dokidoki Manual

kesumu

March 2016

Chapter 1

Overview

dokidoki aimed at being a AVG development tool based on unity3d. It could improve the efficiency to develop AVG games and also could implement the one-time-code-multiple-platform-run based on the features of unity3d.

1.1 Features

1.1.1 Open source

dokidoki project is open source on github.

1.1.2 More clear logic for AVG game

More clear logic. In the game built by this tool, there exists only 3 main parts: WorldControl, World (contains characters), UI. WorldControl is used for controling the flow of the game. World is the container of apperence in this game world, and also character stands on the stage named world. UI is for setting of the game and in game operation.

As the game proceeds, only actions which written in scripts are excuated.

1.1.3 Easy to write script grammer

To be easy to use, we try to make the script of AVG game is as muck as easy to write, just like you are writting Screenplay.

1.1.4 Write once, run on all platforms

Yes, because this tool based on unity3d, it could run on any device the unity3d supports.

Chapter 2

Installation

First go to dokidoki project page on github to download all source code: dokiUnity(Unity assets), dokiScriptCompiler(compile program), dokiScriptSetting(script definition shared by dokiUnity and dokiScriptCompiler).

We suggest to write dokiScript on atom that is a text editor that's modern, approachable. dokidoki project has a package named language-doki-script to support your script writing, which you could download in atom.

After you wrote and compiled all scripts, you should also put those compiled scripts into Unity assets with your other resources(paintings, voices, videos and so on). Finally, the AVG game you create should be built via Unity.

2.1 Develop AVG game on Windows

Download latest dokidoki project source code, atom and Unity.

2.1.1 dokidoki

Inside the dokidoki project folder you downloaded, you would see structures below. There is some explaintation about it.

```
+---doc
|   +---api
|   |   +---html
|   |   \---latex
|   \---manual
+---sample
|   +---android
|   \---windows
\---src
    +---dokiEditor
    |   \---SampleScripts
    +---dokiScriptCompiler
    +---dokiScriptSetting
    \---dokiUnity
```

- **doc**: contains all documents.

  - **api**: contains api documents as reference.
    * **html**
    * **latex**
  - **manual**: contains documents on HOW-TO and architecture design behind this projects in detail.

- **sample**: contains sample game on multiple platforms.
  - **android**: sample game on Android.
  - **windows**: sample game on Windows.

- **src**: contains all source code.
  - **dokiEditor**: sub-project for script writing, because language-doki-script(atom package) is transfered into another GitHub repository, now it only contains sample scripts.
    * **SampleScripts**: sample scripts as reference for script writing.
  - **dokiScriptCompiler** : sub-project used to compile dokidoki scripts
  - **dokiScriptSetting**: sub-project used to define the script keywords and compiled output
  - **dokiUnity**: sub-projects used to combine all resources together, and build the game on all platforms that Unity supports.

What you need from dokidoki project is that: **dokiScriptCompiler** to compile your dokidoki scripts, and **dokiUnity** to combine all your game resources and build your game on Unity.

Inside **dokiScriptCompiler** folder, it is a C# source code project. To open it and use it as a compiler, you could run it with Xamarin directly, or build it into runable program(named dokiScriptCompiler.exe in default) with Xamarin.

2.1.2 Atom

Atom is good editor for code writing, and we have a package named language-doki-script to support you to write dokidoki scripts. Take a look on document for atom, you could install this package after you install Atom.

2.1.3 Unity

We already have a Unity project inside **dokiUnity** folder, you could open it directly and go on this project for your own game. Or you could create a new project and import all assets inside this project.

2.2 Develop AVG game on Mac OS

2.2.1 dokidoki

Almost the same with Windows.

The only thing different is that you could not build **dokiScriptCompiler** into a runalbe program and run it(it is a .exe program on Windows). However, you could run it using Xamarin directly.

2.2.2 Atom

The same with Windows.

2.2.3 Unity

The same with Windows.

2.3 Develop AVG game on Linux

To be tested.

Chapter 3

Get Started

3.1 Run sample games

Inside **sample** folder in dokidoki, there are sample games on multiple platforms. You could run it to see what games dokidoki could build.

3.2 Develop my first game

Here is the step-to-step tutorial to develop a AVG game based on dikidoki. To start development of your game, firstly create a new folder named MyFirstdokidokiProject (In my case C:/_Projects/MyFirstdokidokiProject).

3.2.1 Write my first dokiScript

Open your Atom, new file, and save it to disk (../MyFirstdokidokiProject/scripts/MyFirstScript.dks). **scripts** folder name could be any other name, but it is better not to contain space.

Then start to write your dokidoki scripts. Write the script just the same as below. To learn more detail about scripts writing, you should take a look at Chapter 5. Though you don't know much about dokiScript's grammar now. You could still understand the script easily, for that this script language is designed to be intuitive.

```
world
  background src=ba0 transition=instant;
  bgm src=bg0;
  >There are someone stand beside the school gate.
  >She looks anxious.

Mary
  role type=character name="Mary";
  posture src=mary000;
  voice src=mary000 >It is almost the time.

world
  >I find she is there.
  >And I walk towards her slowly.
  >However, she does not notice me.
```

```
I
  role type=player name="Jack";
  voice >Hello, Mary.

world
  >I said to her.
  >Heared what I said, she turns happy face immediately.

Mary
  voice src=mary001 >I am happy to see you again, Jack.
```

After wrote this above dokiScript code, save it and close Atom.

3.2.2 Create my first dokiUnity project

Setup dokiUnity template

Then, create dokiUnity project to combine all your game resources. Open Unity, create a new project named MyFirstdokiUnityProject. Then import custom package from **dokiUnity** folder named **dokiUnityTemplate.unitypackage**. **dokiUnityTemplate.unitypackage** contains dokidoki's Unity parts and folder structure to arrange all game resources. Each folder has a **readme.md** file to explain about what files should be put here. Up to now, you have setup the dokiUnity template.

Compile my dokiScript files

Your dokiScript files have extension **.dks**. You should compile them using **dokiScriptCompiler** before you put them into dokiUnity project.

Open **dokiScriptCompiler** sub-project in folder **/dokidoki/src/dokiScript-Compiler** with **Xamarin**. To compile all your dokidoki scripts, on the menu bar, click **Run—Run With—Custom Parameters**, insert your scripts folder absolute path into **Arguments** row, and then click **Execute**. If your scripts don't contain any syntax errors, **dokiScriptCompiler** would generate compiled dokiScripts with **.txt** extension for each script files.

Put all game resources into responding folders

With dokiUnity template, you could start to put your game resources into responding folders. See the dokiScript(not compiled one) you write, you could find that **src** key parameters have values like **ba0**, **bg0** and **mary000**. These values are names of your responding game resource files, but they are all without extensions.

- You should have a background image file named **ba0** with proper extension (maybe ba0.png) and put it into folder **/Assets/Resources/-World/Backgrounds**.
- You should have a bgm audio file named **bg0** with proper extension and put it into folder **/Assets/Resources/World/Bgms**.
- You should have a posture image file named **mary000** with proper extension and put it into folder **/Assets/Resources/Characters/Postures**.

- You should have a voice audio file named **mary000** with proper extension and another voice audio file named **mary001** with proper extension, and put them into folder **/Assets/Resources/Characters/Voices**.

And most importantly, your compiled dokiScript file (script with **.txt** extension) is also a kind of game resources. You should put all of them into folder **/Assets/Resources/DokiScripts**.

3.2.3 Build my first game

Have a run for my game

Now, you almost complete your game. Once you put all your game resources into responding folders. Use Unity to run your game, click **play** button on the **Toolbar** and your game would start. It is the most exciting moment for me too.

Build for Windows, Mac OS

Nothing different from normal Unity games build. On the **Topbar** click **File**—**Build Settings**. You would see a pop-up window, on the left option for platforms, choose **PC, Mac, Linux Standalone**, then on the right side choose **windows**, finally click **Build** choose a folder to export the runable game (in my case **/MyFirstdokidokiProject/windows**). Then, Unity would start to build your game, after it finished. Check the export folder you chose before, you would see your game is already there. Wait for what? Play it!

Build for Android

In order to build for mobile platforms like Android, the extra work you need to do is to move all your video resources from original folder **/Assets/Resources/World/Videos** into new folder **/Assets/StreamingAssets/World/Videos**. The other part is similar to **Build for Windows, Mac OS**.

Chapter 4

Architecture

System is responsible for arranging every thing inside dokidoki system. It contains a world which has lots of characters and one player. These characters take behaviors to interact with each other to progress the game. Behavior is a kind of action. Action is the minimum unit of things that happens in the AVG game, such as the position changing of a character, a sentence the character spoke. Effect is another kind of action such as the start of playing a sound, the start of the video and so on.

4.1 Overview

4.2 dokiScriptSetting

4.3 dokiScriptCompiler

4.4 dokiUnity

Chapter 5

dokiScript

dokiScripts screenplay written by AVG game developer, and script screenplay file would be compiled into a list of actions.

5.1 Example

Here give an example.

```
world
video src=video0;
bgm src=bgm0 mode=loop;
background src=background0 transition=instant;
weather type=snow level=0.2;
>??????????
>????????????
sound src=sound0;
>????????


dokiChan
role type=character name = "??";
move position=center transition=instant;
posture src=posture0;
voice src=voice001 >>??????
voice src=voice002 >>???????
voice src=voice003 >>?????????

["????????????????"(option011, sample1) | "???????????????"(
    option012, sample1)]


<option011>


world
>>?????????

I
role type=player;
voice >>??????????????

dokiChan
move position=(0.45, 0.0, 0.0) transition=instant;
```

```
voice src=voice004 >>??????????

I
voice >>????????

(sample2)

<option012>
I
>>?????????????????????

world
>>???????????????????
weather type=sunny;
>>????????

(sample3)
```

5.2 Grammar

	Example	Description
Doki		Mostly, one script file
Flag	`[????????????????(option011, sample1) | ???????????????(option012, sample1)]`	Flag with options to choose
Option	`<option011>`	Option that Flags should jump here when this option is chosen
Lock	`{ xxx }`	For some performance, disable players? operations

Block	`world`
	`video src=video0;`
	`bgm src=bgm0 mode=loop;`
	`backgound src=background0 transition=instant;`
	`weather type=snow level=0.2;`
	`??????????>`
	`????????????>`
	`sound src=sound0;`
	`????????>`
		One block of script code, start with one focused Object, maybe world or characterId (here means those lines code would focus on the world)

Ac-tion	`bgm src=bgm0 mode=loop`	Action, focused on current
Tag	`bgm`	Action tags
Key=Value	`src=bgm0 mode=loop`	Parameters for current action
Key	`src`	Attributes key
Value	`bgm0`	Attributes value

Text	??????????	

Chapter 6

dokiUnity

Chapter 7

dokiBattle