Mac's Fiesta:
A Foreign Language Game
for the Sifteo Platform

by

Karina Cordon

Senior Project

COMPUTER ENGINEERING DEPARTMENT

California Polytechnic State University

San Luis Obispo

June 2012

Table of Contents

List of Figures

List of Tables

I. Acknowledgements

I would like to thank Dr. Oliver for being my project advisor and for the continued encouragement to be creative throughout the process. I would also like to thank Dr. Jipsen, Dr. Danielson and the students in their Child Development classes, whose input and feedback helped shape the project.

This project would not have succeeded without the help of the following people. A special thanks goes to Sean Voisen, who has been a valuable resource for Sifteo game development. Thank you to George Merida for lending his voice to the game. Thanks to Ritchie Ly and Sven Le for their help with sound engineering. I would also like to acknowledge Patricia Jiminez, whose wonderful artwork, done with great thought and care, is featured in the game.

Lastly, I would like to thank Dereck Quock, Kerynne Tejada, Marie Aromin, Angelo Cordon, and Marian Cordon for all of their help and support throughout the development of this project.

II. Abstract

Mac's Fiesta is a Spanish learning game designed on the Sifteo gaming platform for children ages four and up. The Sifteo gaming platform provides a new way of interacting with computers by giving users a set of tangible and interactive cubes. Each Sifteo cube has a clickable colored screen and can sense motion and adjacent blocks. Games on this platform are designed to encourage the development of core thinking skills. This project aims to explore how Sifteo cubes can be used for language learning as well as observe how effective they are as educational tools.

III. Introduction

As the American population continues to grow more culturally diverse, it is also becoming more imperative for people to be aware of other cultures. In order to increase communication among diverse communities, it is highly encouraged for people to learn a foreign language. In our educational system, learning a foreign language is usually required during high school and is sometimes offered as an elective during middle school. But learning a new language is also effective when started at a younger age, while children are developing new skills. Results can be seen through studies of the effects of watching popular TV series *Dora the Explorer* or *Ni Hao Kailan*, which encourages children to learn Spanish or Mandarin.

In today's world, people can easily spend hours playing video games. Contrary to popular belief, these hours are not spent wasted. Games researcher Jane McGonial says that games are motivational and encouraging, "making players feel that they can achieve everything" (1). Video games enhance perceptual and motor skills, as well as promote reasoning and critical thinking (1). The number of people who regularly play video games is expected to grow. New technology is continually being developed to make video games more accessible. Sifteo is working on one of these new technologies.

Sifteo cubes are a new gaming platform that encourages hands-on learning. Users are able to physically interact with digital data by arranging, shaking, flipping, tilting, and

neighboring cubes. Sifteo games are designed to be fun and engaging, while at the same time, encouraging the exercise of a thinking skill. Since they are small and easy to handle, Sifteo cubes are great for children. They do not require children to know how to type on a keyboard or handle a mouse. Most of the games currently available on the Sifteo game store help develop Math or Spelling skills. Using this platform, I believe that the game areas could be expanded to include foreign language learning games.

Game Proposal

For this project, I proposed a Spanish learning game titled Mac's Fiesta. In Mac's Fiesta, the player is able to help a monkey character, named Mac, gather bananas for an upcoming party for his friends and family. The player helps gather bananas by correctly identifying the Spanish words associated with the given image. The more words matched, the more bananas they get!

At the start of the game, the player will be taken to a menu screen. From here, they will be able to choose from different categories of words they want to learn. Categories will include numbers, colors, food, family members, animals, and clothing. They will also be able to select a mode of play—either scored mode or explore mode. In scored mode, the player gets two minutes to play and gain points. At the end of a session, they are given a score. In explore mode, the player will be able to play as much as they want. It is in this mode that players will be able to take the time to learn the Spanish vocabulary. Players will be able to do this by tilting two cubes, one for categories and one for play mode.

Once the mode and category are selected, the game starts. One of the cubes will have a picture and the other cubes will have randomly selected Spanish words displayed, one of which matches with the given picture. The player must identify the correct word and match it to the picture by placing the two cubes next to each other. If the player gets it right, the picture will animate or give some indication that they were able to match it. If the player gets it wrong, a short sound will play and the player will be allowed to try again.

If the player needs a hint, they can press down a word cube and a picture of what the word represents will temporarily show up on the cube and the computer will say the word aloud. If the player presses the picture cube, the computer will also say the word aloud. Shaking the cubes will skip the current picture/word match and go on to the next question. Flipping all of the cubes takes the player back to the menu.

IV. Background

Sifteo Cubes

Sifteo cubes is the product that evolved from a working prototype technology called Siftables, developed by David Merrill and Jeevan Kalanithi at the MIT Media Lab. The idea behind Siftables, thus Sifteo cubes, is to allow users to interact with digital information and media in a tangible way. After leaving the Media Lab, Merrill and Kalanithi founded Sifteo, Inc. to manufacture and market Sifteo cubes.

Each Sifteo cube is only 1.5-inches wide but has a lot of functionality. It has a full-color clickable LCD screen that also acts as a button. In each cube, there is a 3-axis accelerometer that allows the cube to sense tilting, shaking, and flipping. There is also near field object sensing technology that allows a cube to sense an adjacent cube. The user is able to interact with the system by pressing, tilting, shaking, flipping, or rearranging the cubes. The following is a list of components inside a Sifteo cube:

- 32-bit ARM CPU
- 128 x 128 pixel color TFT LCD
- 3-axis accelerometer
- 8MB Flash
- Lithium Polymer rechargeable battery
- 2.4 GHz wireless radio
- Proprietary near field object sensing technology

Most games are played with three to six cubes. The cubes hold up to 4 hours of play on a single charge and connect to a computer wirelessly through a USB radio link. In order to play games, users must install a desktop software application called SiftRunner. Since Sifteo cubes need a computer nearby to play games, the computer must meet the following requirements:

Windows

- 2.0 GHz Intel Pentium 4 or faster processor
- Windows XP SP3 with 512 MB of RAM or
- Vista/Windows 7 with 1 GB of RAM

Mac

- 1.5 GHz or faster Intel Core processor
- Leopard 10.5 or Snow Leopard 10.6 with 1 GB of RAM

General

- 1024 x 768 or larger display
- Available USB 2.0 port
- 200 MB disk space (500 MB recommended)
- Internet connection (for software download and setup)

Intelligent Play

Sifteo games are inspired by activities such as chess, tangrams, and crossword puzzles. They aim to be fun, engaging, and encourage the exercise of one or more of the following thinking skills (4):

- *Spatial Reasoning*

  Understanding relationships between objects in terms of distance and orientation.
- *Logic and Computation*

  Building awareness of associations between numbers.
- *Language and Literacy*

  Comprehending and using words, story elements, and literary devices.

- *Cooperation and Collaboration*

  Building and refining social skills to solve problems with other people.
- *Expression and Emotion*

  Exploring and understanding emotions and how they are expressed.
- *Strategy and Planning*

  Exercises high-level thinking and problem solving skills.
- *Creativity and Design*

  Building and creating different compositions.
- *Patterns and Perception*

  Exercises ability to spot trends and patterns.

Figure 1 is a diagram showing different game genres that Sifteo games currently cover. Mac's Fiesta covers three different genres. The first is the Word game genre, which is described to be games for people that love language. Because Mac's Fiesta aims to teach Spanish, it also falls under the Learning game category. Puzzle games test your memory and thinking abilities. Mac's Fiesta also falls under this category since it tests the player's memory and abilities to match the correct word to the picture.

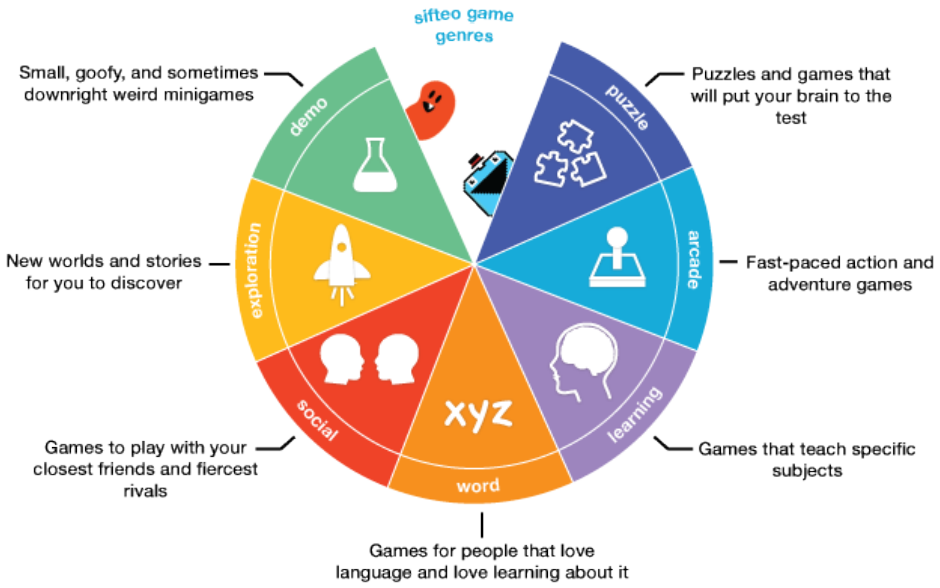

Figure 1: Sifteo Game Genres (4)

Developing for Sifteo

Developing a Sifteo game application is possible with the help of the recently released Sifteo SDK, available for download on the Sifteo website. The SDK features a lightweight C# API and includes four demo games with sample code. In addition to the minimum system requirements, as listed above, developers also need Mono runtime. Mono is an open source, cross-platform C# and .NET environment that supports Mac OS X and Windows. This project uses the Mac OS X version of Mono. Directions for development environment set up in Mac OS X are included in the appendices [*].

V. Requirements

The game must adhere to the following requirements and specifications.

Game Requirements

1. The Sifteo game should aim to have an educational value. (See Learning Standards below.)
2. The game must be easy and intuitive to play.
3. The game should be fun and engaging to encourage users to play more and learn more.
4. The game must be aimed towards children of ages 4 and up.
5. It should have a responsive and fluid user interface.
6. Set up should be easy and set up time should be minimal.
7. Because the game is targeted for children, graphics and/or animations should be attractive and colorful.
8. For the same reason as above, sounds in the game should be pleasant and not alarming for children.

Learning Standards

Mac's Fiesta is designed with the World Language Content Standards for California Public Schools (K-12) in mind (7). The game addresses Stage I of the Content standards listed below.

Stage I

1.0 Students acquire information, recognize distinctive viewpoints, and further their knowledge of other disciplines

1.1 Students address discrete elements of daily life, including:

a. Greetings and introductions

b. Family and friends

c. Pets

d. Home and neighborhood

e. Celebrations, holidays, and rites of passage

f. Calendar, seasons, and weather

g. Leisure, hobbies and activities, songs, toys and games, sports

h. Vacations and travel, maps, destinations, and geography

i. School, classroom, schedules, subjects, numbers, time, directions

j. Important dates in the target culture

k. Jobs

l. Food, meals, restaurants

m. Shopping, clothes, colors, and sizes

n. Parts of the body, illness

o. Technology

Functional Requirements

1. The Sifteo game should take advantage of and incorporate Sifteo cube functionalities. These may include: use of full-color LCD screen, sounds, button press, tilting, flipping, shaking, and adjacent placement of cubes.
2. The game should be responsive to button presses, tilting, flipping, shaking, or adjacent placement of cubes.
3. Graphics and text displayed on the screens should be clear.
4. Words displayed on the screens must be spelled correctly and be grammatically correct.
5. Word-image pairs must correctly match.
6. Audio for word pronunciations and/or game sounds must be clear and audible.
7. Audio for word pronunciations must correctly match with the word or image displayed on the associated cube.

Criteria

1. The system is limited to use up to six cubes at a time.
2. Sifteo cubes must be wirelessly connected to a computer.
3. Maximum range for wireless connection is 20 feet.
4. The cubes only sense each other when two sides touching and do not sense stacking.
5. There is only 8 MB of flash memory included in the cubes.

VI. Design

System Architecture

Top Layer System Design

Figure 2 shows blackbox diagram of the Sifteo system from the user's point of view. The Sifteo cubes connect to a computer via Bluetooth. After downloading the game from the computer to cubes, the user will mainly be interacting with the Sifteo cubes to play.

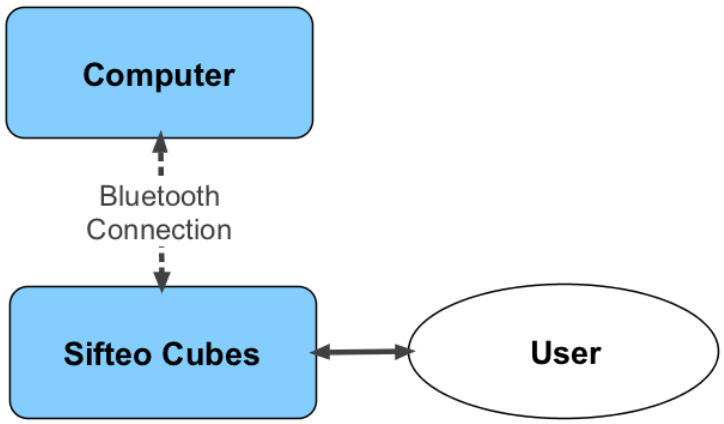

Figure 2: Top Layer System Design

Detailed System Design

Figure 3 shows a detailed diagram of how the user, game application, and Sifteo cubes interface with each other. Since the game application is not distributed through the Sifteo store, the user will need MonoDevelop and SiftDev to play. The computer will run these programs. MonoDevelop builds and runs the game application using the Sifteo API and .NET Framework. SiftDev communicates with MonoDevelop and will download the game onto the Sifteo cubes. The Sifteo cube has a 3-axis accelerometer, button, proprietary near-field communication technology, and an LCD screen. It sends and receives information through a Bluetooth connection. The user can manipulate the information by pressing the Sifteo cube button, tilting, flipping, or shaking the cube, or placing the cube next to another cube. User output includes image display on the cubes and audio from the computer.

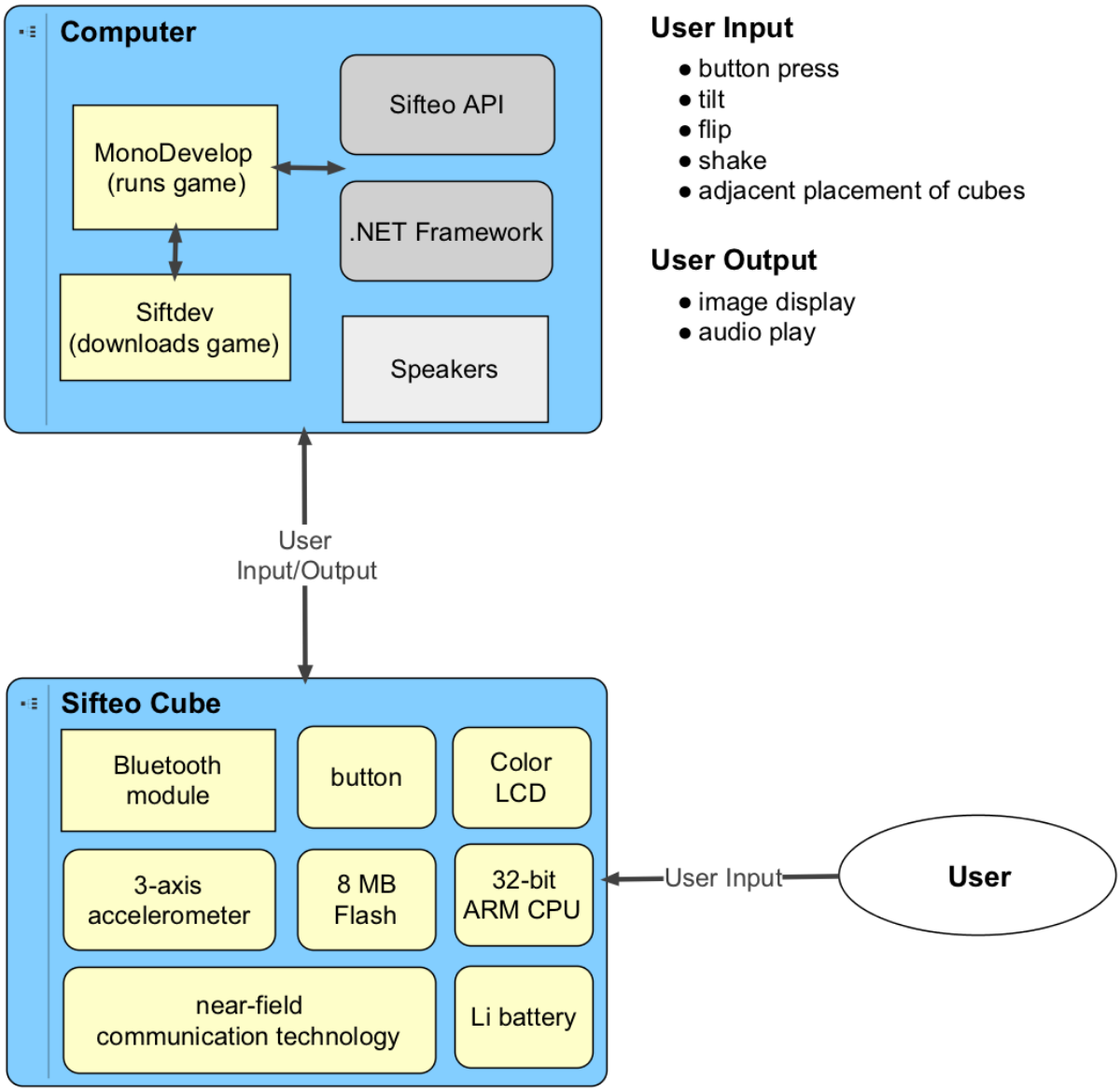

Figure 3: More Detailed System Design

Software Architecture

User Interface Design

Figure 4 is a software flow diagram of the game. It represents the states that the user will navigate through within the game. A directional arrow may represent a user input needed in order to progress to another state. The states that are colored and outlined with solid lines are states that were implemented. States with a dashed outline are states that are not implemented but were a part of the original proposal.

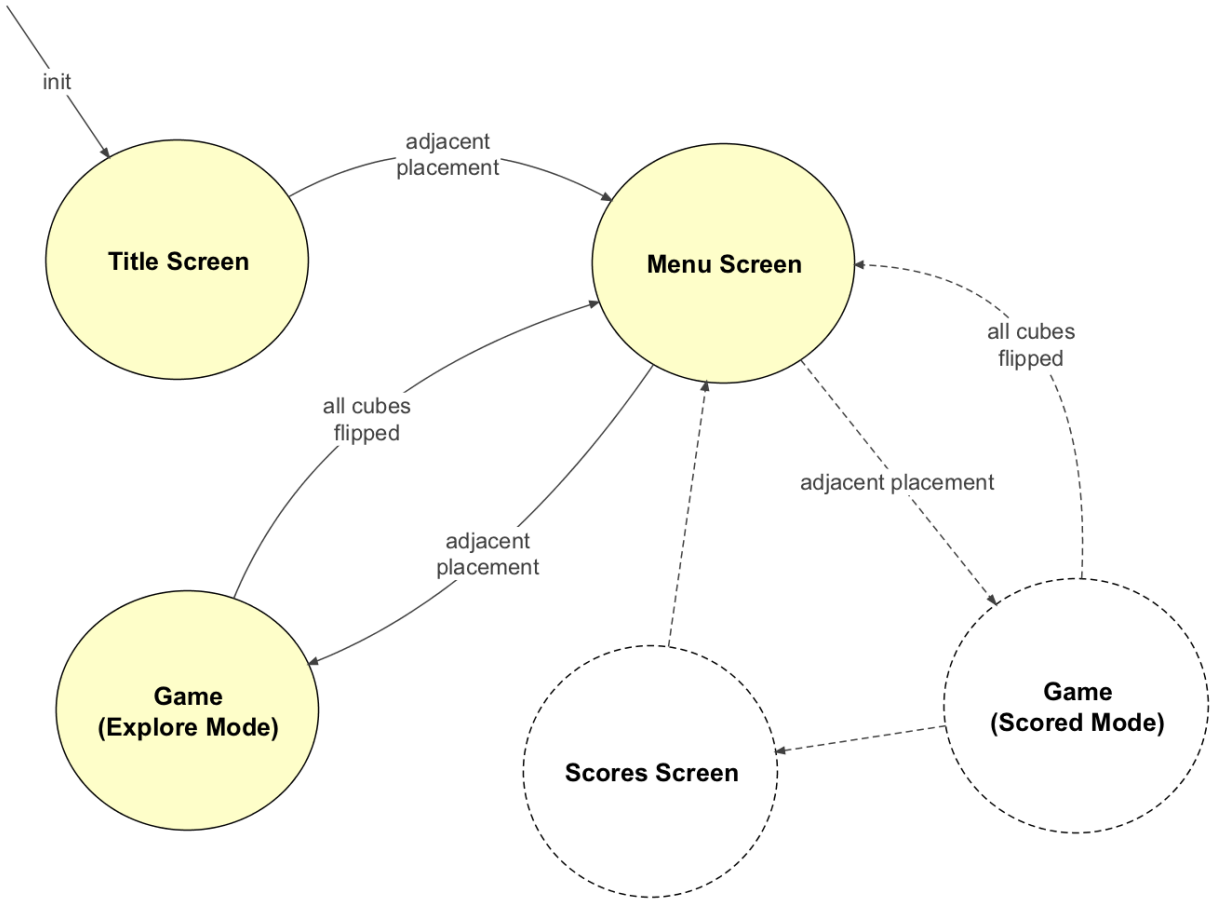

Figure 4: User Interface Diagram

XML Schema

The game will need an XML file to keep track of the list of Spanish words. The diagram below shows the hierarchy of the XML file. It has been extended to include the words in English as well. With this scheme, the file may also be extended to include the words in other languages.

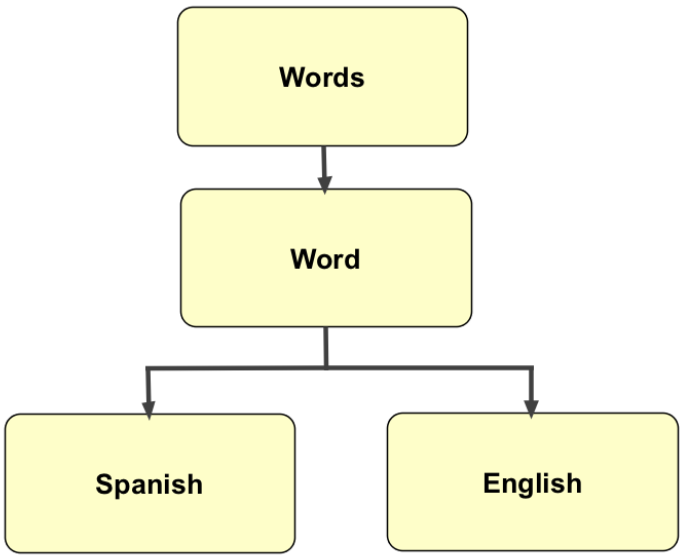

Figure 5: XML Word Repository Hierarchy Tree

VII. Development

Implementation for this project was broken down to different steps and was done incrementally. The first step was to explore the SDK and understand how the game application will output data to and receive input form the Sifteo cubes. At the end of this step, a basic prototype of the game was made. The second step was to configure the basic game flow and build the Title Screen. The third step consisted of creating the Menu Screen. Due to the poor performance seen at this stage during testing, the Menu Screen was redesigned during this stage. The fourth step was to implement a session of the game. At the end of this stage, a category of words was built and ready to be played. This stage proved to be the most time consuming. The last step of implementation was to add sounds to the game to make it more engaging.

Basic Prototype

Since Sifteo only recently released their Software Development Kit, there are not a lot of resources and tutorials available. So the first stage of development was to explore the SDK and try to understand how it works.

The goals of the first prototype were to get images to show up on the cubes and to recognize when two cubes were placed next to each other. With the help of a tutorial written by Sean Voisen, "Up and running with the Sifteo SDK," (6) I learned how to display images on the screen. Getting the application to recognize two adjacent cubes took a little more work, but there was sample code included with the SDK that helped me understand how the cubes communicated with each other. After finding the sample code, it was just a matter of manipulating it so that the images displayed would change only if the two cubes were the correct word-image pair. Figure 6 shows a simulation of the basic prototype of the game.

Figure 6: Basic game prototype

MVC Architecture

To implement the game, MVC architecture was used. Using this design pattern is helpful because it separates the application into three parts that could be implemented separately. Model represents the data to be used by the game. For this project, the model has the application reading from a master XML file of the Spanish words and gathering information from it. View takes care of what is being displayed on the cubes and Controller updates the View and Model based on user actions.

State Machines and Controllers

The Sifteo API gives developers support to implement state machines, found under `Sifteo.Util.StateMachine`. This allows developers to configure and implement a complicated game control-flow in a safe and modular way. With this added support, it is simply a matter of implementing our user interface diagram as a state machine!

The game application can only be in one state at a time. A state is added to the state machine through the function call `StateMachine.State()` which takes in the name of the state and a controller class. The controller class must implement functions in the `Sifteo.Util.IStateController` interface. A transition from state to state is defined through the function call `StateMachine.Transition()` which takes in the name of the state it's exiting, a transition id, and the name of the state it's moving to.

The following is sample code for implementing a State Machine in the main application class:

```
override public void Setup {
      StateMachine sm = new StateMachine();
      sm.State("Title", titleController);
      sm.State("Menu", menuControler);
      sm.State("Game", gameController);

      sm.Transition("Null", "NullToTitle", "Title");
      sm.Transition("Title", "TitleToMenu", "Menu");
      sm.Transition("Menu", "MenuToGame", "Game");
      sm.Transition("Game", "GameToMenu", "Menu");

      sm.SetState("Title", "NullToTitle");
}
```

Set the first state with a call to `StateMachine.SetState()`. Before moving onto the next state, queue a transition by calling `StateMachine.QueueTransition()`. The state machine will then do this transition with a call to a `StateMachine.OnTick()`.

Graphics

Each cube allows for images of 128x128 pixels. Before they are displayed on the cube, they must be converted to a Sifteo image file using the Image Helper tool found in SiftDev. The Image Helper supports GIF and PNG images.

Because the Image Helper tool only converts images one at a time, it is impractical to have multiple files of 128x128 px each. For example, the images for words are combined into one large file and are then converted all at once. In order to have the correct image show up on a cube, we need to know the starting x- and y- pixel coordinates, the width, and the height of the specific image based on the larger image and create a `SpriteData`. The same was done with the images for text. For the purpose of this project, the images are saved in PNG format. The PNG format allows certain parts of the image to be transparent, which is useful if we want to overlay images on top of one another. Figure 7 is an example of an image file used in the game and features artwork done by Patricia Jiminez.

Figure 7: Image file for Spanish words

Sounds

The Spanish and English word pronunciations are the voice recordings of George Merida. The recording was done using a Blue Microphones Snowball USB microphone. After recording, the sound file was split up into individual sound files of each word being spoken and saved in WAV format. Other sounds used in the game are also in WAV format.

A `Sifteo.Sound` is created by calling `Sifteo.SoundSet.CreateSound()` along with the name of the audio file (excluding the file extension). It can then be played

with a certain volume and a number of loops, stopped, paused, or resumed. The following is sample code in the main application of how a sound is played:

```
Sound bgMusic = Sounds.CreateSound("background");
bgMusic.Play(1, -1);       /* 1 is max volume,
                              -1 for continuous loop */
```

Other sound effects were added to the game for entertainment and to make the game more engaging. These sound effects were found on Freesound.org. While the user is at the title and menu screens, a fun Latin-style music song plays as background music. This sound clip was chosen because it is reminiscent of the Dora the Explorer theme song and seemed appropriate for the game. Besides the background music, two other sound clips are used. One of the sound clips was a short clicking sound, which doubled as a sound effect for selecting a category in the menu and a sound effect to indicate an incorrect match. This sound is nice for children as it is not alarming or discouraging if the match was not made correctly. The other sound is a short bell sound to indicate a correct match. Because the sound is pleasant, it excites children and encourages them to keep playing.

Data

Rather than a simple list of words, the XML file contains information regarding the images that will be displayed on the cubes and the sound files associated with it. The following is sample code of how a word and its information is stored:

```
<word category="Animals" image="words" x="0" y="512"
width="128" height="128">
      <sp image="animals" x="108" y="0" width="109" height="38"
sound="dog-s">el perro</sp>
      <en image="animals" x="0" y="0" width="50" height="44"
sound="dog-e">dog</en>
</word>
```

Word contains attributes for the category, important for the game application since a game session will be based on this detail. It also contains attributes for the image name, the starting x- and y- pixel coordinates, and the image width and height. Word has children nodes “sp” and “en” which both contain information for the text image and the name of the sound file. The information for the images is used to create a `Sifteo.Util.SpriteData` class.

VIII. Testing

Function Testing

Testing for functionality was done periodically throughout game development.

Title Screen

Figure 8: Mac's Fiesta Title Screen

Figure 8 shows the title screen displayed on the Sifteo cubes. The title screen was tested to ensure the test cases in Table 1.

Table 1: Title Screen Test Cases

Test Category	Test Case and Expected Result
Display	Images show correctly on the screen (i.e. centered, not skewed, colors match with what is expected, etc.)
	Unused cubes should be blank and do not display anything.
Audio	Background music should play when the game is started.
	Background music does not stop when exiting state.
Functionality	Title cubes should be placed in the correct order before moving to the next state.

Menu Screen

The original design for the menu screen was first developed and tilting seemed to work well on the simulator. When the game was transferred onto the cubes, I found that the display was not very fluid and responsive to tilting. Due to time constraints, I changed the menu design to be simpler and decided to remove play modes as well. Figure 9 shows the final design for the menu screen.

Figure 9: Mac's Fiesta Menu Screen

The menu screen was tested to ensure the following cases in Table 2.

Table 2: Menu Screen Test Cases

Test Category	Test Case and Expected Result
Display	Images show correctly on the screen (i.e. centered, not skewed, colors match with what is expected, etc.)
	Unused cubes should be blank and do not display anything.
Audio	Background music should be playing whenever the game enters this state.
	Background music should stop upon exiting this state.
	A short clicking sound should play when a category is selected.
Functionality	Sides other than the arrow side of the "Select a Category" cube should not have any functionality.
	Blank side of a category cube should not have any functionality.
Data	The category chosen should agree with the category stored in the game model.

Game Session

The game includes the following sets of vocabulary words, taken from a Kindergarten Spanish curriculum (2).

Numbers		**Food**		**Animals**	
0	cero	water	el agua	dog	el perro
1	uno	milk	la leche	cat	el gato
2	dos	juice	el jugo	fish	el pez
3	tres	pizza	la pizza	cow	la cava
4	cuatro	chicken	el pollo	pig	el cerdo
5	cinco	tacos	los tacos	monkey	el mono
6	seis	ice cream	el helado	bird	el ave
7	siete	apple	la manzana	duck	el pato
8	ocho	banana	el plátano	horse	el caballo
9	nueve	strawberry	la fresa	sheep	la oveja
10	diez	orange	la naranja		
		egg	el huevo		

Colors		**Family**		**Clothing**	
red	rojo	dad	el papá	shirt	la camisa
orange	anaranjado	mom	la mamá	pants	los pantalones
yellow	amarillo	grandfather	el abuelo	skirt	la falda
green	verde	grandmother	la abuela	dress	el vestido
blue	azul	brother	el hermano	shoes	los zapatos
purple	morado	sister	la hermana	socks	los calcetines
pink	rosa			jacket	la chaqueta
black	negro			hat	el sombrero
white	blanco				

Figures 10 and 11 show a game session. A picture shows up on one of the cubes representing one of the randomly selected words shown on the rest of the cubes.

Figure 10: Mac's Fiesta Sample Game Session

Figure 11: Mac's Fiesta Sample Correct Match

The game was tested to ensure the following cases in Table 3.

Table 3: Game Session Test Cases

Test Category	Test Case and Expected Result
Display	Images show correctly on the screen (i.e. centered, not skewed, colors match with what is expected, etc.)
	Only one picture cube is displayed.
	The rest of the cubes should display Spanish text, one of which must match the picture displayed.
	There should not be any duplicate Spanish text displayed.
	The English text should be displayed when a cube is pressed.
	Correctly matched cubes should be highlighted with a yellow border.
Audio	Background music should not play during the game.
	The Spanish word pronunciation should be played when a cube is shaken or when a correct match is made.
	The English word pronunciation should be played when a cube is pressed.
	A short clicking sound should play if an incorrect match is made.
	A short bell sound should play if a correct match is made.

Functionality	A new image-word set should show up after the Spanish pronunciation of the previous word has finished playing.
	Flipping all of the cubes should return the game back to the menu.
Data	The image or word shown on a cube should agree with what is stored in the game model.

Feedback from CD 413

CD 413 is a Child Development course titled Children, Adolescents and Technology. Since the course explores different types of educational tools for children, it was appropriate to visit the class and get feedback for this project.

The feedback received from members of the class was very positive. Members of the class mentioned that having both a visual and an audio cue was good for word association. They also said that the repetition of words showing up was good. A lot of members really liked the kinesthetic aspect of the Sifteo cubes as well. The only downside they were able to come up with was that young children of Kindergarten age might throw the cubes around.

The game was still in development and only supported Spanish when it was demonstrated to the class. When I asked for suggestions for improvement, they said that it would be good to add English support as well. The game would also benefit English learners. Luckily, the English pronunciations were already recorded and I was able to add this feature in the final version of the game.

User Study Observations

Unfortunately, there was little time left to do a comprehensive user study. Instead, I was able to demo the game to a group of friends and took notes on how they played. A few of them had prior knowledge of Spanish while others only knew a few or no Spanish words at all.

Most players were able to pick up the point of the game and how to play with the Sifteo cubes easily. A lot of them were excited when an event happened with a button press or placing the cubes next to each other. The bell sound associated with a correct word-image match proved to be a positive addition and encouraged players to keep playing. I was delighted to find that players were using deduction to choose the correct word. When played individually, I found that players would often repeat the pronunciation of a word after the word is said. When played in small groups, players had fun by splitting up the cubes and helping each other find the matches.

Although most players had fun, I noticed that the engagement level of the game was different for various players. Some were excited to be playing with the cubes, while others got bored after a few minutes. For those that already knew Spanish, they wanted the vocabulary words to get increasingly challenging. At the same time, since there is only a small set of Spanish words on the cube, they would quickly see and learn the same words over and over again.

IX. Conclusion

Overall, I believe that Mac's Fiesta turned out to be a successful project. Due to time constraints, the final game product ended up being a simpler version of the proposed idea for the game. Despite this, Mac's Fiesta is still fairly complete and playable. It also included more functionality with English support, which was not originally a part of the plan. The game met the minimum requirements and received a great amount of positive feedback.

Suggested Improvements

- *Include an instructions screen*

  This is mainly to explain special user options (i.e. flipping the cubes to go back to the menu, shaking the cube for Spanish pronunciation, pressing the cube for English, etc.).
- *Find a better way to display text*

  For this project, images were created to display each word. This takes up limited

memory in the cubes. Because images are downloaded to the cubes, this causes the game set up time to be extremely slow.

- *Extend the number of vocabulary words*
  As explained in the Testing section above, players quickly grew tired of seeing the same vocabulary words being shown.
- *Give players different levels of difficulty*
  For the same reason as above, this is to keep players engaged and wanting to learn more. An option would be to make the categories different levels and have the player correctly identify all of the words before moving on to another category.
- *Add interactivity with a monitor screen*
  Having a monitor screen would encourage more interaction with technology. We would also be able to learn how Sifteo cubes could be used to manipulate data on a screen instead of a mouse and keyboard. For the game, adding a monitor screen might allow players to see an overview of words that they have learned.

Appendices

Appendix A: Senior Project Analysis

Summary of Functional Requirements

Mac's Fiesta is a game designed for the Sifteo platform. It is designed to help people of all ages learns basic Spanish vocabulary words. Using the Sifteo cubes, players press the cube buttons, shake the cubes, or move them around to interact with digital data represented on them. The game helps associate recognize Spanish words with a corresponding picture and also helps them learn how to pronounce the word by hearing it said aloud through the computer.

Primary Constraints

One of the big challenges with the project implementation was not having readily available resources to help with game development. Sifteo cubes, along with the Sifteo SDK, have only been released for a little more than one year. Because they are so new, Sifteo

cubes are not yet popular among developers and there have not been a lot of research or tutorials written for them. It was tough to understand how to develop a game for the platform without previous examples.

Economic

Sifteo cubes retail for $149 for a set of 3 cubes. For a complete set of 6 cubes, the retail price is $249. Additional required equipment is a computer, not included in the cost due to previous ownership. Software required for development is free and can be downloaded online. Other costs included payment for original artwork, audio recordings and digital audio splicing. Original estimated software development time was 5 weeks, but actual development time ended up being 8 weeks.

The Sifteo cubes are manufactured by Sifteo, Inc. No information is gathered regarding the manufacturing costs and profits.

Sustainability

The application is still in development and there are still improvements that could be done with the project. This project documentation will be a great resource for anyone trying to develop a game for the Sifteo platform or wanting to continue with this project.

Social and Political

Mac's Fiesta has the potential to be useful in a classroom environment. Because it is a learning tool, the lessons that it is teaching should be made sure to be accurate and correct.

Development

For this project, I learned how to program in C# and with the .NET framework. Developing in C# is similar to developing in Java, so it wasn't too hard to understand. I also learned how to explore the Sifteo SDK and apply the skills that I've learned from different programming classes to understand the Sifteo API and implement the application.

Prior to this project, I had no experience with creating or developing games. I had to do a lot of research about what makes a game successful and engaging for players. I also

had to do research on how educational games are designed. Developing for Sifteo cubes presented a unique challenge. Input and output are based on interactions with the cubes. It's different in the way that the conventional keyboard and mouse aren't used to interact with a computer. As a developer, designing for the gaming application that handled these interactions required creativity and a different way of thinking. Overall, the learning process was fun and challenged me to think of how people interact with technology.

Appendix B: Getting Started with Sifteo SDK

Set up environment on Mac OS X

1. Go to https://www.sifteo.com/account/developer and create an account.
2. Download/Install Mono 2.10.6 Runtime for Mac OS X
   http://www.go-mono.com/mono-downloads/
3. Download/Install MonoDevelop 2.8.2 for Mac OS X
   http://monodevelop.com/Download
4. Download/Install SiftDev SDK for Mac
   https://www.sifteo.com/account/developer

Generate a new project

1. Open a terminal and navigate to *SIFTEO_SDK/tools/project_gen* where SIFTEO_SDK is the directory where you installed the Sifteo SDK.
2. Execute: *mono project_gen.exe MyAppName (-t /pathtotemplate)* where MyAppName is the name of your new app, and the path to the template to use is an optional argument. This creates a new directory *MyAppName*, containing your new app.
3. Copy your new app to wherever you like and open it in MonoDevelop.
4. In MonoDevelop, click on Project > Edit References...
5. Delete the existing Sifteo reference from the Selected references: pane, if any.
6. Click on the .NET Assembly tab.
7. Browse to your Sifteo.dll file in the .NET Assembly tab *SIFTEO_SDK/Siftdev.app/Contents/Resources/Runtimes/Mono/Current/sifteo/Sifteo.dll*
8. Click on the Add button in the bottom, towards the right corner of the window.
9. Click OK.

Build and Run Your Project

1. In MonoDevelop, open the .sin file created by project_gen.
2. Select Build > Build All - this will compile your project. If there are errors, check the console output and correct them.
3. Select Run > Run - the game should now be waiting for Siftdev to connect.
4. Open Siftdev. Click on Developer > Load Apps and choose the folder that contains the *manifest.json* file that describes your new app.
5. Your new app should now be included in the app library. If it's not, check your manifest.json file for errors, ensure it's pointing to the right paths for your asset resources, and try to load again.
6. Click on your app, and hit Play to begin running your app - that's it!

Appendix C: References

[1] Hoy, Trenton Edward, "There's an App for That: Foreign Language Learning Through Mobile- and Social Media-Based Video Games. " Master's Thesis, University of Tennessee, 2011. <http://trace.tennessee.edu/utk_gradthes/883/>.

[2] "Kindergarten Spanish Curriculum At a Glance." *Charlottesville City Schools - Elementary Spanish Program*. Web. <http://www.ccs.k12.va.us/programs/docs/ElemSpanish-K-Curriculum.pdf>.

[3] Merrill, David, and Jeevan Kalanithi. "Make a Riddle and TeleStory: Designing Children's Applications for the Siftables Platform." *MIT Media Laboratory*. 2010. <http://alumni.media.mit.edu/~dmerrill/publications/hunter_kalanithi_merrill_telestory-IDC2010.pdf>.

[4] *Sifteo*. Web. <https://www.sifteo.com/>.

[5] Voisen, Sean. "Sifteo Development with MVC and Ninject." 25 Oct. 2011. Web. <http://sean.voisen.org/blog/2011/10/sifteo-development-mvc-ninject/>.

[6] Voisen, Sean. "Up and Running with the Sifteo SDK." 4 Oct. 2011. Web. <http://sean.voisen.org/blog/2011/10/up-and-running-with-sifteo-sdk/>.

[7] "World Language Content Standards for California Public Schools (K-12)."*California Department of Education*. 13 Sept. 2010. Web. <http://www.cde.ca.gov/be/st/ss/documents/worldlanguage2009.pdf>.

Appendix D: Project Source Code

/MacMonkey

```
/* Bootstrap.cs
 * Generated automatically by Sifteo
 */

/* When debugging, it can sometimes be useful to launch your app directly,
 * outside of Siftrunner's app harness. Because BaseApps are meant to be run
 * inside Siftrunner, a bootstrapping process is required to kick off the
 * program manually.
 */

// All the classes in your app should share a namespace.
namespace MacMonkey
{
	/* The Bootstrap class is a simple wrapper with a Main() method that
	 * kicks off your app. All of your app's logic should go into the app
	 * class.
	 */
	public class Bootstrap
	{
		public static void Main (string[] args)
		{
			// Create the app and start it up.
			(new MacMonkeyApp ()).Run ();
		}
	}
}

/* MacMonkeyApp.cs
 * BaseApp for Mac's Fiesta
 * This is the first place the application enters
 */

using System;
using System.Collections;
using System.Collections.Generic;
using Sifteo;
using Sifteo.Util;
using MacMonkey.State;
using MacMonkey.Cubes;
using MacMonkey.Model;
using MacMonkey.Data;

namespace MacMonkey
{
	public class MacMonkeyApp : BaseApp
	{
		private StateMachine sm;
		private TitleController titleController;
		private MenuController menuController;
		private GameController gameController;
```

```
private bool canvasDirty;
private CubePainter cubePainter;
private PlayModel pm;
private WordRepository wr;
private Sound bgMusic;

/* Setup()
 * This function is where variables are initialized
 * and makes a call to SetupStateMachine().
 */
public override void Setup ()
{
	cubePainter = new CubePainter ();

	// Creates background music Sound object
	bgMusic = Sounds.CreateSound ("background");

	// Plays music at max volume and continuous loop
	bgMusic.Play (1, -1);
	SetupStateMachine ();
}

/* SetupStateMachine()
 * Creates state machine and initializes values.
 * Sets first state to be Title Screen
 */
public void SetupStateMachine ()
{
	sm = new StateMachine ();
	pm = new PlayModel ();

	// Reads and stores data from Word Repository XML file
	wr = new WordRepository ("WordRepository.xml", Sounds);

	// Initialize Controllers
	titleController = new TitleController (
		this.CubeSet,
		this.cubePainter,
		sm);
	menuController = new MenuController (
		this.CubeSet,
		this.cubePainter,
		sm,
		pm,
		Sounds,
		bgMusic
	);
	gameController = new GameController (
		this.CubeSet,
		this.cubePainter,
		sm,
		pm,
		wr
```

```
                );

                // Set states and transitions
                sm.State (States.Title, titleController);
                sm.State (States.Menu, menuController);
                sm.State (States.Game, gameController);

                sm.Transition (States.Null, Transitions.tNullToTitle,
                        States.Title);
                sm.Transition (States.Title, Transitions.tTitleToMenu,
                        States.Menu);
                sm.Transition (States.Menu, Transitions.tMenuToGame,
                        States.Game);
                sm.Transition (States.Game, Transitions.tGameToMenu,
                        States.Menu);

                // Set first state to Title Screen
                sm.SetState (States.Title, Transitions.tNullToTitle);
        }

        public override void Tick ()
        {
                // Call current state's OnTick() function
                sm.CurrentState.OnTick (1);

                if (canvasDirty) {
                        sm.Paint (canvasDirty);
                        canvasDirty = false;
                }
        }
    }
}
```

/MacMonkey/State

```
/* TitleController.cs
 * Controller class to handle the Title Screen.
 * Implements IStateController interface.
 */

using System;
using System.Collections;
using System.Collections.Generic;
using Sifteo;
using Sifteo.Util;
using MacMonkey.Cubes;
using MacMonkey.Data;

namespace MacMonkey.State
{
    public class TitleController : IStateController
    {
        private CubeSet cubeSet;
        private CubePainter cubePainter;
```

```
private StateMachine sm;
private StateMachineLock smLock;
private bool found = false;

public TitleController (CubeSet cubeSet, CubePainter cubePainter,
       StateMachine sm)
{
       this.cubeSet = cubeSet;
       this.cubePainter = cubePainter;
       this.sm = sm;
       smLock = new StateMachineLock (sm);
}

/* OnSetup()
 * Gets called whenever state machine enters this state
 */
public void OnSetup (string transitionId)
{
       Paint ();
       ListenForEvents ();
}

public void OnTick (float dt)
{
       if (sm.Current != States.Title)
              return;

       // Increments tick counter when state is locked
       if (smLock.Locked) {
              smLock.Tick ();
       }  else {
              if (found) {
                     // Queues transition when cubes are placed in
                     // correct order
                     sm.QueueTransition (Transitions.tTitleToMenu);
                     sm.Tick (1);
              }
       }
}

public void OnPaint (bool canvasDirty)
{
       if (canvasDirty) {
              Paint ();
       }
}

/* OnDispose()
 * Gets called whenever state exits
 */
public void OnDispose ()
{
       // Clears event handlers added within state
```

```
        // Ensures these functions don't get called from other states
        cubeSet.ClearEvents ();
}

// Private Methods

private void ListenForEvents ()
{
        cubeSet.NeighborAddEvent += NeighborAddHandler;
        cubeSet.NeighborRemoveEvent += NeighborRemoveHandler;
}

private void NeighborAddHandler (Cube cube1, Cube.Side side1, Cube cube2,
        Cube.Side side2)
{
        CheckCubes ();
}

/* CheckCubes()
 * Checks if cubes are placed in correct order
 */
private void CheckCubes ()
{
        Cube[] row = CubeHelper.FindRow (cubeSet);
        if (row.Length == 3) {
                if (row [0] == cubeSet [0] &&
                        row [1] == cubeSet [1] &&
                        row [2] == cubeSet [2]) {
                        found = true;
                        smLock.LockForTickCount (10);
                }
        }
}

private void NeighborRemoveHandler (Cube cube1, Cube.Side side1,
        Cube cube2, Cube.Side side2)
{
        // Does nothing, but can check if program recognizes removed cubes
        // with a call to Log.Debug()
}

/* Paint()
 * Puts image information on specific cubes that will be used to display
 * the title screen.
 */
private void Paint ()
{
        cubePainter.ClearScreen (cubeSet);

        cubePainter.PaintSpriteCentered (cubeSet [0], Sprites.TITLE1);
        cubePainter.PaintSpriteCentered (cubeSet [1], Sprites.TITLE2);
        cubePainter.PaintSpriteCentered (cubeSet [2], Sprites.TITLE3);
```

```
                        cubePainter.Commit (cubeSet);
                }
        }
}

/* MenuController.cs
 * Controller class to handle the Menu Screen
 */

using System;
using System.Collections;
using System.Collections.Generic;
using Sifteo;
using Sifteo.Util;
using MacMonkey.Cubes;
using MacMonkey.State;
using MacMonkey.Data;
using MacMonkey.Model;

namespace MacMonkey.State
{
        public class MenuController : IStateController
        {
                private CubeSet cubeSet;
                private CubePainter cubePainter;
                private StateMachine sm;
                private PlayModel pm;
                private Cube selectCube;
                private Cube categories1;
                private Cube categories2;
                private SoundSet sounds;
                private Sound bgMusic;
                private StateMachineLock smLock;

                public MenuController (CubeSet cubeSet, CubePainter cubePainter,
                        StateMachine sm, PlayModel pm, SoundSet sounds, Sound bgMusic)
                {
                        this.cubeSet = cubeSet;
                        this.cubePainter = cubePainter;
                        this.sm = sm;
                        this.pm = pm;
                        this.sounds = sounds;
                        this.bgMusic = bgMusic;
                        selectCube = cubeSet [0];
                        categories1 = cubeSet [1];
                        categories2 = cubeSet [2];
                        smLock = new StateMachineLock (this.sm);
                }

                public void OnSetup (string transitionId)
                {
                        // Ensures background music is played whenever game
                        // enters this state
```

```
        if (bgMusic.IsPaused) {
                bgMusic.Resume ();
        }

        ListenForEvents ();
        PaintDefault ();
}

public void OnTick (float dt)
{
        if (sm.Current != States.Menu)
                return;

        if (smLock.Locked) {
                smLock.Tick ();
        }  else if (pm.Category != null) {
                // Queues transition when category is set
                sm.QueueTransition (Transitions.tMenuToGame);
                sm.Tick (1);
        }
}

public void OnPaint (bool canvasDirty)
{
        if (canvasDirty) {
                canvasDirty = false;
        }
}

public void OnDispose ()
{
        cubeSet.ClearEvents ();
}

// Private Methods

private void ListenForEvents ()
{
        cubeSet.NeighborAddEvent += NeighborAddHandler;
        cubeSet.NeighborRemoveEvent += NeighborRemoveHandler;
}

/* NeighborAddHandler()
 * Checks for side of cubes to see which category is selected
 */
private void NeighborAddHandler (Cube cube1, Cube.Side side1, Cube cube2,
        Cube.Side side2)
{
        if (cube1 == selectCube && side1 == Cube.Side.TOP) {
                SetCategory (cube2, side2);
                if (pm.Category != null) {
                        // Pauses background music and plays clicking sound
                        // when category is selected
```

```
                        if (!bgMusic.IsPaused) {
                                bgMusic.Pause ();
                        }
                        sounds.CreateSound ("incorrect").Play (1, 0);
                        MoveToNextState ();
                }
        }  else if (cube2 == selectCube && side2 == Cube.Side.TOP) {
                SetCategory (cube1, side1);
                if (pm.Category != null) {
                        if (!bgMusic.IsPaused) {
                                bgMusic.Pause ();
                        }
                        sounds.CreateSound ("incorrect").Play (1, 0);
                        MoveToNextState ();
                }
        }
}

/* MoveToNextState()
 * Locks state for a short period before starting game
 */
private void MoveToNextState ()
{
        smLock.LockForTickCount (20);
}

/* SetCategory()
 * Determines which side of cube was selected and
 * stores corresponding category name in PlayModel pm
 */
private void SetCategory (Cube cube, Cube.Side side)
{
        if (cube == categories1) {
                switch (side) {
                case Cube.Side.LEFT:
                        pm.Category = Category.Animals;
                        break;
                case Cube.Side.BOTTOM:
                        pm.Category = Category.Colors;
                        break;
                case Cube.Side.RIGHT:
                        pm.Category = Category.Numbers;
                        break;
                default:
                        break;
                }
        }  else if (cube == categories2) {
                switch (side) {
                case Cube.Side.LEFT:
                        pm.Category = Category.Food;
                        break;
                case Cube.Side.BOTTOM:
                        pm.Category = Category.Family;
```

```
                        break;
                    case Cube.Side.RIGHT:
                        pm.Category = Category.Clothing;
                        break;
                    default:
                        break;
                    }
                }
            }

            private void NeighborRemoveHandler (Cube cube1, Cube.Side side1,
                Cube cube2, Cube.Side side2)
            {
                // Nothing to be done here
                // Log.Debug ("Neighbor removed");
            }

            /* PaintDefault()
             * Displays menu screen on cubes
             */
            private void PaintDefault ()
            {
                cubePainter.ClearScreen (cubeSet);

                cubePainter.PaintSpriteCentered (selectCube, Sprites.MENU1);
                cubePainter.PaintSpriteCentered (categories1, Sprites.MENU2);
                cubePainter.PaintSpriteCentered (categories2, Sprites.MENU3);

                cubePainter.Commit (cubeSet);
            }
        }
}

/* GameController.cs
 * Controller class for game session.
 */
using System;
using System.Collections;
using System.Collections.Generic;
using Sifteo;
using Sifteo.Util;
using MacMonkey.Cubes;
using MacMonkey.Model;
using MacMonkey.State;
using MacMonkey.Data;
using MacMonkey.Services;

namespace MacMonkey.State
{
    public class GameController : IStateController
    {
        private CubeSet cubeSet;
        private CubePainter cubePainter;
```

```
private StateMachine sm;
private PlayModel pm;
private GameModel gm;
private WordRepository wr;
private Random random = new Random(new System.DateTime().Millisecond);
private WordCube imageCube;
private Sound correct;
private Sound incorrect;
private TickCounter tickCounter;

public GameController (CubeSet cubeSet, CubePainter cubePainter,
       StateMachine sm, PlayModel pm, WordRepository wr)
{
       this.cubeSet = cubeSet;
       this.cubePainter = cubePainter;
       this.sm = sm;
       this.pm = pm;
       this.wr = wr;
       tickCounter = new TickCounter ();
}

public void OnSetup (string transitionId)
{
       if (gm == null) {
              gm = new GameModel ();
       }

       // Create sounds for correct and incorrect matches
       correct = wr.SoundSet.CreateSound ("correct");
       correct.StoppedEvent += SoundStoppedHandler;

       incorrect = wr.SoundSet.CreateSound ("incorrect");

       WrapCubes ();
       ListenForEvents ();
       StartRound ();
}

public void OnTick (float dt)
{
       if (sm.Current != States.Game)
              return;

       // Checks if cubes are flipped
       CheckOrientation ();
       tickCounter.Tick ();
}

public void OnPaint (bool canvasDirty)
{
       // Nothing to do here, painting is handled by other functions
}
```

```
public void OnDispose ()
{
	cubeSet.ClearEvents ();
	correct.StoppedEvent -= SoundStoppedHandler;
	gm = null;
}

// Private Methods

/* WrapCubes()
 * Wraps each cube into WordCube to handle data for a cube
 */
private void WrapCubes ()
{
	foreach (Cube cube in cubeSet) {
		if (cube.userData == null) {
			WordCube wordCube = new WordCube (cube, cubePainter);
			gm.WrappedCubes.Add (cube, wordCube);
		}
	}
}

private void StartRound ()
{
	gm.CurrentRound = CreateRound ();
	AssignWordsToCubes (gm.CurrentRound);
}

/* AssignWordsToCubes()
 * Assigns Spanish words to cubes and also handles display
 */
private void AssignWordsToCubes (GameRound currentRound)
{
	// Set image cube information
	imageCube = GetRandomWordCube ();
	imageCube.Data = currentRound.CurrentWord;
	imageCube.Type = WordCubeType.Image;
	if (pm.Category == Category.Colors) {
		imageCube.Type = WordCubeType.Color;
	}

	// Set word cube information
	int i = 0;
	foreach (WordCube wordCube in gm.WrappedCubes.Values) {
		if (wordCube != imageCube) {
			wordCube.Data = currentRound.Words [i];
			wordCube.Type = WordCubeType.Text;
			i++;
		}
		cubePainter.ClearScreen (wordCube.Cube);
		wordCube.Paint ();
	}
}
```

```
/* CreateRound()
 * Creates a game round by randomly choosing words from the category
 * chosen and stores them in GameModel gm
 */
private GameRound CreateRound ()
{
	GameRound round = new GameRound ();

	if (pm.Category == null) {
		pm.Category = Category.All;
	}

	round.Words = GetWords (
		pm.Category,
		gm.WrappedCubes.Values.Count - 1
	);
	round.CurrentWord = GetRandomWord (round.Words);

	return round;
}

/* GetWords()
 * Randomly chooses words based on category chosen.
 */
private WordData[] GetWords (string category, int numWords)
{
	List<WordData> wordsToChooseFrom =
		new List<WordData> (wr.GetList (category));

	WordData[] finalWords = new WordData[numWords];

	for (int i = 0; i < numWords; i++) {
		if (wordsToChooseFrom.Count == 0)
			break;

		int randomIndex = random.Next (0, wordsToChooseFrom.Count);
		finalWords [i] = wordsToChooseFrom [randomIndex];
		wordsToChooseFrom.RemoveAt (randomIndex);
	}

	return finalWords;
}

private WordData GetRandomWord (WordData[] words)
{
	return words [random.Next (0, words.Length)];
}

private WordCube GetRandomWordCube ()
{
	int randomIndex = random.Next (0, gm.WrappedCubes.Values.Count);
	return gm.WrappedCubes [cubeSet [randomIndex]];
```

```
}

// Event Handlers

private void ListenForEvents ()
{
	cubeSet.NeighborAddEvent += NeighborAddHandler;
	cubeSet.NeighborRemoveEvent += NeighborRemoveHandler;
}

private void NeighborAddHandler (Cube cube1, Cube.Side side1,
	Cube cube2, Cube.Side side2)
{
	WordCube wrappedCube1 = gm.WrappedCubes [cube1];
	WordCube wrappedCube2 = gm.WrappedCubes [cube2];

	if (wrappedCube1.Data.Word == wrappedCube2.Data.Word) {
		Log.Info ("Matched {0}", wrappedCube1.Data.Word);

		Cube textCube = (wrappedCube1.Type == WordCubeType.Text) ?
			wrappedCube1.Cube : wrappedCube2.Cube;

		// Highlight cube with yellow border to
		// indicate correct match
		cubePainter.PaintSpriteCentered (textCube,
			Sprites.CORRECT);
		textCube.Paint ();

		Cube imageCube = (wrappedCube1.Type == WordCubeType.Image
			|| wrappedCube1.Type == WordCubeType.Color) ?
			wrappedCube1.Cube : wrappedCube2.Cube;

		// Highlight cube with yellow border to
		// indicate correct match
		cubePainter.PaintSpriteCentered (imageCube,
			Sprites.CORRECT);
		imageCube.Paint ();

		// Play correct sound
		correct.Play (1, 0);

		// Start another round when match is made
		tickCounter.CallAfterTicks (50, this.StartRound);
	}  else {
		// Play incorrect sound
		incorrect.Play (1, 0);
	}
}

/* SoundStoppedHandler()
 * Plays Spanish pronunciation of current word-image pair
 * after correct bell sound is played.
 */
```

```
		private void SoundStoppedHandler (Sound sound)
		{
			if (sound != correct)
				return;

			imageCube.PlaySound ();
		}

		private void NeighborRemoveHandler (Cube cube1, Cube.Side side1,
			Cube cube2, Cube.Side side2)
		{
			//Nothing to be done here
			//Log.Debug ("Neighbor removed");
		}

		/* CheckOrientation()
		 * Checks if all cubes are flipped over to go back to menu screen
		 */
		private void CheckOrientation ()
		{
			foreach (Cube c in cubeSet) {
				if (c.IsUpright)
					return;
			}

			// Return to menu
			pm.Category = null;
			sm.QueueTransition (Transitions.tGameToMenu);
			sm.Tick (1);
		}

	}
}

/* StateMachineLock.cs()
 * Written by Sean Voisen and used for Mac's Fiesta
 */

using System;
using Sifteo;
using Sifteo.Util;

namespace MacMonkey.State
{
	public class StateMachineLock
	{
		private int tickCount;
		private int unlockAt;
		private StateMachine sm;
		private bool locked;
		private IDisposable smLock;

		public StateMachineLock (StateMachine stateMachine)
```

```
		{
			this.sm = stateMachine;
			locked = false;
			unlockAt = -1;
		}

		public StateMachine Sm {
			get {
				return this.sm;
			}
			set {
				sm = value;
			}
		}

		public bool Locked {
			get {
				return this.locked;
			}
		}

		public void LockForTickCount (int numTicks)
		{
			locked = true;
			smLock = sm.AquireLock ();
			unlockAt = numTicks;
			tickCount = 0;
		}

		public void Tick ()
		{
			if (!locked)
				return;

			tickCount++;
			if (tickCount >= unlockAt)
				Unlock ();
		}

		public void Unlock ()
		{
			locked = false;
			unlockAt = -1;
			smLock.Dispose ();
		}
	}
}

/* States.cs
 * Constants for State names
 */
using System;

namespace MacMonkey.State
```

```
{
	public struct States
	{
		public const string Null = "";
		public const string Title = "title";
		public const string Menu = "menu";
		public const string Game = "game";
	}
}

/* Transitions.cs
 * Constants for Transition IDs
 */
using System;

namespace MacMonkey.State
{
	public struct Transitions
	{
		public const string tNullToTitle = "nullToTitle";
		public const string tTitleToMenu = "titleToMenu";
		public const string tLanguageToMenu = "languageToMenu";
		public const string tMenuToLanguage = "menuToLanguage";
		public const string tMenuToGame = "menuToGame";
		public const string tGameToMenu = "gameToMenu";
	}
}
```

MacMonkey/Cubes

```
/* CubePainter.cs
 * Written by Sean Voisen and modified for Mac's Fiesta
 * Used to handle image data to be displayed on cubes
 */
using System;
using Sifteo;
using Sifteo.Util;

namespace MacMonkey.Cubes
{
	public class CubePainter
	{
		public void PaintAllImage (CubeSet cubes, String imageName)
		{
			foreach (Cube cube in cubes) {
				PaintFullImage (cube, imageName);
			}
		}

		public void PaintImage (Cube cube, string imageName, int x, int y,
			int width, int height)
		{
			cube.Image (imageName, x, y, 0, 0, width, height, 1, 0);
		}
```

```
public void PaintSprite (Cube cube, SpriteData data, int x, int y)
{
	cube.Image (data.imageName, x, y, data.source.x, data.source.y,
		data.size.x, data.size.y, 1, 0);
}

public void PaintSpriteCentered (Cube cube, SpriteData data)
{
	int x = (int)((Cube.SCREEN_WIDTH - data.size.x) * 0.5);
	int y = (int)((Cube.SCREEN_HEIGHT - data.size.y) * 0.5);

	cube.Image (data.imageName, x, y, data.source.x, data.source.y,
		data.size.x, data.size.y, 1, 0);
}

/* PaintFullImage()
 * Paints 128x128 image on cube
 */
public void PaintFullImage (Cube cube, string imageName)
{
	cube.Image (imageName, 0, 0, 0, 0, Cube.SCREEN_WIDTH,
		Cube.SCREEN_HEIGHT, 1, 0);
}

/* ClearScreen()
 * Fill cube with black color
 */
public void ClearScreen (Cube cube)
{
	cube.FillScreen (Color.Black);
}

public void ClearScreen (Cube cube, Color c)
{
	cube.FillScreen (c);
}

public void ClearScreen (CubeSet cubeSet)
{
	foreach (Cube cube in cubeSet) {
		ClearScreen (cube);
	}
}

/* Commit()
 * Commits data stored to be painted onto cubes and displays them
 */
public void Commit (CubeSet cubeSet)
{
	foreach (Cube cube in cubeSet) {
		cube.Paint ();
	}
```

```
		}

		public void Commit (Cube cube)
		{
			cube.Paint ();
		}
	}
}

/* WordCube.cs
 * Handles individual cube events
 */
using System;
using Sifteo;
using Sifteo.Util;
using MacMonkey.Cubes;
using MacMonkey.Data;

namespace MacMonkey.Cubes
{
	public class WordCube
	{
		private Cube cube;
		private CubePainter painter;
		private WordCubeType type;
		private WordData data;

		public WordCube (Cube cube, CubePainter painter)
		{
			this.cube = cube;
			this.painter = painter;
			type = WordCubeType.Text;
			this.cube.ButtonEvent += ButtonEventHandler;
			this.cube.ShakeStartedEvent += ShakeStartedHandler;
			this.cube.ShakeStoppedEvent += ShakeStoppedHandler;
		}

		public Cube Cube {
			get {
				return this.cube;
			}
		}

		public WordData Data {
			get {
				return this.data;
			}
			set {
				data = value;
			}
		}

		public WordCubeType Type {
```

```
	get {
		return this.type;
	}
	set {
		type = value;
	}
}

/* ButtonEventHandler()
 * If cube is pressed, display English word on screen
 * and play English pronunciation
 */
private void ButtonEventHandler (Cube c, bool pressed)
{
	if (pressed) {
		PaintEnglishWord ();
		PlayEnglishSound ();
	}
}

private void ShakeStartedHandler (Cube c)
{
	//Log.Debug ("Shake started.");
}

/* ShakeStoppedHandler()
 * Plays Spanish pronunciation after cube is shaken
 */
private void ShakeStoppedHandler (Cube c, int duration)
{
	if (duration > 300) {
		this.PlaySound ();
	}
}

/* PlaySound()
 * Plays Spanish pronunciation of word represented by cube
 */
public void PlaySound ()
{
	Log.Debug ("Playing sound {0}", data.Sound.Name);

	// To increase sound volume, sound is overlaid
	// 5 times and played altogether.
	for (int i = 0; i < 5; i++) {
		data.Sound.Play (1, 0);
	}
}

/* PlayEnglishSound()
 * Plays English pronunciation of word represented by cube
 */
private void PlayEnglishSound ()
```

```
		{
			data.EnglishSound.StoppedEvent += HandleStoppedEvent;

			// To increase sound volume, sound is overlaid
			// 5 times and played altogether.
			for (int i = 0; i < 5; i++) {
				data.EnglishSound.Play (1, 0);
			}
		}

		/* HandleStoppedEvent()
		 * Paint correct cube data after English pronunciation
		 * is stopped
		 */
		void HandleStoppedEvent (Sound sound)
		{
			if (sound != data.EnglishSound)
				return;

			Paint ();
		}

		/* Paint()
		 * Decides which image to display on cube
		 */
		public void Paint ()
		{
			if (!data.isValid ())
				return;

			cube.FillScreen (Color.White);

			if (Type == WordCubeType.Text) {
				painter.PaintSpriteCentered (cube, data.WordSpriteData);
			}  else if (Type == WordCubeType.Image) {
				painter.PaintSpriteCentered (cube, data.ImageSpriteData);
			}  else if (Type == WordCubeType.Color) {
				painter.ClearScreen (cube, data.Color);
			}
			painter.Commit (cube);
		}

		private void PaintEnglishWord ()
		{
			if (!data.isValid ())
				return;

			cube.FillScreen (Color.White);
			painter.PaintSpriteCentered (cube, data.EnglishWordSpriteData);
			painter.Commit (cube);
		}
	}
}
```

```
/* WordCubeType.cs
 * Enum for Word Cube Type
 */
using System;

namespace MacMonkey.Cubes
{
	public enum WordCubeType
	{
		Image,
		Text,
		Color
	}
}
```

MacMonkey/Data

```
/* WordRepository.cs
 * Reads XML file and stores it for game
 */
using System;
using System.Collections.Generic;
using System.Xml;
using Sifteo;
using Sifteo.Util;
using MacMonkey.Data;

namespace MacMonkey.Data
{
	public class WordRepository
	{
		private Dictionary<string, List<WordData>> categoryLists;
		private List<WordData> allWords = new List<WordData> ();
		private SoundSet soundSet;

		public WordRepository (string xmlFile, SoundSet soundSet)
		{
			this.soundSet = soundSet;
			BuildLists (xmlFile);
		}

		/* GetList()
		 * Returns a list of words based on category given.
		 */
		public List<WordData> GetList (string category)
		{
			if (category.Equals (Category.All)) {
				return allWords;
			}
			return categoryLists [category];
		}

		/* BuildLists()
```

```
 * Reads XML data file and stores data accordingly
 */
private void BuildLists (string xmlFile)
{
	categoryLists = new Dictionary<string, List<WordData>> ();

	XmlDocument doc = new XmlDocument ();
	doc.Load (xmlFile);

	XmlNodeList nodes = doc.GetElementsByTagName ("word");

	WordData newWord;

	foreach (XmlNode node in nodes) {
		newWord = new WordData ();
		newWord.Color = Color.White;

		newWord.Category =
			node.Attributes.GetNamedItem ("category").InnerText;

		// Store SpriteData for image
		if (newWord.Category.Equals ("Colors")) {
			int r = AttributeToInt32 (node, "r");
			int g = AttributeToInt32 (node, "g");
			int b = AttributeToInt32 (node, "b");
			newWord.Color = new Color (r, g, b);
		}  else {
			string imageName =
			node.Attributes.GetNamedItem("image").InnerText;

			int x = AttributeToInt32 (node, "x");
			int y = AttributeToInt32 (node, "y");
			int width = AttributeToInt32 (node, "width");
			int height = AttributeToInt32 (node, "height");
			newWord.ImageSpriteData = new SpriteData (
				imageName,
				x,
				y,
				width,
				height,
				0,
				0
			);
		}

		foreach (XmlNode wordNode in node.ChildNodes) {
			string lang = wordNode.Name;

			// Store information for Spanish text
			if (lang.Equals ("sp")) {
				newWord.Word = wordNode.InnerText;
				string imageName =
					wordNode.Attributes.GetNamedItem("image
```

```
            ").InnerText;

            int x = AttributeToInt32 (wordNode, "x");
            int y = AttributeToInt32 (wordNode, "y");
            int width = AttributeToInt32 (wordNode,
                  "width");
            int height = AttributeToInt32 (wordNode,
                  "height");
            newWord.WordSpriteData = new SpriteData (
                  imageName,
                  x,
                  y,
                  width,
                  height,
                  0,
                  0
            );

            string soundName =
            wordNode.Attributes.GetNamedItem
                  ("sound").InnerText;

            newWord.Sound =
                  soundSet.CreateSound (soundName);
      }  else if (lang.Equals ("en")) {
            // Store information for English text
            string imageName =
                  wordNode.Attributes.GetNamedItem
                        ("image").InnerText;
            int x = AttributeToInt32 (wordNode, "x");
            int y = AttributeToInt32 (wordNode, "y");
            int width = AttributeToInt32 (wordNode,
                  "width");
            int height = AttributeToInt32 (wordNode,
                  "height");
            newWord.EnglishWordSpriteData =
            new SpriteData (
                  imageName,
                  x,
                  y,
                  width,
                  height,
                  0,
                  0
            );

            string soundName =
            wordNode.Attributes.GetNamedItem
                  ("sound").InnerText;
            newWord.EnglishSound =
            soundSet.CreateSound (soundName);
      }
```

```
                        }
                        AddWordData (newWord);
                }
        }

        /* AddWordData()
         * Add Word Data to Dictionary
         */
        private void AddWordData (WordData data)
        {
                List<WordData> list;
                if (categoryLists.TryGetValue (data.Category, out list)) {
                        list.Add (data);
                }  else {
                        list = new List<WordData> ();
                        list.Add (data);
                        categoryLists.Add (data.Category, list);
                }

                allWords.Add (data);
        }

        private int AttributeToInt32 (XmlNode node, string attributeName)
        {
                return Int32.Parse (node.Attributes.GetNamedItem
                        (attributeName).InnerText);
        }

        public SoundSet SoundSet {
                get {
                        return this.soundSet;
                }
                set {
                        soundSet = value;
                }
        }
    }
}

/* WordData.cs
 * Stores information for a word
 */
using System;
using Sifteo;
using Sifteo.Util;

namespace MacMonkey.Data
{
    public struct WordData
    {
        public string category;
        public string word;
```

```
private SpriteData wordSpriteData;
private SpriteData imageSpriteData;
private SpriteData englishWordSpriteData;
private Color color;
private Sound sound;
private Sound englishSound;

public Color Color {
	get {
		return this.color;
	}
	set {
		color = value;
	}
}

public string Category {
	get {
		return this.category;
	}
	set {
		category = value;
	}
}

public SpriteData ImageSpriteData {
	get {
		return this.imageSpriteData;
	}
	set {
		imageSpriteData = value;
	}
}

public string Word {
	get {
		return this.word;
	}
	set {
		word = value;
	}
}

public SpriteData WordSpriteData {
	get {
		return this.wordSpriteData;
	}
	set {
		wordSpriteData = value;
	}
}

public SpriteData EnglishWordSpriteData {
```

```
                get {
                        return englishWordSpriteData;
                }
                set {
                        englishWordSpriteData = value;
                }
        }
        public bool isValid ()
        {
                return (category != null && word != null
                        && wordSpriteData != null);
        }

        public Sound Sound {
                get {
                        return this.sound;
                }
                set {
                        sound = value;
                }
        }

        public Sound EnglishSound {
                get {
                        return englishSound;
                }
                set {
                        englishSound = value;
                }
        }
    }
}

/* GameRound.cs
 * Stores information for current game round
 */
using System;
using System.Collections.Generic;

namespace MacMonkey.Data
{
    public class GameRound
    {
        private WordData[] words;
        private WordData currentWord;

        public WordData CurrentWord {
                get {
                        return this.currentWord;
                }
                set {
                        currentWord = value;
                }
```

```
		}

		public WordData[] Words {
			get {
				return this.words;
			}
			set {
				words = value;
			}
		}
	}
}

/* Category.cs
 * Constants for categories available in game
 */
using System;

namespace MacMonkey.Data
{
	public struct Category
	{
		// Category
		public const string Colors = "Colors";
		public const string Numbers = "Numbers";
		public const string Food = "Food";
		public const string Family = "Family";
		public const string Animals = "Animals";
		public const string Clothing = "Clothing";
		public const string All = "All";
	}
}

/* Sprites.cs
 * Stores data for static Sprite Data
 */
using System;
using Sifteo;
using Sifteo.Util;

namespace MacMonkey.Data
{
	public class Sprites
	{
		public static SpriteData TITLE1 =
			new SpriteData("title", 0, 0, 128, 128, 0, 0);
		public static SpriteData TITLE2 =
			new SpriteData("title", 128, 0, 128, 128, 0, 0);
		public static SpriteData TITLE3 =
			new SpriteData("title", 256, 0, 128, 128, 0, 0);

		// Menu
		public static SpriteData MENU1 =
```

```
                new SpriteData ("menu", 0, 0, 128, 128, 0, 0);
            public static SpriteData MENU2 =
                new SpriteData ("menu", 128, 0, 128, 128, 0, 0);
            public static SpriteData MENU3 =
                new SpriteData ("menu", 256, 0, 128, 128, 0, 0);

            // Correct Match
            public static SpriteData CORRECT =
                new SpriteData("score", 0, 0, 128, 128, 0, 0);
        }
}
```

MacMonkey/Model

```
/* PlayModel.cs
 * Stores information for current game category chosen
 */
using System;
using MacMonkey.Data;

namespace MacMonkey.Model
{
        public class PlayModel
        {
            private String category;

            public String Category {
                get {
                    return this.category;
                }
                set {
                    category = value;
                }
            }
        }
}

/* GameModel.cs
 * Stores information for current game round.
 */
using System;
using Sifteo;
using System.Collections.Generic;
using MacMonkey.Cubes;
using MacMonkey.Data;

namespace MacMonkey.Model
{
        public class GameModel
        {
            private Dictionary<Cube, WordCube> wrappedCubes = new Dictionary<Cube,
WordCube> ();
            private List<WordData> usedWords = new List<WordData> ();
```

```
		private GameRound currentRound = new GameRound ();

		public Dictionary<Cube, WordCube> WrappedCubes {
			get {
				return this.wrappedCubes;
			}
			set {
				wrappedCubes = value;
			}
		}

		public List<WordData> UsedWords {
			get {
				return this.usedWords;
			}
			set {
				usedWords = value;
			}
		}

		public GameRound CurrentRound {
			get {
				return this.currentRound;
			}
			set {
				currentRound = value;
			}
		}
	}
}
```

MacMonkey/Services

```
/* TickCounter.cs
 * Written by Sean Voisen
 * Used in GameController to count ticks before going on to next round
 */
using System;

namespace MacMonkey.Services
{
	public delegate void TickCompleteCallback ();

	public class TickCounter
	{
		public TickCounter ()
		{
			Enabled = false;
		}

		public void Tick ()
		{
			if (!Enabled)
				return;
```

```
                tickCount++;

                if (tickCount >= tickThreshold) {
                        Enabled = false;
                        tickCount = 0;
                        callback ();
                }
        }

        public void CallAfterTicks (int tickThreshold,
                TickCompleteCallback callback)
        {
                this.tickThreshold = tickThreshold;
                this.callback = callback;

                tickCount = 0;
                Enabled = true;
        }

        public bool Enabled {
                get;
                set;
        }

        private int tickThreshold = 0;
        private int tickCount = 0;
        private TickCompleteCallback callback;
    }
}
```