Siftables Emulator
Singularity Software

November 30, 2011

Alex Mullans
Ethan Veatch
Eric Vernon
Kurtis Zimmerman

Contents

1 Introduction

Developers of applications for the Sifteo Cubes currently must test programs they create for the platform on the Cubes themselves. With a full release of the Cubes and corresponding Application Programming Interface (API) still pending, developers unable to join the Sifteo Early Access program are left without a software-based interface within which to productively develop Sifteo programs. As such, Singularity Software will provide, in the form of the Siftables Emulator, a software-based emulator for the Sifteo Cubes that will allow any developer to try programming in the unqiue environment provided by the Cubes.

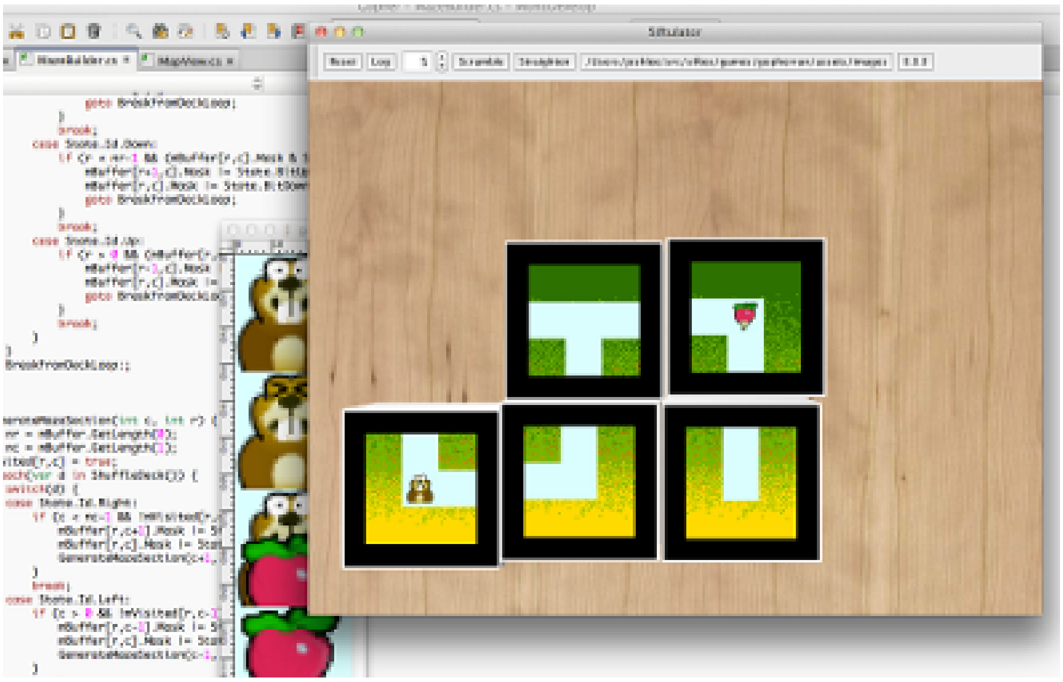
This document summarizes a series of milestone documents that accompany the planning of the Siftables Emulator.

2 Client Background

Clients Tim Ekl and Eric Stokes are alumni of Rose-Hulman. Mr. Ekl is currently working on a M.S. degree in Engineering Management; Mr. Stokes is currently working for n~ask Signal Processing Systems in Denver, Colorado. As former Rose-Hulman students, the clients are avid users of technology who follow new trends in the industry. As a result of this interest in new technology, both Mr. Stokes and Mr. Ekl discovered and purchased Sifteo Cubes, a revolutionary product that consists of a set of (anywhere from 1 to 6) mini computers. While both clients have a set of 3 Sifteo Cubes, they realized that not everyone interested in the project could have the luxury of physical hardware to work with. As such, they asked Singularity Software—via the Junior Project proposal process at Rose-Hulman—to construct a software emulator for the Cubes that would make the development of Sifteo applications easier, especially in the testing phase.

3 Current System

Currently the only system available to developers is the Siftulator, the emulator released in Sifteo's SDK. It allows for emulation of all Cube manipulations, and the numebr of Cubes can be increased or reduced through the interface. An API is available through the Sifteo website, and several example games have been written for use in the emulator workspace. The Siftulator only runs on Windows machines, leaving it inaccessible to some users. A screenshot, taken from Sifteo's developer website, is shown below.



```
                goto BreakFromDeckLoop;
            }
            break;
        case State.Id.Down:
            if (r < nr-1 && (mBuffer[r,c].Mask & 
                mBuffer[r+1,c].Mask |= State.BitUp
                mBuffer[r,c].Mask |= State.BitDown
                goto BreakFromDeckLoop;
            }
            break;
        case State.Id.Up:
            if (r > 0 && (mBuffer[r,
                mBuffer[r-1,c].Mask 
                mBuffer[r,c].Mask |=
                goto BreakFromDeckLo
            }
            break;
    }
}
BreakFromDeckLoop:;


nerateMazeSection(int c, int r) {
nr = mBuffer.GetLength(0);
nc = mBuffer.GetLength(1);
ited[r,c] = true;
ach(var d in ShuffleDeck()) {
switch(d) {
case State.Id.Right:
    if (c < nc-1 && !mVisited[r,c
        mBuffer[r,c+1].Mask |= S
        mBuffer[r,c].Mask |= Stat
        GenerateMazeSection(c+1,
    }
    break;
case State.Id.Left:
    if (c > 0 && !mVisited[r,c-1
        mBuffer[r,c-1].Mask |= S
        mBuffer[r,c].Mask |= Stat
        GenerateMazeSection(c-1,
    }
```

4 User/Stakeholder Description

4.1 User/Stakeholder Profiles

4.1.1 Sifteo application developers

As the primary target audience of the system, developers targeting the Sifteo platform are assumed to have a reasonable amount of technical background; they are not novice computer users and are familiar with programming concepts like Object-Oriented Programming and APIs. As developers may hail from platforms ranging from Windows to Mac to Linux, cross-platform support is an important consideration.

Possible problems for this user type include an unstable or crash-prone system: as developers are writing and testing their own code, it is essential that the emulator does not contribute to the failures that the user must debug. Developers will deem the emulator project a success when they can successfully program and test software that uses any or all of the features of the Sifteo cubes on their development platform.

4.1.2 Clients

The clients (Tim Ekl and Eric Stokes) are assumed to be a subset of the Sifteo application developers user class. However, their programming knowledge and knowledge of the Sifteo

Cubes is known to be more advanced than that of the average application developer. As such, their needs require that the emulator is capable of being pushed to the same limits as the actual platform.

4.1.3 Singularity Software

Singularity Software, as the team behind Siftables Emulator, is primarily interested in the creation of a finished product that can be delivered to the clients at the conclusion of the Rose-Hulman junior project cycle. As a team, we are less familiar with the Sifteo platform and are also relatively inexperienced with the scale of project the emulator entails.

4.1.4 Sriram Mohan (CSSE Department)

As the advisor of the Junior Project, Dr. Mohan has a vested interest in the creation of a finished, deliverable product. His key responsibility is to review documents created within the scope of the Junior Project series of courses.

4.2 User Environments

4.2.1 Sifteo application developers

The typical Sifteo application developer may be working on his own, or he may be working with a team of developers; he or they will be working on workstations or powerful development laptops that have a significant amount of graphics horsepower. They may or may not be connected to the Internet during development, depending on the location in which they are developing. Additionally, they may be Windows, Mac, or Linux users and will be using various Integrated Development Environments (IDEs) specific to their platform; integration between such IDEs and the Siftables Emulator, while possibly desirable, is not a requirement.

4.2.2 Clients

Mr. Stokes and Mr. Ekl are both primarily Mac users, although both clients also own and occasionally use Windows machines as well. Their environment is essentially the same as that of the typical Sifteo application developer.

4.3 Key Needs

4.3.1 Emulate Sifteo Cubes in a desktop GUI application

No emulator currently exists for the Sifteo platform; the need is currently either filled by homebrew efforts like Mr. Ekl's Java-based emulator, or circumvented by using the Cubes themselves as a testing platform. The clients envision a solution where all of the interactions possible with a set of Early Access Sifteo Cubes can be replicated in a software emulator.

4.3.2 Develop an API for creating applications in the emulators Cubes

An API is necessary to facilitate interaction with the virtual Sifteo Cubes. Currently, no API has been made available by Sifteo for the physical cubes, and no emulator API exists because no emulator exists. The clients would like an API with which the virtual Cubes can be programmed. Mr. Ekl stipulated that shadowing the official Sifteo API, while potentially beneficial for long-term development, is not a requirement.

4.3.3 Showcase Cube/emulator functionalities with samples

Sifteo currently provides example games that run on the Cubes as a showcase of what the platform can achieve and what unique features it can offer to the user. The clients would like to have a similar showcase available for the emulator as an aid in understanding both the emulator platform and the larger Sifteo Cubes programming platform.

4.4 Alternatives & Competition

Singularity Software's Siftables Emulator will be the first software of its kind for the Cubes. The only true competition is the Sifteo Cubes themselves. The Cubes have the advantage of physicality—as tactile objects, they will always be superior in terms of user experience when compared to an emulator. However, they are expensive and only manufactured in limited quantities at the moment; the Siftables Emulator is, by contrast, infinitely available as an open source piece of software.

5 Product Overview

5.1 Product Perspective

Siftables Emulator is a free independent system used to emulate the way Sifteo Cubes handle motions and interactions.

5.2 Elevator Statement

Due to the limited availability of Sifteo Cubes, developers unable to obtain a set of Cubes have no good way to test the programs they create for the platform. At Singularity Software, our goal is to develop an emulator for the Cubes that will be able to emulate an arbitrary number of Sifteo Cubes and the way they handle physical motions and interactions. Along with the emulator itself, Singularity will develop an API and example games and programs.

5.3 Summary of Capabilities

The main features of our emulator work together to allow developers to quickly start Sifteo application development by making a virtual edition of the Cubes available for emulation and testing.

Feature	**Benefit**
Workspace where multiple cubes can be emulated	A user-friendly way to develop for multiple cubes
Buttons and gestures to control the cubes	An easier way to control the basic movements of the cubes in place of physical manipulations
Ability to load programs into the cubes	Allows the user to test his own programs and example programs in the emulator
Example games (requirement)	Gets new emulator users started with the platform
Open source (requirement)	Allows the community to contribute improvements to the emulator
API (requirement)	A standard way of interacting with the virtual Cubes

5.4 Assumptions and Dependencies

Sifteo has plans to release an API of their own for the Cubes; Singularity will attempt to make our API shadow much of the language and functionality of the official Sifteo API.

5.5 Estimate of Cost

Because it is an open source piece of software, Singularity Software does not believe that the Siftables Emulator will incur any monetary costs throughout the project.

6 Features

Six attributes accompany each feature described below:

Status: a measure of the feature's progress duing the project definition period, either *Proposed* or *Approved*

Priority: the relative importance of each feature, either *Useful*, *Important*, or *Critical*

Risk: the probability that the feature will bring about undesirable events, either *Low*, *Medium*, or *High*

Stability: the probability that the feature will change, either *Low*, *Medium*, or *High*

Reason: the source of the required feature

Effort: an estimate of the relative amount of work required to complete the feature, either *Low*, *Medium*, or *High*

The features are outlined in the table on the following page.

Feature	Description	Status	Priority	Risk	Stability	Reason	Effort
Individual, virtual Sifteo Cube	A virtual representation of a single Sifteo cube	Approved	Critical	Low	High	Replicates physical Sifteo Cube	Medium
Buttons to manipulate each virtual Cube	Buttons on the virtual Cube will allow the user to flip and tilt it	Approved	Critical	Medium	High	Replaces physical actions where said actions would be impractical with a mouse	Medium
Workspace where multiple cubes can be emulated	Multiple cubes will be displayed on a workspace that replicates the free-form nature of physical Sifteo Cubes	Approved	Critical	Low	High	Replicates multiple Sifteo Cubes in a natural, free-form environment	High
Interactions between Cubes	The Cubes present on the workspace will communicate when they are neighbored	Approved	Critical	Low	High	Cubes can simulate the interactions possible with physical Cubes	High
Load programs into the Cubes	The user will load his own and example programs into the emulators Cubes	Approved	Critical	Medium	High	The ability to program programs for the emulator is dependent on a common interface	High
Snap Cubes to invisible grid	The Cubes will snap into an invisible grid when a button is clicked	Proposed	Useful	Medium	High	Increases productivity by allowing a quick reset if the Cubes are in disarray	Low
Zoom Workspace	The Workspace will zoom to the level of an individual Cube or the whole space	Proposed	Useful	Low	High	Inspecting individual Cubes allows for precise checks of program Graphical User Interfaces (GUIs)	Low

7 Constraints

While much of this project is open-ended, there are a few basic constraints. At the direction of the clients, all code should be open source and version-controlled. Mr. Ekl requested that the emulator run easily on Mac as well as Windows, with the stipulation that Linux compatibility would satisfy the Mac requirement for Singularity's testing purposes. In addition, the clients requested that an issue tracking system be put in place and used throughout the development process. Finally, the emulator must be completed by May 18th—the end of Spring Quarter 10th week—to satisfy the requirements of the clients and of Dr. Mohan.

8 Use Cases

8.1 Load program

Name: Load program

Description: The User selects the program file to be loaded and run by the emulator.

Actors: User

Basic flow:

1. The User presses the "Load a program" button.
2. The User selects a *.siftem file in the file dialog.
3. The User presses "Open" button.
4. The emulator loads the selected program on the Cubes in the emulator.

Alternate flows:

When the User opens an incompatible file (i.e. any file without the .siftem extension),

1. An error dialog is presented to the User with the message: "The selected file is not a .siftem emulator file and cannot be loaded."
2. The use case terminates and no program is loaded.

When the User opens a corrupt or otherwise unloadable file,

1. An error dialog is presented to the User with the message: "The selected file is corrupt and cannot be loaded."
2. The use case terminates and no program is loaded.

When the User presses the "Cancel" button:

1. The use case terminates and no program is loaded.

When the User is already running a program,

1. A warning dialog is presented to the User with the message: "Loading this program will clear all data from the previous program run. Proceed?"

2. If the User presses “Yes” on the warning dialog, flow returns to Step 2 of the basic flow.

When the User presses “No” on the warning dialog:

1. The use case terminates and the program is not reloaded.

Pre-conditions:

1. The emulator is running.

Post-conditions:

1. The program is loaded or the User cancelled loading.

Special requirements

1. The emulator should indicate that loading the program is in progress.
2. The emulator should indicate when the program is finished loading.

8.2 Reload program

Name: Reload program

Description: The User reloads the program currently running in the emulator.

Actors: User

Basic flow:

1. A warning dialog is presented to the User with the message: ”Reloading this program will clear all data from the previous program run. Proceed?”
2. If the User presses “Yes” on the warning dialog, the Emulator loads the program onto the Cubes in the emulator.

Alternate flows:

When the User presses “No” on warning dialog:

1. The use case terminates and the program is not reloaded.

Pre-conditions:

1. A program is loaded in the emulator.

Post-conditions:

1. The program is reloaded or the current program state remains on the Cubes.

Special requirements:

1. The emulator should indicate that loading the program is in progress.
2. The emulator should indicate when the program is finished loading.

8.3 Zoom screen

Name: Zoom screen

Description: The User zooms the Workspace to the desired level.

Actors: User

Basic flow:

1. The User adjusts the zoom slider.
2. The Emulator magnifies the Cubes in the emulator according to the zoom level.

Alternate flows:
None

Pre-conditions:

1. The emulator is running.

Post-conditions:

1. The program running at the beginning of this use case, if any, is still running.

Special requirements:

1. The zoom slider moves in discrete increments. The lowest (and default) level shows the whole workspace and the highest level shows one Cube with the edges of the surrounding Cubes visible for context.

8.4 Add/remove Cubes

Name: Add/remove Cubes

Description: The User adjusts the number of Cubes present in the emulator.

Actors: User

Basic flow:

1. The User drags the "Number of Cubes" slider or uses the up/down arrows on the spinbox to increment or decrement the number of available Cubes by one.
2. The emulator adds/removes Cubes in emulator and resets its Workspace.
3. If Cubes are to be removed, the emulator starts with the bottom right-most of the Cubes (at their current positions) and works left. If more Cubes are to be removed after the second row is depleted, the emulator again starts at the bottom right-most of the remaining Cubes.

Alternate flows:
None

Pre-conditions:

1. The emulator is running.

Post-conditions:

1. The number of Cubes has been adjusted to the number specified.
2. The running program, if any, is terminated.

Special requirements:

1. The “Number of Cubes” slider moves in discrete increments. The leftmost level shows one Cube and the rightmost level shows six Cubes.

8.5 Snap Cubes to grid

Name: Snap to grid

Description: The Users pulls the Cubes into a grid orientation.

Actors: User

Basic flow:

1. The User presses the “Snap to Grid” button.
2. The Emulator moves the Cubes to a grid orientation based on their current positions. It will maintain the Cubes’ positions relative to other Cubes while doing so.

Alternate flows:
None

Pre-conditions:

1. The emulator is running.

Post-conditions:

1. The Cubes are arranged in a grid.

8.6 Manipulate Cube

Name: Manipulate Cube

Description: The User manipulates a Cube by clicking the buttons or the Cube itself.

Actors: User

Basic flow:

1. The User double clicks on a Cube.
2. The Cube responds as if a screen click occured.

Alternate flows:

1. The User clicks on one of the buttons superimposed on the Cubes’ edges.

2. The Cube executes the appropriate action (i.e. flips, rotates, or tilts).

1. The User drags a Cube next to another Cube.
2. Cube communicates (“neighbors”) with the Cube(s) it is adjacent to.

The User “shakes” a Cube back and forth with the mouse (i.e. he laterally moves the Cube back and forth several times).

The Cube responds as if shaken.

Pre-conditions:

1. The emulator is running.

Post-conditions:

1. If a program is running, the emulator has updated its state based on the Cube’s change.

8.7 Other functional requirements

8.7.1 API

The emulator will include an API in order to define a set of rules and specifications via which Cube programs can be created.

8.7.2 Example games

The emulator will include games as examples that demonstrate to the User how the Cubes interact with each other.

9 Use Case Feature Mapping

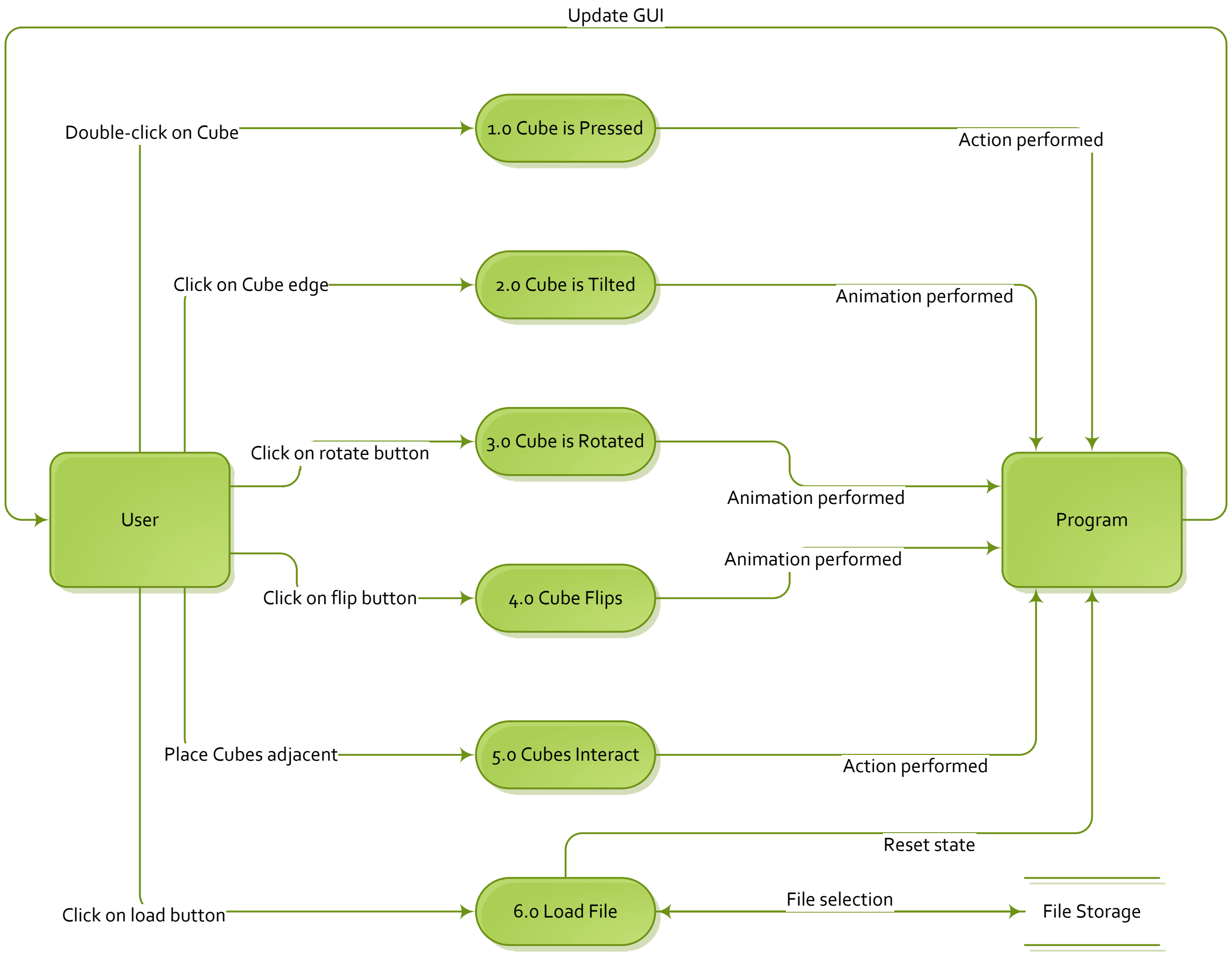
The feature listing can be found in appendix A, and the use case IDs refer to the use cases specified above.

Use case ID	Use case	Feature ID
U1	Load program	F5
U2	Reload program	F5
U3	Zoom screen	F7
U4	Add/remove Cubes	F3, F4
U5	Snap Cubes to grid	F6
U6	Manipulate Cube	F1, F2, F3
OR1	API	F2
OR2	Example games	F3

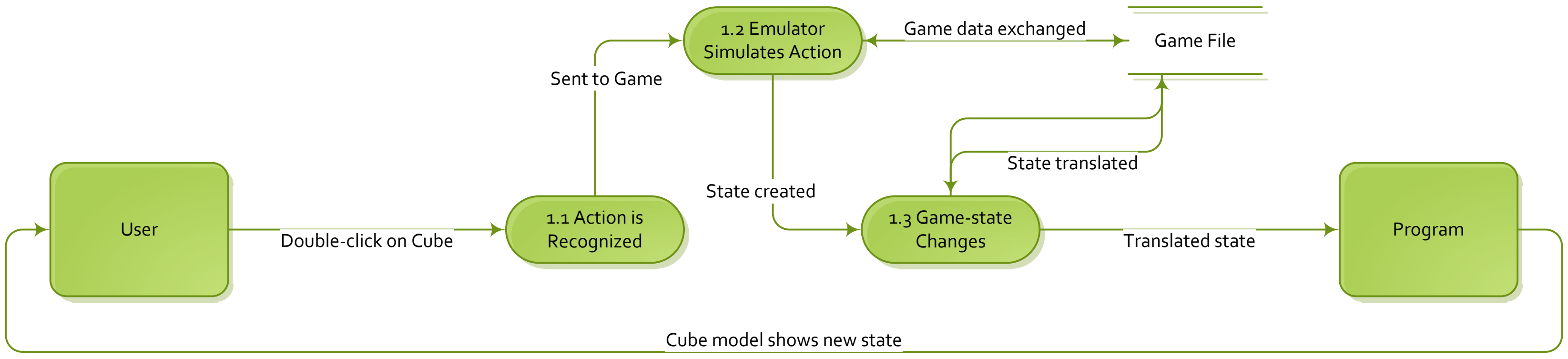
10 Work/Data Flow Diagrams

The following pages contain work/data flow diagrams that elucidate the passage of control and data through the emulator. As there is not much pure data that flows through the emulator system aside from the User's program, Singularity Software has elected to combine the workflow and data flow diagrams.

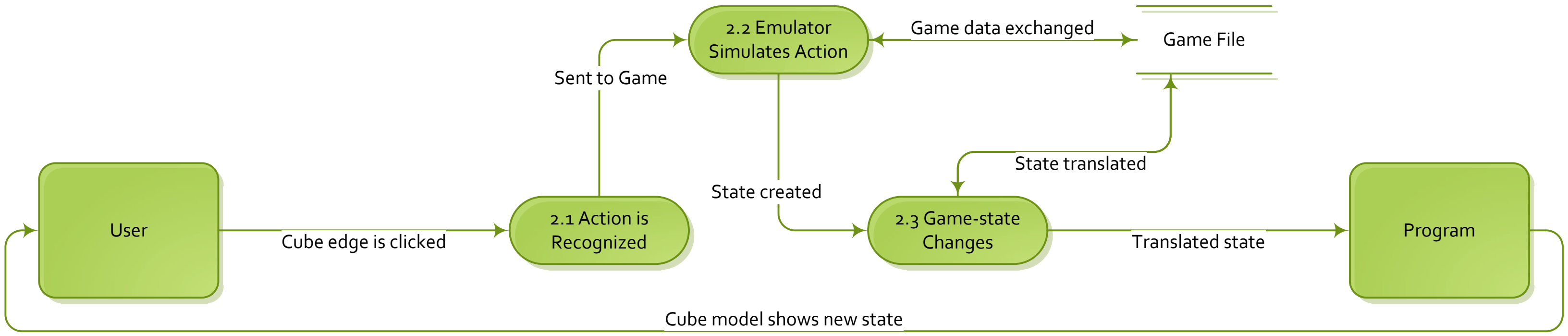
Level 0 (Context) Work/Data Flow Diagram



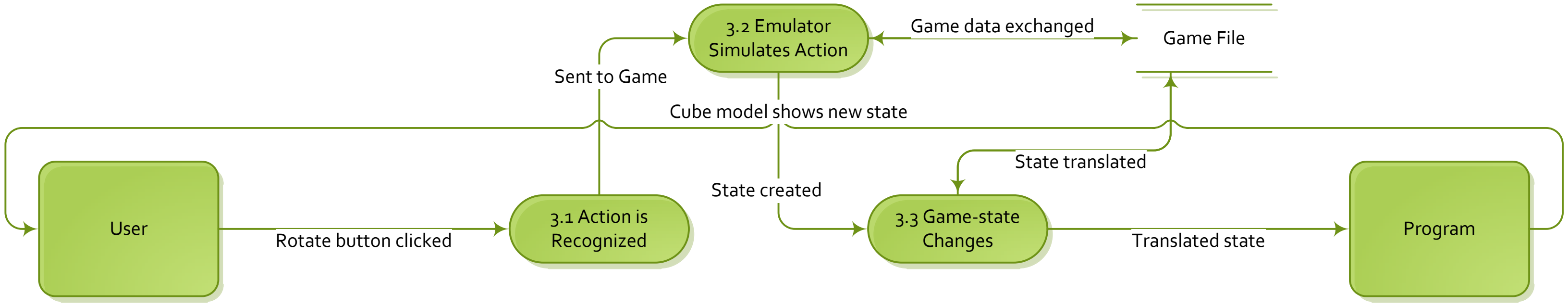
Level 1-1 Data Flow Diagram



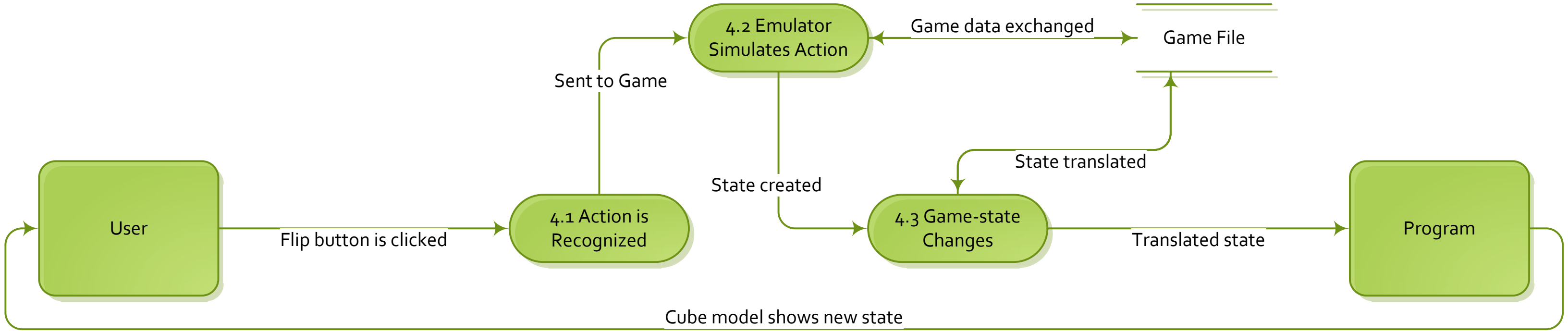
Level 1-2 Data Flow Diagram



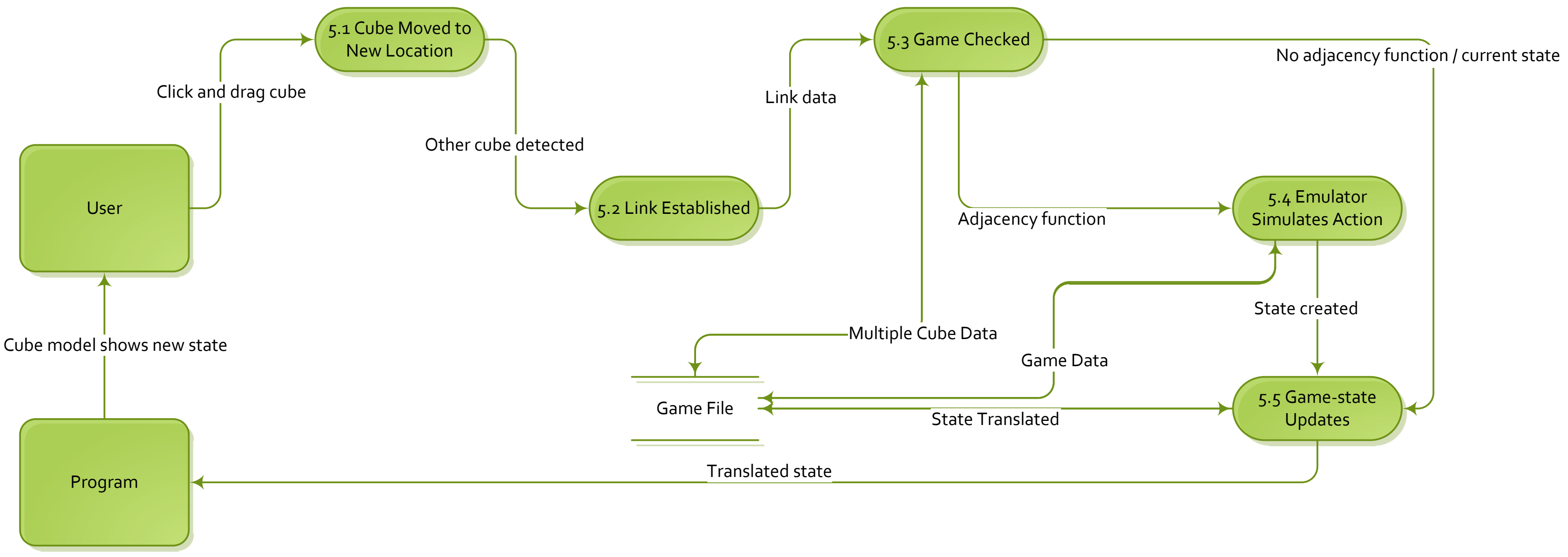
Level 1-3 Data Flow Diagram



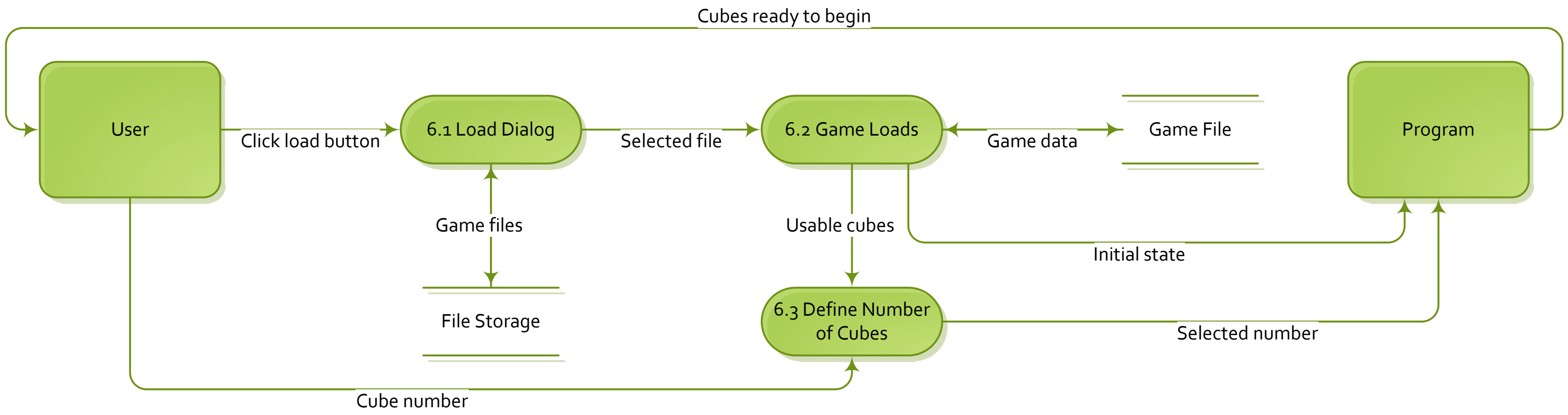
Level 1-4 Data Flow Diagram



Level 1-5 Data Flow Diagram



Level 1-6 Data Flow Diagram



11 Usability Requirements

The client would like the emulator to be accessible to new developers, but also efficient for experienced developers to use. The emulator may have a familiarization time in order for users to be able to use it productively. It will include a help menu containing instructions for the various features. In addition, the client would like keyboard shortcuts and accelerators available for users who prefer keyboard-based controls. As the user must be able to understand how to interact with the Cubes relatively quickly, the Cubes' control motions must be intuitive. The user must also be able to switch between motions quickly.

12 Performance Requirements

The performance requirements for the system are fairly basic. When the user interacts with the system there must be no visible graphics delay. Additionally, games simulated on the Cubes must function with performance similar to that of the physical Cubes. Beyond these two key requirements, actions must be performed in a timely manner for all scenarios that may arise during use of the emulator.

13 Reliability Requirements

As a system that emulates another system and runs user-created code, it is important that the Siftables Emulator does not contribute defects of its own to the application testing process. As such, Singularity Software will aim for a defect rate of 5 bugs/KLOC. As a predictor of this result, we will evaluate the cyclomatic complexity of our code at each development milestone, aiming for a score of 15 or less.[1] When evaluating the reliability of the emulator prior to release, bugs will be analyzed for priority and severity. For example, misspelled words or other aesthetic issues will not be treated with the same severity as crash-inducing bugs.

14 Supportability Requirements

Because all support for Siftables Emulator will be done by the client after the project has ended, it is imperative that the emulator have well-documented code that is easy for developers unfamiliar with the project to follow. It will also need to follow a recognized coding standard for the language chosen. As a result, it must be in an object-oriented language like Python or C# that is fairly well-known and frequently used.

15 Hardware and Software Interfaces

The Siftables Emulator will work as a standalone software package and will not interface with any other software or hardware. Singularity Software considered the integration of the emulator in an IDE but decided that the emulator was best served as an independent tool to allow easy development across the supported platforms. Similarly, the purpose of

[1] Expected cyclomatic complexity scores are based on the blog post at: http://gdub.wordpress.com/2006/07/09/cyclomatic-complexity-for-python-code/.

the emulator is to serve as a method for developing applications for the Cubes without connecting to the physical Cubes, so it will not interface with any hardware either.

16 Documentation, Installation, Legal and Licensing Requirements

The clients have stated that the emulator project is to be licensed under the BSD 2-clause license[2]; therefore, many of the standard legal and licensing issues are eliminated. Mr. Ekl has requested that Singularity Software keep documentation of development actions according to project milestone standards as well as using error tracking and version tracking systems during the design process. He has also requested that the API developed for the project is documented in a well-accepted form like that produced by Javadoc.

The emulator will be installable via an executable (on Windows) or disk image (on Mac) that will place the program's files in a user-specified directory and create shortcuts as appropriate.

17 Design Constraints

The emulator must be able to be function on computers using the Windows, Mac OS, and Linux operating systems that satisfy the minimum system requirements of the previous OS version. In other words, the emulator should run smoothly on a computer that meets the requirements for Windows Vista, Mac OS X 10.6 (Snow Leopard), or Ubuntu 11.04[3].

18 User Interface Mockups

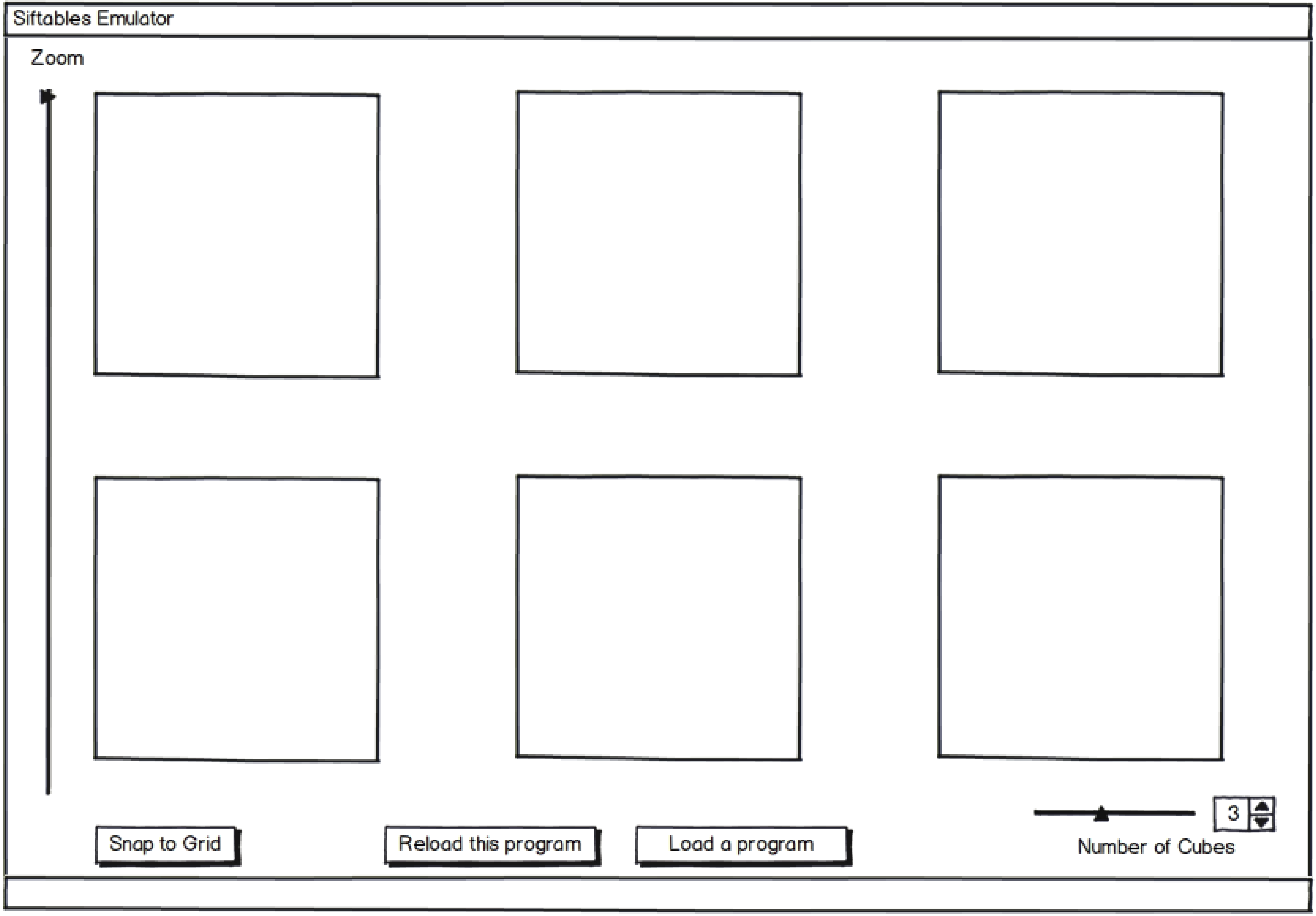
Preliminary mockups of the Siftables Emulator user interfaces are presented below. The mockups are rough sketches not intended to convey the product's final look. Rather, they present a simplified view of what Singularity thinks the emulator will look like when created in code.

In order, the mockups display the main screen of the application, the "Load a program" dialog, and the various warning and alert dialogs that the program can display.

The Manipulate Cube use case (UC6) is not easily presented in mockup format. Essentially, it applies to any of the 6 Cubes in the first mockup; any of the actions described in the use case can be applied to each virtual Cube.

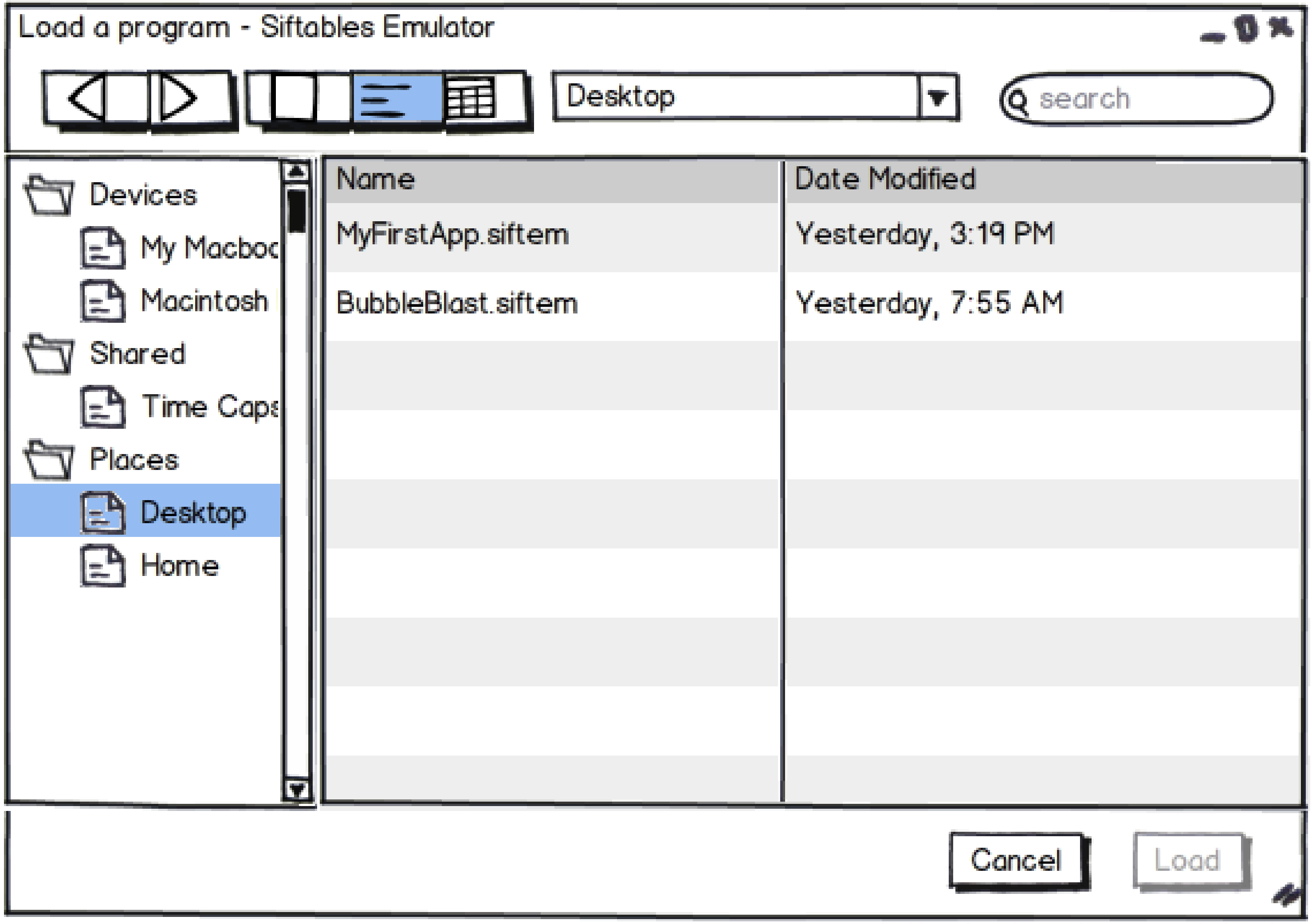
[2] http://www.opensource.org/licenses/bsd-license.php

[3] The Ubuntu machine should support graphics acceleration (i.e. it should be capable of running the GNOME 3 or Unity window managers).



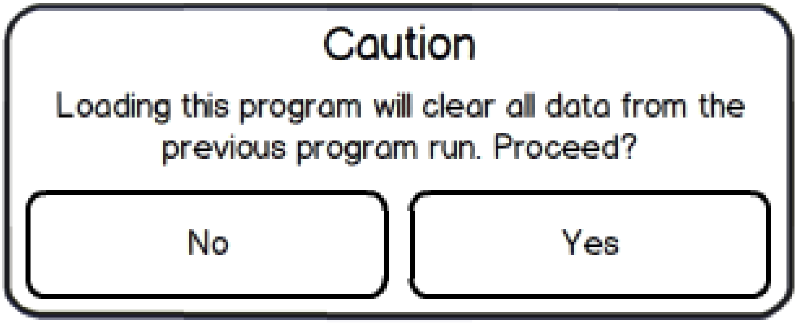
3

Number of Cubes



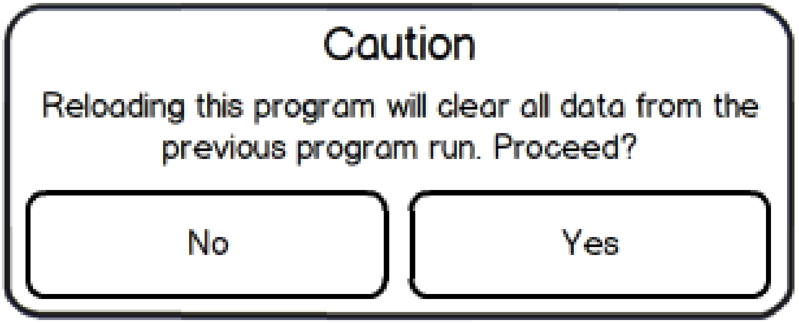
Name	Date Modified
MyFirstApp.siftem	Yesterday, 3:19 PM
BubbleBlast.siftem	Yesterday, 7:55 AM

created with Balsamiq Mockups - www.balsamiq.com



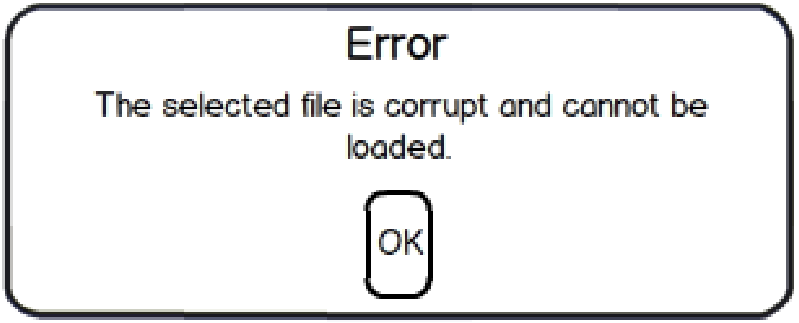
Caution

Loading this program will clear all data from the previous program run. Proceed?



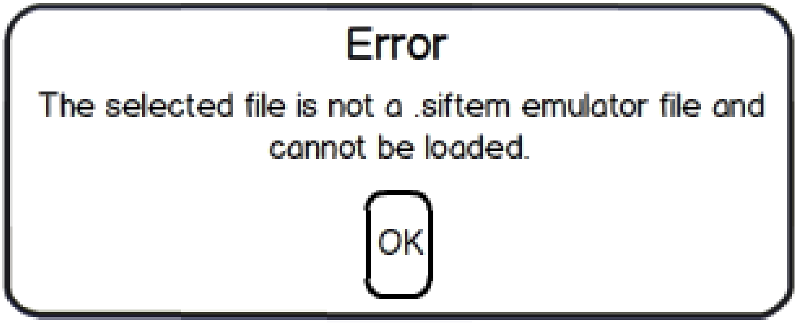
Caution

Reloading this program will clear all data from the previous program run. Proceed?



Error

The selected file is corrupt and cannot be loaded.



Error

The selected file is not a .siftem emulator file and cannot be loaded.

created with Balsamiq Mockups - www.balsamiq.com

19 Coding Standards

The developers of Singularity Software will adhere to a standardized coding "style guide" based on the language chosen for development. In the case of Python, we will reference the official Style Guide for Python Code [6]. Highlights of that document include 4-space indentation, a lack of extraneous space, and camel case for class names. In the case of C#, where no Microsoft-standard coding standards exist, we will use the standards laid out at [7], that emphasize the exclusive use of camel and Pascal case and ANSI-style bracing (where each brace gets its own line).

20 Change Control

Change requests may be submitted as issues on the project's GitHub repository[4]. At a minimum, requests for change should mention the screen(s) to be changed, the new workflow that is desired (preferably in context with how it differs from the current workflow), the rationale for the change, and the priority the submitter assigns to the change.

Changes submitted in the manner described will be considered by Singularity during our weekly project meeting. A majority vote (three of four team members) is required to accept a new feature into the system; however, a team member who feels very strongly may exercise a veto on the team's decision. Each team member will receive one veto for every four submitted changes.

Changes to product documents (i.e. milestones) will be managed using the Git version control framework within which the documents are currently stored. In the case of major changes, the team will consider the use of "delta" documents if such documents will be easier to read than heavily modified original documents.

[4] https://github.com/alexmullans/Siftables-Emulator/issues/new

21 Test Cases

21.1 Load program

Scenario #	Originating Flow	Alternate Flow	Next Alternate
1	Basic flow		
2	Basic flow	Alternate flow 1	
3	Basic flow	Alternate flow 2	
4	Basic flow	Alternate flow 3	
5	Basic flow	Alternate flow 4	
6	Basic flow	Alternate flow 4	Basic flow

Test Case ID	Scenario	Description	Expected Result
1	1	A program is successfully loaded in the emulator.	The Cubes run the selected program in the Workspace.
2	2	User selects incompatible file type.	"The selected file is not a .siftem emulator file and cannot be loaded." error is displayed.
3	3	User selects corrupt or unloadable file.	"The selected file is corrupt and cannot be loaded." error is displayed.
4	4	User cancels loading program.	The emulator returns to its state before the basic flow was entered.
5	5	User loads a program when a program is already running and chooses "Cancel" on the warning prompt.	Emulator returns to its state before the basic flow was entered after "Cancel" is chosen.
6	6	User loads a program when a program is already running and chooses "Continue" on the warning prompt.	Cubes run the selected program in the Workspace after "Continue" is chosen.

21.2 Reload program

Scenario #	Originating Flow	Alternate Flow
1	Basic flow	
2	Basic flow	Alternate flow 1

Test Case ID	Scenario	Description	Expected Result
1	1	The program is successfully reloaded in the emulator.	The Cubes run the selected program in the Workspace after the user presses “Continue” on the warning message.
2	2	The user selects “Cancel” on the warning dialog (which specifies that the emulator will be reset).	The Cubes run the program as originally specified; no change should be made to emulator’s state.

21.3 Zoom screen

Scenario #	Originating Flow
1	Basic flow

Test Case ID	Scenario	Description	Expected Result
1	1	User zooms in fully.	The Workspace view is centered on the first cube with the edges of the adjacent Cubes visible.
2	1	User zooms out fully.	The entire Workspace is visible.
3	1	A zoom level in the middle is selected.	The slider moves to the closest predefined increment and the correct zoom level is shown.

21.4 Add/remove Cubes

Scenario #	Originating Flow	
1	Basic flow	

Test Case ID	Scenario	Description	Expected Result
1	1	The slider is moved to one of the predefined increments or the spinbox is changed to an integer in the range [1, 6].	The specified number of Cubes is shown in the Workspace.

21.5 Snap Cubes to grid

Scenario #	Originating Flow
1	Basic flow

Test Case ID	Scenario	Description	Expected Result
1	1	User presses the “Snap to Grid” button.	The emulator aligns the Cubes in a grid based on their current positions.

21.6 Manipulate Cube

Scenario #	Originating Flow
1	Basic flow
2	Alternate flow 1
3	Alternate flow 2
4	Alternate flow 3

Test Case ID	Scenario	Description	Expected Result
1	1	The user double clicks on a Cube.	The Cube responds as if a screen click occured.
2	2	The user clicks on one of the buttons on the Cube.	The Cube executes and responds to the associated action (flip, rotate, or tilt).
3	3	The user drags a Cube next to another Cube.	The Cube neighbors with the adjacent Cube, and the two recognize each other as neighbors.
4	4	The user drags a Cube back and forth in a shaking manner.	The Cube responds as if shaken.

22 Usability Report

22.1 Process

In order to conduct the usability test, a basic prototype of the Siftables Emulator was developed using Adobe Flash CS5. The prototype, which modeled the basic functionality intended for the final emulator, was then compared to the current Sifteo Inc. emulator, the Siftulator. The tests were conducted using CSSE students and faculty as subjects, because the primary target demographic of the software is developers.

Participants in the study were first presented with an informed consent form and a pre-test questionnaire that collected relevant personal information. Upon completion of these two documents, the user was then taken into the testing area and given a brief set of verbal instructions by one of the team members. After the team member finished presenting the instructions and left the room, the participant was asked to begin the study by watching a short video demonstrating the capabilities of the Sifteo Cubes. We felt that this was the most efficient way to acquaint users with the product and we believed that it was necessary for them to be able to accurately judge the performance of an emulator.

Upon completion of the video, participants were asked to complete a series of simple tasks in the current Siftulator platform. Because of the nature of this product and its lack of a help file, a list of action commands from Sifteo's website was located on the table next to the user. If they could not intuitively determine the correct way to complete a task, participants were told that this sheet could be used to help them. Users were then asked to switch to Siftables Emulator prototype and reminded that this product was in no way based on the software which they had just used. They were then asked to complete a similar set of tasks using the prototype. In the event that the user became stuck and was unable to complete their task at any time on either emulator, help was offered. This help was kept to a minimum, often consisting solely of asking the user a question to help them cogitate in a different fashion.

At the conclusion of the technical portion of the study, participants were thanked for their time and assistance and asked to complete a post-test questionnaire. This questionnaire was designed to allow each user to express his or her opinion about the various aspects of both projects and suggest possible improvements to the interface of the Siftables Emulator prototype.

22.2 Analysis

The results from the usability study can be found primarily in the usability report generated by the Ovo software.

Additionally, a pre-test questionnaire and a post-test questionnaire were distributed to each participant to collect data regarding their overall experience as well as their prior knowledge of the product and similar systems. The informed consent form, pre-test questionnaire, and post-test questionnaire are all included as appendices at the end of this document. The results of the questionnaires are shown in the table following this section. For questions utilizing a Likert scale for answers, the answers were associated with numeric answers, with the most positive answer corresponding to the most positive value.

22.2.1 Pre-Test Results

1. All of the participants reported that they had never heard of Sifteo nor their product. This lack of previous experience working with the product provided a clean slate on which to base our findings.

2. All of the participants but one reported having never programmed for an embedded system before. This question does not directly reflect in any of the results, but it indicates that the emulator is still accessible to first-time programmers for embedded systems.

3. **2.5/4.0**: On average the participants felt somewhere between comfortable and uncomfortable using new software without documentation. As that is the exact median of the possible values, these results are expected.

4. **3.8/4.0**: With documentation, all participants felt at least somewhat comfortable, and all but one reported feeling very comfortable using new software. Because

the participants chosen were all from or had experience in the field of computer science/engineering or software engineering, this is expected.

5. **3.8/4.0**: Similar results were found with the participants' comfort level with using an API.

6. Half of the participants had used an emulator before, and half had not, indicating that our results would accurately depict the user experience on both ends of the spectrum.

22.2.2 Study Results

Results from the usability study with the six participants are available in the usability report. Overall, the times associated with the tasks for Singularity's emulaor were significantly lower than those associated with the tasks for Sifteo's emulator.

One notable difference was in the "Load program" task. Using Sifteo's emulator, a user has to open a separate program loader, and navigating to the "Play" functionality is not intuitive. Additionally, there is not much immediate feedback regarding the success of loading a program. On the other hand, in Singularity's emulator, loading a program was literally as easy as the click of a button, making the task much more straightforward.

22.2.3 Post-Test Results

1. **3.8/4.0**: All but one participant strongly agreed that the main screen of the emulator was accessible. The other participant agreed that the main screen was accessible but did not strongly agree, indicating that the layout works well for efficient use.

2. **3.7/4.0**: All but two participants strongly agreed, and the other two agreed, that the commands were in easy-to-find locations, reinforcing that the layout of the main screen was well-designed.

3. **3.1/4.0**: Moving a Cube was not as straightforward; on average, users agreed that it was easy, but one participant strongly disagreed. Because the moving functionality was not fully implemented in the prototype but participants found it easy to use in Sifteo's emulator, we can ensure that the functionality will be easy to use because we plan on implementing movement in the same way.

4. **3.4/4.0**: Organizing Cubes on the screen on average was determined to be halfway between agreeing and strongly agreeing. Because only one Cube was present in the emulator, some found it less intuitive because they did not receive visual feedback, which is understandable and does not heavily indicate any lack of usability in the product.

5. **3.3/4.0**: Manipulating a Cube on average was indicated to be fairly intuitive, but not all functionality was straightforward or immediately obvious. As the study indicated, shaking and tilting took considerably longer than the other manipulations; some users found the icons associated with these movements unintuitive.

6. **3.2/4.0**: On average, the participants agreed that Singularity's emulator was easier to use than Sifteo's. The participants who found Sifteo's emulator easier to use made

extensive use of the help sheet provided, so they were able to complete the actions more quickly.

Gender	**Major**	**Class**	**Pre1**	**Pre2**	**Pre3**	**Pre4**	**Pre5**	**Pre6**	**Post1**	**Post2**	**Post3**	**Post4**	**Post5**	**Post6**
F	CS	S	A	No	1	4	4	No	4	4	1	2	3	3
F	CE	F	A	No	1	3	4	Yes	3	4	3	4	3	4
M	SE/CS/MA	S	A	No	3	4	3	Yes	4	4	4	4	3	2
M	SE	J	A	No	4	4	4	Yes	4	3	4	4	4	2
M	CSSE	Fac	A	No	3	4	4	No	4	3	2.5	2.5	3	4
M	CPE	J	A	Yes	3	4	4	No	4	4	4	4	4	4
Average					2.5	3.8	3.8		3.8	3.7	3.1	3.4	3.3	3.2

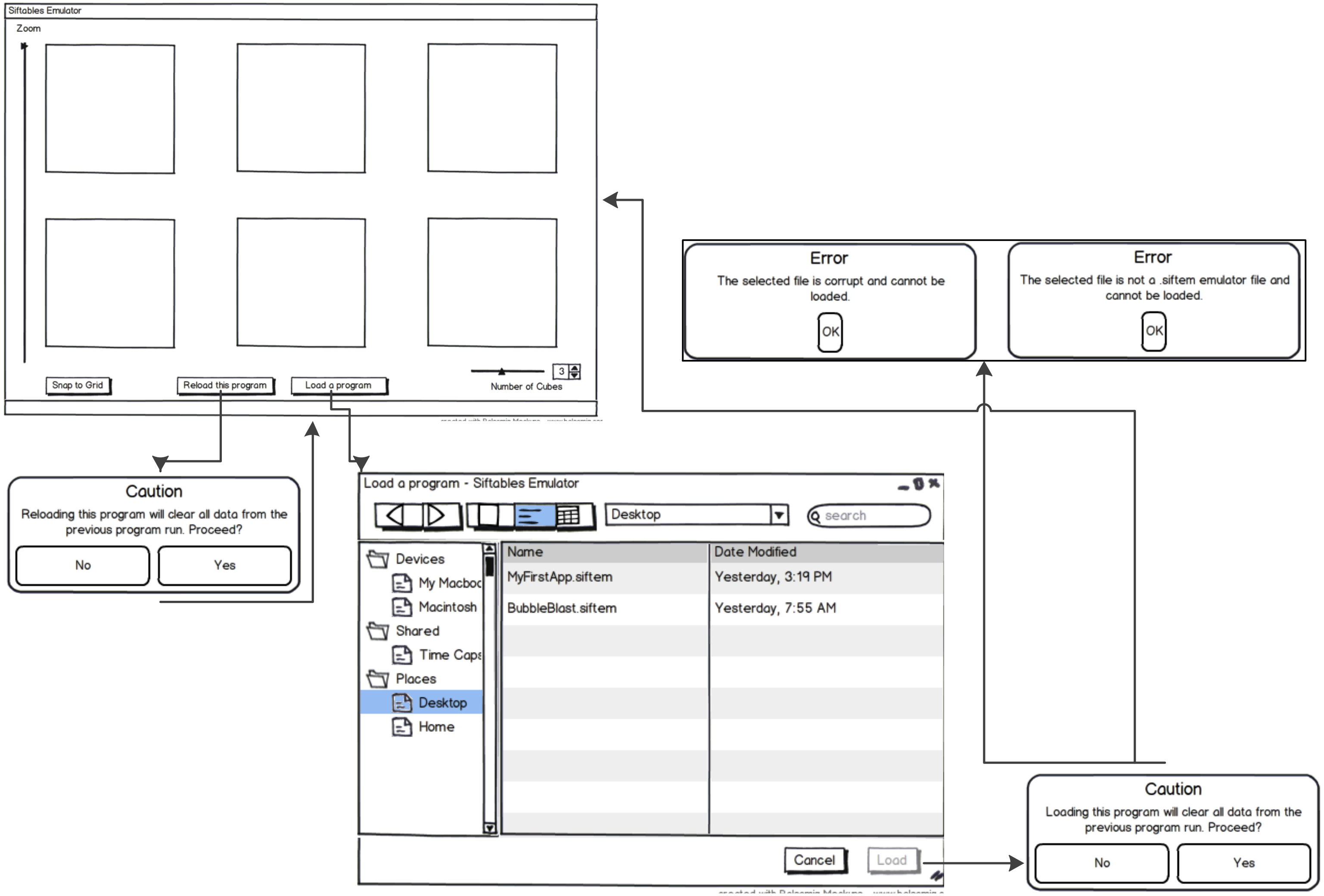
22.3 Findings

Based on the post-test results, our emulator was regarded as having an accessible layout and fairly intuitive controls. All the users at least agreed that the commands were in easy-to-find locations and that the main screen looked accessible.

One aspect that the users saw could be improved is manipulating the Cube. While rotating and flipping were not an issue for most users, they found tilting and shaking to be more difficult to comprehend. For this reason, we made the edges of the Cube bolder to help inform the user where to grab the Cube for tilting and also changed the shake icon to be more recognizable. However, since most of the users first instinct to shake the Cube was to drag it with the mouse, the shake icon is temporary and we plan to implement mouse control for the shake command in the future.

Users, on average, completed the designated tasks quicker and with less help in comparison to Sifteos emulator. Overall, the emulator was viewed as an improvement over Sifteos emulator by outperforming it in various tasks such as adding/removing Cubes, loading programs, and controlling the Cubes.

23 Interaction Architecture



Name	Date Modified
MyFirstApp.siftem	Yesterday, 3:19 PM
BubbleBlast.siftem	Yesterday, 7:55 AM

Caution

Reloading this program will clear all data from the previous program run. Proceed?

No Yes

Error

The selected file is corrupt and cannot be loaded.

OK

Error

The selected file is not a .siftem emulator file and cannot be loaded.

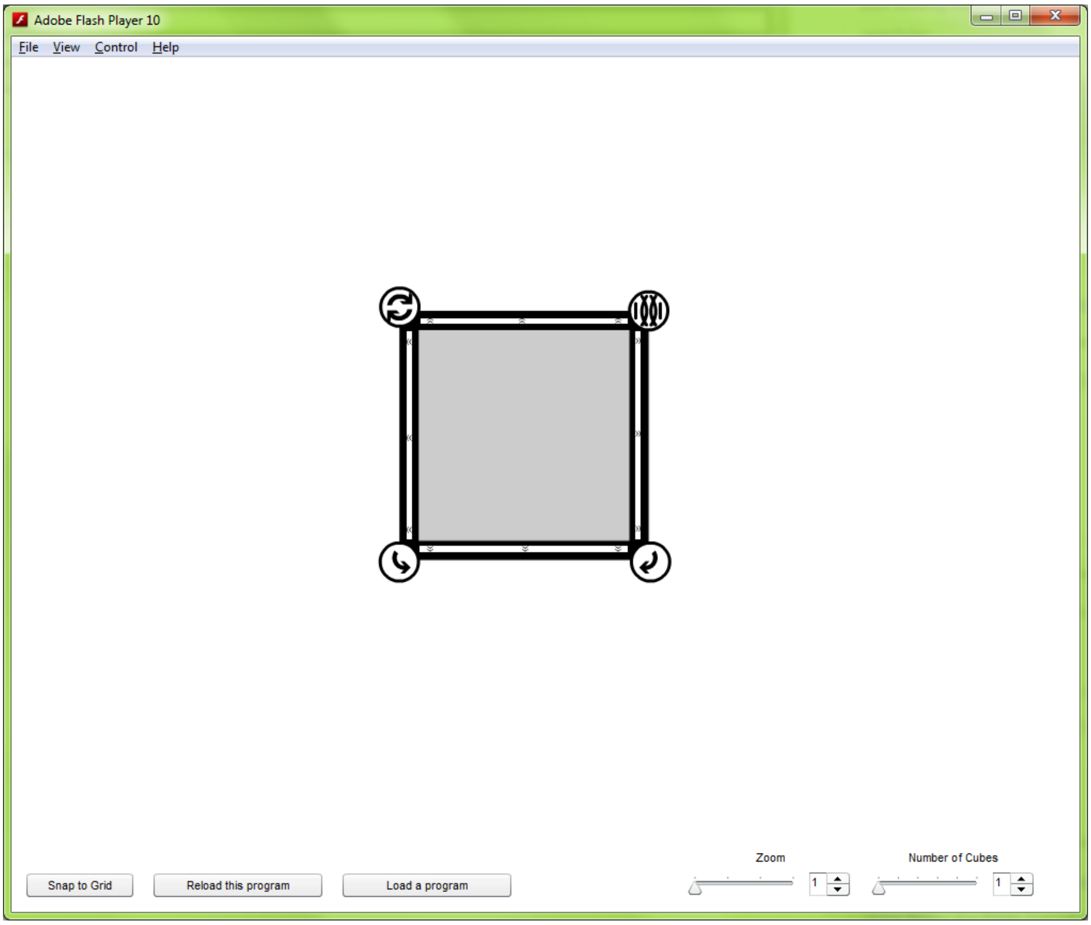
OK

Caution

Loading this program will clear all data from the previous program run. Proceed?

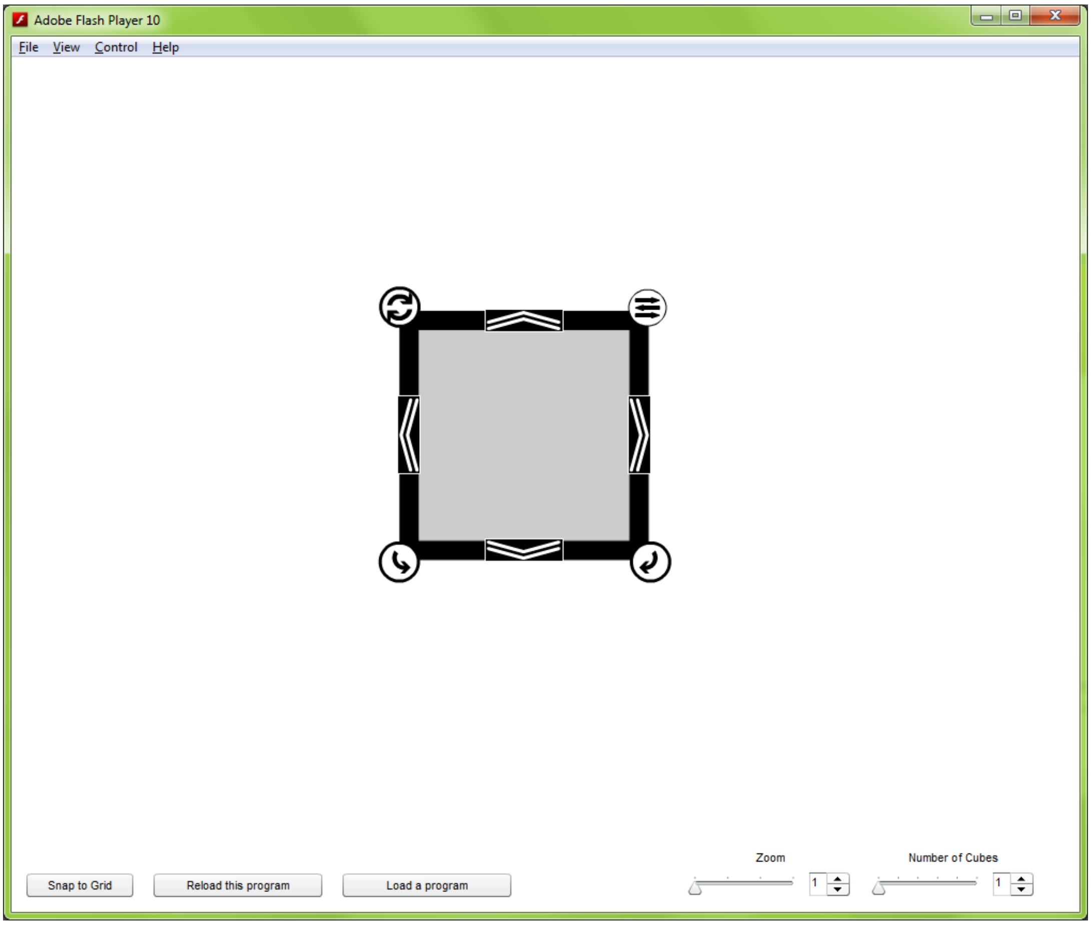
No Yes

24 Initial Interface Design



Singularity's initial interface design was well-received. Usability testers were able to easily access all presented functions except tilt and shake.

25 Revised Interface Design



Based on the results of the usability test, Singularity redesigned parts of the interface to make the tilt and shake actions more obvious. Specifically, we replaced the tilt buttons on each side of the cube with larger, clearer instances. Additionally, we redesigned the shake button to increase its clarity and better relate the icon to the button's purpose.

Glossary

Application Programming Interface is an interface implemented by a software program that enables it to interact with other software. 4

cross-platform support is an attribute given to software implemented and operable on multiple computer platforms. 5

Graphical User Interface is a visual way of allowing the user to interace with a computer program. 10

Integrated Development Environment is software that provides a comprehensive work environment for computer programmers and software developers. 6

issue tracking system is a piece of software used to maintain a list of issues as generated during a project. 11

Linux is a Unix-based operating system based on free and open source software. 5, 6, 11, 25

Mac is a series of lines of personal computers developed by Apple. 5, 6, 11, 25

Object-Oriented Programming is a programming paradigm using objects to design applications. 5

open source is an attribute given to software for which the source code is freely available. 7, 8, 11

Sifteo Cubes are small machines capable of loading programs and interacting with one another as well as responding to predefined movements. 4

version control is the management of documents and programs for a project over many versions in a well-organized manner. 11

Windows is a series of operating systems developed by Microsoft. 5, 6, 11, 25

References

1. Sifteo Inc. Online: http://www.sifteo.com

2. Tim Ekl. Client Meeting. 25 October 2011 2:30 p.m.

3. Milestone 1. Singularity Software. Online: https://github.com/alexmullans/Siftables-Emulator/blob/master/docs/pdfs/m1.pdf

4. Milestone 2. Singularity Software. Online: https://github.com/alexmullans/Siftables-Emulator/blob/master/docs/pdfs/m2.pdf

5. Milestone 3. Singularity Software. Online: https://github.com/alexmullans/Siftables-Emulator/blob/master/docs/pdfs/m3.pdf

6. Milestone 4. Singularity Software. Online: https://github.com/alexmullans/Siftables-Emulator/blob/master/docs/pdfs/m4.pdf

Index