A Study of Early Stage Game Design and Prototyping

Brien Colwell, Richard C. Davis, James A. Landay
DUB Group
University of Washington, CSE Box 352350
Seattle, WA 98195
xcolwell@gmail.com, rcdavis@cs.washington.edu, landay@cs.washington.edu

ABSTRACT
Computer games and simulations can be valuable teaching and communication tools, and they are a powerful form of self-expression. Unfortunately, creating games requires programming, and programming requires time and skill. Some tools facilitate game creation to motivate novice programmers, but programming is still necessary. Other systems require less programming, but they are narrowly focused. To enable faster, simpler, and more expressive tools for professionals and amateurs, we have explored the processes and tools used in the early stages of game and simulation design. Interviews with educators clarified the uses of simulations in the classroom, while interviews with professional game designers uncovered a need for a new medium for prototyping interaction. We also conducted a study that observed seven groups of children designing games with words, sketches, and animations, finding significant advantages to sketches and animations. Finally, we refined an interface optimization design technique and applied it to this domain as a first step toward a new game and simulation prototyping tool.

Author Keywords
Games, simulations, design, animation, prototyping

ACM Classification Keywords
H5.m. H5.2 [Information interfaces and presentation]: User Interfaces - Graphical user interfaces.

INTRODUCTION
End users are starting to develop simple games and simulations for their friends, families, or students. Such programs need not be polished (as shown by addictive games like Sketch Fighter [2], Crayon Physics [29], and the Line Rider game shown in Figure 1 [5]), but they must reflect the vision of their creators. While many can envision new interactive experiences, however, few have the time or programming skill to realize them. Some tools provide simplified programming environments for games, but teaching programming is their primary focus [18, 23, 32]. Other tools reduce or eliminate programming from building and prototyping games or simulations, but they are too narrowly focused to serve as general purpose tools.

Previous research has shown the value of sketching in design [4, 8], as well as the benefits of sketch-based design [19] and animation tools [9]. Starting a prototyping process with sketching would give precedence to crafting the concrete visual and dynamic aspects of a game or simulation. This could facilitate the flow of ideas and give end-users the anchors they need for adding behavioral details. We hypothesize that this approach will help end users unlock creative their potential in this area.

With this goal in mind, we have explored the processes and tools used in early stage game and simulation design. Our exploration began with two sets of interviews. In the first, we interviewed educators who wanted to give students simulations as learning exercises. These interviews showed the need for a simple simulation builder with basic graphics but precise physical motions. We then interviewed professional game designers and discovered a need for fast, simple, and expressive prototyping tools. Designers' needs are similar to end users' needs, because they also need to work quickly and avoid programming.

We tested our hypothesis that sketching and animation would facilitate prototyping of games and simulations by observing seven groups of children prototyping games. Each group worked in three media: text, static sketches, and sketched animation. We found that sketching and animation generated more unique ideas than writing. We also found that animation helped these children temporally situate events, tell stories, and collaborate spontaneously with one another.

We then took a step toward designing a fast, simple, and expressive prototyping tool by performing interface

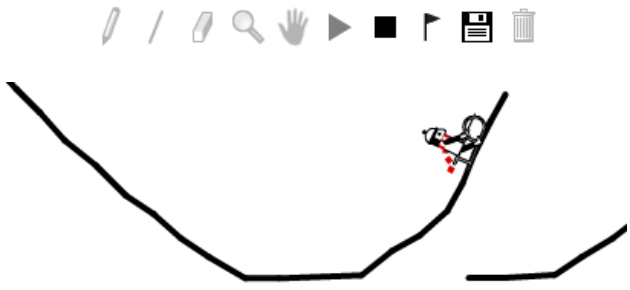

Figure 1: Line Rider is a popular game with simple graphics but rich animation and interactivity.

optimization to visualize the design space. Following the pattern set by K-Sketch [9], we first collected a set of 27 usage scenarios and defined a set of 83 operations for completing those scenarios. Because the existing interface optimization method did not scale to such large operation sets, we developed a new processing algorithm. The resulting data shows several interesting points in the design space, including one occupied by current tools.

In summary, this paper makes the following contributions:

1. Interviews with educators and game designers that clarify the requirements of an end-user game and simulation prototyping tool
2. A study that compares text, static sketches, and sketched animation and shows the benefits of sketches and animation
3. A more scalable interface optimization method with improved handling of the speed dimension
4. Interface optimization results for an end-user game sketching system, including 27 usage scenarios, 83 operations, and a visualization of the design space.

The following section gives an overview of research and tools related to this project. After this, we present our interviews with educators and game designers. The next section describes our study comparing text, static sketches, and sketched animation. We then explain how we collected our usage scenarios and defined operations, after which we describe our modifications to interface optimization, including the optimization results and implications for design of an end-user game prototyping tool. We close with conclusions.

RELATED WORK

We preface our exploration of game prototyping methods and tools by situating our research relative to previous work in three areas. First, we look at research into why end users create simulations and games. We then review current game and simulation prototyping tools. Finally, we give examples of research that shows how sketching and physical action can facilitate creative expression.

Why People Create Games

To build a game and simulation prototyping tool that supports end users, it is important to understand what they wish to create. We have observed common themes running through the literature in education, commercial game customization, and research into games in culture. We also see evidence that professional game designers would benefit from a fast, simple, and expressive prototyping tool.

There are numerous examples of educators and education researchers using games and simulations as teaching tools. Some generally advocate the use of games because of their ability to deeply engage students in goal-directed activities that can teach a variety of ideas [35]. One success in this area has been the use of The Sims to teach foreign languages [30]. To achieve educational goals, however, many games need customizations that take programming skill to implement.

Simulation has been a more widely studied tool, particularly in teaching science. Inquiry learning, for example, is a promising teaching strategy in which students repeatedly form and test hypotheses on a simulator [3]. Though they need not be polished, good simulations are hard to find. The few teachers with the time and skill to make them need lots of support. The Physlets [6] community, for example, helps physics teachers create Java applets that illustrate physical phenomena.

In the game industry, many believe that user-created content and customizations will dominate game play. Games like The Sims [11], Spore [12], and Second Life [20] have formed large communities of players who spend countless hours creating content. Some players get so involved that the lines between reality and fantasy begin to blur [17]. Users' clearly have vast creative energy for games, but these games only allow customized content, not entirely new game play.

Games are a vital part of culture [16]. Some are beginning to study how children engage their culture by creating computer games [28, 32]. The fact that making games is still far less common than collecting internet images into documents may be a sign that there is room for more engaging game building tools [28]. The casual game industry is another sign of this burgeoning culture [13]. Casual games are simple games [5, 29] created by individuals or small teams and are available for free or for a small fee. To survive, casual game makers must create many games on short product cycles with small teams. Some designers are calling for cheaper and faster prototyping methods that avoid programming, such as "sketching" prototypes with wizard of oz methods [1].

We believe that that a fast, simple, and expressive game prototyping tool would serve both educators and individuals participating in gaming culture. Our interviews investigated the needs of these communities in greater depth.

Game and Simulation Prototyping Tools

Today, a user who wishes to create a game or simulation with minimal programming has several, imperfect options. Many reach for an agent-based structured editor like Alice [18], eToys [34], AgentSheets [31], StarLogo TNG [24], or Scratch [23] that structures programming activities around the creation of games. All of these systems allow characters to be defined graphically, after which behaviors can be added programmatically.

These systems are quite powerful, but the transition into programming is difficult for many. We believe that this transition can be softened by allowing end users to spend more time sketching and animating object relationships before making the leap to programming. We also believe that such systems could benefit from an analysis of

scenarios like the one presented here as a means for selecting the most important programming operations.

Programming by demonstration systems are a close relative of agent-based structured editors that attempt to simplify programming by inferring behavior from demonstrated examples [22, 27, 33, 37]. Unfortunately, these systems have not been popular, because inferring a program from examples is a difficult problem. We avoid inferring behavior for that reason, but we do seek to take advantage of demonstrated animation, which is similar in spirit.

A third approach to prototyping a simulation is to use a systems modeling or domain specific prototyping tool. Systems modeling tools, like Stella [15], Simulink [36], or LabView [25], have visual builders for modeling dynamic systems and can connect to 3D models and graphs. These are powerful systems, but their visual languages are no simpler than the others presented here. Domain specific prototyping tools, like Interactive Physics [10] and the Molecular Workbench [7], have less programming but they do not allow users to create new interactions. Thus, these all fall short of allowing a wide variety of games to be prototyped quickly and easily.

Informal Interfaces

Since Csikszentmihalyi introduced the concept of optimal experience [8], many have sought to support users' creative flow by removing unnecessary obstacles. Informal interfaces are systems that support creative flow by deferring the specification of details until they become necessary [9, 19]. These systems often involve sketching, because sketches are the most valuable representations of thought in the early stages of design. As Buxton puts it, "Their value lies not in the artifact of the sketch itself, but in its ability to provide a catalyst to the desired and appropriate behaviors, conversations, and interactions" [4].

K-Sketch is an informal interface for sketching and demonstrating animations that targets novice animators [9]. Evaluations of K-Sketch have shown that users can focus on higher level tasks while using it, much as they can while sketching. Since animations can reflect much of the dynamic activity in games and simulations, we have hypothesized that sketched animation will help end users explore games designs in new and powerful ways.

Stimulating Creativity with Physical Action

K-Sketch animations are created by recording real-time hand gestures, and there is evidence that this physical action may also promote creativity. Oulasvirta and colleagues found inspiration for ubicomp applications through bodystorming, i.e., placing themselves in the physical relationships required by their designs [26]. Lundgren found that physical experimentation with a complex mechanical table gave people a surprising ability to "program" games on it [21]. We have found that people experience similar benefits when physically demonstrating motions in a game prototype.

Our research lies at the convergence of these disparate themes. Through sketching and demonstration, we hope to provide end users with a prototyping tool that is faster, simpler, and more expressive than any that is currently available.

INTERVIEWS WITH EDUCATORS

To better understand the needs of end-user simulation programmers, we interviewed four educators who had a desire to create simulations. The results of these interviews are summarized in Table 1. All four participants were education graduate students or post-docs, and all were charged with developing new curricula as part of an inquiry learning project. Two were men, and two were women.

For educators 1 and 2, simulations played a role that is fairly common in inquiry learning exercises. Students would repeatedly formulate and test hypotheses on the simulation. Educator 3 wanted to show his students that the same principles of equilibrium applied to many natural processes. He hoped to connect the same simulation to multiple sets of graphics (see Figure 2). Educator 4 wanted her students to create simulations as a learning exercise.

None of these educators had found the time to build the simulations they envisioned. Educator 1 did not know of any appropriate simulation system and was searching for one. Educator 2 had a planetary motion simulator, but she wanted it to use real physical units, and she wanted some planets to look like stars. She did not have time to build a simulator and was looking for a better one to avoid changing her curriculum. Educator 4 wanted to give her students a fast way to make simulations. She considered the Molecular Workbench, but feared that it provided too much help and too few opportunities for learning. Educator 3 liked Stella, because it could produce pleasing continuous graphs, but he was thinking of using AgentSheets, because it was simpler. Though this tool has limitations, this was the only participant with a plan for building his simulation.

#	Subject	Suggested simulations
1	earthquakes	mass–spring models of buildings shaken with waves of varied amplitude and frequency
2	gravity	planets and stars of varied mass affecting each other's trajectory
3	ecology	equilibrium: sharks–fish, farms–fish, bulldozer–trees
4	molecular theory	particles vibrate, change from solid to liquid to gas

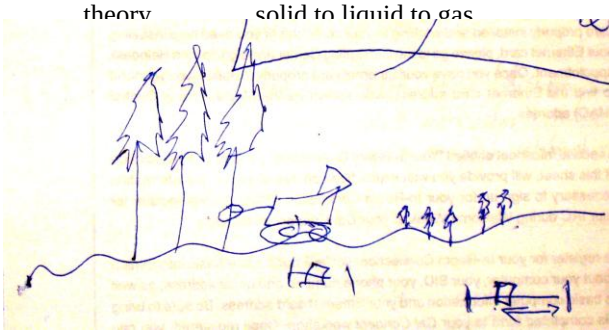

Figure 2: Drawing from Educator 3 showing a bulldozer clearing trees and new trees growing. Sliders control the the growth rate and death rate of trees.

Though the subject matter of these simulations varies, the controls and behaviors are fairly similar. All had variables connected to slider controls that affected the motion of objects or the rate at which they appeared and disappeared. With the exception of the ecology simulations, the objects in these simulations move as if they were subject to physical forces. While these educators didn't demand that motions be perfect, they did need to be close enough that students would recognize physical processes.

As an experiment, we produced one of Educator 2's simulations with eToys. This simulation showed a moon revolving around a planet in a mathematically accurate way. The educator rejected the simulation, because the motion was too choppy. We suspect that Educators 1 and 4 would have rejected eToys versions of their simulations for the same reason.

These interviews show how educators would benefit from a simple simulation tool that is expressive enough for a variety of disciplines. The tool should allow quick assembly of user controlled variables and objects that move with simple, predefined motions or according to physical laws.

INTERVIEWS WITH GAME DESIGNERS

To better understand the state of the art in game prototyping methods, we interviewed three professional game designers, each for one hour. As shown in Table 2, all played key roles in their teams, and all had at least a decade of experience. Designers 1 and 2 were from small studios that delivered casual and mixed-reality experiences over the web; Designer 3 was from a large console game studio.

Each member of Designer 3's team worked in a different stage of a production pipeline, and communication from later stages to earlier stages was limited. The first two stages were creating a spec and design document; the last stage was creating the game experience. If the game experience team had an idea for a new game element motivated by how people were experiencing the game, there was little they could do. Those designers could have benefitted from a medium for prototyping game modifications.

Designers 1 and 2 had more interactive teams, but in both cases, at least one person on the team worked remotely. At the time of the interview, Designer 1 was working on an educational game with a remote developer. The game started with a high-ideation sketch composed of a drawing and a two paragraph description. Over the course of 140 revisions, the text grew to a five page description of game elements and interactions. Designer 1 was not able write code; text was the only way he could collaborate. He wanted a tool that would allow him to communicate his ideas.

#	Occupation	Experience	Game types
1	lead designer	10+ years	mixed reality, casual
2	lead designer	10+ years	casual
3	art lead	15+ years	console

Table 2: Results from interviews with game designers

At the time of the interview, Designer 2 was working on a storyboard for a new casual game. He often used paper-prototyping and annotated sketching to experiment with interactions, but at the time of the interview, he was working with a remote developer. The best he could do was send snapshots and textual design documents. Because of the volume of minutia in each interaction and game element, this designer used relative language when communicating with text. For example, if he was exploring ideas for a game with a remote developer, he might say "this game is like X but we're going to heat drums."

Both Designers 1 and 1 used Flash to prototype. Designer 2 found Flash slow for prototyping since it focused on details, and he had tried several other tools, including GameMaker and GameBrix. However with every tool he tried, he eventually hit a wall. He observed either "there are too many features that the prototyping tool is made to support... or it's [so] specific that only certain games can be made." He mentioned processing as an example of what he considers good design for a tool: being able to drop out into one page of code and keeping the GUI minimal. He also wanted the ability to tweak game rules as a game is running. These are key features for a professional tool, though they are less important for novices.

These interviews show that designers also need a simple medium for discussing interactivity. The medium should allow remote collaboration, and it should be fast for team members with no programming skill. Next, we describe a study exploring possible media for such collaboration.

A STUDY OF GAME DESIGN IN THREE MEDIA

We hypothesized that being able to sketch and move game objects would help users generate ideas for those objects' interactions. To test our hypothesis, we designed a study that compares the description of a game expressed using three different media: writing, sketching, and animation. Our participants were children interested in creating games. The study started with a high ideation period, where the coordinator would talk with each participant about games in general, what games interested them, and what game they would like to design. The coordinator would then give participants a piece of paper and ask them to draw or write about the game.

As the participants were creating their games, the coordinator would ask questions about their design goals. When the coordinator felt a participant had formed a good enough notion of his/her game, the coordinator would give the participant a workbook and ask him/her to start with either sketching or writing responses (alternated each time). When the participant had finished sketching or writing answers to all questions, the coordinator would ask him/her to answer the questions again with the other medium

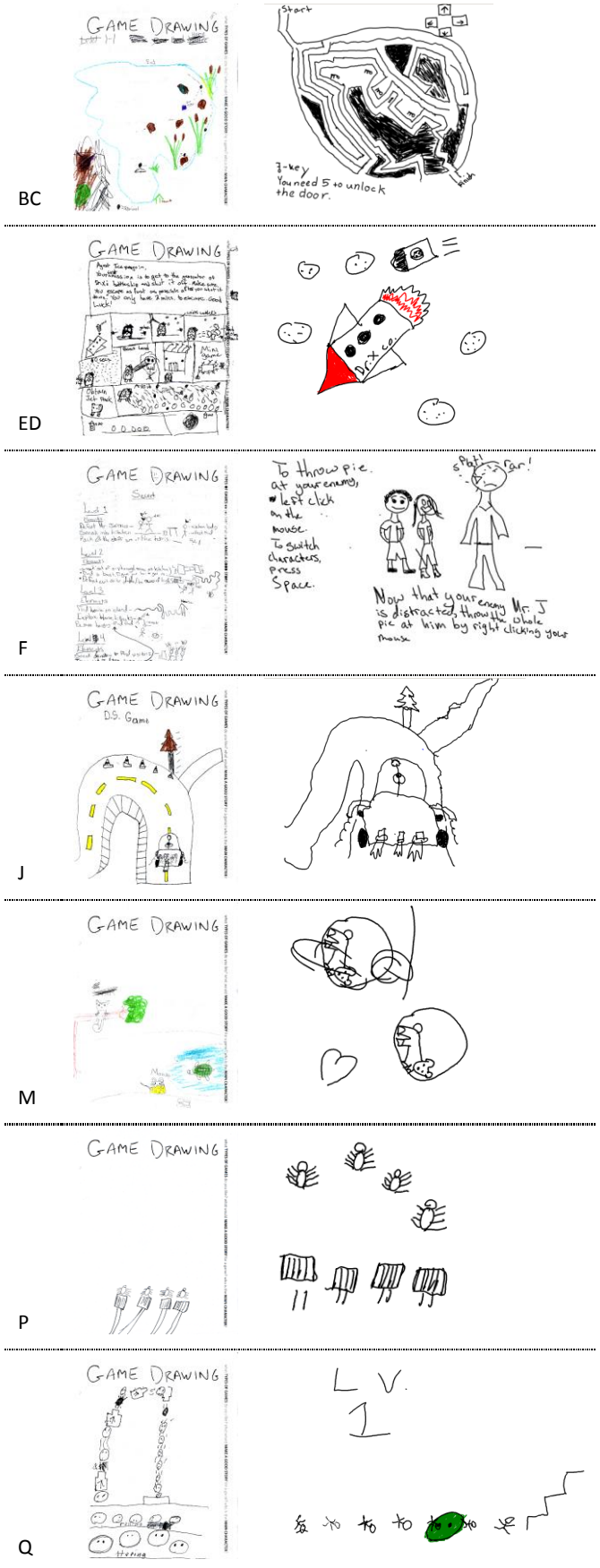

Figure 3: Data from the three media study: participant ID, initial concept, and a scene from the animation.

(writing if sketching was initially used, and vice-versa). The workbooks contained four questions: how do you control the main character, what is the goal, what are the obstacles, and how do you win?

When a participant finished answering the questions with both sketching and writing, the coordinator would give them a Tablet PC with K-Sketch [9] running in full screen mode. The coordinator would let the participant acclimate to drawing with the tablet before showing him/her how to create basic animations (using four K-Sketch operations: translate, rotate, scale, and orient to path). Once the participant was able to create basic animations the coordinator would ask him/her to answer the workbook questions using animation. The coordinator would not restrict the time spent using each medium.

A pilot of the study was run at three community centers in the Seattle area. After modifications to remove paper-prototyping and add more scaffolding (the workbook questions), the study was run in four sessions spanning two weeks at an elementary school summer program in the Seattle area. The coordinator set up at a table in the main room and let children come over as they were interested. 12 groups total participated in the study (15 children, six working in pairs); of those, seven groups (composed of two pairs of girls, two more girls, and three boys) answered the workbook using all three media. It took a average of 10–15 minutes to teach a group how to use the animation tool.

Our target demographic was children ages 7–14, since they are largely unbiased by the current creative paradigm yet have incredible creativity and enthusiasm. All of the children who participated fit our age demographic. However, Participant E had attended a summer workshop for creating games, but did not have programming experience.

When asked to complete a workbook with a medium, a common response from participants was "I don't know how to X this", where *X* was *draw* or *animate*. Of the seven groups who worked in all three media, none said they could not write a response. We asked each participant to give their best effort and we marked "N/A" for each question a participant attempted but could not answer.

Results

Figure 3 shows the games created by the seven groups. In our data, each participant is assigned a letter (e.g., E). The group name of a pair is the concatenation of the individual participants' letters (e.g., ED).

Using the categories in Table 3, we counted the number of total and unique elements expressed in each category, for each medium. A unique element is defined as an element that appears in only one medium. The initial sketch or writing was included in the counts. The final counts are listed in Table 4.

In all but two categories (Obstacles and Actors) one medium showed considerable favor. More Objective ideas

were expressed with writing; more Scene and Interaction ideas were expressed with sketching; more Action ideas were expressed with Animation. Obstacle and Actor ideas were close between Writing and Sketching, with Writing being used slightly more. The uniqueness counts agreed with the total counts in all but one category (Actors).

We identified three trends in the role of animation, which we detail below with concrete examples.

1. Temporally situating events

In this role an animation with one or more actors moving was looped continuously. The creators would observe the movement and consider *what could happen*. At some point the creators would stop the playback, seek to a point in time, and add new movement to the animation. The process would then repeat.

Concretely this role was observed in three groups of the seven groups: J, M, and Q. Participant J ("car racer") created an animation where a car would turn around a bend, past a tree. After continual looping he decided the tree should fall down as the car drives past. He then stopped the animation and added the falling motion. Participant M ("animal sims") started with two hamsters racing side by side. After watching the animation she decided that one hamster should hit a wall and another should pick up a heart. She then stopped the animation, added a wall and a heart, and modified the motion of one hamster to reflect the new wall. Participant Q created an animation of an airplane moving through the sky and a spinning blob. After looping the airplane movement he decided that the blob should shoot into the sky past the airplane. He grabbed the blob while the animation was playing and moved it into the sky.

2. Storytelling or role playing in a scene

In this role an animation is a backdrop in which the motion of one actor is demonstrated while the creator tells the story of why the movement is happening. We likened this role to puppetry.

Concretely this role was observed in two groups: F and Q. Participant F ("the secret") moved a kid through a kitchen while a cook was moving. From the movement of the kid she evolved the story of why the cook was moving and what the kid was trying to do. Participant Q started with a row of army men and then moved a blob through them and up to the right while explaining "the blob has to step on the army men to get to the finish." He went back and animated the army men falling over and drew stairs for the blob to reach the finish.

Scene:	backdrop and environment; buildings and walls; placement of actors
Objective:	a short- or long-term goal that drives actors' actions
Obstacle:	a situation an actor encountered that inhibits flow (movement, action, etc) of an actor
Action:	an exchange between actor and itself or another object in the world; causes change
Actor:	object that creates actions; can be player or non-player
Interaction:	Physical controls human uses

Table 3: Categories used in study analysis

	Scene			Objective			Obstacle			Action			Actor			Interaction		
	W	S	A	W	S	A	W	S	A	W	S	A	W	S	A	W	S	A
BC	1	1	1	2	1	2	2	0	2	5	1	2	8	3	2	2	1	1
ED	2	12	6	4	5	3	3	5	4	1	5	9	2	5	4	0	3	1
F	7	3	2	10	6	3	7	6	3	5	6	7	6	4	3	1	3	2
J	0	1	1	3	0	1	3	4	3	3	3	3	1	1	2	0	1	0
M	0	3	1	3	2	2	4	2	1	1	4	2	0	2	1	0	1	0
P	0	1	1	2	1	0	2	2	1	2	1	2	3	4	2	1	1	0
Q	0	1	2	2	1	3	2	2	2	2	3	7	1	3	4	0	1	0
BC	1	1	1	2	1	2	2	0	2	5	1	2	6	1	1	1	0	0
ED	0	6	0	0	1	2	1	1	2	0	1	8	0	1	0	0	2	0
F	4	0	0	5	2	2	4	2	1	3	3	4	3	0	0	0	1	0
J	0	0	0	3	0	1	3	1	0	3	3	2	0	0	1	0	1	0
M	0	3	1	3	2	2	2	1	1	0	3	1	0	2	1	0	1	0
P	0	0	0	2	1	0	0	0	0	1	0	1	0	1	0	1	1	0
Q	0	0	1	1	1	2	0	1	0	1	0	3	0	0	1	0	1	0
mean	1.4	3.1	2.0	3.7	2.3	2.0	3.3	3.0	2.3	2.7	3.3	4.6	3.0	3.1	2.6	0.6	1.6	0.6
mean	0.7	1.4	0.4	2.3	1.1	1.6	1.7	0.9	0.9	1.9	1.6	3.0	1.3	0.7	0.6	0.3	1.0	0.0
std	2.6	4.0	1.8	2.9	2.3	1.2	1.8	2.1	1.1	1.7	1.9	3.0	2.9	1.3	1.1	0.8	1.0	0.8
std	1.5	2.3	0.5	1.6	0.7	0.8	1.5	0.7	0.9	1.9	1.4	2.4	2.4	0.8	0.5	0.5	0.6	0.0

Table 4: Element counts per category in each medium. White rows are total counts; gray rows are unique counts. The maximum mean in each category is highlighted. W: writing; S: sketching; A: animation

3. Spontaneous collaboration

In this role an animation serves to show what is possible in a world, allowing others to grasp what has been explored and contribute ideas on what could be explored.

Concretely this role was observed in two groups: BC and Q. Group BC ("tadpole rescue") internally alternated drawing a maze and keys hidden inside a maze. One of them would draw a few walls, and the other would consider the best next addition and draw it, etc. Participant Q created an animation with a blob flying into the sky past an airplane. Another kid in the room noticed the blob, came to the table, and almost immediately suggested "what if the blob bounced off the airplane." Within seconds Q had integrated the feedback and animated the blob bouncing off the airplane.

Discussion

One value we did not count in the analysis is how many times an element recurred within a medium. If we had, we suspect writing would have had the lowest net recurrence count. When writing, participants tended to produce smaller, fragmented ideas rather than developing a single idea. For example, rather than developing a single scene for a game, Participant F created a plot that spanned escaping an orphanage, finding, navigating, and fighting on a boat, and then delivering food to visitors in a basement. The fragmented nature of written ideas may also explain the high unique Actor and Objective count in writing.

Confirming our hypothesis, we believe the principles of bodystorming came out in the Animation data. While Scenes, Actors, and Interactions are static, Actions flow with time. Animation enabled the students to visually situate the actors and scenery in a moment, which let them "live in" possible actions and explore them as they came. However, surprisingly Animation was used least frequently to specify Obstacles, which are also situational. One explanation is that Animation was used to explore the details of a single obstacle.

From this data we conclude that sketching and animation are not merely a step in the creative process; they are part of a continuous process that makes ideas tangible. Once tangible, ideas can be discussed among several people, reconsidered, and evolved. The visual overview afforded by sketching and the quick feedback of motion and causality afforded by animation cannot easily be recreated in a text document.

A SCENARIO LIBRARY OF GAME AND SIMULATIONS

To study how well tools for creating games and simulations balance expressivity, simplicity, and speed, designers need data with which to test them. The data should help illuminate what users of the tool would be able to create and what operations they would need to master. In this section we present data we believe will be useful to help test all game and simulation design and prototyping tools.

Our process involved creating a library of games and simulations, extracting the essential elements, identifying common operations (which can correspond to interface elements), and then encoding as many different approaches using these operations as time permitted.

#	Name	Description
1	edit	edit animation (in ways not found in K-Sketch)
2	play	control playback of motions
3	spatial	test and control the spatial relationship of objects
4	cnd-sp	choose a subset of objects for spatial tests and controls
5	i/o	create controls that receive input and create visual output
6	draw	control the drawing system
7	pixel	test and control individual display pixels
8	phys	move objects using a physical model
9	visual	control visual properties of objects
10	rand	generate random numbers or selections
11	state	store state in a variable
12	seq	execute a sequence of operations
13	camera	control and interact with the user's camera (view)
14	grid	create & control grid of visual objects
15	sound	control playback of sounds
16	cnd-vis	choose a subset of objects to make visible
17	cue	conditionally cue events
18	flow	control how information flows to and from variables

Table 5: Categories of operations in our data

Choosing a Set of Games and Simulations

We created a library of fourteen games and thirteen simulations that we believe are representative of the types of games and simulations our target audiences want to create. These games and simulations range from casual games to academic simulations. The casual games (10) came from each category listed in the IGDA (International Game Developer's Association) 2006 whitepaper [14]. The remaining four games were taken from a list of innovative casual and console games. We found simulations spanning

10 disciplines: economics (1), business (1), theater (1), dance (1), marine biology (1), physics (4), genetics (1), brain science (1), air traffic control (1), and biology (1). All but one of the games and simulations are 2D.

Extracting Essential Elements

Each game and simulation in the library involves many elements. To focus on the essential elements, we studied answers to the following two questions: 1. What actions happen in the game? 2. Which elements of the game, if removed, would make the game no longer fun?

One researcher answered these questions for each game and simulation in the library. Additionally, we posed the question on Amazon's Mechanical Turk. We submitted 27 tasks, one for each game and simulation, and received 102 responses. There were 51 unique respondents, each completing an average of 2 tasks (standard dev. 2.1). They were paid $.30 for each completed task (none were rejected). For each game or simulation we kept the first four responses that satisfactorily answered the questions. Common discarded responses were ones that answered relative to another similar game or simulation (e.g., "it's just like X") and ones that focused solely on the playability of the game (e.g., "this game is too hard"). A final description of the essential elements for each game and simulation was created by taking the union of the essential elements mentioned by the researcher with essential elements mentioned in at least two of four responses obtained from our online participants.

Defining a Set of Operations

Following the pattern set by K-Sketch [9], one researcher iteratively coded the library to define a set of operations. This process begins with enumerating the *features* that a user would have to represent to complete each game or simulation. For each feature, the researcher then listed one or more *approaches* to representing that feature and noted common *operations* required by each approach. The final encoding was reached after five iterations through the library. Operation categories are listed in Table 5, and the operations themselves are listed in Table 6. The final encoding had an average of 1.84 approaches per feature (recent analysis of the K-Sketch encoding shows that it had an average of 1.77 approaches per operation).

UNDERSTANDING DESIGN TRADEOFFS

The interface optimization technique processes a coded library of features, approaches, and operations to help interface designers produce the fastest, simplest, and most expressive points in a design space [9]. The technique identifies small sets of operations (simple) that support large numbers of scenarios (expressive) using fast approaches (fast). A set O of operations *supports* scenario S if all the features of S can be represented with one or more approaches for which all operations are contained in O.

The existing interface optimization technique had two significant problems: slow execution time of the optimization algorithm and a poor definition for "fast approaches." In this section we first explain how we addressed these two problems and then present the results of an interface optimization of our data.

Revising Interface Optimization

The original optimization technique searched through all possible subsets of operations to find solutions (i.e., small sets of operations that support large numbers of scenarios). This exhaustive search had a running time that grew exponentially with the number of operations (18 in the case of K-Sketch), and was inappropriate for the present domain (84 operations).

We designed a new optimization algorithm that uses two heuristics. The first takes advantage of the fact that optimal solutions of similar size tend to have many common operations. Using this heuristic, we assumed that it was good enough to search through sets of operations that are up to K operations removed from solution.

The second was a greedy heuristic that chooses operations to keep based on the total number of approaches they appear in. These heuristics were validated against the optimal K-Sketch data and results were very close to the K-Sketch results for small values of K.

To integrate speed into the analysis, generate multiple sets of solutions. The first assumes that all scenarios are completed using the fastest available approaches. Successive solution sets allow approaches that take longer to execute. We classify sets of solutions by the total time needed to complete the scenario: fastest time, 25% longer, 50%, 75%, 150%, 300%, 600% longer, and unlimited time. In this way, we could observe the change in solutions as slower approaches were used.

We ran the greedy heuristic optimization using K=7. The process took 2 hours to complete on eight 2.66 GHz Xeon cores and gave the results shown in Table 6. Figure 4 shows an overview of the data. Each line shows the simplest and most expressive solutions for a given speed. The lines converge near 300% slower than the fastest speed, which means that using alternative approaches will never add more than 300% to the total task time.

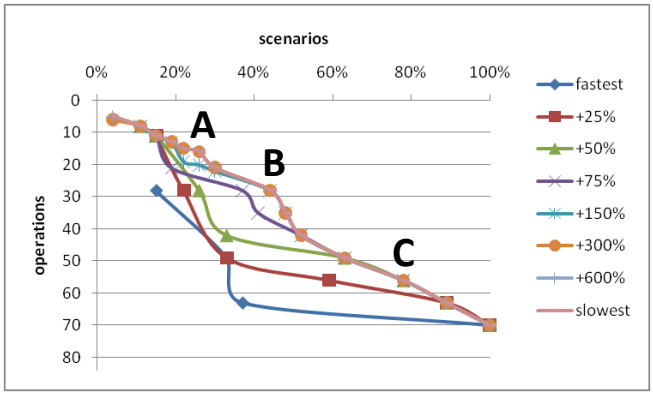

Figure 4: Minimum operation counts. Add operations (y-axis) to support more scenarios (x-axis).

Discussion

These results help us see this tradeoff between speed and simplicity. If we allow some scenarios to take up to 300% longer to perform, we can support more scenarios with fewer operations. However, requiring scenarios to take any longer would provide little benefit. For this reason, we generated Table 6 assuming that scenarios could take up to 300% longer than the fasted available time.

We identified three interesting points in the results, marked A, B, and C in Figure 4, and Table 6. To identify these spots we look for convex inflection points, which indicate fewer operations than average were required to support the number of scenarios at that point. At Point A, 7 scenarios (26%) are supported with 16 operations (19%). At Point B, 12 scenarios (44%) are supported with 28 operations (33%). At Point C, 21 scenarios (78%) are supported with 56 operations (66%).

Interestingly, Point A includes the set of operations that approximately make up the Scratch programming environment, minus input, pixel, and sound operations. In our encoding the "object instance variable" operation approximates named sprites in Scratch. Point A has one of the highest supported scenarios per operation value of any point; and it also is the sweet spot with the smallest number of operations. This may explain how skilled end-user programming systems designers, such as those who created Scratch, are able to intuitively find it.

The operations and results of this optimization span all interfaces, whether textual or informal. Each of Points A, B, and C represents an appreciable difference in the expressiveness and simplicity of such a tool for the given speed (300% of the fastest). In this respect, Points A, B, and C can be thought of as checkpoints to guide the development process of any game and simulation design tool. When tool designers can agree on goals for the speed, simplicity, and expressivity of their tools, they can focus on building interfaces to better match the mental models of a specific audience of users.

CONCLUSIONS

We have explored the processes and tools used in the early stages of game and simulation design. Interviews with four educators clarified the uses of simulations in the classroom, while interviews with three professional game designers uncovered a need for a new medium for prototyping game interaction. We also ran a study that observed seven groups of children designing games with words, sketches, and animations, finding significant advantages to sketches and animations. Finally, we refined the interface optimization design technique and applied it to this domain as a first step toward a new game and simulation prototyping tool.

REFERENCES

1. Agustin, M., *et al.* Game Sketching. In *Proc. the Second International Conference on Digital Interactive Media (DIMEA)* (2007).
2. Ambrosia Software Inc. Sketch Fighter 4000 Alpha. http://www.ambrosiasw.com/games/Sketchfighter/.
3. Bell, T., *et al.*, Technology-Enhanced Collaborative Inquiry Learning: Four Approaches under Common Aspects, in *Contributions from Science Education Research*, R. Pintó and D. Couso, Editors. Springer Netherlands, 2007. 451-463.
4. Buxton, B. *Sketching User Experiences: Getting the Design Right and the Right Design*. Morgan Kaufmann, 2007.
5. Čadež, B. Line Rider. http://linerider.com/en/node/365244.
6. Christian, W. and Belloni, M. *Physlet Physics: Interactive Illustrations, Explorations and Problems for Introductory Physics*. Prentice Hall, Upper Saddle River, NJ, 2004.
7. Concord Consortium Inc. Molecular Workbench. http://workbench.concord.org.
8. Csikszentmihalyi, M. *Flow : The Psychology of Optimal Experience*. First ed. Harper and Row, New York, 1990.
9. Davis, R.C., *et al.* K-Sketch: a "kinetic" sketch pad for novice animators. In *Proc. the SIGCHI Conference on Human Factors in Computing Systems*, ACM (2008), 413-422.
10. Design Simulation Technologies Inc. Interactive Physics. http://www.design-simulation.com/ip.
11. Electronic Arts Inc. The Sims. http://thesims.ea.com/.
12. Electronic Arts Inc. Spore.
13. Goodloe, K., Online: Games, *The Wall Street Journal Online*, December 9, 2006. Dow Jones and Company, Inc. http://online.wsj.com/article/SB116561569237944971.html.
14. International Game Developer's Association, Casual Games White Paper, 2006. http://www.igda.org/casual/IGDA_CasualGames_Whitepaper_2006.pdf.
15. ISEE Systems Inc. Stella. http://www.iseesystems.com/softwares/Education/StellaSoftware.aspx.
16. Jenkins, H. *Convergence Culture: Where Old and New Media Collide*. NYU Press, 2006.
17. Jenkins, H. Playing Politics in Alphaville, *Technology Review*, Issue, Number, May 7, (2004).
18. Kelleher, C., *et al.* Storytelling alice motivates middle school girls to learn computer programming. In *Proc. the SIGCHI Conference on Human Factors in Computing Systems*, ACM (2007), 1455-1464.
19. Landay, J.A. and Myers, B.A. Sketching interfaces: toward more human interface design. *IEEE Computer* 34,3 (2001), 56-64.
20. Linden Research Inc. Second Life.
21. Lundgren, S., *Joining Bits and Pieces – How to Make Entirely New Board Games Using Embedded Computer Technology*, Master's Thesis, Department of Computing Science, IT University of Göteborg, Göteborg, Sweden, 2002.

22. McDaniel, R., Demonstrating the Hidden Features that Make an Application Work, in *Your Wish is My Command: Programming by Example*, H. Lieberman, Editor. Morgan Kaufmann, San Francisco, CA, 2001. 161-174.
23. MIT Media Lab. Scratch. http://scratch.mit.edu/.
24. MIT Scheller Teacher Education Program. StarLogo TNG: The Next Generation. http://education.mit.edu/drupal/starlogo-tng.
25. National Instruments Corporation. LabVIEW. http://www.ni.com/labview/.
26. Oulasvirta, A., *et al.* Understanding contexts by being there: case studies in bodystorming. *Personal and Ubiquitous Computing* 7,2 (2003), 125-134.
27. Pane, J.F., *et al.* Using HCI Techniques to Design a More Usable Programming System. In *Proc. the IEEE Symposia on Human Centric Computing Languages and Environments* (2002), 198-206.
28. Peppler, K. and Kafai, Y.B. What videogame making can teach us about literacy and learning: alternative pathways into participatory culture. In *Proc. the Digital Games Research Association Conference* (2007), 369-376.
29. Purho, P. Crayon Physics. http://www.kloonigames.com/blog/games/crayon.
30. Purushotma, R. Commentary: you're not studying, you're just... *Language Learning and Technology* 9,1 (2005), 80-96.
31. Repenning, A. and Ioannidou, A. Agent-based end-user development. *Communications of the ACM* 47,9 (2004), 43-46.
32. Repenning, A. and Ioannidou, A. Broadening participation through scalable game design. *ACM SIGCSE Bulletin* 40,1 (2008), 305-309.
33. Smith, D.C., *et al.*, Novice Programming Comes of Age, in *Your Wish is My Command: Programming by Example*, H. Lieberman, Editor. Morgan Kaufmann, San Francisco: CA, 2001. 7-19.
34. Squeak. eToys. http://www.squeakland.org/.
35. Squire, K. Games, learning, and society: building a field. *Educational Technology* 4,5 (2007), 51-54.
36. The MathWorks Inc. Simulink. http://www.mathworks.com/products/simulink/.
37. Wolber, D., Pavlov: where PBD meets Macromedia's Director, in *Your Wish is My Command: Programming by Example*, H. Lieberman, Editor. Morgan Kaufmann, San Francisco: CA, 2001. 345-350.

#	CAT	OPERATION	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
1	seq	numeric operations or function calls																											
2	i/o	pick with mouse																											
3	state	scalar variable																											
4	flow	read or write state variable																											
5	cue	cue at regular intervals																											
6	play	create or destroy characters and motions																											
7	i/o	numeric text or slider controls																											
8	seq	iterate over selected group of objects																											
9	rand	pick scalar from range																											
10	play	transform motion every time step																											
11	i/o	press button/key																											
12	spatial	test for intersects																											
13	state	object instance variable																											
14	cue	cue when condition changes																											
15	spatial	get intersection points																											
16	play	swap characters or motions																											
17	play	control playback speed of motion																											
18	cnd-sp	choose objects that match color pattern																											
19	flow	clip a variable to a range of values																											
20	edit	re-apply motion to object																											
21	play	create character in a specific environment																											
22	visual	transform motion with numeric operations																											
23	i/o	get data from sensor																											
24	rand	pick object from set with replacement																											
25	play	play motion sequence conditionally																											
26	flow	test if variable is set or clear																											
27	i/o	make object follow mouse																											
28	play	test if at specific time in motion																											
29	edit	play motion multiple times																											
30	edit	conditionally allow editing while playing																											
31	spatial	make objects impede each other's motion																											
32	phys	allow objects to transfer momentum																											
33	spatial	get point on stroke																											
34	cnd-sp	choose objects that match condition sequence																											
35	spatial	test if inside object																											
36	rand	condition passes with variable probability																											
37	i/o	press mouse button																											
38	grid	add or remove object in grid																											
39	flow	variable directly controls visual property																											
40	draw	test if strokes are similar																											
41	i/o	get input from multiple sources																											
42	i/o	get gestures and misc. mouse input																											
43	grid	test if object's cell overlaps another object																											
44	grid	test if grid cells contain pattern																											
45	flow	get max/min of a set of variables																											
46	draw	make object trace a line																											
47	flow	get recent displacement of object																											
48	phys	apply force every time step																											
49	spatial	constrain motion to region																											
50	edit	get input mode as a variable																											
51	edit	erasing break strokes into pieces																											
52	i/o	drag with mouse																											
53	spatial	move object along an ink stroke																											
54	spatial	test if object is near another object or point																											
55	draw	group new drawings with an object																											
56	flow	increment variable at regular intervals																											
57	grid	get index at screen location																											
58	play	use motion as graphic or graphic as motion																											
59	sound	play sound																											
60	grid	slide group of grid cells recursively																											
61	spatial	move object toward another object																											
62	rand	pick scalar from a set without replacement																											
63	spatial	use ink strokes to divide space into regions																											
64	cnd-vis	show only objects that match conditions																											
65	edit	move with inverse kinematics																											
66	grid	swap two groups of grid cells																											
67	visual	stretch an object between two points																											
68	spatial	test if object is above or below another object																											
69	pixel	set pixel value																											
70	flow	use predefined variables for branching stories																											
71	grid	hide or show elements in a grid																											
72	sound	play sound faster or slower																											
73	phys	apply force from stroke																											
74	i/o	point with mouse																											
75	i/o	plot graphs of variables																											
76	edit	change stacking order of objects																											
77	flow	set variable sequence conditionally																											
78	edit	change object color																											
79	rand	pick scalar from set with replacement																											
80	flow	use special features for tracking scores																											
81	state	game state variables																											
82	camera	track or zoom in on object with camera																											
83	grid	slide group of grid cells non-recursively																											
84	seq	manage game stage transitions																											

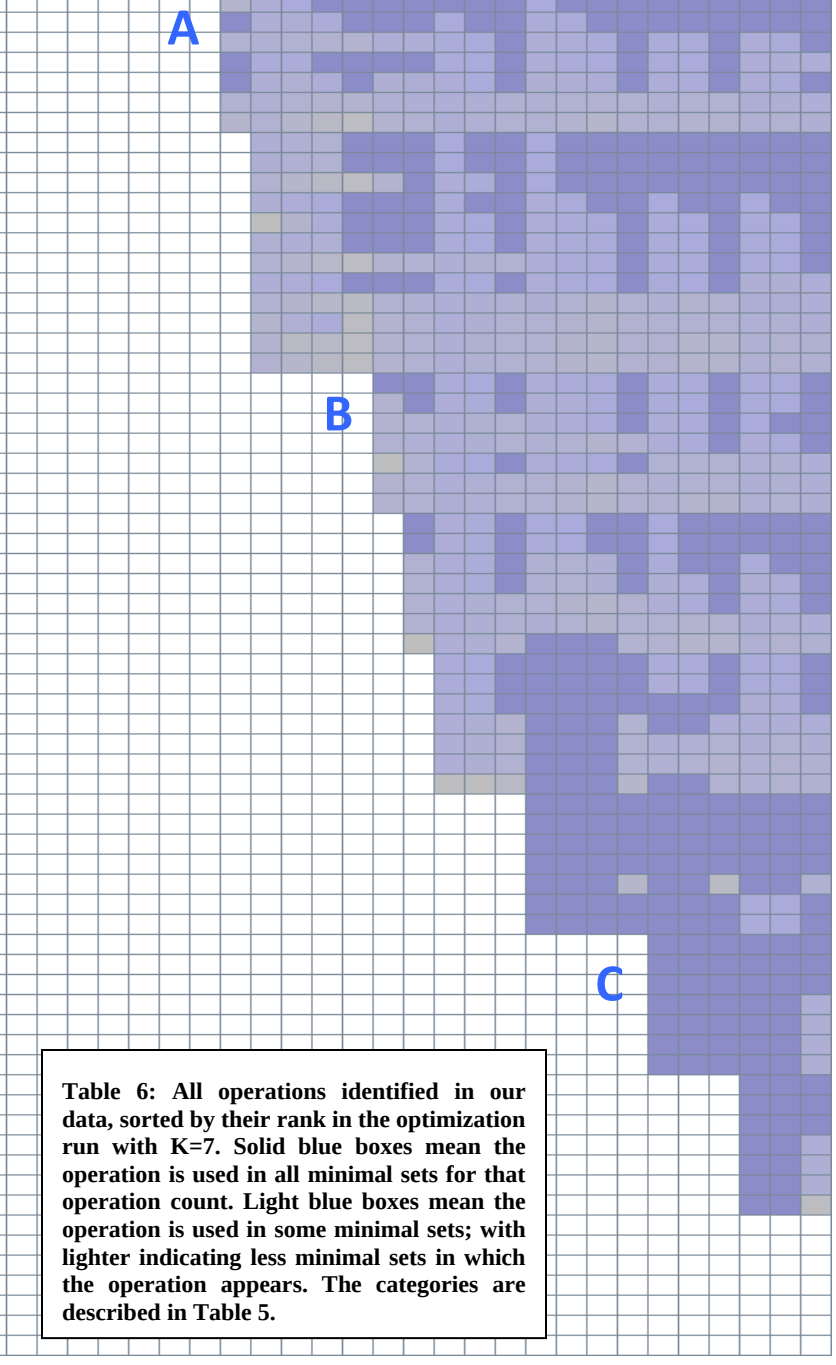

Table 6: All operations identified in our data, sorted by their rank in the optimization run with K=7. Solid blue boxes mean the operation is used in all minimal sets for that operation count. Light blue boxes mean the operation is used in some minimal sets; with lighter indicating less minimal sets in which the operation appears. The categories are described in Table 5.