Jeff Bender | UNI: jrb2211 | 3/17/2015 | COMS E6123 Midterm Paper

Tooling Scratch
Designing a Collaborative Game-Based Learning System to
Infuse Computational Thinking within Grade 6-8 Curricula

ABSTRACT

At an unrivaled and enduring pace, computing has transformed the world. The resulting global economic circuitry demands from its operators more than reading, writing, and arithmetic: a fourth foundation, computational thinking. This new baseline need arises in an era of incommensurate education reform focused more on assessment than durable learning. Meanwhile, computing's proliferation has empowered students with versatile programming, enabling passion-driven discovery that engages the minds that schools too frequently fail to attract or satiate. Educators increasingly acknowledge that game-based learning activates student motivation and engagement, but schools infrequently equip teachers with the tools to harness games effectively. After introducing the challenges involved in infusing computational thinking in K-12 curricula, this article analyzes alternative initial learning environments in order to identify the platform, Scratch, best suited for students in grades 6-8. It then investigates how the learning sciences and self-determination theory, when unified along four axes, social addictive gameful engineering (SAGE), drive effective game-based learning. This theoretical grounding enables the sculpting of four learning principles which guide the formulation of design tactics fit for tooling Scratch with a collaborative game-based learning system that catalyzes teachers and students to immerse in computational thinking. The article concludes by proposing a design based research methodology that interweaves development with evaluation.

1. INTRODUCTION

"Computational thinking will influence everyone in every field of endeavor. This vision poses a new educational challenge for our society, especially for our children" [1].

1.1. Computational Thinking Primer

In 2006, writing as the head of the Computer Science Department at Carnegie Mellon University, Jeanette Wing articulated a call for educational action that releases her subject from the constraints of its scholarly seclusion. Stipulating that ubiquitous computing is to today as computational thinking (CT) is to tomorrow, she frames CT as a "universally applicable attitude and skill set everyone, not just computer scientists, would be eager to learn and use" [2]. Although a consensus definition for the term has remained elusive, CT aims to describe how humans, not computers, think logically, algorithmically, procedurally, analytically, recursively, and creatively, with an engineering-attunement [3]. This hybrid thinking helps us devise models and representations which enable problem solving, data transformation, and system automation [4]. The primacy of conceptualization, not programming, of ideas, not artifacts, frees computer science (CS) fundamentals from their constituent levees, allowing strong mental models to flood diverse disciplines both in the sciences and humanities, as well as in in their applied counterparts, medicine, law, and journalism [5]. Of course, computers retain their central role: "Abstractions are the 'mental' tools of computing. The power of our 'mental' tools is amplified by the power of our 'metal' tools. Computing is the automation of our abstractions" [1]. The emphasis on semantics rather than syntax, however, engenders a common language in which to explore the possibilities of computing, and a means by which to confront the dwindling wonderment experienced by digital natives commonly capable of using enveloping technologies, but devoid of digital fluency.

1.2. CT in K-12

Educational technology evangelist Marc Prensky coined this classification, digital native, in 2001, as means by which to emphasize the manner in which Internet-era students process information fundamentally differently from their predecessors [6]. Although accurately differentiating digital natives from their digital immigrant parents by identifying their desire for instant gratification and penchant for parallel processing, multi-tasking, randomly accessing information, hypertext thinking, and networking, he effectively limits his analysis to the extent to which this technologically savvy generation efficiently uses computers, video games, and the Internet [7], a set of competencies otherwise known as computer literacy [8]. Mitchell Resnick, leader of the Lifelong Kindergarten group at the MIT Media Lab, refines Prensky's discourse by focusing on digital fluency, which he argues "requires not just the ability to chat, browse, and interact but also the ability to design, create, and invent with new media" [9]. This spotlight on digital construction naturally elicits an emphasis on learning programming, a skill which substantially expands the range with which students can express themselves [10]. Digital construction thus becomes the setting for students to program while developing problem-solving and design strategies, such as modularization and iterative design, which interlock with non-programming domains.

The resulting interdisciplinary educational experience ultimately underwrites an expansive, yet inclusive, orientation around CT, extending beyond both computer literacy and digital fluency to encompass analytical skills which enhance how we approach problems and how we leverage computing to augment our abilities [11]. Such a sweeping mandate reinforces Wing's compelling vision for how CS educators, researchers, and practitioners can act to change society's sometimes narrow image of the field; instead of equating CS with computer programming, thereby isolating widely applicable conceptual competencies within the nooks of spare computer classes, we can expose the urgent need to foster a K-12 computational culture [12]. Existing curriculum standards, infrastructure deficiencies, and limited professional development opportunities for teachers to learn CT all present significant challenges [13], but implementing CT during school hours does not entail slicing a new Carnegie unit out of an already-eaten pie [14]. Programming, for instance, is reflexive with other domains; learning to code while exploring concepts from other subjects can ease the understanding of all involved topics concurrently [15, 16]. We can, therefore, among numerous other examples, demonstrate binary search when teaching how to find the roots of an equation, or introduce optimization when training students in Excel [12]. By consistently integrating CT where it fits in practice in established classroom curricula, we infuse a set of shared attitudes, values, goals, and practices which ultimately catalyze this computational culture.

Fortunately, in the last several years, many organizations have recognized the imperative of introducing CT to K-12 students, and progressive change has begun to impact schooling. The National Science Foundation (NSF) has established standards which formalize CT's core: problem formulation; logically organizing and analyzing data; representing data through abstractions such as models, automating solutions through algorithmic thinking; implementing effective solutions optimally; and transferring solutions to solve a wide variety of problems [17]. The NSF additionally bolsters its commitment through the CS10K Project, which aims to recruit ten thousand teachers prepared to teach CS in ten thousand schools by 2016. Considering that in 2012, only 2,000 teachers qualified to teach the Advanced Placement CS course, which less than 10% of U.S. high schools offer, the NSF has set aggressive goals [18]. Importantly, however, increasing the quantity of computer scientists remains only a tangential agenda of the broader CT movement [19]. The National Research Council has declared CT a cognitive skill which the "average person is expected to possess", and noted that "students can learn thinking strategies such as computational thinking as they study a discipline" already entrenched [3]. From this perspective, CT does not demand from school stakeholders protracted debate about wrangling time-slices for computer classes; instead, CT infuses within existing curricula [20]. In particular, the momentum of science, technology, engineering, and math (STEM) provides an apt opportunity: "There is inherently a C (Computing) in STEM. Learning STEM without learning computing is fundamentally inadequate. Learning computing while solving STEM problems, on the other hand, would inevitably foster one's computational thinking ability" [12]. Devising how to seize this opportunity adeptly emerges as a crucial arena for pragmatic field research in the formal and informal learning environments inhabited by K-12 students.

1.3. CT via Game-Based Learning

Conducting field studies before the CS community agrees on a definition for CT, however, reduces the pertinence of any findings. Some researchers trace CT to procedural thinking, first discussed in the literature in the 1960's [11] , and others argue CT has existed for centuries [21], but Wing's 2006 formulation essentially equates to a galvanizing, yet unwieldy, three-page definition. Several follow-up characterizations of CT attempt to inject varying degrees of pithiness:

- [D]ata collection, data analysis, data representation, problem decomposition, abstraction, algorithms and procedures, automation, parallelization, simulation [22].
- [D]efining, understanding, and solving problems, reasoning at multiple levels of abstraction, understanding and applying automation, and analyzing the appropriateness of the abstractions made [13].
- [D]ecomposition, pattern recognition, pattern generalization and abstraction, and algorithm design [23].
- [T]hinking to solve problems, automate systems, or transform data by constructing models and representations, concrete or abstract, to represent or to model the inner-working mechanism of what is being modeled or represented as an information process to be executed with appropriate computing agents. Such thinking is necessarily: logical; algorithmic; scientific; mathematical; analytical; engineering-oriented; creative [12].
- [A] problem-solving methodology that can interweave computer science with all disciplines, providing a distinctive means of analyzing and developing solutions to problems that can be solved computationally. With its focus on abstraction, automation, and analysis, computational thinking is a core element of the broader discipline of computer science [24].

These definitions, when uncoupled from context, unfortunately do little to advance Wing's visions for CT as an instrumental element of discovery in all disciplines, and as an integral part of childhood education. To meet the challenge of determining effective ways for children to learn CT, the abstract renderings must subsume within operational definitions that enable example-driven practical approaches to incorporating CT in classrooms. Since doing computation makes us more competent computational thinkers, precisely defining CT recedes as a secondary concern to the pragmatic promotion of computational doing in K-12 [12, 25].

This call for active learning indicates the use of game-based learning (GBL) as a pedagogical approach, because GBL most enticingly affords interactivity: a multi-way exchange with feedback and invitation to probe deeper [26]. CT's versatility as a glue that bridges domain-specific expertise and technical innovation further solicits GBL, since game content can support a broad diversity of subjects within existing curricula while providing students an opportunity to develop and apply CT synergistically [3]. The contextualized representations concretize CT's abstract concepts, resulting in both higher learning gains and increased engagement [27]. Furthermore, the multidisciplinary integration of CT helps students recognize appropriate use cases across domains, thereby instilling a more fluid kind of problem-solving than cultivated through siloed study [28]. By adaptively subdividing big problems into smaller ones solvable with the computational competence of each learner, games inculcate increased levels of self-efficacy. The associated elevated confidence inspires students to practice communicating with CT within enjoyable social environments designed to encourage both collaborative problem-solving and iterative group reflection that enables not only the digestion of new learning, but also the collective construction of new knowledge [29].

Several literature reviews of the increasing volume of GBL research in recent years demonstrate the effectiveness with which games improve general learning [30, 31, 32, 33, 34], increase motivation [35, 36, 37], and improve performance [38, 39, 40, 41], but close examinations of typical experimental designs reveal limited generalizability with results that apply only to the specific population under study [42]. The conclusions do not assert the efficacy of all games for all learners in all situations. The theoretical foundation from which these studies depart nonetheless has progressed enormously since video games' early years when edutainment gained prominence as "a sugarcoating of entertainment for the bitter medicine of education to become palatable" [43]. The interdisciplinary field of the learning sciences (LS), which studies teaching and learning, most significantly has influenced the structure and robustness of GBL research, but several psychology subdomains, such as self-determination theory (SDT), also imprint modern educational game design. These bedrocks provide sturdy support for research that braids learning theory with ludology, the study of games [44]. In order to increase the generalizability of experimental results, however, researchers must evolve from their orientation around coarse game instances and instead focus on granular game attributes that impact specific learning outcomes [45]. By determining the linkages between instructional objectives and game attributes, researchers can construct authoritative arguments based on empirical evidence, and accelerate the intervention needed to help lift schooling from obsolescence [46].

1.4. Goals & Structure

As visualized in Figure 1, the primary goal of this article is to employ theory from LS, SDT, and ludology to craft learning principles organized along four axes, social addictive gameful engineering (SAGE). These principles then inform design tactics for the pedagogically effective tooling of the visual programming environment, Scratch [47], herein identified as the best extant platform within which to construct a collaborative GBL system that infuses CT in grade 6-8 curricula. Secondary goals involve articulating motivation for the general public to learn CT, analyzing how GBL successfully immerses students in CT, and illuminating the need for deliberate research methodology that interweaves design-based research with rigorous experiments extracting empirical results.

The remainder of the article is structured as follows. Section 2 analyzes the needs for a CT platform for grades 6-8 and evaluates alternatives which incompletely suffice. Section 3 leverages the narrowed field to choose Scratch before examining its origins, design, curricular framework, and assessment capacity. Section 4 extracts from LS and SDT the theories most appropriate to enlace within GBL, and synthesizes these theories to craft four learning principles that guide the derivation of design tactics for tooling Scratch with a collaborative GBL system. Section 5 concludes the article by proposing an optimal set of research methodologies for actualizing SAGE.

2. EVALUATING COMPUTATIONAL THINKING PLATFORMS FOR GRADES 6-8

2.1. Needs Analysis

2.1.1. Students

At MIT in the late 1960s, learning theorist and technologist Seymour Papert helped develop the Logo programming language that introduced turtle graphics and an environment of play to a generation coming-of-age alongside personal computers. Extending the constructivist theory of knowing evangelized by developmental psychologist Jean Piaget, Papert introduced constructionism, asserting not only that learning entails an active process in which people construct knowledge from their experiences in the world, as in constructivism, but also that this process proves particularly effective during the construction of personally meaningful artifacts [48]. In contrast to the traditional vision of schooling known as instructionism, which revolves around a single speaker and many listeners who then need to take tests to prove they have listened adeptly [49], constructionism engenders a participatory culture in which learners build relationships between old and new knowledge during interactions with peers and mentors who provide formative feedback and contribute to the creation of socially relevant products suitable for summative assessments. Since CT both complements and combines mathematical and engineering thinking [2], two subjects rooted in the evolution of this learning theory, constructionism provides a valuable lens through which to analyze the needs for a platform supportive of learning CT in grades 6-8.

Students, the primary target, need low floors, high ceilings, and wide walls [9]. Low floors allow easy starts. High ceilings offer opportunities to construct increasingly complex projects over time. Wide walls support a diversity of project types so that students with varying interests and learning styles become engaged. A platform that satisfies these needs invites novices, elicits iterative refinement culminating in complex productions, and presents incrementally challenging experiences to maintain cognitive flow. Infrastructure to encourage use-modify-create activity cycles further helps students transition between experiential, exploratory, and experimental learning phases without departures to consult potentially distracting external resources [13]. These transitions also apply between problem domains and across time, thereby fostering the transfer of knowledge from one concrete instantiation to others, and to generalized abstract formulations over a continuum [50]. Within a rich, collaborative computational environment, students overcome any discomfort derived from isolated interaction with the computer; pair programming encourages imaginative programming and concept communication [28, 51], and larger group activities allow for coordinated, knowledge constructive jigsawing that motivates each student to integrate her own CT understandings and perspectives with those of her peers [52]. Importantly, however, the platform must not focus on programming at the expense of broader CT conceptualizations. Toy-like programming languages might provide short bursts of enthusiasm, but correct solutions to problems at lower code levels do not necessarily indicate conceptual traces to CT [53]. Students need a CT environment that encourages them to look beneath the hood, and that empowers them to develop a sense of agency as they innovate by imagining, playing, problem-solving, creating, sharing, and reflecting [13, 54].

2.1.2. Schools & Teachers

To ensure such a learning environment becomes available to all, not only to those families and communities privileged with purchasing power, public schools must intercede. As of 2010, however, 73% of states had adopted the ACM/CSTA standards for CS skills, 60% for capabilities, and 37% for concepts [55]. This focus on lower-level skills rather than deep concepts jeopardizes the feasibility of integrating CT, but the Common Core State Standards Initiative that currently sweeps the nation more directly counters the momentum of the CT movement. These data-driven standards depend upon assessments, leaving CS, which offers few assessments for K-12, excluded and at a disadvantage for program funding. As a consequence, underfunded schools infrequently can justify the allocation of resources for CT, and even if they do, the resulting classes must lure student time away from standards-aligned schedules. This confluence of circumstances indicates the need for educational policy change, but policy makers consistently conflate the use of technology and the teaching of computer literacy with teaching CS, or otherwise falsely assume frameworks for CT and CS exist within STEM curricula. As earlier intimated by the CT scope discussion, the CS community deserves some blame for the confusion over the swirl of terms related to technology, computing, and computer science. Although inherently a dynamic field shaped by rapid technological change, CS can contribute more cogent definitions, economical authoring tools, efficient means of assessment, and relevant, age-appropriate pedagogical examples [56, 25]. Schools need these resources in order to pervasively plug-in CT within curricula at all levels [12], and to establish professional development regimes for their teachers [13].

The need for teacher professional development arises not only out of an effort to keep pace with the rate of technological change associated with CS, but also because state requirements for CS teacher certification remain disorganized, deeply flawed, and inconsistent in their formulation of CS [55]. Although some new CS teachers have received training, many more hold certifications in another area or arrive from industry with a CS background but no teaching experience. Given this deficit in human capital and the aforementioned synergistic benefits of learning CT while studying another discipline, as advocated by the National Research Council, the goal of infusing CT within existing curricula, rather than inserting a new CS class, fits best [3]. A broad set of teachers with diverse content knowledge, therefore, needs tools that help develop effective pedagogies which incorporate CT but do not prove burdensome to integrate with existing, time-constrained classroom practices. The CS community can contribute sensible orderings of concepts that sculpt developmentally appropriate CT learning progressions, demonstrate how these concepts operate across subjects, and influence the structure of systemic and sustainable CT platforms which embed ecosystems of accessible training materials [57, 58]. Teachers need these resources to become empowered computational thinkers capable of designing learning with CT [11].

2.1.3. Nurturing Affinity Spaces

As complements to the contributions from CS, teachers additionally need nurturing affinity spaces [59]. These communities offer low barriers to expression and civic engagement, robust facility for creation and sharing creations, support for mentoring relationships, the transfer of knowledge from experts to novices, and generally support good learning and human growth. As social networks revolving around production, they organize people across time and space by interest and passion. Since they enable easy information storage and retrieval, the production of content becomes distributed across people and tools; as a result, not only do participators develop a sense of belonging, but they also come to see themselves as "insiders" in a realm of shared meaning. Teachers need such virtual spaces to support their integration of CT within curricula because few like-minded peers might live locally, and those who do might not teach the same subject. Having an existing advocate outside of CS illustrate how to incorporate CT into one's discipline proves crucial to propagating CT momentum [20]. Meanwhile, students, too, need nurturing affinity spaces of their own so that they can support one another inclusively and connect to CT in personally meaningful manners. Naturally, not all students develop passion for CT, but in these spaces they respect the passion around which interactions unfold, and leverage it for help when needed while still perceiving proactive trial-and-error as a reliable path to success. Expertise thus becomes rooted in the space rather than the individual, and knowledge becomes a tool used to create and solve problems instead of a disconnected measure of student achievement.

2.2. Alternatives

2.2.1. Karel & Cousins

Choosing a CT platform for grades 6-8 involves identifying a set of environments supportive of novices and evaluating how well each candidate addresses the needs earlier analyzed. Since the relatively nascent CT movement post-dates the design and development of many available options, the process sometimes entails retrofitting systems with alternative primary targets. Karel the Robot (Fig. 2), for example, first implemented in the 1980s, focused more narrowly on teaching programming in a motivating context [60]. Integrating a development environment with a simple simulation consisting of avenues, streets, walls, beepers, and robots, Karel offers visual appeal and anthropomorphism that lends life to abstractions. Researchers responsible for the development of more recent incarnations, such as Karel the Robot in Java, report that these features enable simple presentation of problems through initial and final pictures, accurate visual feedback on correctness of an algorithm, and easy error detection during animation rendering [61]. These promising results have inspired a variety of off-shoots involving novices controlling robots, including two targeting CT: Light-Bot (Fig. 3) and an Interaction-Feedback prototype (Fig. 4). Unlike Karel's Java cousin, which divorces the development environment from the simulator in order to remove the need to switch to a more sophisticated environment mid-course, Light-Bot drops the syntax of a specific language altogether. Instead, it leverages programming semantics to direct focus toward problem-solving within its CT framework comprised of processes and transformations, models and abstractions, patterns and algorithms, and tools and resources [5]. The researchers acknowledge concerns about transference similar to those expressed by the developers of Karel, however; the link between the robot's movement and CT concepts might not consistently reach students. The Interaction-Feedback prototype aims to address this problem by including an "Equivalent Programming Logic" view alongside the simulation and semantic programming interface [53]. In addition to mirroring the semantic interaction with real syntax, this view provides user-friendly debugging feedback during compilation in a manner similar to related work targeting runtime errors in Java [62]. The integration of game attributes, such as clear goals, scores based on efficient and well-balanced solutions, and achievements, aim to further increase student motivation to learn CT. Karel and cousins, however, offer only low floors, not high ceilings or wide walls. While encouraging imagining and playfulness, they become too toy-like, devoid of simple authoring tools. When associated with related Karel textbooks, they provide learning progressions, but do not embed training materials, nor interconnect to nurturing affinity spaces. Grades 6-8 need more.

2.2.2. StarLogo TNG

More open-ended Initial Learning Environments (ILE), such as StarLogo TNG (Fig. 5), focus less directly on providing a motivating context, and instead aim to reduce the friction novices encounter when attempting to write their first programs [60]. Syntactical struggles with command names, parentheses, the order of parameters, and similar language-level details needlessly distract from the goal of learning CT. By encasing these minutiae within graphical objects, visual ILEs help novices recognize command names and shapes so that they can construct a wide variety of programs as they learn design strategies for solving a diverse set of problems. In this sense, StarLogo TNG offers considerably wide walls, because the colorful palette of interlocking programming constructs invites a dexterous range of expressivity. The environment's target of teaching science through the study of decentralized systems, however, has led to the time-consuming creation of pedagogical examples related to only a small slice of the typical middle school curriculum [63]. These structured classroom worksheets prove crucial in helping teachers guide students to implement sound experimental design and data collection within the 3D simulation environment. Without this guidance, the wide walls within science too easily allow students to wander away from desired learning outcomes, and the high floors associated with the complexity of highly-populated 3D representations sometimes discourage engagement with the programming portions within design-build-test cycles. As a result, some students engage only when running a simulation or playing a game, and flounder during the development cycle. Structured lesson plans help keep all students aligned, but they require significant investments in time, both in preparation and implementation, and since no nurturing affinity space exists, these investments rarely amortize across the community. For the already overburdened middle school teacher, then, StarLogo TNG does not offer a practical platform for incorporating CT.

2.2.3. Alice

Other visual programming environments, such as Alice (Fig. 6), demand less from each individual teacher while consistently producing increases in academic achievement compared to traditional instructional methods [64]. Alice's robust nurturing affinity space, replete with lesson plans and a supportive community, substantially simplifies the process of integrating CT within existing curricula [65]. The rich and flexible 3D environment, representative of wide walls, supports a variety of subjects inclusive of both STEM and the social sciences [66]. Furthermore, the carefully evolved naming of programmatic constructs helps reduce confusion commonly associated with manipulating objects in 3D [67]. For example, keywords such as *Resize*, *Move*, *Speed*, and *AsSeenBy* replace the more problematic originals, *Scale*, *Translate*, *Rate*, and *CoordSys* [68]. These refinements effectively enable novices to program in 3D and learn CT by creating animated stories. Unfortunately, however, few studies have demonstrated successful outcomes at the middle school level, and the overwhelming majority of the Alice community targets undergraduate CS students [69]. Some features, such as the capability to view full Java syntax while visual programming, allow Alice to lure students deeper into CS classes and careers [70], but do little to lower the floors for middle school students. Without shape differentiation, the visual programming constructs often overtake the complex, dropdown-entangled interface, thereby occluding the 3D world, and demotivating struggling students. While a worthy platform for undergraduates, Alice does not suffice for grades 6-8.

3. SCRATCH

3.1. Origins & Design

The CT platform that best satisfies the needs for these grades is Scratch (Fig. 7), the web-based visual programming environment designed not to prepare children for careers as professional programmers or computer scientists, but to empower them to program as a means of creative expression [9]. First developed in 2007 as a desktop application, Scratch targeted underprivileged youth in the Computer Clubhouse network, a globally dispersed collection of community technology centers offering informal learning environments in which children engage in constructionism as they demonstrate autonomy by actively exploring their own interests during cycles of creation and by solving relevant, authentic problems [71]. Scratch's media-centric environment nicely supports this approach to learning since it eliminates the barriers between pop culture and high-status knowledge, such as software design and development [72]. "Scratchers" assume the roles of artist, performer, audio engineer, director, and more as they remix each other's projects and collaboratively learn as peers within a design culture facilitated by mentors rather than teachers. This structure alters the dynamic of learning from "push" to "pull", increasing motivation, deep understandings, and rich conceptual connections. Self-directed learning during creative design spirals of imagining, creating, experimenting, sharing, and reflecting, which leads to fresh imagining, enables children to become comfortable expressing their ideas programmatically, sometimes even before learning to read or write with the same proficiency [54, 72].

Scratch develops this comfort in computational authoring by offering especially low floors and wide walls, in addition to sufficiently high ceilings. Compared to several precursor novice environments, Scratch more successfully overcomes typical challenges associated with integrating programming into curricula, such as difficult syntax, uninteresting activities, and meager instructive guidance [9]. The Lego-like interlocking shapes of the visual programming constructs eliminate syntax errors while providing vivid feedback encouraging effective arrangements, allowing learners to focus on semantics [73]. The simultaneously visible script area and stage, the scripts that highlight as the action unfolds, and the active in-palette blocks similarly simplify the mechanics of programming while inviting experimentation. The latest web-based version of Scratch, first released in 2013, extends this invitation to tinker to the online social domain via the Scratch website, where students remix, explore, and "love" one another's animations, stories, games, and more as they discuss their creations and plans in highly active forums [47]. Collectively, these projects, discussions, and social bonds, centered on personally meaningful drawings, audio recordings, video, and programming logic, become "the YouTube of interactive media" [9]. This alluring nurturing affinity space even has a teacher-centric twin, ScratchEd, where educators gather to share age-appropriate pedagogical examples, discuss learning progressions, and develop accessible training materials [74]. Scratch, then, meets the needs of students, schools, and teachers, and emerges as the best platform for infusing CT into grades 6-8.

3.2. CT Curricular Framework

To guide this infusion, Scratch researchers offer a CT curricular framework centered on the creation of computational artifacts [75]. This constructionist approach supports the development of computational concepts, practices, and perspectives across subjects and contexts while emphasizing designing, collaborating, personalizing, and reflecting [76]. Computational concepts include iteration, parallelism, events, conditionals, variables, and lists. Computational practices, such as debugging, reusing, and modularizing, develop alongside these concepts as learners form computational perspectives about themselves and the world around them, leading to increased expression, connection, and questioning. By encouraging a culture of fearlessness, exploration, and peer collaboration, this model allows teachers to develop computing expertise on an as-needed basis as the CT learning progression unfolds. Although timely teacher-led instruction initiates the discovery and integration of new concepts, role-switching often elicits the deepest absorption of practices and perspectives, turning the student into teacher of peers, or teacher of teachers [71]. This distribution of expertise further manifests in examples offered by millions of projects available for remixing in the associated nurturing affinity spaces, which effectively present a canonical literature for Scratch [9]. The freedom to design, create, and explore CT, however, does not absolve students or teachers from the responsibility to measure and evaluate learning outcomes. To support efficient assessment, Scratch requires additional tooling.

3.3. Assessment Tools

Although CT curricular integration implicitly demands accompanying assessment, several researchers have proposed more traditional means than platform-embedded tools. Since 2010, when the NSF highlighted the lack of extant CT learning instruments, many apparatuses have surfaced, such as the CT Score, a 4-point Likert evaluation, the CT Pattern Quiz, which measures transfer, and the CT Problem Solving Inventory, which quantifies usage of computational strategies during problem solving [5]. These test-like mechanisms, while perhaps satisfactory as complements to instructionist teaching, disrupt constructionism's creative flow within authentic activities. Furthermore, they assume skill-sets independent from CT that do not apply uniformly. For grade 6-8 students "who may not have the same reading ability, who may have fragmentary schema or emerging skill levels, who may not be able to hold the key components of the question and the answers in their working memory, a standardized item may... be meaningless, a salad of words and ideas that have no discernable connection to what they thought they were learning" [77]. To isolate CT learning, Scratch tools, such as Scrape (Fig. 8) and Scratch Analyzer (Fig. 9), assimilate the data exhausted during student-Scratch interaction, and synthesize automated formative and summative feedback [78, 79]. Scrape identifies blocks used within Scratch projects and presents a concise visualization useful during longitudinal project portfolio analyses. Scratch Analyzer further distils these projects into inputs fit for educational data mining and learning analytics capable of surfacing similarities between projects and peers, fueling recommendation engines which adaptively direct learning, and clustering and classifying students during automated evaluations. This automation frees teachers from time-consuming artifact-based interviews and students from design scenarios that disconnect assessment from interests and abilities. Once combined within an engaging GBL system, these tools enable Scratch to transform assessment from a caustic chore into an embedded, enjoyable game mechanic. We examine the theory, learning principles, and design tactics which inform the construction of the GBL system next.

4. GBL THEORY, LEARNING PRINCIPLES, & DESIGN TACTICS

4.1. SAGE Motivation

A prerequisite of this examination is the identification of the users of the GBL system. Three key roles, game player, game designer, and game system designer, require differentiation:

> Game players inhabit that wonderland space where the frame of the game intersects the frame of the real world. Game designers have the supreme pleasure of creating their own rabbit holes, hoping players find their way down inside, in order to create their own meanings. Game designers are the architects, the meaning-makers, the storytellers that make the play of wonderland possible... Game system design is a kind of metagame design. A game system designer designs the structure within which other game designers will create games. The "rules" of the system are the physical qualities of the game system components; the "play" that takes place is game design itself, resulting in sets of rules that make use of the game system [80].

Constructing SAGE within Scratch therefore involves designing the structures that teachers, in the role of game designers, will use to construct games for students to play as they learn CT. Since the majority of similar research focuses on learning by implementing games rather than through game play [81, 82], the system should not preclude advanced students from designing games for beginner and intermediate peers, but SAGE's primary aim positions teachers as game designers for students as game players. As discussed in the introduction, GBL's interactivity offers close pedagogical fit for computational doing; the concretized contextualization of CT abstractions across subjects increases learning gains, as well as motivation [15]. Instilling intrinsic motivation for CT in grades 6-8 emerges as a crucial goal because during the middle school years, students often reach conclusions about their own skills and aptitudes, and as a result, forsake many STEM subjects [83, 56]. Since over 99% of boys and 94% of girls age 12-17 play videogames in their own free time [78], game play presents an enticing means to motivate interest in CT. Recent research that inspires SAGE gamifies the development environment Eclipse, and shows promise in helping undergraduate students learn software design and testing [84]. But educational games, including those intended to teach aspects of CS currently recognized as CT, have existed for decades, and have produced inconsistent results with limited widespread impact [85, 86]. Most educational game studies do not employ the lens of LS to isolate specific game attributes that lead to specific learning outcomes, and thus frequently suffer from lack of generalizability, even when demonstrating a particular game enhances learning for a particular convenience sample population [44, 45]. By tooling Scratch with SAGE, a collaborative GBL system, we can provide the instrumentation necessary to trace CT learning outcomes to game attributes, and subsequently develop embedded guidance for effective game design techniques supportive of each computational concept, practice, and perspective along age-appropriate CT learning progressions. To inform this tooling, the following subsections, organized by SAGE's axes, first extract from LS and SDT the theories most appropriate to enlace within GBL. Syntheses of these theories then enable the crafting of four learning principles that guide the derivation of design tactics for tooling Scratch with SAGE. Forthcoming deeper examinations in future work should further ground SAGE, but the short format presented here (and consolidated in Table 1) suffices for surfacing the theories, learning principles, and design tactics required for initiating near-term development.

4.2. Social

Social learning aligns with the LS socio-cultural perspective in which individuals collectively construct knowledge though joint activities as they participate in cultural practices that shape cognition [87]. From this viewpoint, "[d]esigning learning environments is not merely a matter of getting the curricular material right but is crucially also a matter of getting the situated, emergent community structures and practices 'right'" [88]. Social interaction within the learning environment thus emerges not only as a motivator for engaging with educational content, but also as a vehicle in which to enact collaborative skills consistent with CT practices [89]. Since modern problem-solving occurs in communities rather than in isolation, developing social agility while encountering CT concepts fosters a positive attitude to "failure" as learning unfolds, and encourages not only experimentation and collective construction of computational models, but also a culture of model critique that questions underlying assumptions and identifies pitfalls [90, 3].

The theories of social gameflow, computer supported collaborative learning (CSCL), and cognitive apprenticeship might best support social learning in a GBL system. Social gameflow involves instilling in players perceptions of interdependent goals and rewards, but also hinges upon a conception of games inseparable from contexts in which social relations and surrounding environments necessarily impact in-game and transferable learning [91]. CSCL emphasizes that learning occurs socially during shared knowledge construction and group cognition when reinforced by adaptive media which prompts, analyzes, and selectively responds [92]. Cognitive apprenticeship leverages more knowledgeable others or agents to illustrate the power of particular techniques applied in diverse settings, thereby bringing cognitive process into the open for observation and adaptive practice in which task complexity modulates so that component skills optimally integrate [93, 94]. Combined, these theories influence the crafting of SAGE's social learning principle, which in turn leads to the derivation of three social design tactics:

Nurturing Affinity Space Principle
SAGE affiliates students by enveloping them in shared endeavors that require cross-functional collaboration, thereby creating multiplicities of respected expertise.

Social Design Tactics
- Role provisioning
- Peer-visible explanation facilitation (elected and prompted)
- Script sharing and embedded chat (elected and event-driven)

The nurturing affinity space principle amalgamates SAGE's social theories and establishes a foundation for their application by advocating cross-functional collaboration. The design tactic of role provisioning enables the realization of this student interdependency, as role play entails active thinking about actions, decisions, and choices from diverse perspectives, while simultaneously encouraging interpretation, synthesis, and evaluation of the work of others [37, 95]. Peer-visible explanation facilitation further fosters this peer-review, but also results in 20-44% learning gains for the explainer [96]. Lastly, elected and prompted script sharing with embedded chat reduces collaborative friction and knits the learning process with cultural cohesion.

4.3. Addictive

SAGE's addictive learning does not disrupt this cohesion in any medically pathological manner; in the game industry, the term refers to engaged, repeated play, and harmonious rather than obsessive passion [80]. In this context, addicting students to learning is a desirable goal. The theories of flow, interest, motivation, and engagement might best support addictive learning in a GBL system. Characteristics of flow include: a clear sense of required action; intense concentration; continuous feedback; a sense of control; a capacity to act to overcome challenges; an accelerated distortion of time; and a sense of intrinsic reward [97]. Flow enables students to practice CT with the attentiveness required to develop deep understandings, but studies indicate that fully immersive flow might inhibit metacognition, and therefore immersive flow provides better fit when aiming to proceduralize knowledge than when trying to acquire or reflect on it [98]. Furthermore, since individuals cannot infinitely sustain flow, and students can meet many educational CT goals without achieving such a demanding psychological state, addictive learning must extend beyond flow, and capture longer-term student interest, motivation, and engagement [99]. Interest describes a predisposition to reengage with particular disciplinary content over time. A complex blend of environment, cognition, and affect compose motivation. Engagement measures a learner's connection to the socio-emotional and cognitive aspects of the learning environment as evidenced by initiation of action, effort, and persistence in academic tasks. When unified, these theories empower games to addict students to learning without necessarily presenting high-fidelity graphics and soundscapes; production values need not match commercial quality to challenge students adaptively to the cusp of their capabilities [100]. SAGE's addictive learning principle and derived design tactics directly advance and apply these theories:

Regime of Competence Principle
SAGE induces flow and cultivates engagement by presenting pleasantly frustrating challenges that stretch the competencies of each individual student.

Addictive Design Tactics
- Game mechanic blocks with adaptive constraints
- Configurable block point system integrated with game mechanic blocks and levels
- Achievement system

The regime of competence principle encourages the use of flow as a means to capture student interest and propel that interest into sustainable motivation and engagement. The design tactic of game mechanic blocks, when integrated with a configurable block point system, introduces flexible constraints to SAGE that optimally match student abilities to skills required to accomplish goals, thereby creating and sustaining motivational tension [101]. These new blocks constrict students so that they do not attempt to manipulate palettes within Scratch randomly; instead, students evolve from novices to experts through structured play as they focus on specific CT learning outcomes [37]. The associated leveling tooling, in combination with the achievement system, offers means to provide consistent corrective feedback and real-time assessment [102]. When embedded endogenously within game contexts which promote mastery rather than performance, achievements also encourage goal setting, goal commitment, strategy formulation, and self-assessment, all of which enhance self-efficacy and improve learning outcomes [37].

4.4. Gameful

Similarly, when CT concepts, practices, and perspectives meld within game contexts, the learning becomes gameful: goal-oriented, curiosity-driven, failure-fearless, optimistic, and fun [103]. Instead of overlaying the game cycle, CT content blends throughout the environment as an intrinsic component of game-play [53]. The theories of experiential learning, learning in activity, embodiment, scaffolding, goal-based scenarios, and story-centered curricula might best support gameful learning. Experiential learning theory emphasizes experiencing problem-solving strategies prior to deliberate cognition and reflective behavior during which abstract concepts subsequently develop [104]. Learning in activity affords groups opportunities to work with authentic objects and technological activity systems that enable close inspections before expanding toward broader goals and generalizations [105]. Embodiment equivalently guides learners from immersive action to structured reflection while making unconscious, tacit knowledge explicit [106]. Scaffolding provides such structure by embedding guidance in context so that learners can perform expert-like tasks and develop understandings of a discipline's objectives and the relationship between its objectives and procedures [107]. Goal-based scenarios compartmentalize scaffolding within school-sized slices of class-time by establishing hierarchies of accomplishable missions which elicit intrinsic motivation [108]. These hierarchies combine to form a story-centered curriculum, "a carefully designed apprenticeship-style learning experience in which the student encounters a planned sequence of real-world situations constructed to motivate the development and application of knowledge and skills in an integrated fashion" [109]. Purposeful narratives, therefore, provide the meaningful and motivating roles for student as they face adaptive progressions of challenges that stretch their abilities. SAGE's gameful learning principle and design tactics aim to fuse these theories by emphasizing endogenous learning within story-centered game contexts:

<u>Situated Meaning Principle</u>
SAGE does not assemble CT concepts disconnected from gameplay, but instead envelopes students in gameful environments in which concept exploration emerges as narratives unfold.

<u>Gameful Design Tactics</u>
Integrated storyboard and leveling controls
Palette/block restriction system supportive of narrative structure
Block-crippling mechanism facilitating endogenous stories

The situated meaning principle distills the gameful theories into a clear directive to embed learning endogenously, thereby precluding students from spending time learning game mechanics without overtly learning CT [110]. The design tactic of integrating a storyboard and incorporating leveling controls within Scratch enables the construction of motivating stories containing goal-based scenarios that scaffold learning using game levels in order to drive experiential, embodied, learning in activity. Reinforced by a palette/block restriction system, these stories not only motivate, but also offer metaphorical loft as they dually establish immediate goal-based scenarios and more expansive connections to possible real-world circumstances [111, 112]. The block-crippling mechanism further facilitates these connections by situating students as collaborative or competitive debuggers, and therefore as crucial characters who deeply internalize learning as narratives unfold [95, 37]. Together, these gameful design tactics enable the construction of complex stories with multiple routes to completion that appropriately sequence CT activities in which students enjoy considering options, negotiating compromises, making decisions, solving problems, and thinking computationally.

4.5. Engineering

Exploring multiple routes while solving problems engenders engineering habits of mind and mental processes that require important aspects of CT [28]. This engineering problem-solving becomes fun in GBL since, after all, a game is "a problem-solving activity, approached with a playful attitude" [113]. The theories that might best support an engineering approach when learning CT include problem-based learning (PBL), and a sub-theory of SDT, cognitive evaluation theory (CET). PBL situates students in real-world contexts so that they can construct principled arguments while collaborating to solve complex, ill-structure problems [114]. This method enhances students' capacities to transfer knowledge to new problems and achieve more coherent understandings than traditional instruction because it consistently activates prior knowledge and demands cumulative reasoning, theory building, and conceptual change. CET promotes self-motivated and self-determined learning by highlighting the first two psychological nourishments of SDT's triad, competence, autonomy, and relatedness [115]. Equifinality, manifested in in the form of multiple routes, offers students autonomy over their own actions as they achieve tasks within game contexts, resulting in experiences of competence and the sense of self-efficacy crucial for youth momentum in engineering disciplines [116, 117, 56]. SAGE's engineering learning principle and design tactics advance these theories in the GBL context:

Explicit Information On-Demand and Just-In-Time Principle
SAGE explains directly only when students eagerly seek or desire information and have accrued the experience and motivation imperative to consume that information lucidly.

Engineering Design Tactics

- Context-aware, event-driven, just-in-time CT knowledge system
- Embedded, on-demand, multi-sourced instruction modules
- Support for diagnostics and performance measurements

The explicit information on-demand and just-in-time (JIT) principle leverages timely delivery of information to support an engineering approach to learning CT. As GBL scholar James Paul Gee notes, this immersion-first, knowledge-second approach drives robust learning: "When you have played a video game for a while, something magical happens to the texts associated with it. All of a sudden they seem lucid and clear and readable" [118]. The design tactic of a context-aware CT knowledge system preserves mystery to enhance learning by sparking engaging sensations of novelty, surprise, and cognitive curiosity [119]. Motivated to fill in the gaps between the known and unknown, students exert their autonomy to influence elements of the learning environment that trigger role-targeted, event-driven information that arrives JIT to establish student-centered, personalized learning progressions [120, 121, 122]. The on-demand instruction modules offer tactics for future improvement that reinforce self-efficacy and empower students to stretch their perceptions of competence as they hone complex strategies to overcome adaptive, ill-structured problems [123, 124]. When multi-sourced, the embedded instruction provides opportunities for students to synthesize knowledge with sufficient structure to improve subsequent, higher-order problem-solving activity [37]. Lastly, the support for diagnostics and performance measurements emphasizes the importance of collecting evidence as a means to map cause-and-effect CT relationships when immediate feedback proves insufficiently conclusive [125]. Combined, these engineering design tactics support the development of CT practices and perspectives that expose gaps in student knowledge and motivate self-directed remediation.

5. Conclusion

> "Computers have enormous impact on the way we live, think, and act. It is hard to overestimate their importance in the future. In fact, many believe that the true computer revolution will not happen until everyone can understand the technology well enough to use it in truly innovative ways" [24].

5.1. Research Methodology & Future Work

The CSTA's full-throated advocacy for CT, when considered alongside studies concluding that the development of CT occurs over a continuum, and thus should begin in childhood [13], howls for a creative intervention that reduces the rarity of rigorous CT in K-12 education. SAGE, herein examined as a collaborative GBL system which infuses CT into grade 6-8 curricula, offers the start of a solution within Scratch, a CT platform best satisfying the needs of students, teachers, and schools, and already gaining traction in classrooms [74]. By conducting design-based research [126], leveraging the technique of scale-down [127], and employing exploratory sequential mixed methods [128], researchers can experiment with SAGE to develop a systemic perspective while advancing new theory and practice. This iterative, cohesive methodology uses qualitative phases to inform analyses and inform the construction of instruments used in follow-up quantitative phases which derive empirical evidence and generalizable results. These results, reinforced by grounded LS and SDT theory, crafted learning principles, and purposeful GBL design tactics, can become guidelines for cultivating a collaborative ecosystem of game-type innovation which couples specific game attributes to CT learning outcomes. As GBL scholar Kurt Squire suggests, "[w]e shouldn't 'teacher-proof' curricula so that it can be tested; instead, we should create compelling materials that address teachers' needs and inspire them to teach creatively and effectively" [129]. SAGE within Scratch aspires to offer this inspiration to teachers and students so that widespread CT understandings spark the pervasive innovation of the true computer revolution.

REFERENCES

[1] J. Wing. "Computational thinking and thinking about computing," *Philosophical Transactions: Mathematical, Physical and Engineering Sciences*, vol. 366, no. 1881, 2008.

[2] J. Wing. "Computational Thinking," *Communications of the ACM*, vol. 49, no. 3, pp. 33-35, 2006.

[3] Committee for the Workshops on Computational Thinking, "Report of a workshop on the scope and nature of computational thinking," National Academies Press, Washington, D.C., 2010.

[4] Computer Science Teachers Association (CSTA). (2011) Operational definition of computational thinking for K-12 education. [Online]. http://csta.acm.org/Curriculum/sub/CurrFiles/CompThinkingFlyer.pdf

[5] L. A. Gouws, K. Bradshaw, and P. Wentworth, "Computational thinking in educational activities: an evaluation of the educational game light-bot," in *Proceedings of the 18th ACM conference on Innovation and technology in computer science education*, 2013.

[6] M. Prensky. "Digital natives, digital immigrants part 1," *On the Horizon*, vol. 9, no. 5, pp. 1-6, 2001.

[7] M. Presnky. *Don't Bother Me, Mom, I'm Learning!: How Computer and Video Games are Preparing Your Kids for 21st Century Success and How You Can Help!* Paragon House, 2006.

[8] A. A DiSessa. *Changing minds: Computers, Learning, and Literacy*. MIT Press, 2000.

[9] M. Resnick et al. "Scratch: programming for all," *Communications of the ACM*, vol. 52, no. 11, pp. 60-67, 2009.

[10] M. Resnick and B. Silverman, "Some reflections on designing construction kits for kids," in *Proceedings of International Conference for Interaction Design and Children*, 2005.

[11] A. E. Weinberg, "Computational thinking: an investigation of the existing scholarship and research," Colorado State University, Ph. D. dissertation, 2013.

[12] C. Hu, "Computational thinking – what it might mean and what we might do about it," in *Proceedings of the 16th annual joint conference on Innovation and technology in computer science education*, 2011.

[13] I. Lee et al. "Computational thinking for youth in practice," *ACM Inroads*, vol. 2, no. 1, pp. 32-37, 2011.

[14] W. Bainbridge. "Carnegie Unit," in *Encyclopedia of Educational Reform and Dissent*, T. Hunt et al., Eds. SAGE Publications, Inc., 2010, pp. 136-138.

[15] P. Sengupta, J. Kinnebrew, S. Basu, G. Biswas, and D. Clark. "Integrating computational thinking with k-12 science education using agent-based computation: a theoretical framework," *Educational Information Technology*, vol. 18, no. 2, pp. 351-380, 2013.

[16] U. Wilensky and M. J. Jacobson. "Complex systems and the learning sciences," in *The Cambridge Handbook of the Learning Sciences*, K. Sawyer, Ed. Cambridge University Press, 2014.

[17] A. Basawapatna, K. Koh, A. Repenning, D. Webb, and K. Marshal, "Recognizing computational thinking patterns," in *Proceedings of ACM Special Interest Group on Computer Science Education*, 2011.

[18] A. Yadav and J. Korb. "Learning to teach computer science: the need for a methods course," *Communications of the ACM*, vol. 55, no. 11, 2012.

[19] A. Yadav, N. Zhou, C. Mayfield, S. Hambrusch, and J. Korb, "Introducing computational thinking in education courses," in *Proceedings of the 42nd ACM technical symposium on Computer science education*, 2011.

[20] C. Dierbach et al., "A model for piloting pathways for computational thinking in a general education curriculum," in *Proceedings of the 42nd ACM technical symposium on Computer science education.*, 2011.

[21] S. Salinger, L. Plonka, and L. Prechelt. "A coding scheme development methodology using grounded theory for qualitative analysis of pair programming," *Human Technology: An Interdisciplinary Journal*

on Humans in ICT Environments, vol. 4, no. 1, pp. 9-25, 2008.

[22] H. Bort and D. Brylow, "CS4Impact: Measuring Computational Thinking Concepts," in *Proceeding of the 44th ACM technical symposium on Computer science education*, 2013, pp. 427-432.

[23] Google. (2015) Exploring Computational Thinking. [Online]. https://www.google.com/edu/resources/programs/exploring-computational-thinking/

[24] The CSTA Standards Task Force, "K–12 Computer Science Standards," Computer Science Teachers Association, 2011.

[25] V. Barr and C. Stephenson. "Bringing computational thinking to K-12: what is involved and what is the role of the computer science education community," *ACM Inroads*, vol. 2, no. 1, pp. 48-54, 2011.

[26] C. Crawford. *The Art of Computer Game Design*. Amazon Digital Services, 1984.

[27] S. Hambrusch, C. Hoffmann, J. T. Korb, M. Haugan, and A. L. Hosking, "A multidisciplinary approach towards computational thinking for science majors," in *Proceedings of the 40th ACM Technical Symposium on Computer Science Education*, 2009, pp. 183-187.

[28] Committee for the Workshops on Computational Thinking, "Report of a workshop on the pedagogical aspects of computational thinking," National Academies Press, Washington, D.C., 2010.

[29] M. Prensky. "Digital natives, digital immigrants part 2: do they really think differently?," *On the Horizon*, vol. 9, no. 6, pp. 2-6, 2001.

[30] S. De Freitas, "Learning in immersive worlds," Joint Information Systems Committee, London, 2006.

[31] M. Bober, "Games-Based Experiences for Learning.," Futurelab, Bristol, 2010.

[32] M. Ulicsak and M. Wright, "Games in education: serious games," Futurelab, Bristol, 2010.

[33] M. Ulicsak and B. Williamson, "Computer games and learning: a handbook," Futurelab, London, 2011.

[34] D. Clark, E. Tanner-Smith, S. Killingsworth, and S. Bellamy, "Digital games for learning: a systematic review and meta-analysis (executive summary)," SRI International, Menlo Park, 2013.

[35] J. M. Randel, B. A. Morris, C. D. Wetzle, and B. V. Whitehead. "The effectiveness of games for educational purposes: a review of recent research," *Simulation and Gaming*, vol. 23, pp. 261-276, 1992.

[36] J. J. Vogel et al. "Computer gaming and interactive simulations for learning: a meta-analysis," *Journal of Educational Computing Research*, vol. 34, no. 3, pp. 229-243, 2006.

[37] K. A. Kapp. *The Gamification of Learning and Instruction: Game-Based Methods and Strategies for Training and Education*. Pfeiffer, 2012.

[38] R. T. Hays, "The effectiveness of instructional games: A literature review and discussion (Technical Report 2005-004)," Naval Air Warfare Center, Orlando, FL, 2005.

[39] F. Ke. "A qualitative meta-analysis of computer games as learning tools," *Handbook of research on effective electronic gaming in education*, vol. 1, pp. 1-32, 2009.

[40] K. Becker. "Distinctions between games and learning: a review of current literature on games in education," in *Gaming and Cognition: Theories and Practice from the Learning Sciences: Theories and Practice from the Learning Sciences*, R. Van Eck, Ed. IGI Global, 2010.

[41] T. Sitzmann. "A meta-analytic examination of the instructional effectiveness of computer-based simulation games," *Personnel Psychology*, vol. 64, no. 2, pp. 489-528, 2011.

[42] R. Van Eck. "Forward," in *Gaming and Cognition: Theories and Practice from the Learning Sciences: Theories and Practice from the Learning Sciences*, R. Van Eck, Ed. IGI Global, 2010.

[43] M. Resnick. "Computer as paintbrush: technology, play, and the creative society.," in *Play= Learning: How Play Motivates and Enhances Children's Cognitive and Social-Emotional Growth*, D. G. Singer, R. M. Golinkoff, and K. Hirsh-Pasek, Eds. Oxford University Press, 2006.

[44] W-H. Wu, H-C. Hsiao, P-L. Wu, C-H. Lin, and S-H. Huang. "Investigating the learning-theory foundations of game-based learning: a meta-analysis," *Journal of Computer Assisted Learning*, vol. 28, pp. 265-279, 2012.

[45] K. A. Wilson et al. "Relationships between game attributes and learning outcomes review and research proposals," *Simulation & Gaming*, vol. 40, no. 2, pp. 217-266, 2009.

[46] T. Wagner. *The Global Achievement Gap: Why Even Our Best Schools Don't Teach the New Survival Skills Our Children Need - and What We Can Do*. Basic Books, 2008.

[47] MIT Media Lab. Scratch. [Online]. http://scratch.mit.edu

[48] S. Papert. *Mindstorms: Children, Computers, and Powerful Ideas*. Basic Books, 1980.

[49] S. Papert. *The Children's Machine: Rethinking School in the Age of the Computer*. Basic Books, 1993.

[50] W. Booth, "Mixed-methods study of the impact of a computational thinking course on student attitudes about technology and computation," Baylor University, Ph. D. dissertation, 2013.

[51] B. Köhler, M. Gluchow, and B. Brügge. "Teaching basic software engineering to senior high school students," in *K-12 Education: Concepts, Methodologies, Tools, and Applications*, Information Resources Management Association, Ed., 2014.

[52] N. Miyake and P. A. Kirschner. "The social and interactive dimensions of collaborative learning," in *The Cambridge Handbook of the Learning Sciences*, K. Sawyer, Ed. Cambridge University Press, 2014.

[53] C. Kazimoglu, M. Kiernan, L. Bacon, and L. MacKinnon. "Understanding computational thinking before programming: developing guidelines for the design of games to learn introductory programming through game-play," *International Journal of Game-Based Learning (IJGBL)*, vol. 1, no. 3, pp. 30-52, 2011.

[54] M. Resnick. "Sowing the seeds for a more creative society," *Learning & Leading with Technology*, vol. 35, no. 2, pp. 18-22, 2008.

[55] C. Wilson, L. A. Sudol, C Stephenson, and M. Stehlik. (2010) Running on empty: the failure to teach K-12 computer science in the digital age. [Online]. http://www.acm.org/runningonempty/fullreport2.pdf

[56] A. Reppening, D. Webb, and A. Ioannidou, "Scalable game design and the development of a checklist for getting computational thinking into public schools," in *Proceedings of the 41st ACM technical symposium on Computer science education*, 2010.

[57] P. Morreale, C. Goski, L. Jimenez, and C. Stewart-Gardiner. "Measuring the impact of computational thinking workshops on high school teachers," *Journal of Computing Sciences in Colleges*, vol. 27, no. 6, pp. 151-157, 2012.

[58] D. Barr, J. Harrison, and L. Conery. "Computational thinking: a digital age skill for everyone," *ISTE Learning and Leading*, vol. 38, no. 6, pp. 20-23, 2011.

[59] J. P. Gee and E. Hayes. "Nurturing affinity spaces and game-based learning," in *Games, Learning, and Society: Learning and Meaning in the Digital Age*. Cambridge University Press, 2012.

[60] C. Kelleher and R. Pausch. "Lowering the barriers to programming: a taxonomy of programming environments and languages for novice programmers," *ACM Computing Surveys*, vol. 37, no. 2, pp. 83-137, 2005.

[61] B. W. Becker. "Teaching CS1 with karel the robot in Java," *ACM SIGCSE Bulletin*, vol. 33, no. 1, pp. 50-54, 2001.

[62] C. Murphy, E. Kim, G. Kaiser, and A. Cannon, "Backstop: a tool for debugging runtime errors," in *Proceedings of ACM Special Interest Group on Computer Science Education*, 2008.

[63] E. Klopfer, H. Scheintaub, W. Huang, D. Wendel, and R. Roque. "The simulation cycle: combining games, simulations, engineering and science using StarLogo TNG," *E-learning*, vol. 6, pp. 71-96, 2009.

[64] B. Moskal, D. Lurie, and S. Cooper. "Evaluating the effectiveness of a new instructional approach," *ACM*

SIGCSE Bulletin, vol. 36, no. 1, pp. 75-79, 2004.

[65] Carnegie Mellon University. (2015) Alice. [Online]. http://www.alice.org/

[66] M. Sontag, "Critical thinking with Alice: a curriculum design model for middle school," in *Proceedings of the 2009 Alice Symposium*, 2009.

[67] S. Cooper, W. Dann, and R. Pausch. "Alice: a 3-D tool for introductory programming concepts," *Journal of Computing Sciences in Colleges*, vol. 15, no. 5, pp. 107-116, 2000.

[68] M. Conway, S. Audia, T. Burnette, D. Cosgrove, and K. Christiansen, "Alice: lessons learned from building a 3D system for novices," in *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, 2000, pp. 486-493.

[69] C. Kelleher and R. Pausch. "Using storytelling to motivate programming," *Communications of the ACM*, vol. 50, no. 7, pp. 58-64, 2007.

[70] W. Dann, D. Cosgrove, D. Slater, D. Culyba, and S. Cooper, "Mediated transfer: Alice 3 to java," in *Proceedings of the 43rd ACM technical symposium on Computer Science Education*, 2012, pp. 141-146.

[71] J. H. Maloney, K. Peppler, Y. Kafai, M. Resnick, and N. Rusk. "Programming by choice: urban youth learning programming with Scratch," *ACM SIGCSE Bulletin*, vol. 40, no. 1, pp. 367-371, 2008.

[72] Y. Kafai, K Peppler, and R. Chapman. *The Computer Clubhouse: Constructionism and Creativity in Youth Communities*. Teachers College Press, 2009.

[73] D. J. Malan and H. H. Leitner. "Scratch for budding computer scientists," *ACM SIGCSE Bulletin*, vol. 39, no. 1, 2007.

[74] MIT. (2015) ScratchEd. [Online]. http://scratched.gse.harvard.edu

[75] K. Brennan and M. Resnick, "New frameworks for studying and assessing the development of computational thinking," in *Proceedings of the 2012 annual meeting of the American Educational Research Association*, 2012.

[76] K. Brennan, M. Chung, and J. Hawson, "Creative computing: a design-based introduction to computational thinking," MIT, unpublished guide, 2011.

[77] T. Hammond, Alexander R., and A. Bodzin. "Assessment in authentic environments: designing instruments and reporting results from classroom-based TPACK research," in *K-12 Education: Concepts, Methodologies, Tools, and Applications*., 2014.

[78] U. Wolz, C. Hallberg, and B. Taylor, "Scrape: a tool for visualizing the code of Scratch programs," in *ACM Technical Symposium on Computer Science Education*, 2011.

[79] J. Bender. (2014) Scratch analyzer: transforming Scratch projects into inputs fit for educational data mining and learning analytics. [Online]. http://systemg.research.ibm.com/bigdata/reports/201412/report-201412-76.pdf

[80] K. Salen and E. Zimmerman. *Rules of Play: Game Design Fundamentals*. MIT Pess, 2003.

[81] Y. Kafai. "Playing and making games for learning: instructionist and constructionist perspectives for game studies," *Games and Culture*, vol. 1, pp. 36-40, 2006.

[82] M. Muratet, P. Torguet, J. P. Jessel, and F. Viallet. "Towards a serious game to help students learn computer programming," *International Journal of Computer Games Technology*, vol. 3, 2009.

[83] S. J. Sears. "Career and educational planning in the middle level school," *NASSP Bulletin*, vol. 79, no. 570, pp. 36-42, 1995.

[84] S. Sheth, J. Bell, and G. Kaiser, "A competitive-collaborative approach for introducing software engineering in a cs2 class," in *Proceedings of IEE Conference on Software Engineering Education and Training*, 2013.

[85] W. H. Bares, L. S. Zettlemoyer, and J. C. Lester. "Habitable 3D learning environments for situated

learning," *Intelligent Tutoring Systems*, pp. 76-85, 1998.

[86] M. Dickerson. "Multi-agent simulation and netLogo in the introductory computer science curriculum," *Journal of Computing Sciences in Colleges*, vol. 27, no. 1, pp. 102-104, 2011.

[87] J. Lave and E. Wenger. *Situated Learning: Legitimate Peripheral Participation*. Cambridge University Press, 1991.

[88] C. Steinkuehler, "Learning in massively multiplayer online games," in *Proceedings of the 6th International Conference on Learning Sciences*, 2004, pp. 521-528.

[89] F. Garzotto, "Investigating the educational effectiveness of multiplayer online games for children," in *Proceedings of the 6th International Conference on Interaction Design and Children*, 2007, pp. 29-36.

[90] S. M. Pulimood and U. Wolz, "Problem solving in community: a necessary shift in cs pedagogy," in *ACM SIGCSE Bulletin*, 2008, pp. 210-214.

[91] C. Bachen and C. Raphael. "Social flow and learning in digital games: a conceptual model and research agenda," in *Serious Games and Edutainment Applications*, M., Oikonomou, A., and Joain, L. Ma, Ed. Springer, 2011.

[92] G. Stahl, T. Koschmann, and D. Suthers. "Computer-supported collaborative learning," in *The Cambridge Handbook of the Learning Sciences*, K. Sawyer, Ed. Cambridge University Press, 2014.

[93] A. Collins and M. Kapur. "Cognitive Apprenticeship," in *The Cambridge Handbook of the Learning Sciences*, B. Sawyer, Ed. Cambridge University Press, 2014.

[94] A. Collins and R. Halverson. *Rethinking Education in the Age of Technology: The Digital Revolution and Schooling in America*. Teachers College Press, 2009.

[95] S. Barab, P. Pettyjohn, Gresalfi, M., and M. Solmou. "Game-based curricula, personal engagement, and the modern prometheus design project," in *Games, Learning, and Society: Learning and Meaning in the Digital Age*, C. Steinkuehler, K. Squire, and S. Barab, Eds. Cambridge University Press, 2012.

[96] M. Roy and M. T. Chi. "The self-explanation principle in multimedia learning," in *The Cambridge Handbook of Multimedia Learning*, R. Mayer, Ed. Cambridge University Press, 2005.

[97] M. Csikszentmihalyi. "Flow," in *The Encyclopedia of Psychology*, A. Kazdin, Ed. American Psychological Association and Oxford University Press, 2000, pp. 381-382.

[98] M. P. J. Habgood, S. E. Ainsworth, and S. Benford. "Endogenous fantasy and learning in digital game," *Simulation & Gaming*, vol. 36, no. 4, pp. 483-498, 2005.

[99] S. Järvelä and K. A. Renninger. "Designing for learning: interest, motivation, and engagement," in *The Cambridge Handbook of the Learning Sciences*, K. Sawyer, Ed. Cambridge University Press, 2014.

[100] A. Przybylski, C. Rigby, and R. Ryan. "A motivational model of video game engagement," *Review of General Psychology*, vol. 14, no. 2, 2010.

[101] J. E. Driskell and D. J. Dwyer. "Microcomputer videogame based training," *Educational Technology*, vol. 24, no. 2, pp. 11-17, 1984.

[102] M. Evans, E. Jennings, and M. Andreen. "Assessment through achievement systems: a framework for educational game design," in *Developments in Current Game-Based Learning Design and Deployment*. IGI Global, 2012.

[103] J. McGonigal. *Reality Is Broken: Why Games Make Us Better And How They Can Change The World*. Penguin Press, 2011.

[104] D. A. Kolb, R. E. Boyatzis, and C. Mainemelis. "Experiential learning theory: previous research and new directions," in *Perspectives on Cognitive, Learning, and Thinking Styles*, R. J. Sternberg and L. F. E. Zhang, Eds. Routledge, 2001.

[105] G. Greeno and Y. Engeström. "Learning in activity," in *The Cambridge Handbook of the Learning Sciences*, K. Sawyer, Ed. Cambridge University Press, 2014.

[106] D. Abrahamson and R. Lindgren. "Embodiment and Embodied Design," in *The Cambridge Handbook of the Learning Sciences*, K. Sawyer, Ed. Cambridge University Press, 2014.

[107] M. J. Reiser and K. Sawyer. "Scaffolding," in *The Cambridge Handbook of the Learning Sciences*, K. Sawyer, Ed. Cambridge University Press, 2014.

[108] R. Schank. (1992) Goal-based scenarios. [Online]. http://cogprints.org/624/1/V11ANSEK.html

[109] R. Schank. *Teaching Minds: How Cognitive Science Can Save Our Schools*. Teachers College Press, 2011.

[110] K. Squire, "Replaying history: learning world history through playing Civilization III," Indiana University, Ph. D. dissertation, 2004.

[111] D. W. Shaffer. "Models of situated action: computer games and the problem of transfer," in *Games, Learning, and Society: Learing and Meaning in the Digital Age*, C. Steinkuehler and K., Barab, S. Squire, Eds. Cambridge University Press, 2012.

[112] D. W. Shaffer. *How Computer Games Help Children Learn*. Palgrave Macmillan, 2008.

[113] J. Schell. *The Art of Game Design: A Book of Lenses*. CRC Press, 2008.

[114] J. Lu, S. Bridges, and C. E. Hmelo-Silver. "Problem-based learning," in *The Cambridge Handbook of the Learning Sciences*, K. Sawyer, Ed. Cambridge University Press, 2014.

[115] R. M. Ryan and E. L Deci. "Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being," *American Psychologist*, vol. 55, no. 1, 2000.

[116] R. M. Ryan, C. S. Rigby, and A. Przybylski. "The motivational pull of video games: a self-determination theory approach," *Motivation and Emotion*, vol. 30, no. 4, pp. 344-360, 2006.

[117] A. K. Przybylski, N. Weinstein, K. Murayama, M. F. Lynch, and R. M. Ryan. "The ideal self at play: the appeal of video games that let you be all you can be," *Psychological Science*, vol. 23, pp. 69-76, 2012.

[118] J. P. Gee. *What Video Games Have to Teach Us about Learning and Literacy*. New York Palgrove, 2003.

[119] T. W. Malone and M. R. Lepper. "Making learning fun: a taxonomy of intrinsic motivations for learning," in *Aptitude, Learning, and Instruction*, R. W. Snow, M. J. Farr, and M. J. Farr, Eds. Lawrence Erlbaum, 1987.

[120] C. M. Christensen, M. B. Horn, and C. W. Johnson. *Disrupting Class: How Disruptive Innovation Will Change the Way the World Learns*. McGraw-Hill, 2008.

[121] S. Khan. *The One World Schoolhouse: Education Reimagined*. Twelve, 2012.

[122] S. L. Turner. "Student-centered instruction: integrating the learning sciences to support elementary and middle school learners," *Preventing School Failure*, vol. 55, no. 3, pp. 123-131, 2011.

[123] D. I. Cordova and M. R. Lepper. "Intrinsic motivation and the process of learning: beneficial effects," *Journal of Educational Psychology*, vol. 88, pp. 715-730, 1996.

[124] C. Lewis. *Irresistible Apps: Motivational Design Patterns for Apps, Games, and Web-based Communities*. Apress, 2014.

[125] J. Rice. "Assessing higher order thinking in video games," *Journal of Technology and Teacher Education*, vol. 15, no. 1, pp. 87-100, 2007.

[126] S. Barab. "Design-based resarch: a methodological toolkit for engineering change," in *The Cambridge Handbook of the Learning Sciences*, K. Sawyer, Ed. Cambridge University Press, 2014.

[127] M. J. Nathan and M. W. Alibali. "Learning sciences," *Wiley Interdisciplinary Reviews: Cognitive Science*, vol. 3, no. 5, pp. 329-345, 2010.

[128] J. W. Creswell. *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches.* Sage, 2013.

[129] K. Squire. *Video Games and Learning: Teaching and Participatory Culture in the Digital Age*. Teachers College Press, 2011.

FIGURES & TABLES

Figure 1. Tooling Scratch: Concept Map

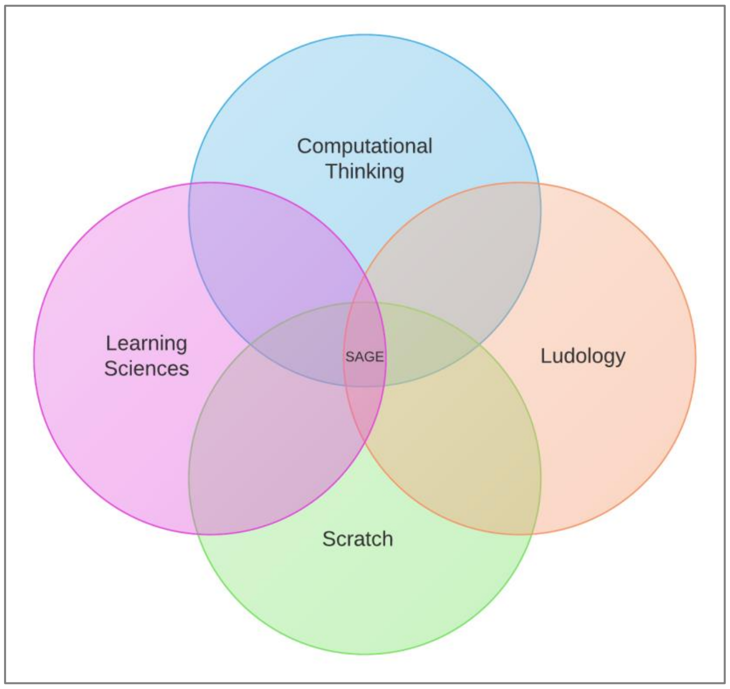


Figure 2. Karel the Robot [61]

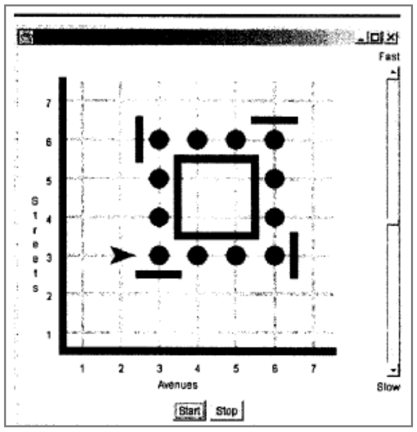


Figure 3. Light-Bot [5]

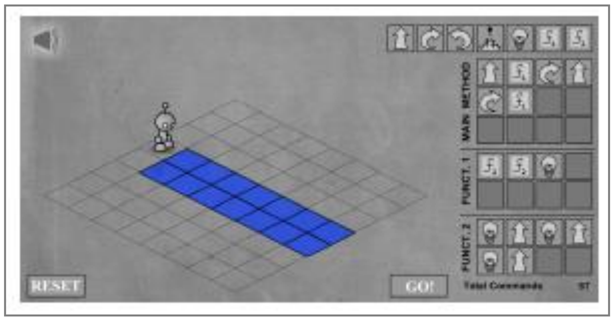
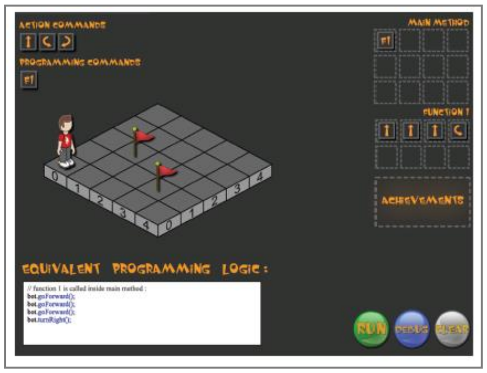


Figure 4. Interactive-Feedback Prototype [53]



```
// function 1 is called inside main method :
bot.goForward();
bot.goForward();
bot.goForward();
bot.turnRight();
```

Figure 5. StarLogo TNG [63]

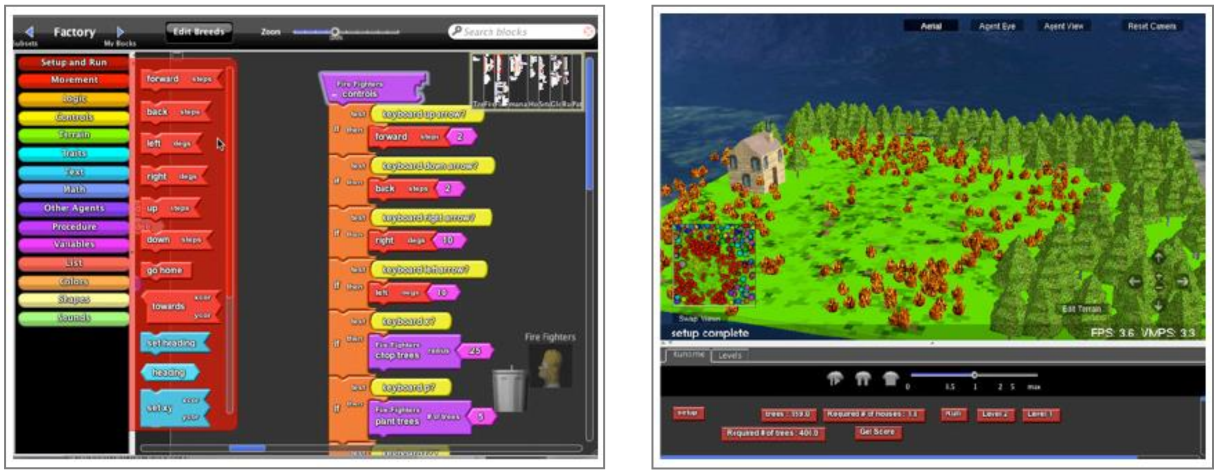


Figure 6. Alice [65]

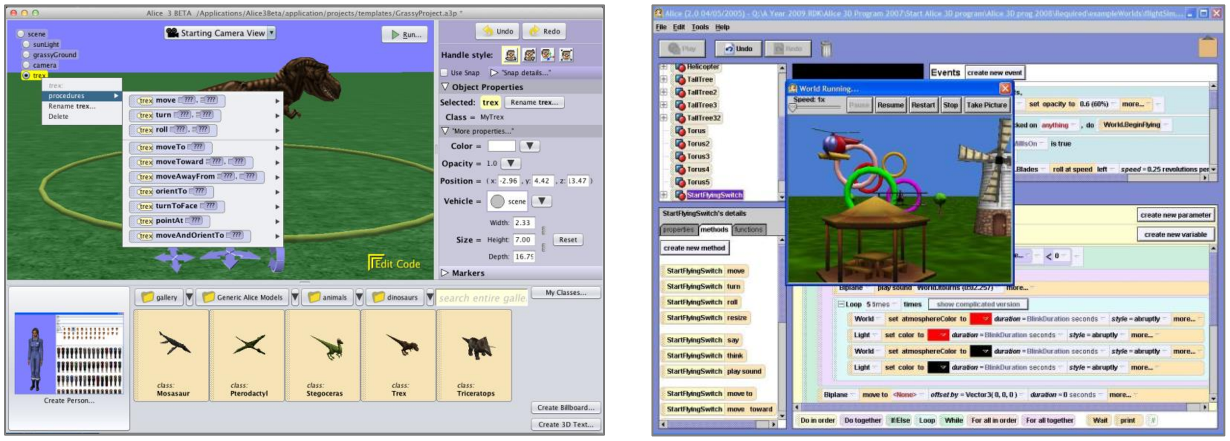
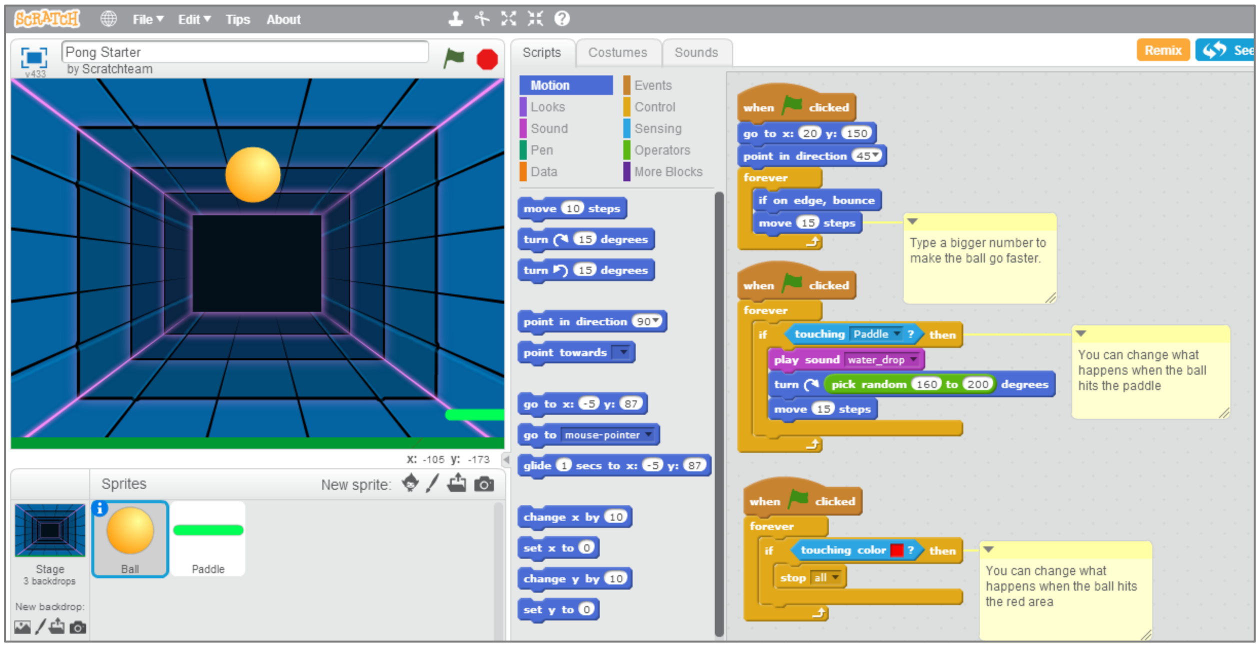


Figure 7. Scratch [47]



```
when ⚑ clicked
go to x: 20 y: 150
point in direction 45
forever
    if on edge, bounce
    move 15 steps

when ⚑ clicked
forever
    if touching Paddle ? then
        play sound water_drop
        turn ↻ pick random 160 to 200 degrees
        move 15 steps

when ⚑ clicked
forever
    if touching color 🟥 ? then
        stop all
```

Type a bigger number to make the ball go faster.

You can change what happens when the ball hits the paddle

You can change what happens when the ball hits the red area

Figure 8. Scrape Output [75]

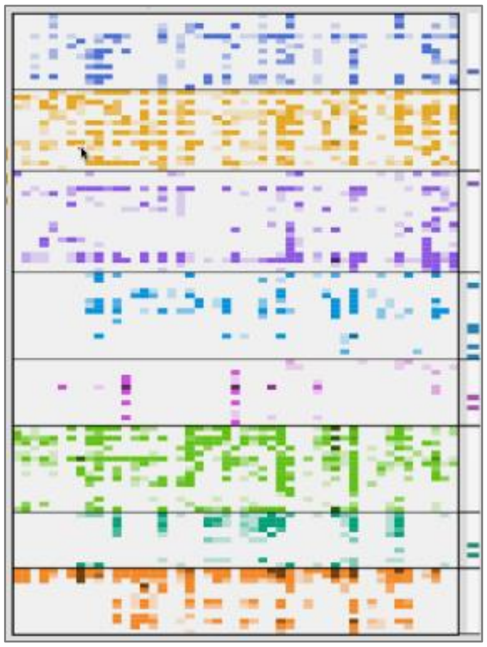


Figure 9. Scratch Analyzer [79]

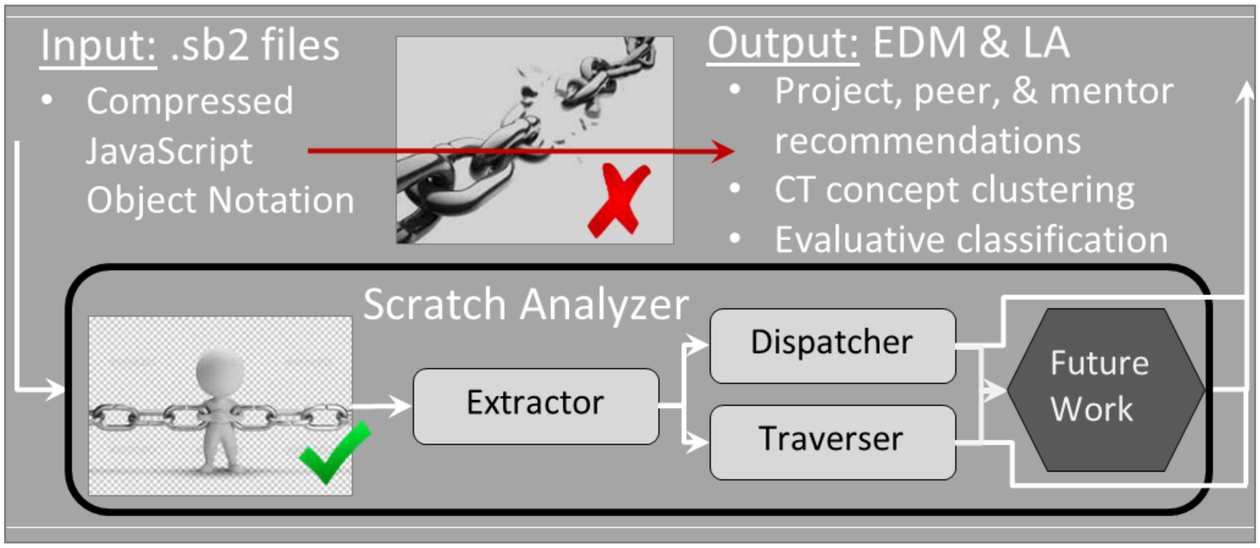


Table 1. SAGE Theories, Learning Principles, and Design Tactics

	Theories	Learning Principles	Design Tactics
Social	• Social gameflow • Computer supported collaborative learning • Cognitive apprenticeship	<u>Nurturing Affinity Space</u> SAGE affiliates students by enveloping them in shared endeavors that require cross-functional collaboration, thereby creating multiplicities of respected expertise.	• Role provisioning • Peer-visible explanation facilitation (elected and prompted) • Script sharing and embedded chat (elected and event-driven)
Addictive	• Flow • Interest • Motivation • Engagement	<u>Regime of Competence</u> SAGE induces flow and cultivates engagement by presenting pleasantly frustrating challenges that stretch the competencies of each individual student.	• Game mechanic blocks with adaptive constraints • Configurable block point system integrated with game mechanic blocks and levels • Achievement system
Gameful	• Experiential learning theory • Learning in activity • Embodiment • Scaffolding • Goal-based scenarios • Story-centered curricula	<u>Situated Meaning</u> SAGE does not assemble CT concepts disconnected from gameplay, but instead envelopes students in gameful environments in which concept exploration emerges as narratives unfold.	• Integrated storyboard and leveling controls • Palette/block restriction system supportive of narrative structure • Block-crippling mechanism facilitating endogenous stories
Engineering	• Problem-based learning • Self-determination theory • Cognitive evaluation theory	<u>Explicit Information On-Demand and Just-In-Time</u> SAGE explains directly only when students eagerly seek or desire information and have accrued the experience and motivation imperative to consume that information lucidly.	• Context-aware, event-driven, just-in-time CT knowledge system • Embedded, on-demand, multi-sourced instruction modules • Support for diagnostics and performance measurements