Illumination Software Creator : The Book

Build apps for iOS, Android, Web and more... without writing a drop of code.

by Bryan Lunduke

Book Version - 1.0
Covering Illumination Software Creator Version - 4.4

Table of Contents

Chapter 1

Welcome To Illumination

Have you ever watched a movie where a bright, young “Computer Hacker” would sit down at a computer and, with just 60 seconds left before something explodes, quickly creates a piece of software that saves the day? And, so often, he (or she) does it simply by moving around visual “blocks” on the screen.

Of course, this is just fiction. Making software is hard. Computer Programmers go to school for years, wear thick glasses and sit in dimly lit rooms, well into the night, writing line after line of computer code.

But what if it didn't have to be that way?

That's where Illumination Software Creator comes in. The idea is simple: Allow anyone, even people with no computer programming experience, to create their own software applications in a fun, visual way. Just like what you see in those movies. A worth-while goal, to be sure. (And one of the core design concepts behind Illumination.) But we wanted to take it even further.

We set out to fix one other big problem that so many programmers run in to:

Building apps and games for multiple platforms and devices.

And, boy howdy, did we nail it.

With Illumination Software Creator you can visually build your app (in that awesome Sci-Fi movie, 100% visual way) for an incredibly wide array of platforms. As of this writing the following are supported:

1. iOS – iPhone and iPad
2. Android
3. HTML5 Interactive Websites
4. Flash Interactive Websites
5. Python Desktop Applications
6. Python Mobile Applications

And, what's even better, the resulting Android and iOS apps are 100% native and can be submitted to the iTunes App Store and Android's Google Play market.

So not only is this the absolute easiest way to build software (anyone can do it, even without programming experience)... and the fastest way to build software (nothing beats the speed of laying out colorful blocks!)... but it also also lets you target more modern platforms than any programming language on the planet.

Let's get started.

Chapter 2

Installing Illumination Software Creator

Windows Users:

1. Download Illumination for Windows from http://www.lunduke.com and run "iscsetup.exe".
2. Illumination supports Windows XP, Windows Vista, Windows 7 and Windows 8.

Linux Users:

1. Download the package for your particular flavor of Linux from http://www.lunduke.com.
2. Alternately there is a distro-generic, 32bit version available as a tar.gz archive.

MacOS X Users:

1. Download Illumination for MacOS X from the “Download” section.
2. Illumination requires MacOS X 10.6 or newer.

Illumination Software Creator itself has very few requirements. The only real requirements you need to be aware of are for the applications you build yourself:

Python/Desktop Applications require that the “Python/GTK” software be installed on your computer in order to run properly. See the chapter on building Python applications for more details.

Adobe Flex/Flash Web Applications require that you have the Adobe Flex SDK installed in order to build your Flash website.

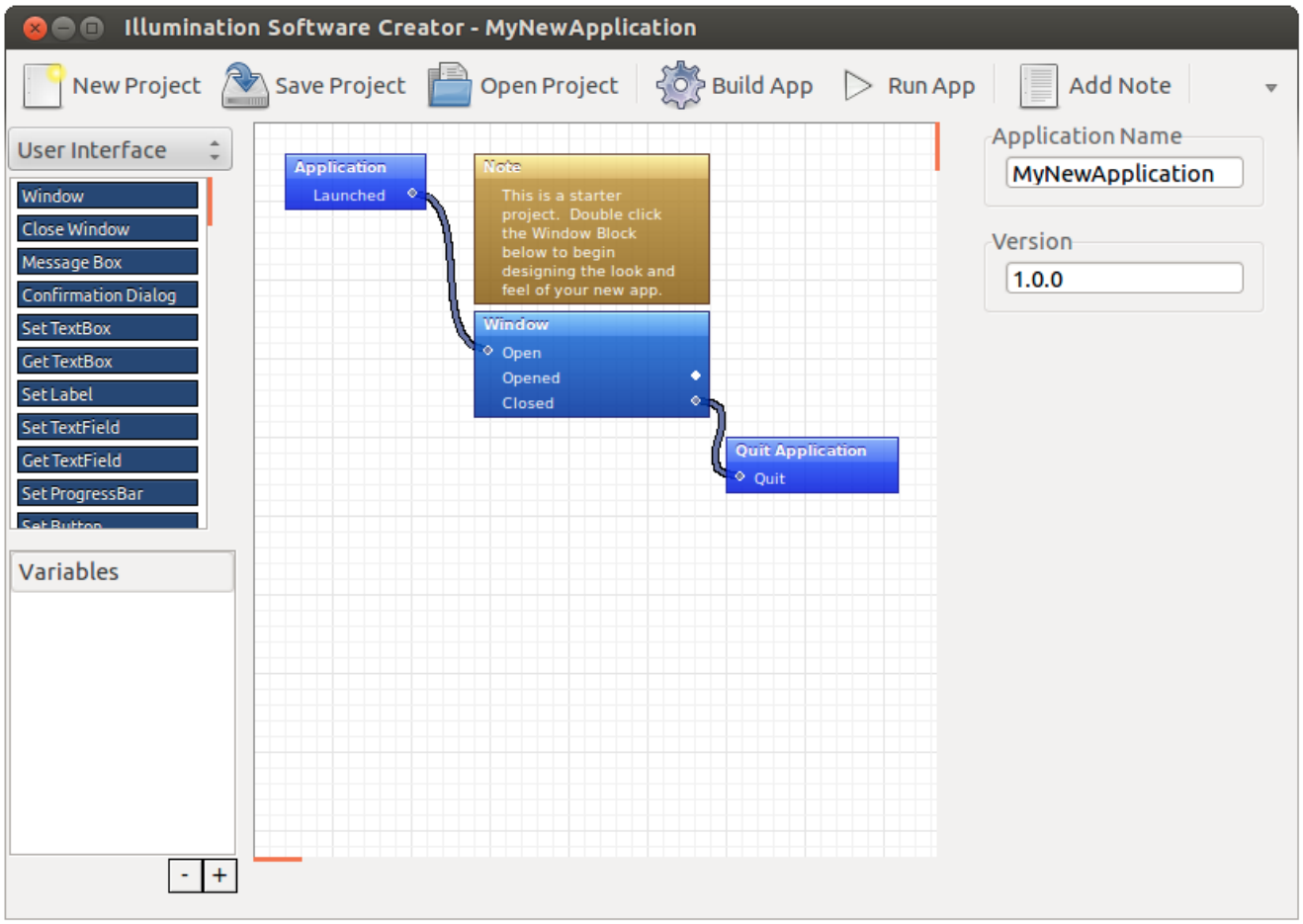
Building iPhone and iPad Applications require a Macintosh with a current version of Xcode installed. And Android Applications require the Android SDK.

Chapter 3

Hello World!

When you first launch Illumination you are greeted with an empty project with two "Blocks" in it. A "Block" within Illumination represents something you would like your software to do.

For example, the two starting blocks are "Application" and "Quit Application". These represent your application "Launching" and "Quitting" (two things every piece of software does).



You'll note that within each block are inputs and outputs (designated by small white circles). An input has a white circle on the left hand side, an output has a circle on the right hand side. In on the left, out on the right. If you click your mouse on an output circle (right hand side) you can drag a "link" to any input (left hand side) on any other block. In this way you can tell your software which order to perform each block in. (If you create a link that you want to remove, simply right click on the block the link originates from and choose “Remove Link”.)

Every computer programmer creates a "Hello World" program when they are first learning how to create software. And Illumination is no different! (Okay, maybe a little different... Illumination is much easier...)

On the left hand side of Illumination you'll notice a colored list. These are blocks that you can use however you like by simply dragging them with your mouse onto the project grid (the main area in the center of Illumination).

For this example, go ahead and drag the item that reads "Message Box" to the project area. Then drag a link from the "Launched" output of your Application block to the "Open" input of your Message Box block.

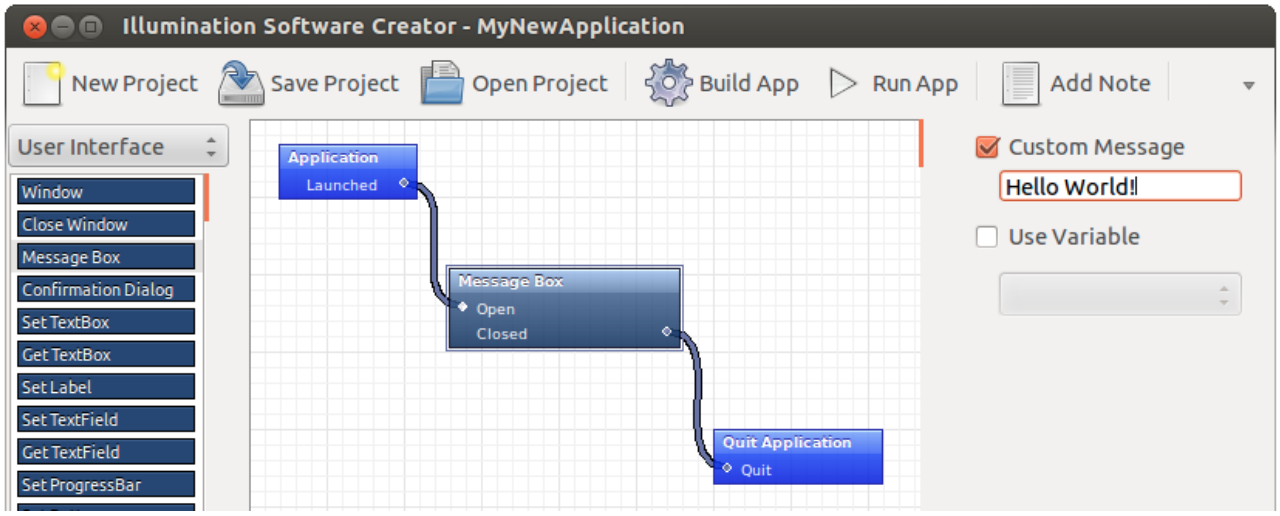
This tells your application "When you launch, open the Message Box".

From there drag a link from the "Closed" output of your Message Box to the "Quit" input of your "Quit Application" block.

This tells your application "When the Message Box is closed, quit the application".

Nice and easy, right? You can think of it almost like water flowing from one container to the next, connected together by pipes (in this case, the “links”). Now, what message would you like your Message Box to display?

Simply click on the Message Box block, check "Custom Message" from the area on the right, and enter any message you would like to display. (A standard Hello World! message is always great to start with!)



You now have a fully functioning application! Save your project somewhere (anywhere you like will be fine) and click Run App (or select “Run Project” or “Build Project...” from the “Project” menu) and choose the HTML5 option to see your app run within your default web browser.

Chapter 4

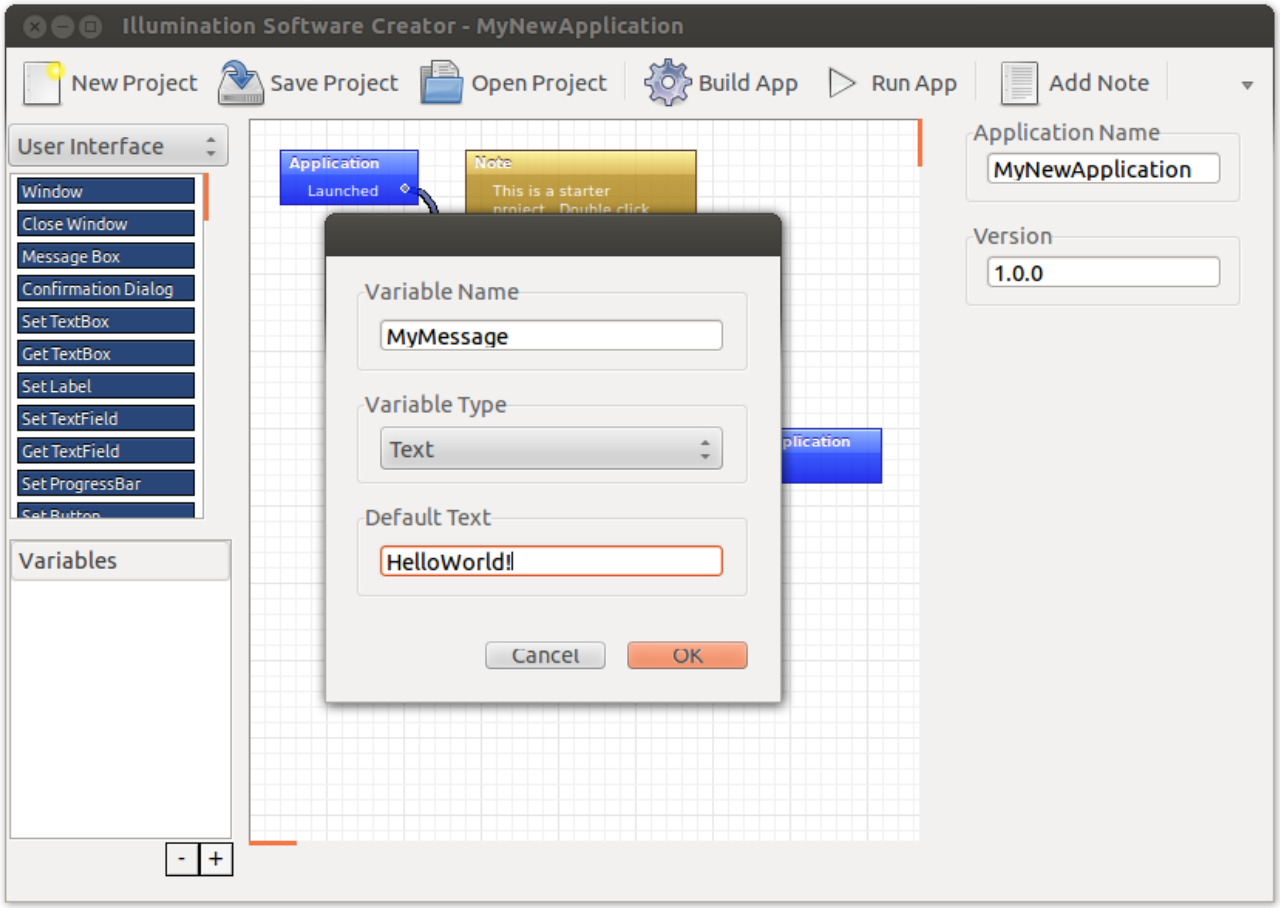
Working With Variables

A "Variable" is a simple idea. It is nothing more than "A thing that holds some information". Think of variable's like the fields in your address book. There are fields for “Name”, “Phone Number”, “Address”, etc.. Each field holds a different type of information. These are variable's.

In Illumination there are three key types of variables:
1. Text – Stores a line of text. A name. A sentence. That sort of thing.
2. Number – Stores a number.
3. Dictionaries – A collection of Text variables that can be used via keys. In typical programmer speak these are “Key/Value Pairs”. This makes up a simple database of information.

Easy, right?

Within Illumination, all variables are listed in the bottom left hand corner (aka "The Variable List"). To add a new variable simply click on the "+" below the variable list.

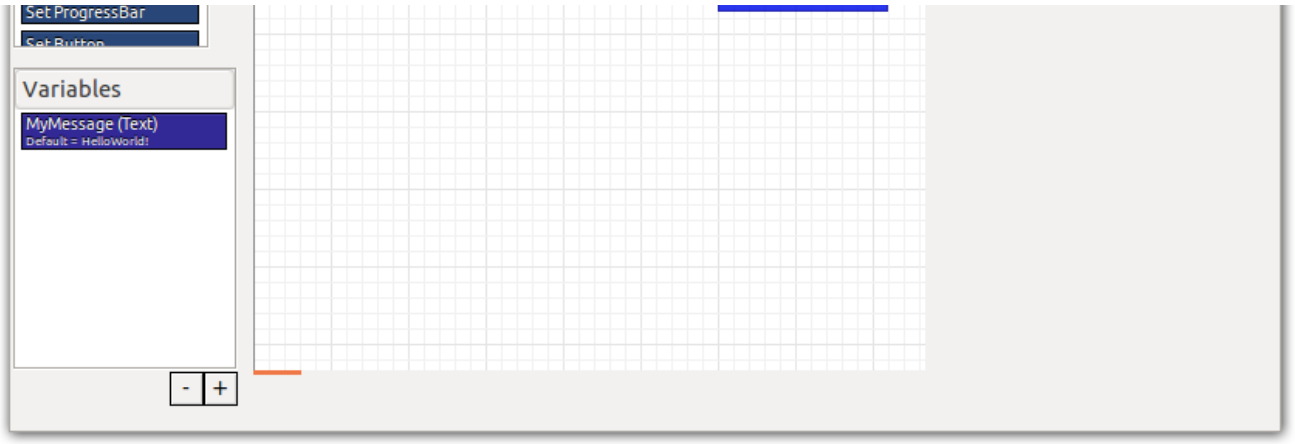


From there you can enter the name you wish to give that variable (which is purely for your own use... you can name your variables anything that will help you remember what they are for). In this example I named mine "MyMessage", but you can call it anything you like.

Then you choose what type of variable it is: In this case, choose Text.

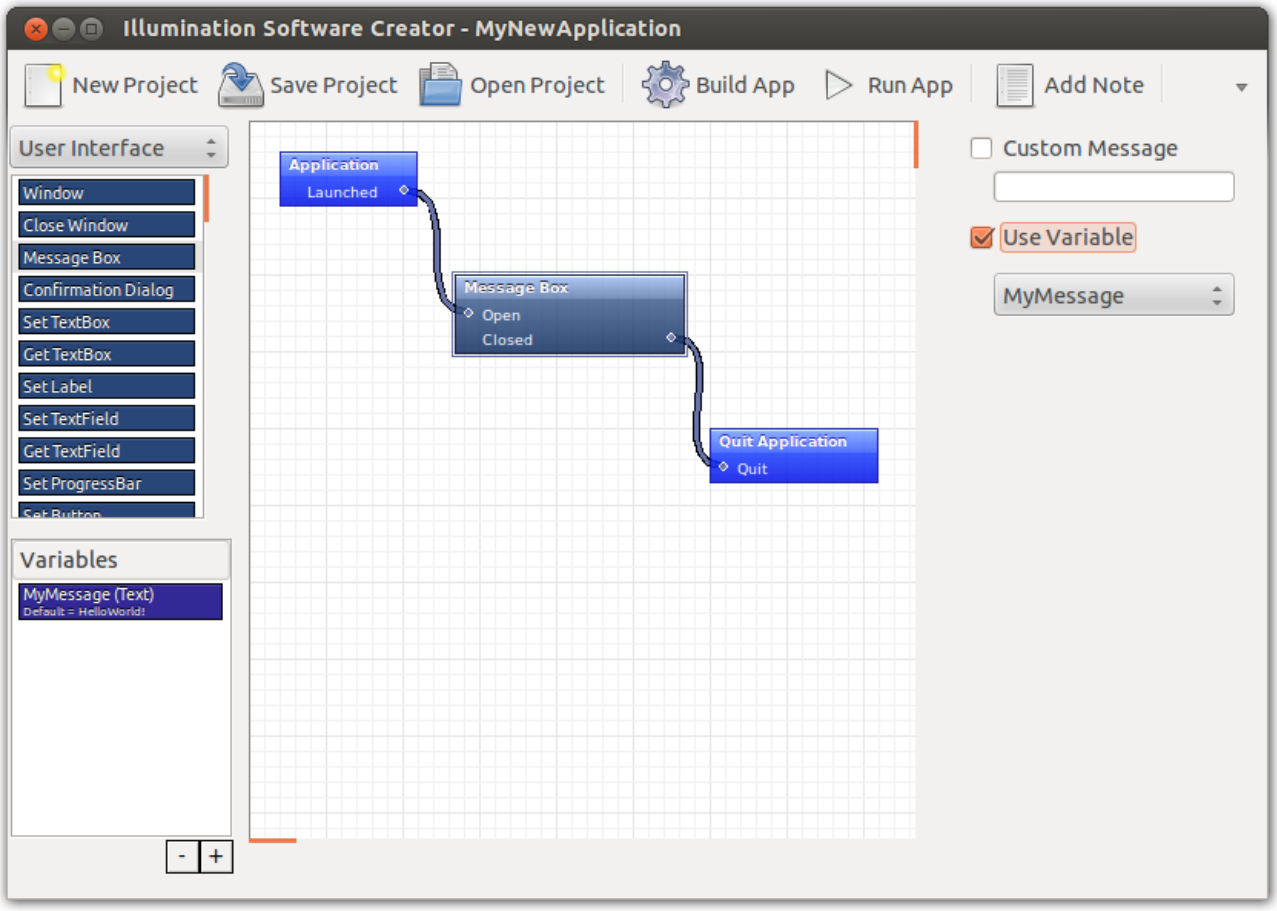
The final step in adding a new variable is to set its default value and click OK. In this case, I went with “Hello World!”.

Once you've done this your new variable will show up in the Variable List.



Now you can re-select your Message Box block in your project (the one you created in Chapter 3) and check "Use Variable" from the area on the right (to tell the Message Box to display the text contained in a variable) and select your newly created variable from the drop down list right below the "Use Variable" checkbox.

Save your project and run it again. This time it will display the message from your Text variable! And if, at any time, you would like to change anything about your variable, all you need to do is double click on the variable in the Variable List.



With this basic understanding you have already created your own simple software!

Now you can feel free to look through the many different types of blocks available (click on the blocktypes drop down list in the upper left to choose from "User Interface", "Text", "Numbers", etc. to see different types of blocks you can use to build your own applications).

Chapter 5

Building a User Interface

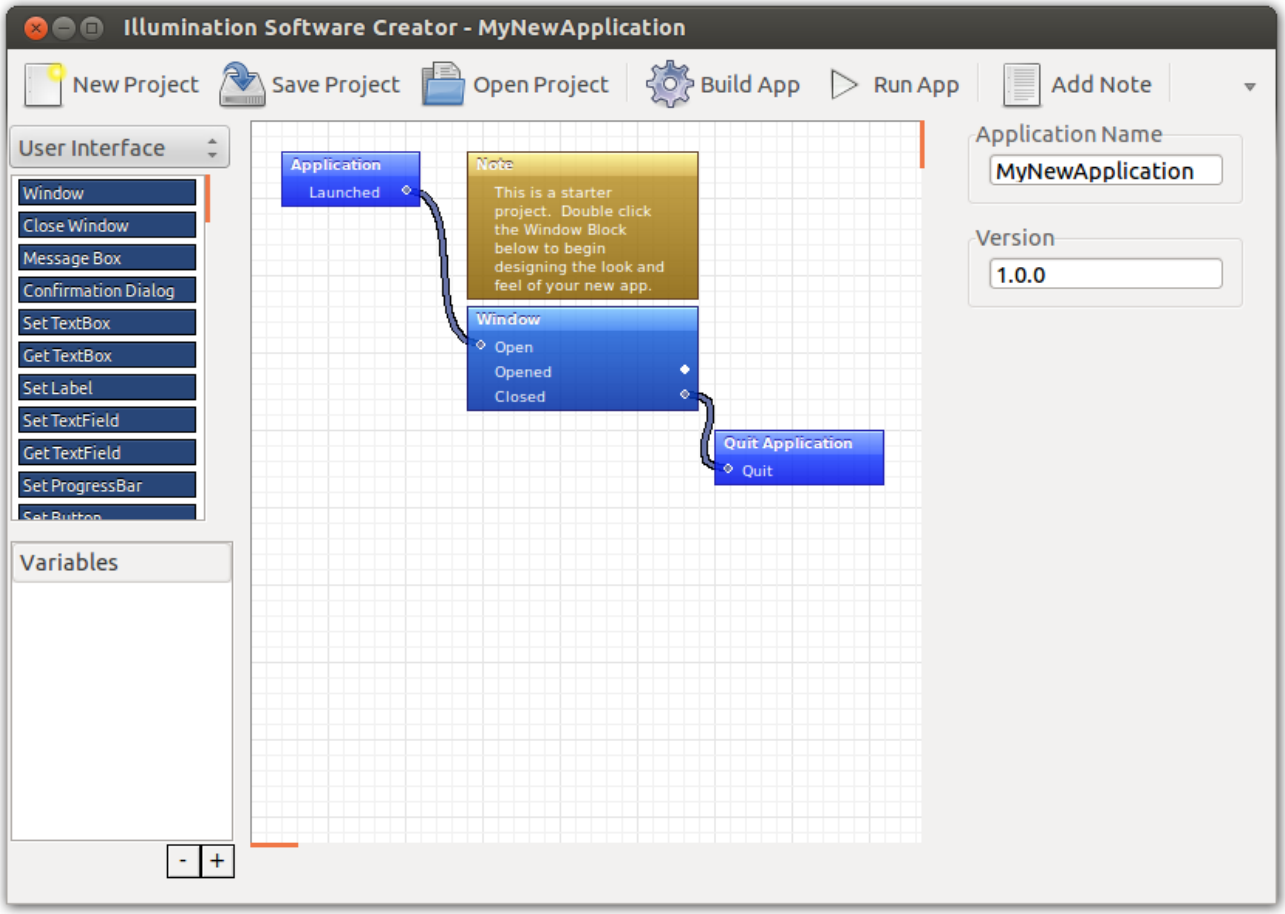
Almost every application, be it for desktop computers, mobile phones or websites, needs a good user interface. Buttons to click. Boxes to type text in to. Etc..

Luckily Illumination Software Creator makes creating these User Interfaces easy as pie.

Step 1) Adding A Window

Start a new project in Illumination by clicking the New Project button on the top left. Make sure "User Interface" is selected from the list of available blocks in the top left corner. Now drag the "Window" block to anywhere on the project grid. (A default new project will already contain one blank Window block, so this step is really only necessary when you want multiple windows in your application.)

A new Window block will be added to your project. You can have as many Window blocks in your project as you wish.

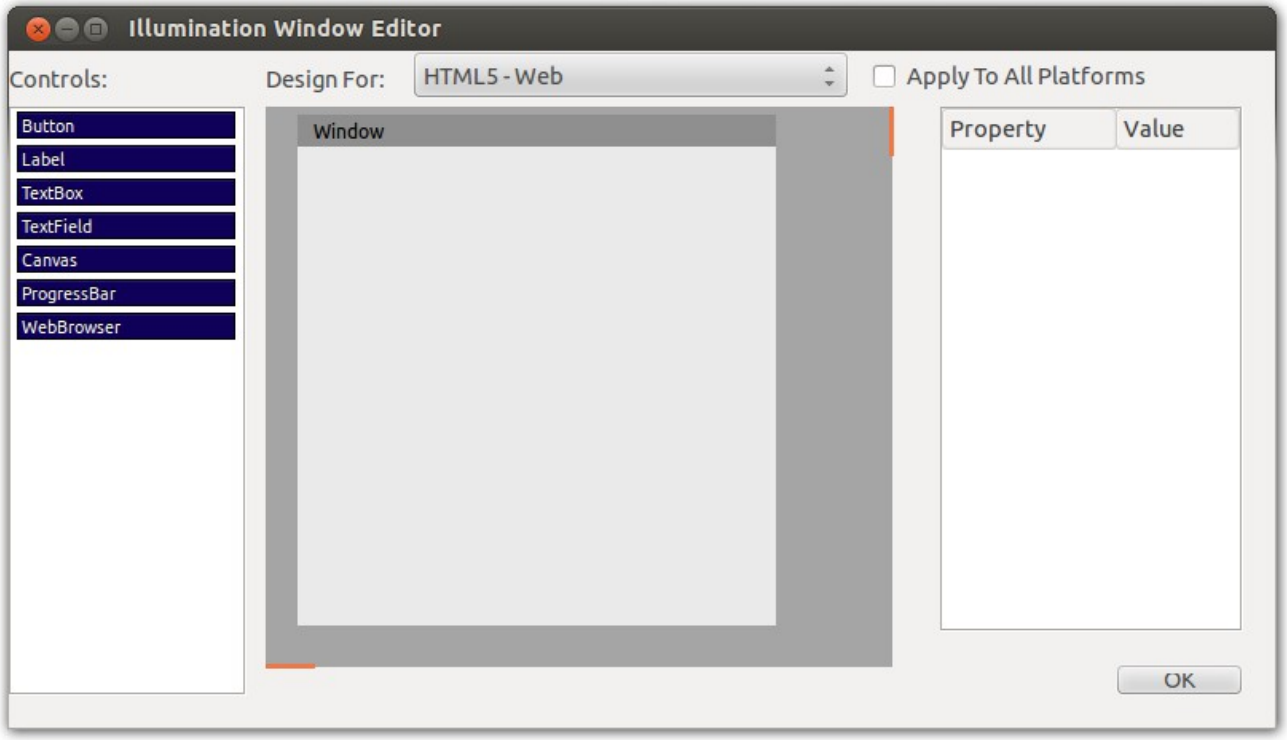


By default you'll notice one input (“Open”) and two outputs (“Opened” and “Closed”) on your new Window block. We'll talk more about these in step three.

Step 2) Designing Your Window

Now double click on your Window block (or select the Window block and click the “Design Window” button on right hand side) to open the Window Editor.

The Illumination Window Editor is divided into three main parts:
1. The list of available controls on the left hand side (buttons, labels, etc.) that can be used in your new Window.
2. The visual design of the window itself in the center.
3. And on the right, a list of properties that you can modify for any selected item in your Window (width of a button, default text to show in a label, etc.).

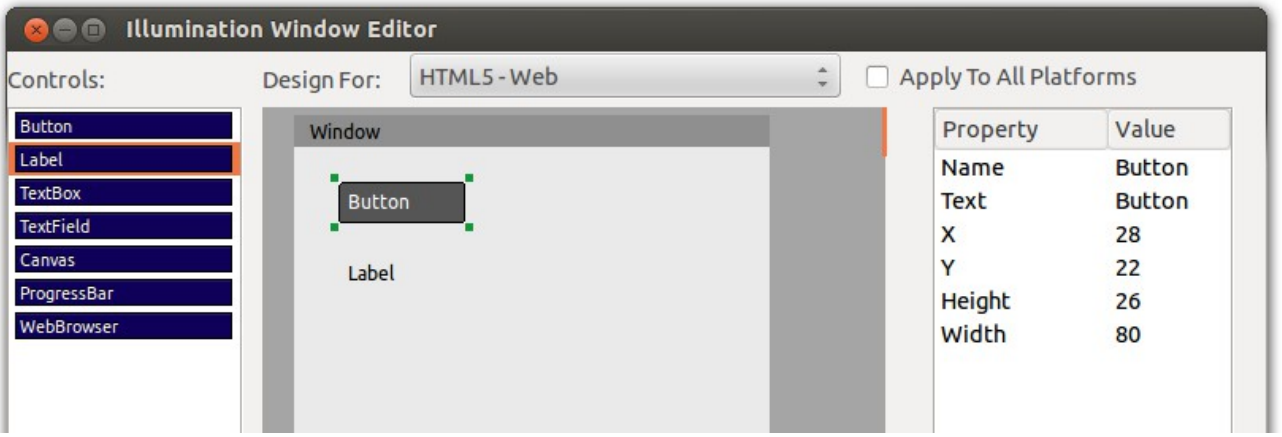


For now, let's create a simple window, with one button and one label.

To do so drag a “Button”, from the list on the left, to your window. You may place the button anywhere you like.

Next, drag a “Label”, from the list on the left, to your window.

You'll notice that, whenever you select a control on your window (or the window itself) by clicking on it, green squares will appear in each corner of that control. These allow you to resize the control by clicking and dragging on them.



You may also modify the size and location of your selected control by manually entering that information into the property list on the right hand side. Once you are happy with your window design, click the OK button to return to your project. (You may return to the Window Editor to modify your window design at any time.)

Step 3) Using Your Window

You'll now notice that your Window block has a new "ButtonClicked" output. This allows you to trigger another block when that particular button is clicked. To see this in action, go ahead and add a new Text variable by clicking on the "+" button on the bottom left below the Variables list.

Enter anything you like for the Variable Name, such as "MyText" and enter "Hello World" for the Default Text, then click OK.

Now drag a "Set Label" block from the list on the left, to your project.

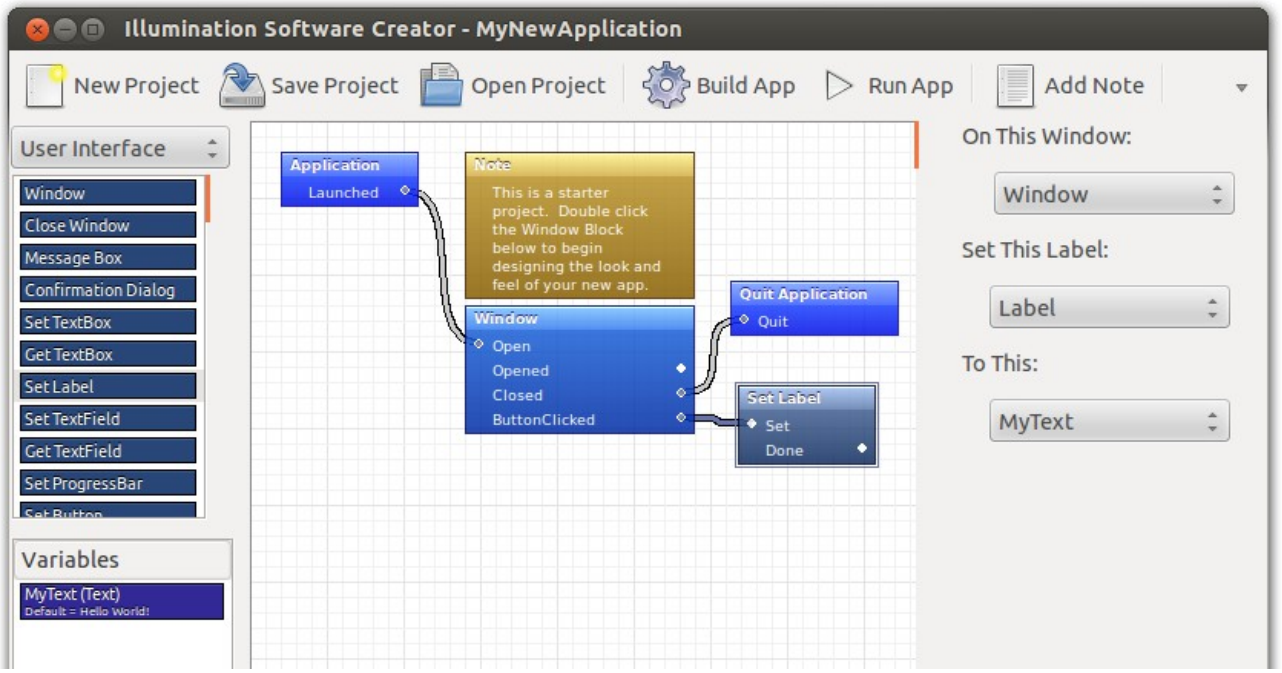
Click on your new Set Label block and set the "On This Window" property to your Window, the "Set This Label" property to the label control you created on that Window, and set the "To This" property to the Text Variable you just created.

What this is doing is, quite simply, setting the label on that window to the text that is stored in that Text Variable.

Next you need to connect up your blocks.

First drag your mouse from the Launched output (the circle on the right) of the Application block to the Open input (the circle on the left) of the Window block. This tells the Window to open when the application is launched.

Then connect the ButtonClicked output of the Window block to the Set input of the Set Label block. Now, when you run this application, clicking the button will set the label on the window to "Hello World".



And you didn't have to write a single line of "code" to make it happen.

Things To Know

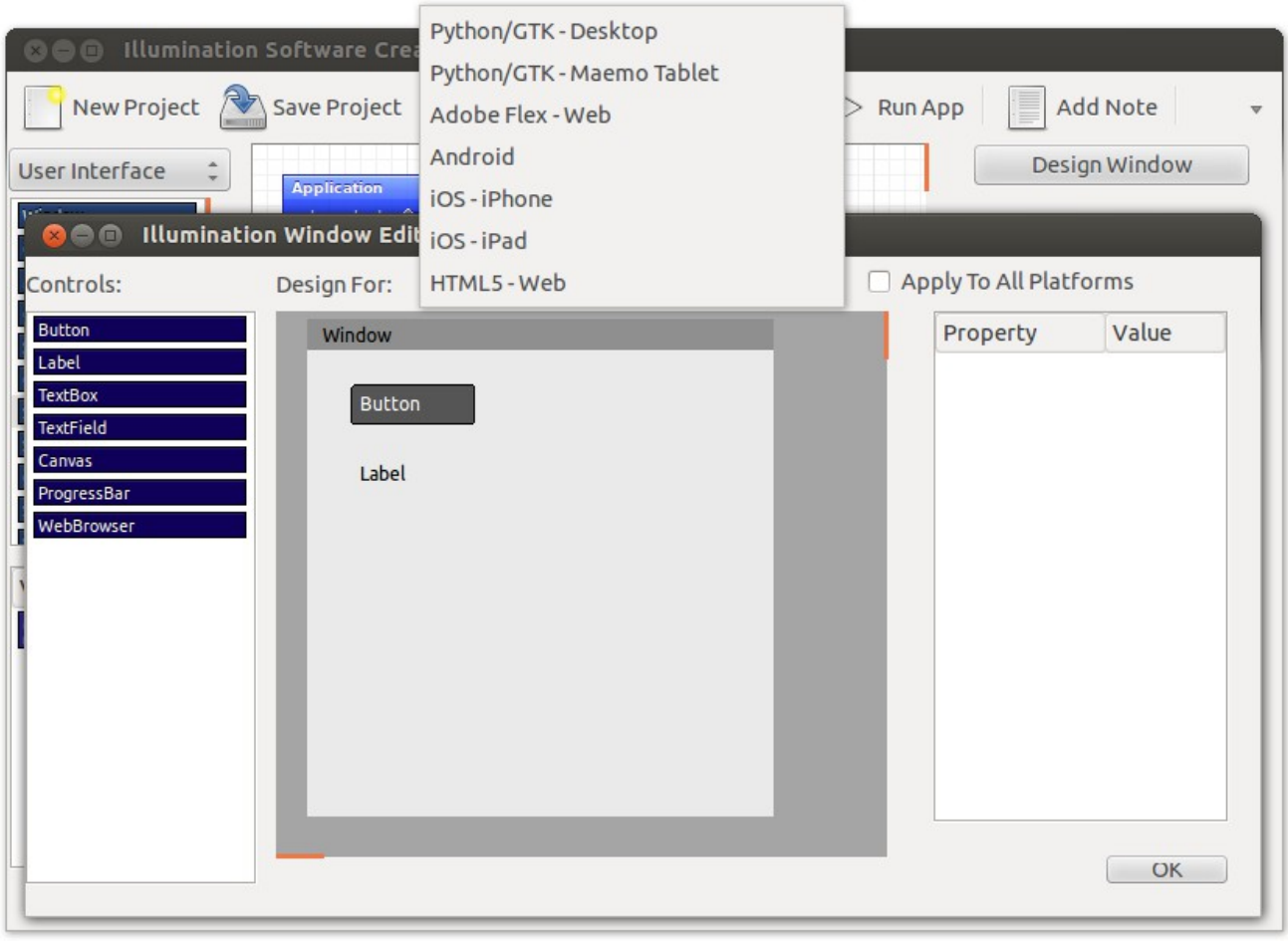
Illumination Software Creator allows you to build applications for a number of different platforms (iPhones, Android phones, Desktop applications, Flash web applications, etc.). And each of those platforms may have different needs for the user interface.

For example: If you design your application to work great on a Windows, Mac or Linux PC... it may not look right on an iPhone (often times the controls may not even be able to fit on the iPhone screen).

Because of this Illumination offers the ability to lay out the same Window differently for each target platform.

On the top of the Window Editor there is a "Design For" drop down list. Clicking that allows you to select which target platform you would like to work on the user interface layout for.

Your Window will still have the same controls on each platform, and no change to your project logic is necessary. The only difference is in how the controls are sized and arranged for each target platform.


Illumination Software Crea
New Project
Save Project
Run App
Add Note
User Interface
Application
Design Window
Python/GTK - Desktop
Python/GTK - Maemo Tablet
Adobe Flex - Web
Android
iOS - iPhone
iOS - iPad
HTML5 - Web
Illumination Window Edit
Controls:
Design For:
Apply To All Platforms
Button
Label
TextBox
TextField
Canvas
ProgressBar
WebBrowser
Window
Button
Label
Property
Value
OK

Chapter 6

Using The Image Library

So you want to add graphics to your application, you say?

Luckily, Illumination Software Creator makes that incredibly easy thanks to the Image Library.

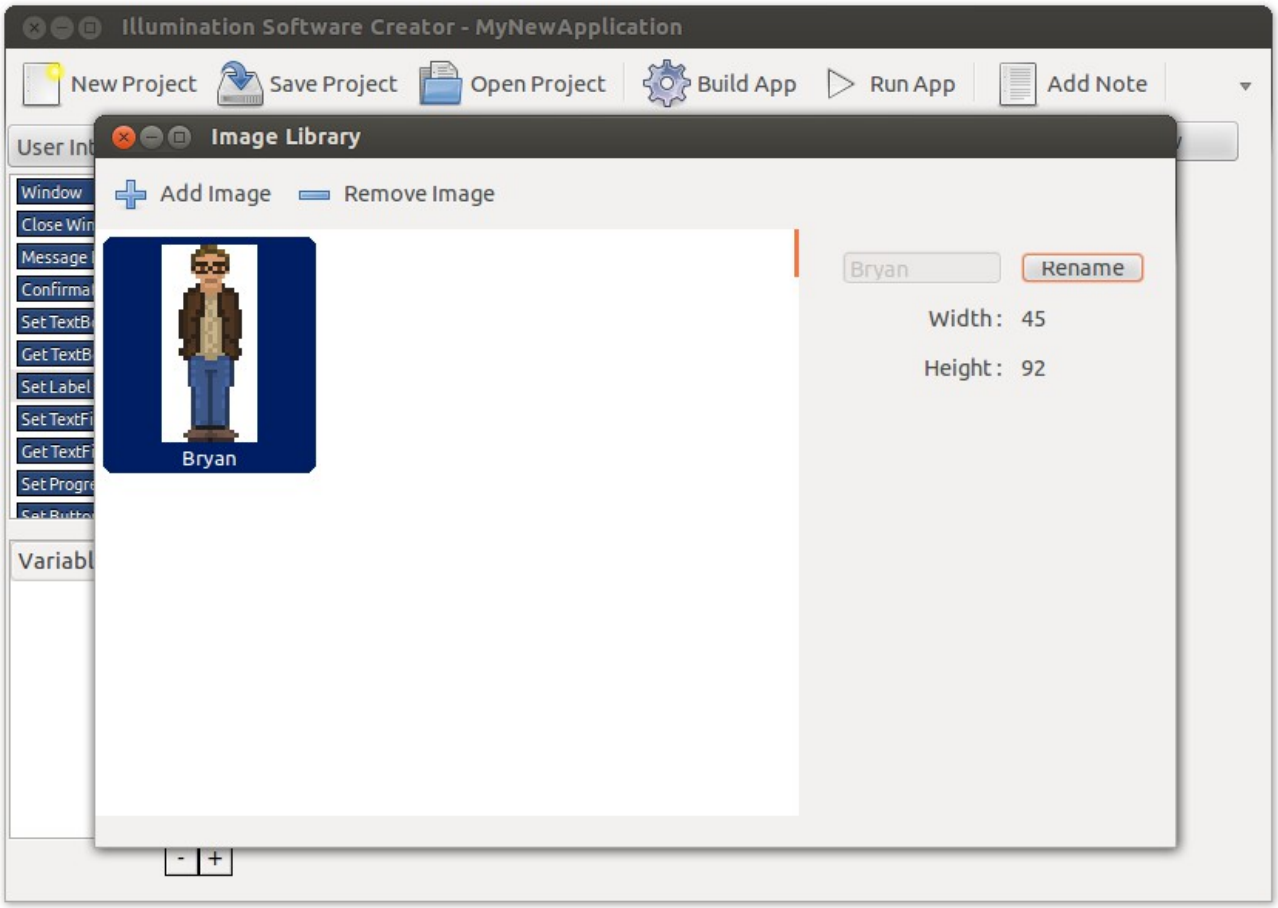
The Image Library is a simple way to add, modify and work with images within your application. To get you started, let's work through a quick example of how to add an image to your project and display it on a window.

Step 1) Add An Image

When you run Illumination Software Creator you'll notice a toolbar button on the top right that looks a bit like a roll of film and says “Image Library”. Click that and, as if by magic, the Image Library is opened.

The Image Library itself is a very simple thing. You can add and remove any .jpeg image you like, as well as rename and get information on the various image.

For now, simply add an image (any old jpeg will do... especially if it's a picture of your dog in a cowboy hat). When you've got at least one image in the Library, close the Library.

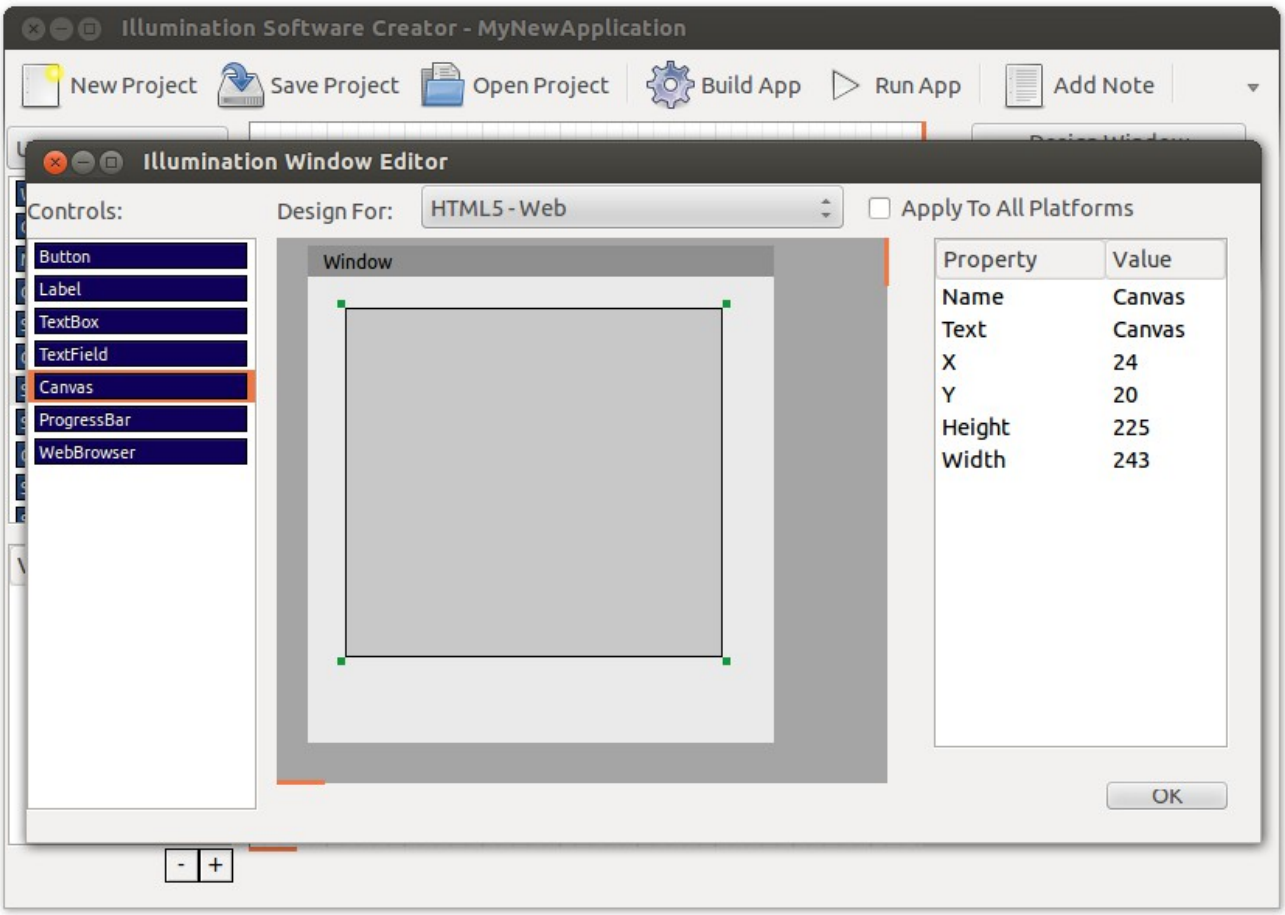


Step 2) Add A Canvas

In order to actually display images in a window, you'll need a Canvas.

Double click on the Window block to open the Window Editor and select “HTML5 – Web” from the drop down box at the top. (This tells the Window Editor that we want to design the user interface layout for the HTML5 version of the app – though what we learn here works exactly the same for all types of applications.)

Then drag a “Canvas” control from the list on the left to the window layout. Place it anywhere you like. Then close the Window Editor.



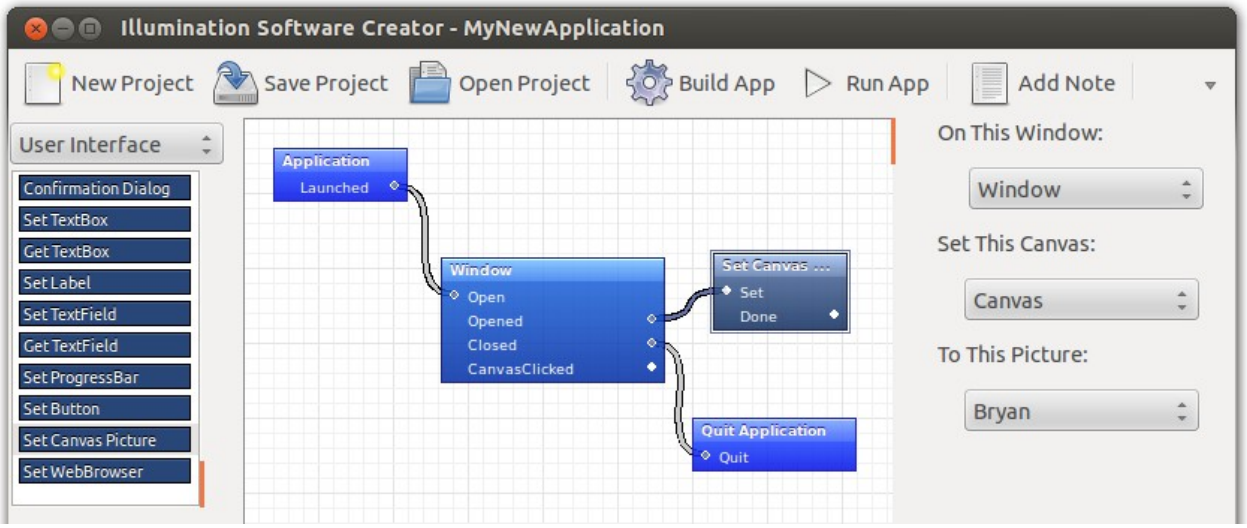
Step 3) Set Canvas Picture

In order to show a picture, we need to use the “Set Canvas Picture” block.

With “User Interface” selected in the top left hand corner drop down box, drag “Set Canvas Picture” from the list of blocks on the left to the main project grid.

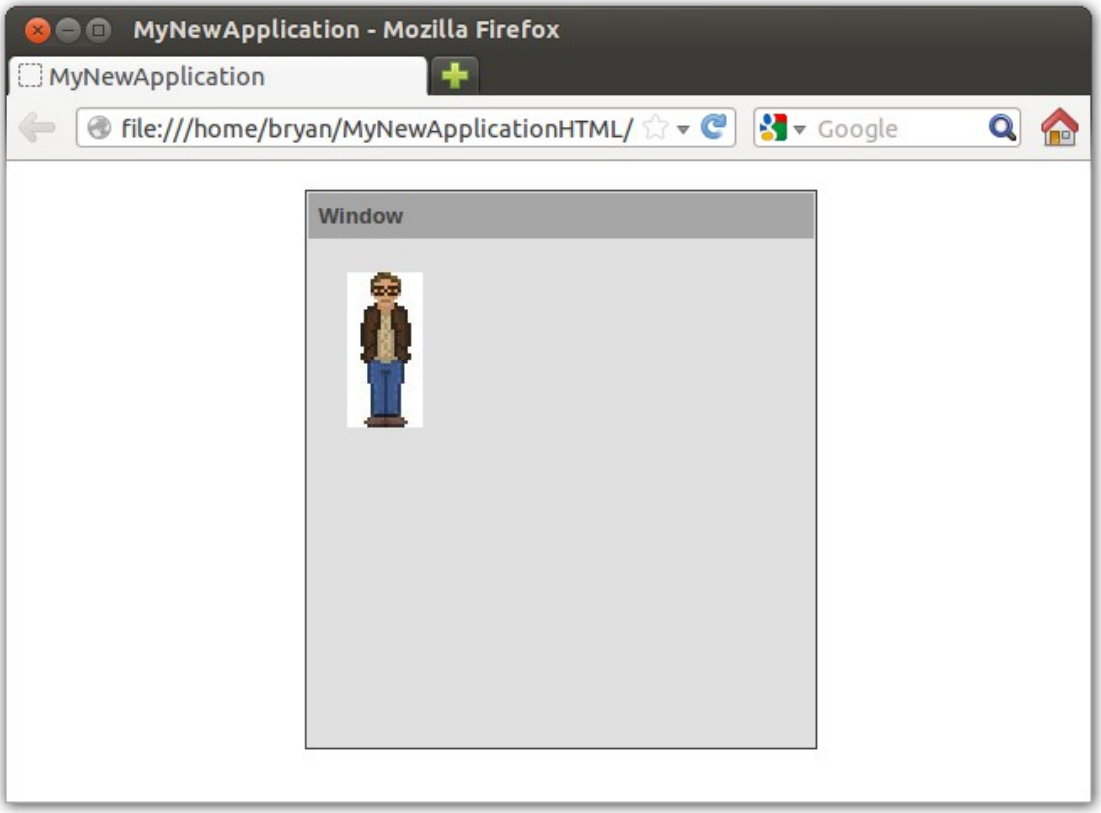
Now draw a line from the Window block's “Opened” output to the Set Canvas Picture block's “Set” input. This tells your application to set the canvas picture when this window is opened.
Now select the “Set Canvas Picture” block. Select the Window and Canvas you wish to display the picture on using the drop down boxes on the right.

Then choose which picture you wish to display from the Image Library from the bottom drop down box.



Step 4) Enjoy the fact that you are done.

That's really it. There's nothing left to do except run your application and enjoy.



Chapter 7

Portal Blocks

Portal Blocks allow you to have one block call another block... without having the two be visually connected.

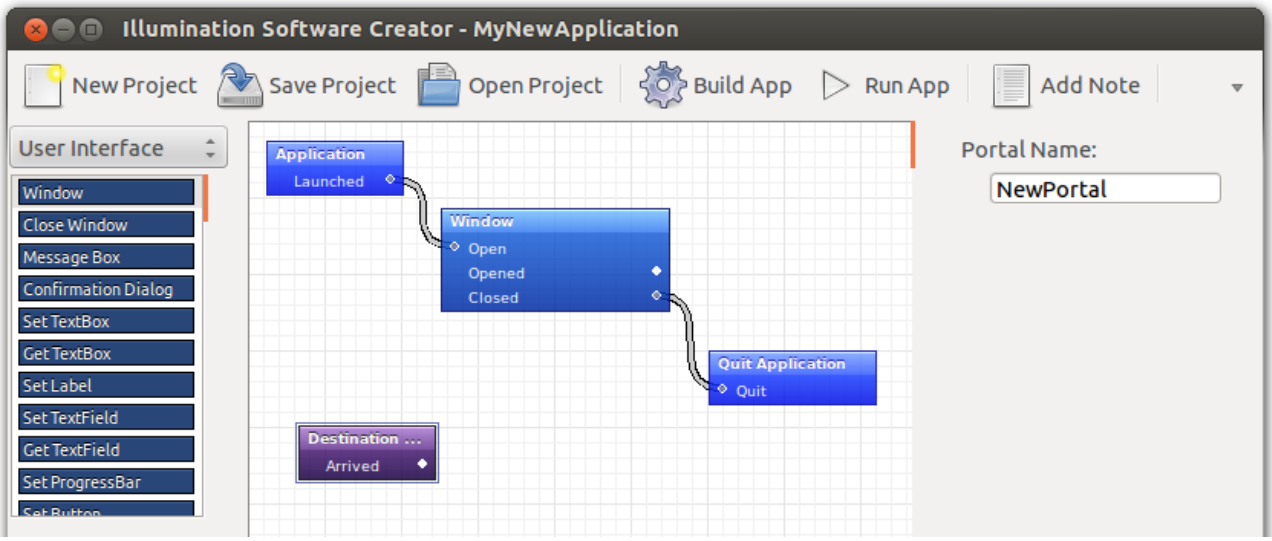
This can be incredibly useful if you need one block to call another block that is on the far side of your project (where the two blocks in question are not visible at the same time). It can also come handy to organize your project into discrete (and visually disconnected) chunks.

Here's how they work:

Step 1) Add A Destination Portal Block

Select "Logic" from the drop down in the top left then drag and drop a "Destination Portal" to wherever you want it to appear in your project.

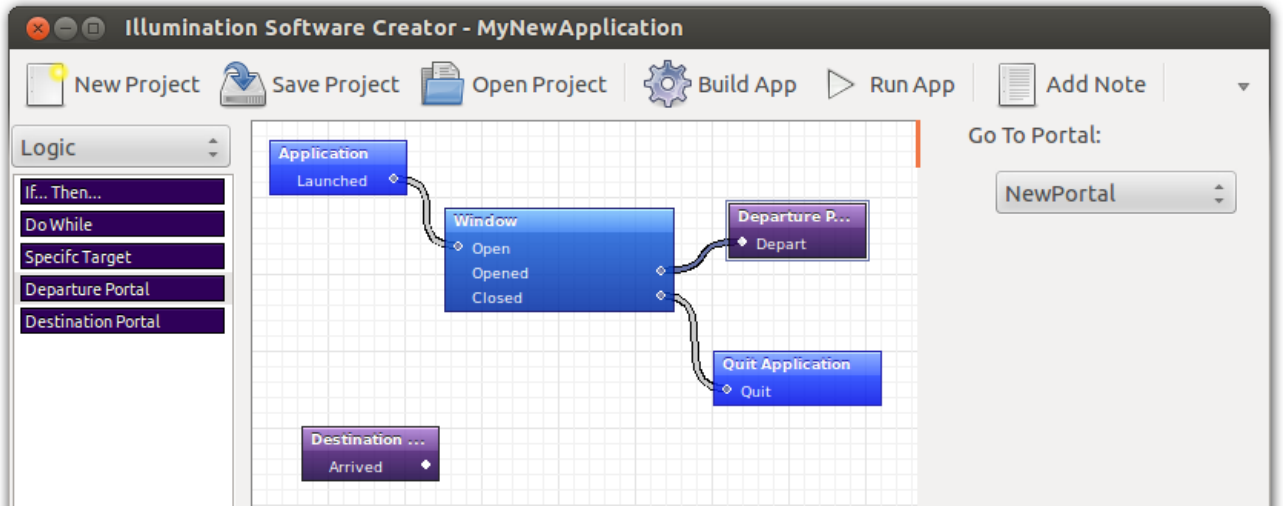
Now select that newly added Destination Portal and give it a Portal Name (on the far right properties view). This can be anything you like. Be descriptive so you'll remember what this Portal is for.



Step 2) Add A Departure Portal Block

Now drag and drop a "Departure Portal" to somewhere in your project. Select the new Departure Portal and choose which Destination Portal to go to from the "Go To Portal" drop down on the right.

That's it! Now when you link up to the "Depart" input of the Departure Portal, the logic of your software will jump to the Arrived output of the Destination Portal you selected.



Worth Noting:

You can call the same Destination Portal from as many Departure Portals as you like. In this means you can create what are, effectively, modular sections of logic within your project.

If you are used to traditional forms of programming, you can think of these as sort of a hybrid of "Methods/Functions" and "GoTo Statements".

Chapter 8

Building An HTML5 Web Application

Illumination Software Creator provides the ability to build and run your applications as 100% pure HTML and JavaScript based web applications.

What this means is that your applications will look and behave like standard desktop applications (including windows that you can drag around) that are completely contained within a webpage – and can be used from any HTML5 compatible browser.

As of the writing of this document, the following browser are supported:

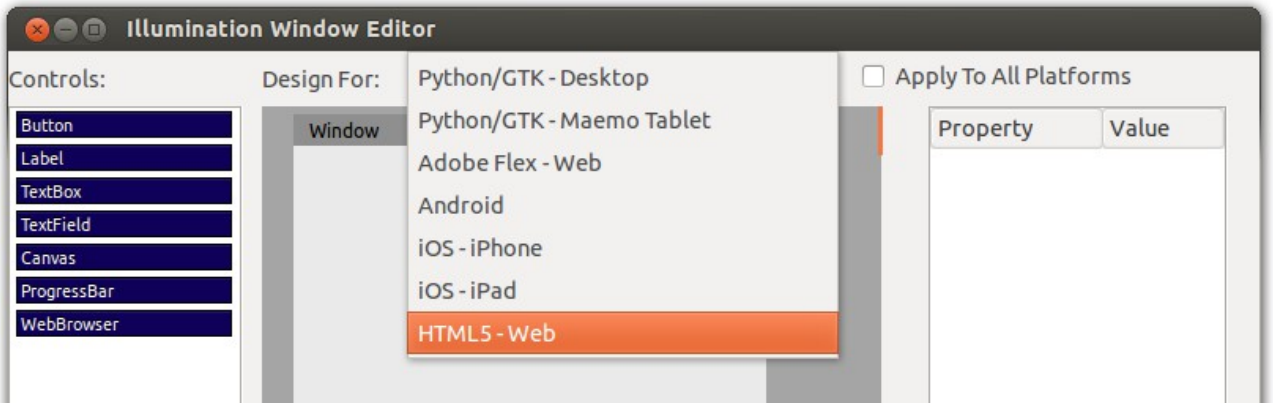
- FireFox 4
- Safari 5
- Chrome 10

Other web browser (and other versions of the browsers listed) may also work well with HTML5 Web Applications built with Illumination Software Creator – these are simply the ones most heavily tested at this time.

Step 1) Create Your Application

HTML5 Web Applications built with Illumination Software Creator will function the same way as the same application built for any other platform (iOS, Android, etc.). The logic and workflow of your application will not need to be changed in any way.

The only consideration you may have is to tweak the UI layout of any of your windows (which you can do by double clicking on any Window block and selecting "HTML5" from the layout dropdown box at the top of the UI Editor window).



Step 2) Build For HTML5

When you are ready to build and run your application, click the Build App (or Run App) toolbar button at the top of the main Illumination Software Creator window and select “Build HTML5 – Web”.
This will create a new folder that contains all of the files necessary to run your new web application.

Step 3) Sharing Your Application

When you are ready to share your app with the world, all you need to do is upload all of the files and folders created in Step 2 to your webserver. (Be sure to keep the folder structure in tact – don't rename any of the files.)

There are no server requirements for running this web application (other than the ability to serve up HTML web pages).

Chapter 9

Building an Android Application

Illumination Software Creator allows you to build applications, for all available target platforms, with no changes to your project.

However, when building Android applications there are a few additional items you will need to setup (outside of Illumination).

Step 1) Installing the Android SDK

You must have a functioning copy of Eclipse and the Android SDK installed in order to compile and run Android applications created with Illumination Software Creator. Both Eclipse and the Android SDK are made available free of charge.

You can download them, and find full instructions on setting them up for Windows, Linux or MacOS X from: http://developer.android.com/sdk/index.html

Step 2) Build your project for Android

Open your project Illumination and select "Build Project – Android" from the Project menu.

Once complete (this is typically a very fast process that takes only seconds) you will be presented with a folder full of files. These files are a complete, ready to go Eclipse project – including full source code and everything needed to build a complete Android application.

Open up Eclipse and go to the File menu, select Import, and choose General -> Existing Projects Into Workspace. Then select the root directory of your new Android project and click Finish. This will add your project into Eclipse.

At this point your project is all set and ready to run! Simply choose Run from the Run menu in Eclipse and select Android Application (only necessary the first time you run this project) and the Android Simulator will launch and load your application!

Things To Know

By default your project is set to use the Android SDK version 2.2. If you have this version of the Android SDK installed you are all set. If not, you can either follow the instructions on the URL in step 1 to install version 2.2, or you can right click on your project name on the right hand side of Eclipse, choose Properties, and choose a different SDK version from the Android section.

If you have difficulty setting up Eclipse or the Android SDK feel free to ask any question in the Illumination Software Creator Support forum at : http://www.lunduke.com/forum

Chapter 10

Building an iOS (iPhone / iPad) App

Illumination Software Creator allows you to build applications, for all available target platforms, with no changes to your project.

However, when building iPhone or iPad applications there are a few additional items you will need to setup (outside of Illumination).

Step 1) Installing XCode and the iOS SDK

You must have a current copy of XCode and the iOS (iPhone / iPad) SDK installed. You can find more information at: http://developer.apple.com/devcenter/ios/index.action

Installing XCode is only possible on MacOS X. However, you may use Illumination Software Creator on Windows or Linux to design your iPhone or iPad application and then, when you wish to perform the final step of "building" your iOS application, you will need access to a Mac.

Step 2) Build your project for iOS

Open your project Illumination and select "Build Project - iOS - iPhone" or "Build Project - iOS – iPad" from the Project menu.

Once complete (should only take a few seconds) you will be presented with a folder full of files. These files are a complete, ready to go XCode project – including full source code and everything needed to build a complete iOS application.

On your Mac, with XCode installed, double click on the .xcodeproj file in this folder. This will launch XCode and open your project.

At this point you are all set! Just hit 'Build and Run' from the XCode toolbar and you can run your new app in the iPhone simulator.

Things To Know

There is no technical difference between applications created with Illumination Software Creator, and those apps created with XCode alone. Your application can be modified, run on actual iOS devices and even submitted to the iTunes App Store.

If you have difficulty setting up XCode or building iPhone/iPad applications feel free to ask any question in the Illumination Software Creator Support forum at : http://www.lunduke.com/forum

Chapter 11

Building an Adobe Flex/Flash Web App

Illumination Software Creator allows you to build applications, for all available target platforms, with no changes to your project.

However, when building Flash based web applications there are a few additional items you will need to setup (outside of Illumination).

Step 1) Installing Adobe Flex SDK

Illumination Software Creator builds Flash web applications utilizing the Adobe Flex SDK. All you need is to download and install the latest version of the Adobe Flex SDK from here:

http://opensource.adobe.com/wiki/display/flexsdk/Flex+SDK

If you have difficulty getting the Flex SDK properly installed on your system, another approach is to download and install Adobe Flash Builder (which comes with a pre-setup installation of the Flex SDK). You can download a trial version here: http://www.adobe.com/products/flashbuilder/

Once you've done that, run Illumination and open the Preferences window (under the Edit menu). From there you can locate where you have installed the Flex SDK.

Step 2) Build your project for iOS

Open your project Illumination and select “Build Project – Adobe Flex - Web” from the Project menu or select “Run Project – Adobe Flex – Web” from the Run menu in the Project menu (this will compile your Flash web application and auto-launch the default web browser and run your new application).

Once complete (this is typically a very fast process that takes only seconds) you will be presented with a folder full of files. These files are a complete, ready to go Flash web application including an html file with code that will show you how to embed your new application on your own website.

Things To Know

The Adobe Flex SDK is freely available on MacOS X, Windows and Linux.

Chapter 12

Building a Python Application

Illumination Software Creator allows you to build applications, for all available target platforms, with no changes to your project.

However, when building Python applications there are a few additional items you will need to setup (outside of Illumination).

Step 1) Installing Python and GTK requirements

Python applications built with Illumination Software Creator can run on Linux, Windows, MacOS X and Maemo (N900 phones). However, depending on your platform, setting up the required Python (and GTK) libraries can be a little tricky.

For Linux:

1. Most major versions of Linux fully support Illumination-created Python applications out of the box. If not, simply install Python and PyGTK (example: "sudo apt-get install pygtk").

For Windows:

On Windows you will need to install Python and GTK support files.

Installing and configuring Python and PyGTK can be a bit tricky on Windows. Luckily the PyGTK team has put together a great guide and installer to walk you through it:

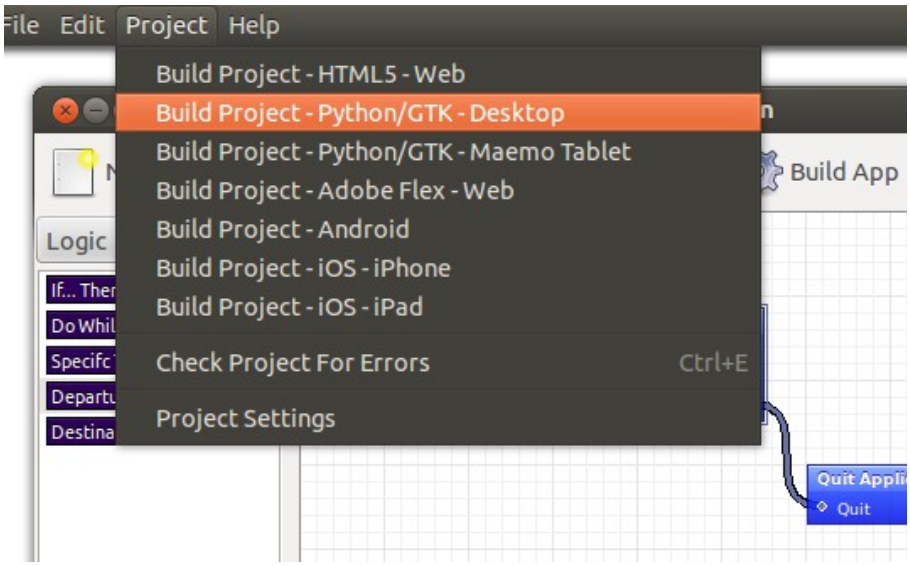
http://www.pygtk.org/downloads.html

For MacOS X:

MacOS X does not ship with support for Python/GTK applications by default. And, as of this writing, there was no easy to use (and supported) installer available. In order to utilize Python/GTK applications on MacOS X you will need to do a fair bit of leg work. See the forum (lunduke.com/forum) for more discussions.

Step 2) Build your project for Python/GTK

Open your project Illumination and select “Build Project – Python/GTK - Desktop” or “Run Project – Python/GTK - Desktop” from the Run menu on the Project menu (this will run the Python application directly).



Once complete (this is typically a very fast process that takes only seconds) you will be presented with a folder which contains your application as a fully prepared ".py" file. This is all that is needed to run your application, which can be done from the command line (ex: "python myapp.py").

Things To Know

Python applications are run via an interpreter. This means that the raw source code for your Python application is available to anyone you distribute your application to.

Chapter 13

Extending Your Software With Custom Blocks

Occasionally you may want to accomplish something that Illumination simply isn't capable of "out of the box". This is where Custom Blocks come in.

Custom Blocks allow you to use actual "code" to create new functionality that can be re-used in your Illumination project just like any other block. (For example, you can use your own JavaScript code in a Custom Block to extend the functionality of the HTML5 target.)

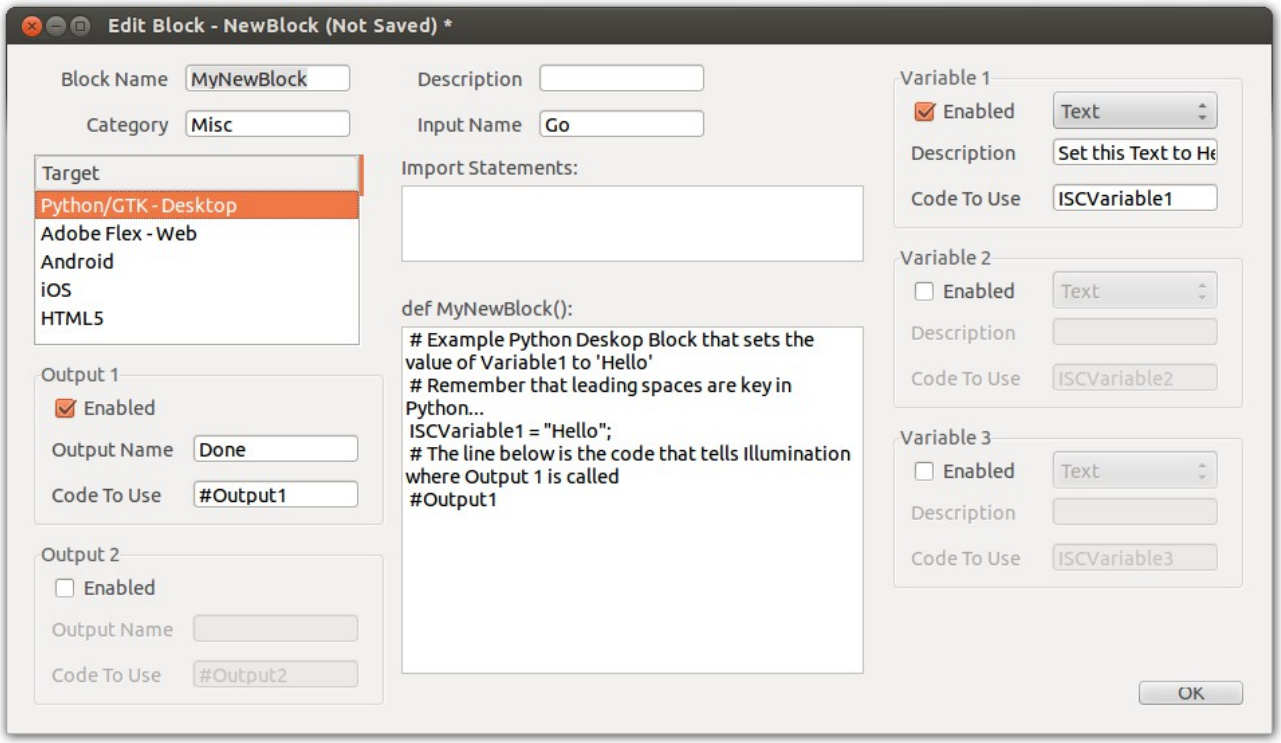
Each Custom Blocks is stored in its own ".isb" file. This file can be easily moved between machines (and even distributed to other people using Illumination Software Creator). In this way, it becomes possible to extend the capabilities of Illumination almost infinitely.

And, other than the part where you need to write some "code", the process is actually pretty easy. Let's walk through it step by step.

Step 1) Create The Custom Block

This part is nice and easy. From the File menu, choose "New Custom Block..."

This will display the Edit Block window.



Step 2) Set The Properties

Each Custom Block has a set of properties that you can set:
– Block Name : This is the name of the Block as you would see it within Illumination.
– Category & Description : These fields can be set to anything you like for your own reference (these will be used in a future version of Illumination for categorizing Custom Blocks).
– Input Name : This is the text for the input. Could be anything you like ("Go", "Start", etc).

Step 3) Choose Your Outputs

Custom Blocks can have up to two unique Outputs that you can name whatever you like. This allows you to have multiple possible branches of logic (perhaps your block is a true/false test... one Ouput could be named "True", the other "False").

Simply check "Enabled" next to the Output you'd like to use and set the text that will show for that Output within Illumination.

Step 4) Choose Your Variables

On the right hand side of the Custom Block editor is where you can set which project variables will be used by your Custom Block.

Whatever variables you set here (including the type of variable you'll require and the description) is what you'll set using the right hand side block property in the main Illumination project view.

Example: If you want your block to use a Number variable from your project, you'd check the "Enabled" checkbox for a variable and select "Number" from the drop down box.

Step 5) Write The Code

This is the tricky bit where you'll need to write some code (and so you'll need to have some ability to write code in whatever language you want this Custom Block for).

But here are a few things to know as you work on developing your Custom Block:

– Each Custom Block can be built for any individual target (HTML5, Android, etc) that Illumination supports... or all of them at once. You can toggle between the code for each target by selecting the target you'd like to work on from the list on the left.

– If you would like to call an output in your code, simply add the text from "Code To Use" for the desired Output to the spot in your project where you'd like the output to be triggered (typically this is something like "#Output1").

– The same goes for Variables. Whenever you'd like to use a variable within your code, simply add in the "CodeToUse" from the desired variable in your code.

– If your new Custom Block requires some additional import statements, be sure to add those statements to the "Import Statements" box in the center of the Custom Block Editor window.

That's it! Once you've created (and saved) a Custom Block you can use them by selecting the “Custom Block” from the block category drop down on the main Illumination window.

Chapter 14

Build A Choose Your Own Adventure Game

Remember those great “Choose Your Own Adventure” books? Well, building a game in the same style (similar, in many ways, to the Text Adventure video games of the good old days) is extremely easy – and fun! – in Illumination Software Creator.

The basics of a “Choose Your Own Adventure” game are simple: Some text is displayed to the player that describes where they are at -- and some things that they might do. Then a series of options are given for the player to pick from. When a selection is made, some new text is displayed with some new options.

Simple, right?

Well, there are many ways you could go about building just such a game within Illumination.
For this first tutorial we are going to focus on a simple, but flexible approach to building the basic game.

This tutorial assumes a basic working knowledge of how to get around in Illumination. Luckily, that's a pretty easy thing to pick up. Check the “Hello World”, “User Interface” and “Portal Blocks” tutorials to give you a good overview of what you'll need to know.

There is an example project supplied with this tutorial that shows a working example of what we go over here. Feel free to use it to get a jump start on your project.

Step 1) Set Up The Project

To make things simple we are going to make extensive use of Portal Blocks (if you are unfamiliar with Portal Blocks, now might be a good time to take a look at the Portal Block tutorial).

First, delete the Window and Note that Illumination creates with the default project.

Add a new Destination Portal, call it “Quit”, and hook up its Arrived output to the Quit Application block.

Then add a new Departure Portal and connect the Application block's Launched output to the Depart input. Don't worry about setting the “Go To Portal” property of the new Departure Portal just yet. We'll take care of that later (this will allow us to start the game at any point we wish).

The result should look something like what you see to the right.

Now let's get to the fun stuff.

Step 2) Design Your First Window

One of the great things about an adventure game is how flexible it is. Every place the player goes to

can be unique – with new things to see and new things to do.

Because of this, it becomes difficult to design a single user interface that will work perfectly for every single location in your game... So why not simply have a different window for every location?

There are some drawbacks to this method (which we will get into in detail in the next tutorial), but the flexibility of being able to have a completely different window for every location in your game (including different buttons, etc.) is almost too handy to do it any other way.

Plus... it's easy to do. Which is also nice.

So add a new Window to your project and double click that window to open up the Window Editor. Then do the following:
– Name your window something that will mean something to you. In this example I named mine "OpenField"... as, in my game, this window describes an open field.
– Add a label control and resize it to take up most of the window. Then place your text in the Text property. This will be what the player reads. It could be the description of a location or simply a small bit of action or dialog. [I recommend keeping a text document with this text for every window in your game, then simply copy and pasting that text from the document into the Text property.]
– Add a button that will represent a user action. Be sure to name the button something that will help you remember what that button does (such as "GoToHouse").
– Add any additional buttons for any other actions you want to present to the user.

When done, you will have something that looks approximately like the screenshot below.

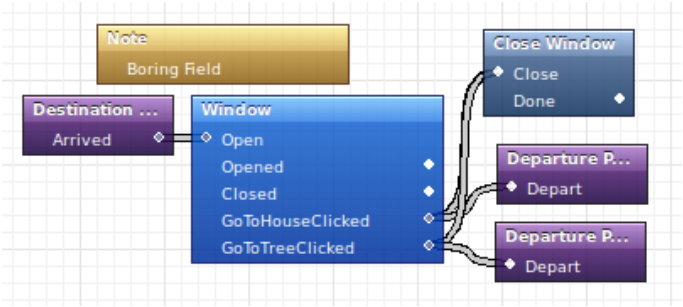
You can always come back and tweak the layout and content of your window later.

Step 3) Making It Do Something

Now that we've got our first window, let's finish adding the parts that make it... do something.

Do the following:
– Add a new Destination Portal, give it a name (could even be the same name you gave the window), and connect its Arrived output to the window's Open input. Now, any time we want the player to see this window, we simply need to use a Departure Portal to call this Destination Portal. Very convenient.
– Add a new Departure Portal for each of the buttons you created in step 2. Connect the button's clicked Output to the corresponding Departure Portal's Depart Input. If you already know the name of the Destination Block for the Window you want to display when this button is clicked, you can set it now.
– Add a new "Close Window" block, set it to close the window you are currently working with, and connect the Clicked Output of all of the buttons on this window to the Close Window's Close input. Now, the current window will disappear before the next window will appear.
– Add a new Note block and add a note about what this location does. Something that will mean something to you later as you are looking over your project.

Once you're done, you'll have something that looks very roughly like what follows.



Remember that Departure Portal from Step 1 (that we added to Application Launched)? Now you'll want to select that Departure Portal and select the name of the new Destination Portal that calls the Window Open Input.

Now, when you run your application, that is the first window that will open. (And, hence, the first location of your game.)

Step 4) Make It A Game

Of course, having one location would make for a pretty boring Choose Your Own Adventure game. So it's time to add more locations.

How, you ask? Repeat Step 3. Over and over again for each "location" or "choice" screen you want in your game.

In the example project you'll see a very simple game with only a handful of window's in use. But it gives a good demonstration of how you can lay your project out to allow your player to interact with your new game world.

At this point you have everything you need to create a Choose Your Own Adventure game. Beginning to end.

But what about graphics? Inventory?

Once you have mastered the principles behind those two features, you'll be able to build a wide variety of game types (including traditional role playing games, interactive fiction, interactive comic books and more).

Step 5) Adding Graphics

The focus of a Choose Your Own Adventure game is usually the words. But adding graphics to your game definitely spruces things up a bit. And there's no reason you couldn't have an adventure game like this use only graphics (no text at all).

Luckily adding, and using, graphics in Illumination is a cake-walk.

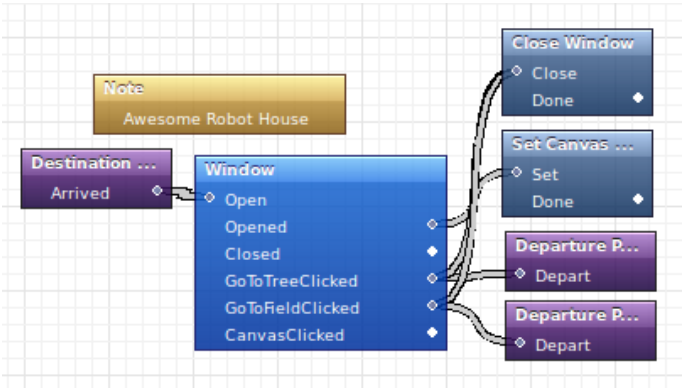
– Prepare the graphics files you would like to use. JPG and PNG files are supported by

Illumination.
– Click the “Image Library” toolbar button on the main Illumination window.
– Click the “Add Image” toolbar button and select the graphics file you would like to use in your game.

Now that you've got an image you would like to use, close the image library and do the following to your project:

– Double click on the Window block that you would like to use the new image in.
– Drag a “Canvas” control to your window and set its size and position to your liking (in the example project I set the Canvas to be 50 pixels wide by 50 pixels high, and moved it to the top left corner of the window).
– If you are going to have multiple Canvases on a single Window I recommend setting it's Name property to something that will be meaningful to you.
– Close the Window Editor (click “OK”).
– Drag a “Set Canvas Picture” block (in the “User Interface” section) to your project and put it next to the Window block you are working on.
– Connect the Window blocks “Opened” output to the “Set Canvas Blocks” Set input.
– Select the “Set Canvas Block” and set the Window that you are working with, the canvas name that you added to the window, and the picture you would like it to display.

When you are done, it should look something roughly like this.



Now, whenever this particular Window is opened (shown to the player), the image you added to your project will be displayed. Nice and easy, right? Once you've done it once, you'll be able to add new graphics to your game in a matter of seconds.

Step 6) The Inventory

What happens when you want to take your Choose Your Own Adventure game and add some more complex logic and interactions?

Let's say, you need a key to open a door? Or a phone to make a phone call?

Luckily adding features like inventory items and score keeping are easy as pie. It may seem like there are a lot of steps below, but it goes very quickly (and really isn't all that complicated).

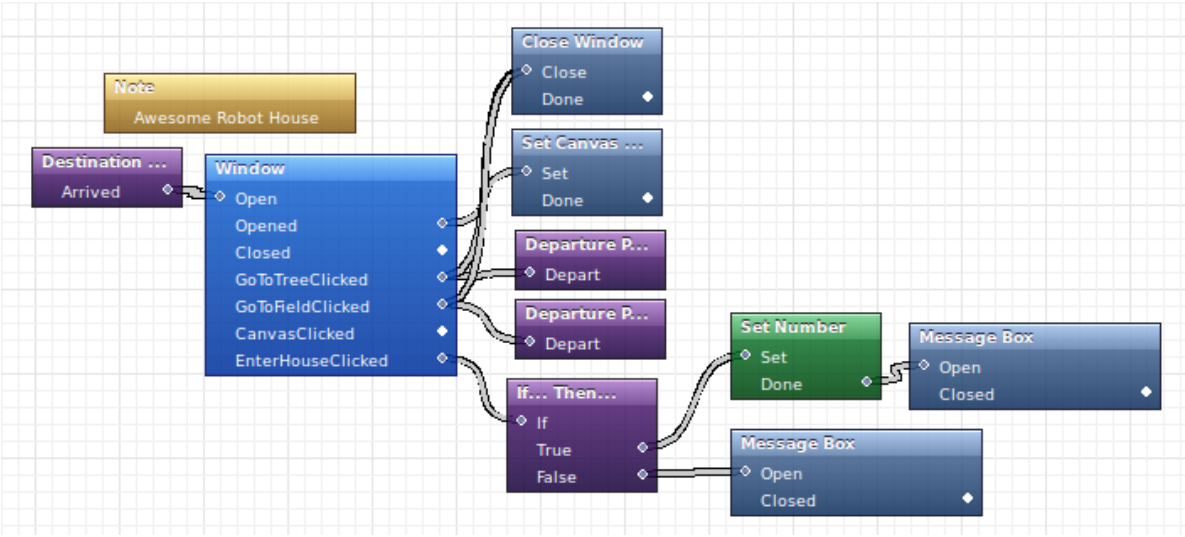
– Add a new variable to your project (click the “+” in the lower left corner) named “Keys”, type “Number”, default value 0 (zero). This variable we will used to track how many keys the player has.
– Add another new variable to your project named “0” (zero), Type “Number”, default value 0 (zero).
– Double click on the Window you want to add new functionality to (in the example project we are using the “Awesome Robot House”).
– Add a new button, set the text to “Enter House” and name it something that you will recognize (such as “EnterHouse”).
– Position the button wherever it makes sense to you. If you need to rearrange your existing controls, feel free to do so to your liking now. You can even make the Window itself larger to accommodate the new layout. Remember how in Part 1 we made each “location” have its own Window? This is why.
– Hit OK.

Now, back on the project we're going to add some very simple logic. Here, when the new “Enter House” button is clicked, we'll check to see if we have a key. And then do something based on that result.

– Add an “If... Then...” block (in the “Logic” section) to your project next to the Window you are working on. Connect the Clicked output of the button you just created to the “If” input of the new “If... Then...” block.
– Select the “If... Then...” block and set the Type to “Numbers”, checking if the variable “Keys” is “>” (more than) the number “0”. What we are doing here is seeing if there is more than zero keys. (ie. “We have keys we can use.)
– Add a “Set Number” block (in the “Numbers” section) to your project. Set the properties of this block to set the variable “Keys” to the custom number “0” (zero). This makes it so that, when the Key is used, the key is lost (inventory count is now zero). Connect the “True” output of the “If... Then...” block to the “Set” input of this block.
– Add two new “Message Box” blocks to your project.
– For the first one: Connect the “Done” output of the “Set Number” block you just added to this Message Box's “Open” input. Set the “Message Box” to show a custom message which reads “You use the key and the door opens! The door ate the key!”.
– For the second one: Connect the “False” output of the “If... Then...” block to the “Open” input of this block. Set the custom message to read “But... you have no key.”.
Now, whenever this new “Enter House” button is clicked, the player will be shown different messages based on if there is a key available or not.

From here you could do anything you like based on this. For example, if the user has a key, a new window with the inside of the house is opened. (For now we just show a message.)

You should have something that looks roughly like this.



But, wait! How does the player get a key? Here's a nice, easy way:

– Locate any window you like (in the example project we'll use the “Boring Tree”).
– Add a new “Set Number” block to your project.
– Connect the “Open” output of the “Boring Tree” window to the “Set” input of the new “SetNumber” block.
– For the “Set Number” block set the variable “Keys” to the custom number “1”.

This adds a key to the players inventory! Of course you can make this much fancier, and display this info to the player however you like (a message box saying “You got a key!”, modified the text for the window saying that “You pick up a key off the ground”... or you could even have the key variable changed when a player clicks a “Pick Up Key” button).

For any additional items you have, you simply need to add a new variable to store how many of that particular item you have... and any logic to check that variable (and add to it).

Step 7) Taking It Further

Now that you know how to implement an inventory, you can use the same principle to build a Score system. Simply add a Number variable named “Score”, add to it when you want the score to increment, and display it however you like (label controls on a Window's user interface, message boxes, etc.).

And now that you know how to work with the Image Library (for even more info here, see the “Adding graphics with the image library” tutorial), you can build advanced user interfaces. The “canvas” control that you use to display your graphics can also be used as a button. Notice that “CanvasClicked” output on the Window above? You could easily have the player use an inventory item, or go to another location, when any canvas is clicked.

The sky's the limit.

See the accompanying project to see this in action.

Chapter 15

Where To Go From Here

Now that you've got a handle on Illumination Software Creator, where do you go from here? Whether you are using it to prototype large projects, build mobile apps... or make Custom Blocks for usage in other people's projects, the sky is the limit!

Here are a few online resources to help you out along the way.

Online Resources

The latest version can always be found at : http://lunduke.com/

The official Illumination Forum : http://lunduke.com/forum/