Design para a exploração criativa

Título original: Designing for Tinkerability

Mitchel Resnick e Eric Rosenbaum, MIT Media Lab

(Publicado em *Design, Make, Play*, organizado por Margaret Honey e David Kanter)

1. Introdução

Revista *Make*. Feiras *Makers*. Makerspaces. Clubes *Makers*. No último ano, houve um aumento no interesse pelo *fazer* (em inglês, *making*). Um número cada vez maior de pessoas está participando da construção, criação, personalização e customização das coisas ao seu redor, fazendo suas próprias joias, móveis e até robôs. O surgimento do Movimento *Maker* é potencializado por tendências tecnológicas e culturais. As novas tecnologias facilitam e tornam mais acessível para as pessoas a criação e o compartilhamento de coisas, tanto no mundo físico quanto no digital. Ao mesmo tempo, o Movimento *Maker* tem suas bases em uma guinada cultural mais ampla, pautada pela abordagem *do-it-yourself* (faça você mesmo), na qual as pessoas têm orgulho e prazer em criar coisas pessoais em vez de meramente consumir produtos produzidos em massa.

Embora a maioria das pessoas envolvidas no Movimento *Maker* não esteja focada explicitamente na educação ou aprendizagem, as ideias e práticas do Movimento *Maker* vão ao encontro de uma longa tradição da área da Educação que encoraja uma abordagem de ensino focada em projetos e na experiência: do progressivismo de John Dewey (Dewey, 1938) até o construcionismo de Seymour Papert (Papert, 1980, 1993) - que encoraja a aprendizagem baseada em projetos, abordagem experiencial para a aprendizagem. Essa abordagem tem relativamente pouco espaço nos sistemas educacionais atuais, que têm uma forte ênfase na apresentação de conteúdo e na avaliação quantitativa. No entanto, o entusiasmo que envolve o Movimento *Maker* oferece uma nova oportunidade para revitalizar e revalidar a tradição progressista-construcionista da Educação.

Para isso, é necessário mais do que apenas "fazer". Frequentemente, temos visto escolas que introduzem o "fazer" no currículo de uma maneira que minimiza todo o espírito da atividade: "Estas são as instruções para fazer um carro robótico. Siga as instruções com cuidado. Você será avaliado de acordo com o desempenho do carro." Ou: "Construa uma ponte que suporte 100 quilos. Com base na sua criação, calcule a tensão da ponte. Quando tiver certeza de que sua ponte suporta 100 quilos, construa a ponte e confirme que ela aguenta essa massa."

Nessas atividades, os alunos estão fazendo algo, mas a experiência de aprendizagem é limitada. Apenas fazer coisas não é suficiente. Há várias abordagens diferentes para fazer coisas, e algumas levam a experiências mais ricas de aprendizagem do que outras. Neste artigo, focamos em uma abordagem específica do fazer que descrevemos como *tinkering*[1]. A abordagem do *tinkering* é caracterizada por um estilo lúdico, experimental e iterativo, na qual a pessoa que está criando reavalia continuamente suas metas, explorando novos caminhos e imaginando novas possibilidades. O *tinkering* não é valorizado (e é até mesmo desestimulado) em vários sistemas educacionais atuais. No entanto, o tinkering é alinhado às metas e ao espírito da tradição progressista-construcionista. Além disso, do nosso ponto de vista, é exatamente do que precisamos para ajudar jovens a se prepararem para a vida na sociedade contemporânea.

Nosso principal objetivo com este artigo é analisar estratégias que incentivem e apoiem a abordagem do *tinkering* nos processos de criação e aprendizagem. Para nós, trata-se de um desafio de design: como podemos criar tecnologias e atividades para o *tinkering*? Nós iniciamos, nas próximas duas sessões, apresentando uma descrição mais completa do significado de *tinkering* e por que achamos que ele é uma parte tão valiosa do processo de aprendizado. Em seguida, examinamos tecnologias e atividades específicas que criamos com nosso grupo de pesquisa do MIT Media Lab, discutindo nossas estratégias para incentivar e apoiar o *tinkering*.

[1] A palavra tinkering não tem uma tradução direta em português. Em geral, ela costuma ser entendida como "explorar livremente ideias e materiais", adaptar, improvisar ou mesmo "pensar com as mãos". Seymour Papert costumava usar os termos *bricolage* ou *bricoleur* para se referir ao processo de *tinkering* e à pessoa que o pratica, respectivamente. No caso deste texto, optou-se por uma combinação de termos, focando principalmente na "exploração livre" tão comum no *tinkering*.

2. O que é *tinkering*?

O termo *tinkering* significa diferentes coisas para diferentes pessoas. Não é incomum ver esse termo sendo usado com desdém (por falantes de língua inglesa) — por exemplo, *just tinkering (um mero tinkering)* — em referência a alguém que trabalha sem uma meta ou propósito claro, ou sem apresentar progresso perceptível. Mas, do nosso ponto de vista, *mero (just)* e *tinkering (tinkering)* não são termos compatíveis. Percebemos o *tinkering* como um estilo de trabalho válido e valioso, caracterizado por um estilo lúdico, exploratório e iterativo de lidar com um problema ou projeto. Quando as pessoas estão *tinkering*, elas estão constantemente testando ideias, fazendo ajustes e testando novas possibilidades várias e várias vezes.

Muitas pessoas acreditam que o *tinkering* é oposto ao *planejamento*, e é comum pensarem que o planejamento é uma abordagem inerentemente superior. O planejamento parece mais organizado, direto e eficiente. As pessoas que planejam, analisam a situação, identificam problemas e necessidades, desenvolvem um plano claro e o executam. Fazer só uma vez e fazer certo. O que poderia ser melhor que isso?

O processo de *tinkering* é mais bagunçado. As pessoas que praticam o *tinkering* estão sempre explorando, experimentando e tentando coisas novas. Quem planeja costuma basear-se em regras formais e cálculos abstratos (por exemplo, calcular a melhor posição para uma viga de sustentação de uma estrutura), enquanto quem pratica o *tinkering* costuma reagir aos detalhes específicos de uma situação (testar diferentes locais para a viga de sustentação ou explorar outras maneiras de sustentar a estrutura). Nas palavras do teórico do *design*, Don Schoen, quem pratica o *tinkering* tem "uma conversa com o material" (Schoen, 1983).

Às vezes, quem pratica o *tinkering* começa sem uma meta. Em vez de seguir a abordagem de *cima para baixo,* do planejamento tradicional, quem pratica o *tinkering* usa uma abordagem de *baixo para cima*. Eles iniciam brincando com materiais (por exemplo, juntando peças de LEGO formando diferentes padrões), e o objetivo surge com base nessa exploração lúdica (por exemplo, decidir construir um castelo fantástico). Em outros casos, as pessoas que estão praticando o *tinkering* têm uma meta geral, mas não sabem muito bem como atingi-la. Eles pode começar por um plano experimental, mas eles adaptam-o e renegociam-o com base nas suas interações com os materiais e com as pessoas com quem estão trabalhando. Por exemplo, uma criança pode começar com a

meta de construir um sistema de segurança para o quarto dela, e testar diversos materiais, estratégias e *designs* antes de criar uma versão final.

Há uma longa tradição de *tinkering* em várias culturas ao redor do mundo. Em quase todos os países, as tradições locais de artesanato evoluíram ao longo de séculos, sendo caracterizadas pela experimentação e pelo *tinkering* com materiais regionais. Em diversos locais, as pessoas desenvolvem uma mentalidade de faça-você-mesmo , às vezes por necessidade econômica, utilizando quaisquer ferramentas e materiais disponíveis naquele momento. Esse estilo de interação às vezes é chamado de bricolagem. O antropólogo Claude Levi-Strauss (1966) descreveu como pessoas em várias partes do mundo agem como *bricoleurs*, sempre improvisando por meio do uso de materiais disponíveis para construir e consertar objetos da vida cotidiana deles. Levi-Strauss compara a figura do *bricoleur* com a de um engenheiro que desenvolve um plano de maneira sistemática e coleta os materiais necessários para executá-lo.

O *tinkering* e a bricolagem estão muito próximos da brincadeira. Várias pessoas veem a brincadeira como uma forma de entretenimento ou diversão, mas nós a vemos como algo um pouco diferente. Para nós, brincar é uma forma de interagir com o mundo, um processo de testar limites e novas possiblidades. Nós vemos o *tinkering* como um estilo lúdico de criar e fazer, no qual você constantemente experimenta, explora, e testa novas ideias durante o processo de criação. O *tinkering* pode ser um trabalho difícil, algumas vezes sequer parece uma brincadeira. No entanto, há sempre um espírito lúdico que permeia o processo de *tinkering*.

É comum associar o *tinkering* a construções físicas: construir um castelo com peças de LEGO, construir uma casa na árvore com madeira e pregos ou criar um circuito com componentes eletrônicos. O Movimento *Maker* reforçou essa imagem, porque foca na criação de coisas no mundo físico. No entanto, nós temos uma visão mais ampla de *tinkering*. Nós vemos o processo de *tinkering* como uma abordagem de fazer coisas, independentemente de serem coisas físicas ou virtuais. Você pode *tinker* (explorar livremente) quando estiver programando uma animação ou escrevendo uma história, não apenas quando estiver criando algo físico. A questão principal é o estilo de interação, não a mídia ou os materiais utilizados.

3. Por que o *tinkering* é importante?

O *tinkering* não é uma ideia nova. Desde que os humanos de antigamente começaram a criar e usar ferramentas, o processo de *tinkering* tem sido uma estratégia valiosa para fazer coisas. Mas o *tinkering* é mais importante hoje do que nunca. Vivemos em um mundo caracterizado pela incerteza e por mudanças rápidas. Várias das coisas que você aprendeu hoje logo estarão obsoletas. No futuro, o sucesso não dependerá do que ou de quanto você sabe, mas sim da habilidade de pensar e agir de maneira criativa; da sua habilidade de elaborar soluções inovadoras para situações inesperadas e problemas imprevisíveis.

Nesse ambiente de mudanças rápidas, o *tinkering* é uma estratégia especialmente valiosa. O *tinkerer* sabe como improvisar, adaptar e iterar, sem depender de planos velhos quando surgem situações novas. O *tinkering* prioriza a criatividade e a agilidade acima da eficiência e da otimização, o que é uma troca válida em um mundo em constante mudança.

Apesar dos benefícios, o *tinkering* costuma ser menosprezado na sociedade atual, especialmente nos sistemas formais de educação. As escolas tendem a enfatizar o valor do planejamento, ensinando aos alunos analisar todas as opções, desenvolver uma estratégia e posteriormente executar os planos. Por isso, alunos que são planejadores por natureza costumam ter bons resultados na escola. Mas e quanto aos alunos que são *tinkerers* por natureza? Eles tendem a se sentir excluídos e alienados, especialmente em aulas de STEM (ciências, tecnologia, engenharia e matemática), que enfatizam especialmente o planejamento de cima para baixo. Assim, vários estudantes acabam sem motivação para estudar matemática e ciências, resultando em uma população menos cientificamente letrada e em um gargalo nas profissões de STEM.

Não precisa ser assim. Disciplinas de STEM não são inerentemente voltadas ao planejamento. Na verdade, profissionais especialistas em disciplinas de STEM costumam utilizar muito mais o *tinkering* em seus trabalhos do que normalmente se faz nas atividades de sala de aula de STEM (Brown, Collins, & Duguid, 1989). Vários grandes cientistas e engenheiros da história -de Leonardo da Vinci a Alexander Graham Bell e Barbara McClintock, até Richard Feynman - viam a si mesmos como praticantes do *tinkering*. Para aumentar a participação e incentivar a inovação em disciplinas de STEM, precisamos repensar e revisar o currículo de STEM para que ele seja mais convidativo e

fascinante para *tinkerers*, e não apenas para planejadores.

Turkle e Papert (1990) defendem o "pluralismo epistemológico", ou seja, o respeito e a valorização de múltiplos estilos de aprendizagem e de diversas formas do saber. Eles sugerem que a lógica e o planejamento devem estar "on tap" (disponíveis conforme necessários para situações específicas), e não "on top" (considerando serem superiores). Como o *status* da lógica e do planejamento é privilegiado, Turkle e Papert levantam a preocupação de que várias pessoas sejam excluídas de disciplinas de STEM não por regras explícitas, "mas pelas maneiras de pensar que geram sua relutância em participar". Eles argumentam que o *tinkering* e a bricolagem devem ter o mesmo *status* que a lógica e o planejamento: "A *bricolagem* é uma maneira de organizar o trabalho. Ela não é um estágio para chegar a uma forma superior" (p. 141).

Vários educadores continuam céticos em relação ao *tinkering*. Existem várias críticas comuns. Alguns educadores acreditam que pessoas que praticam o *tinkering* podem ter sucesso na criação de algo sem entender completamente o que estão fazendo. Isso pode ser verdade em alguns casos, no entanto, mesmo nesses casos, o *tinkering* oferece aos alunos a oportunidade de desenvolver fragmentos de conhecimento que eles podem integrar posteriormente a uma compreensão mais completa (Hancock, 2003). Outros se preocupam que o *tinkering* seja desestruturado demais para ter êxito. Essa crítica confunde o *tinkering* com a exploração aleatória. O processo de baixo para cima do *tinkering* começa com explorações que podem parecer aleatórias, mas não termina aí. Quem pratica o *tinkering* de verdade sabe como transformar suas explorações iniciais (*de baixo*) em uma atividade focada (*para cima*). É a combinação de *baixo* e *cima* que torna o *tinkering* um processo valioso.

É claro que o planejamento de cima para baixo também pode ser valioso. Mas, em várias situações, o planejamento é visto como a abordagem correta para resolver problemas, não apenas uma várias alternativas possíveis. Nossa meta é eliminar o *status* privilegiado do planejamento e dar a mesma ênfase ao *tinkering*.

4. Computação + habilidade *tinkering*

Vários materiais - como blocos de madeira e massa de modelar - favorecem o *tinkering*, permitindo criar casas, castelos, pontes, esculturas e outras estruturas. Mas e se quisermos criar coisas que sentem, reagem, interagem, se movem e se comunicam?

Para isso, normalmente são necessários materiais e ferramentas computacionais. Materiais computacionais podem não parecer muito adequados para o *tinkering*, já que a computação costuma ser associada à lógica e à precisão. E, de fato, atividades computacionais (especialmente a programação) são comumente apresentadas por meio de atividades mais atraentes para planejadores do que para quem pratica o *tinkering* - por exemplo, aprender como classificar uma lista de números.

Em nosso grupo de pesquisa Lifelong Kindergarten, do MIT Media Lab, estamos tentando mudar a forma como jovens utilizam e veem a computação. Desenvolvemos um conjunto de atividades e *kits* de construção computacional que incentivam explicitamente a criação e o *tinkering* com a computação. Por exemplo, em colaboração com o LEGO Group, nós desenvolvemos os *kits* de robótica LEGO Mindstorms e WeDo, que permitem que jovens construam dispositivos de robótica que se movimentam, sentem, interagem, e se comunicam. Nesse processo, os jovens aprendem conceitos importantes de matemática, engenharia e computação. Ainda mais importante, eles aprendem a pensar de maneira criativa e a trabalhar colaborativamente: habilidades essenciais para ter uma participação ativa na sociedade contemporânea.

Nesta seção, descrevemos dois *kits* de construção computacional do nosso grupo, o Scratch e o MaKey MaKey, que foram criados explicitamente para permitir aos jovens aplicar o *tinkering*. Na seção seguinte, usaremos esses dois *kits* como base para a análise de como criar atividades para a exploração criativa.

Scratch

Com o Scratch, você pode programar suas próprias histórias interativas, jogos, animações e simulações, além de compartilhar essas criações *online*. Para criar um programa no Scratch, basta unir blocos gráficos de programação em um *script*, como peças de LEGO (Figura 1). Para cada personagem (ou ator) do seu projeto Scratch, você precisa montar um conjunto de *scripts* para controlar o comportamento dele (Figura 2). Para um ator que é um peixe, por exemplo, um *script* controla o movimento do peixe pela tela, enquanto outro *script* diz para o peixe mudar de direção se ele bater em um coral. A partir de um conjunto simples de blocos de programação, combinado com imagens e sons, é possível criar uma grande variedade de projetos. Desde o lançamento do Scratch, em 2007, jovens de todo o mundo já compartilharam mais de 2,6 milhões de projetos no

site do Scratch (Figura 3), incluindo mensagens interativas, simulações científicas, excursões virtuais, anúncios de serviços públicos, jogos de videogame e concursos de animações de dança (Resnick et al., 2009; Maloney, Resnick, Rusk, Silverman, & Eastmond, 2010; Brennan & Resnick, 2012).

Figura 1 — Blocos de programação do Scratch[2]

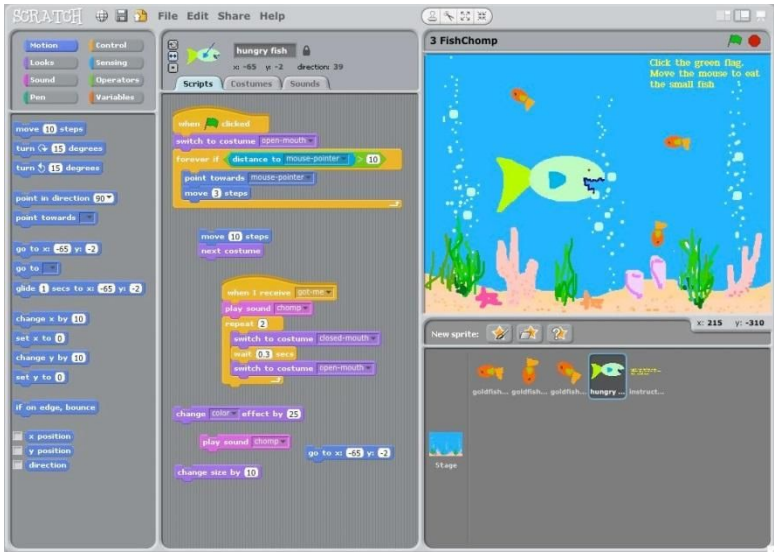

Figura 2 — Interface de programação do Scratch

[2] Tradução dos blocos: se tocar o gato? diga "lindo gatinho" por 2 segundos, toque o som miau.

Figura 3 — Site do Scratch e comunidade online

Durante a criação de projetos no Scratch, os jovens normalmente se engajam a um projeto extenso de *tinkering*, criando *scripts* de programação e fantasias para cada ator, testando para ver se eles se comportam como esperado e, em seguida, revisando e adaptando várias e várias vezes. Para ter uma ideia desse processo, veja o trabalho de uma integrante da comunidade Scratch cujo nome de usuário é EmeraldDragon. Nos seus primeiros sete meses na comunidade, EmeraldDragon compartilhou 25 projetos no site do Scratch. Em um de seus primeiros projetos, EmeraldDragon criou um jogo no qual o usuário controla os movimentos de um dragão animado. Ela criou 12 imagens de um dragão, cada uma com as pernas do dragão em posições um pouco diferentes, e depois criou um *script* de programação que mudava as imagens para criar a noção de movimento, como um *flipbook*.

EmeraldDragon testou diferentes versões do *script* para fazer o dragão se mover em diferentes direções, conforme o usuário apertava diferentes teclas. Quando ela compartilhou o projeto no site do Scratch, ela incluiu o seguinte comentário: "Eu estava

apenas explorando (em inglês, *tinkering*) os *scripts* do jogo e descobri como fazer ele ir para lá e para cá! Vou consertar o jogo para lançar uma versão nova e melhorada que pareça mais com um jogo de verdade!"

EmeraldDragon deu ao projeto o nome "*My Dragon Game (NOT finished)"* [*Meu Jogo do Dragão (NÃO acabado)*] para deixar claro que ainda estava trabalhando no projeto (Figura 4). Nas Notas do Projeto, ela escreveu: "Estou trabalhando para conseguir ir de um lado para o outro sem a pedra desaparecer. Alguma dica ou ajuda?" Na seção de Comentários do projeto, outros membros da comunidade Scratch ofereceram sugestões. Pouco tempo depois, ela compartilhou uma nova versão do projeto, dessa vez com o nome "*My Dragon Game (Still NOT finished)"* [*Meu Jogo do Dragão (Ainda NÃO acabado)*].

EmeraldDragon claramente entendeu que o *tinkering* é um processo contínuo de revisão e adaptação. Como ela escreveu nas Notas do Projeto: "Esta é só uma etapa de um longo processo."

Figura 4 — Jogo feito por EmeraldDragon

MaKey MaKey

Como o MaKey MaKey (http://makeymakey.com), você criar interfaces para computador a partir de qualquer objeto que conduza eletricidade: por exemplo, criar um teclado de piano a partir de pedaços de fruta, ou usar massa de modelar para criar um controle para um jogo de PacMan (Silver & Rosenbaum, 2012). Para fazer uma nova interface, você usa garras jacaré para conectar objetos à placa de circuito do MaKey MaKey (Figura 5), a qual, por sua vez, é ligada ao computador por uma porta USB. O MaKey MaKey finge que é um teclado de computador, então você pode fazer suas próprias teclas. É possível substituir a tecla de espaço (ou qualquer outra tecla) por uma banana ou qualquer outro objeto que conduza um pouco de eletricidade. O MaKey MaKey detecta quando você toca um objeto, e completa o circuito que envia um sinal para o computador, informando que uma tecla foi pressionada (Figura 6).

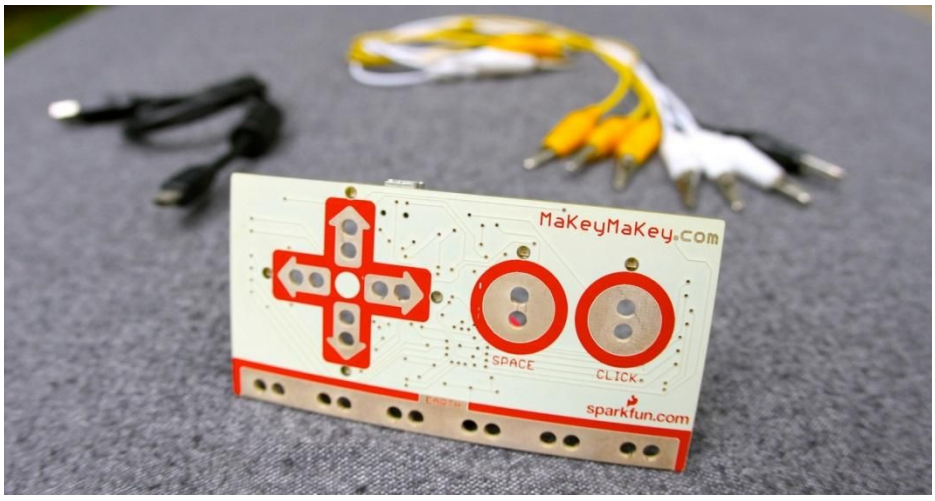

Figura 5 — Placa de circuito do MaKey MaKey com um cabo USB e garras jacaré

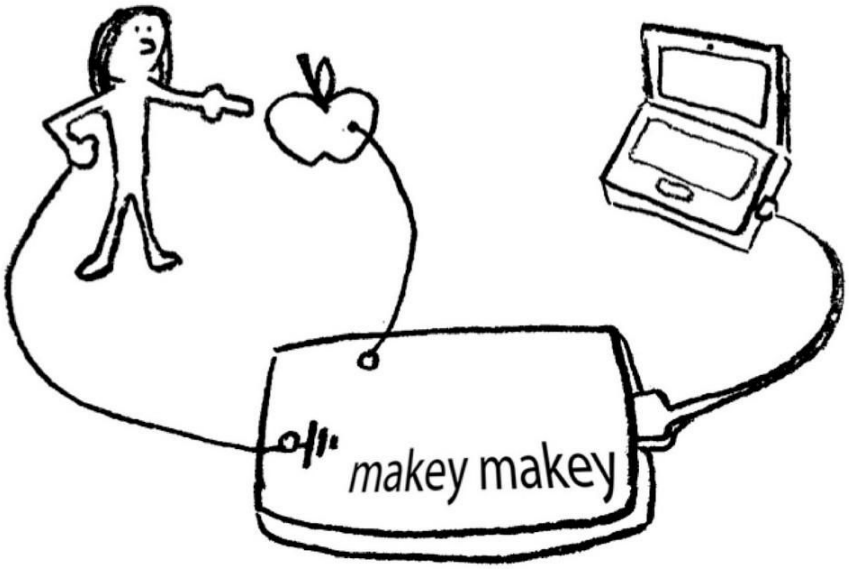

Figura 6 — Circuito do MaKey MaKey

Com base em vários projetos observados nas oficinas, apresentamos, aqui, um caso que ilustra mais detalhadamente como o MaKey MaKey funciona. Anna, uma menina de 12 anos, era fascinada por um piano de bananas que viu em um vídeo do MaKey MaKey (Figura 7), e então decidiu fazer um para si mesma. Primeiro, ela encontrou um site que permitia que ela tocasse uma escala de piano usando as teclas de uma das linhas do teclado do computador: a, s, d, f, etc. Depois, ela conectou a placa do circuito do MaKey MaKey ao computador, para criar suas próprias teclas e acionar as notas do piano. Ela organizou as bananas em sequência e usou as garras jacaré para conectar cada uma delas a uma letra diferente no MaKey MaKey. Em seguida, ela se conectou ao MaKey MaKey conectando um limão ao "terra" (também chamado de "base") no MaKey MaKey e segurou o limão com uma das mãos. Ao tocar em uma das bananas, o circuito ficou completo: um pouco de corrente fluiu a partir da tecla de entrada "a" no MaKey MaKey, pela banana, por ela, pelo limão e de volta para o "terra" no MaKey MaKey. O MaKey MaKey detectou a conexão e informou ao computador que a tecla "a" havia sido pressionada, fazendo com que o site do piano tocasse uma nota musical.

Figura 7 — Piano de bananas

Conforme ela conectava mais bananas, fez algumas descobertas acidentais. Primeiramente, as bananas estavam na ordem errada. Por isso, em vez de formar uma escala, elas tocavam um trecho de uma melodia familiar. Isso fez com que ela reorganizasse as bananas, como parte de um experimento musical. Durante esse processo, Anna percebeu que duas bananas estavam se tocando. Por isso, quando ela tocava qualquer uma delas, as duas acionavam as notas e tocavam um acorde. Posteriormente, chegou um amigo de Anna, Leo, e juntos tentaram criar um circuito que passasse pelo corpo dos dois. Quando Anna segurava o limão e Leo tocava uma banana, eles conseguiam acionar uma nota quando batiam as mãos, ou quando Anna tocava no nariz de Leo. Isso fez com que eles tivessem a ideia de criar um *kit* de bateria humano: eles encontraram um site que transformava toques no teclado em diferentes sons de bateria. Em seguida, juntaram alguns amigos para conectar ao circuito, de modo que cada um acionasse um som diferente.

Mais tarde, Anna e Leo testaram outros materiais. Eles fizeram uma busca pela casa e encontraram vários itens que funcionaram bem, como jujubas, moedas, massa de modelar e o grafite de um desenho feito a lápis em um papel. Enquanto estavam moldando a massa de modelar em diferentes formatos, eles tiveram a ideia de criar um controle de videogame, com botões de massa de modelar. Quando conectaram os botões às teclas de direção do MaKey MaKey, eles conseguiram jogar Mario Brothers (Figura 8).

Figura 8 — Controle de massa de modelar

Por meio desses processos de *tinkering* com o MaKey MaKey, Anna e Leo conseguiram rapidamente testar ideias, dedicar-se a descobertas fortuitas, testar uma grande variedade de materiais físicos e criar suas próprias invenções, dos mais diversos gêneros.

5. Como criar *kits* adequados ao *tinkering*

Como é possível criar atividades adequadas ao *tinkering*? Após refletir sobre os *kits* de construção desenvolvidos pelo nosso grupo de pesquisa ao longo dos anos, identificamos três princípios-chave que guiam nossas criações: *feedback* imediato, experimentação fluida e exploração aberta.

5.1 *Feedback* imediato

O processo de *tinkering* normalmente envolve uma série de experimentos rápidos — e, para realizar experimentos rápidos, são necessários resultados rápidos. Nos *kits* de construção que favorecem o *tinkering*, o intervalo de tempo entre a realização de uma mudança e a observação do seu efeito é muito curto. Vários processos físicos têm essa

propriedade: quando uma folha de papel é dobrada, não é necessário esperar até que ela fique marcada. Mas, alguns processos físicos (como assar, fundir metais ou colar objetos) exigem um tempo de espera, então aplicar o *tinkering* a eles pode ser mais desafiador. Alguns *kits* de construção computacional estão presos a um paradigma antigo, de uma época em que a computação era lenta e era necessário esperar para ver os resultados do programa. Nós criamos nossos *kits* para que ofereçam *feedback* imediato, de modo que você consiga ver os resultados de suas ações imediatamente, além de ver uma representação do processo conforme ele acontece.

Ver o resultado

No Scratch, é possível simplesmente clicar em um bloco de programação e ver o que acontece. Não há etapas separadas para compilar, nem modos separados para editar. Para a maioria dos blocos, os resultados são imediatos, como um movimento, uma mudança de cor ou um efeito sonoro. Para descobrir o que os blocos fazem, basta testar. Você pode até clicar em um bloco enquanto ele ainda está na paleta, sem precisar arrastá-lo para a área de *scripts* (onde os blocos são conectados em *scripts*).

Idealmente, o *tinkering* deve ser um processo contínuo. No Scratch, é possível continuar o processo de *tinkering* com *scripts* do Scratch mesmo enquanto eles estão sendo executados. Por exemplo, você pode começar clicando em um *script* do Scratch para fazer um ator se mover pela tela e, em seguida, clicar em outros *scripts* para que eles sejam executados em paralelo (por exemplo, adicionando uma trilha sonora ou animando um ator), enquanto o primeiro *script* ainda está sendo executado. Você também pode modificar um *script* enquanto ele está sendo executado (por exemplo, alterando o número de um bloco *mova* para aumentar a velocidade de um ator), ou inserir um novo bloco no *script*. Essa capacidade de ver mudanças em tempo real permite testar várias possibilidades e ver os resultados imediatamente, no contexto de um programa em execução. A meta é fazer as pessoas sentirem que podem interagir com os blocos de programação da mesma forma que interagem com objetos físicos.

O MaKey MaKey também é feito para oferecer *feedback* imediato. Digamos que você criou um teclado usando massa de modelar para controlar um personagem de videogame. Assim que você toca a massa de modelar (para completar o circuito), o personagem se move na tela. Também é possível testar a placa de circuito do MaKey

MaKey antes de criar qualquer interface. A placa em si tem quadros condutores que permitem completar o circuito com seus dedos. Se você tocar o quadro "terra" (também chamado de "base") com um dedo e o quadro "espaço" com outro dedo, o MaKey MaKey fará o computador pensar que a barra de espaço do teclado foi pressionada.

Ver o processo

Na maioria dos ambientes de programação, não é possível observar diretamente as propriedades internas do programa enquanto ele está sendo executado. Essas propriedades incluem o que o programa está fazendo no momento (o local atual da execução) e o que o programa "sabe" (os valores e variáveis e outras estruturas de dados). Assim como às vezes é mais fácil entender o que um carro tem de errado olhando o capô (ou, pelo menos, vendo as luzes do painel), é mais fácil corrigir *bugs* de um programa de computador quando ele está sendo executado e você está vendo suas propriedades internas.

O MaKey Makey tem alguns indicadores simples do seu processo interno: as luzes LED da placa indicam quando um circuito está completo, para que você possa depurar a placa e o circuito independentemente do que esteja acontecendo na tela do computador.

O Scratch tem uma variedade de recursos que permitem monitorar programas enquanto eles estão sendo executados. Os *scripts* do Scratch sempre ficam em destaque quando estão sendo executados, de modo que você consegue ver qual código está sendo executado e quando. Se você selecionar o modo *etapas únicas*, cada bloco individual de um *script* ficará em destaque quando for executado, permitindo análises e depurações mais detalhadas.

A interface do Scratch também tem monitores opcionais que permitem ver os valores atuais de dados armazenados em variáveis e listas. Na maioria das outras linguagens de programação, as estruturas de dados não podem ser visualizadas. Mas, no Scratch, estruturas de dados são visíveis e manipuláveis, o que melhora a adequação ao *tinkering*. É possível ver variáveis e listas serem atualizadas em tempo real enquanto o Scratch é executado. Também é possível digitar diretamente em monitores de lista, modificando os valores dos itens de uma lista, mesmo enquanto um programa está sendo executado. A meta é permitir que Scratchers apliquem o *tinkering* aos dados e tenham

uma compreensão melhor de como os dados estão relacionados ao restante do programa.

5.2 Experimentação fluida

O processo de *tinkering* é inerentemente iterativo. Quem pratica o *tinkering* começa explorando e testando para depois revisar e refinar metas, planos e criações. Em seguida, elas iniciam um novo ciclo de exploração e testagem, para depois revisar e refinar, várias e várias vezes. Quanto mais rápida for a iteração, mais rápida será a geração e o refinamento de ideias. Para apoiar esse tipo de interação, criamos nossos *kits* de construção para que fosse fácil para as pessoas começarem a testar ─ e continuarem testando, conectando (ou desconectando e reconectando) objetos dentro do projeto.

Fácil de começar

Um dos maiores desafios do *tinkering* com ferramentas tecnológicas é o tempo necessário para começar. Quando você tem uma ideia que deseja expressar, ou alguns materiais que quer testar, é normal querer ir direto ao ponto, sem gastar muito tempo com a preparação. Projetos eletrônicos normalmente precisam de uma estrutura básica para serem preparados (como a fiação de uma placa de ensaio), antes de começarem a interagir com partes novas. Da mesma forma, na programação, costumam ser necessários alguns códigos de preparação antes de começar a expressar novas ideias. Para permitir a experimentação fluida, criamos nossos *kits* de modo a minimizar ao máximo esses processos de preparação.

O Scratch permite testar coisas logo após a inicialização. Há um personagem padrão (o gato do Scratch), com algumas mídias prontas para usar: duas imagens que formam uma animação do gato caminhando e um som de "miau". É possível começar a programar comportamentos para o gato imediatamente: clicar no bloco *mova* para fazer o gato se mover; clicar no bloco *próxima fantasia* para animá-lo; clicar no bloco *toque o som* para o gato miar. Os blocos começam com valores padrões razoáveis de entrada, para que seja possível usá-los imediatamente, sem preencher nenhuma entrada.

O MaKey MaKey não exige configuração no computador, porque é um dispositivo

USB *plug-and-play* que aparece no computador como um mouse e teclado padrão. O sistema funciona com qualquer *software* ou site que responde a comandos de teclado e mouse; não é necessário nenhum *software* especial. A placa do circuito MaKey MaKey tem um arranjo fixo de entradas mapeadas nos comandos do teclado e do mouse, então ela também não precisa ser configurada antes de começar. É possível conectar a placa a um computador e começar a criar circuitos (e teclas) imediatamente. É possível começar a fazer experimentos com diferentes materiais físicos e criar sua própria interface física, sem nenhum tipo de preparação.

Fácil de conectar

A adequação de um *kit* de construção ao *tinkering* é determinada, em boa parte, pela forma como as partes do *kit* de construção se conectam umas às outras. Durante a criação de conectores para nossos *kits* de construção, nós nos inspiramos em peças de LEGO: elas são fáceis de juntar e separar, além de terem a quantidade certa de "agarramento" para que as estruturas sejam resistentes. Essa junção cuidadosa é o que torna os *kits* de LEGO tão adequados para o *tinkering*. Blocos de madeira comuns são fáceis de empilhar, mas também são fáceis de derrubar. No outro extremo, conjuntos Erector são bons para criar estruturas resistentes, mas elas também acabam sendo mais difíceis de juntar e separar. As peças de LEGO são um meio-termo, permitindo experimentos rápidos e iterativos.

No *kit* do MaKey MaKey, as garras jacaré são usadas para conectar interruptores caseiros à placa de circuito. As garras são fáceis de usar e podem testar diferentes conectores com trocas rápidas, mas oferecem força mecânica suficiente para aguentar a tensão aplicada a um controle de videogame ou durante uma apresentação musical. Isso faz com que elas sejam mais adequadas para o *tinkering* do que outros conectores eletrônicos típicos, como barras de pinos fêmeas (pequenas entradas para cabos das quais eles podem sair com facilidade) ou conectores borne (que seguram fios com firmeza, mas que precisam de uma chave de fenda para serem abertos e fechados).

No Scratch, o formato dos blocos de programação limita a forma como eles se juntam. Quando você tira um bloco da paleta, o formato dele deixa imediatamente visível com quais outros blocos ele pode ser conectado. Diferente de outras linguagens de programação tradicionais baseadas em texto, não há uma sintaxe obscura (ponto e

vírgula, colchetes, etc.) que precisa ser aprendida. Em vez disso, a gramática é visual e indicada pelo formato dos blocos e conectores. Os blocos só se juntam se a combinação fizer sentido. É claro que os blocos podem não se comportar como você esperava, mas não há erros de sintaxe. Se os blocos se juntam, eles certamente serão executados.

Blocos que recebem entradas têm "tomadas" de diferentes formatos, que indicam o tipo de bloco que deve ser conectado. Por exemplo, o bloco *mova* tem uma tomada oval, o que indica que ele aceita entradas numéricas. É possível inserir um bloco oval, como o bloco que informa a posição y do cursor do mouse. Os blocos condicionais *se, se-então* e *espere até que* têm uma entrada hexagonal, o que indica que aceitam valores booleanos (verdadeiro-falso) como entrada. Por exemplo, é possível inserir um bloco hexagonal *tocando em?* que informa (verdadeiro ou falso) se o ator está tocando em outro ator (ver Figura 1). Alguns blocos são mais tolerantes. Por exemplo, o bloco *diga* tem um formato quadrado que indica que ele aceita qualquer tipo de entrada (um número, uma *string* ou um valor booleano).

Os formatos e os conectores facilitam o *tinkering* e os experimentos com os blocos de programação do Scratch, assim como as peças de LEGO. Você pode juntar os blocos, testá-los e desconectá-los para testar outras combinações. O custo dos experimentos é baixo. Além disso, você pode deixar alguns blocos (ou pilhas de blocos) na área de trabalho (ver Figura 2), caso queira usá-los depois, como faria com peças de LEGO. Diferentemente da maioria dos ambientes de programação, em que é necessário manter a área de trabalho organizada.

5.3 Exploração aberta

No entanto, apoiar o *feedback* imediato e a experimentação fluida não é suficiente. Também é importante permitir e inspirar as pessoas a testar uma grande variedade de possibilidades. Para isso, os *kits* de construção precisam ser compatíveis com uma ampla variedade de materiais e gêneros.

Variedade de materiais

O Scratch vem com uma grande biblioteca de mídias para inspirar novas ideias de projetos. Essas mídias incluem imagens e cenários, efeitos sonoros e loops de música,

além de atores com *scripts* de programação e mídias incorporadas. O Scratch também oferece diversas maneiras de o usuário criar e importar sua própria mídia. Você pode criar imagens com o editor de imagens da plataforma, tirar fotos com a câmera do seu computador, ou gravar sons com o gravador da plataforma. Também é possível importar imagens e sons do seu disco rígido ou da Web; basta arrastá-los e soltá-los no Scratch.

Mais importante, o Scratch fornece acesso a uma biblioteca cada vez maior de projetos criados por outros membros da comunidade Scratch. Você pode usar imagens, sons e *scripts* de outros projetos e integrá-los ao seu. Tudo o que é compartilhado no site do Scratch é protegido pela licença de compartilhamento Creative Commons, de modo que os membros da comunidade podem emprestar coisas livremente, contanto que insiram os créditos do criador ou criadora. Aproximadamente um terço dos 2,5 milhões de projetos dos sites do Scratch são reformulações, nas quais um membro da comunidade complementa o trabalho de outro. O site serve como uma fonte contínua de inspiração, com milhares de novos projetos compartilhados todos os dias.

Diferentemente do Scratch, o MaKey MaKey não vem com materiais (exceto pela placa de circuito e os conectores). Em vez disso, ele foi criado para que os usuários vejam “o mundo” como seu *kit* de construção. É possível fazer circuitos e interruptores MaKey MaKey a partir de qualquer coisa que conduza eletricidade: alimentos, plantas, desenhos a lápis, papel alumínio, água, massa de modelar e até seu próprio corpo. Você pode usar qualquer objeto ou material (isolante ou condutor) para criar estruturas para suportar o circuito, de papelão a bolas de praia, de baldes a chapéus.

Variedade de gêneros

No início do processo, quem pratica o *tinkering* costuma não ter uma ideia clara do que quer fazer. Por isso, são úteis *kits* de construção que possibilitem projetos de diferentes gêneros. Esses *kits* devem oferecer flexibilidade para a troca de gênero conforme os usuários adaptam e revisam iterativamente suas criações ao longo do tempo.

Diferentemente de vários *kits* de construção de *software* (como Gamemaker e Gamestar Mechanic ─ que, como os próprios nomes sugerem, focam explicitamente em jogos de videogame), o Scratch permite a criação uma ampla variedade de tipos de projetos, incluindo histórias interativas, jogos, animações, simulações, arte e música.

Diferentes partes do *kit* de construção do Scratch são compatíveis com diferentes gêneros. O editor de imagens e os blocos de imagem e efeito são compatíveis com animações; os blocos *diga* e *pense* (para balões de diálogo) são compatíveis com histórias; blocos de detecção de colisão e de interação com o teclado são compatíveis com jogos; blocos de caneta (para fazer os atores desenharem conforme eles se movem) são compatíveis com arte interativa; blocos de operações matemáticas são compatíveis com simulações; e blocos de notas, instrumentos e ritmo são compatíveis com música. Como essas ferramentas fazem parte do mesmo *kit*, um único projeto do Scratch consegue misturar múltiplos gêneros. Um praticante da exploração criativa, o *tinkerer*, pode começar trabalhando com um tipo de projeto e depois decidir alterá-lo para um gênero diferente (ou uma combinação de gêneros) conforme evolui; por exemplo, transformando uma animação em um jogo ou uma simulação em uma arte interativa.

O MaKey MaKey também é compatível com criações de vários gêneros, porque pode ser usado para criar um controle para qualquer *software* que possa ser controlado com um teclado. Por exemplo, é possível criar um dispositivo de interface do MaKey Makey que controle um videogame, um sintetizador de som, um *video player*, um editor de texto, um navegador Web, um programa de edição de imagens ou até um projeto do Scratch.

6. *Tinkering* e adequação ao *tinkering*

Neste artigo, delineamos parte do nosso pensamento atual sobre como criar *kits* de construção adequados ao *tinkering*. No entanto, a criação de *kits* de construção é apenas uma parte do que é necessário. Até o *kit* de construção mais adequado ao *tinkering* não será bem-sucedido se não for acompanhado pelos tipos certos de atividades, materiais de suporte, facilitação, espaço e comunidade. Resumindo, criar *contextos adequados ao tinkering* é tão importante quanto criar *kits* adequados ao *tinkering*.

Este artigo não se propõe a examinar essas questões de maneira mais profunda. Mas, para concluir, compartilhamos um breve resumo de algumas lições essenciais que aprendemos durante o processo de criação de contextos adequados ao *tinkering*. Estas ideias podem ser úteis para educadores que desejam apoiar jovens no processo de criação e *tinkering*.

● *Ter como foco o processo, não no produto*. Embora a criação seja uma parte importante do processo de *tinkering*, focar demais no produto final pode prejudicar a experimentação que está no cerne do *tinkering*. Uma forma de dar atenção ao processo é criar registros e discutir sobre os estágios intermediários, experimentos que não deram certo e fontes de inspiração.

● *Definir temas, não desafios*. Em vez de propor desafios a serem resolvidos (como é normal em oficinas de criação), proponha temas a serem explorados. Selecione temas de oficinas que sejam amplos o suficiente para dar a todos a liberdade de trabalhar em projetos com os quais as pessoas se importam, mas que sejam específicos o bastante para incentivar um senso de experiência compartilhada entre os participantes (Rusk, Resnick, Berg, & Pezalla-Granlund, 2008). Por exemplo, podemos pedir que os participantes de uma oficina criem um cartão interativo para comemorar uma data especial.

● *Destacar exemplos diversos*. Mostre exemplos de projetos que ilustram a ampla diversidade do que é possível fazer, fazendo com que pessoas pensem de maneira divergente. Deixe exibidos exemplos e documentos para ter fontes de inspiração sempre presentes.

● *Explorar o espaço*. Estude uma forma de reorganizar ou mudar o local, para abrir novas possibilidades de exploração e colaboração. Por exemplo, como podemos organizar as mesas e as telas para que as pessoas vejam os trabalhos umas das outras? Como podemos organizar os materiais para incentivar combinações interessantes e inusitadas?

● *Incentivar o engajamento com pessoas, não apenas com materiais*. Além de "conversar com o material", quem pratica o *tinkering* também pode se beneficiar conversando (e colaborando) com outras pessoas.

● *Apresentar perguntas em vez de respostas*. Resista à vontade de explicar demais e de resolver os problemas. Em vez disso, alimente o processo de exploração

fazendo perguntas, apontando fenômenos interessantes e sugerindo outras possibilidades.

● *Combinar aprofundamento e análises de etapas anteriores*. Embora seja produtivo o aprofundamento no processo do fazer durante o *tinkering*, também é importante dar um passo para trás e refletir sobre o processo.

Nossa meta é proporcionar a todos - de todas as idades, histórias e interesses - novas oportunidades para aprender por meio do *tinkering*. E para fazer isso bem, nós próprios precisamos continuar praticando (e estudando) o *tinkering*, fazendo experimentos lúdicos com novas formas de criar atividades adequadas ao *tinkering*. Este artigo é apenas um começo. Nosso plano é continuar praticando o *tinkering* e iterando ideias e tecnologias, refinando nosso pensamento sobre a natureza do *tinkering* e as estratégias para aprimorá-lo.

Agradecimentos

Vários membros do grupo de pesquisa Lifelong Kindergarten do MIT Media Lab contribuíram com as ideias e projetos discutidos neste artigo. Agradecemos a Seymour Papert e Sherry Turkle, que influenciaram profundamente a forma como pensamos sobre *tinkering* e aprendizagem.

Referências bibliográficas

Brennan, K., & Resnick, M. (2012). Imagining, creating, playing, sharing, reflecting: How online community supports young people as designers of interactive media. In N. Lavigne & C. Mouza (org.), *Emerging technologies for the classroom: A learning sciences perspective*. Nova York: Springer.

Brown, J. S., Collins, A., & Duguid, P. (1989). Situated cognition and the culture of learning. *Educational Researcher*, *18*(1), p. 32–42.

Dewey, J. (1938). *Experience and Education*. Nova York: Simon & Schuster.

Hancock, C. (2003). *Real-time programming and the big ideas of computational literacy*.

Dissertação de Doutorado. MIT Media Lab, Cambridge, MA.

Levi-Strauss, C. (1966). *The savage mind*. Chicago: University of Chicago.

Maloney, J., Resnick, M., Rusk, N., Silverman, B., & Eastmond, E. (2010). The Scratch programming language and environment. *ACM Transactions on Computing Education (TOCE)*, *10*(4).

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. Nova York: Basic Books.

Papert, S. (1993). *The children's machine: Rethinking school in the age of the computer*. Nova York: Basic Books.

Resnick, M. (2007). All I really need to know (about creative thinking) I learned (by studying how children learn) in kindergarten. Trabalho apresentado na conferência ACM Creativity & Cognition, Washington DC, junho de 2007. *Proceedings of the ACM Creativity & Cognition Conference*. Nova York: ACM.

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

Rusk, N., Resnick, M., Berg, R., & Pezalla-Granlund, M. (2008). New pathways into robotics: Strategies for broadening participation. *Journal of Science Education and Technology*, *17*(1), p. 59–69.

Silver, J., & Rosenbaum, E. (2012). Makey Makey: Improvising tangible and nature-based user interfaces. Trabalho apresentado na 6ª International Conference on Tangible, Embedded and Embodied Interaction, Queen's Human Media Lab, Kingston, Ontario, 19 a 22 de fevereiro de 2012. *Proceedings of the International Conference on Tangible, Embedded and Embodied Interaction*. Nova York: ACM.

Schoen, D. (1983). *The reflective practitioner: How professionals think in action*. Londres: Maurice Temple Smith Ltd.

Turkle, S., & Papert, S. (1990). Epistemological pluralism. *Signs*, *16*(1).