por meio do Git, que foi mais que suficiente para o masueio de scripts e arquivos de código baseados em texto, mas que deixou a desejar com os inúmeros arquivos binários existentes nos projetos de Unity. Utilizamos Git extensivamente para manusear código compartilhado entre as diversas instancias do projeto que cada um de nós mantinha em seus respectivos repositórios locais de código fonte, e era através dele que os membros da equipe tinham acesso ao código que os outros desenvolviam.

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

Desenvolvimento em Equipe

Parte do projeto também foi desenvolvida em conjunto por todos os membros da equipe. Por vezes, tínhamos reuniões de implementação onde nos propusemos a implementar uma feature 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 em um brainstorm de concepção quanto na implementação dos scripts de level que gerenciam as instâncias de Game Object desses níveis.

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

Pesquisa de Campo

Com o início das atividades de nosso trabalho de conclusão de curso, entramos em contato com diversos *websites* 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*, formado em Ciência da Computação do 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 dos jovens que deveríamos focar em 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 cursos disponíveis pela MadCode, eles acreditam que a idade de doze anos é ideal para o início de tal atividade. Como foi apontado, crianças mais novas ainda não tem a base matemática para tanto, e mais que isso, não possuem a capacidade de dedicar atenção por muito tempo à mesma atividade. Além disso, a partir dessa 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 games. 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 (lançado para fliperama em 1978) ou MegaBall (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 aprende, 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, é necessária a utilização de um banco de dados em um servidor online para que os usuários e seus avanços sejam armazenados.

Após nossa conversa, tivemos ainda a oportunidade de conhecer o ambiente e material de trabalho utilizados pelos profissionais daquela unidade. A estrutura pedagógica é composta por duas salas, telões e notebooks, a fim de que os alunos não precisem levar nada ao local de ensino, o que para jovens representa uma grande comodidade.

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.

Design de Jogos

Em design de jogos, os processos de concepção e criação podem ser agrupados basicamente em duas linhas de pensamento: desenvolvimento orientado à mecânica de jogo e desenvolvimento orientado à narrativa.

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[23]. 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, 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 atingida recentemente, por exemplo, pelo jogo Final Fantasy XIII de 1 milhão de cópias vendidas em seu primeiro fim de semana à venda na América do Norte[24]. A franquia, pertencente a empresa Square-Enix, também figura entre as dez franquias mais bem-sucedidas da história dos jogos e reúne fãs ao redor do mundo inteiro[25].

Naturalmente, esses dois princípios 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 como 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[26]. 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 Torre de Controle, optamos pela abordagem focada na mecânica do jogo, e isto se deu por duas razões. É importante salientar que nossa principal meta era apresentar os conceitos essenciais e comuns às linguagens de programação através do recurso da ludificação, mascarando, assim, conceitos complexos e abstratos da programação por meio de uma interface responsiva e uma mecânica intuitiva (veja seção de 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.

Design

Torre de Controle é uma aplicação disponível livremente na Internet e é auto-suficiente 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.

Assim sendo, 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 usuário 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.

(**TODO**: explicar os princípios usados no desenvolvimento da interface)

Temática do Jogo

O jogo consiste em uma expedição intergalática, onde o aluno é convidado a programar uma nave espacial à 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, o jogador deve se empenhar e fazer bom uso de seu arsenal limitado para cumprir a missão espacial.

Inicialmente, somos apresentados à Tião, o mascote do jogo responsável por transmitir a missão, além de dar as explanações 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.

A interface é composta essencialmente por um conjunto de menus interativos e pode ser destrinchada em três partes principais, conforme a figura abaixo (**TODO**: incluir imagens das telas)

- 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; permitem a movimentação nas quatros direções triviais, giros de 45° em ambos os sentidos, além de disparos laser. 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.

- 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.

- 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. Contrariamente aos outros dois, 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

TODO: Descrever a divisão em quadrantes, asteroides e campos de força, minérios raros e o planeta de destino.

Pedagogia

Empregamos elementos pedagógicos clássicos da cátedra construtivista de ensino no jogo **Torre de Controle**. Podemos observar com o aluno constrói seu aprendizado a medida que utiliza os comandos oferecidos para completar o objetivo da fase em que se encontra. A construção 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 que descobre, por meio de experimentação no ambiente que oferecemos o que cada comando faz, e é a forma como os comandos são empregados para solucionar o nível que de fato ensina o aluno o chamado raciocínio computacional.

Acreditamos que o método construtivista é mais adequado aos nossos propósitos, dado que precisamos fixar a atenção do usuário, sabidamente de faixa etária infantil, em nossa aplicação, de maneira a garantir a retenção de alunos. Aplicações similares que seguem a mesma linha de construção de conhecimento que nosso projeto já existem. *Scratch*, *Light Bot* e *Code Combat* já foram todos testados e amplamente utilizados no ensino infantil com graus variados de sucesso, como vimos na seção *Influências*. Tomamos como base a forma como essas aplicações introduzem os conceitos às crianças no desenvolver de nossa aplicação.

Construtivismo

O construtivismo é uma das principais teorias modernas cognitivas e afirma que é a partir de interações entre humanos, suas idéias e suas experiências prévias que conhecimento é gerado[27]. A teoria construtivista foi fundada por Jean Piaget, suiço de nascimento, filósofo e psicólogo, fomentando grande impacto nas correntes de ensino e pensamento do século XX, em meados de 60.

Segundo Piaget, o conhecimento não é inerente ao sujeito que aprende, ou seja, ele não nasce pronto com o sujeito e precisa ser despertado; e tampouco provê completamente de observações do meio como dita a corrente empirista de ensino. Para ele, o conhecimento surge a partir da interação com

o meio em que se está inserido a partir das estruturas cognitivas inerentes ao sujeito[28].

TODO: Citar práticas construtivistas de ensino.

Modelo Tradicional

Em contraponto ao método que utilizamos em nossa aplicação temos o modelo Tradicional de ensino, onde um professor, detentor do conhecimento, o distribui aos seus alunos. Acreditamos que essa forma de aprendizagem não pode ser expressada no formato de jogo, que preza pela autonomia do jogador. Também acreditamos que essa forma de ensino se assemelha fortemente ao padrão de ensino que temos na grande maioria das escolas brasileiras, e receamos por isso que o usuário crie a falsa associação de **Torre de Controle** como uma extensão da escola, o que pode ter resultados negativos a curto prazo na retenção do usuário.

Fernando Becker, Professor de Psicologia da Educação da Faculdade de Educação de Universidade Federal do Rio Grande do Sul - UFRGS, afirma que a escola, como instituição generalizada, é motivo de insatisfação justamente pela forma como transmite os conhecimentos:

> "[...]. Tendências que têm em comum a 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 - a próxima e, aos poucos, as distantes." [29]

Por esses e pelos outros motivos já explicitados, escolhemos a abordagem construtivista na concepção do jogo.

TODO: Escrever sobre prós e contras do modelo contrutivista e tradicional.

Melhorias Futuras

TODO: Adicionar esta seção

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 encontramos 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 por isso 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 a altura de nossas expectativas quanto ao seu aspecto.

Acreditamos que a maior frustração que sofremos se deve a um problema conhecido no *framework 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 por meio de um *collider*, *Componente* do *Unity* que define uma caixa ao redor de um *Game Object*.

Descobrimos um problema do Unity com o posicionamento do *collider* da lixera, que entre duas execuções do programa muda de lugar de forma automática se ele 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, por vez que a introdução do bug se deve ao *Unity* em sí. O que fizemos para solucionar o problema imediato é reposicionar via script a posição do colider para o mesmo lugar que a imagem da lixeira, depois que o colisor foi 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 framework.

Também nos deparamos com um problema no sistema de versionamento que utilizamos para desenvolver Torre de Controle, o *Git*, 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 o *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 semelhante a XML[30]. Dada a apresentação dos dados no formato texto, seria possível resolver os conflitos nos recursos manualmente e portanto acelerar o passo de desenvolvimento da aplicação por meio do desenvolvimento paralelo nas cenas do projeto, onde nos revezávamos na edição dos arquivos do Unity.

Acreditamos que um grande desafio foi a implementação do algoritmo de interpretação da lista de comandos. Grande parte de nosos esforços foram focados no desenvolvimento dos algoritmos que são executados no interpretador do jogo. Grande parte do tempo foi dispendido nas refatorações que garantiram melhoria da legibilidade de código.

TODO: Adicionar o desafio da mecânica de Arrasto dos comandos

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 e a importância que têm para o projeto.

MAC0316 *- Conceitos Fundamentais de Linguagem de Programação*

Conceitos Fundamentais de Linguagem de Programação nos ajudou ativamente a definir os comandos que o usuário poderia utilizar no jogo, 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 em MAC0316.

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 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 idéia 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 Conclusão de Curso.

MAC0121 *- Principio 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.

Referências:

TODO: Formatar as referências segundo ABNT

[1]: Vídeo do presidente Barack Obama (https://www.youtube.com/watch?v=6XvmhE1J9PY)

[2]: (revista matéria da ACM, adicione a referencia aqui)

[3]: Artigo de Michel Resnik (https://www.edsurge.com/n/2013-05-08-learn-to-code-code-to-learn)

[4]: Artigo do US News sobre STEM (http://www.usnews.com/news/articles/2011/08/29/stem-education--its-elementary)

[5] : Artigo John Naughton - “Why all our kids should be taught how to code” (http://www.theguardian.com/education/2012/mar/31/why-kids-should-be-taught-code)

[6]: Artigo "Kindergarten is the model for lifelong learning." (http://www.edutopia.org/kindergarten-creativity-collaboration-lifelong-learnin)

[7]: Artigo sobre Scratch (https://repositorium.sdum.uminho.pt/bitstream/1822/29944/1/RuiSousa%26JALencastre_EJML_2014.pdf)

[8]: Los Angeles Times (http://articles.latimes.com/2014/apr/07/news/la-ol-teach-students-code-computer-science-20140406)

[9]: Tiger Leap Foundation (http://blogdetec.blogfolha.uol.com.br/2012/09/07/escolas-estonianas-incluem-programacao-no-curriculo-do-ensino-fundamental/#)

[10]: Gabe

(http://revistagalileu.globo.com/Revista/Common/0,,EMI291109-17773,00-CONHECA+A+GAMIFICACAO+QUE+TRANSFORMA+SUAS+TAREFAS+COTIDIANAS+EM+GAMES.html)

[11]: Lynn Alves, Universidade do Estado da Bahia (http://educacao.uol.com.br/noticias/2014/02/21/com-desafios-missoes-e-rankings-gamificacao-pode-turbinar-ead.htm).

[12]: Estatísticas do Unity (http://unity3d.com/pt/public-relations)

[13]: Documentação de Cenas do Unity (http://docs.unity3d.com/Manual/CreatingScenes.html)

[14]: Documentação Unity - Game Objects (http://docs.unity3d.com/ScriptReference/GameObject.html)

[15]: Monodevelop website *(http://www.monodevelop.com/)*

[16]: GNU - Licença pública geral (https://pt.wikipedia.org/wiki/GNU_General_Public_License)

[17]: Fog Creek, criadora do Trello (https://www.fogcreek.com/)

[18]: Kanban. (https://en.wikipedia.org/wiki/Kanban_(development))

[19]: Ensino de programação para crianças de 7 anos na Austrália (http://mashable.com/2015/09/21/coding-schools-australia/#8lH4lcvbiGkx)

[20]: Kids Grow, website de aconselhamento e tutelagem de crianças (http://www.kidsgrowth.com/resources/articledetail.cfm?id=1007)

[21]: Gamificação, Wikipédia (https://en.wikipedia.org/wiki/Gamification)

[22]: *Mad Code,* Empresa de ensino infantil e juvenil *(http://madcode.com.br/)*

[23]: Storytelling

(http://www.i-scoop.eu/using-storytelling-strengthen-brand)

[24]: Vendas FFXIII
(http://www.gamasutra.com/view/news/27750/FFXIII_Achieves_1_MillionUnit_FiveDay_Sales.php)

[25]: Video Games mais populares
(http://top10mais.org/top-10-franquias-de-games-mais-populares-de-todos-os-tempos)

[26]: Vendas Assassin's Creed
(http://venturebeat.com/2014/05/15/ubisoft-sells-11-million-copies-of-assassins-creed-iv-and-looks-to-release-franchises-more-regularly/)

[27]: Construtivismo, Wikipédia
(https://en.wikipedia.org/wiki/Constructivism_(philosophy_of_education)
(https://pt.wikipedia.org/wiki/M%C3%A9todo_de_ensino)

[28]: Artigo sobre construtivismo, professor da ufrgs
(http://penta.ufrgs.br/~luis/Ativ1/Construt.html)

[29]: Artigo de Fernando Becker sobre Construtivismo
(http://www.crmariocovas.sp.gov.br/pdf/ideias_20_p087-093_c.pdf)

[30]: YAML, Especificação da Linguagem
[http://www.yaml.org/spec/1.2/spec.html]

[31]: Relatório Anual de Tecnologia da Informação
[http://info.abril.com.br/noticias/mercado/2015/04/numero-de-smartphones-supera-o-de-computadores-no-brasil.shtml]

[32]: Número de veículos no Brasil
[http://omundoemmovimento.blogosfera.uol.com.br/2015/05/08/brasil-tem-415-milhoes-de-veiculos]

[33]: Artigo de Daniel Burrus sobre Internet das Coisas
[http://www.wired.com/insights/2014/11/the-internet-of-things-bigger/]