UNIVERSIDADE ESTADUAL DA PARAÍBA
CAMPUS I – CAMPINA GRANDE
CENTRO DE CIÊNCIAS E TECNOLOGIAS
CURSO DE LICENCIATURA EM COMPUTAÇÃO

FLÁVIO MEIRA LIMA

O SOFTWARE EDUCACIONAL SCRATCH

POSSIBILIDADES PARA UMA APRENDIZAGEM SIGNIFICATIVA NA ERA DIGITAL

CAMPINA GRANDE–PB
2015

FLÁVIO MEIRA LIMA

O SOFTWARE EDUCACIONAL SCRATCH

POSSIBILIDADES PARA UMA APRENDIZAGEM SIGNIFICATIVA NA ERA DIGITAL

Monografia apresentada ao Curso de Licenciatura em Computação da Universidade Estadual da Paraíba em cumprimento à exigência para obtenção do Grau de Licenciado no curso de Licenciatura em Computação.

Orientador: MSc. Antonio Carlos de Albuquerque

CAMPINA GRANDE-PB
2015

L732d Lima, Flávio Meira.
O software educacional Scratch [manuscrito] : possibilidades para uma aprendizagem significativa na era digital / Flávio Meira Lima. - 2015.
74 p. : il. color.

Digitado.
Trabalho de Conclusão de Curso (Graduação em Computação) - Universidade Estadual da Paraíba, Centro de Ciências e Tecnologia, 2015.
"Orientação: Prof. Me. Antonio Carlos de Albuquerque, Departamento de Computação".

1. Software educacional. 2. Scratch. 3. Fluência digital. 4. Lógica. 5. Programação. I. Título.

21. ed. CDD 005.13

FLAVIO MEIRA LIMA

O Software Educacional Scratch - Possibilidades para uma Aprendizagem Significativa na Era Digital

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Licenciatura plena em Computação da Universidade Estadual da Paraíba, em cumprimento à exigência para obtenção do grau de Licenciado em Computação.

Aprovada em 01 de Dezembro de 2015.

Prof. Me. Antônio Carlos Albuquerque (UEPB)
Orientador(a)

Profª. Me. Maria Lúcia Serafim (UEPB)
Examinador(a)

Prof. Me. Adriano Araújo Santos (FACISA)
Examinador(a)

DEDICATÓRIA

Ao meu Deus, por permitir tornar esse momento possível. Aos meus pais pela educação que me proporcionaram e a formação do meu caráter. As minhas filhas e aos meus netos, tesouros da minha vida. Aos meus irmãos e aos amigos inseparáveis que sempre me incentivaram nos momentos mais difíceis para que esse sonho fosse realizado.

Feliz aquele que transfere o que sabe e aprende o que ensina.

(CORALINA[1], 2007)

[1] **Trecho do poema "Exaltação de Aninha (O Professor)"**

AGRADECIMENTOS

A todos os meus mestres que me dedicaram um pouco do seu tempo nessa tarefa árdua que é o ensinar. E em especial, ao meu orientador Antonio Carlos de Albuquerque por todo apoio recebido e pela sua luta incansável em defesa do nosso curso. Ao professor Adriano Araújo dos Santos e a professora Maria Lúcia Serafim que me honraram como examinadores quando da minha defesa deste trabalho.

RESUMO

Este trabalho tem por objetivo apresentar o software educacional *SCRATCH* e analisar as possibilidades para catalisar uma aprendizagem significativa tão necessária para fluência digital no século XXI. Tem por fundamentação teórica a abordagem Construtivista de Jean Piaget; o Construcionismo de Seymour Paper; o pensamento computacional e as ideias inovadoras e criativas propostas por Mitchel Resnick. Foi aplicada a metodologia de pesquisas bibliográficas e ao final foram desenvolvidos alguns projetos de exemplos que podem ser aplicados em sala de aula de forma colaborativa e interdisciplinar. Destina-se, portanto, a professores mais notadamente do ensino fundamental e médio; iniciantes em cursos de computação ou qualquer pessoa sem nenhum conhecimento em linguagens de programação. Pretendeu-se dessa forma, contribuir para a formação de cidadãos críticos, criativos e reflexivos ao desenvolver o raciocínio lógico-matemático e computacional através do software.

Palavras chave: Construtivismo, Construcionismo, Fluência Digital, Lógica, Programação.

ABSTRACT

This work aims to present the educational software SCRATCH and examine ways to catalyze meaningful learning so necessary for digital fluency in the twenty-first century. Its theoretical foundation the Constructivist approach of Jean Piaget; constructionism Seymour Paper; computational thinking and innovative and creative ideas proposed by Mitchel Resnick. The methodology of bibliographic research and in the end were developed some examples of projects that can be applied in classroom collaborative and interdisciplinary approach was applied. It is intended, therefore, most notably primary and secondary school teachers; beginners in computer courses or anyone with no knowledge of programming languages. It was intended that way, contribute to the formation of critical, creative and reflective citizens to develop logical-mathematical reasoning and computer through the software.

Keywords: Constructivism, Constructionism, Digital Fluency, Logic, Programming

LISTA DE ILUSTRAÇÕES

SUMÁRIO

INTRODUÇÃO

A aprendizagem é um processo contínuo que permeia por toda vida do ser humano desde o seu nascimento. E nessa interação com o meio o ser humano vai adquirindo novas formas de pensar e agir modificando o próprio meio. Com o advento das novas Tecnologias da Informação e Comunicação – TIC, o homem vivencia a cada dia a necessidade de adquirir cada vez mais novos conhecimentos. E é essa capacidade inventiva do ser humano que faz surgir a cada momento um novo equipamento eletrônico, um novo software. Celulares, *notebooks, tablets, iphone, ipad, smartphone*, lousa digital, TV digital, são artefatos comuns na era digital do século XXI. Mas, ao mesmo tempo em que surgem novas tecnologias, surgem também novas preocupações. Ou seja, o antes chamado analfabeto na leitura e na escrita, hoje é chamado de analfabeto digital aquele que não se incorporar nesses novos meios de comunicação. Segundo (PRENSKY, 2001), o mundo é dividido pelos nativos e os imigrantes digitais. Os nativos, são aqueles que já nasceram em um mundo submerso pelas novas TIC; conseguem realizar várias tarefas ao mesmo tempo, encaram o mundo virtual como uma extensão do mundo real e aprendem de forma não linear. Já os imigrantes, são as pessoas que nasceram em um período anterior ou no início das novas tecnologias; encontram dificuldades ou não possuem hábito em adaptar-se nessa nova realidade, pois aprenderam de forma linear (começo, meio e fim). Por isso, é importante que os professores como imigrantes, pensem novos modelos metodológicos de ensino-aprendizagem que atendam a demanda dos nativos digitais.

Surge então a necessidade de ser fluente digital. Assim como se aprende uma nova língua, para ser fluente, não basta saber ler um cardápio ou perguntar onde fica uma determinada rua, mas saber não só ler e escrever como também interpretar. A linguagem computacional segue o mesmo princípio, ser fluente digital requer uma capacidade mínima em ler, escrever e interpretar códigos de programação de computador. Sem essa fluência, educador e educando serão meros consumidores de tecnologia.

Esse trabalho tem por objetivo apresentar o software educacional *SCRATCH*, não como uma ferramenta de apoio na iniciação em linguagem de programação. Pode até ser, mas principalmente a desenvolver o raciocínio lógico-matemático do usuário a partir de construções simples da lógica computacional. O objetivo maior é tornar o cidadão mais criativo, crítico, reflexivo, construtor do conhecimento e não apenas um sujeito passivo (consumidor). Neste sentido, o público alvo deste trabalho vai desde professores de todas as

áreas, alunos de computação ou não, como também aos pais ou a qualquer pessoa que se interessar sem um mínimo de conhecimento em linguagem de programação. Na verdade, utilizar-se do *Scratch,* se aprende com diversão. Como um passatempo. Como diz RESNICK (2007), o chão é baixo, e o teto é alto. Vale salientar que este trabalho não é um manual de instruções nem um curso na ferramenta. O objetivo principal é analisar as possibilidades do *software* e orientar para quem deseja se aprofundar.

A fundamentação teórica tem por base as contribuições para educação de Jean Piaget, Seymour Papert e Mitchel Resnick. A metodologia baseia-se em pesquisas bibliográficas, sites educacionais que se utilizam da ferramenta e principalmente o ambiente corporativo dos desenvolvedores do *Scratch* e a partir disso serão apresentado o software e suas funcionalidades mais básicas com desenvolvimento de algumas aplicações práticas. Neste sentido, pretende-se mostrar se o software educacional *Scratch* atende essas expectativas, ou seja, se é realmente uma ferramenta de fácil uso e catalisadora no processo de ensino-aprendizagem de forma colaborativa, interdisciplinar e significativa para o século XXI.

1 FUNDAMENTAÇÃO TEÓRICA

O processo de aprendizagem pode ser definido de forma sintética como o modo como os seres humanos adquirem novos conhecimentos, desenvolvem competências e mudam o comportamento. O estudo desse desenvolvimento constitui uma área da Psicologia concentrando seus esforços em compreender o homem desde o nascimento até o seu mais completo grau de estabilidade e maturidade.

A fundamentação teórica deste trabalho tem por base a teoria Construtivista de Jean Piaget; o Construcionismo de Seymour Papert; os conceitos inovadores para uma aprendizagem criativa propostas por Mitchel Resnick e o Pensamento Computacional.

1.1 O Construtivismo

Na abordagem educacional, o Construtivismo é uma das correntes teóricas empenhadas em explicar como a inteligência humana se desenvolve partindo do princípio de que o desenvolvimento da inteligência é determinado pelas ações mútuas entre o indivíduo e o meio.

A ideia é que o homem não nasce inteligente, mas também não é passivo sob a influência do meio, isto é, ele responde aos estímulos externos agindo sobre eles para construir e organizar o seu próprio conhecimento, de forma cada vez mais elaborada. Entre os estudiosos dessa corrente teórica, destaca-se Jean Piaget como o mais importante deles.

> [...] o conhecimento não procede, em suas origens, nem de um sujeito consciente de si mesmo, nem dos objetos já constituídos (do ponto de vista do sujeito) que se lhe imporiam: resultaria de interações que se produzem a meio caminho entre sujeito e objeto, e que dependem, portanto, dos dois ao mesmo tempo, mas em virtude de uma indiferenciação completa e não de trocas entre formas distintas (PIAGET, 1990, p. 8).

1.2 Jean Piaget

Jean Piaget (Neuchâtel, 9 de agosto de 1896 – Genebra, 16 de setembro de 1980) foi um biólogo, zoólogo, psicólogo, filósofo e epistemólogo suíço, considerado um dos maiores pensadores do século XX. Estudou inicialmente biologia na universidade de Neuchâtel, onde concluiu seu doutorado, e posteriormente se dedicou a área de Psicologia, Epistemologia e Educação. Durante sua vida, Piaget escreveu mais de 75 livros e centenas de trabalhos científicos. Suas pesquisas o levaram da Biologia à Filosofia e Psicologia. Defendeu uma abordagem interdisciplinar para a investigação epistemológica e fundou a Epistemologia

Genética, teoria do conhecimento com base no estudo da gênese psicológica do pensamento humano.

Revolucionou as concepções de inteligência e de desenvolvimento cognitivo através de observações com seus filhos, e principalmente com outras crianças derrubando várias teorias tradicionais relacionadas à aprendizagem. Suas teorias buscam implantar nos espaços de aprendizagem uma metodologia inovadora que busca formar cidadãos criativos e críticos. De acordo com suas teorias, o professor não deve apenas ensinar, mas sim e antes de tudo, orientar o educando no caminho da aprendizagem autônoma.

1.3 A Epistemologia Genética de Jean Piaget e a construção do conhecimento

Sua Epistemologia Genética defende que o indivíduo passa por várias etapas de desenvolvimento ao longo de sua vida. Para Piaget, a aprendizagem é um processo que começa no nascimento e acaba na morte. A aprendizagem dá-se através do equilíbrio entre a assimilação e a acomodação, resultando em adaptação. Segundo este processo, o ser humano assimila os dados que obtém do exterior, mas uma vez que já tem uma estrutura mental que não está "vazia", precisa adaptar esses dados à estrutura mental já existente (esquema) modificando ou adicionando novos esquemas. Uma vez que os dados são adaptados a si, dá-se a acomodação. Ainda segundo Piaget, o pleno desenvolvimento da personalidade sob seus aspectos mais intelectuais é indissociável do conjunto das relações afetivas, sociais e morais que constituem a vida da instituição educacional.

1.3.1 Os estágios cognitivos segundo Piaget

Piaget, quando descreve a aprendizagem, tem um enfoque diferente do que normalmente se atribui a esta palavra. Piaget separa o processo cognitivo inteligente em duas palavras: aprendizagem e desenvolvimento. Para Piaget, segundo MACEDO (1994), a aprendizagem refere-se à aquisição de uma resposta particular, aprendida em função da experiência, obtida de forma sistemática ou não. Enquanto que o desenvolvimento seria uma aprendizagem de fato, sendo este o responsável pela formação dos conhecimentos.

Piaget, quando postula sua teoria sobre o desenvolvimento da criança, descreve-a, basicamente, em 4 estados, que ele próprio chama de fases de transição (PIAGET, 1975). Essas 4 fases são :

- Sensório-motor (0 – 2 anos);
- Pré-Operatório (2 – 7 anos);
- Das operações concretas (7 – 11/12, anos);
- Das operações formais (11/12 – 14/15 anos);

1.3.1.1 Sensório-motor

A interação com o mundo é quase que exclusivamente feita por movimento e percepção. A linguagem não existe e o mundo não tem representação interna. O tempo não existe. É o aqui e agora no seu sentido mais literal. A mãe só existe enquanto visível ou audível. Não tem o conceito de objeto (algo com permanência e identidade). É a fase do egocentrismo; a criança é o mundo. Nesta fase ela elabora o pequeno número de esquemas com que nasceu. Desenvolve as seguintes habilidades fundamentais ao término dessa fase:

Linguagem: Seu início implica na capacidade de manipular símbolos.

Conceito do objeto: Pode representar sua identidade e permanência.

Coordenação de atividade: Pegar uma caneta implica em coordenar o visual com o motor.

Casualidade: Se A causa B, então pode fazer A para obter B (intencionalidade)

1.3.1.2 Pré-Operatório

É nesta fase que surge na criança, a capacidade de substituir um objeto ou acontecimento por uma representação (PIAGET e INHELDER, 1982), e esta substituição é possível, conforme Piaget, graças à função simbólica. Assim este estágio é também muito conhecido como o estágio da Inteligência Simbólica.

A criança deste estágio:

- É egocêntrica, centrada em si mesma, e não consegue se colocar, abstratamente, no lugar do outro.
- Não aceita a idéia do acaso e tudo deve ter uma explicação (é fase dos "por quês").
- Já pode agir por simulação, "como se".
- Possui percepção global sem discriminar detalhes.
- Deixa se levar pela aparência sem relacionar fatos.

Exemplos:

Mostram-se para a criança, duas bolinhas de massa iguais e dá-se a uma delas a forma de salsicha. A criança nega que a quantidade de massa continue igual, pois as formas são diferentes. Não relaciona as situações.

1.3.1.3 Operatório concreto

Desenvolvimento do pensamento lógico sobre coisas concretas. Compreensão das relações entre as coisas e capacidade para classificar objetos. Superação do egocentrismo da linguagem. Aparecimento das noções de conservação de substância, peso, volume, tempo, velocidade. Um importante conceito desta fase é o desenvolvimento da reversibilidade, ou

seja, a capacidade da representação de uma ação no sentido inverso de uma anterior, anulando a transformação observada.

Exemplos:

Despeja-se a água de dois copos em outros, de formatos diferentes, para que a criança diga se as quantidades continuam iguais. A resposta é afirmativa uma vez que a criança já diferencia aspectos e é capaz de "refazer" a ação.

1.3.1.4 Operacional formal

Desenvolvimento da capacidade para construir sistemas e teorias abstratos, para formar e entender conceitos abstratos como: amor, justiça, democracia. Do pensamento concreto sobre coisas passa para o pensamento abstrato, "hipotético-dedutivo", isto é, o indivíduo se torna capaz de chegar a conclusões a partir de hipóteses: se A é maior que B e B é maior que C, A é maior que C.

1.4 Seymour Papert, o Construcionismo e o LOGO

Seymour Papert nasceu em 1928 em Pretória, África do Sul. PhD em Matemática. Sua formação deu-se na Universidade de Cambridge, onde desenvolveu trabalhos de pesquisa em matemática de 1954 a 1958. Optou pelo doutorado na mesma área, devido ao seu grande interesse a ela direcionado. Trabalhou e conviveu com Jean Piaget na University of Geneva de 1958 a 1963.

Seu principal objetivo era considerar o uso da matemática a fim de entender como as crianças podem aprender e pensar. É considerado um dos pais do campo da inteligência artificial (IA), sendo internacionalmente reconhecido como um dos principais pensadores sobre as formas pelas as quais a tecnologia pode modificar a aprendizagem. É autor de Mindstorms: children computers and powerful ideas (1980) e The children's machine: rethinking school in the age of the computer (1992). Também, publicou inúmeros artigos sobre matemática, inteligência artificial, educação, aprendizagem e raciocínio.

No início dos anos 1960, Papert afiliou-se ao Instituto de Tecnologia de Massachusetts (Massachusetts Institute of Technology-MIT) e, juntamente com Marvin Minsky, fundou o Laboratório de Inteligência Artificial (Media Lab).

O MIT, centro universitário de educação e pesquisa privado localizado em Cambridge, Massachusetts, nos Estados Unidos, é um dos líderes mundiais em ciência e tecnologia, além de outros campos, como administração, economia, linguística, ciência política e filosofia. Dentre os professores e ex-alunos do MIT estão incluídos vários políticos, executivos,

escritores, astronautas, cientistas e inventores proeminentes. O MIT já produziu mais de 70 Prêmios Nobel, oito dos quais são membros do seu corpo docente atual.

Na educação, Papert cunhou o termo Construcionismo, como sendo a abordagem do Cosntrutivismo que permite ao educando construir o seu próprio conhecimento por intermédio de alguma ferramenta, como o computador por exemplo. Desta forma, o uso do computador é defendido como auxiliar no processo de construção de conhecimentos, uma poderosa ferramenta educacional, adaptando os princípios do Construtivismo cognitivo de Jean Piaget a fim de melhor aproveitar-se o uso de tecnologias. Em plena década de 60, quando os computadores eram muito limitados (não existia a interface gráfica e muito menos a internet), além de restritos, por se tratar de uma tecnologia bastante cara à época, Papert já dizia que toda criança deveria ter um computador em sala de aula. Sua atitude visionária fazia sua teoria parecer ficção científica. Em sua teoria, propõe-se a explicar as relações aprendiz-computador para produzir o máximo de aprendizagem com o mínimo de ensino. Isto, porém, não significa deixar que a criança aprenda sem qualquer auxílio. O Construcionismo busca meios de aprendizagem que valorizem a construção das estruturas cognitivas do sujeito a partir de suas ações, apoiadas em suas próprias construções de mundo.

> Embora a tecnologia desempenhe um papel essencial na realização de minha visão sobre o futuro da educação, meu foco central não é a máquina, mas a mente e, particularmente, a forma em que movimentos intelectuais e culturais se auto-definem e crescem (PAPERT, 1988, p. 23).

A característica fundamental do conceito construcionista é o uso do computador na realização de construções concretas, visível na tela da máquina, e atua como fonte de ideias para o desenvolvimento de construções cognitivas, criando um movimento dialético entre o concreto e o abstrato (ALMEIDA, 2000).

A Teoria Construcionista é uma forma de conceber e utilizar o computador na educação, envolvendo o aluno, o professor e os recursos computacionais, constituindo um ambiente de aprendizagem no qual o computador se torna um elemento de interação que propicia o desenvolvimento da autonomia do aluno, não direcionando a sua ação, mas auxiliando-o na construção de conhecimentos de distintas áreas do saber. (ALMEIDA, 1999, p. 29), por meio de exploração, experimentações, descobertas e reflexão.

O mundo complexo em que vivemos exige profissionais críticos, criativos, reflexivos, com capacidade de trabalhar em grupo. Certamente, essas atitudes não podem ser transmitidas ao sujeito, mas sim construídas e desenvolvidas por cada um. Por isso, a educação deve

oferecer condições para que o aluno vivencie situações que lhe permitam construir e desenvolver essas competências. Um ambiente pedagógico informatizado, aliado a práticas pedagógicas fundamentadas na Teoria Construtivista, pode auxiliar nesse processo (PAPERT, 1988; VALENTE, 1993).

No Construcionismo a aprendizagem é baseada em projetos, os alunos enquanto autores, deixam de ser criaturas do processo pedagógico (sujeitos passivos) e passam ao patamar de criadores (sujeitos ativos).

Embora, segundo alguns autores, o uso do computador na educação desempenhe uma função instrucionista (máquina de ensinar), Papert, ao contrário, defende e desenvolve uma metodologia de utilização do computador na educação em uma abordagem construcionista.

> No construcionismo, duas idéias se destacam: a primeira está em que o aprendizado se realiza por meio do fazer, ou seja, colocar a mão na massa; a segunda está no fato de que o aprendiz constrói algo do seu interesse, enfatizando o envolvimento afetivo, tornando a aprendizagem mais significativa (VALENTE, 1993).

Papert não evidencia as etapas estabelecidas por Piaget no desenvolvimento da inteligência da criança. Importa para ele como o desenvolvimento cognitivo acontece, como as ideias piagetianas podem contribuir para uma aprendizagem mais efetiva na construção do conhecimento. Assim, ele propõe o uso do computador como ferramenta educacional, construindo situações de aprendizagens que possibilitem acelerar as etapas do desenvolvimento (MARCHI, 2001).

Nesta visão, considera-se a importância do significado dos conhecimentos a serem construídos pelo sujeito. Sendo assim, admite-se, ainda, a importância do aspecto afetivo e as conexões estabelecidas entre o que está aprendendo com o que já conhece. Nesse processo, destaca-se o mecanismo de equilibração. Os desafios propostos provocam os desequilíbrios necessários para a construção de novas estruturas, resultando em novos conhecimentos (PAPERT, 1988; PIAGET, 1964).

O termo construcionismo é esclarecido por Valente (1999) como:

> [...] a construção de conhecimento baseada na realização concreta de uma ação que produz um produto palpável (um artigo, um projeto, um objeto) de interesse pessoal de quem produz. Contextualizada, no sentido do produto ser vinculado à realidade da pessoa [...] (VALENTE, 1999, p. 141).

Nesta direção, o aprendiz faz uso do

> [...] computador para construir o seu conhecimento, o computador passa a ser uma máquina para ser ensinada propiciando condições para o aluno descrever a resolução de problemas usando linguagens de programação, refletir sobre os resultados obtidos, depurar suas idéias por intermédio da busca de novos conteúdos e novas estratégias. [...] A construção do conhecimento que advém do fato de o aluno ter de buscar novos conteúdos e estratégias para incrementar o nível de conhecimento que já dispõe sobre o assunto que está sendo tratado via computador (VALENTE, 1999, p. 3).

No paradigma construcionista, a ênfase está na aprendizagem ao invés de estar no ensino (VALENTE, 1993). Esta observação é de fundamental importância para se entender as novas relações entre aluno/objeto/professor que devem acontecer nesse novo paradigma educacional.

O computador é, comumente, utilizado nas escolas como um instrumento que facilita a aquisição de informações e conhecimentos técnicos sobre a máquina. A proposta construcionista requer uma nova prática, uma nova visão e compreensão sobre o uso do computador como ferramenta educacional.

A proposta defendida por Papert utiliza o computador como ferramenta para a aprendizagem, por meio da qual o aluno resolve problemas utilizando suas estruturas cognitivas. As atividades propostas necessitam considerar os princípios de liberdade e autonomia. O sujeito constrói seu conhecimento, por meio de descobertas próprias, elaborando e reelaborando as situações-problema propostas. Nesse processo de reflexão para a superação do estado de desequilíbrio que pode ser ocasionado, ocorre a construção de novas estruturas cognitivas, sempre em um nível de conhecimento superior.

Em um ambiente informatizado, o professor desempenha a função de facilitador, encaminhando o processo para que cada aprendiz avance na construção do seu conhecimento. Assim, o aprendiz, utilizando-se do computador, interage por meio das intervenções e interações com o facilitador, possibilitando a construção do conhecimento.

Nessa direção, Papert (1994) destaca a existência de uma palavra que significa a arte de ensinar, a Didática, considerando-a importante. No entanto, aponta para a falta de um termo, igualmente importante, que signifique a arte de aprender. Considera que a didática oferece subsídios que podem tornar o professor mais hábil na arte de ensinar, porém não se encontra qualquer destaque sobre como compreender o próprio processo de aprendizagem.

Esta constatação demonstra que os estudos acerca do processo de ensino e aprendizagem são tendenciosos para o lado do professor. A preocupação com a arte de ensinar

revela que o professor é quem está no comando do processo educativo; desta forma, é ele quem precisa de habilidade para produzir um ensino eficaz. Ao aluno, é necessário somente obedecer ao que for estabelecido para o seu aprendizado (PAPERT, 1994).

Papert (1994, p.79) afirma que na escola, ensina-se às crianças mais sobre números e gramática do que sobre o pensar. Sua fala explicita a importância que tem o aprendizado de conteúdos em detrimento da reflexão. Como primeiro passo para corrigir esta deficiência, o autor sugere uma área de estudo dedicada à arte de aprender, definida por ele de matética.

Abordando a questão da resolução de problemas, o mesmo autor destaca que utilizar uma regra para resolver problemas não promove a aprendizagem, mas sim o pensar sobre o problema proposto. A arte de aprender seria, então, em vez de pensar em regras eficazes para aplicação imediata, procurar explicações de como trabalhar com essas regras, contribuindo, em longo prazo, para uma aprendizagem efetiva.

Nesta perspectiva, destacam-se temas mateticamente importantes que levam à aprendizagem (PAPERT, 1994):

• dar-se tempo - dar tempo a si mesmo para que possa refletir sobre a situação/problema para, em seguida, resolvê-la. No entanto, a escola transgride esse princípio matético quando retalha o tempo.

• falar - uma boa discussão promove a aprendizagem. A escola deve promover circunstâncias que favoreçam debates entre os alunos. Esta prática, freqüentemente, esbarra no medo que os alunos têm de falar sobre o que aprenderam e sobre suas dúvidas, acreditando que as demais pessoas já sistematizaram e organizaram o conteúdo. Isto ocorre, em especial, quando encontram-se na presença de pessoas que exercem poder sobre ela.

• conexões - construir uma forma individualizada de conexões, ou seja, associações significativas. A aprendizagem por intermédio da busca de conexões ocorre de forma simples, em um processo gradativo, diferente da memorização mecânica das características definidoras. As conexões acontecem em regiões de conhecimento fortemente evocativas para o sujeito.

Com base nesse aprendizado, Papert juntamente com outros pesquisadores do MIT, desenvolve uma linguagem de programação, chamada: LOGO.

> Minha meta tornou-se lutar para criar um ambiente no qual todas as crianças – seja qual for sua cultura, gênero ou personalidade – poderiam aprender Álgebra, Geometria, Ortografia e História de maneira mais semelhante à aprendizagem informal da criança pequena, pré-escolar, ou da criança excepcional, do que ao processo educacional seguido nas escolas (PAPERT, 1986:56)

O termo Logo foi escolhido como referência à sua significação grega: pensamento, raciocínio, discurso. E, ao contrário de muitas outras linguagens pensadas para a programação de computadores, Bossuet (1985, p. 43) coloca que Logo designa, ao mesmo tempo, uma teoria de aprendizagem, uma linguagem de comunicação e um conjunto de unidades materiais que permite demonstrar os processos mentais empregados por um indivíduo para resolver um problema, num contexto de ação sobre o mundo exterior.

O Logo nasceu do confronto entre o computador e a Ciência do Conhecimento. Papert, durante cinco anos afastou-se do Departamento de Matemática do MIT para estudar a psicologia da criança com Jean Piaget, em Genebra. Na sua volta, a sua equipe integrou-se com a equipe de Minsky e, juntos, elaboraram um projeto situado dentro das pesquisas em inteligência artificial e em ciências da educação baseadas na teoria de Jean Piaget sobre cognitivismo e sobre epistemologia genética. O resultado foi uma proposta de transformação na concepção do processo ensino-aprendizagem.

A respeito das relações entre a linguagem Logo e a aprendizagem, Papert afirma:

> No ambiente Logo, a criança, mesmo em idade pré-escolar, está no controle – a criança programa o computador. E, ao ensinar o computador a "pensar", a criança embarca em uma exploração sobre a maneira como ela própria pensa. O foco dos estudos de Piaget foi o "sujeito epistêmico", ou seja, o estudo dos processos de pensamento presentes no indivíduo desde a infância até a idade adulta. Pensar sobre modos de pensar faz a criança tornar-se um epistemólogo, uma experiência que poucos adultos tiveram (PAPERT, 1986:25).

E Papert complementa:

> Esta imagem poderosa da criança como epistemólogo veio à minha imaginação quando eu trabalhava com Piaget. Em 1964, depois de cinco anos no Centro de Epistemologia Genética de Jean Piaget, fiquei impressionado com sua maneira de ver as crianças como construtores ativos de suas próprias estruturas intelectuais (PAPERT, 1986:44).

A linguagem de programação LOGO propicia a representação e a construção de conhecimentos de quem manipula o computador e não apenas do especialista que elabora programas. Papert reconhece a linguagem LOGO como um ambiente no qual a tarefa do aprendiz não é aprender um conjunto de regras formais, mas desenvolver idéias para a solução de um problema. Na linguagem LOGO, a tarefa do professor não é fornecer respostas imediatas às questões apresentadas, mas desafiar e encorajar o aluno na busca de uma solução, pois [...] a melhor aprendizagem ocorre quando o aprendiz a assume. (PAPERT, 1988, p. 250).

Ao utilizar o LOGO, o computador é considerado uma ferramenta que propicia à criança as condições de entrar em contato com algumas das mais profundas idéias em ciências, matemática e criação de modelos. O erro é tratado como uma tentativa de acerto, ou seja, uma fase necessária à nova estruturação cognitiva, fortemente relacionada à teoria de equilibração de Piaget.

Nessa direção, Valente (1993; 1999), considerando a maneira como ocorre a construção do conhecimento pelo aluno em uma perspectiva construcionista, complementa os estudos estabelecendo o ciclo descrição-execução-reflexão-depuração-descrição.

O ambiente LOGO tradicional envolve uma tartaruga gráfica, um robô pronto para responder aos comandos do usuário. Uma vez que a linguagem é interpretada e interativa, o resultado é mostrado, imediatamente, após digitar-se o comando – incentivando o aprendizado. Nela, o usuário aprende com seus próprios erros. Se algo está errado em seu raciocínio, isso é claramente percebido e demonstrado na tela, num processo conhecido pelos jovens em jogos – feedback, fazendo que o aluno pense sobre o que poderia estar errado e tente, com base nos erros vistos, encontrar soluções corretas para os problemas. Como exemplo, os passos (instruções LOGO) para a tartaruga desenhar uma casa pode ser visto na Figura 1.

Figura 1- Exemplo em LOGO

Fonte 1: Site Informática em Educação

A metodologia de Papert supõe que a iniciação à linguagem de diálogo com as máquinas computadorizadas se dê através do lúdico. A linguagem viva, passo a passo, se estabelece e a criança aprende noções de forma, de velocidade, espaço, de procedimento, número, ângulo, variáveis, cálculo diferencial, limites, que se encontram no coração do sistema LOGO, e não aprende segundo moldes formais e teóricos, mas no sentido profundo e

utilitário. O diálogo que a criança estabelece com a tartaruga LOGO reproduz o próprio modo de cada um se relacionar com o material de suas experiências e reproduz seu próprio modo de pensar (Almeida, 2012).

O LOGO, desde a sua criação, por volta de 1967, até 1976, foi utilizada em computadores de médio e grande porte, fato que fez com que, até o surgimento dos microcomputadores, o uso do LOGO ficasse restrito às universidades e laboratórios de pesquisa. As crianças e os professores se deslocavam até esses centros para usarem o LOGO e nessas circunstâncias os resultados das experiências com o LOGO se mostraram interessantes e promissores (VALENTE e ALMEIDA, 1997).

Segundo Valente e Almeida (1997), o LOGO foi a única alternativa que surgiu com uma fundamentação teórica diferente no uso do computador na educação. Era possível de ser usado em diversos domínios do conhecimento e com muitos casos documentados que mostravam a sua eficácia como meio para a construção do conhecimento.

A comunidade pedagógica só passou a incorporar as idéias de Papert a partir de 1980, quando ele lançou o livro Mindstorms: Children, Computers and Powerful Ideas. Esse livro mostrava caminhos para a utilização das máquinas no ensino.

Com a disseminação dos microcomputadores, o LOGO passou a ser adotado e usado em muitas escolas. No período de 1983 até 1987 aconteceu uma verdadeira explosão no número de experiências, na produção de material de apoio, livros, publicações e conferências sobre o uso do LOGO surgindo uma grande variedade de versões tais como:

- **Superlogo** - Aqui no Brasil, em 1982, A Unicamp, através do Núcleo de Informática Aplicada à Educação-NIED, traduziu o LOGO pela primeira vez para o português, e lançou o SuperLogo para ambiente *Windows*. Foi uma iniciativa bastante arrojada e exitosa para os padrões da época tendo sido bastante utilizada em universidades e escolas. Esse projeto ajudou muito a semear a cultura do LOGO no Brasil e influenciou outras versões para o português. Sua última versão a 3.0 foi em março de 2004. Não houve mais evolução. (COMMONS, 2009).
- **KTurtle** - desenvolvida em Java e que foi incorporada ao pacote de programas do Linux Educacional 4.0 (Programa do Governo Federal) atualmente na versão 5.0 que foi desenvolvida pelo Centro de Computação Científica e Software Livre (C3SL) da Universidade Federal do Paraná-UFPR e que é distribuído para todas as escolas públicas do Brasil que possuem laboratórios de informática. (COMMONS, 2009).
- **xLogo** - escrita em Java sob licença GPL (software livre), bastante versátil, roda em 8 idiomas: alemão, árabe, francês, inglês, espanhol, português, galês e esperanto. Em

novembro de 2006 lançou sua primeira versão para comandar uma interface externa para robótica. Atualmente incorpora várias primitivas de comunicação em rede que além de ser utilizadas para experimentação em robótica, permitem tarefas grupais em rede, chat, comandar a *tartaruga* em um PC a partir de outro PC, etc. (COMMONS, 2009).

- **Elica** – Versão excelente para construção de objetos 3D, infelizmente, como foi originalmente desenvolvida pra Windows XP, seu projeto foi descontinuado em 2013. Mas, ainda existe uma grande biblioteca de projetos que podem ser baixados e utilizados. (COMMONS, 2009).
- **FMSLogo** – Oferece recursos que versões comerciais não tem como. Exemplo: pode usar qualquer recurso de som e vídeo disponível no computador (cd, DVD, wav, mp3, etc). Uma limitação dessa versão é não possuir plugin que permita publicar projetos em páginas Html (web). Pode ser baixado livremente. Padrão Windows, no Linux é necessário instalar o wine (interpretador Linux), mas torna a aplicação lenta. (COMMONS, 2009).

1.5 Mitchel Resnick e a Aprendizagem Criativa

Michel Resnick foi graduado em Haverford High School na Pensilvânia, bacharel em física na Universidade de Princeton em 1978, PhD em ciência da computação pelo MIT em1988. Trabalhou por cinco anos como jornalista de ciência para a revista *Business Week*, pesquisando amplamente sobre o uso de computadores na educação. Resnick foi premiado com um National Science Foundation Young Investigator Award, em 1993. Ele é co-editor da obra "Construcionismo na Prática: a concepção, Pensamento e Aprendizagem em um Mundo Digital" de 1996, e co-autor de "Aventuras em Modelagem: Explorando sistemas complexos e dinâmicos com StarLogo" em 2001. Dr. Resnick foi listado como uma das 100 pessoas mais criativas nos negócios em 2011 pela Fast Company .

Pesquisador do MIT e seguidor da teoria do construcionismo de Papert, Mitchel Resnick busca aliar criatividade e aprendizagem, partindo do princípio que, na educação infantil, as crianças constroem brinquedos, instrumentos criativos que lhes possibilitam pensar, testar e aprender. Também, no ensino fundamental e médio, o caminho do aprender não deveria ser diferente. Os estudantes não são ouvintes passivos, mas permanecem inativo sem sala de aula, quando o professor os sobrecarrega de dados e teorias.

Em suas pesquisas sobre aprendizagem, Resnick cita a lógica do "jardim da infância para toda a vida". As ciências da computação devem desenvolver materiais para que as crianças possam crescer, construir soluções criativas que envolvam as várias áreas do

conhecimento. Brinquedos vão adquirindo formas mais complexas, como pequenos robôs e engenhocas a serem montados e programados pelas crianças. Nesse processo, o computador torna-se um grande aliado, uma nova ferramenta para o aprendizado criativo. (RESNICK,2006)

O propósito colocado pelo pesquisador é aprender por toda a vida, pesquisando, movimentando-se, deixando de lado as atividades estanques. Assim, criam-se atividades que se integram às diversas áreas do currículo, para se chegar a uma maior compreensão do assunto em questão. As crianças vão evoluindo no processo e descobrem soluções para problemas do dia a dia.

A tecnologia tem de ser dada às crianças e aos jovens de modo que lhes faça sentido, e o mais importante é que eles podem mudar os sistemas que criam, devendo fazer isso de acordo com seus próprios interesses e necessidades e usar as novas tecnologias na busca de soluções de seus problemas. O aprendizado, por meio de projetos e experiências, deve ultrapassar o sentido de sociedade de informação, evoluindo para o conceito de sociedade do conhecimento. (RESNICK, 2006)

No momento, busca-se evoluir para "sociedade criativa", sendo preciso, para tanto, saber dar uso à informação. "As pessoas precisam continuar aprendendo a vida toda e dando soluções criativas para seus problemas e necessidades", conclui o pesquisador. (2006)

Difundir novas idéias, fazer as escolas conhecê-las é importante para que possam vingar. O trabalho de Mitchel Resnick fala, justamente, na dificuldade em atingi-las, prevendo que as mudanças serão lentas, mas que devem ocorrer com maior facilidade ao longo das próximas gerações. "As crianças de hoje é que estarão melhor [sic] preparadas para as mudanças sistêmicas". (RESNICK, 2006)

Vivemos o momento da inclusão digital maciça, de onde saem, entre outros, estudos da aplicação de novas tecnologias na educação. Por meio delas, pode-se melhorar e ampliar o aprendizado, usando-se músicas, esportes, laboratórios de ciências, ou, mesmo, observando-se a natureza.

É preciso ajudar os alunos a usarem as tecnologias de forma inovadora e produtiva, promover experiências criativas, abrindo portas para essas crianças às novas e infinitas possibilidades de aprender. Com isso, é possível operar a busca pelo desenvolvimento de um pensamento criativo sistêmico e intencional.

1.6 O que é Tinkering?

Quando Resnick propôs o termo Tinkering, seu objetivo era uma abordagem caracterizada por um estilo divertido, interativo, experimental, onde os criadores estão sempre redefinindo os seus objetivos, explorando novos caminhos e imaginando novas possibilidades (RESNICK, 2013).

O processo do Tinkering, que pode ser traduzido como exploração despreocupada, vai de encontro com o ideal de planejamento prévio. Sua implicação é em uma produção mais orgânica e pessoal. Planejar parece mais organizado e eficiente, porém limita o processo criativo não dando espaço para a intervenção contínua da criatividade e da inventividade. Planejadores pesquisam uma situação, identificam um problema e as necessidades, desenvolvem um plano transparente e depois o executam. Já o processo de explorar despreocupadamente é desorganizado, mas envolto em experimentação buscando novas alternativas. Enquanto planejadores trabalham por regras formais, exploradores despreocupados tendem a reagir a detalhes específicos de uma situação em particular (RESNICK, 2013). O ato de explorar despreocupadamente não é um conceito novo. Desde os tempos remotos em que os primeiros humanos começaram fazendo e usando ferramentas, o ato de explorar sem exigências está presente. Mas trabalhar assim é mais importante hoje do que o foi em todos os tempos. Com o atual dinamismo profissional e as mudanças do mercado atual, apenas um profissional qualificado pode reagir rapidamente a mudanças, por vezes imprevistas. A característica necessária ao profissional é desenvoltura, criatividade e adaptação ao cenário moderno. (RESNICK, 2013).

Uma característica dos exploradores despreocupados é a de que eles trabalham da base para o topo, ou seja, começam “brincando com os materiais” sem uma preocupação extenuante até conseguirem estabelecer uma relação de interesse e objetivo em função de uma montagem e vão subindo e especificando, detalhando sua obra, incrementando a qualidade do trabalho.

Utilizamos o método da exploração despreocupada em nossa proposta de sala de aula, onde o aluno precisa descobrir, com o professor na função de mediador, a melhor condição para a construção de sua obra, implementando sua alma no processo de criação. É função do professor sugerir temas e não desafios para os alunos criarem suas histórias. Esses temas precisam ser o mais amplo possíveis para que os alunos tenham espaço para criarem as idéias. Inicia-se com o ato de imaginar do aluno. O que se quer construir? Que tipo de animação? jogos ou apresentação se tem interesse de empregar sua energia? Chama-se esse passo de Desenvolvimento de Projetos Passionais. Resnick menciona esse processo dentro de sua

espiral de conhecimento. O modelo em espiral proposto por Resnick (Media Lab - MIT) incorpora o processo sequencial de criação e abordagem metodológica para o estudante. Tal processo inicia-se pelo ponto Imaginar, seguido por Criar, Divertir-se, Compartilhar e Refletir, para depois reiniciar o processo de Imaginar, novamente. Imaginar

O aluno inicia sua jornada aprendendo com o professor os passos básicos do programa e alguns exemplos de jogos/ apresentações criadas anteriormente. A partir daí, o aluno deve imaginar o seu próprio projeto, estabelecendo uma conexão com o programa e tentando dar vida a sua obra.

1.6.1 Crie

O aluno passa a utilizar os recursos do programa, de forma intuitiva e com o auxílio de ferramentas pedagógicas digitais como apostilas e vídeoaulas, para entender os comandos básicos e suas funcionalidades. O professor interfere apenas quando é solicitado e apenas oferecendo mais perguntas, para que o aluno possa chegar a uma conclusão por conta própria. Não importa se o tempo gasto pelo aluno utilize toda a aula, o importante é que ele entenda o processo para alcançar o seu objetivo. Com o tempo o aluno apropria-se das ferramentas e entende como o programa funciona. Neste momento a função do professor é de sugerir algumas ideias para que o aluno possa ir além do que ele já conseguiu, ampliando seu programa, ou dando outras funções ao projeto. O importante é que o aluno tenha diversas oportunidades de criação e várias formas de realização da idéia.

1.6.2 Divirta-se/corrigir

Divertir-se é uma das características mais interessantes para a educação, porque traz um sentimento de felicidade e de não obrigatoriedade ao processo de aprendizado. Divertir-se deve ser o que move o aluno a aprender.

O aluno quando desenvolve o projeto, traz em si o processo de realizar algo que tenha imaginado. Em sua imaginação o jogo possui uma forma, mas ao terminar o projeto, o produto final nem sempre atende às expectativas dos alunos. O importante é que o projeto seja divertido e que o resultado final também. Para isso, os alunos devem ter sempre uma visão aberta sobre o processo de falha e de novas tentativas. Corrigindo os erros e ampliando a qualidade do produto, tanto visual como a jogabilidade.

1.6.3 Compartilhe

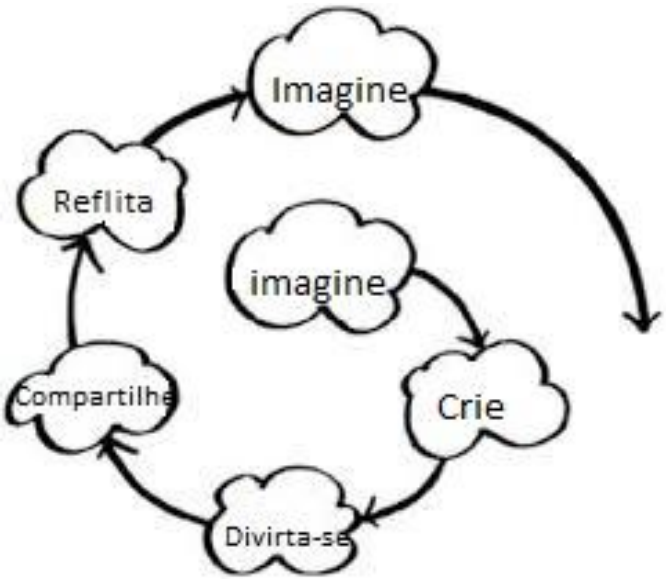
Levando-se em consideração as diferenças culturais, o Compartilhar impacta mais do que o restante do processo e possui uma conexão intrínseca com o estado: motivação. O aluno compartilha para ver sua criação aceita ou rejeitada pela comunidade, seja ela de colegas de

sala, de alunos da instituição, dos parentes, dos amigos ou da população em geral. Esse compartilhamento tem um olhar micro e macro. No Micro, compartilha-se inicialmente entre os alunos da sala de aula, outros professores e diretores. No Macro, postam-se os trabalhos no site do *Scratch*, onde existe uma comunidade imensa de participantes e se observa os programas serem "remixados" por outros usuários e os comentários a respeito do trabalho aparecerem. Em um escopo ainda maior, apresentam-se os trabalhos em feiras de ciência, onde alunos de outras escolas podem jogar os programas criados dos alunos.

1.6.4 Reflita

No estágio do refletir cabe ao aluno, novamente com a mediação do professor, pensar se o resultado final atingiu o objetivo e como poderiam ser modificados para um melhor aproveitamento. Neste estado o aluno consegue entender como o programa funcionou e se a audiência aprovou ou não, demonstrando dificuldade excessiva ou facilidade extrema.

Figura 2 - Espiral do pensamento criativo (Resnick)



Fonte 2: eduteka[2]

1.7 O que é Fluência Digital?

Para Mitchel Resnick, considerando-se a analogia com a aprendizagem de uma língua, se alguém aprendesse algumas frases que lhe permitissem ler o menu de um restaurante ou perguntar sobre a direção para determinado destino, isso não significa que a pessoa tenha

[2] Disponível em: http://www.eduteka.org/modulos.php?catx=9&idSubX=277&ida=914&art=1 Acesso em abril/2013

fluência naquela língua. Esse tipo de conhecimento, proveniente de livros de frases prontas, é o equivalente ao que a maioria das pessoas sabe sobre computadores hoje. Esse conhecimento é útil? Sim, mas não é fluência.

Para ser fluente em uma língua, você precisa saber articular uma ideia complexa ou contar uma história, em outras palavras, você precisa saber "fazer coisas" com essa língua. Fazendo a analogia, ser digitalmente fluente envolve não apenas saber como usar ferramentas tecnológicas, mas também saber como construir coisas significativas com essas ferramentas. (PAPERT e RESNICK 1995).

A fluência em uma língua não tem apenas valor nas tarefas do dia a dia, mas, exerce também um efeito catalisador na aprendizagem. A partir do momento que a pessoa aprender a ler e escrever, um mundo de outros saberes surge. O mesmo acontece com a fluência digital. Ser fluente digitalmente no novo milênio é pré-requisito não só para obtenção de um emprego, mas ter uma participação mais significativa na sociedade, e aprendizagem ao longo da vida. A cada dia, com o declínio dos custos computacionais, o acesso às novas tecnologias digitais é cada vez mais crescente. No entanto, nem todas as pessoas saberão utilizá-las fluentemente.

1.8 Computer Clubhouse Network

Mitchel Resnick e Natalie Rusk (MIT Media Lab); Yasmin Kafai (UCLA- Los Angeles University of California); Stina Cooke (Museu de ciências de Boston); fundam em 1993 na cidade de Boston o Compute Clubhouse Network, um ambiente de programação em rede e estudos pós-escola principalmente em centros comunitários mais carentes com o objetivo de proporcionar às pessoas mais jovens (entre 10-18 anos), não só o acesso, mas a oportunidade de se tornar fluente digitalmente. O projeto se expandiu rapidamente para várias localidades em vários países e recebeu o prêmio Peter Drucker (1909-2005, pai da administração moderna) de prestígio para inovação sem fins lucrativos. Em 2000, a Intel incorpora-se ao projeto investindo mais de U$ 50 milhões e atualmente possui uma rede de 100 clubhouses em 20 países: Argentina, Austrália, Brasil, Colômbia, Costa Rica, Dinamarca, Hungria, Índia, Irlanda, Israel, Jordânia , México, Nova Zelândia, Palestina, Panamá, Filipinas, Rússia, Taiwan, África do Sul, e os Estados Unidos.

Computer Clubhouses são muito diferentes da maioria dos centros e comunidades, que normalmente caem em uma das duas categorias. Alguns centros tecnológicos meramente

fornecem acesso. As pessoas podem fazer o que quiserem: jogar, navegar na Web, usar salas de bate-papo online. Outros centros de ensino oferecem cursos estruturados, habilidades básicas de computador (como digitação) e aplicativos básicos(como processamento de texto e planilhas). *O Computer Clubhouse* oferece uma terceira via, com objetivos diferentes e uma abordagem diferente. A finalidade não é simplesmente ensinar habilidades básicas, mas ajudar aos jovens a aprender a expressar-se e ganhar confiança em si mesmos como aprendizes. Se eles estão interessados em jogos de vídeo, eles não vêm para o *Clubhouse* para jogar; eles vêm para criar seus próprios jogos. Eles não vão baixar vídeos a partir da *Web*, mas criar os seus próprios vídeos. No processo de juventude, aprender a heurística de ser um bom *designer*: como conceituar um projeto, como fazer uso dos materiais disponíveis, como persistir e encontrar alternativas quando as coisas dão errado, como colaborar com os outros, e como ver um projeto através do olhos dos outros. Em suma, eles aprendem a gerenciar um complexo projeto do início ao fim. A abordagem *Computer Clubhouse* estabelece um equilíbrio entre estrutura e liberdade no processo de aprendizagem. Os jovens aprendem a trabalhar em projetos com base em seus próprios interesses., eles recebem uma grande apoio de outros membros do *Clubhouse* (por exemplo, os funcionários, professores voluntários, e *Clubhouses*; eles fornecem aos jovens um sentido do possível, e múltiplos pontos de entrada através dos quais eles podem começar. O objetivo é proporcionar liberdade suficiente para permitir que os jovens não só sigam suas fantasias, mas também recebam apoio suficiente para ajudá-los a transformar essas fantasias em realidade.

1.9 O Pensamento Computacional

O pensamento computacional pode ser definido como o pensamento analítico que compartilha com o pensamento da matemática, engenharia e ciência com o objetivo de aprimorar a busca por soluções de problemas. É uma forma de pensar que utiliza conceitos e metodologias da computação para resolver questões em um amplo espectro de assuntos oferecendo um conjunto de habilidades importantes para qualquer das ciências modernas. Habilidades como abstração, modularização e decomposição presentes no pensamento computacional podem ser aplicadas na resolução de uma grande gama de problemas do dia a dia das pessoas, tanto em aspectos cotidianos como profissionais. Uma quantidade cada vez maior de pesquisadores considera que, além da competência para leitura, escrita e aritmética, deve-se adicionar o pensamento computacional a capacidade analítica que a escola deve formar em cada criança.

Neste contexto, este trabalho irá apresentar algumas habilidades do pensamento computacional tais como: sequências, iterações, repetições, condicionais, operações aritméticas e randômicas, paralelismo, tratamento de eventos, recursividade e variáveis; através de exemplos práticos com o Scratch.

2 PERCURSO METODOLÓGICO DO ESTUDO

A metodologia desenvolvida para elaboração deste estudo baseou-se em pesquisas bibliográficas e através da internet em: artigos; experiências exitosas realizadas no Brasil com o *software*; análises de projetos desenvolvidos e a proposta educacional dos próprios desenvolvedores do *SCRATCH* através da plataforma: http://scratch.mti.edu e o desenvolvimento de algumas aplicações práticas com a ferramenta *SCRATCH* com o objetivo de demonstrar a sua facilidade de uso e como exemplos que podem ser aplicados em sala de aula dando a pesquisa um caráter qualitativo, quantitativo e experimental.

Inicialmente investigou-se a plataforma de desenvolvimento do *SCRATCH* na qual se verificou uma comunidade mundial de desenvolvimento criativa com mais de 17 milhões de projetos armazenados (utiliza-se da computação em nuvem), do mais simples ao mais sofisticado, elaborados por pessoas de todas as idades em todos os níveis de escolaridade. Nessa plataforma têm orientações, estatísticas, fóruns de discussões para diversos perfis como: pais, educadores e desenvolvedores.

Como fundamentação prática, foi realizada uma leitura exaustiva do livro: Aprenda a Programar com o Scratch (MAJED, 2014) que serviu de fonte base para explorar as possibilidades computacionais do *software* refazendo algumas aplicações elaboradas e outras propostas constantes no livro.

Por trata-se de um *software* no qual todo o ambiente de programação compõe-se de componentes visuais, foram elaborados diversos *screenshots* (capturas de tela) da própria ferramenta para dar uma visão genérica das possibilidades de desenvolvimento de um projeto não sendo, portanto, um manual de referência.

Por fim, para demonstrar a facilidade de uso do *software* e justificar os objetivos do estudo, foram desenvolvidos alguns projetos práticos que demonstram a versatilidade e o caráter educacional da ferramenta que pode ser aplicado em qualquer área de estudo tendo como premissa fundamental o desenvolvimento do raciocínio lógico-matemático e computacional do educando e não sua expertise no *software*. Para tanto, apresentou-se algumas aplicações que podem ser elaboradas por um professor de matemática ou geografia, apenas como exemplo; e ao final um projeto de um jogo constante na plataforma do *SCRTACH* para demonstrar uma das principais características do *software* que é o compartilhamento de forma colaborativa, ou seja, a partir de ideias de outras pessoas pode-se reconstruir, e ou, elaborar novas.

3 O SCRATCH

O *Scratch* é uma linguagem gráfica de programação desenvolvida pelo *Lifelong Kindergarten* Group no MIT *Media Lab* coordenado por Mitchel Resnick em colaboração com o grupo de Alan Kay da UCLA (Resnick, 2003). O projeto foi iniciado em 2003 com o apoio financeiro da *National Science Foundation, Microsoft, Intel Foundation, MacArthur Foundation, LEGO Foundation, Code-to-Learn Foundation, Google, Dell, Fastly, Inversoft* e MIT *Media Lab research*. Sua primeira versão foi disponibilizada em maio de 2007. O software foi idealizado para atender jovens na faixa de 8 aos 16 anos sem nenhum conhecimento prévio de programação, no entanto, ele é utilizado por pessoas de todas as idades e em todas as faixas de conhecimento. A ferramenta utiliza-se de mídias bastante diversificadas como som e imagens permitindo criar aplicações (programas) interativas como: histórias, artes, jogos e animações. Sua interface bastante lúdica se assemelha as peças de um brinquedo *LEGO* no qual os usuários vão juntando blocos de instruções de forma lógica como se estivesse montando um quebra-cabeça. Tal facilidade decorre do fato de que as peças (instruções) já estão previamente programadas e se encaixam sintaticamente corretas. Para MARQUES (2008), diferentes tipos de dados possuem diferentes formas, eliminando a possibilidade de combinações erradas. A programação, ou a ordenação dos blocos gráficos logicamente inseridos pode ser alterada, ou testada a execução, mesmo com o programa em andamento, recurso que facilita a experimentação de novas ideias ou a depuração do programa.

O termo *Scratch* provém da técnica de *scratching* utilizada pelos *Disc-Jockey* (DJ) do *Hip-Hop* na qual giram discos de vinil com as suas mãos para frente e para trás de modo a fazer misturas musicais de forma criativa e inesperada. Processo análogo que se faz com o *Scratch* ao se mesclar mídias (gráficos, fotos, sons, músicas) de forma criativa.

Uma das características principais da plataforma é o compartilhamento de projetos e sua reutilização em novas ideias ou aprimoramentos (*remixagem*). O *Scratch* está presente em mais de 150 países e já foi traduzido para mais de 40 idiomas. Sua biblioteca atual possui mais de 11 milhões de projetos postados. Seu slogan é: “imagine, programe e compartilhe” dentro do processo espiral intitulado por Resnick. Nesse contexto, o momento reservado à programação tem como fundamento o ciclo proposto por Valente (1997) descrição-execução-reflexão-depuração. E, ainda mais, a possibilidade da partilha de saberes como um novo agente de aprendizagem eleva esse pensamento para além do ciclo. Segundo Resnick:

> [...] as pessoas imaginam o que elas querem fazer, criar um projeto com base em suas idéias, brincar com as suas criações, partilhar as suas idéias e criações com outros, e refletir sobre as suas experiências, tudo de que os leva a imaginar novas idéias e novos projetos. Como os alunos envolvem nesse processo, mais e mais, aprendem a desenvolver as suas próprias idéias, julgá-las, testar os limites, experimentar outras alternativas, obter construções dos outros, e gerar novas idéias com base em suas experiências (RESNICK, 2007b, P.18).

3.1 Analisando as possibilidades do SCRATCH

3.1.1 Histórico

A partir de observações em centros comunitários como os *Computer Clubhouse* (Resnick, 2002), os pesquisadores do MIT constataram que os jovens se interessavam por vídeo games, vídeo clipes, animações de personagens e criação de arte interativa. Com isto em mente eles começaram a desenvolver um ambiente de programação multimídia que possibilitasse aos jovens criar suas próprias interfaces e aplicações tornando-os consumidores críticos e produtores, ou seja, fluentes digitais. Desenvolver uma ferramenta que atenda tais pré-requisitos e que ao mesmo tempo seja simples o seu entendimento não é uma tarefa fácil. No entanto, tendo por base principalmente o LOGO e outras linguagens de programação tais como: *StarLogo, Smalltalk, Hypercard, Squeak Etoys*; surge então, o *Scratch*.

> Ao criarem seus projetos em *Scratch*, os jovens aprendem muitas habilidades do século 21 que serão críticas para um futuro de sucesso: pensar criativamente, comunicar-se claramente, analisar sistematicamente, usar tecnologias fluentemente, colaborar efetivamente, projetar iterativamente e aprender continuamente. (Resnick, 2007).

3.1.2 O ambiente de programação do Scratch.

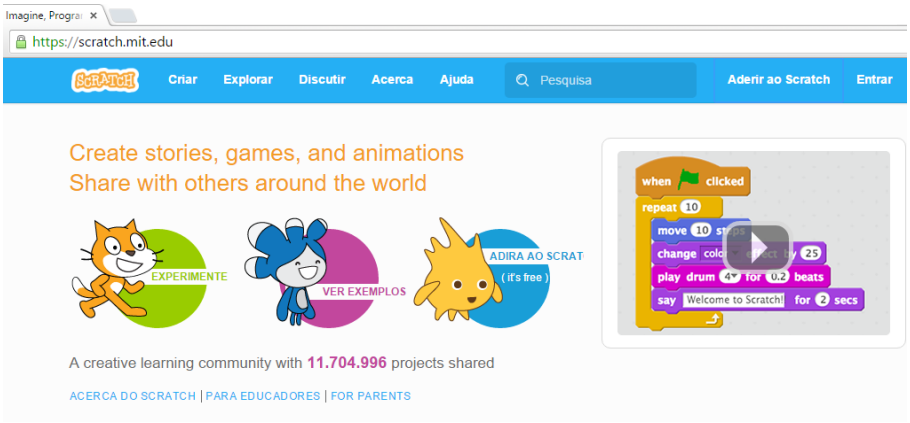
Neste tópico, será apresentada uma visão geral do *Scratch* e algumas funcionalidades.

Um programa de computador nada mais é do que um conjunto de instruções que dizem ao computador o que ele deve fazer. Essas instruções (comandos) são escritas por meio de uma linguagem de programação, geralmente baseadas em texto. Aprender essas linguagens e entender suas regras de sintaxe, conceitos computacionais, é uma tarefa tediosa, principalmente para iniciantes.

Como dito anteriormente, no *Scratch*, as instruções já estão previamente codificadas em forma de blocos gráficos bastando encaixá-los adequadamente, ou seja, dentro de uma lógica para que os resultados pretendidos sejam apresentados.

Ao acessar a plataforma do *Scratch*[3], o usuário já poderá ver tutoriais, exemplos ou desenvolver seus próprios projetos *online* clicando na imagem do gato (logomarca do *Scratch*) que tem a legenda: EXPERIMENTE. Mas, para fazer *upload*, *download e compartilhamento*, deve-se criar um perfil bastando para tanto clicar na opção: "Aderir ao *Scratch*", como mostra a Figura 3. Logo após, será apresentada uma tela de cadastro, Figura 4, e seguir os passos nos quais serão solicitados: nome de usuário, senha, e-mail de contato, mês/ano nascimento, sexo, país. Estas informações são sigilosas e muito importantes para as estatísticas e funcionalidades colaborativas dos usuários *Scratch.* Por exemplo, o usuário será capaz de saber através de uma árvore todas *remixagens* dos projetos e de que país.

Figura 3- Plataforma do SCRATCH



Fonte 3: Screenshot da homepage do Scratch (pelo autor)

[3] Disponível em: https://scratch.mit.edu/. Acesso em 25/05/2012

Figura 4 - Tela de cadastro do SCRATCH

Fonte 4: Screenshot do SCRTACH (pelo autor)

No rodapé da *homepage* da plataforma encontram-se cinco seções: **Sobre**, **Community**, **Suporte**, **Termos Legais** e **Família Scratch**. Nessas seções estão disponíveis todas as informações para um uso adequado da ferramenta tais como: orientações aos pais ou educadores; fóruns de discussões, documentação, estatísticas de projetos e usuários; créditos. O *Scratch* também disponibiliza um editor *offline* (versão atual – 441.1) para ambientes: *Windows*, *Mac* ou *Linux*; na seção **suporte**, opção: Editor *Offline*, conforme Figura 5.

Figura 5 - Seções do SCRATCH

Fonte 5: Screenshot da plataforma SCRATCH (pelo autor)

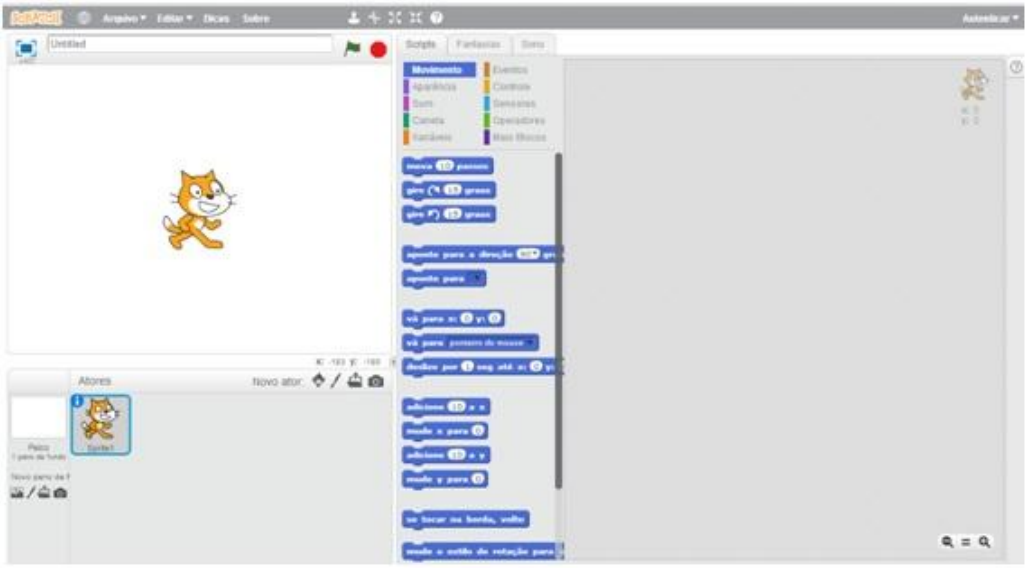
Como dito anteriormente, ao clicar na opção: EXPERIMENTE, será apresentado ao usuário o ambiente de desenvolvimento: a *interface* do Editor *Scratch*, conforme a Figura 6.

. Os painéis principais do *Scratch* são: **O palco** (na parte superior à esquerda); A lista de ***Sprites*** (na parte inferior à esquerda); **Os blocos** ao centro e a **Área de *Scripts*** (à direita).

Um ***Sprite*** (ator) é um objeto qualquer (um carro, uma casa, um animal, etc) que é representado por uma figura 2D. O *Scratch* aceita os formatos de imagens mais usuais (bmp, gif, tif, jpeg).

Scripts (roteiros) são os blocos de instruções que associados ao(s) ***Sprite***(s) construirão o projeto.

Figura 6 - Interface do Editor do SCRATCH



Fonte 6: Screenshot da interface do SCRATCH (pelo autor)

3.1.2.1 *O Palco*

É o local em que os *Sprites* se movem e interagem, ou seja, é área de apresentação dos resultados do projeto. Ele tem 480 passos de largura e 360 passos de altura como um plano de coordenadas (x,y), o centro do palco possui coordenadas x igual a zero e y igual a 0. Essas posições são facilmente encontradas observando-se os números no canto inferior direito do palco ao mover-se com o mouse.

Logo acima do Palco encontra-se o ícone para mostrar o *Scratch* em forma de tela de Apresentação (esse ícone também mostra a versão do *Scratch*). Em seguida têm-se uma caixa de edição para nomear o projeto e os ícones de **bandeira verde** (inicia o projeto) e botão **vermelho** para finalizar um projeto em execução como mostrados na Figura 7.

Figura 7 - Ícones de apresentação, caixa de edição (nome do projeto), exceção e parada

Fonte 7: Screenshot do Scretch (pelo autor)

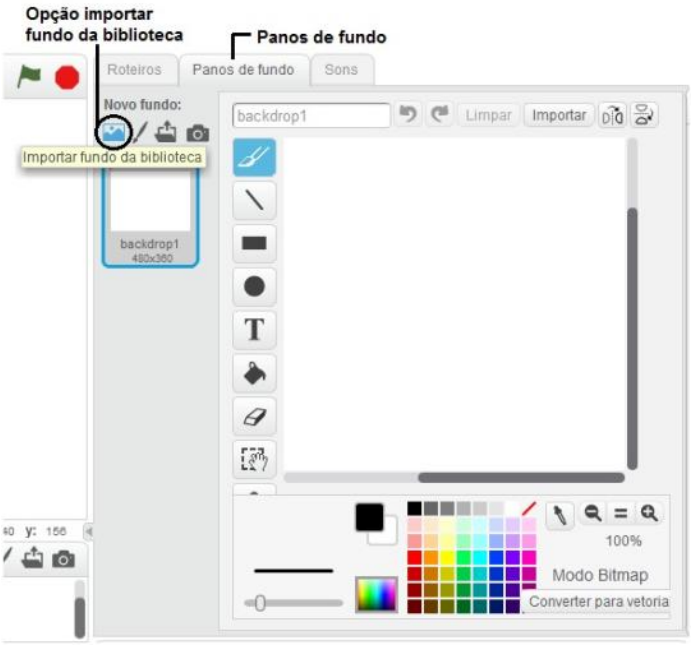
Logo abaixo do palco (na parte inferior esquerda) têm-se uma miniatura do palco e as opções de se modificar a apresentação do palco (cenários). Nesse local pode-se adicionar qualquer imagem, som, desenhar. Posicionando-se o mouse sobre os ícones das ferramentas de palco têm-se uma pequena ajuda das funcionalidades conforme Figura 8.

As abas: Roteiros (*Scripts)*, pano de fundo e Som; são intuitivas conforme Figura 9. O *Scratch* já traz uma biblioteca de sons e cenários que associados aos *scripts* tornam as apresentações mais elaboradas e dinâmicas conforme Figura 10.

Figura 8 - Palco do SCRATCH

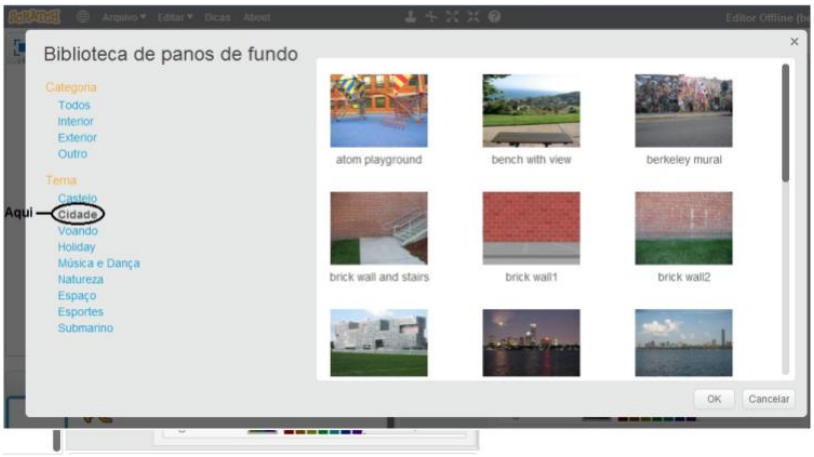
Fonte 8: Screenshot do Scratch (pelo autor)

Figura 9 - Abas do palco SCRATCH



Fonte 9: Screenshot do Scratch (pelo autor)

Figura 10 - Bibliotecas de palco SCRATCH



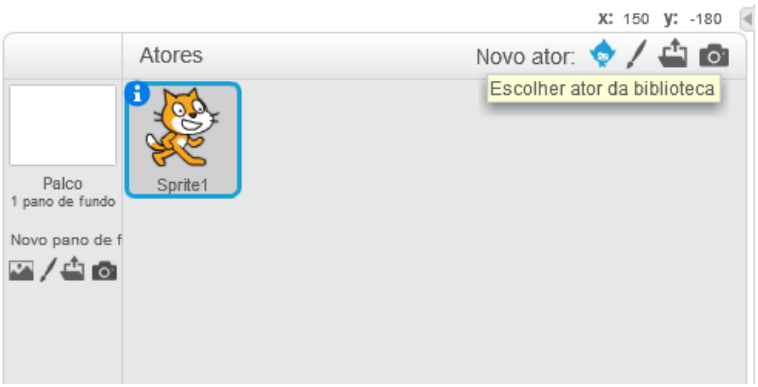
Fonte 10: Screenshot do Scratch (pelo autor)

3.1.2.2 *A lista de Sprites*

A lista de *Sprites* exibe os nomes e as miniaturas de todos os *sprites* do seu projeto. Novos projetos começam com o palco em branco e um único *sprite* representado por um gato (Logomarca do *Scratch*) na posição (0,0) do palco. Os botões (ícones) acima da lista de *sprites*, permitem adicionar novos *sprites* ao projeto a partir de uma das opções: biblioteca de *sprites* do *Scratch*, do *Paint Editor* incluído (similar ao *Paint* do *Windows*) para desenhar

fantasias, de uma câmera conectada ao computador ou arquivos de imagens do próprio computador conforme Figura 11.

Figura 11 - Lista de Sprites



Fonte 11: Screenshot do Scratch (pelo autor)

Cada *Sprite* do projeto possui seus próprios *scripts*, fantasias e sons. Para verificar os seus pertences basta selecioná-lo (contorno com uma borda azul na miniatura) ou dá um duplo clique no próprio *sprite* do palco. Pode-se também duplicar, apagar, salvar (juntamente com os *scripts*) ou esconder o *sprite* clicando-se com o botão direito sobre o *sprite* conforme mostra a Figura 12.

Figura 12 - Menu suspenso do sprite

Fonte 12: Screenshot do Scratch (pelo autor)

Para visualizar a área de informações sobre o *sprite*, clique no pequeno ícone "i" no canto superior esquerdo da miniatura de um *sprite*. Essa área mostra o nome do *sprite*, sua

posição atual (x,y) e a direção, seu estilo de rotação e o estado quanto à visibilidade, além de mostrar se ele pode ser arrastado em modo de Apresentação conforme Figura 13.

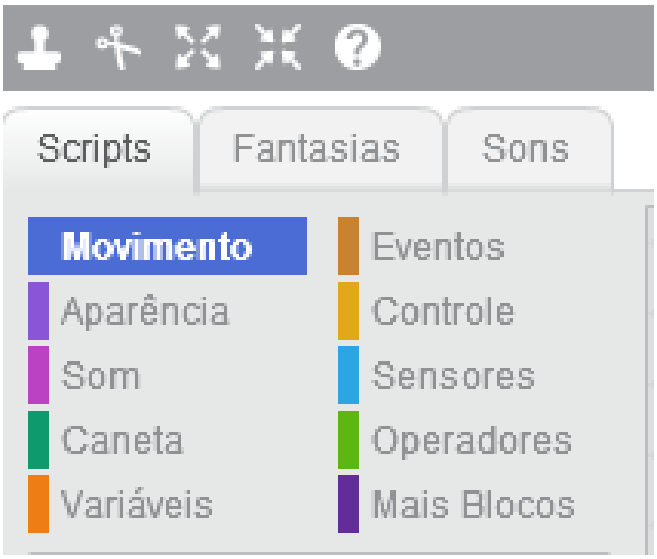
Figura 13 - Informações do sprite

Fonte 13: Screenshot do Scratch (pelo autor)

3.1.2.3 *Aba de blocos do Scratch*

Os blocos no *Scratch* estão divididos em dez categorias (paletas): Movimento, Aparência, Som, Caneta, Variáveis, Eventos, Controle, Sensores, Operadores e Mais Blocos (a partir da versão 2.0) conforme mostra a Figura 14. Os blocos são diferenciados por uma cor para ajudar na identificação daqueles que estão relacionados. Alguns blocos exigem uma ou mais entradas (também chamadas de argumentos) que dizem ao bloco o que ele deve fazer. Os blocos além de ter uma descrição bastante intuitiva, possuem também uma tela de ajuda, basta selecionar o ícone de interrogação na barra de ferramentas e clicar no bloco que se deseja obter ajuda. Ainda com relação a barra de ferramentas, pode-se duplicar, apagar, aumentar ou diminuir o *sprite* clicando no ícone apropriado.

Figura 14 - Blocos de Categorias do Scratch



Fonte 14: Screenshot do Scratch (pelo autor)

3.1.2.4 *A Área de Scripts*

Para fazer com que um *sprite* faça algo, é necessário programá-lo arrastando blocos para a área de scripts, unindo-os. Ao arrastar um bloco para esta área, será destacado um contorno em branco indicando em que local esse bloco pode ser solto para formar uma conexão válida com outro bloco conforme mostra a Figura 15. Os blocos do *Scratch* somente se encaixam de determinadas maneiras, eliminando os erros de digitação que tendem a ocorrer quando as pessoas usam linguagens de programação baseadas em texto.

Figura 15 - Área de scripts (roteiros)

Fonte 15: Screenshot do Scratch (pelo autor)

Não é preciso completar os *scripts* para executá-los, o que significa que podem ser testados à medida que os criar. Clicar em qualquer ponto de um *script*, esteja ele totalmente ou parcialmente criado, faz o script todo ser executado de cima pra baixo. Pode-se facilmente desmontar uma pilha de blocos e testar cada uma das partes individualmente. Aliás, essa é

uma estratégia muito utilizada para entender *scripts* longos ou criar um projeto por partes. Para mover uma pilha de blocos, basta clicar na parte superior da pilha e arrastá-la ou em qualquer parte da pilha quando se deseja desmembrar. Pode-se também arrastar ou copiar uma pilha de blocos de um *sprite* para outro. Basta arrastar a pilha da Área de *Scripts* do sprite de origem para a miniatura do *sprite* de destino na lista de *Sprites*. Quando um *script* está em execução uma área amarela aparece em torno do *script*.

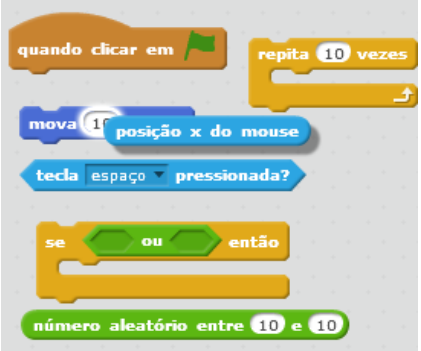
O *Scratch* possui quatro tipos de blocos: blocos de comando, blocos de controle, blocos de *trigger* (gatilhos) e blocos de função.

Os blocos de comandos e os blocos de controle (também chamados de blocos de pilha) têm saliências na parte inferior e/ou nas reentrâncias na parte superior que podem ser unidos na forma de pilhas.

Os blocos de *trigger* (também chamados de chapéus) têm a parte superior arredondada porque são colocados no topo de uma pilha. Esses blocos de *trigger* conectam eventos a *scripts*. Eles esperam por um evento – por exemplo, um pressionamento da tecla barra de espaço ou um clique do mouse – e executam os blocos que estão abaixo deles quando esse evento ocorre.

Os blocos de função (também chamados, informantes) não têm saliências nem reentrâncias. Eles não podem construir uma camada de um *script* sozinhos, em vez disso, são usados como entradas para outros blocos. Os formatos desses blocos indicam o tipo de dado que eles retornam. Por exemplo, os blocos com extremidades arredondadas informam números ou strings (cadeia de caracteres-letras), enquanto os blocos com as extremidades pontiagudas informam se algo é verdadeiro ou falso. Alguns blocos de função apresentam uma caixa de seleção ao lado deles. Se a caixa for selecionada, um monitor aparecerá no Palco para mostrar o valor corrente do bloco informante. Por exemplo, selecione um *sprite* e marque a caixa de seleção do bloco [posição x – paleta movimento]. Em seguida arraste o *sprite* pelo palco, observe que o monitor será alterado à medida que o *sprite* for movido. Alguns exemplos dos tipos de blocos podem ser vistos na Figura 16.

Figura 16- Tipos de blocos do Scratch

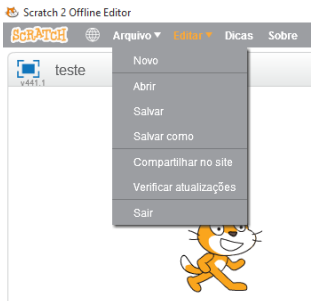


Fonte 16: Screenshot do Scratch (pelo autor)

3.1.2.5 *A barra de Menu*

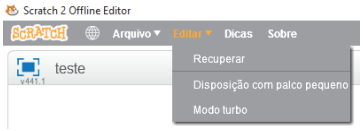
Após a logomarca *Scratch*, encontra-se o ícone de seleção de idiomas. Em seguida vem a barra de menus. A partir do *menu* Arquivo, pode-se criar novos projetos, abrir um projeto existente, salvar o projeto ou desfazer todas as alterações feitas no projeto corrente conforme Figura 17. Os projetos do *Scratch* 2 têm uma extensão de arquivo .sb2 para distingui-los dos projetos criados com a versão anterior (.sb). No *menu* Editar, a opção Recuperar trará de volta o último bloco, *script*, *sprite*, fantasia, ou som que por acaso tenha sido apagado. A opção Palco pequeno, proporciona mais espaço para a Área de *Scripts*. A opção Modo Turbo, aumenta a velocidade de execução de alguns blocos.

Figura 17 - Barra de menus do Scratch



Fonte 17 : Screenshot do Scratch (pelo autor)

Figura 18 - Opção de menu Editar do Scratch



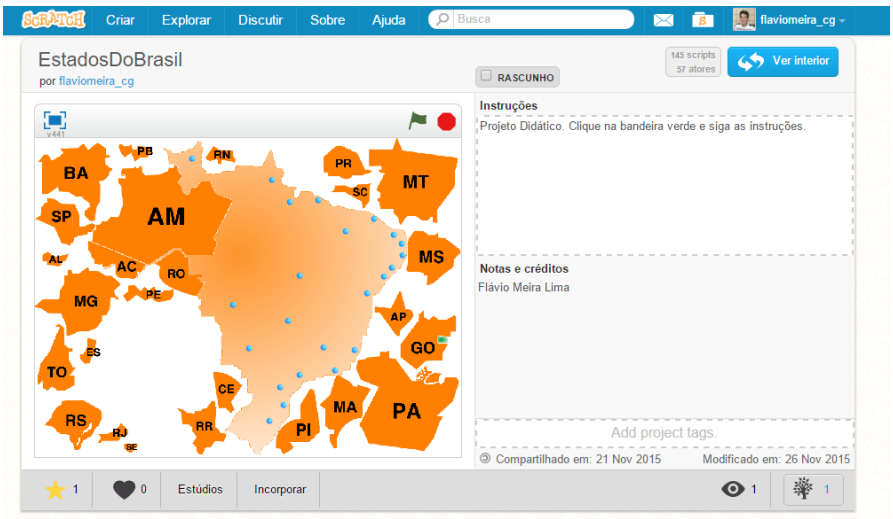
Fonte 18: Screenshot do Scratch (pelo autor)

Se o usuário estiver logado (autenticado) aparecerá logo após a barra de ferramentas as opções de compartilhar o projeto (Figura 19) e as informações sobre o projeto (autor, instruções, número de sprites, remixagens, etc) para edição conforme Figura 20, como também uma área de Mochila (parte inferior à direita) que permite usar *Sprites e Scripts* de projetos existentes conforme Figura 21.

Figura 19- Opções de compartilhar e mochila do Scratch

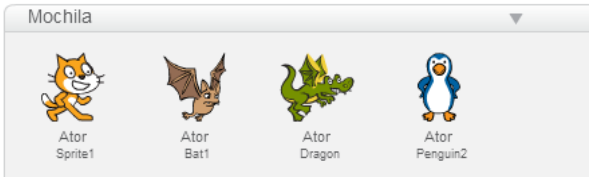
Fonte 19: Screenshot do Scratch (pelo autor)

Figura 20 - Informações do projeto



Fonte 20: Screenshot do Scratch (pelo autor)

Figura 21 - Mochila do Scratch



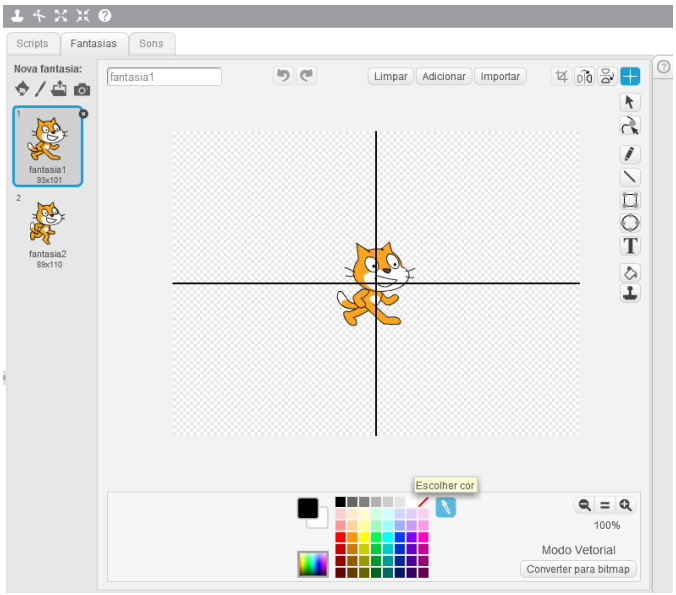
Fonte 21: Screenshot do Scratch (pelo autor)

3.1.2.6 *O Paint Editor*

O Paint Editor é utilizado para criar ou editar fantasias e panos de fundo. É um editor simples dado a natureza e finalidade do software. Seus ícones e funcionalidades são muito intuitivos e com criatividade pode-se fazer projetos muito interessantes. Nesse ponto vale ressaltar dois recursos: definir o centro de uma imagem e a cor transparente.

Ao dar um comando para que um *Sprite* vire (para a esquerda ou para a direita), ele irá girar em relação a um ponto de referência – o centro de sua fantasia. O botão (especificar centro da fantasia, no canto superior direito do *Paint Editor*) permite selecionar esse centro. Ao clicar nesse botão aparecerá uma cruz na área do desenho. O ponto central é determinado pela intersecção desses dois eixos, para alterar o centro da imagem basta arrastá-los para uma nova posição conforme Figura 22.

Figura 22- Paint Editor e o centro de imagem



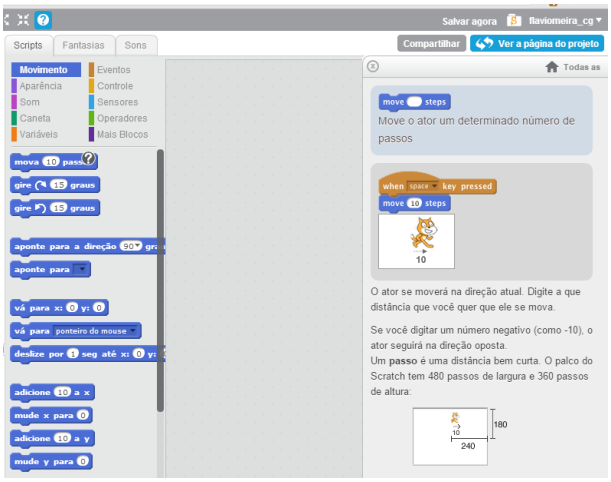
Fonte 22: Screenshot do Scratch (pelo autor)

Outra funcionalidade muito importante do *Paint Editor*, é a capacidade definir uma cor transparente. Recurso muito versátil e de grande utilidade, principalmente na elaboração de jogos e animações. Quando duas imagens se sobrepõem, a imagem de cima cobrirá algumas partes da imagem de baixo. De modo semelhante, os *sprites* cobrem partes do Palco. Para se obter a aparência do Palco por trás de uma imagem, será necessário usar o *Paint Editor* para deixar pelo menos parte da imagem transparente. Na paleta de cores, basta clicar no quadrado com uma linha vermelha na diagonal e pintar com essa cor "transparente" para deixar algo invisível. O *Paint* Editor, definindo o centro da imagem e a opção de cor transparente.

3.1.3 *Blocos das Categorias do Scratch*

O *Scratch* possui 122 blocos agrupados por categoria: Movimento, Aparência, Som, Caneta, Variáveis, Controle, Sensores, operadores e mais blocos. Como dito no início, o objetivo deste trabalho não é ser um manual ou guia de referência. Estes podem ser encontrados na plataforma do Scratch com muitos exemplos de projetos e fóruns de discussões. Será feito apenas uma breve descrição de que trata cada categoria. Uma funcionalidade muito importante pra quem está iniciando no *Scratch*, é utilizar a opção de ajuda de contexto que fica localizada no *ícone* representado por uma interrogação "?" na barra de ferramentas. Ao clicar nessa opção, basta arrastar o símbolo de interrogação para o bloco que deseja informações e dar um clic. Será apresentado uma tela de ajuda com um pequeno exemplo conforme mostra a Figura 23.

Figura 23 - Ajuda de contexto do Scratch

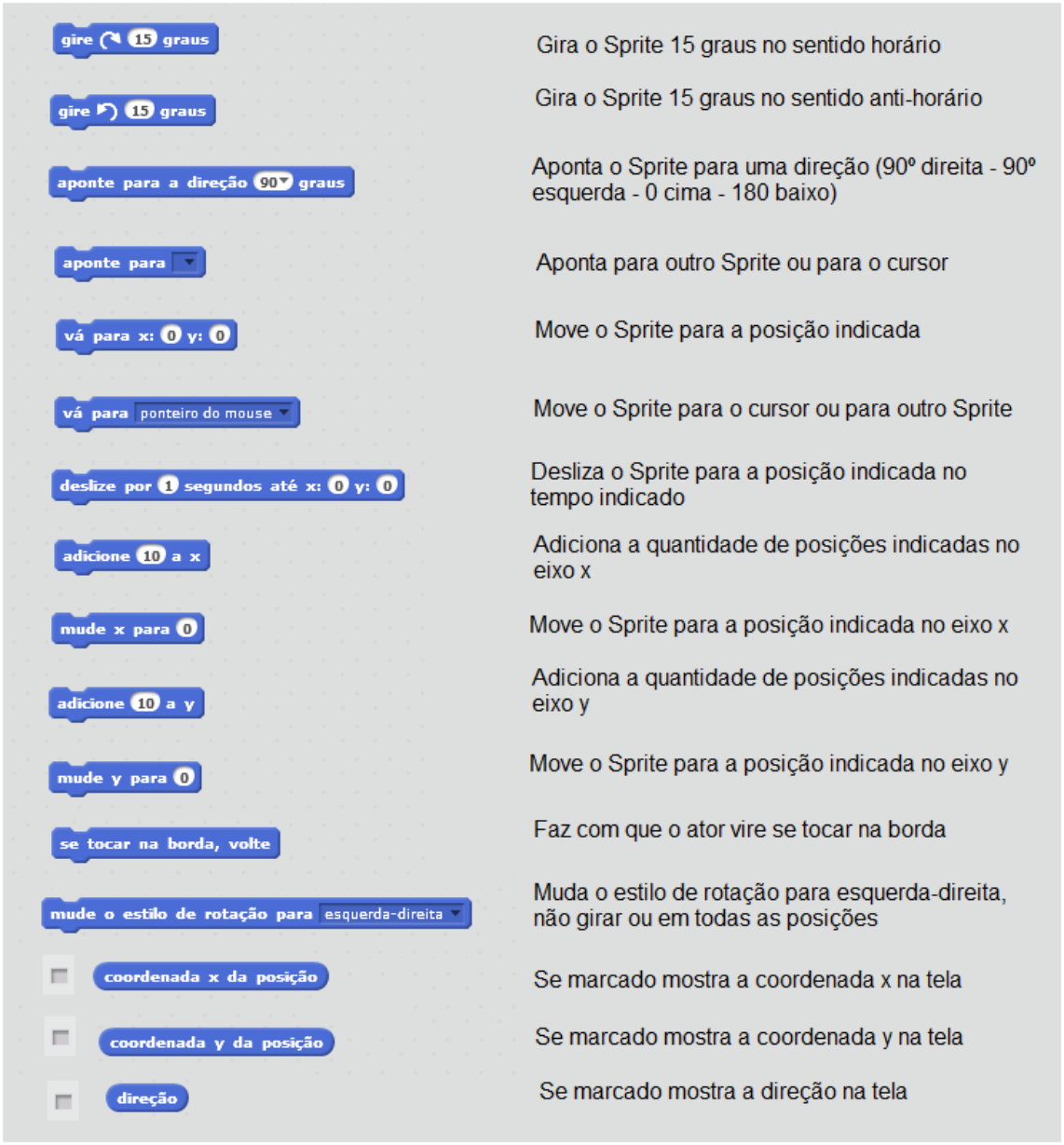


Fonte 23: Screenshot do Scratch (pelo autor)

3.1.3.1 *Blocos de Movimento:*

Nesta categoria encontram-se todos os blocos de deslocamentos, giros e posicionamentos dos *sprites* que serão montados seguindo a lógica do que se pretende fazer. (Figura 24).

Figura 24 - Categoria MOVIMENTO do Scratch

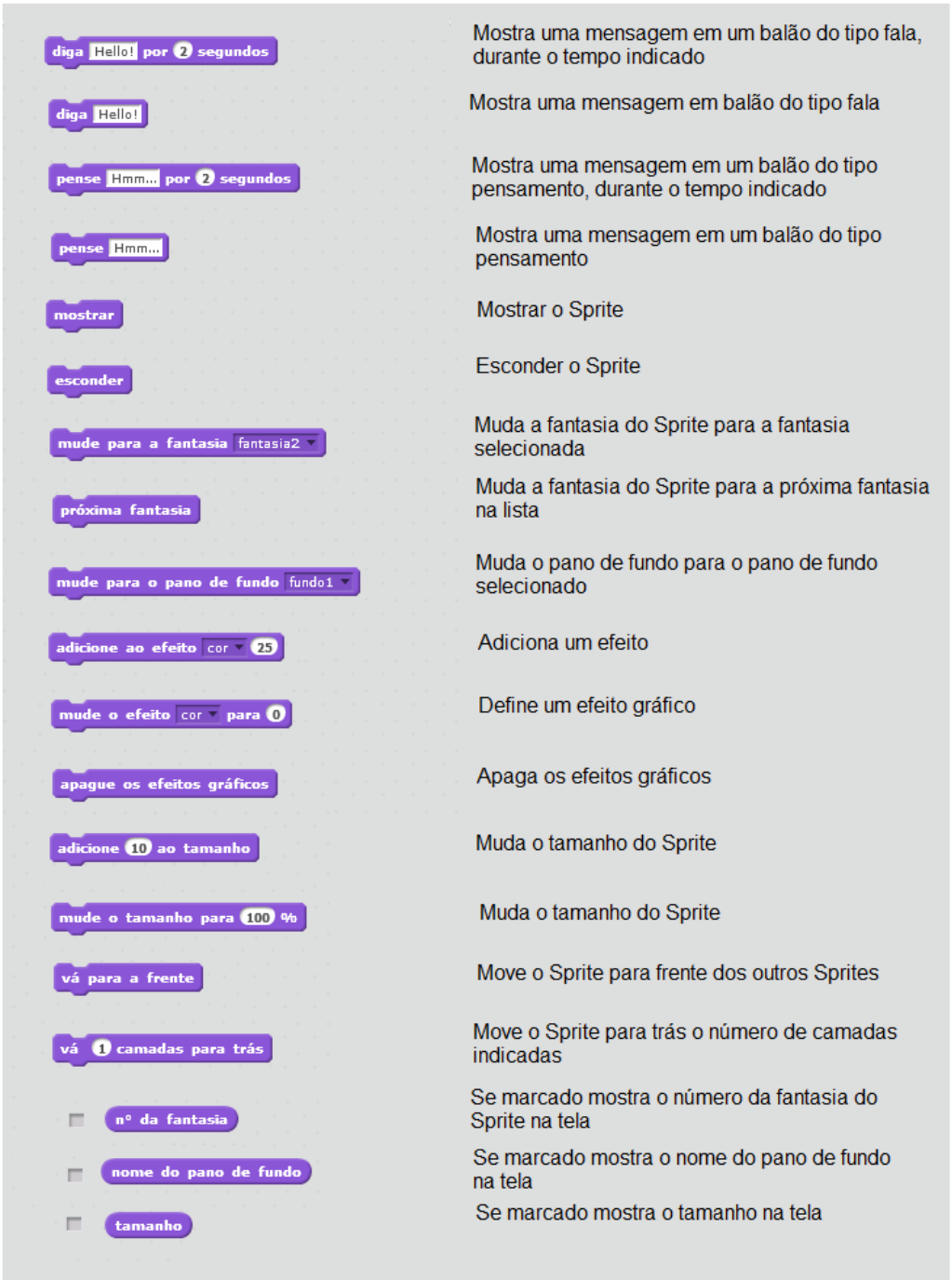


Fonte 24: (HACKERS, 2015)

3.1.3.2 *Blocos de Aparência:*

Esta categoria é responsável pelos efeitos que são dados ao *sprite*: mudar a fantasia, aparecer, desaparecer, efeitos de cores, tamanho, etc. Uma funcionalidade muito importante nessa categoria, é a relacionada a camadas. As camadas são níveis em que os *sprites* podem se sobrepor uns sobre os outros. Pode-se subir ou descer camadas como também apresentar o *sprite* em palcos diferentes no mesmo projeto. Os efeitos de aparência são muito utilizados em jogos e animações. (Figura 25).

Figura 25 - Categoria APARÊNCIA do Scratch



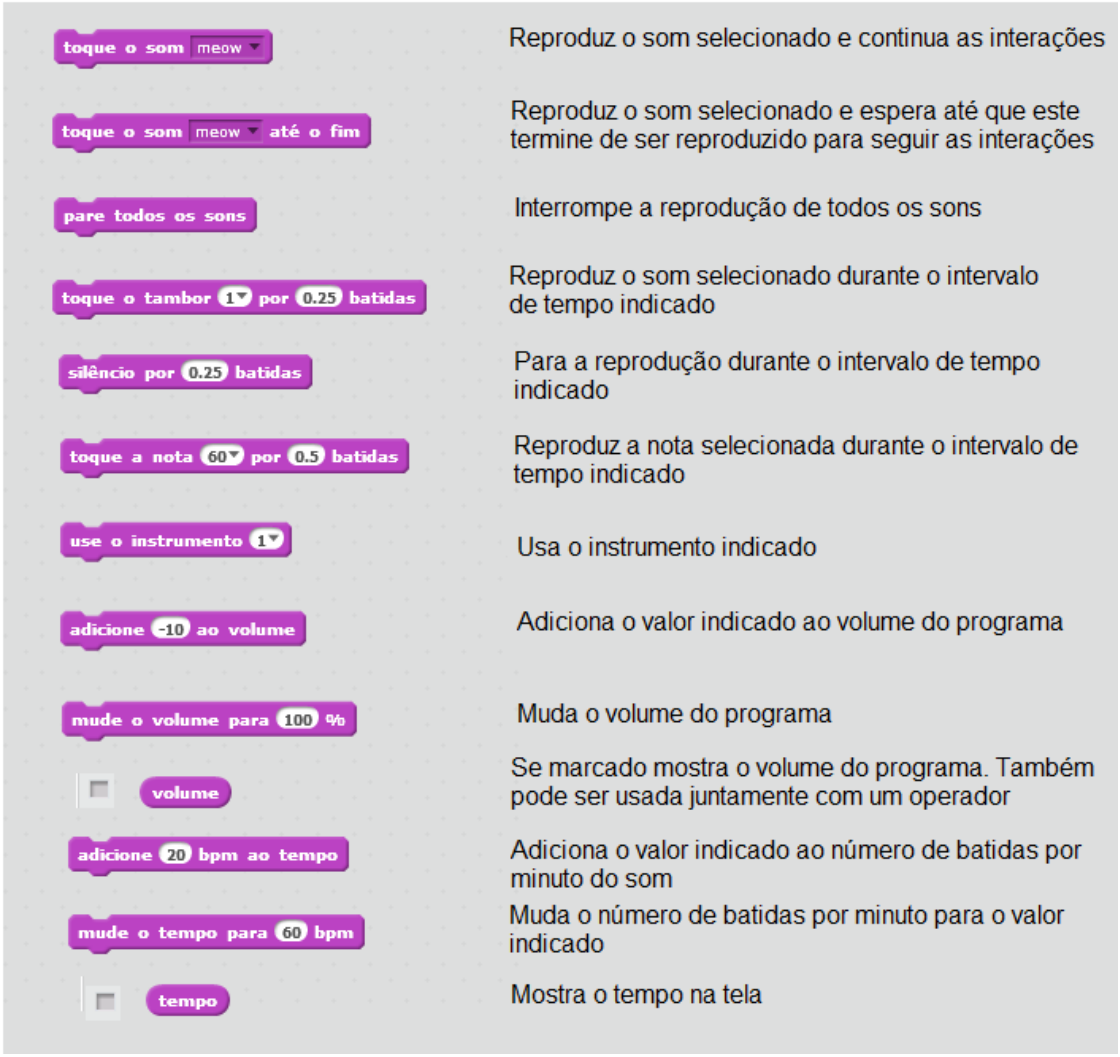
Fonte 25: (HACKERS, 2015)

3.1.3.3 *Blocos de Som:*

Os *Scratch* traz uma biblioteca de sons que simula os efeitos de alguns instrumentos musicais (funcionalidade do LOGO) como também alguns efeitos de sons que podem ser

aplicados em jogos. Mas, não se limita a essa biblioteca. Pode ser adicionado qualquer efeito de som. (A Figura 26).

Figura 26 - Categoria SOM do Scratch

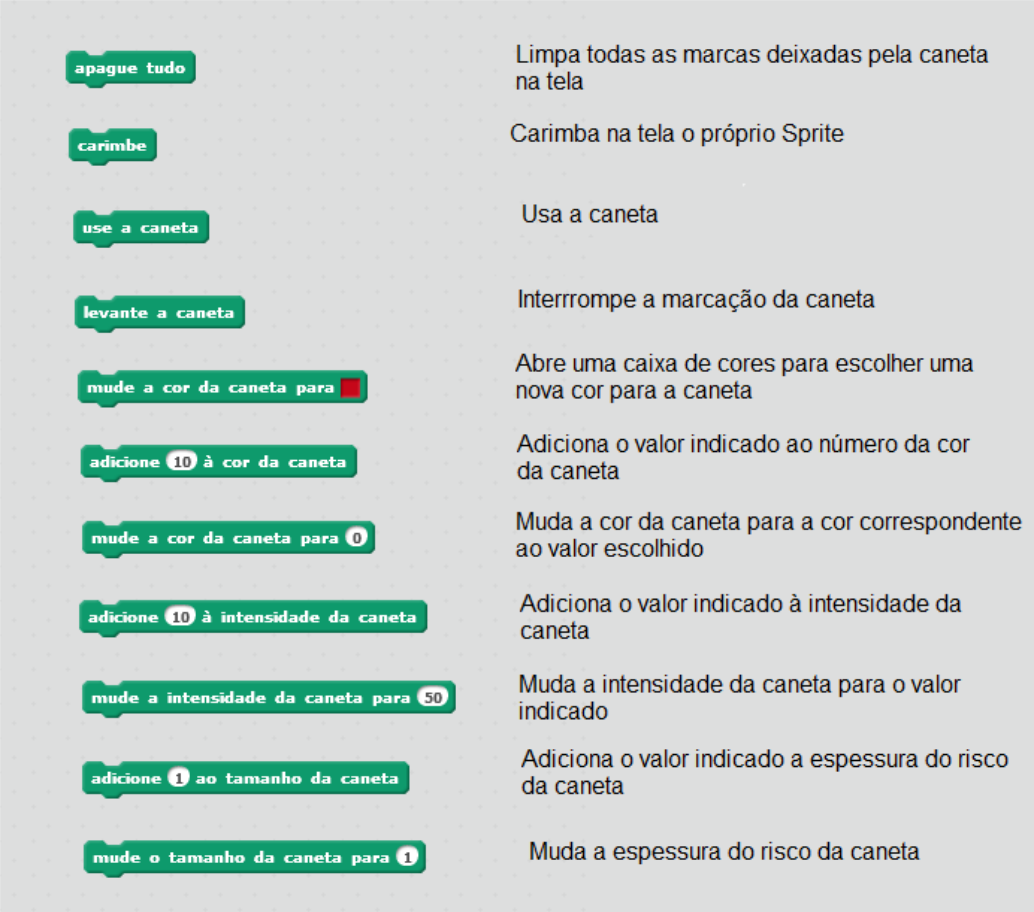


Fonte 26: (HACKERS, 2015)

3.1.3.4 *Blocos Caneta:*

Essa categoria contém os blocos necessários para se desenhar com o *Scratch* e todos os efeitos de que simula uma caneta: cores, espessura, etc. Uma funcionalidade interessante é o bloco: carimbe, que duplica o *sprite* em tempo de execução simulando um carimbo. Neste trabalho alguns projetos foram desenvolvidos mostrando como se pode trabalhar geometria em sala de aula. (Figura 27).

Figura 27 - Categoria CANETA do Scratch

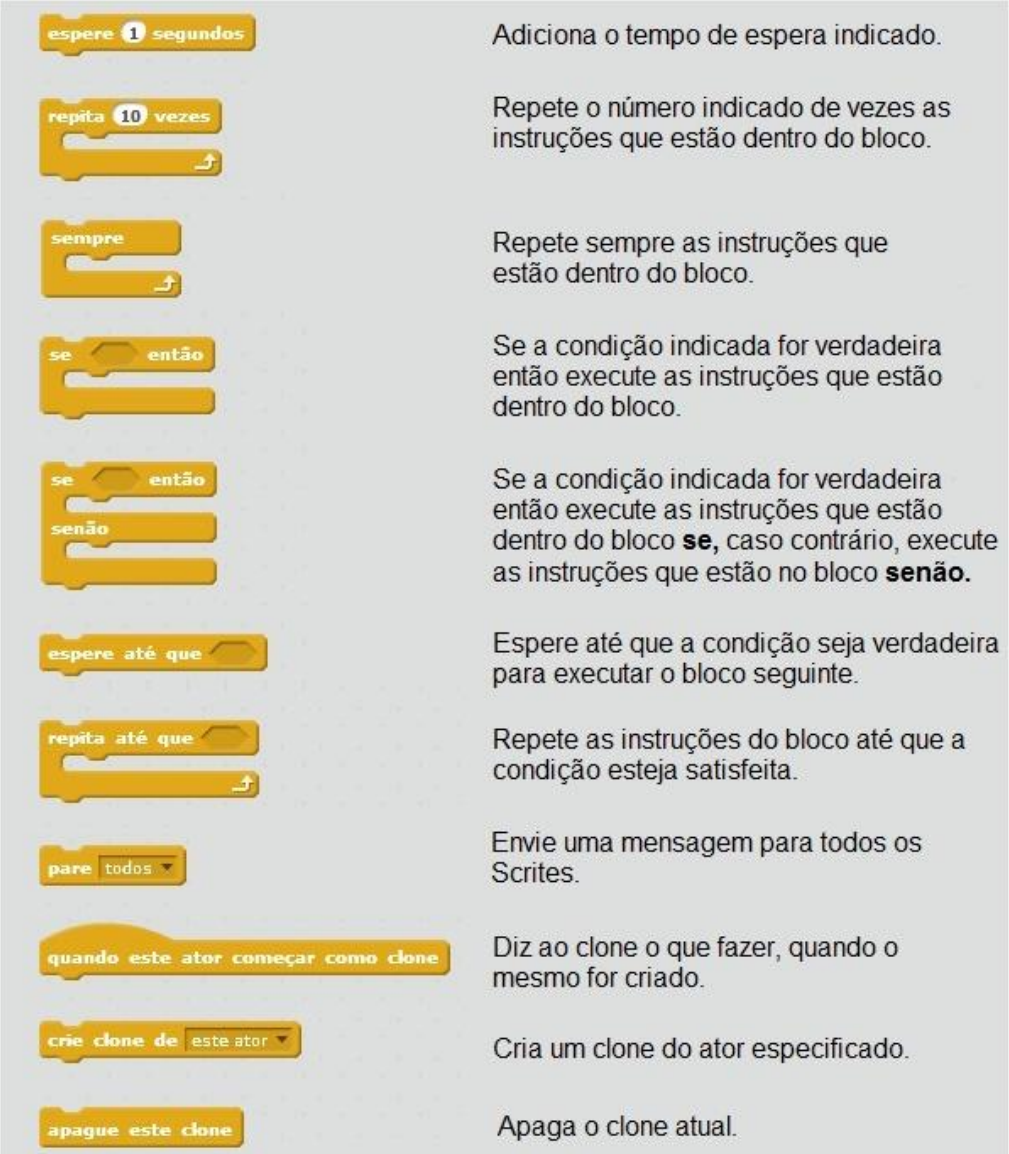


Fonte 27: (HACKERS, 2015)

3.1.3.5 *Blocos Controle:*

Nesta categoria estão disponíveis todos os blocos responsáveis pelas estruturas da lógica de programação tais como: repetições, desvios, chamadas de eventos, espera, etc. (Figura 28).

Figura 28- Categoria CONTROLE do Scratch

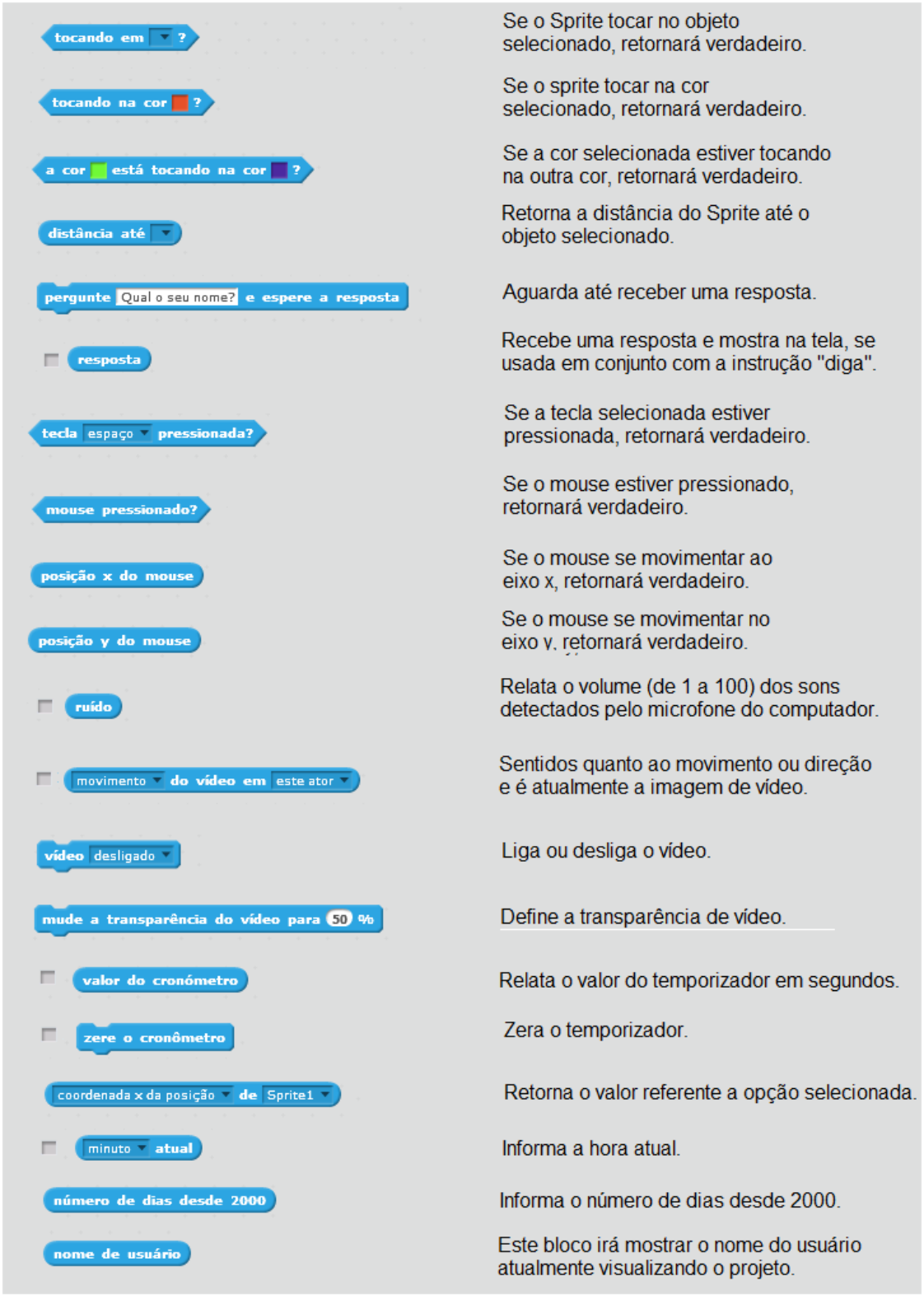


Fonte 28: (HACKERS, 2015)

3.1.3.6 *Blocos Sensores:*

Esta categoria apresenta os blocos que são comumente usados em jogos e animações tais como: toques, distâncias, posicionamento, cronômetro, eventos do mouse e do teclado. (Figura 29).

Figura 29 - Categoria SENSORES do Scratch

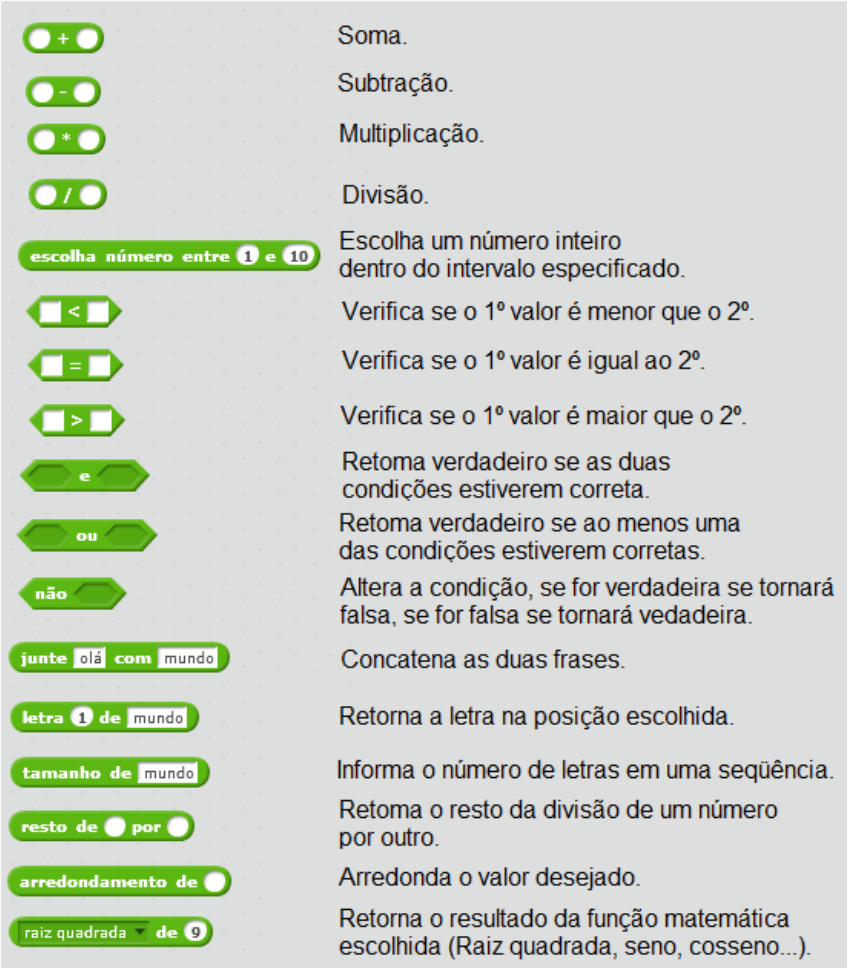


Fonte 29: (HACKERS, 2015)

3.1.3.7 *Blocos Operadores:*

Esta categoria apresenta os blocos necessários para se realizar as operações matemáticas como também juntamente com outros blocos realizar toda lógica *booleana* comumente usada em linguagens de programação. (Figura 30).

Figura 30- Categoria - OPERADORES



Fonte 30: (HACKERS, 2015)

3.1.3.8 *Blocos de Variáveis:*

O conceito inicial mais importante em qualquer linguagem de programação é o de variável. Assim como o conceito de variável em matemática, em linguagem de programação, uma variável é uma posição de memória do computador que pode assumir qualquer valor: texto ou número. Dependendo do valor atribuído a variável, ela será do tipo texto ou número. Com as variáveis pode-se trabalhar: controles de

fluxo, contagens, *flags* (sinalizadores lógicos). Ao clicar nessa categoria na opção: criar uma variável, será apresentada uma caixa de diálogo para nomear a variável que será criada; em seguida a variável será criada juntamente com cinco blocos e atribuições iniciais relacionados a variável para que sejam manipulados no projeto. (Figura 31).

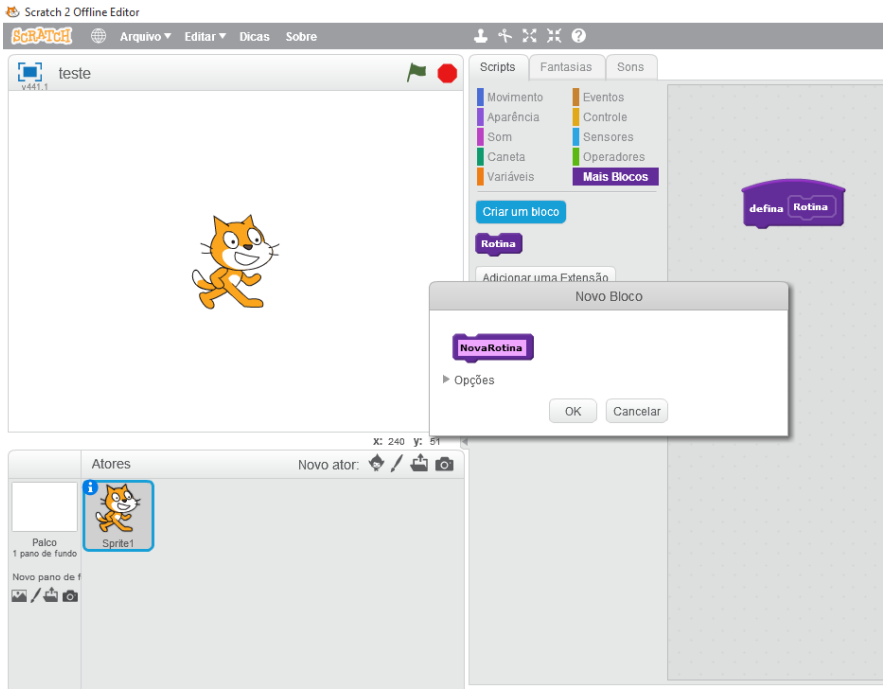
Figura 31 - Categoria VARIÁVEIS do Scratch

Fonte 31: (HACKERS, 2015)

3.1.3.9 *Mais Blocos*

Esta funcionalidade só está presente a partir da versão 2, foi a mudança mais importante no projeto de *Scratch*, pois com essa funcionalidade o usuário pode construir suas próprias funcionalidades como se tivesse construindo uma nova categoria. Em linguagens de programação chama-se: funções ou procedimentos. Ao clicar na opção mais blocos, será apresentado uma caixa de diálogo para nomear o bloco (procedimento) e logo após será criados 2 blocos, o que vai receber os scripts e o que vai ativar o procedimento conforme Figura 32.

Figura 32- Categoria MAIS BLOCOS do Scratch



Fonte 32: Screenshot do SCRATCH (pelo autor)

3.2 Competências e conceitos de programação explorados no Scratch

Através do processo de criação de histórias interativas, jogos e animações com *Scratch*, os jovens podem aprender importantes conceitos e competências sobre computadores.

3.3 Competências para resolução de problemas e para concepção de projetos

- raciocínio lógico
- identificação e eliminação de erros
- desenvolvimento de ideias, desde a concepção até à concretização do projeto

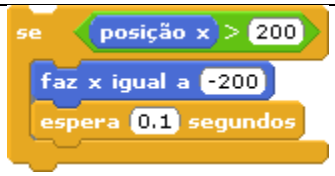
- concentração e perseverança

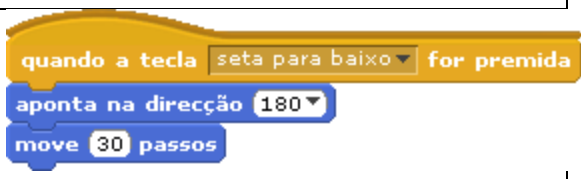
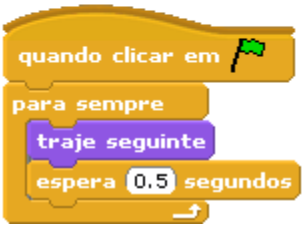
3.4 Noções Básicas sobre Computadores e Programação

- os programas indicam aos computadores exatamente o que devem fazer, passo a passo.
- criar programas para computador não exige perícia especial, apenas raciocínio claro e cuidadoso

3.5 Conceitos Específicos de Programação

As figuras abaixo apresentadas são *prints screens do Scratch* elaboradas pelo autor.

Conceito	Descrição	Exemplo
SEQUÊNCIA	Para criar um programa Scratch é preciso pensar de forma sistemática na ordem de execução das instruções.	
INTERAÇÃO (CICLOS)	repete e para sempre podem usar-se para repetir um bloco de instruções.	
INSTRUÇÕES CONDICIONAIS	se e se...senão verificam a ocorrência de uma condição.	

VARIÁVEIS	A categoria Variáveis permite a criação de uma nova variável e a sua utilização num programa. As variáveis podem conter números ou sequências de caracteres (texto). O *Scratch* permite a definição de variáveis globais e locais.	
GESTÃO DE EVENTOS	quando a tecla for pressionada e quando clicar em *sprite* são exemplos de gestão de eventos – resposta a eventos invocados pelo utilizador ou por outra seção de um programa.	
EXECUÇÃO PARALELA	Lançar dois blocos de comandos ao mesmo tempo cria dois fluxos independentes que são executados em paralelo.	 

COORDENAÇÃO E SINCRONIZAÇÃO	anuncia e quando receber permitem coordenar a execução de múltiplos blocos de comandos. anuncia e espera possibilita a sincronização de execução.	Por exemplo, *Sprite1* envia a mensagem vitória quando a condição é verificada:  Este bloco de comandos do *Sprite2* é acionado quando a mensagem vitória é recebida: 
ENTRADA DE DADOS VIA TECLADO	pergunta e espera solicita ao utilizador que escreva. resposta armazena o que foi escrito no teclado.	
NÚMEROS ALEATÓRIOS	O bloco sorteia número escolhe números inteiros aleatoriamente dentro de uma certo intervalo.	
LÓGICA BOLEANA	e, ou e não são exemplos de lógica *boleana*.	
INTERAÇÃO EM TEMPO REAL	x do mouse, y do mouse e volume do som podem ser usados dinamicamente para interação em tempo real.	

DESENHO DE INTERFACE DO UTILIZADOR	No *Scratch* é possível desenhar interfaces de utilizador interativas – por exemplo usando *sprites* clicáveis para criar botões.	
LISTAS	Os comandos relativos a listas permitem armazenar e acessar a conjuntos ordenados de números e texto. São estruturas de dados dinâmicas sobre as quais é possível efetuar operações de adição, enumeração, substituição e remoção de elementos.	

3.6 Conceitos de Programação atualmente não introduzidos no Scratch

- Gestão de exceções
- Definição de classes de objetos
- Herança
- Leitura e escrita em arquivos

3.7 Projetos desenvolvidos:

3.7.1.1 Projeto – 01

Nome: RotaçãoDeQuadrados.sb2.

Objetivo: Traçar vários quadrados apresentando um aspecto final de uma circunferência.

Conceitos Computacionais: Sequência; atribuição de valores; estrutura de repetição.

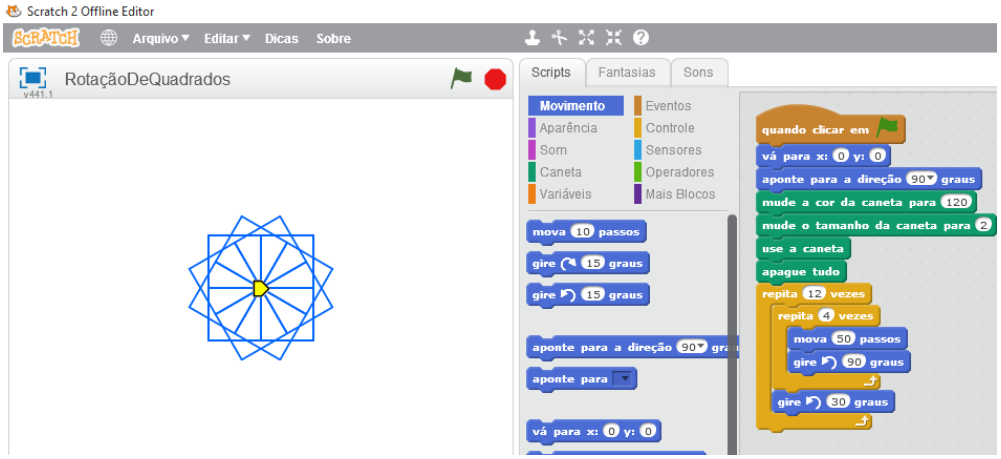
Recursos do Scratch: Movimentos, Caneta, Eventos e Controle. 1-Script;1-Ator.

Comentários: Essa atividade apresenta duas estruturas de repetições, uma mais externa e outra mais interna. A interna desenha um quadrado e desloca a caneta em 30 graus, após esse passo é que a estrutura mais externa é incrementada de mais um passo e assim sucessivamente, formado ao final os quadrados em forma de circunferência. É interessante modificar esse projeto alterando os valores, tanto do tamanho dos lados do quadrado, a quantidade de repetições externas ou o giro. Essa é a principal característica do *Scratch*, aprender se divertindo, modificando, executando, refletindo nos resultados, depurando e modificando novamente. (Figura 33).

Grau de dificuldade: Baixo.

Link de acesso: - https://scratch.mit.edu/projects/88720074/

Figura 33 - Projeto 1 – Rotação de Quadrados



Fonte 33: Screenshot do Scratch (pelo autor)

3.7.1.2 Projeto – 02

Nome: Circunferência.sb2.

Objetivo: Traçar várias circunferências modificando suas cores.

Conceitos Computacionais: Sequência; atribuição de valores e estrutura de repetição

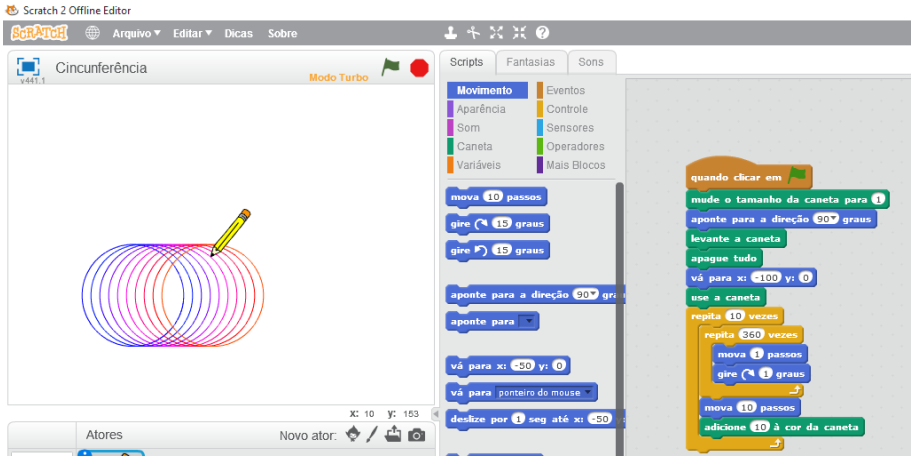
Recursos do Scratch: Movimentos, Caneta, Eventos e Controle. 1-Script; 1-Ator.

Comentários: Assim como no projeto anterior os quadrados deram um aspecto de circunferência, essa atividade constrói uma circunferência propriamente dita. É nesse momento que o aluno irá refletir e analisar a construção. E a partir de então modificar o projeto para fazer outras evoluções. (Figura 34).

Grau de dificuldade: Baixo

Link de acesso: https://scratch.mit.edu/projects/88720221/

Figura 34- Projeto 2 - Circunferência



Fonte 34: Screenshot do Scratch (pelo autor)

3.7.1.3 Projeto – 03

Nome: Esfera.sb2.

Objetivo: Traçar várias circunferências apresentando um aspecto 3D ao final

Conceitos Computacionais: Sequência; atribuição de valores; estrutura de repetição e chamada de procedimento.

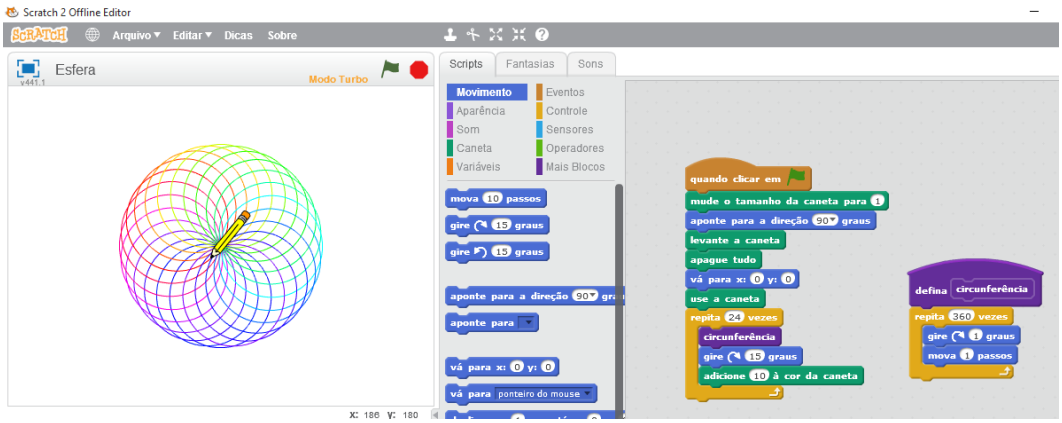
Recursos do Scratch: Movimentos, Caneta, Eventos, Controle e Mais Blocos. 1-Script;1-Ator.

Comentários: Essa atividade é muito semelhante a anterior, apenas foi dado o deslocamento necessário para simular uma esfera. A construção fica lenta pois são necessários vários passos. É para situações como essa que existe a opção modo turbo. Vá na opção: Editar do menu principal do *Scratch* e selecione a opção: Ativar modo turbo. Aparecerá na parte superior uma mensagem dessa opção ativada. Execute novamente a aplicação e o *Scratch* realizará a tarefa rapidamente. (Figura 35).

Grau de dificuldade: Médio.

Link de acesso: https://scratch.mit.edu/projects/88720310/

Figura 35 - Projeto 3 - Esfera



Fonte 35: Screenshot do Scratch (pelo autor)

3.7.1.4 Projeto – 04

Nome: Poligonos.sb2.

Objetivo: Traçar vários polígonos à medida que vai aumentando o número de lados

Conceitos Computacionais: Sequência; atribuição de valores; estrutura de repetição e chamada de procedimento.

Recursos do Scratch: Movimentos, Caneta, Variáveis, Eventos, Controle e Operadores.

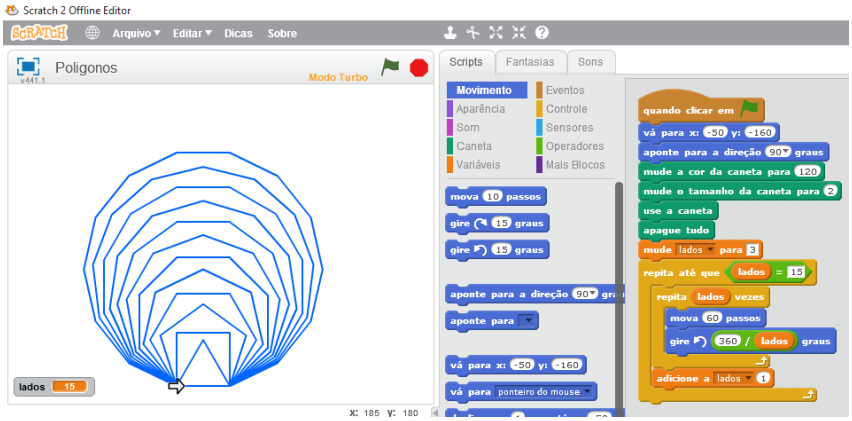
1-Script; 1-Ator.

Comentários: Essa atividade é bem didática e pode ser muito bem explorada com a mediação de um professor de matemática. Modifique os valores para realizar novos polígonos. (Figura 36).

Grau de dificuldade: Médio.

Link de acesso: - https://scratch.mit.edu/projects/88720156/

Figura 36 - Projeto 4 - Polígonos



Fonte 36: Screenshot do Scratch (pelo autor)

3.7.1.5 Projeto – 05

Nome: Petalas.sb2.

Objetivo: Traçar várias semi-circunferências para simular as pétalas de uma flor.

Conceitos Computacionais: Sequência; atribuição de valores; estrutura de repetição e chamada de procedimento. (Figura 37).

Recursos do Scratch: Movimentos, Caneta, Variáveis, Eventos, Controle e Operadores.

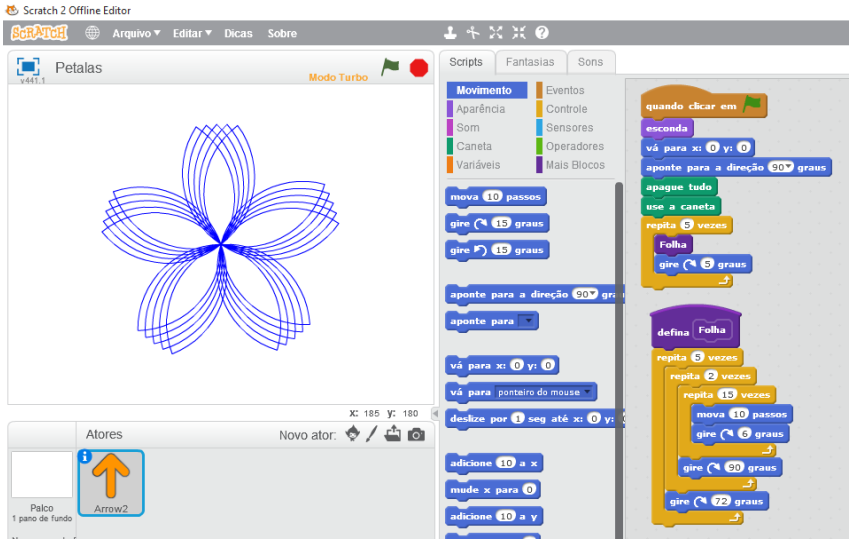
2-Scripts;1-Ator

Comentários: Essa atividade tem instruções simples, mas possui uma lógica bastante sutil, que também pode ser explorada mediada por um professor de matemática, modificando e provocando os alunos para construírem novos desafios.

Grau de dificuldade: Elevado.

Link de acesso: - https://scratch.mit.edu/projects/88720381/

Figura 37- Projeto 5 - Petalas



Fonte 37: Screenshot do Scratch (pelo autor)

3.7.1.6 Projeto – 06

Nome: ProjGeometria.sb2.

Grau de Dificuldade: Elevado.

Objetivo: Incorporar em uma única aplicação os projetos acima apresentados em uma lógica computacional mais refinada.

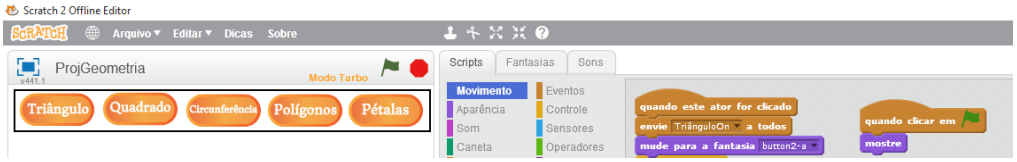
Conceitos Computacionais: Sequência; atribuição de valores; estrutura de repetição, chamada de procedimento, tratamento de eventos e recursividade.

Recursos do Scratch: Movimentos, Aparência, Caneta, Variáveis, Eventos, Controle, Operadores, e Mais Blocos. 68-Scripts; 6-Atores.

Comentários: Esse projeto apresenta os mesmos conceitos de construções geométricas antes descritas, mas de uma forma mais elaborada com a construção de um *menu* de eventos e blocos de procedimentos organizados. Apresenta também algumas aplicações com triângulos e explora também o conceito de recursividade. Recursividade é quando um bloco de instrução está interno a um procedimento e chama a si próprio. Mas, deve-se ter bastante cuidado para construções desse tipo, pois pode facilmente deixar o programa eternamente em execução. Passo esse que em programação chama-se de *loop* infinito. Sendo necessário abortar a aplicação. (Figuras 38 e 39).

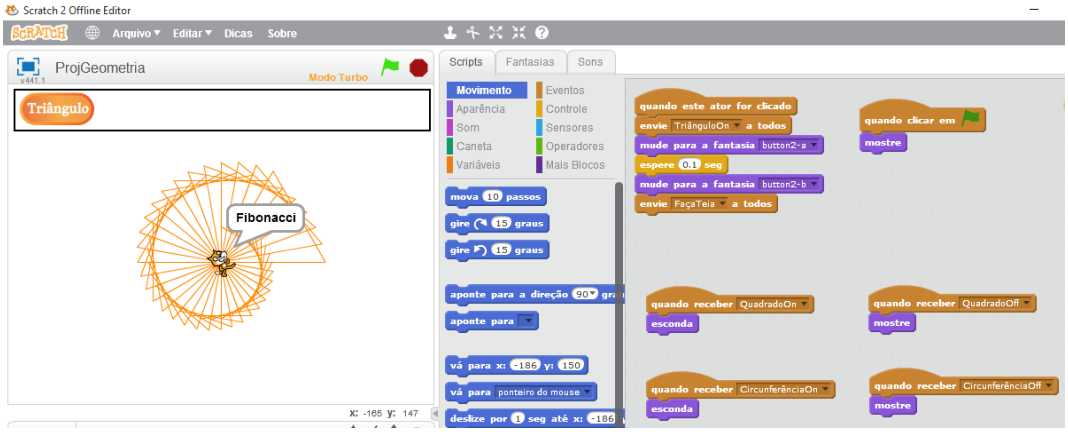
Link de acesso: https://scratch.mit.edu/projects/88723036/

Figura 38 - Projeto 6 - Geometria



Fonte 38: Screenshot do Scratch (pelo autor

Figura 39 - Projeto 6 - Geometria



Fonte 39: Screenshot do Scratch (pelo autor)

3.7.1.7 Projeto – 07

Nome: EstadosDoBrasil.sb2.

Objetivo: Posicionar os Estados do Brasil no mapa ao arrastar e clicar do mouse.

Conceitos Computacionais: Sequência; atribuição de valores; estrutura de repetição, chamada de procedimento e tratamento de eventos.

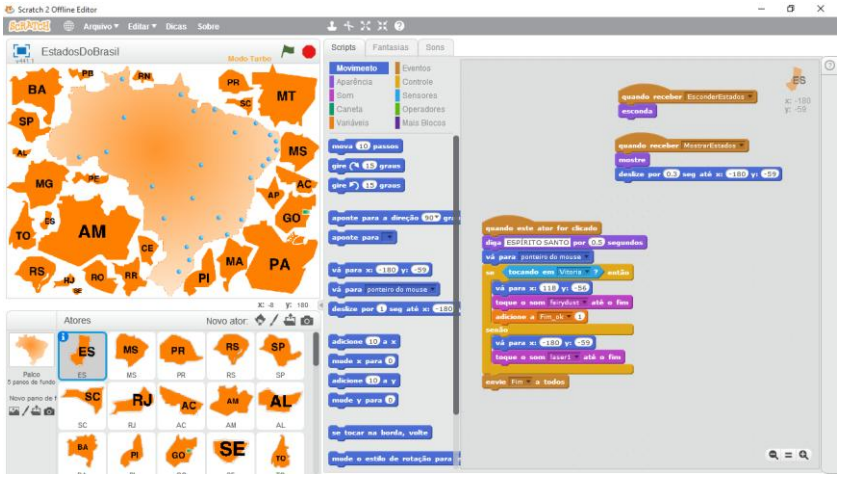
Recursos do Scratch: Movimentos, Aparência, Som, Variáveis, Eventos, Controles, Sensores. 145-Scripts;45-Atores.

Comentários: Esse projeto apresenta dois novos recursos: Som e Sensores. Torna-se trabalhoso, pois requer uma grande quantidade de scripts e atores, mas sua lógica é bastante simples e repetitiva. A finalidade é mostrar que o Scratch pode ser aplicado em qualquer área educacional. Nesse caso, trabalha-se Geografia.

Grau de dificuldade: Médio.

Link de acesso: - https://scratch.mit.edu/projects/88720819/

Figura 40 - Projeto 7 - Estados do Brasil



Fonte 40: Screenshot do Scratch (pelo autor)

3.8 Aplicação em jogos

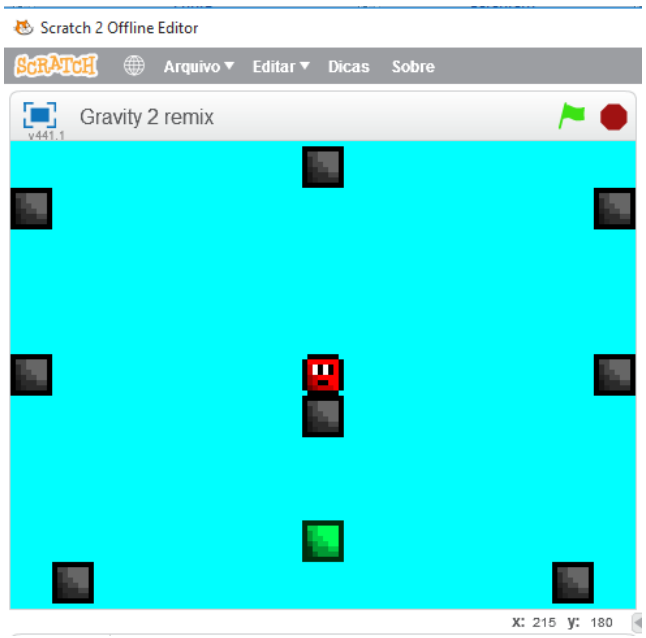
Nome: Gravity 2.sb2

Autor: steve115

Link de Acesso: <https://scratch.mit.edu/projects/82996216/>

Descrição: Talvez a parte mais cativante do *Scratch* seja a criação de jogos e o ambiente corporativo da plataforma em que projetos podem ser compartilhados e *remixados* (reutilizados). Este é um projeto exemplo em destaque na plataforma do *Scratch.*

Figura 41 - Tela do Jogo Gravity 2



Fonte 41: Screenshot do Jogo Gravity 2 (pelo autor)

4 ANÁLISES DOS RESULTADOS

Os exemplos de projetos apresentados focam em duas áreas do conhecimento: Matemática e Geografia, e voltado para educadores, como forma de incentivar os alunos na construção do conhecimento. Mas, é extensivo para qualquer disciplina desde o ensino fundamental, médio ou superior. Como os blocos de instruções por si só são bastante elucidativos, faz-se desnecessário um passo a passo, e como a linguagem é visual, é bem mais simples acompanhar a aplicação tentando reconstruí-la (*remixar)*.

Foi apresentado a cada exemplo uma nova funcionalidade, uma nova técnica, de forma a apresentar de uma maneira generalizada, todas as dez categorias (paletas) do *Scratch:* Movimento, Aparência, Som, Caneta, Variáveis, Eventos, Controle, Sensores, Operadores e Mais Blocos.

O projeto: ProjGeometria, engloba todos os projetos desenvolvidos inicialmente (Projeto 01 ao Projeto 06) reescrito com uma técnica computacional: chamada de procedimentos; apenas para demonstrar um projeto mais elaborado e para explorar outras funcionalidades do software que o usuário vai adquirindo à medida que se aprofunda na ferramenta, mas sem perder a lógica inicial dos projetos anteriores, ou seja, para mostrar que a partir de construções simples pode-se chegar a um projeto final mais sofisticado com uso de *menus* e botões interativos que respondem ao evento de um *clic* do *mouse*.

Já o projeto: EstadosDoBrasil, apesar de conter um números elevado de *scripts* e atores, sua lógica é bastante simples e repetitiva, a dificuldade maior está no tratamento das imagens e no *designer*. Ou seja, requer um conhecimento maior em como capturar as imagens na internet e tratá-las. Esse também foi elaborado de forma propositiva para mostrar como um projeto pode ter uma lógica simples, mas ser bastante trabalhoso na sua construção.

Ao final, foi apresentado um jogo disponível na plataforma do *Scratch*. O objetivo é mostrar o compartilhamento de projetos e que depois que o ao usuário adquirir maturidade com o software pode desenvolver aplicações de qualidade profissional.

Enfim, os projetos apresentados são apenas exemplos do que se pode trabalhar em sala de aula. Claro, que como qualquer formação, faz-se necessário um curso introdutório com algum profissional da área de computação na ferramenta, ou até mesmo de forma autodidata a depender perfil do usuário. Mas, inúmeros exemplos práticos do mais simples aos mais elaborados podem ser encontrados no site do S*cratch*. Mais importante pra quem é professor e deseja utilizar a ferramenta é discutir informações com outros professores que já se utilizam

do software. Para tanto, o ambiente corporativo do *Scratch* tem uma seção com essa finalidade com fóruns de discussões.

5 CONSIDERAÇÕES FINAIS

São indiscutíveis os avanços tecnológicos alcançados pelo homem no século XXI mais notadamente a partir do advento da *internet*. A cada dia surge um novo equipamento eletrônico, um novo *software*, formando dois grupos com relação à utilização das novas tecnologias: os nativos e os imigrantes digitais (PRENSKY, 2001), os nativos já nascidos ao meio e os imigrantes tentando adaptar-se. No entanto, as tentativas em usar o computador de forma adequada como uma ferramenta que possa potencializar o ensino-aprendizagem na educação não é recente. Também não foi objeto de estudo deste trabalho discutir a utilização do computador em si na educação, mas a aplicação do software *Scratch* como uma ferramenta catalisadora do desenvolvimento do raciocino lógico-matemático e computacional como fonte auxiliar para projetos de ensino desde as primeiras séries.

Seguindo a abordagem construtivista de Jean Piaget e construcionista de Seymour Papert; as ideias inovadoras propostas por Mitchel Resnick e o pensamento computacional; analisou-se o *SCRATCH* e suas possibilidades para uma aprendizagem significativa na era digital. Ou seja, o objetivo não foi formar programadores de computador, mas a usabilidade do *software* como uma ferramenta de apoio educacional e suas potencialidades em desenvolver o raciocínio dos aprendizes em situações-problemas usando a lógica computacional: operações matemáticas; repetições; desvios, etc.

Após analisar as funcionalidades do *Scratch* e os resultados obtidos com os projetos exemplos, constatou-se que o software é de fácil aprendizagem e segue a abordagem construtivista e construcionista inicialmente proposta pela linguagem de programação *LOGO* com o diferencial de ser bastante flexível com seus blocos de construções visuais e previamente codificados (códigos computacionais) bastando para tanto juntá-los como se fossem peças de um jogo de quebra-cabeças. Ou seja, o foco é criar uma situação problema e através do ciclo: descrição-execeção-reflexão-depuração-descrição, chegar à solução. Este é um ponto fundamental da ferramenta, pois o "erro" não existe, apenas observa-se se chegou ou não onde se desejava inicialmente. E é nesse processo que o raciocínio é desenvolvido.

Conclui-se que o *software* educacional *SCRATCH* atendeu todas as expectativas propostas, ou seja, potencializar o processo de ensino-aprendizagem para a formação de cidadãos criativos, críticos e reflexivos tão necessários para a educação do século XXI.

Como mais uma fonte de sugestões de aplicabilidades e projetos futuros com o *software*, segue abaixo alguns *links* de acesso de alguns projetos exitosos desenvolvidos no Brasil:

http://mutirao.upf.br/escoladehackers/

http://www.scratchbrasil.net.br/

http://www.computacaonaescola.ufsc.br/?p=113

http://projetoseducacionais.educapx.com/

6 REFERÊNCIAS

ALMEIDA, Maria Elizabeth B. T. M. P. de. O aprender e a informática: a arte do possível na formação do professor. **Cadernos Informática para a Mudança em Educação**. MEC/SEED/PROINFO, 1999.

ALMEIDA, Maria Elizabeth B. T. M. P. de. **O computador na escola**: contextualizando a formação de professores. Praticar a teoria e refletir a prática. 2000. Tese de Doutorado em Educação: Currículo. Pontifícia Universidade Católica de São Paulo, São Paulo, 2000.

ALMEIDA, F. J. 2012. **Educação e informática: Os computadores na escola**. 5ed - São Paulo: Cortez.

BOSSUET, Gérard. **O computador na escola: sistema Logo**. Porto Alegre: Artes Médicas, 1985.

COMMONS, Creative. Site ProjetoLogo, atualizado em 02/ago/2009. Disponível em: <http://projetologo.webs.com/texto1.html> Acesso em: 15/04/2014.

CORALINA, Cora. **Vintém de Cobre:** meias confissões de Aninha. 9 ed - São Paulo: Global Gaia, 2007.

HACKERS, Escola de. Site do Projeto Educacional com SCRATCH da Prefeitura Municipal de Passo Fundo. Rio Grande do Sul, 2015. Disponível em: < http://mutirao.upf.br/escoladehackers/?page_id=9 >. Acesso em: 26/11/2015.

MACEDO, Lino. **Ensaios Construtivistas.** 3. Ed. São Paulo : Casa do Psicólogo, 1994.
MARJI, Majed. **Aprenda a Programar com** SCRATC.Tradução de Lúcia.Kinoshita. São Paulo: Novatec, 2014.

MARCHI, Maria Teresinha Golon. **Informática na educação**: uma experiência na formação inicial do professor. 2001. Dissertação de Mestrado. Faculdade de Filosofia, Ciências e Letras de Jandaia do Sul, Jandaia do Sul, 2001.

MARQUES, M. T. P, M., 2009. **Recuperar o engenho a partir da necessidade, com recursos às tecnologias educativas**: Contributo do ambiente gráfico de programação Scratch em contexto formal de aprendizagem. Tese de doutorado, Universidade de Lisboa. Disponível em: < http://repositorio.ul.pt/handle/10451/847> Acesso em 25/05/2015.

MARTINS, Amilton Rodrigo de Quadros. Usando o Scratch para Potencializar o Pensamento Criativo em Crianças do Ensino Fundamental. Dissertação de Mestrado. Universidade de Passo Fundo, 2012. Disponível em <http://www.upf.br/ppgedu/images/stories/defesa-dissertacao-amilton-rodrigo-de-quadros-martins.PDF> . Acesso em:17/09/2014.

PAPERT, Seymour. **Mindstorns: children, computers and powerful ideas**. New York: Basic Books, 1980.

PAPERT, Seymour**. The children's machine: rethinking school in the age of the computer The children's machine: rethinking school in the age of the computer**. New York: Basic Books, 1993

PAPERT, Seymour. **Logo**: computadores e educação. Tradução de José Armando Valente, Beatriz Bitelman. Afira V. Ripper. 2. ed. São Paulo: Brasiliense, 1986.

PAPERT, Seymour. **Logo:** computadores e educação. 3. ed. São Paulo: Brasiliense, 1988.

PAPERT, Seymour. **A máquina das crianças:** repensando a escola na era da informática. 1. ed. Porto Alegre: Artes Médicas, 1994.

PAPERT, S.; RESNICK, M. Technological Fluency and the Representation of Knowledge. Proposal to the National Science Foundation. MIT MediaLab (1995).

PIAGET, Jean. **Seis estudos de psicologia.** Rio de Janeiro: Forense, 1964.

PIAGET, Jean. **A equilibração das estruturas cognitivas.** Rio de Janeiro: Zahar, 1975.

PIAGET, Jean. **Epistemologia genética.** São Paulo: Martins Fontes, 1990.
PIAGET, Jean e INHELDER, Bärbel. **A psicologia da criança.** São Paulo: DIFEL, 1982.

PRENSKY (2001). Digital natives, digital immigrants. Disponível em: < https://docs.google.com/document/d/1XXFbstvPZIT6Bibw03JSsMmdDknwjNcTYm7j1a0noxY/edit. > Acesso em 12/11/2015.

VALENTE, José Armando. **Diferentes usos do computador na educação**. In: Computadores e conhecimento: repensando a educação. Campinas: NIED-Unicamp, 1993.

VALENTE, José Armando (org.). **O computador na sociedade do conhecimento**. 1.ed. Campinas: UNICAMP/NIED, 1999.

VALENTE, J. A., ALMEIDA, F. J. (1997**) Visão Analítica da Informática na Educação no Brasil:** A Questão da Formação do Professor. Disponível em: <http://www.lbd.dcc.ufmg.br/colecoes/rbie/1/1/004.pdf> Acesso em 24/09/2015.

RESNICK, M.; ROSENBAUM, E., 2013. Designing for Tinkerability. In Honey, M., & Kanter, D. (eds.), Design, Make, Play: Growing the Next Generation of STEM Innovators, pp. 163-181. Routledge.

RESNICK, M., 2007. All I Really Need to Know (About Creative Thinking) I Learned (By Studying How Children Learn) in Kindergarten. ACM Creativity & Cognition conference, Washington DC, June 2007.

RESNICK, M., KAFAI, Y., & MAEDA, J. (2003). ITR: A Networked, Media-Rich Programming Environment to Enhance Technological Fluency at After-School Centers in Economically Disadvantaged Communities: Proposal submitted to National Science Foundation.

RESNICK, Mitchel. **O computador como pincel**. In: VEJA. Limpeza de Alto Risco. Especial: um guia do mundo digital, São Paulo: Abril Cultural, n. 41, out. 2006.
RUSK, Natalie; RESNICK, Mitchel; MALONEY, John. **The report Learning for the 21st Century.** Lifelong Kindergarten Group MIT Media Laboratory, 2003.
SCRATCH. *Imagine, program, share*. Disponível em: < http://scratch.mit.edu >. Acesso em: 25/05/2012.