Dominando para não ser dominado: Autonomia tecnológica com o Projeto Jovem Hacker

Tel Amiel[1,2], Gabriel de Souza Fedel[2]
Flávia Linhalis Arantes[1], Alexandre Garcia Aguado[3]

[1]Núcleo de Informática Aplicada à Educação (NIED)
Universidade Estadual de Campinas (Unicamp)
Campinas – SP – Brasil

[2]Coletivo Revoada
Campinas – SP – Brasil

[3]Instituto Federal de São Paulo (IFSP)
Campus Capivari – SP – Brasil

tamiel@unicamp.br, fedel@fedel.net.br
farantes@unicamp.br, ale.garcia.aguado@gmail.com

Abstract. *This article discusses the Jovem Hacker project, an ongoing initiative aimed at creating a curriculum focused on "end user programming". In this article we present the project's trajectory discussing the principles behind the project and how the results of the first implementation have impacted the development of the second edition, presented in greater detail here. The course is now being offered to adolescents in two different cities. We aim, over implementations, to learn how to best construct a short-term experience that empowers youngsters to tinker and remix software and technology.*

Resumo. *Este artigo apresenta o projeto Jovem Hacker, uma iniciativa que busca criar um currículo focado no conceito de "end user programming". Neste artigo apresentamos a trajetória do projeto, apontando seus princípios e como os resultados da primeira implementação impactaram o desenvolvimento da segunda edição do projeto. O curso está sendo ofertado para jovens em duas cidades diferentes. Buscamos, através dessas iterações, gerar aprendizados sobre como melhor oferecer uma experiência de curto prazo que possa empoderar jovens a melhor "fuçar"e remixar software e tecnologia.*

1. Introdução

Há um redobrado interesse em inserir a programação de computadores como parte da formação de crianças e jovens. Alguns projetos buscam a programação como forma complementar ao ensino, e outros buscam inserir a programação como parte do currículo formal. Por trás do discurso da equidade e da inclusão, particularmente de mulheres e de minorias, há um interesse em garantir uma massa de trabalhadores que possa suprir a demanda atual e futura da indústria de software. Como ressalta Audrey Watters a noção de que "todos devemos saber programar" não deve ser vista de maneira tão abrangente. É necessário melhor entender o que "programar" significa e para quem, ou sejá, é importante entender as diferentes metodologias e objetivos que compõe as motivações do

“aprender a programar” [Watters 2014]. Nesse sentido, podemos tentar acelerar cada vez mais a exposição de crianças às novas mídias para que possam melhor competir por vagas em universidades. Podemos também fazer do ensino médio cada vez mais uma coleção de competências, e inserir a programação como mais uma “linguagem” a ser desenvolvida. Mas também é possível pensar no aprendizado da técnica como uma prática emancipadora.

Existem paralelos com o movimento *maker* (fazedores). Enquanto há entusiasmo com a possibilidade da fabricação de bens de consumo em casa ou em espaços públicos [Garcia 2015], tendemos a esquecer que, no longo prazo, o movimento *maker* tem paralelos na historia da mecanização. Soderberg argumenta que o movimento tem se esquecido da história das máquinas de controle digital (CNC) e como elas foram construídas com o intuito de promover a automação e o controle do capital intelectual desenvolvidos pelos trabalhadores [Soderberg 2013]. Alguns veem uma alternativa emancipatória na fabricação [Blikstein 2008]. Com base nas discussões sobre “epistemologias” de Papert e Turkle além da educação emancipatória de Freire, é possível configurar atividades *maker* que sejam inclusivas de diferentes tipos de saberes [Turkle and Papert 1990, Freire 2005]. Sendo assim, valoriza-se mais do que o conhecimento erudito formalizado e abstrato, colocando-o ao menos no mesmo pé que o “saber fazer” Não por menos, o pensamento computacional [Guzdial 2008], cada vez mais emaranhado no universo das mídias [Manovich 2011] pode ser utilizado fora do âmbito da computação, trazendo um outro olhar sobre problemas complexos da arte até a educação.

O projeto Jovem Hacker, fazendo jus ao seu nome, é fortemente alicerçado em uma cultura de partilha, busca do conhecimento e valorização da liberdade, pilares essencias da cultura hacker. Os hackers não programam necessariamente por ganhos financeiros, mas sim porque sentem prazer em fazê-lo. Linus Torvalds costuma dizer em suas exposições que "para um hacker, o computador por si só já é uma diversão"[Himanen 2001], (p. 19). Muitos bons projetos podem ser resultado desse tipo de atividade, como ocorreu com o sistema operacional Linux, a World Wide Web e tantas outras criações relevantes [Himanen 2001].

No projeto Jovem Hacker, criamos uma estrutura de curso focada na apropriação tecnológica que perpassa o hardware e o software, porém se, um enfoque no domínio de uma única linguagem de programação ou da necessidade de saber programar, estritamente. Nos alinhamos a um dos preceitos da cultura digital – habilidade de partir de algo que existe, para então fuçar, remixar e alterar para outros propósitos. A noção da apropriação e do remix é discutido no contexto educacoinal por Papert como *tinkering*; uma apropriação do termo *bricolage* de Levi-Strauss. No *tinkering* presa-se a improvisação e a “negociação“ ao longo do processo de criação mais do que um planejamento rígido e formal[Papert 1993].

Os preceitos dessa prática na computação podem ser notados no *end user programming* [Ko et al. 2011]. Aqui não há como enfoque, o nível do programador (expert ou novato), mas a prática da programação. Ou seja, qualquer um, mesmo um programador veterano, em alguns momentos, se engaja em *end-user programming*. São situações onde há menor preocupação com toda a estrutura de software e sua arquitetura, e muito maior interesse em fazer algo funcionar para o momento e a demanda que se apresenta. É menos prioritário a construção do código como um produto/entregável e maior ênfase

na resolução de um problema em um domínio específico de ação de um usuário. Exemplos incluem a programação em planilhas de cálculo, criação de scripts para automatizar rotinas de trabalho ou ainda a criação de animações por crianças [Ko et al. 2011].

A noção não nos é estranha, já que a prática da "gambiarra" é nativa ao Brasil e exemplifica o *tinkering* de sua maneira mais funcional desde as artes até a educação. Reúne ao mesmo tempo um "saber fazer" inteligente, com ingenuidade, e um senso de urgência e de limitações (financeiras, de tempo, de recursos, entre outros) [Amiel 2013]. Para além do discurso da penúria, o *end-user programming* é uma prática sensata em um ambiente web dominado por plataformas customizáveis como Wordpress, Drupal e Joomla e uma infinidade de aplicativos disponíveis em código aberto, fomentado pelo movimento do software livre. Muitas das atividades de um programador, hoje, são focadas em criar *templates*, customizar aparências, criar ou alterar funcionalidades, criar *mashups*, entre outras demandas que são caracterizadas muito mais por reutilizar código, aprimorado aos longo da execução do projeto, do que criá-lo do zero (codificado no mantra, *release early*, *release often*; disponibilize cedo, disponibilize com frequência).

Muitos destes conceitos estão conectados diretamente com o movimento hacker, particularmente a aplicação de uma "ética hacker" à educação. Busca-se que professores e alunos estejam cada vez mais engajados, juntos na produção de conhecimento [Pretto 2012]:

> "Recombinagem. Remixagem. Nova produção e diálogo permanente com o instituído, produzindo-se, a partir daí, novos produtos, novas culturas e novos conhecimentos. Tudo no plural. Com isso, temos a possibilidade de retomar o papel de liderança acadêmica do professor, que, em conjunto com os alunos, no coletivo e individualmente, passam a interagir de forma intensa com esse labirinto de possibilidades." (p. 314)

Com base em uma noção emancipatória da tecnologia, um olhar para *tinkering* e *end-user programming*, uma ética hacker fundamentada no software livre e no pensamento computacional, embasamos a nossa proposta. Com isso, nos perguntamos: Quais as demandas e particularidades dessas práticas? No que tange a formação de jovens, como essas práticas podem ser ensinadas, em contraste com um currículo tradicional e programação? No projeto Jovem Hacker buscamos explorar essas questões.

2. O Projeto Jovem Hacker

O projeto foi criado a partir da confluência de interesses de entusiastas do software livre, da cultural livre e do movimento hacker e pesquisadores da universidade. Não é portanto, um projeto que se iniciou com um interesse em pesquisa. Através da colaboração descentralizada em um documento colaborativo (*pad*) reunimos as ideias e sugestões de um grande grupo de pessoas para organizar a estrutura do que seria um curso para jovens. Em paralelo, encontros presenciais foram organizados para melhor organizar as discussões em torno de ações concretas.

Propusemos a esturtura do curso através de oficinas modulares. Ao longo das discussões contemplamos inúmeras possibilidades de formação desde a estrutura de dados, hardware livre, dados abertos até bancos de dados. Ao final de um ano de discussões e deliberações online, definimos a estrutura básica de curso com base em um público-alvo

de jovens, uma estrutura modular, e alguns temas centrais que seriam abordados. A proposta buscava fomentar o interesse do jovem pela programação com vista sa criar algo que fosse socialmente útil e pessoalmente gratificante. Em grandes traços, o "consenso" apontou para um currículo tinha como base (1) fazer uma introdução à computação e ao computador, com enfoque em software/hardware livre e o trabalho colaborativo e distribuído; (2) conceitos básicos de lógica de programação como forma preparatória para (3) a programação para web (HTML/CSS/Javascript), e finalizando com uma (4) linguagem de alto nível com aplicação além da web.

Nossa primeira experiência no contexto do Projeto Jovem Hacker (www.jovemhacker.org) aconteceu entre abril e maio de 2014, por meio da realização de um piloto realizado no CIS-Guanabara (www.cisguanabara.unicamp.br) em Campinas. Nove jovens de escolas públicas, com idades entre 16 e 18 anos, participaram do piloto. Foram 8 encontros semanais, com duração de 1 hora e 30 minutos cada. O projeto piloto teve portanto como enfoque, a primeira parte da proposta – uma introdução à programação e a lógica de programação, fazendo uso do Scratch. Com essa experiência, identificamos algumas lições para o desenvolvimento do projeto (para saber mais veja [Arantes et al. 2014]:

- Menor uso de atividades expositivas. De acordo com a avaliações que realizamos, a aula que eles menos gostaram foi a expositiva. Isso nos indica, que ao menos no contexto das oficinas do Jovem Hacker, devemos buscar uma maneiras mais participativa e exploratória de apresentar para trabalhar conceitos teóricos e muitas vezes de difícil compreensão. É perceptível a mudança na motivação quando a aula é exploratória. Felizmente, o ensino de computação para iniciantes se beneficia de inúmeras possibilidades mão na massa, incluindo o uso de atividades da computação desplugada [Feaster et al. 2011]. A computação desplugada consiste em realizar atividades com o objetivo de ensinar os fundamentos da Ciência da Computação sem a necessidade de computadores. Uma vantagem dessa abordagem reside na sua independência de recursos de hardware ou software, bem como na possibilidade de abordar temas de computação de uma maneira lúdica e mais divertida [Bell et al. 2009].
- Promover a socialização. O encontro que os alunos mais gostaram foi um dia em que criamos uma dinâmica (justamente com a computação desplugada) para inserir conceitos de paralelismo, variáveis e condicionais. Com essa dinâmica, abordamos conceitos, muitos deles avançados, de uma maneira lúdica. Além disso, a atividade proporcionou socialização e descontração ao grupo, o que torna os encontros mais leves e agradáveis.
- Usar exemplos em crescente nível de complexidade. Os alunos sentiram dificuldade em um dos encontros porque o exemplo era muito grande. Nesse caso, o ideal seria mostrar exemplos menores, que eles possam explorar e, passo a passo, construir um programa maior.
- Encontros mais longos. Em paralelo com os anseios de educadores tradicionais, os alunos reclamaram do fato das aulas serem curtas (1h e 30 min cada), com pouco tempo para explorarem os programas. Sendo essa uma experimentação "não formal", é possível explorar possibilidades de engajamento estendido com os alunos.

- Alunos precisam se sentir capazes. O maior reforço para o aprendizado é o sucesso: a motivação dos alunos claramente aumentou quando eles se sentiram capazes de fazer os programas.

Avaliamos esses aprendizados, e reformulamos o projeto Jovem Hacker com uma nova proposta/design para 2015. Em termos metodológicos nos alinhamos à pesquisa baseada em design, ou *design-base research* (DBR; ou PBD). Nesse modelo as implementações são vistas como designs que são estudados profundamente para que fatores e condições que afetam a qualidade da implementação sejam identificados. O acumulado de experiências, a cada nova iteração, leva a um conhecimento mais abrangente e integral da proposta e do contexto. Sendo assim, o design em si é alvo de investigação e não é uma proposta imutável. Cada nova implementação deve compreender os aprendizados das iterações anteriores e documentar detalhadamente as condições de implementação e sua fidelidade, com relação ao inicialmente proposto. As revisões devem ser repensadas com base nos conhecimentos adquiridos em campo, com um olhar cuidadoso para os contextos e limitações reais de cada implementação [Colletive 2003].

Abaixo apresentamos a nossa proposta de reformulação do curso. Passado o ano de discussões abertas, online, reunimos um grupo mais coeso para que, ao longo de seis meses nos engajássemos em leituras sobre pesquisas já realizadas no tema. De forma colaborativa resumimos artigos e compartilhamos os dados abertamente, utilizando o Zotero (www.zotero.org/groups/jovem_hacker). Com base em discussões semanais, alinhamos a proposta que apresentaremos a seguir. Em seguida descrevemos o plano de implementação do projeto Jovem Hacker que acontecerá em paralelo em duas cidades e dois contextos diferentes, enriquecendo nossos designs e bases para pesquisa e melhoria da proposta.

3. As edições

Em 2015, o projeto conta com duas edições, em Campinas e Capivari (SP). Serão dois contextos de estudo, com designs paralelos e complementares.

3.1. Edição Cultura Campinas

Com intuito de realizar a edição completa do curso (contemplando as parte 1-4 acima), onde pudéssemos colocar em pratica o currículo inicialmente proposto, submetemos o projeto a editais, e fomos contemplados com o edital da Secretaria do Estado da Cultura de São Paulo (Edital de Cultura Digital). Teremos espaço para que alunos contemplem ao longo do projeto, a realização de um produto final. Ao final do projeto todos participarão de uma fase intensiva de desenvolvimento com apoio individual, para que coloquem em pratica o que estudaram, e possam compreender a capacidade que possuem de desenvolver projetos para o bem comum.

A edição Campinas contará com apoio de um oficineiro e de um auxiliar para cada tema, bem como uma pequena equipe de apoio para seleção e construção de material didático, que será compartilhado com licenças e formatos livres, online (versão em desenvolvimento disponível com licença CC-BY-SA em http://jh.rgaiacs.com). Nesta edição desenvolveremos atividades ao longo de quatro meses e meio, não incluindo o período para o desenvolvimento de projetos ao final do curso que terá duração de aproximadamente um mês. Participarão das oficinas vinte jovens provenientes de escolas públicas, que deverão se inscrever no projeto por meio de formulário próprio e aberto.

3.2. Edição IFSP Capivari

A Edição Capivari é uma ação promovida como projeto de extensão do Instituto Federal de São Paulo (IFSP) – Campus Capivari. Participarão do projeto aproximadamente vinte adolescentes, alunos do nono ano das escolas públicas de Capivari. Serão dois encontros semanais de maio a dezembro de 2015. A escolha do perfil dos adolescentes (nono ano do ensino fundamental), esta diretamente relacionada com a possibilidade de continuidade da formação. Os interessados poderão dar continuidade direta na formação técnica através de um Ensino Médio Técnico Integrado, presente no próprio IFSP, em diferentes áreas, como Química, Informática e Gestão.

As oficinas acontecerão em um telecentro do projeto de inclusão Capivari Digital, onde toda infraestrutura técnica, desde computadores até a rede serão construídos ao longo do projeto pelos próprios participantes. Ao final do projeto, este telecentro e toda estrutura montada, ficará como uma contribuição para a comunidade local, servindo também de referência para os adolescentes participantes, afinal, perceberão de forma bastante clara o quão possível é contribuir para a comunidade e construírem coisas juntas que beneficiem a todos.

Todas as etapas do projeto envolvem tanto discentes bolsistas do IFSP, docentes e técnico-administrativos, que juntos com o público-alvo buscam desenvolver um ecossistema de colaboração.

Ambas as edições acontecerão ao longo de 2015. No entanto, o calendário da edição Capivari acontecerá com um mês de atraso. Este atraso permitirá que as reflexões imediatas sobre o curso em Campinas possam servir de base para modificações em nossos designs para a implementações em Capivari. A equipe trabalha desde 2014 em conjunto, contanto com aproximadamente vinte membros em uma variedade de funções.

4. O Projeto de Formação

A capacitação ou empoderamento tecnológico é fundamental para evitar uma grande massa de "usuários" que seja dependente da tecnologia sem entender seu funcionamento, além de ser um passo que pode incentivar jovens a explorar futuros profissionais que envolvam a tecnologia da informação [Venceslau 2013] desde a tecnologia da informação, ao jornalismo e artes visuais.

A formação no Projeto Jovem Hacker está distribuída em módulos e tem seu foco no aprendizado através de projetos práticos e colaborativos. O projeto abordará arquitetura básica de computadores e de software. Em seguida utilizará a ferramenta Scratch para ensinar lógica de programação. Num terceiro momento introduziremos o desenvolvimento web com HTML, CSS e JavaScript. Seguimos com a introdução de uma linguagem de programação para web (e além) utiliando Python. Como momento final da formação definiremos, em conjunto com os alunos, um projeto de interesse coletivo ou individual, que seja desenvolvido com base nas atividades apresentada nas oficinas.

Além do objetivo de incentivar a autonomia tecnológica, o Projeto Jovem Hacker busca encontrar formas alternativas de construir tal conhecimento. Partindo de referências construídas de maneira coletiva e descentralizada, tais como Transparência Hacker, Ônibus Hacker e Hackatons, o projeto buscará fomentar um interesse pelo conhecimento construído colaborativa e compartilhadamente. Nas subseções a seguir, detalhamos cada

módulo previsto na formação do Projeto Jovem Hacker.

4.1. Hardware/software livre

Neste primeiro modulo, apresentaremos o funcionamento físico do computador. Os participantes terão a possibilidade de abrir um computador e fuçar em todos seus componentes. Também apresentaremos a relação do computador com outros tipos de hardware, realizando experimentações coletivas com Raspberry Pi e smartfone/celulares, buscando discutir as relações entre esses tipos de equipamentos computacionais, suas limitações e seus contextos de uso.

Consideramos importante discutir o aspecto ético do uso das ferramentas partindo de usos específicos e contextualizados. Buscamos, logo no início, desmistificar o funcionamento dos equipamentos. É essencial, dentro de nossa proposta, que os alunos percebam estes equipamentos como ferramentas cognitivas [Kim and Reeves 2007] e não como telas de consumo. Abordaremos aqui uma discussão sobre o conceito do "livre" (hardware e software) tanto de forma prática como uma noção ética (no sentido da ética hacker) e do trabalho colaborativo, temas que permearão as discussões ao longo dos encontros, de forma transversal.

Na edição Capivari, haverá maior ênfase nesta fase. Os participantes montarão o seu próprio computador e terão a possibilidade de construir a rede de computadores do laboratório, com vistas a um melhor entendimento sobre o funcionamento de redes e da Internet.

4.2. Introdução à lógica de programação

Para trabalhar introdução à programação utilizaremos o ambiente Scratch [Resnick et al. 2009], que é um sucessor do Logo [Papert 1980] e baseia-se nas mesmas bases construcionistas de seu antecessor para promover o pensamento computacional. Com uma interface gráfica drag-and-drop, o Scratch permite a criação de histórias interativas, jogos e animações, além de facilitar a importação e criação de vários tipos de mídia (imagens, sons, músicas). No site da comunidade do Scratch (scratch.mit.edu), os usuários compartilham seus projetos, recebem *feedback* e são encorajados a executar e a remixar projetos já publicados. Os programas são feitos arrastando-se blocos de comandos que devem ser encaixados uns nos outros como peças de lego. Quando combinados, os comandos formam programas sintaticamente corretos, com isso o usuário pode ter seu foco na lógica de programação sem se preocupar com a sintaxe.

O público do Scratch é bem amplo, com trabalhos na literatura que vão desde o ensino de programação para crianças até cursos de Ciência da Computação [Aureliano and Tedesco 2012]. O projeto Code Club (codeclubbrasil.org), por exemplo, tem seu foco em crianças. O projeto oferece material de ensino de Scratch para uma rede de voluntários, com a meta de ensinar crianças a programar através de jogos e animações. Com foco em um público jovem, os trabalhos de Malan e Leitner e Harvey e Mönig descrevem experiências feitas em cursos de Harvard e Berkeley, respectivamente, tendo Scratch como primeira linguagem para jovens que querem seguir carreira em Ciência da Computação. Ambas experiências colaboraram para construção de nosso módulo, visto que trabalham com Scratch como um ambiente facilitador para introduzir conceitos básicos de programação que antecedem linguagens de programação reais [Malan and Leitner 2007, Harvey and Mönig 2010].

Com base na experiência piloto (veja seção 2), expandimos o tempo de interação com os jovens com encontros de maior duração. O módulo de introdução à lógica de programação acontecerá em quatro encontros, com quatro horas cada num total de 16 horas. Ao projetarem soluções interativas com Scratch, os participantes podem se envolver de maneira lúdica com um conjunto de conceitos de computação que poderão ser transferidos (com apoio) às outras atividades, utilizando linguagens de programação como Javascript e Python.

O prório Scratch é um facilitador para atividades de exploração individual e em grupo. Não nos reduziremos somente ao uso do Scratch para esse fim. Com maior ênfase nesta fase, mas também ao longo dos outros módulos, lançaremos mão de atividades que terão como base a computação desplugada [Bell et al. 2009] que obtiveram bons resultados durante o piloto. Nos basearemos, em boa parte, da sequência didática apresentada por França e Amaral, que propõe uma metodologia para estimular o pensamento computacional com Scratch para jovens do ensino fundamental II e médio [França and Amaral 2013]. Para cada conceito, será apresentada sua definição, exemplos de uso e exploração por parte dos alunos. Dentre outros, abordaremos:

- Sequência: é uma série de instruções que, quando executadas pelo computador, produzem uma ação;
- Evento: é um acontecimento que produz uma ação;
- Paralelismo: são sequências de instruções executadas ao mesmo tempo;
- Loop: mecanismo que permite executar a mesma sequência várias vezes;
- Condicionais: permitem que "decisões" sejam tomadas considerando condições pré-estabelecidas;
- Operadores: oferecem suporte a operações matemáticas, lógicas e com strings;
- Dados: armazenamento, recuperação e atualização de valores com a utilização de variáveis e listas;
- Remix: Além de criar seus próprios projetos, deverão reutilizar e remixar projetos produzidos por outros usuários Scratch que estejam disponíveis na comunidade.

4.3. Programação para a web

Uma vez abordadas as questões básicas de programação no módulo anterior, buscaremos um engajamento rápido com linguagens de programação que impactam diretamente a experiência desses alunos no seu convívio com a tecnologia.

Neste módulo começaremos, com maior força, a enveredar pelo *end-user programming*, conduzindo atividades largamente baseadas no remix, no reuso de código, e na investigação do que já é compartilhado. Atividades que envolvem a programação por modelo (ou programming by sample) se enquadra no modelo de ensino proposto [Hartmann et al. 2007]. Focaremos o curso em CSS, HTML e Javascript, uma tríade básica e complementar que permite, em poucas investidas, um grande poder de transformação no layout e funcionamento de páginas web. Nos basearemos largamente nas experiências do Mozilla Webmaker (webmaker.org), particularmente no que tange a desmistificação da web através de atividades práticas de remix.

Começamos com atividades sobre o funcionamento da web, e seguimos com momentos práticos explicando o funcionamento de tags e seletores. Com isso, os alunos poderão começar a construir e remixar código para criar suas próprias páginas. Com um conhecimeto básico da estrutura de páginas, passamos para a edição de imagens utilizando GIMP, para que possam gerar logos e outros elementos visuais. Terminamos com noções básicas de DOM e de jquery para que possam manipular elementos nas páginas.

Faremos uso, quando apropriado, de tutoriais disponíveis para a introdução às linguagens, como o módulo de Javascript disponível no Code Academy (codecademy.com) que pode servir de referência para os alunos para apendizado para além dos módulos. Ademais, apresentaremos as grandes possibilidades do *remix* e *mashup* de código-fonte já disponível (e.g., jsfiddle.net). Neste módulo, que terá duração aproximada de 24 horas, queremos que os jovens já comecem a interrogar questões importantes que podem servir de base para seus projetos de "final de curso".

Toda essa prática é permeada por valores essenciais da cultura hacker, onde o compromisso ético com uma web aberta não será ignorado assim como a busca pelo conhecimento, a valorização da liberdade e o espírito de partilha se farão presentes durante todo o processo.

Um dos objetivos intrínsecos do projeto é contribuir para que os participantes percebam-se autores. Contribuímos então para o sentido da autoria colaborativa, intimamente ligada às práticas de reuso e do remix; da desconstrução do autor soberano. A autoria poder ser sintetizada através do upload de imagens e filmes, publicação de posts e outras formas de exposição na rede. Incluiremos oportunidades para que os jovens publiquem suas criações na web ao longo do percurso, para que possam perceber-se como autores, e compartilhar suas criações.

É por isso que este módulo trataremos com maior intenção o compartilhamento e remix de código. Não almejamos que alunos consigam, em tão curto prazo, dominar o versionamento de código; para isso, haverá apoio central dos oficineiros. No entanto vemos como essencial que para além do discurso sobre compartilhamento, alunos consigam enxergar seu código em repositórios abertos e sintam, de imediato, o poder da colaboração aberta.

4.4. Programação de alto nível para a web

Com intuito de coloca-los em contato com uma linguagem de programação poderosa e flexível, este módulo abordara o uso de Python, linguagem largamente recomendada como linguagem de alto nível para iniciantes [Mannila and de Raadt 2006]. Além de ser uma linguagem de fácil aprendizado, pode ser utilizada em diversos contextos, como o desenvolvimento de programas, jogos e, particularmente, aplicações web. Seguiremos a mesma lógica do módulo web. Não esperamos que ao final do módulo, tenhamos programadores independentes em Python. Queremos no entanto, que alunos possam compreender a estrutura de um código básico, e fazer alterações significativas em código já existente.

Algumas características da linguagem Python a tornam uma boa escolha para este momento da formação: (1) através dela é possível ter retorno imediato das ações de programação; (2) é uma linguagem de facil aprendizado, porém de alto nível; (3) é bastante pertinente em ambientes móveis (4); e é uma linguagem livre, aberta e uma referência no mundo do software livre. Essas características do Python poderão contribuir fortemente

para que os participantes do projeto consigam ir além do curso em sua aprendizagem e possam se envolver em outros projetos. Este módulo terá duração aproximada de 24 horas.

O enfoque do módulo será a construção de jogos 2D utilizando Pygames e, seguindo a discussão do módulo anterior sobre a manipulação do DOM, trabalharão com Brython (www.brython.info). Nesse projeto os alunos serão expostos a interação "server side"e noções básicas de programação orientada a objetos. Ao final do módulo terão um jogo completo pronto para uso.

Tanto na edição de Campinas quanto em Capivari, este módulo apresentará em sua metodologia um forte apelo a partilha e desenvolvimento colaborativo, afinal, espera-se que neste momento os participantes estejam mais familiarizados com as ferramentas e possam de fato contribuir tecnicamente com seus projetos.

4.5. Projeto final

Ao longo do curso, iniciando na fase web, temos como meta ajudar os alunos a definir um projeto de interesse coletivo ou individual, exequível, que possa ser trabalhado ao longo de um mês após os módulos/encontros. Para potencializar as atividades, para além dos oficineiros e auxiliares, convidaremos voluntários para apoiar a criação do projeto. Para tanto, o penúltimo encontro será um *hackday* reunindo interessados e voluntários para auxiliar no alinhamento e roupagem final do projeto, que será apresentado no encontro (final) de fechamento do curso.

O projeto visa, de maneira prática, demostrar como o trabalho com o software livre e o código aberto, pode e deve ser feito de maneira colaborativa. Busca também demonstrar que apesar da formação ser em nível iniciante, os alunos poderão criar algo que seja de utilidade pessoal e social. Neste sentido a tarefa ou projeto proposto deve ser "autêntico"[Herrington and Oliver 2000], seguindo alguns critérios:

1. Tem relevância ao mundo real e à produção de um produto significativo - queremos projetos que, mesmo que incipientes, sejam de genuíno interesse do aluno;
2. Atividades não são definidas a priori - devem surgir ao longo do curso e da interação dos alunos com seus pares, tutores e voluntários;
3. Não acontecem de maneira rápida - teremos pouco mais de um mês para a condução do projeto, mas continuaremos a apoiar os alunos (os incorporando a comunidades online e presenciais) nos meses que seguirão;
4. Promovem múltiplas perspectivas e vão além de um único domínio ou disciplina - não queremos fomentar projetos que visam somente o aprendizado de código;
5. Providenciam oportunidades para colaboração e reflexão - o trabalho em grupo, colaborativo, entre novatos e programadores experientes será essencial.

Na edição Capivari, os alunos desenvolverão um projeto prático que atenda alguma necessidade da comunidade local, além de terem que promover oficinas e um *install fest* para a comunidade, praticando assim, o compromisso social local e desde já compreendendo que conhecimento é algo a ser partilhado.

5. Conclusão e trabalhos futuros

Com este projeto buscamos criar um currículo novo e experimental para a formação de jovens. Buscamos com isso encontrar um espaço entre o um curso formal de computação

de longa duração, e cursos online de duração curta ou de "sensibilização". Aqui, buscamos introduzir, de forma prática, conceitos básicos e essenciais da computação para que alunos possam construir projetos práticos e começar a sua inserção no mundo da programação e do software livre.

Não basta que jovens aprendam a programar, seja como parte do currículo formal na escola, ou fora desta. O tipo de formação e os seus objetivos importam. Ressaltamos portanto, os pontos que nos parecem mais relevantes desse projeto de formação, e que, cremos, faz com que este possa contribuir para nossas continuadas reflexões sobre a formação de jovens:

1. Produzir um projeto de formação originado na esfera do ativismo e da sociedade civil que conta com a participação da universidade de maneira horizontal;
2. Criar um currículo de maneira aberta e descentralizada;
3. Encontrar maneiras de ir além do ensino de programação abstrata/visual (como Scratch) para linguagens de alto nível e de utilidade prática em contextos que extrapolam a educação formal;
4. Utilizar o conceito de remix como tema transversal para discussão da ética e da prática da programação.

Não sabemos se esse modelo obterá sucesso no que tange a formação de jovens que possam remixar e reutilizar código para atinigr seus objetivos utilizando linguagens de programação. Avaliaremos a experiência ao longo do curso, e no espírito da DBR, buscaremos aprimorar, documentar e divulgar o modelo com base nos aprendizados para novas ofertas do curso (seja por nós ou por outros interassados).

Teremos, com base nessa experiência, a oportunidade de estudar mais dois modelos de implementação, aprimorando o design do curso, e contribuindo com o conhecimento sobre a formação de jovens para a apropriação tecnológica.

* Agradecemos a participação de João Sebastião de Oliveira Bueno, Melissa Devens, Raniere Silva, Ricardo Panaggio.

Referências

Amiel, T. (2013). Um computador por aluno: Gambiarras – jeitinhos – geringonças. Presented at the II UCABASC, Salvador, BA. Retrieved from http://educacaoaberta.org/um-computador-por-aluno/.

Arantes, F. L., Amiel, T., and Fedel, G. (2014). Nos rumos da autonomia tecnológica – desafios e lições aprendidas para a formação de jovens. In *Anais do XX Workshop de Informática na Escola.*, pages 1–10, Dourados, MS. Retrieved from http://www.br-ie.org/pub/index.php/wie/article/view/3113.

Aureliano, V. C. O. and Tedesco, P. C. A. R. (2012). Ensino-aprendizagem de programação para iniciantes: uma revisão sistemática da literatura focada no sbie e wie. In *Anais do SBIE 2012*, pages 1–10, Rio de Janeiro. Retrieved from http://br-ie.org/pub/index.php/sbie.

Bell, T., Alexander, J., Freeman, I., and Grimley, M. (2009). Computer science unplugged: School students doing real computing witwith computers. *The New Zeland Journal of Applied Computing and Information Tecnology*, 13(1):20–29.

Blikstein, P. (2008). *Travels in Troy with Freire: technology as an agent for emancipation*, chapter Social Justice Education for Teachers: Paulo Freire and the possible dream, pages 205–244. Sense, Rotterdam, Netherlands.

Colletive, D. B. R. (2003). Design-based research: An emerging paradigm for educational inquiry. *Educational Researcher*, 32(1):5–8.

Feaster, Y., Segars, L., Wahba, S. K., and Hallstrom, J. O. (2011). Teaching cs unplugged in the high school (with limited success). In *Proceedings of the 16th Annual Joint Conference on Innovation and Technology in Computer Science Education*, ITiCSE '11, pages 248–252, New York, NY, USA. ACM.

França, R. S. and Amaral, H. J. C. (2013). Proposta metodológica de ensino e avaliação para o desenvolvimento do pensamento computacional com o uso do scratch. In *Anais do Workshop de Informática na Escola (WIE 2013)*. Retrieved from http://www.br-ie.org/pub/index.php/wie/article/view/2646.

Freire, P. (2005). *Pedagogia do Oprimido.* Paz e Terra, Rio de Janeiro.

Garcia, G. (2015). Prefeitura irá instalar 12 fablabs públicos em são paulo. Retrieved from http://info.abril.com.br/noticias/tecnologia-pessoal/2015/02/prefeitura-ira-instalar-12-fablabs-publicos-em-sao-paulo.shtml.

Guzdial, M. (2008). Education: Paving the way for computational thinking. *Commun. ACM*, 51(8):25–27.

Hartmann, B., Wu, L., Collins, K., and Klemmer, S. R. (2007). Programming by a Sample: Rapidly Creating Web Applications with D.Mix. In *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology*, UIST '07, pages 241–250, New York, NY, USA. ACM.

Harvey, B. and Mönig, J. (2010). Bringing "no ceiling" to scratch: can one language serve kids and computer scientists? In *Proceedings of Construcionism Conference*, pages 1–10, Paris.

Herrington, J. and Oliver, R. (2000). An instructional design framework for authentic learning environments. *Educational Technology Research and Development*, 48(3):23–48.

Himanen, P. (2001). *A ética dos Hackers e o espírito da era da informação: a diferença entre o bom e o mau hacker.* Editora Campus, Rio de Janeiro.

Kim, B. and Reeves, T. (2007). Reframing research on learning with technology: in search of the meaning of cognitive tools. *Instructional Science*, 35(3):207–256.

Ko, A. J., Abraham, R., Beckwith, L., Blackwell, A., Burnett, M., Erwig, M., Scaffidi, C., Lawrance, J., Lieberman, H., Myers, B., Rosson, M. B., Rothermel, G., Shaw, M., and Wiedenbeck, S. (2011). The state of the art in end-user software engineering. *ACM Comput. Surv.*, 43(3):21:1–21:44.

Malan, D. J. and Leitner, H. H. (2007). Scratch for budding computer scientists. *SIGCSE Bull.*, 39(1):223–227.

Mannila, L. and de Raadt, M. (2006). An objective comparison of languages for teaching introductory programming. In *Proceedings of the 6th Baltic Sea Conference on*

Computing Education Research: Koli Calling 2006, Baltic Sea '06, pages 32–37, New York, NY, USA. ACM.

Manovich, L. (2011). *The language of new media*. Leonardo Book Series, Cambridge, MA.

Papert, S. (1980). *Mindstorms: Children, Computers, and Powerful Ideas*. Basic Books, Inc., New York, NY, USA.

Papert, S. (1993). *The children's machine: Rethiking school in the age of the computer*. BasicBooks, New York.

Pretto, N. (2012). *Recursos Educacionais Abertos: Práticas colaborativas e políticas públicas*, chapter Professores-autores em rede., pages 91–108. Casa da Cultura Digital/Edufba., São Paulo.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., and Kafai, Y. (2009). Scratch: Programming for all. *Commun. ACM*, 52(11):60–67.

Soderberg, J. (2013). A ilusória emancipação por meio da tecnologia. Le Monde Diplomatique. Retrieved from http://www.diplomatique.org.br/artigo.php?id=1339.

Turkle, S. and Papert, S. (1990). Epistemological Pluralism: Styles and Voices within the Computer Culture. *Signs*, 16(1):128–157.

Venceslau, M. (2013). Falta de profissionais de ti se agravará no brasil, diz idc. INFO Exame Online. Retrieved from http://info.abril.com.br/noticias/carreira/falta-de-profissionais-de-ti-se-agravara-no-brasil-diz-idc-19032013-12.shl.

Watters, A. (2014). "everyone should learn to code.". Retrieved from http://www.wise-qatar.org/learn-to-code-audrey-watters.