Siftables Emulator
Singularity Software
Milestone 5

February 8, 2012

Alex Mullans
Ethan Veatch
Eric Vernon
Kurtis Zimmerman

Contents

1 Project Background

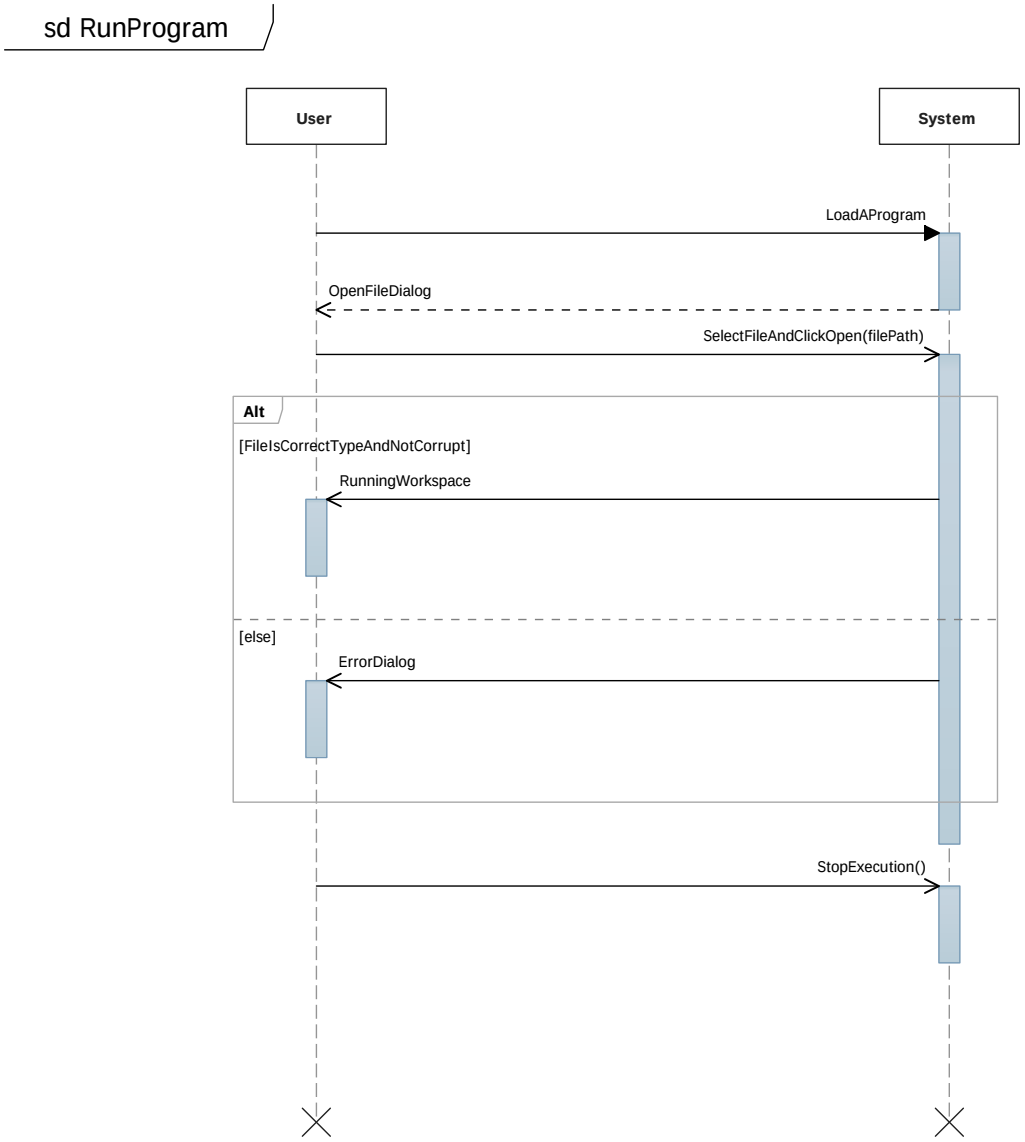
The Siftables Emulator is being developed by Singularity Software as part of the Junior Project sequence of classes at Rose-Hulman Institute of Technology. When projects were solicited for the sequence, clients Tim Ekl and Eric Stokes (both Rose-Hulman alumni) submitted a request for an emulator for 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 interface through which to program those virtual Cubes, and a set of example games designed to show off the first two features. Singularity's Emulator is intended to build on the foundation of Sifteo, Inc.'s existing emulator by creating a more fluid and natural user interface.

The clients' only implementation-specific specification was the ability to run the finished emulator on a Mac.

2 Analysis Models

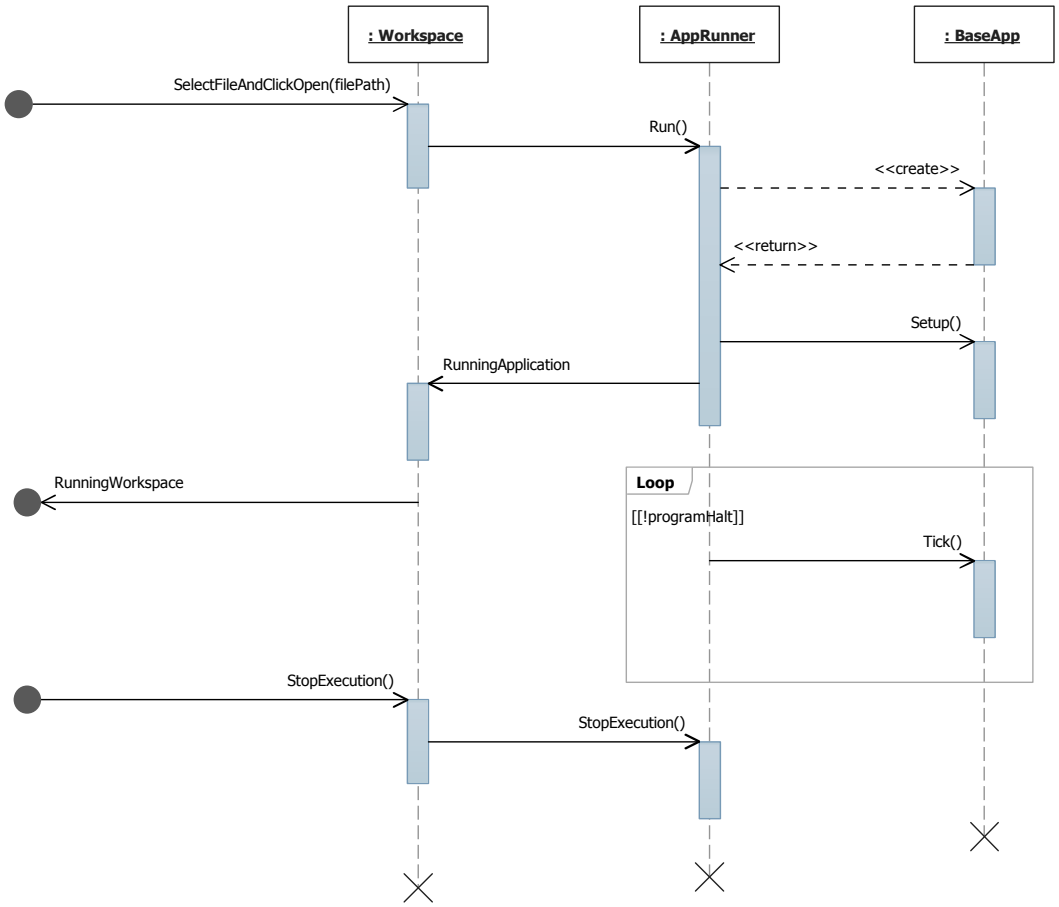
2.1 System Sequence Diagram

The initial RunProgram system sequence diagram had a duplicate step which was annihilated by replacing the *FileOpened* result with the *RunningWorkspace* result because the two were essentially duplicates.

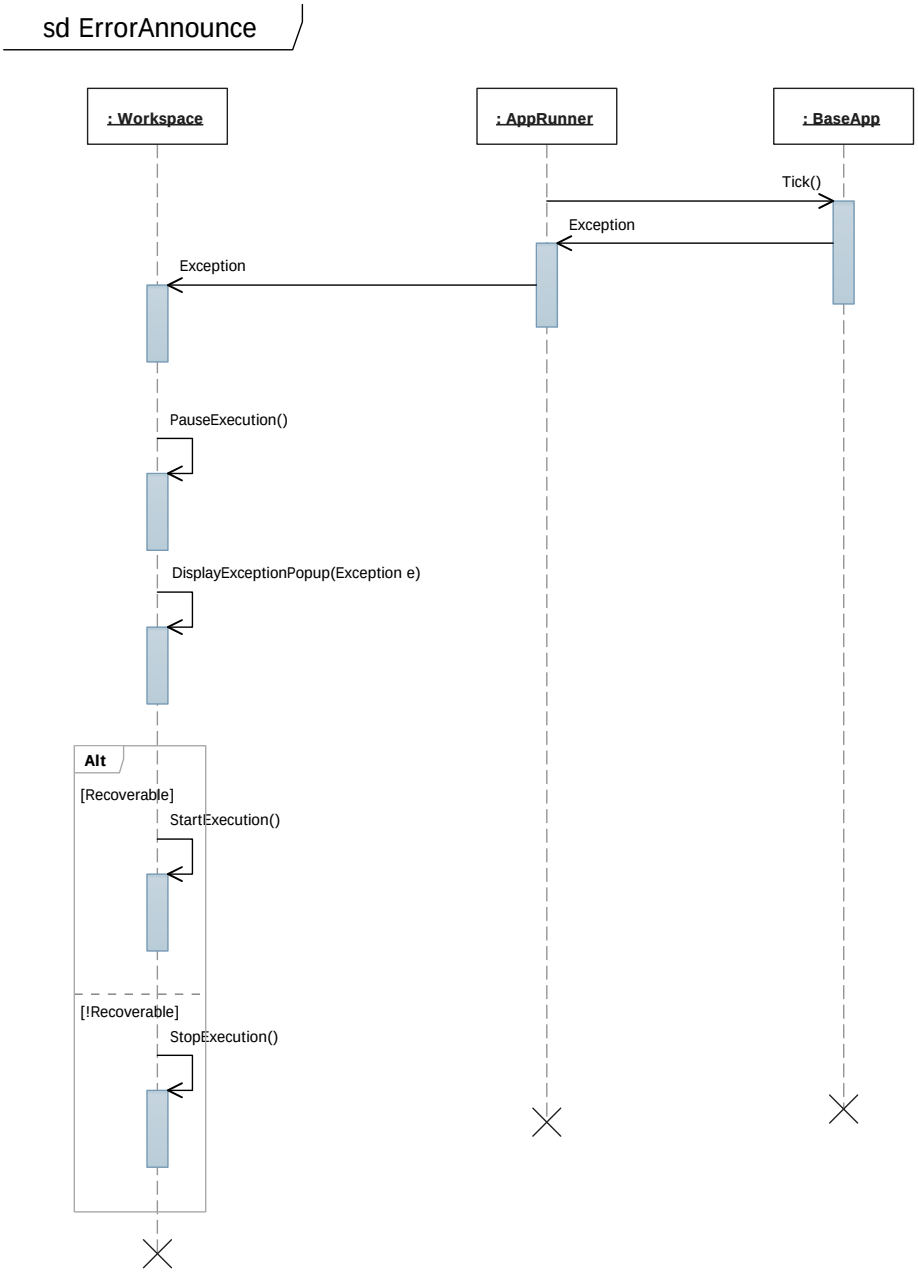


2.2 Sequence Diagrams

The *Setup()* operation is necessary in addition to a constructor because this is the way in which initialization of an application is handled per Sifteo's API. An *AppRuner* entity was added to illustrate the way in which our current system handles application execution and to show that the call to *StopExecution* in the workspace propagates to the *AppRunner* object.

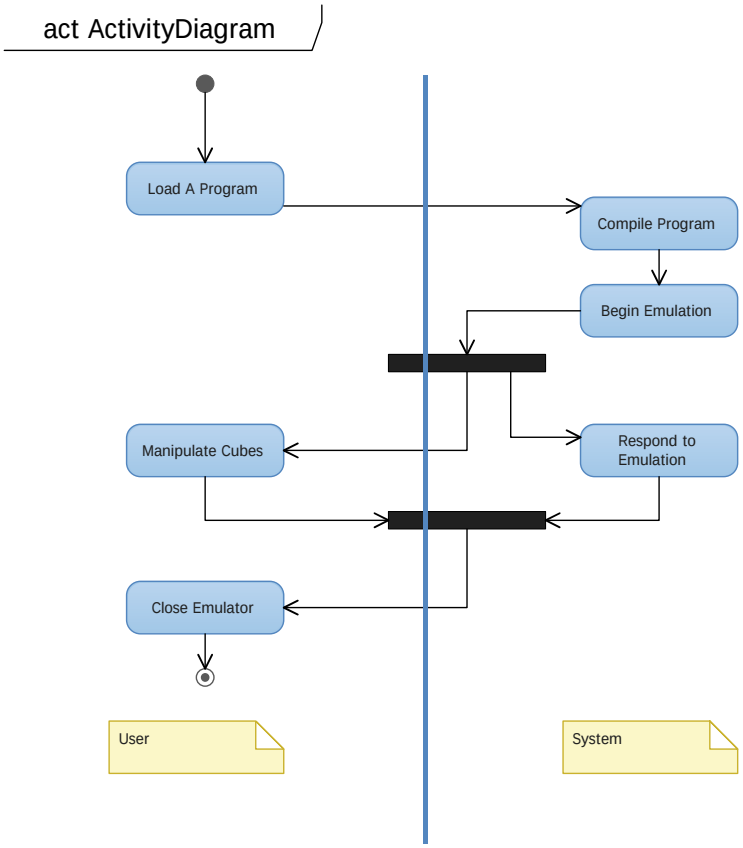


The *Tick()* operation was unintentionally omitted from the original diagram, and it has been added for clarity to indicate that the action is taking place during an actively running application.



2.3 Activity Diagram

The below activity diagram details the process beginning with the user loading a program to the point at which the user closes the program.



2.4 Design Class Diagram

Any external relationships our shown beneath the class name to emphasize interaction with classes we did not create. For instance, each *Cube* is implemented as a *UserControl* object.

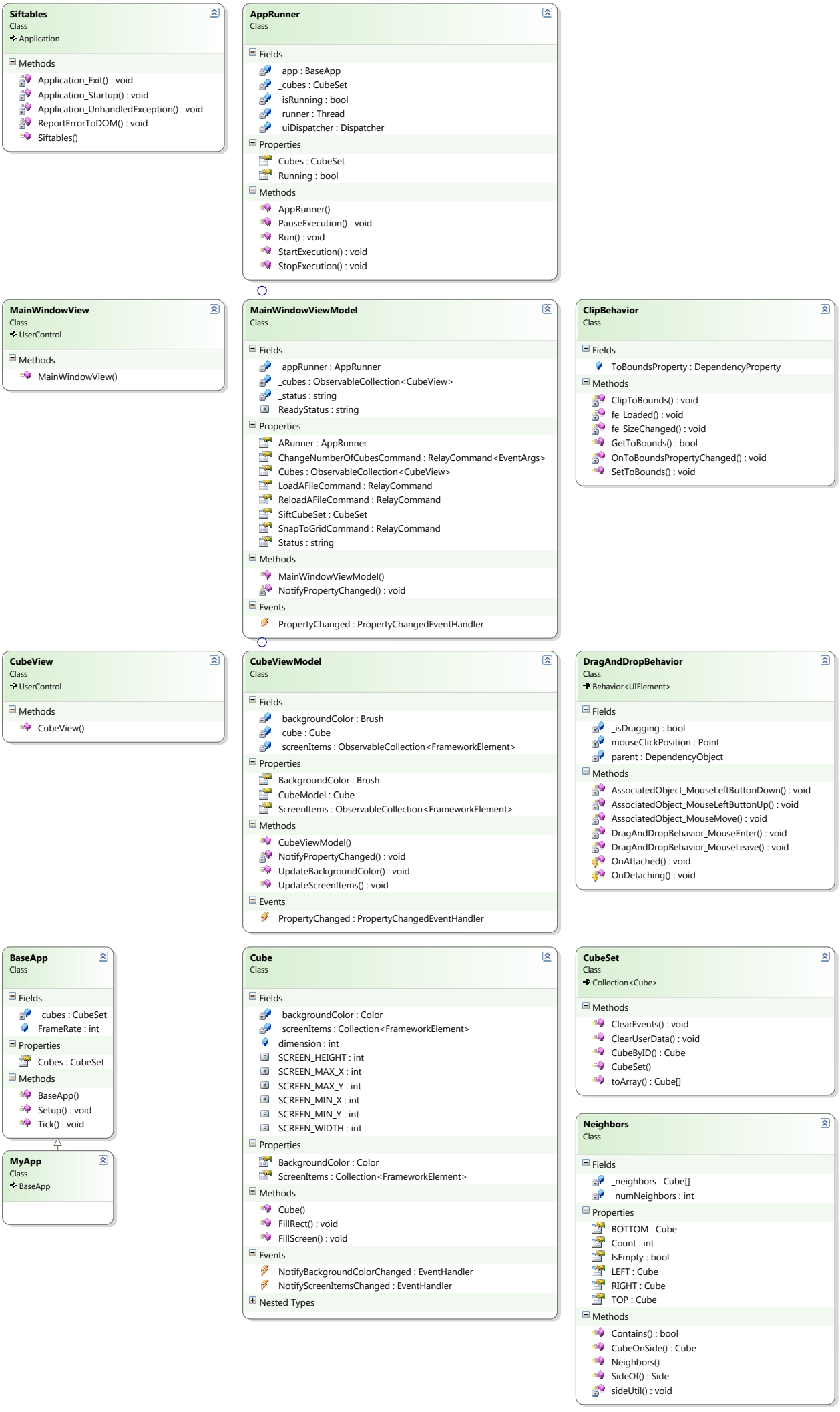
Additionally, every entity in our diagram is being implemented in our own original code, though many classes are inspired by the Sifteo API which we have to mirror. Those entities which do mirror the objects in the Sifteo domain will be separated into a distinct package to clarify the divide between those objects which carry out the emulation and those which the user primarily manipulates in their original applications.

Because we are mirroring the structure of Sifteo's API as well as the methods and properties available for each entity in the domain, the process by which a developer translates their application to a usable state for our emulator should be relatively seamless. Instead of building their application against Sifteo's libraries, the developer would build their application against our Sifteo domain level library. Then this compiled assembly could be loaded in to the emulator, and the proper method calls to run the application could

be made by the AppRunner object. At this time, there are no libraries available which support compiling C# code within a Silverlight application, so dynamic compilation is not a feasible option.

When an application is loaded, the process by which it is run within the context of the workspace requires another thread of execution. The original thread carries out all requirements to constantly update the UI and handle events at the UI level, while a newly created thread executes the code in the application, and anytime code specifically makes a change which would affect the UI, it must be handled by the UI dispatcher to which the AppRunner has a reference. By this process, running an application appears seamless and emulates the execution of the Sifteo cubes.

Note: Because the class diagram was generated by Visual Studio, no associations were created (for instance, between the entity *Cube* which is associated with a unique *Neighbors* entity). We feel no clarity was lost by ommitting these associations.


Siftables
Class
Application
Methods
Application_Exit() : void
Application_Startup() : void
Application_UnhandledException() : void
ReportErrorToDOM() : void
Siftables()
AppRunner
Class
Fields
_app : BaseApp
_cubes : CubeSet
_isRunning : bool
_runner : Thread
_uiDispatcher : Dispatcher
Properties
Cubes : CubeSet
Running : bool
Methods
AppRunner()
PauseExecution() : void
Run() : void
StartExecution() : void
StopExecution() : void
MainWindowView
Class
UserControl
Methods
MainWindowView()
MainWindowViewModel
Class
Fields
_appRunner : AppRunner
_cubes : ObservableCollection<CubeView>
_status : string
ReadyStatus : string
Properties
ARunner : AppRunner
ChangeNumberOfCubesCommand : RelayCommand<EventArgs>
Cubes : ObservableCollection<CubeView>
LoadAFileCommand : RelayCommand
ReloadAFileCommand : RelayCommand
SiftCubeSet : CubeSet
SnapToGridCommand : RelayCommand
Status : string
Methods
MainWindowViewModel()
NotifyPropertyChanged() : void
Events
PropertyChanged : PropertyChangedEventHandler
ClipBehavior
Class
Fields
ToBoundsProperty : DependencyProperty
Methods
ClipToBounds() : void
fe_Loaded() : void
fe_SizeChanged() : void
GetToBounds() : bool
OnToBoundsPropertyChanged() : void
SetToBounds() : void
CubeView
Class
UserControl
Methods
CubeView()
CubeViewModel
Class
Fields
_backgroundColor : Brush
_cube : Cube
_screenItems : ObservableCollection<FrameworkElement>
Properties
BackgroundColor : Brush
CubeModel : Cube
ScreenItems : ObservableCollection<FrameworkElement>
Methods
CubeViewModel()
NotifyPropertyChanged() : void
UpdateBackgroundColor() : void
UpdateScreenItems() : void
Events
PropertyChanged : PropertyChangedEventHandler
DragAndDropBehavior
Class
Behavior<UIElement>
Fields
_isDragging : bool
mouseClickPosition : Point
parent : DependencyObject
Methods
AssociatedObject_MouseLeftButtonDown() : void
AssociatedObject_MouseLeftButtonUp() : void
AssociatedObject_MouseMove() : void
DragAndDropBehavior_MouseEnter() : void
DragAndDropBehavior_MouseLeave() : void
OnAttached() : void
OnDetaching() : void
BaseApp
Class
Fields
_cubes : CubeSet
FrameRate : int
Properties
Cubes : CubeSet
Methods
BaseApp()
Setup() : void
Tick() : void
MyApp
Class
BaseApp
Cube
Class
Fields
_backgroundColor : Color
_screenItems : Collection<FrameworkElement>
dimension : int
SCREEN_HEIGHT : int
SCREEN_MAX_X : int
SCREEN_MAX_Y : int
SCREEN_MIN_X : int
SCREEN_MIN_Y : int
SCREEN_WIDTH : int
Properties
BackgroundColor : Color
ScreenItems : Collection<FrameworkElement>
Methods
Cube()
FillRect() : void
FillScreen() : void
Events
NotifyBackgroundColorChanged : EventHandler
NotifyScreenItemsChanged : EventHandler
Nested Types
CubeSet
Class
Collection<Cube>
Methods
ClearEvents() : void
ClearUserData() : void
CubeByID() : Cube
CubeSet()
toArray() : Cube[]
Neighbors
Class
Fields
_neighbors : Cube[]
_numNeighbors : int
Properties
BOTTOM : Cube
Count : int
IsEmpty : bool
LEFT : Cube
RIGHT : Cube
TOP : Cube
Methods
Contains() : bool
CubeOnSide() : Cube
Neighbors()
SideOf() : Side
sideUtil() : void

3 Applications of the GoF Principles

3.1 Facade

The emulator system is designed in such a way that application developers have just one point of entry: the Sifteo API. The developer can only interact via objects like Cubes, CubeSets, etc. as defined in the Sifteo package, and need not (and do not) have any notion of the specific implementation of the interactions between system-level components and the domain objects exposed by the Sifteo API.

3.2 Observer

The MVVM framework makes use of the interface INotifyPropertyChanged. The View-Models have properties, and the changing of those properties is observed by the view, which responds as defined in the user interface. The use of bindings and ItemsControl objects allows the views to "observe" the ViewModels and their properties or observable collections. This supports loose coupling by abstracting the ViewModel away from the view so the ViewModel logic can be implemented independently.

3.3 Singleton

One instance of an AppRunner will be created, and when an application is loaded by the user it is associated with the AppRunner, along with the current CubeSet in the environment. If there were more than one AppRunner in the emulator workspace, then multiple applications could end up running and causing inconsistencies in the expected execution and the visual output in the workspace. Having AppRunner as a Singleton removes this potential inconsistency.

4 Acceptance Test Plan

The test cases included in this section are integration tests that simultaneously serve as acceptance tests for the user interface. Because the user interface is the focal point of the project, a functional user interface reasonably exemplifies a functional project. In the future, it may become necessary to test our implementations of the Sifteo package by adding further acceptance tests.

4.1 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 proper Siftables application 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.

4.2 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.

4.3 Zoom screen

Scenario #	Originating Flow
1	Basic flow

Test Case ID	Scenario	Description	Expected Result
1	1	User zooms in fully.	The Workspace view has been zoomed to twice its original size.
2	1	User zooms out fully.	The entire Workspace is visible.

4.4 Add/remove Cubes

Scenario #	Originating Flow	
1	Basic flow	

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

4.5 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.

4.6 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.