Making Home Speech Therapy Fun
A case study in assistive games design

John M. Chambers
Dept. of Math and Computer Science
University of South Carolina Aiken
Aiken, South Carolina, USA
jmchambers90@gmail.com

Zachary Rubin, Sri Kurniawan
Dept. of Computer Engineering
Baskin School of Engineering
University of California, Santa Cruz
Santa Cruz, California, USA
zarubin@ucsc.edu, srikur@soe.ucsc.edu

Abstract—**This paper details and evaluates the process used to create an iOS game to motivate children who have had reparative surgery for cleft lip or cleft palate and who are subsequently participating in speech therapy to do their home therapy exercises.**

Keywords—cleft palate; assistive games; speech therapy

I. A NOTE ON CITATION

In the interest of completion in the time allotted, sources are often not cited. Instead, the most important ones have been listed at the end. Thus, uncited ideas are not necessarily original to the authors.

II. MOTIVATION

Across the world, children born with cleft lip or cleft palate struggle in their speech development, and the intensity of their struggle depends on which of the two defects they experience.

Those with cleft lip are typically eligible to have their defect repaired in the first week of life, and if it is indeed repaired, the children develop normally. Unfortunately, not all of these children have access to the surgery, so their speech develops to compensate for the defect.

In contrast, because of the growth patterns of the mouth, those with cleft palate are ineligible to have reparative surgery until they are two years old. Thus, their speech also develops to compensate.

In general, the longer one waits, the worse the condition becomes. As with learning to play a musical instrument, children progressively lose their capacity for language learning and speech development as they grow older.

This window between 0 and 2 years is especially important. It is during this period that children learn to form syllables and to combine them into words. In fact, the "babbling" that is often observed in children of this age is not mere play; it is crucial practice in syllable formation.

It is during this period that the defects begin having a negative effect. Children who suffer from the defects learn to produce their syllables as best they can. But because their anatomy is different, they cannot produce the sounds in the same way as normal children. The approximations they use instead are known as "compensatory structures."

After these children undergo reparative surgery, since they have already passed the normal period for learning to produce syllables, they do not easily adapt to their new anatomy. Thus, instead of naturally adopting normal pronunciations of which they are now physically capable, they usually keep using the compensatory alternatives with which they are already familiar.

Speech therapy is commonly employed to correct this problem. For the purposes of this paper, command and control is the specific therapy method used. Children usually spend between 1 and 2 hours a week with their therapists, and they are given exercises to work on at home between sessions. These exercises usually focus on a particular problematic vocal structure, such as plosives of glottal stops.

Unfortunately, these exercises are highly redundant and boring, so they are unable to keep the children's attention. As a result, the children often actively resist doing the exercises, which ultimately leads to them not getting done. Sadly, the longer they go without performing their exercises, the less their potential for full recovery becomes. Conversely, again following the musical instrument analogy, home practice between sessions can lead to significantly faster improvement.

Therefore, to encourage the children to practice and, thus, to guarantee a faster and better recovery, the researchers disguised the home exercises as a voice-controlled iOS (e.g. iTouch, iPhone) game. The target age group included children from 2 to 5 years of age.

III. PROCESS OVERVIEW

The entire process from initial training to study development took approximately 3 months. The first month was devoted to introducing the junior researcher to human-computer interaction, developmental psychology, and the iOS platform. The second month included requirements development; a study of popular recent voice-controlled,

This research was supported by NSF Grant #1156606.

iOS, and children's games; and practical training in Xcode (the IDE for iOS and other Apple OSs). The final month was devoted to storyboarding and writing, iOS game platforms research and training, graphical resource generation, game engine design and implementation, and study design and legal preparation.

IV. PREPARATION

The junior researcher surveyed HCI, developmental psychology, and iOS programming to prepare for the project.

A. HCI

For this fundamental area, chapters 1-6 in [1], as well as [2], were used and are recommended for the preparation of future researchers in similar projects. A number of helpful ideas are drawn from these sources.

First is *low-fidelity* (or "lo-fi") *prototyping*, which essentially involves the use of everyday office and art supplies such as index cards and stickers to develop and test new user interfaces. Its more advanced relative is high-fidelity (or "hi-fi"), which, in the case of computer interfaces, involves creating an *actual* computer program to demonstrate the proposed final app's look and feel.

Lo-fi has a number of advantages over hi-fi. First and most importantly, it allows flaws to be discovered and eliminated early. Flaws correction becomes increasingly difficult as the project gets farther along, so early flaw elimination prevents the unnecessary waste of effort. The other major advantage this method offers is preventing supervisors from thinking the project is farther along than it actually is.

Another helpful idea from these sources is the *interface metaphor*. These use or combine concepts with which the user is familiar to help them understand an application's purpose. Mac OS's Time Machine is a fine example. It is essentially just a backup and restore mechanism, but its metaphor suggests to its users that it can take the computer "back in time" to a previous state, such as before they got that virus. In essence, a *good* metaphor colors the user's perception of either how a program works or what it does. In this project, the best example of an interface metaphor is the "cue" described below.

A third idea is being entirely user-centered. It suggests that any interface should be adapted to the user and his/her everyday habits, perceptions, ideas, and expectations rather than forcing the user to adapt to it. Said differently, the interface designer must be careful to *always* understand and interpret the interface from the user's perspective and not from his/her own.

Consequently, it is imperative that users be involved in the design process as early and as much as possible. Interviews and surveys of the target user group are essential to gain a thorough understanding of how they think and work, which better informs the requirements generation and prototyping phases. And consistent user testing throughout the prototyping process ensures the product truly conforms to its users.

In the current project, the user-centered approach involves close contact with and study of the children and their parents and therapists. One example of this is the researchers' effort to understand the problem-solving potential of children such that the challenges they include in the game are neither too difficult to solve nor too simple to be entertaining.

A number of other concepts in the early chapters of [1] are also worth noting. In particular, the various interaction types (i.e. instructing, conversing, manipulating, and exploring) described in Chapter 2 are a good source of inspiration for thinking about several fundamentally different game designs. The disambiguation in that chapter between models, theories, and frameworks may also be of interest to the reader. However, in the interest of space, these are left for the reader's exploration.

B. Developmental Psychology

Study of this area helped to further enhance understanding of how game challenges should be designed. Chapters 1-5 and 9 in [3] were used, but the reader is advised to exclude Chapters 2-4 in his/her own study. Obviously, there are a number of important concepts specifically relevant to the project in this book.

1) *Chapter 1:* First are the implications of some of the most prominent historical models: the ethological, the psychosocial, the learning, and the cognitive-developmental. The ethological suggests simply that there are different kinds of learning at different ages. Erikson's psychosocial model says that each stage of life has a unique crisis that must be overcome for successful development in the long term. For *1 to 3-year-olds*, that crisis is realizing one's *autonomy and ability to make decisions*, as well as learning to deal with shame and doubt. For *3 to 6-year-olds*, this crisis is *learning to try new things and being able to handle failure*.

For the learning model, there are several different approaches and facets: conditioning, imitation, social cognitive, and self-efficacy. *Conditioning* is simply the idea that the consequences of one's behavior will determine future behavior, and it has both a negative form in which wrong actions are *punished* and a positive form in which good actions are *rewarded*. In contemporary America—and particularly in games for children—positive reinforcement seems to be the preferred conditioning method.

The second facet of learning, imitation, is the simply idea of "monkey see, monkey do." That is, it is the simple copying of another's behavior. The social-cognitive facet is simply a more constrained conception of imitation. In it, it is only the *rewarded and approved behaviors* (such as those of the smart, talented, and popular people) which are imitated.

Thus, in the game's design, it is important to consider how deeply to conform with, or to give the impression of conforming with, a certain society's (whether real or not) values. Though there is a strong contemporary current in favor of individualism and against caving to peer pressure, the simple fact is that offering approval works, so it remains a possible tool towards the end of teaching these children.

The final dimension of learning is self-efficacy—the idea that *experience* shapes children's beliefs about their own abilities. The implication for game design in the current context is two-fold. First of all, because these children have difficulty speaking correctly, they are unlikely to have faith in themselves to improve. Thus, it is imperative that the game allow for tangible improvements quickly to build up the children's confidence in themselves and consequent ability to improve.

The third historical developmental model is that of Piaget. The relevant component of his theory is his description of the *Preoperational stage*, which he says lasts from 2 to 6 years of age. He claims that during this period, children learn *symbols (i.e. words and numbers)* to represent aspects of the world, but they relate to the world only through their *own perspectives*. One tangible example of this latter idea given in [2] is that of 5-year-olds not being able to describe how a certain object looks from another person's perspective. Though these components alone are helpful in children's games design, an examination of the earlier and later stages of Piaget's theory serve to better inform the reader what children are *already* capable of in the target age group as well as what they are *not* yet capable of.

A few other ideas beyond the historical models in Chp. 1 are also of note. First, it must be noted that children cannot be thoroughly shaped by external influence, for they inevitably have influence on their *own* development. For the game, what this means is that the kids cannot be forced to like and understand whatever game design is used. Instead, the game design must adapt to *their* tastes (e.g. in game mechanics) and capacities for understanding. In many ways, this is simply a reiteration of the user-centered principle.

The second supplementary idea worth noting from the chapter is that, during studies, it is imperative that the researchers not allow their observation methods to corrupt reality. In other words, it is the *natural behavior* that is of interest and not the behavior that is distorted by the artificial environmental factors created by the observation. Thus, for example, it is preferable during studies to *not* remove children to a special room for observation and, instead, to observe them where they would otherwise *normally* be.

Essentially, what this means for the project is that studies must be crafted so as not to introduce any unnecessary conditions to the study environment which might inadvertently distort results. More specifically, it would be most preferable to observe a child's use of the game during *regular* practice sessions with their parents rather than at another and thus unusual time and place.

2) *Chapter 5:* A number of observations regarding sensory preference and ability, perception, and motor development are helpful to the project.

First, for auditory considerations, it is important to note that pitch range is limited in younger children, but that the range of human speech is usually safe. Also, as far as music is concerned, young children prefer *consonant* (happy-sounding) melodies, are able to recognize rhythmic structures, and will sometimes produce regular body movements along with their perceptions of those structures.

As for visual considerations, they enjoy looking at *patterns*, and prefer the colors red and blue over others.

Lastly in the sensory department, children pay more attention to *multi-modal and redundant* presentations of information than they do singular presentations. In fact, they even process it better. Thus, within the game, it is important to appeal to and engage as many senses as possible to ensure maximum attention and understanding.

There are also a number of important considerations in perception and attention. First and foremost, children prefer *new stimuli* over familiar ones, and in a phenomenon known as *habituation*, they usually tune out the redundant. Second, in general, they very much enjoy and are able to pay more attention to *movement*. Third, staying focused is typically a very demanding exercise. Fourth, they *don't adapt to new rules well*. And last, they find it difficult to filter out *distractions*; therefore, it is important to make *relevant information* more *obvious and salient*.

Finally, in the way of motor development, children typically need to master the *components* of a difficult motion before they can master the *whole*. These processes are known as differentiation and integration. For the game, the general implication is that, unless the difficulty is specifically part of the challenge, it is better to demonstrate or teach the components of a particularly difficult game mechanic or syllables of a word before requiring its full use.

3) *Chapter 9:* Possibly the most relevant among all the chapters is that on language and communication, and the notes from it are equally so.

First, speech development is a largely *imitation-based* process. Therefore, it is important that proper syllable pronunciations be demonstrated both well and frequently.

Second, with word teaching, it is important to visually demonstrate or point out the object being referred to. And, on a related note, it important to be sensitive to the *vocabulary sizes* at different ages.

Vocabulary development takes an interesting path. Before their first birthday, children begin to gesture, which demonstrates their ability to differentiate different objects. By 15 months, half of all objects are referred to by names rather than gestures, and between 2 and 3 words per week are being learned. Around 18 months (though it can be as early as 14 months and as late as 22 months), a "naming explosion" occurs during which the vocabulary rapidly expands with a particular emphasis in names of objects. Average vocabulary size at this age is 75, but it can range from less than 25 to more than 250. By 24 months, new words are being added every day, with a few hundred typically already known. And by 6 years, a typical vocabulary includes more than 10,000.

As an aside, it should be noted that this so-called naming explosion usually coincides with an expansion of cognitive ability—especially the ability to express *goals and intentions*.

A few semantic mistakes commonly made by 1 to 3-year-olds could become a source of misunderstanding and difficulty for them in playing the game. The first is underextension, which is defining a word too narrowly, and the second is overextension, which is defining a word too

broadly. Thus, children may understand any proper names of characters as referring to their entire class or *vice versa*.

Another factor to be aware of with word definition is *shape bias*. Because of this phenomenon, children tend to be better able to differentiate between and to group objects by class because of either different or similar shapes, respectively.

Finally, it is notable that 3- and 4-year-olds are much less likely to learn a word when the teacher (not necessarily academic) appears unfamiliar with the word's referent.

Factors possibly more notable than vocabulary size for game design are the ages at which children can understand *scale models* and *maps*. In general, 2½-year-olds are completely incapable of connecting scale models to their larger correspondents; however, as demonstrated by virtually any child at play, they are not limited from identifying larger and smaller members of the same class (such as real trains and small plastic ones). As far as map reading is concerned, this skill emerges naturally around 4 years age, but such children are only are only capable of reading relatively *simple* maps.

Syntactical structure and complexity are two more rather significant considerations for game script. In essence, linguistic complexity should match the ability of the age group, being neither so simple as to bore, nor so complex as to confuse.

Children's early sentences (around 1½ years of age) usually take one of several *two-word forms*: agent + action, possessor + possession, action + object, agent + object, action + location, entity + location, attribute + entity, and demonstrative + entity. As they progress to 2½ and 3 years of age, their sentences lengthen as they learn to use *grammatical morphemes* such as the word ending "-ing"; the articles "a," "an," and "the"; and auxiliary verbs like "am" or "should." Finally, between 3 and 6 years, children begin to understand negation and embedded sentences (i.e. dependent clauses). They also begin to understand passive voice, but it is best to avoid this if possible, as full understanding of passive voice does not develop until some time in the elementary years.

As a final note on game script construction, it is best to *avoid using figurative language*, as it often confuses younger children.

C. iOS Programming

A number of different resources were used to teach the junior researcher in the capabilities and use of the iOS platform. [12] served to illuminate the true possibilities of what could be done on the platform. [13] offered an overview of the components necessary for the proper functioning of an iOS app. And [11] provided the practical education necessary for making the junior researcher a capable iOS programmer.

For training of future researchers, it is recommended that only the first four classes of [11] be used, as they are all that is truly needed to establish functional familiarity with the platform. Though later classes may certainly be of assistance, it is important to avoid those that focus on UI creation, as they are aimed more at business-type interfaces that contain buttons, labels, tables, and text fields rather than game-type UIs. [12] and [13], while certainly helpful, are primarily auxiliary—not offering any information that's absolutely *central* to game design and programming. Thus, they should be avoided unless needed for the purposes described below.

2) *iOS Technology Overview:* As its name suggests, [12] provides a survey of the full capabilities of iOS out of the box. Discussion is included on which these features may be of interest to the game designer or programmer and on how they might be employed.

First, simply as a description of available tools, *Xcode* is the IDE for all things iOS and Mac OS, and *Instruments* is a handy performance analysis and debugging tool—though, it should be noted, the latter was never used in the development of the researchers' game.

Within the platform, there are a number of technologies which may at some point become useful in the game's development. The most prominent, *storyboarding*, is the visual design and connection of regular views (i.e. windows or UIs)—a feature which could potentially be used in *any* of the game's interfaces which use standard UI features like labels, buttons, text boxes, and toolbars. Another feature, *documents*, is a way of storing data in Apple iCloud and is meant as a model for text documents. However, it could be used to create multiple user profiles that can be shared among all the users' devices.

Multitasking is not *multithreading*, but is instead the feature of iOS that allows multiple apps to run simultaneously with one in the foreground and others in the background. Unfortunately, it is necessary for programmers to specifically define the behavior of their apps when running in or transitioning between these states. However, this definition need not be included until later in the development process when the code base is more established. For a more detailed discussion of how to prepare an app for these, see [13], pp. 33-91.

Printing and *file-sharing* are two features that may be useful for progress tracking and evaluation. Any iOS app can print wirelessly with the right infrastructure—a feature that could be used to print progress reports if desired. And file-sharing allows an app's Documents folder to be exposed in iTunes 9 and above—a feature that could be used to connect the game and a child's progress and profile with a desktop-based tracking, evaluation, or analysis app.

There are two possible notification technologies available. *App push notifications* can be sent at any time from a remote server. And *local notifications* can be issued by an app while it is running in the background state, or it can schedule a notification to go off at a specific time. The former could be used to notify end users of new content, and the latter could be used to remind them to do their exercises periodically.

Gesture recognizers can be used to detect complex finger movements such as pinches and swipes. If touch-based input is ever incorporated into the game, such supportive AI would be highly beneficial.

Another form of supported input that might at some point become helpful is the *accelerometer*, which measures the *motion* of the device itself.

One rather special component which greatly expands the potential for the game is the *Game Kit*. This framework allows peer-to-peer engagement over Bluetooth, in-game voice communication, turn-based matches with states stored in iCloud, and a Game Center including aliases, leaderboards, matchmaking, and achievements.

As an overview of iOS graphics technologies, Core Graphics is primarily for 2D-vector and image-based rendering, Core Animation is for animating *views* (not game animation), Core Image is for video and still frame display, Core Text is for text layout and rendering, and OpenGL ES and GLKit are for 2D and 3D rendering straight from the hardware. In general, OpenGL ES is better for apps requiring high frame rates (though it is more complex to program in), and Quartz (iOSs native drawing technology) is easier for object-oriented people.

For audio technologies, Media Player is used for playing tracks in iTunes; AV Foundation is for simple Objective-C playback and recording; OpenAL is for *positional* audio; Core MIDI is for playback of MIDI sound files (usually used for music); and Core Audio is for *vibration* and buffering and playback of multichannel local and streamed audio. Note that Core Audio is the *only* framework that allows for *haptic feedback*.

For speech-based input, there is unfortunately no native speech recognition technology other than *Siri*, which depends on a remote server. Because this presents a possibility for lag as well as a dependency on having an internet connection, it is not well-suited to this game's purposes.

As an aside, the speech-recognition technology used in place of Siri is the combination platform OpenEars and PocketSphinx. All speech-to-text translation in this platform is done against a programmer-defined local vocabulary, solving *both* of the problems inherent with Siri.

In the video department, Media Player can do either full- or partial-screen playback, and AV Foundation can also do playback as well as capture. Allowable video file types on iOS include *mov, mp4, m4v,* and *3gp*. These may be important if the game ever ends up employing *cut scenes*.

Broadening out a bit, as far as programming is concerned, automatic reference counting, block objects, and Grand Central Dispatch may prove useful.

Automatic Reference Counting (ARC) is a feature introduced in iOS 5 that reduces the amount of memory management one has to do (reducing the need for *retain* and *release*); a better explanation of its proper use can be obtained in [11]. Doubtlessly, employing this technology will reduce error and headache in the game's development.

Block Objects are somewhat strange constructs that allow instructions to be stored in a variable, passed around to different parts of the program as such, and executed later. An example and explanation of their use can be obtained in [13] on pp. 55-56.

Grand Central Dispatch (GCD) actually makes extensive use of block objects. Intended as a replacement for threads and to eliminate the great headache and overhead inherent in managing them, GCD proposes that all required tasks be broken up into discrete units and assigned to pools for execution by the system. These pools can either be queues with a first-in-first-out design, or general pools out of which any waiting task can be executed at random.

The aforementioned pools are all C-based (function-based) technologies, but GCD does provide an Objective-C-based portion that allows for more complex dependency description among individual instruction object units. A more detailed description of GCD can be found in [16]. Clearly, this replacement for multi-threading could serve the game well.

As far as *data* is concerned, iOS provides SQLite, Core Data, and rather elementary XML support. Regular files, of course, are stored in the application's bundle folder, but files in this folder cannot be modified after shipped; it is thus necessary to place any changing files in the app's Documents folder. SQLite, just as it sounds, is a lightweight SQL database technology that can run locally without the need for a separate remote server. The purpose and use of Core Data is somewhat unclear, but it appears to be tied to more business- and standard-views-based apps. Lastly, the nominal XML support offered by iOS is little more than a sequential file reader.

Unfortunately, tree-based or *"DOM-style" XML file reading is not available natively on iOS*, so third-party packages must be employed to accomplish the task. For the requirements of the game, *GDataXML* turned out to be the best such package.

The guide makes several parting observations and suggestions worthy of note. First of all, the *size of the screen* is very limited, so one must be sure to make all UI elements large enough to be visible and accurately invoked, and one must be sure to break up all complex interfaces into parts or sections.

Another factor to be aware of is *orientation changes*. Many iOS apps adapt to the orientation of the device as horizontal or vertical. However, this type of change is less common in games.

The device's *limited memory* must also be borne in mind—particularly for high-intensity games.

Finally, it is critical to *regularly test an app on the device* because the simulator is merely an approximation to the real thing and does not behave in precisely the same way.

2) *iOS App Programming Guide:* This resource ([13]), contrary to its name, is not absolutely essential to the construction of a functioning iOS application. Its contents apply largely to business-oriented application and to fine-tuning methods that only become relevant later in the development cycle. Its discussion of app resources (such as icons and launch images) is primarily of concern for release on the iTunes store. And its detailed prescription for defining app state transitions (such as how an app behaves when it is forced to move into the background, like when a call comes in) is necessary only when the app is closer to release.

3) *iTunes U Stanford iOS Class:* As has already been discussed, the first four classes of this resource ([11])

provides the best guided practice in using Xcode for iOS programming. While the other resources provide a great background, this is what actually gets someone ready to program with the platform and IDE.

V. REQUIREMENTS

In light of the above sources, a number of specific requirements were established for the game.

A. Voice-based Considerations

The centrality of the vocal component of this project gives rise to a number of very specific requirements in this category.

1) *Is there a reason for using voice control is used over other forms of input?* Touch is the primary form of input on iOS devices, so traditionally touch-based commands (such as left, right, up, down) should not arbitrarily be replaced with voice commands. Employment of natural language in a way not easily done with touch would meet this requirement. Examples include *commanding* a character or the device and *conversing* or communicating with a pet or personality.

2) *Are structures spoken as much as possible?* To encourage maximum practice, the game ought to ensure homework exercise vocal structures are spoken as much as possible. A variety of words using the same structure should probably be used to keep the child's attention.

4) *Does the system cause vocal fatigue?* Though maximum practice is desired, the system should not demand so much talking from the children that their vocal cords begin to hurt.

3) *Does the system recognize compensatory structure and promote correct ones?* The system's chief objective is to promote correct speech, but it is obvious that the children will have trouble doing so. Thus, the system ought to recognize when they *are* having prolonged difficulty and help them self-correct while still maintaining morale.

There are two known methods for providing self-correction assistance when necessary. First, a diagram of the physical form producing a correct pronunciation could be displayed, though it might not be well understood by those younger than 4. Second, the therapist could record the child in the therapy session saying the structure *correctly*, and this recording could be played back to the child. Since language development is a primarily imitation-based exercise, the latter is the more promising.

Another concern in this area is how to treat *partial correctness*—when the child's pronunciation is somewhere between the compensatory structure and the correct one. Four options are known. First, the requested task could be performed partially, providing some form of positive reinforcement. Second, the requested task could somehow backfire, providing negative reinforcement (which is likely not desirable). Third, the game could simply do nothing, enabling the child to try again. And finally, something that could be incorporated with any of the other three, the game could offer some form of feedback about how close they were to the right form (*i.e.* through colors, varied levels of verbal affirmation, *etc.*).

4) *Does the game make use of Cleft Lip Foundation recommendations for getting the children to improve?* These recommendations are quite simple. The game should *speak* to them, demonstrating the correct form for them to imitate. And it should *encourage* them to speak their best.

B. Child Development Considerations

Because this system is designed for children at such early stages in their development, it is impossible to safely ignore the differences with them that might not be so obvious.

1) *Is the problem presentation understandable?* Challenges in the game must be presented in such a way that the children can understand them. If they don't understand them, they certainly won't be able to solve them. In [4], it seemed that problems that adhered to *physical laws and goals* were typically better understood, but the introduction of abstract rules causes confusion.

2) *Is the problem salient?* Too much information on screen creates a distraction from what's *actually* important and needs to be solved, and children in this age range are particularly susceptible to such distraction. Nintendo games (such as Mario or Pokémon) might provide a good idea of what kind of *information density* is appropriate on smaller screens.

3) *Does the problem have too many steps?* Children *18 months* of age can remove a *single obstacle* to a physical goal (such as the cookie jar), and *6-year-olds* can solve up to *6-step problems in their heads* in a Towers of Hanoi game (see [4]). The target children ought to be bounded by these two extremes.

4) *Is the full age range addressed?* A substantial amount of developmental progress occurs from 2 to 5. Therefore, it is imperative that the game design grow with the child in order to keep his/her attention.

C. Making It Fun

1) *Do kids want to play it?* One of the main objectives of the project is to make home therapy exercises less tedious and tiresome by making them fun. Though children find a wide variety of thing interesting, they can also become bored rather quickly—especially by redundancy. In this game, motivational fun is absolute imperative.

2) *Is it big, colorful, loud, and animated?* Children are captivated by movement and comprehensive and intense sensory appeal. Thus, appealing to these likes will increase the fun factor.

3) *Does the voice processing delay or error negatively affect game play?* This requirement is double. First, because there is a 2 second delay between when the speaker finishes speaking and when the system can respond, time-critical challenges should be avoided. Second, if the system consistently misunderstands and misprocesses a child's command, the child will be almost guaranteed to get frustrated. Thus, ensuring processing fidelity should be a *high* priority.

4) *Does it have replay value?* Because these kids will be playing the same exercise multiple time a week, and this same overall game for as long as a their therapy takes (weeks, months, or even years), the game *must* be specially designed to hold their attention for the full period.

D. Technical Considerations

1) *Does the system allow for minimal level setup?* A limitation of staff and funding dictates that level setup not be made too complex. The roles of designer, writer, programmer, artist, and tester are difficult to accomplish within the limitations. Thus, a happy balance must be struck between simplicity of setup and prolonged attention-holding. Arcade games provide a nice means to this kind of objective.

2) *Does it respect screen size?* As has already been noted, screen space is very limited. Specifically, the iPhone 4S has a 3.5" screen, and the iPad has a 9.7" screen. In that kind of visual environment, it is imperative that text, images, and other UI elements be large enough to be decipherable. Yet again, Nintendo provides for a nice comparison with its *Gameboy* and *DS* series of gaming systems.

VI. INSPIRATION

In an effort to create a game that was fun for kids and made legitimate use of voice, it was useful to survey and profile some of the most popular games in both categories.

A. Voice-controlled Games

[24] provides a list of some of the most famous and/or popular voice controlled games, though it should be noted that other games have been included above and beyond this list.

1) *Seaman:* In this game, the player speaks to a pet, which is a strange species of fish. This "Seaman" is not always responsive, and the player often has to work persistently to get his attention. His responses change from innocent while he is young to rude and sarcastic as he gets older.

2) *Lifeline:* Here, the player is trapped and has no option but to guide a remote robot named *Rio* to get rid the monsters in a hotel and free him. All commands *must* be verbal.

This provides possible inspiration for the current game. The children could be required to verbally guide on-screen characters to accomplish certain tasks.

Also of note is how *Rio* reacts when she doesn't understand the player. She may shrug, do something completely unwanted, or simply do absolutely nothing. According to reviews, this behavior can be *very* frustrating when the listening system doesn't work well.

3) *Hey You, Pikachu!:* This Nintendo 64 game is similar to Seaman in that the player gets to raise his/her own pet: in this case, a wild Pikachu. Activities include a wealth of minigames, fishing, and purchasing items *for* the Pikachu.

There are several notable design features in this game. First of all, to make the speech processing delay more acceptable, the game encapsulates every utterance in a bubble and physically transmits it across the stage to Pikachu. This feature makes the delay more tolerable by giving it an apparent real reason for occurring—a feature that could be adapted for the current game.

In addition, the game uses icons in response to each utterance to indicate either understanding or the lack thereof. This is a form of guaranteed feedback not seen in the other games. While it does not guarantee the utterance will be properly processed, it consistently provides a base level of feedback indicating whether the utterance was even received.

4) *Odama:* In this game, the player issues vocal commands to battalions of troops to protect him or herself while simultaneously pushing a large ball across the field. Though this idea of simultaneously employing multiple modes of input to accomplish more is intriguing for the adult, it is unlikely that children could handle so many things at one time.

5) *Pah!*: This game is as simple with voice control as it gets. Essentially, the only command is "pah," and the system responds differently depending on the length of the utterance. The game itself is a side-scrolling space-like shoot-em-up, and long utterances are used to lift the ship while short utterances are used to shoot projectiles.

Though this simple mechanic definitely effects the repeated use of a particular syllable seemingly infinite times, it has a tendency to induce vocal fatigue after a while. Further, it has a time-critical component, which makes it unusable for the current project.

However, one of this game's great positives is its cross-cultural appeal. Whereas many vocal games are bound by linguistic barriers, this game was able to jump across countries without any modification rather quickly (see [25]).

B. iOS Children's Games

A number of lists around the internet provided recommendations of iOS games for kids at different ends of the required age range. Some of the most popular mechanics among these games were *physics-based games* (such as Angry Birds, Cut the Rope, Fruit Ninja, Paper Toss 2, Bubble Popper, and Rat on a Skateboard), *puzzle games*, *matching games* (such as AniMatch and Giraffe's Matching Zoo), and *creative games* (such as Toca hair salon, Cake Doodle, and Drawing Pad).[1]

C. Overanalysis Paralysis

After gathering much the above information, the junior researcher, who was in charge of choosing a game direction, was uncertain how to proceed. The requirements were almost too extensive to satisfy, and the possibilities for design choice were practically infinite.

The main issue was an attempt to use a deterministically narrowing strategy to identify the right direction when the

[1] Note that all games used in this analysis are downloaded and installed on ones of the UCSC ATLab iPod Touches for study by future researchers in that lab.

correct strategy at this point was to freely brainstorm. That is, to create.

Still stuck, the senior researchers eventually stepped in and decided to pursue an *interactive storybook* mechanic, using Scribblenauts as a chief point of reference. The Wheels on the Bus game might also provide a good point for comparison.

D. *The Dora the Explorer Breakthrough*

Continued brainstorming eventually led to the realization that a wildly successful model of the interactive storybook had already been implemented: *Dora the Explorer*. In this show, the story and animation capture the children viewers' attention, and it is periodically interrupted by opportunities for the viewers themselves to *verbally participate* in the show.

There are at least two different ways in which viewers can engage in these instances. First, they may identify the right direction, location, or object in response to Dora's questions. Secondly, when the villain fox Swiper attempts to steal something, Dora and the child must say, "Swiper, no swiping!" three times before he gets away in order to stop him. These two options provide potential models for the game, but the latter is particularly noteworthy. Because it gets kids to *repeat* the key phrase, it demonstrates a way in which homework exercise repetition might also be incorporated.

Figure 1. Map demonstrating the diverse locations Dora can visit in a single episode. [26]

There are other features in Dora the Explorer that are worth preserving in the researchers' game. First is *story flexibility*. Because of her age (7—older than most viewers), Dora is able to adventure anywhere from her neighbor's house to the Russian tundra. This kind of flexibility has allowed for the adventure diversity necessary for keeping children's attention for years on end—the same thing this game will have to do. Second, *each episode has a specific goal to achieve*, allowing for a sense of individual purpose and involvement every time. Third, Dora is always accompanied by her *monkey companion* Boots, who provides a greater sense of community. There is something warming about knowing that the character on the screen isn't completely alone without the player. Finally, a *map* is included to describe the plan of almost every episode. Though not critical for the vocal objective of the game, it could provide training and/or practice in map reading for the older children who are just beginning to understand it.

See [9] for more details on the design of Dora the Explorer. In particular, Larsen's detailed discussion of the process of creating an individual episode from start to finish could serve as a decent analogue for the current game.

VII. DESIGN PHASE

After gathering all the requirements, background information, and inspiration material—and with a general direction decided—it was time to begin actual design. Everything from mechanics to storyline were discussed, as detailed below.

A. *Initial Storyboarding*

Figure 2 and **Figure 3** make up the initial storyboard of the game. The first contains a bridge covered with balloons, and the player must say "pop a balloon" 3 times in order to succeed in the stage. This phrase specifically works on "plosives"—the p's and b in this case. After the third balloon is popped, *fireworks* are displayed as a form of *positive reinforcement*, celebrating their success in successfully completing the stage.

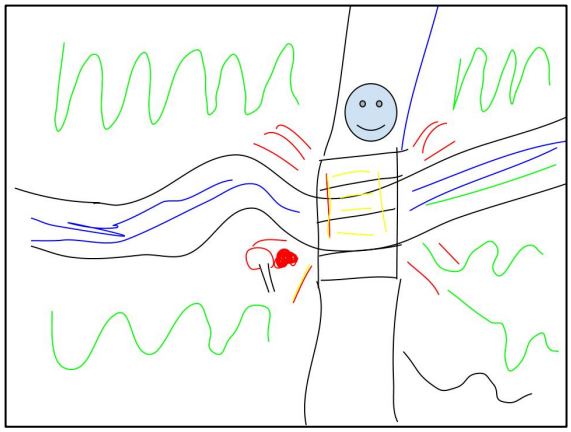

Figure 2. First rough storyboard frame.

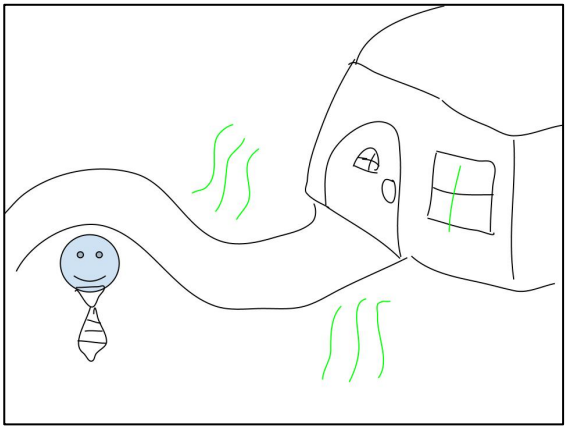

Figure 3. Second rough storyboard frame.

The second of these frames was far less fleshed out, but several words and phrases that were considered as completion commands included "red tie" and "open the door."

B. Responding to Requirements

Though a number of design decisions were made regarding how to meet the requirements, among the most important was a new *support system* to assist the children whenever they had trouble with a particular vocal structure. The system keeps count of how many times a particular word, phrase, or syllable (the particulars among these were not made clear) is mispronounced. After the third mispronunciation, the assistive diagram and audio sample of the child correctly saying the syllable are deployed. After three more time, if the child is still unable to properly produce the sound, the system automatically completes the task for him/her. One additional feature for this system that was not discussed involved keeping a record of the structures that prove particularly problematic (as determined by reaching the auto-completion mark) for review by the therapist in weekly sessions.

Another notable feature that was discussed was voice filtering. Because the children will likely perform the exercises with the assistance of their parents—and because their parents will probably demonstrate the proper pronunciation for them—it is important that only utterances from the child are processed as commands. A test of fundamental frequency was suggested as a way to exclude the naturally lower-range parental voices.

Another moderately important feature discussed was the termination of OpenEars listening after a certain length of time (e.g. 5 seconds). The logic was that if the child had not gotten the proper pronunciation after that period, s/he was not going to get it on that try. Thus, it is bad design to let OpenEars keep recording when the child may be making multiple separate utterance attempt.

C. Detailing the Story

Assisted by Katie A. Prior, junior researcher John Chambers brainstormed the overarching story belying the game.

The main character, Sam, is pictured in **Figure 4**. His age has not yet been set in stone, but he is at least old enough to be in elementary school.

The story successfully ties in the first two scenes, and it keeps a significant degree of the flexibility afforded by Dora the Explorer. By employing a particular plot device, this story enables use of such settings as a regular home and school, the Egyptian pyramids, Arthurian castles, Caribbean pirates, the Incan ruins, the Taj Mahal, the Canadian tundra, the Amazonian jungle, and even *space*!

D. An Excerpt

Getting into the meat of the story, for some reason, Sam's family has always been very close with the wizard Merlin—the very same Merlin from the Legend of King Author. In fact, he and Sam are known to have a relationship as close as any grandfather and grandson. On his way home from school, Sam regularly takes the long way home (see **Figure 5**) through the woods so he can spend some time with Merlin before finally going home for the evening.

On the day the player happens to meet Sam, he finds that the bridge he normally crosses on the way to Merlin's house is crowded out by balloons (**Figure 2**), and he needs the assistance of his newfound player friend to help him bellow the balloons into popping. (Note that the decision has not yet been made whether Sam will be aware of the player or not. First instinct suggests that having him aware would engage the players more. However, it would be helpful, if that route is chosen, to come up with a good way for Sam to realize the player is there.)

After crossing the bridge, knowing full well that those balloons were *not* supposed to be there, Sam realizes there is something unusual going on in this highland forest of redwoods. He makes his way on to the loving, but ever so kooky, Merlin's house filled with curiosity as to what is going on.

Merlin quickly informs him that *he* placed the balloons there as a harbinger of celebration. He has a present for Sam—a magical medallion which will carry him on adventures untold. But he *doesn't* tell Sam about its capabilities, instead charging him to *always* keep it with him no matter what. Sam, with great trust and respect for his elder friend, promises to do no less, though his curiosity can't help but fester. He finally makes his way home for the night after failing at every attempt to get Merlin to let secret out. And his eyes resist any semblance of sleep as he wonders like a child on Christmas Eve.

The next day, Sam rides the bus to school like any other day, but this day is *special*. It's one of those days that every student, no matter how old, inevitably looks forward to: a field trip day! Today's visit just happens to be to the Museum of World History, and Sam is raring to go.

On arrival, in an unusual decision, Sam's teacher tells him and his class that they can freely explore the museum to their delight. No lines. No groups. Just complete *freedom*!

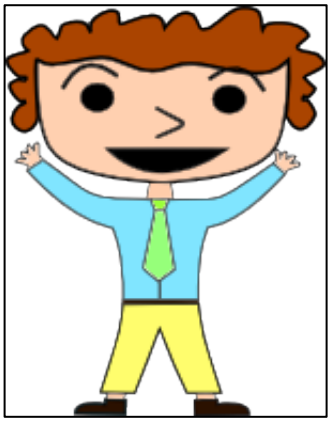

Figure 4. Sam, the main character.

Fascinated by *every* exhibit, Sam can't help but want to spend a day at them *all*. But he settles, after much internal deliberation, on exploring the section on Medieval Times, feudalism, castles, jesters and ladies, and King Arthur and his Round-table Knights. After all, that *was* the time that Merlin was always telling him stories about.

But as he enters the vast room filled with seemingly innumerable relics of the Middle Ages, he notices a very strange phenomenon occurring: a fading, a rising… It's the same room—with the same still artifacts. But not a one of his friends is there. Not his teacher. Not the museum staff.

Just as the gulp is reaching the top of his neck from uneasiness, he is suddenly jerked out of his tension—or maybe pleasantly distracted—by the new change that occurs. The relics, not budging out of place, seem almost to polish themselves—to reinvigorate—to be made new. Shining swords, armor, and jewels dazzle his eyes. And just as he completes his survey of the room, another but unexpected glint catches his eye… out of his pocket. That old medallion isn't so old anymore. In fact, it's so bright it's lighting the room.

Finally mustering the courage to see what might be going on outside, Sam timidly creeks the room's vaulted door open. But what he finds is not a museum… but a castle! And so begins his journey—and his realization.

This medallion could make him travel. But it wasn't like any old airplane. Given any simple relic from a place and time, he could go there. And, to be sure, this museum had quite the collection of relics.

Figure 5. The story in a nutshell.

VIII. PROTOTYPING

The above perspective, requirements, design direction, and story framework led to a series of efforts that become the partly finished end product. Because of the pressure of time, no lo-fi prototyping phase was employed.

A. Graphical Resources

For lack of an artist for hire, it was necessary to create the graphical resources independently. The free vector graphics package *Inkscape*—an alternative to Adobe Illustrator—served as the generation platform. **Figure 6** and **Figure 7** are examples of the use of this program.

It should be noted that vector graphics programs are preferable not only for their objectification of paths, but also for their scalability (especially by comparison with raster-based programs and formats).

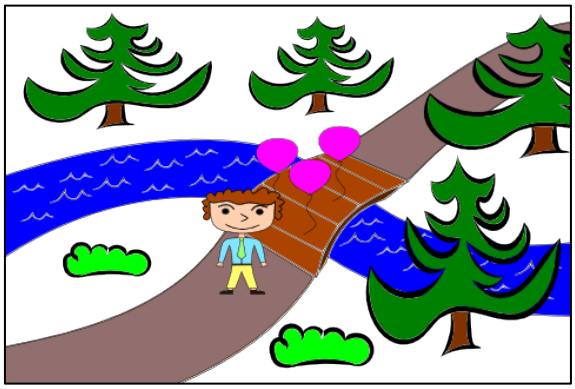

Figure 6. First storyboard frame brought to fruition.

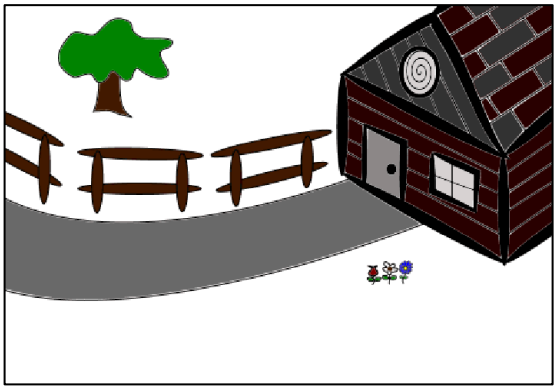

Figure 7. Second storyboard frame partly implemented.

B. iOS Game Engine Selection

From the perspective of the junior researcher, if a game engine can be found that satisfies the majority of the system's needs, it ought to be used to avoid reinventing the wheel.

Two of the most promising game engines in the iOS world at the time of writing included Cocos2D and Sparrow. According to reviews, the latter is easier to use, but the former allows for more control and is a more mature and,

thus, better documented platform. Further, the games that were currently on the market that made use of it looked *very* professional. Therefore, Cocos2D was selected as the core engine.

Unfortunately, because of design modification (in external level and behavior description, specifically) made later on, Cocos2D proved to be an insufficient platform.

C. Event-driven or Play-n-Prompt Model

Capitalizing on the Dora the Explorer model, the typical interactive storybook can be divided into two types of states. First is the *playback state* in which a portion of the story is being automatically played back, supported by pre-recorded narration. Second is the *prompt state* during which the player has been prompted with a challenge, and the system is awaiting a response. It is during this state that the above-described support system is employed.

This model allows all activity to be described sequentially and in temporal increments. In the context of Cocos2D, this saves the programmer from having to focus on frame-by-frame control in a Δt loop. The only top level changes that ever need be handled are executing a certain playback piece and waiting for or processing a vocal prompt.

Unfortunately, this approach can limit the implementation of animations and other activities that *span* these states. Thus, its implementation has a limit on usefulness, and must, at the very least, be supplemented by other engine features.

D. XML Stage Loader and Data Model

The stage depicted in **Figure 6** was fully implemented in code a few weeks before project's end. Subsequently generalizing level description for XML thus proved quite the challenge. A number of skillful data modeling choices were necessary to make the system work.

First of all, all graphics were assigned to *layers*. These layers are exactly comparable to layers in any graphic design program in which higher layers mask lower ones. The three conventional layers included the base stage, the activity layer, and the foreground. All movement takes place on the activity layer. The other two are reserved for stationary objects.

This leads to the two types of objects in the system. The first type is *scenery*, which serves the exact purpose it appears to. A scenery item has an associated stationary image (*i.e.* not sprite sheet), and it is assigned to a layer at a specified position. Over the course of the execution of a level or "stage," it never moves or disappears. It does not change.

The other type of object is the *actor*. An actor is much more complex, but, in general, it is any object that either moves or changes appearance. An actor can be assigned either a single image or a sprite sheet as its image source. Further, each still frame in the sprite sheet is named—usually according to the emotion it portrays. Multiple frame sets that are used for animation are also assigned a name for reference, along with parameters describing the frame delay, total number of frames, and optional sound associated with the animation—since, after all, animations would typically reflect actions that might produce sound. In fact, in the data model, these effects are called as "actions."

During the execution of the stage, actors' positions can be changed, and they can either be animated or set to display a different persistent still frame by referencing the names associated with either of these features. Thus, for example, the actor "Sam" could be set to move to a certain location using the animation "walkLeft."

A special type of actor was introduced to handle the balloons: *actor collections*. Like a class of objects, these actors are described once, but multiple instances of them can be placed on the stage. Further, there sometimes situations in which members of the collection must be referenced in sequence or all together. The balloons are a fine example because, when the "pop" action is performed, the system must iterate through the collection to find a balloon that is not yet popped.

This last example provides a nice transition into the idea of a cue. Drawn from the world of the theater—and specifically a stage manager—in theater show, every single occurrence from the dimming of the house lights at the beginning of the production to the countless position changes of the actors are documented as a cue. Position changes for actors are known are "blocking." And there are other cues for sound, light, and virtually everything else that goes on during the production.

In a word, the production can be fully described as a summation of sequential and sometimes simultaneous cues—with sound, movement or otherwise. This is exactly the metaphor used to describe the stages. Every bit of activity is described using these cues, except that, unlike in theater, the show is effectively "paused" at various locations to allow for the necessary player or user vocal input. Once the input has been received, the show picks up right where it left off.

Getting into the data model side, there are three types of queues. First is the simple *single* queue in which an actor or member of an actor collection can move, play a sound, modify its still frame, or play an animation over a specified period. Movements can be either Bezier curves or straight lines, reflecting the movement options in Cocos2D. Further, destination points in movements can be specified in either absolute coordinates (relative to the stage origin) or relative coordinates (relative to the actor).

Unfortunately, because of restrictions in Cocos2D's design, single cues cannot be created without being associated with an actor—a flaw that led to the failure of the particular implementation, but *not* of the data model.

The other two cue types are sequences and simultaneous cues. As their names imply, these are simply combinations of other cue types. Usually, sections of a stage script will have an overall sequence cue linking all of its component cues together. But within that sequence, there may be simultaneous cue collections, such as two actors moving at once.

Unfortunately, this abstraction of cues reflects the same problem of the event-driven game programming model: no cues can easily cross boundaries. Therefore, a *better* model

would be to specify individual cues to occur a certain amount of time from a *universal reference time*. In this way, the discrete boundary is eliminated.

Moving forward, next is the description of the level's *voice-to-text* system. Its components are the OpenEars vocabulary files and the definition of the commands it contains. The vocabulary files include a dictionary file and a language model file. The commands are essentially a complex connection between a text (converted from voice) command and a cue-based result. Its extra parameters include a specification of the sound file to be used by the support system if necessary as well as a *correctness threshold*.

This latter parameter is of particular importance and brings up one of the issues with using a more dynamic language backend. Different words and phrases have different correctness ratings that can be considered correct. For example, while "left" will usually return 0 (*i.e.* perfect) when said correctly, "shoot a balloon" won't return a value over -500 (*i.e.* less than perfect). The implication is that correctness thresholds may have to be set for each and every complete phrase; and, what's worse, correctness ratings may not be universal across different speakers, though this is a subject for empirical testing.

Returning to command description, a special type of cue was introduced to accommodate its needs: the *actor multiplicity type* of *plural one at a time*. This cue parameter was necessary, again, to accommodate the balloon situation. Whenever the "pop a balloon" command is issued, the system triggers the plural-one-at-a-time cue associated with it. The way the system handles this cue type is it seeks out all members of a plural actor class, seeks through these members until it finds one on which its assigned action has not yet been performed. When it finds such a member, it performs the assigned action on it.

The way memory of action performance is maintained is through the use of state assignments associated with actions. For example, when the "pop" action is performed on one the "balloon" objects, that balloon object is assigned the state "popped." Thus, the next time the command is called, the system will recognize that the state effect the action has has already been effected, and it will not effect it again.

The next to last component of the data model is the reward condition specification. This is the condition checked after the performance of any command to see if the requirements for stage completion have been met. It is defined using actor states; that is, while checking the reward condition, the system checks all specified actors against their associated specified required states.

The final component of the data model is the specification of initial locations for all actors. It is in this section that plural actor instances are assigned their respective locations. It should be noted that support has not been implemented for plural actors with a sprite sheet image source; only single frame plural actors are allowed.

IX. TESTING

The junior researcher's objective was to execute an observational case study at the UCSC Child Care Center by summer's end to evaluate normal kids' level of interest in the game. Unfortunately, there was not enough time to get the study off the ground; and, besides, the game never quite made it to the point of being worth executing a study with real kids. However, two very good things products came out of the efforts to establish the study.

A. IRB Protocol, Consent and Assent Forms

An annotated copy of the UCSC *IRB Protocol* guidelines ([23]) has been left in the ATLab for assistance in creating later studies. In general, studies with children in this category of research can be evaluated on an expedited basis, needing only the approval of the IRB Chair. Unfortunately, because these studies involve children, they cannot be entirely exempt from review.

In general, there are a few fundamental guidelines for the construction of protocols, consent, and assent forms (or assent methods). First and foremost, voluntary and informed consent must be elicited from the parent(s) before *any* attempt to obtain voluntary assent from the child may be made. Second, performance of the study is strictly dependent on continued consent. If consent or assent is withdrawn at any point during the study, all employment and observation of the subject *must* stop. If conditions surrounding the study change at any point, the subject *must* be informed so that he/she may evaluate his/her continued participation. Finally, all private information, such as medical information, *must* be kept securely so. However, immediate destruction of gathered data is *not* encouraged because such data may become useful later on. Protection of such data is the primary concern.

Finally for the project's specific protocol[2], the junior researcher included specific procedures for the care of those vulnerable to *seizures*. Because some games have been thought to induce seizures, those with a history of epilepsy are screened from the study. Secondly, the protocol requires that the researcher familiarize him or herself with the symptoms and treatment of seizures such that, in the event the game under considerations ends up provoking seizure symptoms, all gameplay will immediately cease and desist; emergency personnel will be notified; and known first aid will be applied.

B. Notes from the UCSC Child Care Center Director

Contact was made with the director of the UCSC daycare for the purpose of conducting the study. Among the recommendations offered, some of the most important were those affecting time.

In general, such studies are typically planned at least 6 months in advance. This early planning is critical for a number of reasons. First, it allows the daycare to adapt its curriculum to the study's needs. Further on that point, it allows the teachers to be and feel more involved in the

[2] UCSC IRB Protocol #1907, approved for 3 years from 8/28/2012.

process instead of having the study's procedures forced on them; and the more they feel a part of it, the more they'll support it.

Second, such extensive advance time allows the parents to consider and return the study forms, which can often take quite a while. Unfortunately, it is commonly the case that these forms must be actively chased down. However, because the parents are in an academic setting, they are typically better predispositioned towards allowing their children to be involved in such studies.

Third, the time allows for a better study environment. More specifically, under the laws of the State of California, unless an individual has undergone the rigorous background check associated with child care (which takes about 3 months to go through), that individual must be accompanied by a chaperone approved daycare staff member for the duration of the study. This is both a burden on the study conditions and a financial burden on the daycare center, as they must pay for an extra teacher to chaperone during the study. The director expressed a willingness to support the study with no cost to the researchers; however, earlier planning can prevent this inconvenience.

Finally, it is helpful to spend approximately 3 days getting to know the kids and allowing them to get used to the researchers before actually performing the study. This allows for better flow during observation, and it prevents the lack of familiarity from distorting observations.

X. RETROSPECTIVE EVALUATION

A number of lessons have been learned in review the process above described.

First of all, it took far too long to move from requirements generation to prototyping. Again, the main roadblock was an unintentional insistence on using a deterministic approach to design, and this insistence took a large chunk out of the time that should have been used for development.

Second, involvement with and incorporation of the "command-and-control" speech therapy method was far too undeveloped—particularly with the central focus of the project being on assisting and enhancing this process. The main roadblock to that component of the project was difficulty locating relevant literature; relevant databases were difficult to identify, and keyword searches in the databases available did not provide any promising content. In general, for researchers so unfamiliar with the field, it would have been helpful to have greater assistance in locating relevant resources.

Third, too much time was spent exploring iOS instructional materials. Though it was difficult to know without first exploring them, many of the resources were barely relevant to the project—at least for *this* stage of it. In the future, the above-included survey of iOS technologies is recommended for the guidance of those unfamiliar with the platform.

Finally, if there is to be any hope of actually performing a study, the study must be prepared and initiated *well* in advance of the program. However, being forced to write a study from scratch was likely as educational for the junior researcher as actually participating in one.

XI. FUTURE WORK

Clearly, in review of the requirements, much work remains to be done. In particular, a greater exploration of the cleft lip speech therapy process and collaborative (with therapists and parents) generation of a promising revised therapy method would be beneficial. Further, a greater examination needs be done into what kinds of problems children are capable of solving at various ages, as well as into what kinds of game mechanics are preferred in each age group. Finally, the game ought to be adapted for languages other than English and, possibly, for cultures beyond the West.

Moving out of the overarching requirements and on to the more meticulous implementation level, a number of substantial problems need to be remedied.

First of all, as has already been mentioned, the nature of OpenEars *correct ratings* needs to be more deeply described. Specifically, the researchers need to test if there is a generalizable rule governing the expected correctness rating across different words, syllables, and phrases of particular lengths. Such a rule would eliminate the need to establish an individual correctness rating for every single phrase or recognizable utterance in the system.

The researchers must also test the variation of correctness ratings of the same phrase among different people. If there is indeed variation among different individuals, an exploration must be should into how the system might be calibrated for different voices. For example, certain parameters of the voice may give rise to a mathematical relationship between the correctness ratings for one voice and those of another. Different accents should also be considered.

Next, a method must be devised for the elimination of erroneous responses. Specifically, when words or phrases not included in the current vocabulary are processed, the system will often return false positives. These may be able to be eliminated by checking correctness ratings; but, as has already been made clear, those have their own problems.

Finally, a system ought to be developed for the confident detection of absolute mispronunciations, as well as for mispronunciations that are somewhere along the spectrum between the known compensatory structure and the known correct pronunciation. One possible way is to retrieve the top n guesses for a particular utterance, search for the appearance of both the compensatory and correct versions, and compare the respective correctness values of each. Some combination of screening using established exclusion values (or baseline values) and generating a ratio between the values ought to provide a believable placement of an utterance actually spoken along the spectrum between compensatory and correct.

ACKNOWLEDGMENTS

John Chambers wishes to thank Katie A. Prior for her invaluable help brainstorming the game storyline. He would

also like to thank Alexandra Hollaway for her help with paper-writing strategies. Finally, he would like to thank the participants, staff, and advisors of UCSC's SURF-IT summer program for providing the daily input, feedback, and moral support necessary to the successful completion of this kind of project.

REFERENCES

[1] Y. Rogers *et al.*, *INTERACTION DESIGN: beyond human-computer interaction*. Chichester, West Sussex, UK: John Wiley & Sons Ltd, 2011.

[2] M. Rettig. "Prototyping for tiny fingers," in *Magazine Communication of the ACM*, vol. 37, no. 4, pp. 21-27, Apr., 1994.

[3] R. V. Kail. *Children and their Development*. Upper Saddle River, NJ: Pearson Education, Inc., 2010.

[4] D. Klahr and M. Robinson, "Formal Assessment of Problem-Solving and Planning Processes in Preschool Children," in *Cognitive Psychology*, vol. 13, no. 1, pp. 113-148, Jan., 1981.

[5] UC Davis Cancer Center and UC Davis Children's Hospital. (2012, July). "Children learn through play" [Online]. Available: http://www.ucdmc.ucdavis.edu/CANCER/pedresource/pedres_docs/ChildrenLearnThruPlay.pdf

[6] "How Children Solve Problems," *Scholastic* [Online]. Available: http://www.scholastic.com/browse/subarticle.jsp?id=3434

[7] "Cleft Palate Speech Glossary," *Children's Hospitals and Clinics of Minnesota* [Online]. Available: http://www.childrensmn.org/Manuals/PFS/ChildDev/193474.pdf

[8] "Speech Development Related to Cleft Palate," *Children's Hospitals and Clinics of Minnessota* [Online]. Available: http://www.childrensmn.org/Manuals/PFS/ChildDev/193473.pdf

[9] B. K. Larsen. "Ultimate Guide to Dora the Explorer," *TLC Family* [Online]. Available: http://tlc.howstuffworks.com/family/how-dora-the-explorer-works.htm

[10] Wikipedia contributors, "Video game genres," *Wikipedia, The Free Encyclopedia* [Online]. Available: http://en.wikipedia.org/w/index.php?title=Video_game_genres&oldid=517793515

[11] P. Hegarty. (2011). *iPad and iPhone Application Development* [Online]. Available: https://itunes.apple.com/itunes-u/ipad-iphone-application-development/id473757255

[12] Apple Inc. (2012). *iOS Technology Overview* [Online]. Available: http://developer.apple.com/library/ios/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/iPhoneOSTechOverview.pdf

[13] Apple Inc. (2012). *iOS App Programming Guide* [Online]. Available: http://developer.apple.com/library/ios/documentation/iphone/conceptual/iphoneosprogrammingguide/iphoneappprogrammingguide.pdf

[14] Apple Inc. (2011). *The Objective-C Programming Language* [Online]. Available: https://developer.apple.com/library/mac/documentation/Cocoa/Conceptual/ObjectiveC/ObjC.pdf

[15] Apple Inc. (2011). *View Programming Guide for iOS* [Online]. Available: http://developer.apple.com/library/ios/documentation/WindowsViews/Conceptual/ViewPG_iPhoneOS/ViewPG_iPhoneOS.pdf

[16] Apple Inc. (2012). *Concurrency Programming Guide* [Online]. Available: http://developer.apple.com/library/mac/documentation/General/Conceptual/ConcurrencyProgrammingGuide/ConcurrencyProgrammingGuide.pdf

[17] (2012). "Welcome to OpenEars: free speech recognition and speech synthesis for the iPhone," *Politepix* [Online]. Available: http://www.politepix.com/openears/

[18] R. Wenderlich. (2010, February 12). "How to Make a Simple iPhone Game with Cocos2D Tutorial," *RayWenderlich* [Online]. Available: http://www.raywenderlich.com/352/how-to-make-a-simple-iphone-game-with-cocos2d-tutorial

[19] R. Wenderlich. (2010, June 22). "How to Use Animations and Sprite Sheets in Cocos2D," *RayWenderlich* [Online]. Available: http://www.raywenderlich.com/1271/how-to-use-animations-and-sprite-sheets-in-cocos2d

[20] (2012, February 28). "cocos2d Basic Concepts," *cocos2d for iPhone* [Online]. Available: http://www.cocos2d-iphone.org/wiki/doku.php/prog_guide:basic_concepts

[21] (2011, March 30). "Programming Guide," *cocos2d for iPhone* [Online]. Available: http://www.cocos2d-iphone.org/wiki/doku.php/prog_guide:index

[22] R. Wenderlich. (2010, March 18). "How To Read and Write XML Documents with GDataXML," *RayWenderlich* [Online]. Available: http://www.raywenderlich.com/725/how-to-read-and-write-xml-documents-with-gdataxml

[23] University of California, Santa Cruz. (2007, May). *Human Subject Research Guidelines* [Online]. Available: http://officeofresearch.ucsc.edu/orca/irb/irb-forms/misc-forms/guidelines.pdf

[24] Leon the Hart. (2012, February 22). "The Top 5 Voice-controlled Games," *HardcoreGamer* [Online]. Available: http://www.hardcoregamer.com/2012/02/22/the-top-5-voice-controlled-games/

[25] P. Sawers. (2011, November 12). "The story of 'Pah!', the voice-controlled game that took the mobile world by storm," *TNW Magazine* [Online]. Available: http://thenextweb.com/apps/2011/11/12/the-story-of-pah-the-voice-controlled-game-that-took-the-mobile-world-by-storm/

[26] "Dora the Explorer: Lost City Adventure," *MobyGames* [Online]. Available: http://www.mobygames.com/game/windows/dora-the-explorer-lost-city-adventure/screenshots/gameShotId,257510/