Singularity Software
Milestone 2

October 5, 2011

By signing below, I approve the contents of the following document.

Alex Mullans _____

Ruben Rodriguez _____

Ethan Veatch _____

Kurtis Zimmerman _____

Contents

1 Executive Summary

This document is the second in a series of milestone documents that will accompany the planning of the Siftables Emulator. The Emulator project is a first of its kind application that will allow developers of Sifteo applications to test the features of the Cubes in a virtual programming environment. Currently, the only way to test apps developed for the Sifteo Cubes platform is with the physical Cubes; this project will eliminate that need and serve as a demo for the possibilities of the Sifteo platform.

This milestone elaborates the functional features of the the Siftables Emulator project. It first gives a brief background of the project before moving on to elaborate features with use cases and declarative statements, where appropriate. It also maps all covered use cases to their relevant features. Finally, it presents mockups of the Emulator's design for consideration and to solicit client feedback. Future milestones will present plans for change control, coding standardization, and testing. Finally, design and usability reports will make up the core of milestones near the end of the quarter as the software stabilizes.

2 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.

Milestone 2 lays the foundation of the Siftables Emulator specification based on the requirements gathered in Milestone 1. It will be supplemented by the specification and prototypes in Milestone 3. Milestone 4 will rely on these early milestones as they define a change control plan and test cases, and Milestone 5 will elaborate the usability guidelines and interface design that implement the features and use cases described herein.

3 Project Background

The Siftables Emulator is being developed by Singularity Software as part of the Junior Project sequence of classes at Rose-Hulman Institute of Technology. When projects were solicited for the sequence, clients Tim Ekl and Eric Stokes (both Rose-Hulman alumni) submitted a request for an emulator for Sifteo Cubes, a new platform intended for "intelligent play." After Singularity was chosen for the project, we met with Mr. Ekl to determine the three primary features of the Emulator: a Workspace where 1-6 Cubes could mimic the manipulations possible with physical Cubes, an API to program those virtual Cubes, and a set of example games designed to show off the first two features. Singularity's Emulator will be the first program of its kind on the market for Sifteo Cubes.

A full feature listing and description can be found in Appendix A.

4 Use Cases

4.1 Load program

Name: Load program

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

Actors: User

Basic flow:

1. User presses “Load a program” button.
2. User selects *.siftem file in file dialog.
3. User presses “Open” button.
4. Emulator loads program on Cubes in emulator.

Alternate flows:

When the user opens an incompatible file (i.e. any file without the .siftem extension), or
When the user opens a corrupt or otherwise unloadable file, or
When the user presses “Cancel” button:

1. An error dialog summarizing the issue is presented to the user.
2. The use case terminates and no program is loaded.

When the user is already running a program,

1. A warning dialog indicates to the user that the existing program state will be lost if a new program is loaded.
2. If user presses “Continue” on dialog warning that the existing program state will be lost if the program is reloaded, flow returns to Step 2 of the Basic flow.

When User presses “Cancel” on warning dialog:

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

Pre-conditions:

1. Emulator is running.

Post-conditions:

1. Program is loaded or user cancelled loading.

Special requirements

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

4.2 Reload program

Name: Reload program

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

Actors: User

Basic flow:

1. User presses "Reload this program" button.
2. User presses "Continue" on dialog warning user that the existing program state will be lost if the program is reloaded.
3. Emulator loads program on Cubes in emulator.

Alternate flows:
When User presses "Cancel" 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. Program is reloaded or current program remains loaded on the cubes.

Special requirements:

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

4.3 Zoom screen

Name: Zoom screen

Description: User zooms the Workspace to the desired level.

Actors: User

Basic flow:

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

Alternate flows:
None

Pre-conditions:

1. Emulator is running.

Post-conditions:

1. Program running at beginning of use case, if any, is still running.
2. 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.

4.4 Add/remove Cubes

Name: Add/remove Cubes

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

Actors: User

Basic flow:

1. 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. Emulator adds/removes Cubes in emulator and resets emulator 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. Emulator is running.

Post-conditions:

1. Number of Cubes has been adjusted to number specified.
2. Program running, if any, is terminated.

Special requirements:

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

4.5 Snap Cubes to grid

Name: Snap to grid

Description: Users pulls the cubes into a grid orientation.

Actors: User

Basic flow:

1. User presses "Snap to Grid" button.

2. Emulator moves Cubes to a grid orientation based on their current positions. It will maintain Cubes' positions relative to other Cubes while doing so.

Alternate flows:
None

Pre-conditions:

1. Emulator is running.

Post-conditions:

1. Cubes are arranged in a grid.

4.6 Manipulate cube

Name: Manipulate cube

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

Actors: User

Basic flow:

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

Alternate flows:

1. User clicks on buttons superimposed on cube edges.
2. Cube executes appropriate action (i.e. flips, rotates, tilts).

1. User drags cube next to another cube.
2. Cube communicates ("neighbors") with the cube(s) it is adjacent to.

Pre-conditions:

1. Emulator is running.

Post-conditions:

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

4.7 Other functional requirements

4.7.1 API

The emulator will include an API in order to define a set of rules and specifications for the software.

4.7.2 Example games

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

5 Use Case Feature Mapping

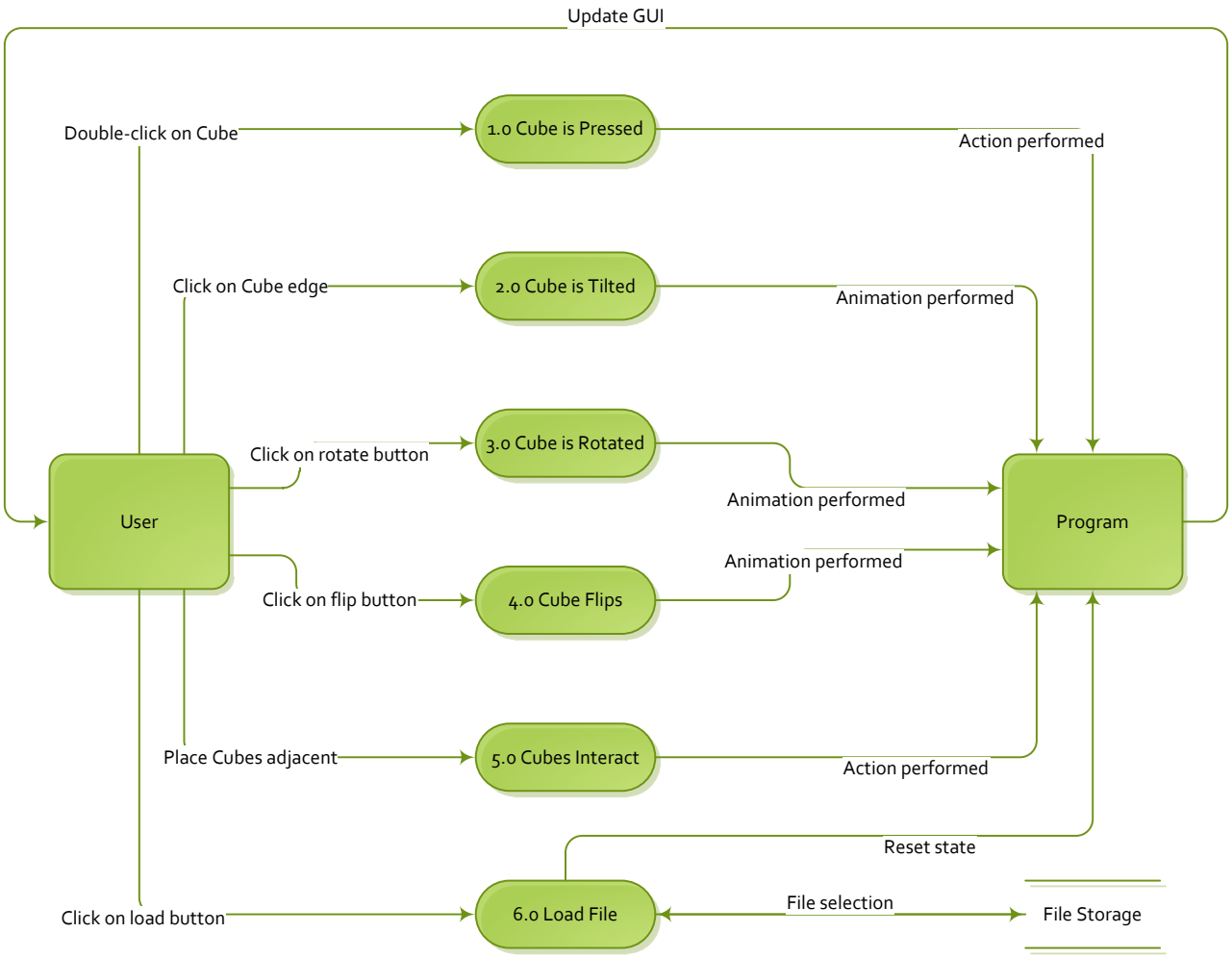
The feature listing can be found in appendix A, and the use case IDs refer to the use cases specified above.

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

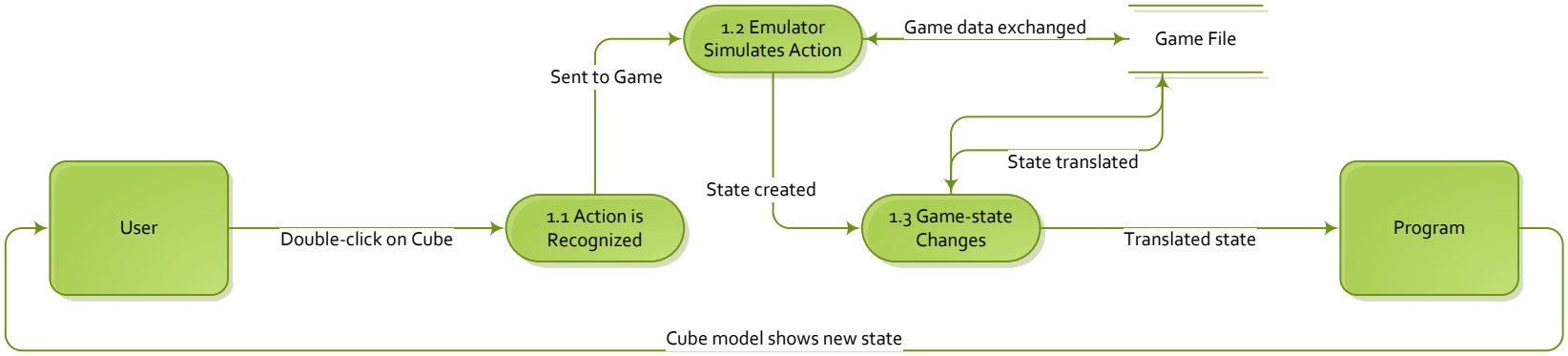
6 Work/Data Flow Diagrams

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.

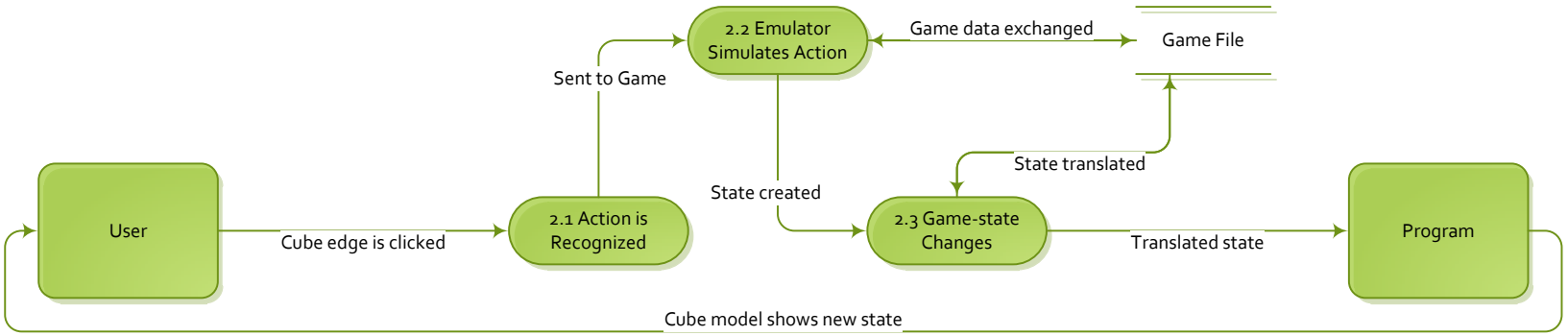
Level 0 (Context) Work/Data Flow Diagram



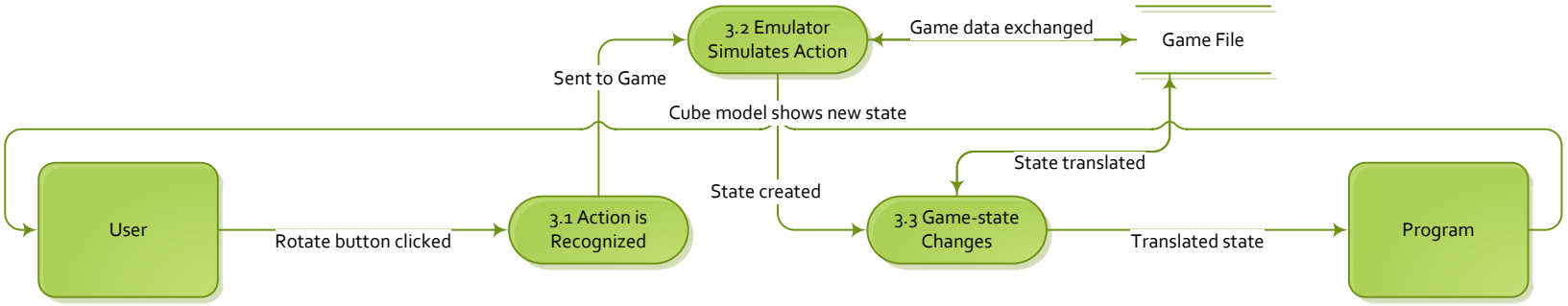
Level 1-1 Data Flow Diagram



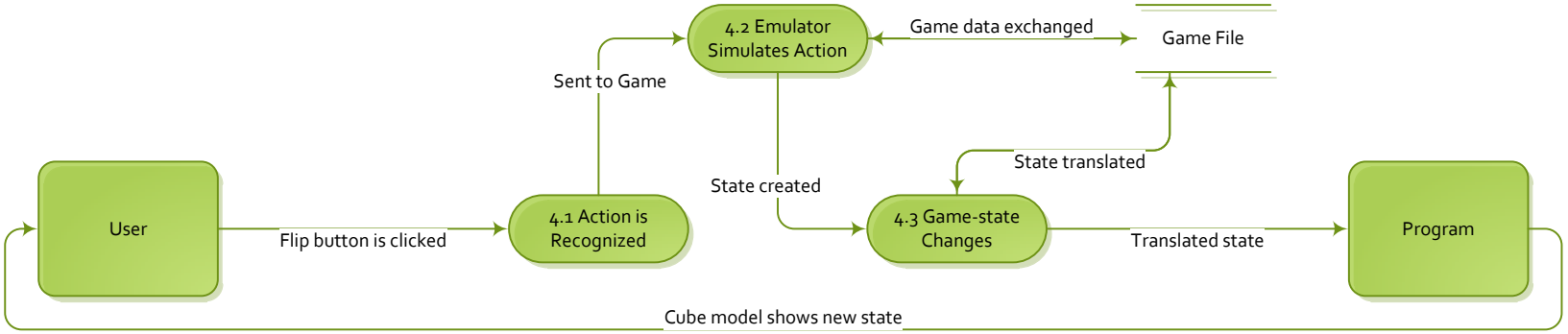
Level 1-2 Data Flow Diagram



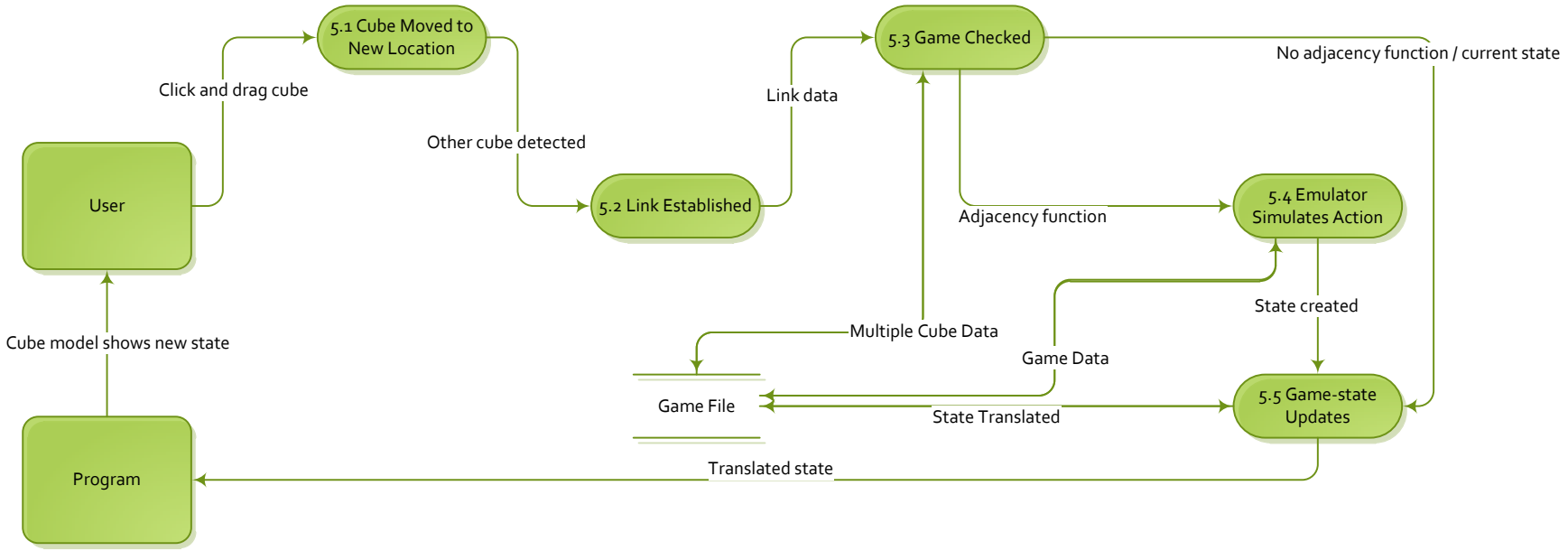
Level 1-3 Data Flow Diagram



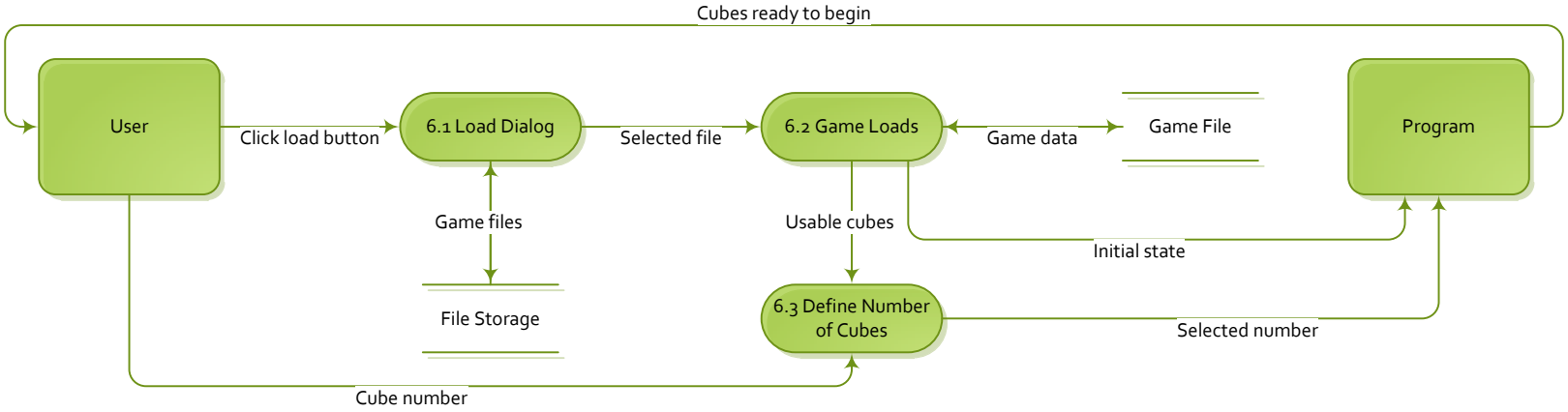
Level 1-4 Data Flow Diagram



Level 1-5 Data Flow Diagram



Level 1-6 Data Flow Diagram



A Features

ID	Feature	Description	Reason
F1	Individual, virtual Sifteo Cube	A virtual representation of a single Sifteo cube	Replicates physical Sifteo Cube
F2	Buttons to manipulate each virtual Cube	Buttons on the virtual Cube will allow the user to flip and tilt it	Replaces physical actions where said actions would be impractical with a mouse
F3	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	Replicates multiple Sifteo Cubes in a natural, free-form environment
F4	Interactions between Cubes	The Cubes present on the workspace will communicate when they are neighbored	Cubes can simulate the interactions possible with physical Cubes
F5	Load programs into the Cubes	The user will load his own and example programs into the emulators Cubes	The ability to program programs for the emulator is dependent on a common interface
F6	Snap Cubes to invisible grid	The Cubes will snap into an invisible grid when a button is clicked	Increases productivity by allowing a quick reset if the Cubes are in disarray
F7	Zoom Workspace	The Workspace will zoom to the level of an individual Cube or the whole space	Inspecting individual Cubes allows for precise checks of program Graphical User Interfaces (GUIs)

Glossary

Application Programming Interface is an interface implemented by a software program that enables it to interact with other software. 3

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

Sifteo Cubes are small machines capable of loading programs and interacting with one another as well as responding to predefined movements. 3

References

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

2. Tim Ekl. Client Meeting. 12 September 2011 12:45 p.m.

Index