Siftables Emulator
Deployment and Usage

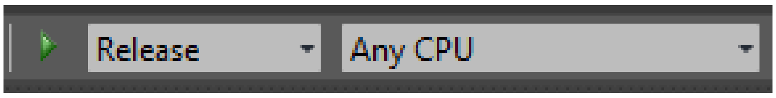
Singularity Software

April 29, 2012

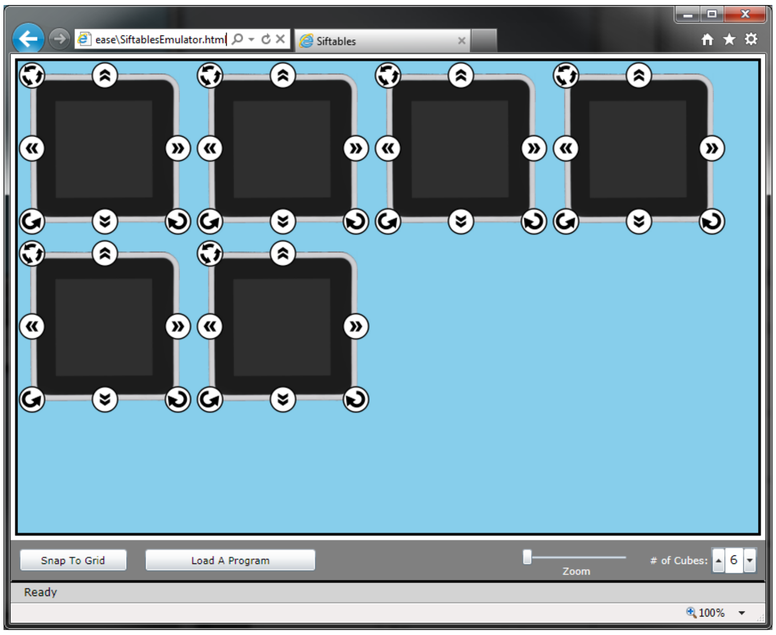
1 Install the emulator

Siftables Emulator is a cross-platform Silverlight application. To install it for the first time:

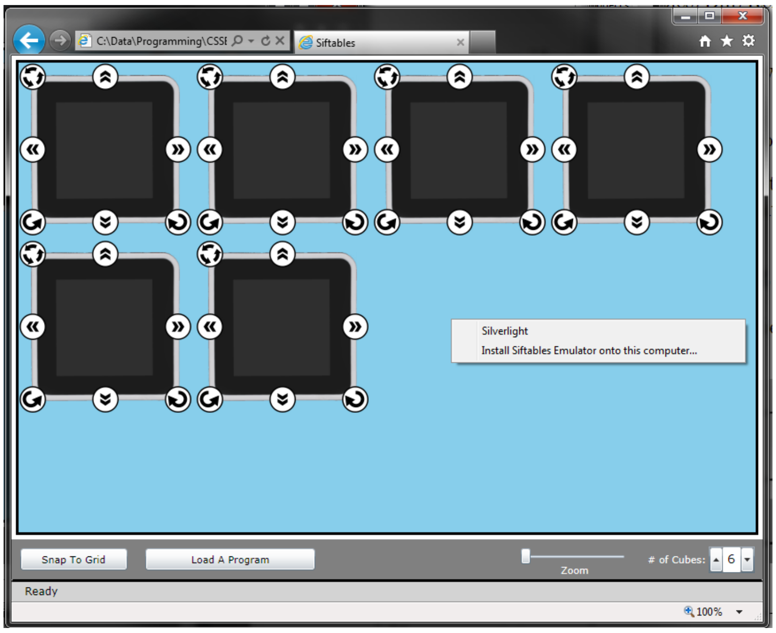
1. Build the project using the Release configuration in Visual Studio.



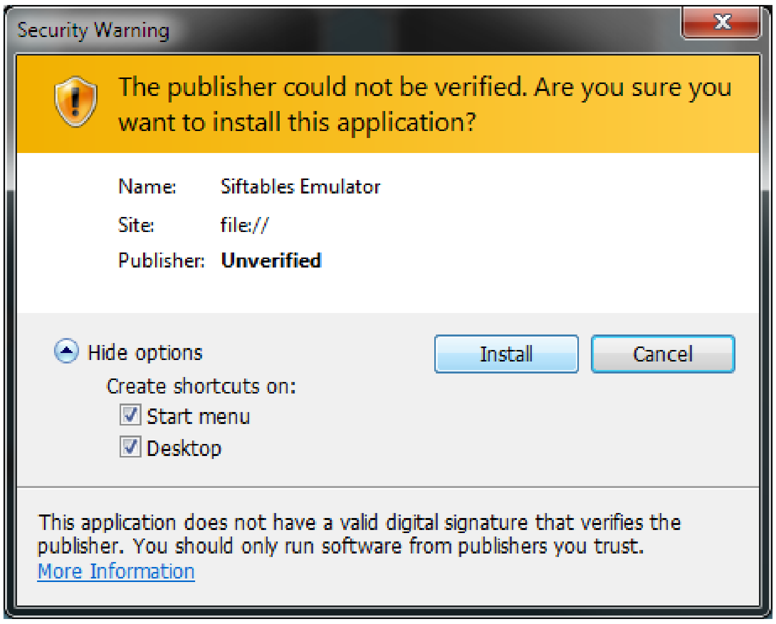
2. Open [project root]/Siftables/Bin/Release/SiftablesEmulator.html in a Silverlight-compatible web browser.



3. When the app loads, right click anywhere and choose "Install Siftables Emulator onto this computer..."



4. Follow the install wizard, choosing your preferred shortcut locations.



Security Warning

The publisher could not be verified. Are you sure you want to install this application?

Name: Siftables Emulator

Site: file://

Publisher: **Unverified**

Hide options

Create shortcuts on:

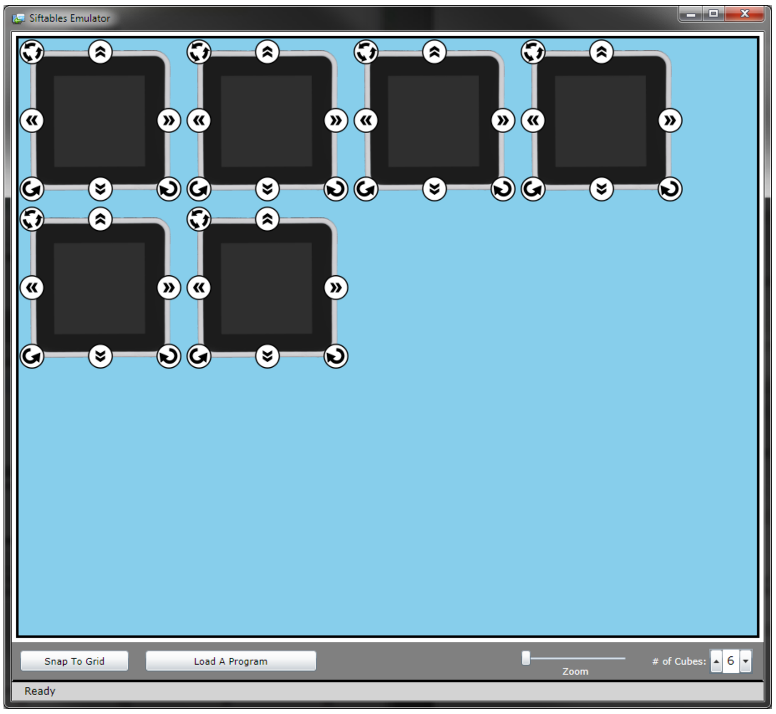
☑ Start menu

☑ Desktop

Install | Cancel

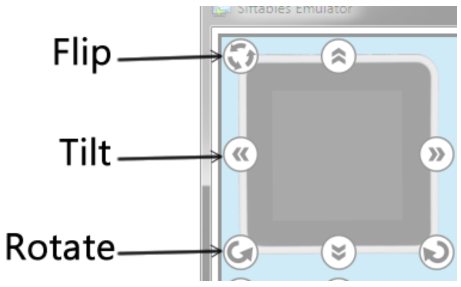
This application does not have a valid digital signature that verifies the publisher. You should only run software from publishers you trust.
More Information

5. Siftables Emulator should launch automatically. If not, it can be launched from wherever you opted to install shortcuts in the previous step.



2 Walkthrough: Interact with the emulator

Once you've launched the emulator, play around! If you get stuck, consult the following mapping of actions on the physical Sifteo Cubes to their digital equivalents.



If you want to...	
Flip the cube,	click the flip button.
Tilt the cube,	click the tilt button for the direction to tilt [left, up, right, down].
Rotate the cube,	click the rotate button for the direction to rotate [counterclockwise, clockwise].
Shake the cube,	drag the cube horizontally back and forth rapidly.
Press the cube,	click the virtual screen.

If you click...	
Snap to Grid,	the emulator rearranges the cubes into a 4-cube wide grid based on order of cube creation.
Load a Program,	the emulator opens the Load dialog for running a Siftables application DLL.
Zoom,	the emulator zooms the canvas in (right) or out (left) to a maximum of 2x zoom.
# of Cubes,	the emulator changes the number of cubes available on the emulator "screen" to a [maximum of 9, minimum of 1] cubes.

3 Program for the emulator

3.1 Application Programming Interface

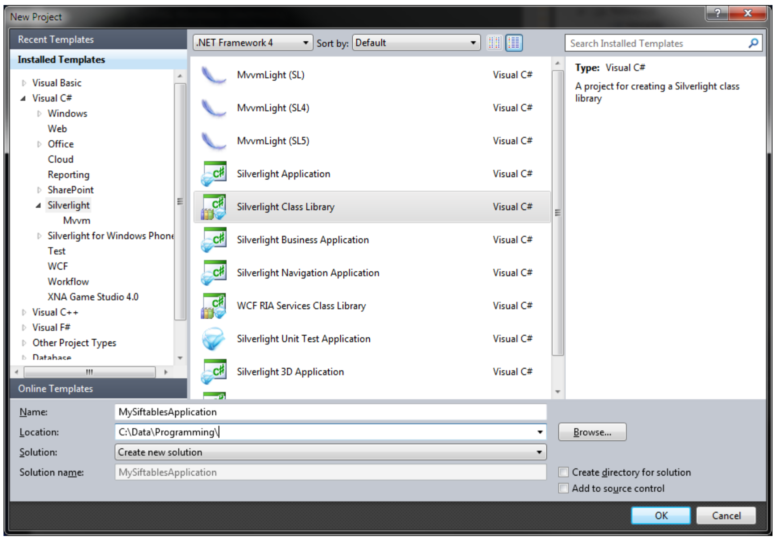
Siftables Emulator can be programmed using the official Sifteo API available at http://developer.sifteo.com/. The team believes that our implementation of the API outlined there is complete and is functionally on par with the native Sifteo.dll provided for use with the physical Sifteo cubes. This implementation is a combination of work done by the team specifically for Silverlight and the Siftables project and work done by the Sifteo team. The latter part comes in the form of SifteoExtensions.dll, a partial version of Sifteo.dll decompiled and retargeted to Silverlight with Sifteo's permission.

3.2 Target the emulator with an existing application

To target an existing application targeted for Sifteo Cubes to run in Siftables Emulator, follow the instructions in step 1 of the following walkthrough. When you've created the new project, add your existing application files to it.

3.3 Walkthrough: Prepare and run a new application

1. Create a new project for the application.
   (a) Select the Silverlight template group in the left pane, then select Silverlight Application in the center.
   (b) Give the application a name and a location. We recommend not creating a directory for the solution.



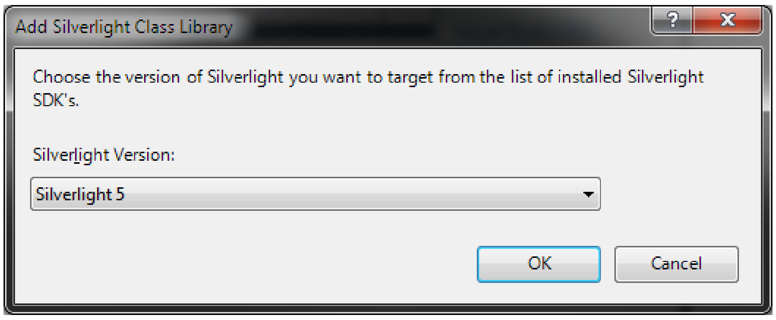
MvvmLight (SL)	Visual C#
MvvmLight (SL4)	Visual C#
MvvmLight (SL5)	Visual C#
Silverlight Application	Visual C#
Silverlight Class Library	Visual C#
Silverlight Business Application	Visual C#
Silverlight Navigation Application	Visual C#
WCF RIA Services Class Library	Visual C#
Silverlight Unit Test Application	Visual C#
Silverlight 3D Application	Visual C#

.NET Framework 4 Sort by: Default

Type: Visual C#
A project for creating a Silverlight class library

Name: MySiftablesApplication
Location: C:\Data\Programming\
Solution: Create new solution
Solution name: MySiftablesApplication
Create directory for solution
Add to source control

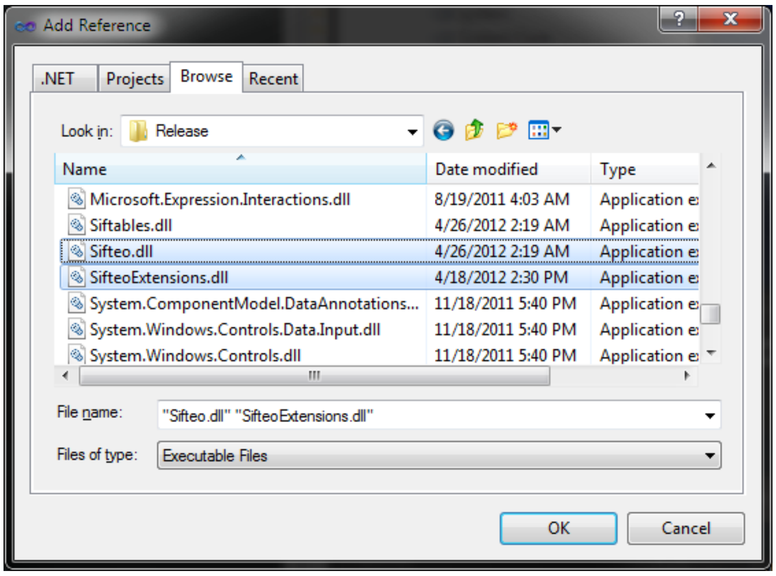
(c) Ensure that you use Silverlight version 5.



Choose the version of Silverlight you want to target from the list of installed Silverlight SDK's.

Silverlight Version: Silverlight 5

(d) Add Sifteo.dll and SifteoExtensions.dll to the project references. Note that those DLLs will only exist if you have already built the Release configuration of the emulator as specified in the installation instructions.



Look in: Release

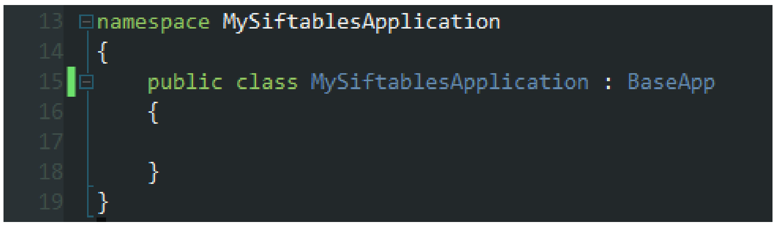
Name	Date modified	Type
Microsoft.Expression.Interactions.dll	8/19/2011 4:03 AM	Application e
Siftables.dll	4/26/2012 2:19 AM	Application e
Sifteo.dll	4/26/2012 2:19 AM	Application e
SifteoExtensions.dll	4/18/2012 2:30 PM	Application e
System.ComponentModel.DataAnnotations...	11/18/2011 5:40 PM	Application e
System.Windows.Controls.Data.Input.dll	11/18/2011 5:40 PM	Application e
System.Windows.Controls.dll	11/18/2011 5:40 PM	Application e

File name: "Sifteo.dll" "SifteoExtensions.dll"
Files of type: Executable Files

2. Build a blank runnable application.

(a) Create a new class.

(b) Have it extend BaseApp.

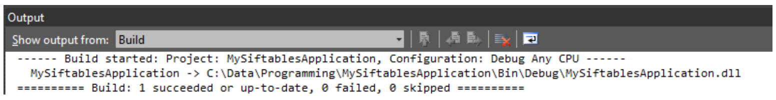


```
namespace MySiftablesApplication
{
    public class MySiftablesApplication : BaseApp
    {

    }
}
```

(c) Have it use the Sifteo namespace (Visual Studio should offer to do this for you).

(d) Build the solution. Note the location of the DLL in Output.



```
------ Build started: Project: MySiftablesApplication, Configuration: Debug Any CPU ------
  MySiftablesApplication -> C:\Data\Programming\MySiftablesApplication\Bin\Debug\MySiftablesApplication.dll
========== Build: 1 succeeded or up-to-date, 0 failed, 0 skipped ==========
```

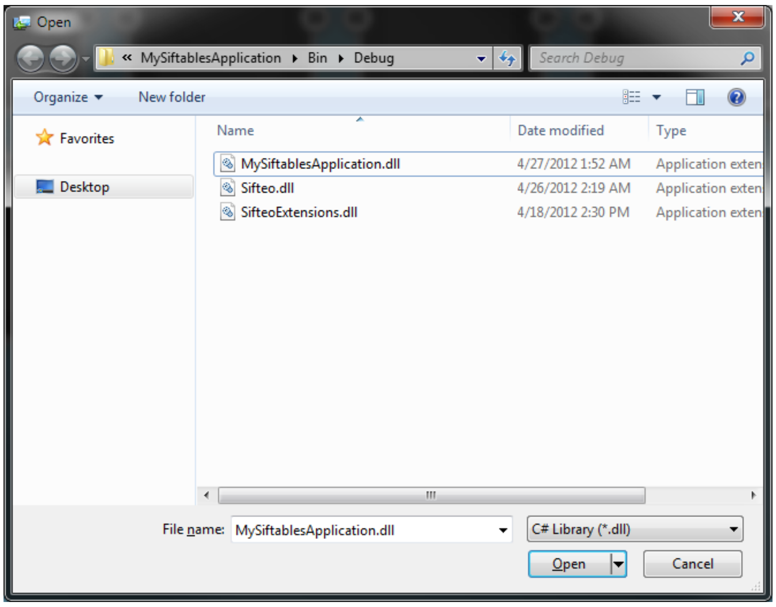
3. Run the blank application in the emulator.

   (a) Launch the emulator.

   (b) Click "Load A Program".

   (c) Select the DLL built in the previous step.

   (d) Click Open.



Name	Date modified	Type
MySiftablesApplication.dll	4/27/2012 1:52 AM	Application exten
Sifteo.dll	4/26/2012 2:19 AM	Application exten
SifteoExtensions.dll	4/18/2012 2:30 PM	Application exten

File name: MySiftablesApplication.dll

C# Library (*.dll)

At this point, you have a fully runnable Siftables application. It doesn't do anything... but that part is up to you!

3.4 Walkthrough: Create a basic application

At the completion of this walkthrough, you will have an application that responds to all cube events (see the next section for a complete list). We suggest starting with the MySiftablesApplication you created in the previous step.

1. Set up the Setup and Paint methods. Setup runs once when your application is first loaded. Tick is called periodically according to the FrameRate (we'll use the default framerate for now).

```
public override void Setup()
{
    foreach (Cube cube in CubeSet)
    {
        cube.FillScreen(Color.White);
        cube.Paint();
    }
}
public override void Tick()
{

}
```

2. Add methods to describe what happens when each event occurs.

   (a) Add a method to paint the screen red when the cube is pressed/clicked.

   ```
   private void OnPress(Cube cube)
   {
       cube.FillScreen(new Color(255, 0, 0));
       cube.Paint();
   }
   ```

   (b) Add a method to paint the screen various shades of green depending on which tilt direction is activated.

   ```
   private void OnTilt(Cube cube, Cube.Side direction)
   {
       switch (direction)
       {
           case Cube.Side.TOP:
               cube.FillRect(new Color(0, 200, 0), 54, 54, 20, 20);
               break;

           case Cube.Side.BOTTOM:
               cube.FillRect(new Color(0, 150, 0), 54, 54, 20, 20);
               break;

           case Cube.Side.LEFT:
               cube.FillRect(new Color(0, 100, 0), 54, 54, 20, 20);
               break;

           case Cube.Side.RIGHT:
               cube.FillRect(new Color(0, 50, 0), 54, 54, 20, 20);
               break;
           default:
               cube.FillRect(Color.White, 54, 54, 20, 20);
               break;
       }
   ```

```
        cube.Paint();
    }
```

(c) Add a method to paint the screen blue when the cube is flipped.

```
    private void OnFlip(Cube cube, bool newOrientationIsUp)
    {
        cube.FillRect(new Color(0, 0, 255), 54, 54, 20, 20);
        cube.Paint();
    }
```

(d) Add a method to paint the screen a random color when the cube begins to shake.

```
    private void OnShake(Cube cube)
    {
        Random rand = new Random();
        cube.FillRect(new Color(rand.Next(0, 256), rand.Next(0, 256),
                      rand.Next(0, 256)), 54, 54, 20, 20);
        cube.Paint();
    }
```

(e) Add a method to paint the screen white when the cube stops shaking.

```
    private void OffShake(Cube cube, int duration)
    {
        cube.FillScreen(Color.White);
        cube.Paint();
    }
```

3. Modify the Setup method to add the new methods to the appropriate events on each cube.

```
public override void Setup()
{
    foreach (Cube cube in CubeSet)
    {
        cube.FillScreen(Color.White);
        cube.Paint();

        cube.NotifyButtonPressed += OnPress;
        cube.NotifyCubeTilt += OnTilt;
        cube.NotifyCubeFlip += OnFlip;
        cube.NotifyShakeStarted += OnShake;
        cube.NotifyShakeStopped += OffShake;
    }
}
```

4. Modify the Tick method to paint the screen a different colors based on the amount of neighbors the cube has.

```
    public override void Tick()
    {
        foreach (var c in this.CubeSet)
        {
            var num = c.Neighbors.Count;
            switch (num)
            {
                case 0:
                    c.FillScreen(Color.Black);
                    break;
                case 1:
                    c.FillScreen(new Color(255, 255, 0));
                    break;
                case 2:
                    c.FillScreen(new Color(255, 255, 50));
                    break;
                case 3:
                    c.FillScreen(new Color(255, 255, 100));
                    break;
                case 4:
                    c.FillScreen(new Color(255, 255, 150));
                    break;
            }
            c.Paint();
        }
    }
```

Build your app, then load MySiftablesApplication/bin/Debug/MySiftablesApplication.dll in the emulator.

3.5 Respond to cube events

All cube interactions appear to your program as events. If you want to do something when one of those interactions occurs, simply add a handler to the appropriate event.

If you want to respond when the user...	
Flips the cube,	bind to cube.NotifyCubeFlip.
Tilts the cube,	bind to cube.NotifyCubeTilt.
Rotates the cube,	you can't - at least, you can't directly. The orientation of the cube only affects operations that use neighboring to make multiple cubes interact.
Shakes the cube,	bind to cube.NotifyShakeStarted and/or cube.NotifyShakeStopped.
Presses the cube,	bind to cube.NotifyButtonPressed.
Has neighbored cubes,	access cube.Neighbors.

3.6 Example applications

Example applications are located in [project root]/Applications.

- CubeTestApp has examples of how to respond to most cube interactions and is a good place to start.
- ChangingColorsApp is an example of how to work with neighbored cubes.
- FractionOrderingApp uses neighboring and images in a simple ordering game.