X/Text

Team Members:

Xiaotian Zhang

Zhiheng Xu

Qian Zhu

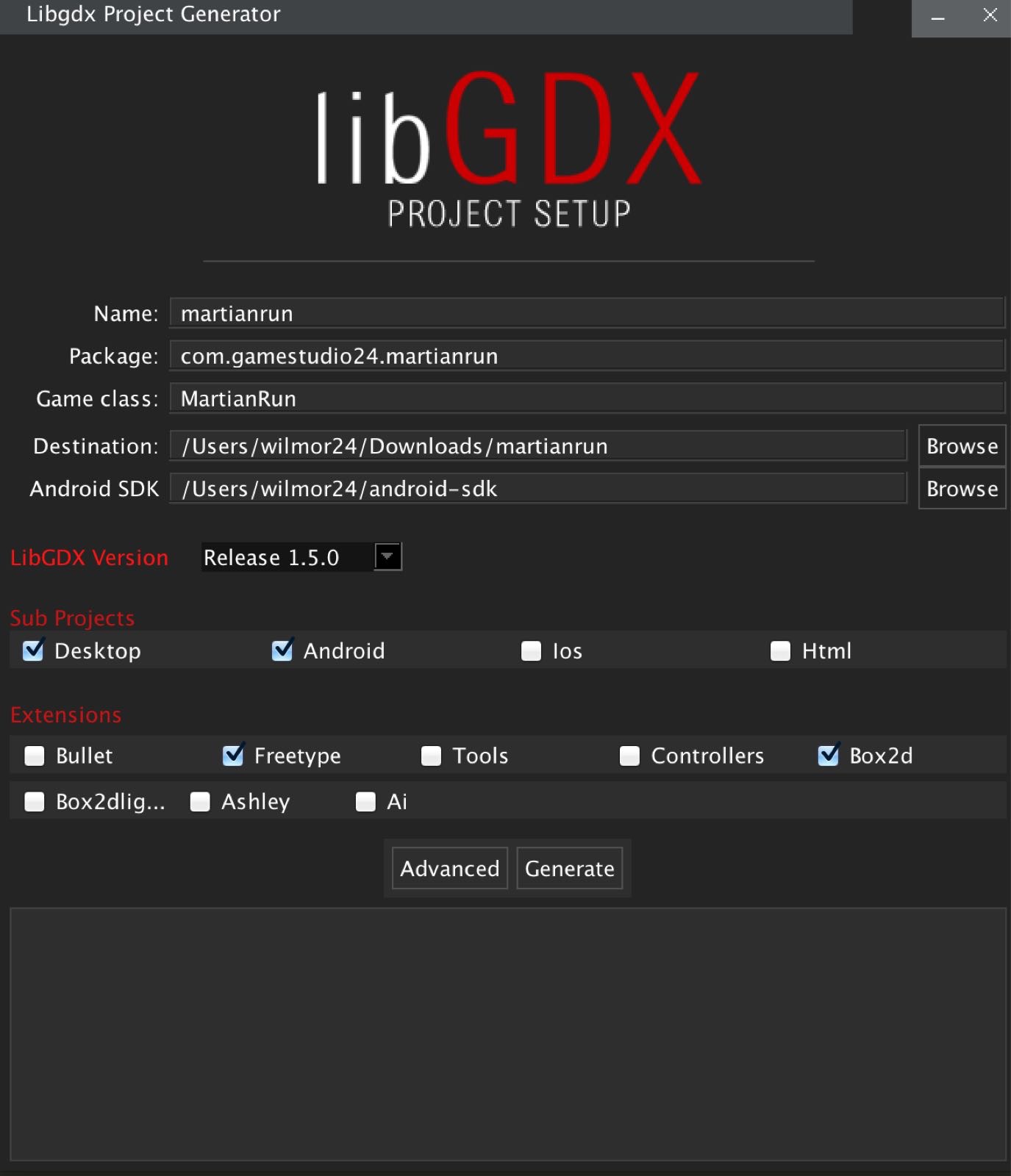
1.Introduction

What we have done in this project is to find out a way to represent different fonts of the text in the big project XDF. At the very beginning, it was very hard to use freetype library in java. After a few days of searching for materials we finally found out a way to implement that library. And the next whole pages are going to introduce this way and show a small demo we did in this project.

2. Environment Setting

2.1 libGDX

Libgdx is a cross-platform game and visualization development framework. It currently supports Windows, Linux, Mac OS X, Android, Blackberry, iOS, and HTML5 as target platforms. Libgdx allows you to write your code once and deploy it to multiple platforms without modification. Instead of waiting for your latest modifications to be deployed to your device or to be compiled to HTML5, you can benefit from an extremely fast iteration cycle by coding your application mainly in a desktop environment. You can use all the tools of the Java ecosystem to be as productive as you can be. Because it is on the JVM, you can also feel free to use other awesome and popular non-Java languages (Kotlin, Scala, Clojure, etc) Libgdx lets you go as low-level as you want, giving you direct access to file systems, input devices, audio devices and OpenGL via a unified OpenGL ES 2.0 and 3.0 interface.

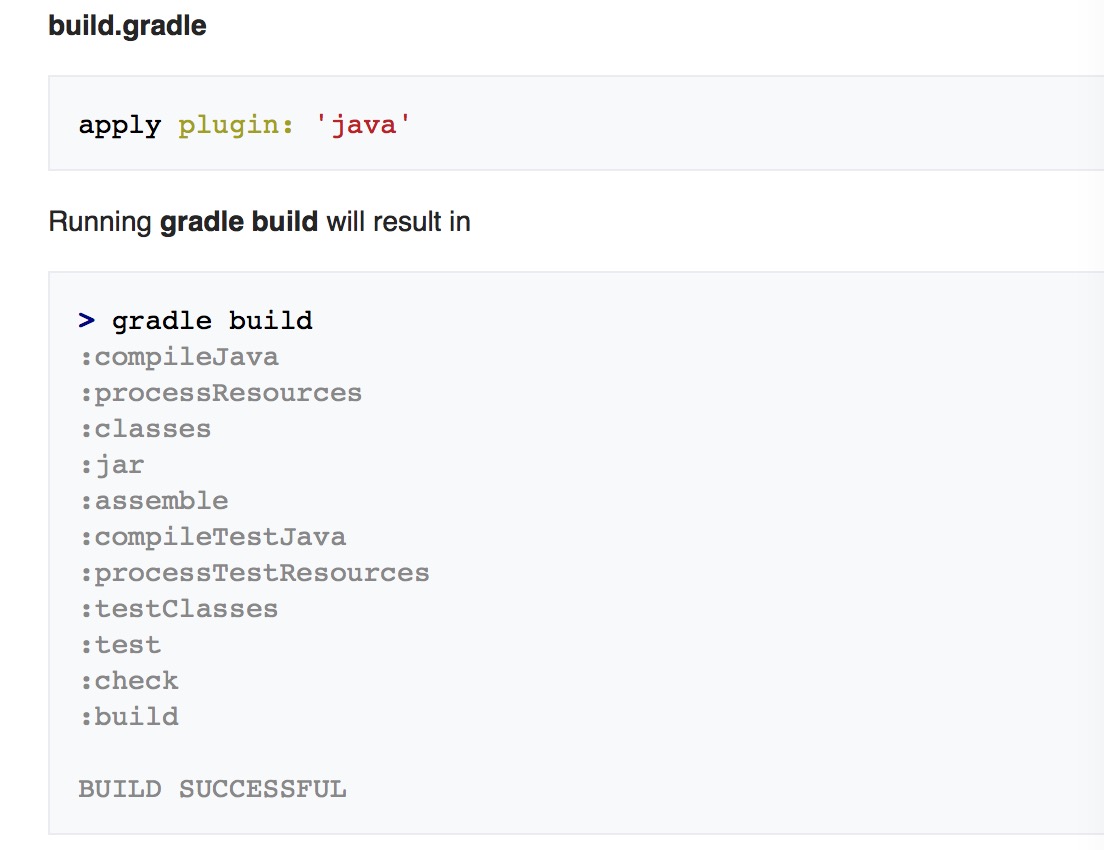


*Image: An example of setting libGDX framework.*

2.2 Gradle

Gradle is an open source [build automation](https://en.wikipedia.org/wiki/Build_automation) system that builds upon the concepts of [Apache Ant](https://en.wikipedia.org/wiki/Apache_Ant) and [Apache Maven](https://en.wikipedia.org/wiki/Apache_Maven) and introduces a [Groovy](https://en.wikipedia.org/wiki/Groovy_(programming_language))-based [domain-specific language](https://en.wikipedia.org/wiki/Domain-specific_language) (DSL) instead of the [XML](https://en.wikipedia.org/wiki/XML) form used by [Apache Maven](https://en.wikipedia.org/wiki/Apache_Maven) of declaring the project configuration.[[2]](https://en.wikipedia.org/wiki/Gradle#cite_note-2) Gradle uses a [directed acyclic graph](https://en.wikipedia.org/wiki/Directed_acyclic_graph) ("DAG") to determine the order in which tasks can be run.

Gradle was designed for multi-project builds which can grow to be quite large, and supports incremental builds by intelligently determining which parts of the build tree are up-to-date, so that any task dependent upon those parts will not need to be re-executed.



*Image: An example of implementing gradle build automation system.*

3.Freetype Implementation

3.1 Freetype Introduction

If you want to draw text in your game, you usually use a BitmapFont. However, there is a downside: BitmapFonts rely on an image, so you have to scale them if you want a different size, which may look ugly. You could just save a BitmapFont of the biggest size needed in your game then and you never have to scale up, just down, right? Well, that's true, but such a BitmapFont can easily take up two times as much space on your hard drive as the corresponding TrueType Font (.ttf). Now imagine you have to ship all your big BitmapFonts with your game and your game uses ten different fonts... on an Android device. Bye bye, free storage!

The solution to your problem is the gdx-freetype extension: ship only lightweight .ttf files with your game generate a BitmapFont of your desired size on the fly user might put his own fonts into your game

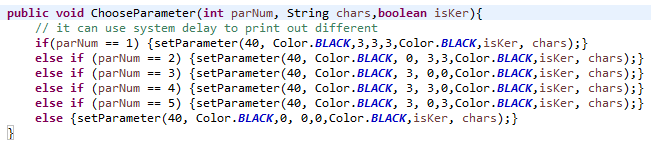
Since this is an extension, it is not included in your LibGDX project by default. How you add the extension differs based on the setup of your project.

3.2 Create Own FontFactory

This is the core of the project. Through the introduction ahead we can create our own font factory class. But before we create that class, we need a parameter that inheriting from the class FreeTypeFontParameter. So that we can use the freetype library to create an object of BitmapFont.

**3.2.1 MyParamter**

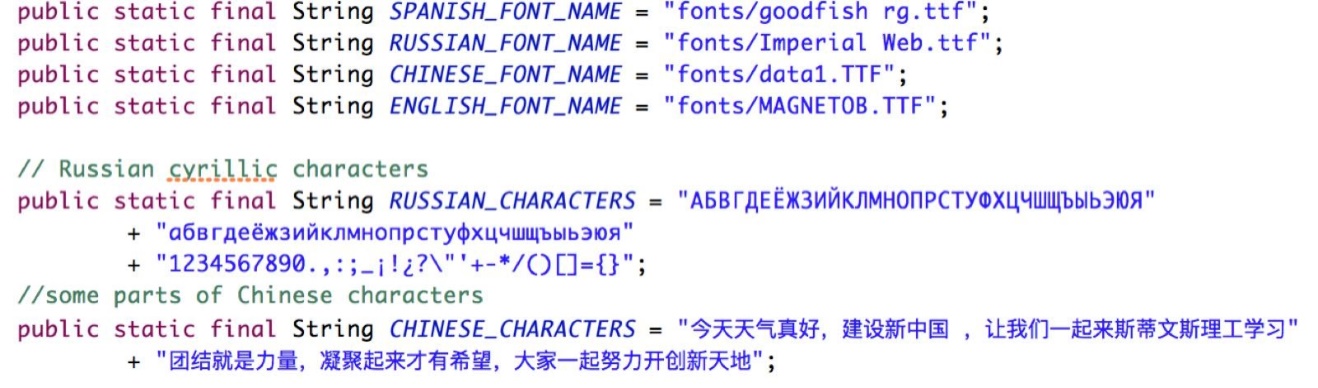
This is a class we create on our own which inherits from the FreetypeFontParameter. We create four functions to change the fonts into different types. Such as, set size, color, shadow, shadow size, shadow color, and kerning. In this program, there are too many combinations so that we choose 6 fonts’ parameter and let it show on our screen.



*Image: An function which set six different parameters of the fonts.*

**3.2.2 FontFactor**

This is a class we create to generate different language with different fonts Firstly we need to have the fonts files which ends with .ttf or .TTF and implementing them into our class by the gdx.internal function. And then we input the characters which are not only the default English characters. We also create Russian characters, a few Chinese characters and all of the Spanish characters which is included by the default characters. After that, we create the function generatefont function and the initialize the object BitmapFont so that we can continually create it in our screen class which would be introduced later.

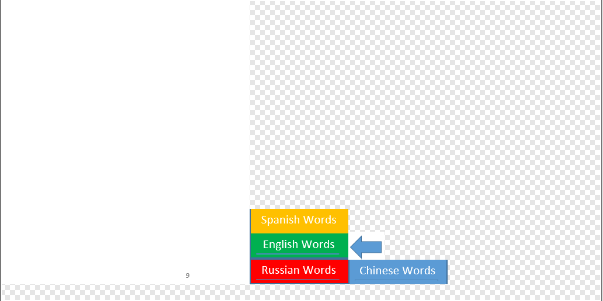


*Image: Different languages with different fonts file and characters used by program.*

4. Screen Implementation

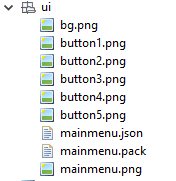
4.1 Main Menu Screen

In this part, we are going to the visualization parts which gives you an introduction of how we are going to let the different texts show on our screen. As we introduced before we have four screens of different languages to show. So that we need four classes that are all implement the interface of screen. Also in this situation we need a menu screen in our application so that we can just click the button and switch the screen.



*Image: Menu screen resources with four buttons and back button that can be clicked and switch to different languages.*

In this part we need to add files to our local directory. Also we need four button pictures and pack the four pictures of the button and the white background screen picture also give the background a color which in this demo being blue. The pack file can be created by a java api called gdx texturepacker. And also we studied a little bit example to write the JSON (JavaScript Object Notation ). Resources in a skin typically come from a [texture atlas](https://github.com/libgdx/libgdx/wiki/Texture-packer#textureatlas), widget styles and other objects defined using JSON, and objects added to the skin via code. Even when JSON is not used, it is still recommended to use Skin with a texture atlas and objects added via code. This is much more convenient to obtain instances of drawables and serves as a central place to obtain UI resources. So that we can make the program be more efficient.



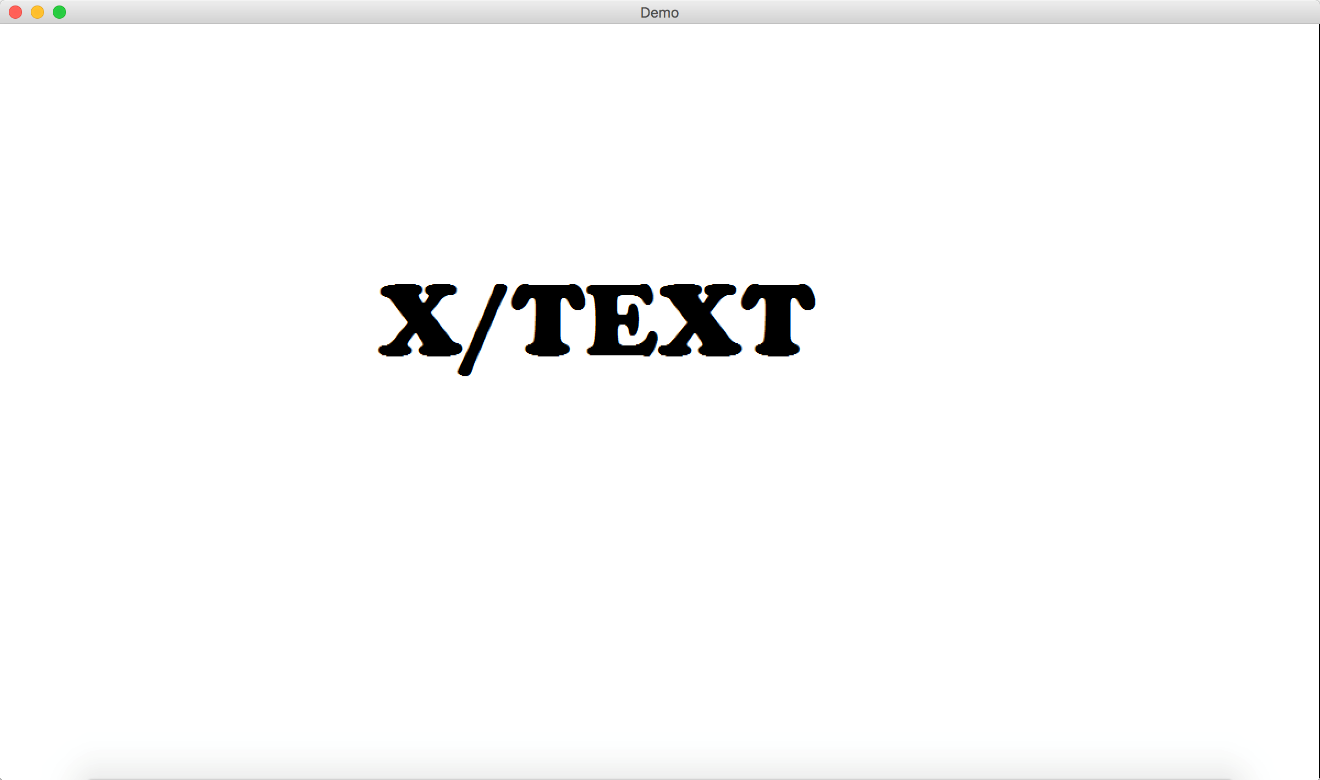
*Image: The images and the json and*

*pack file we need to use in our program.*

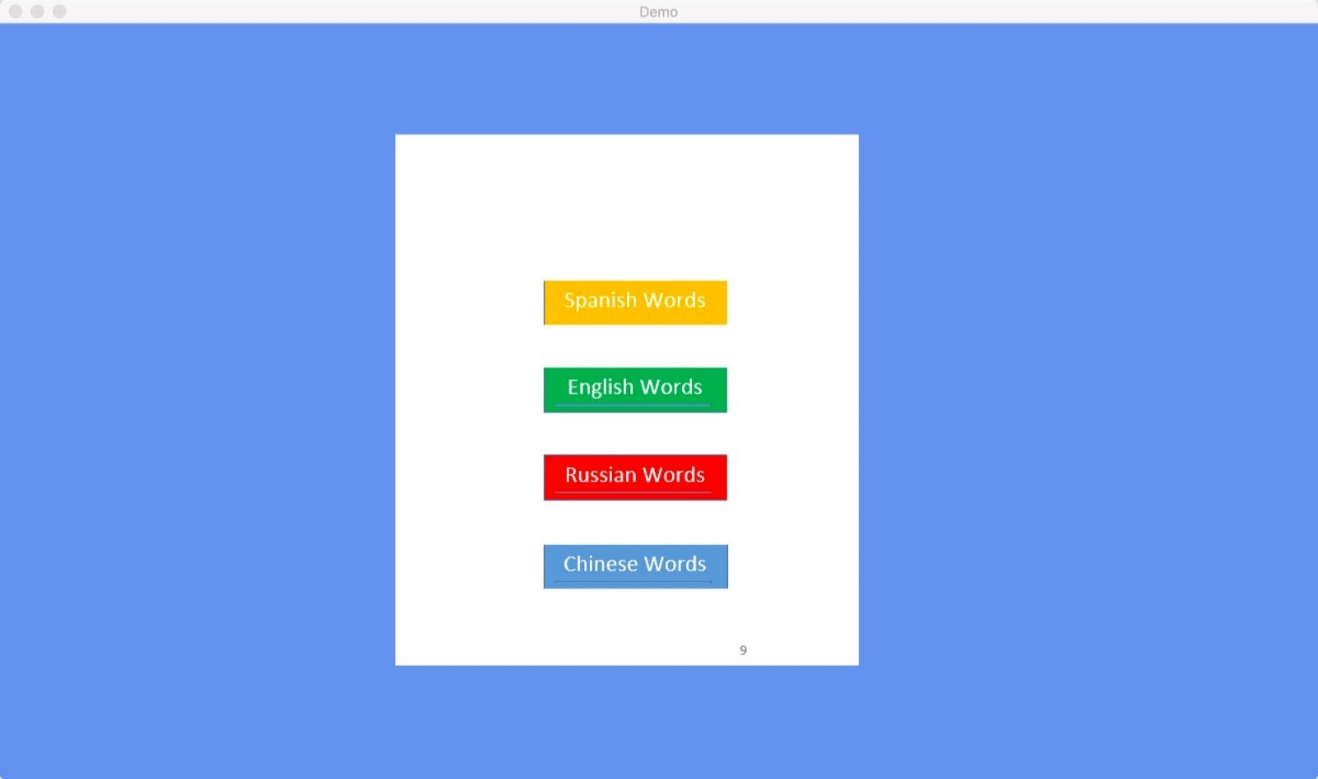
4.2 Different Language Screens

After we create the class of menu screen, we need to create the four classes of language screens. We implement the font factory class and create these BitmapFont objects and let write them into show function in the class. The results of the screens would show in next chapter.

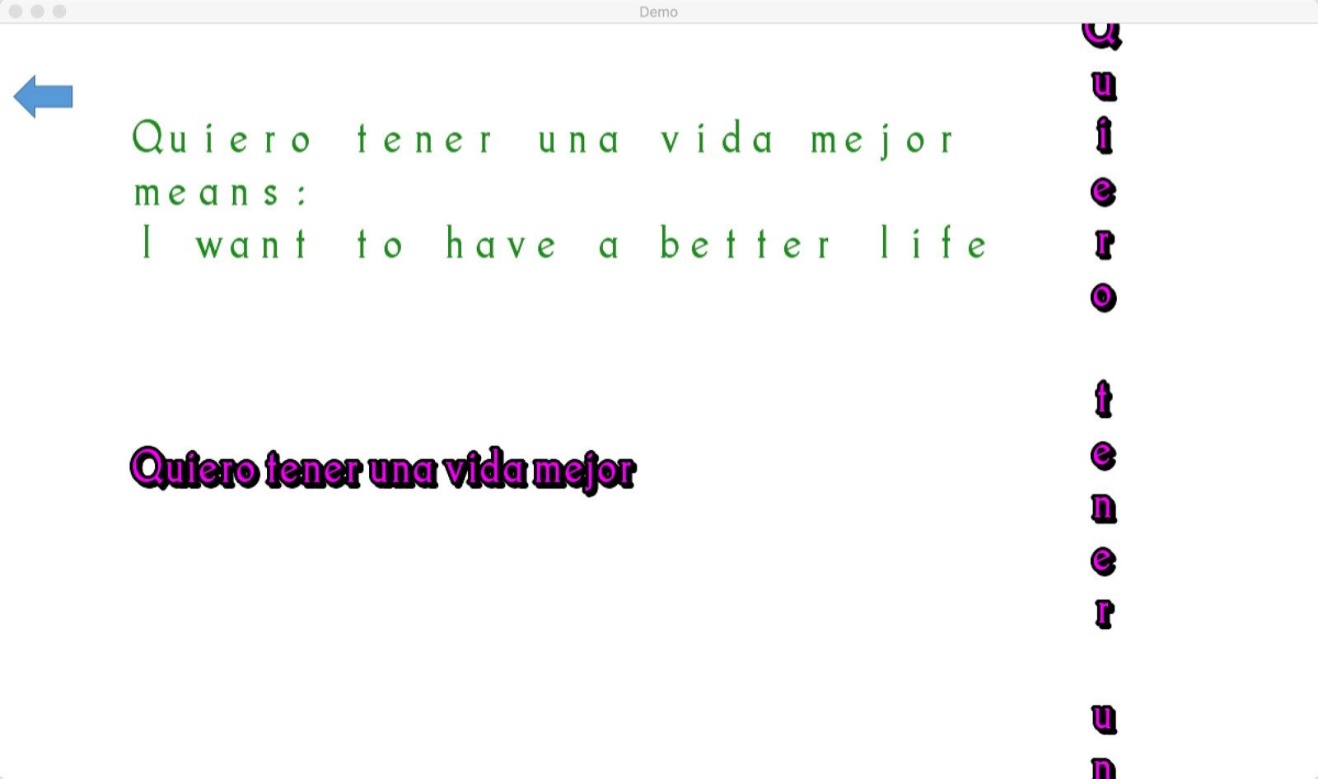
5.Demo



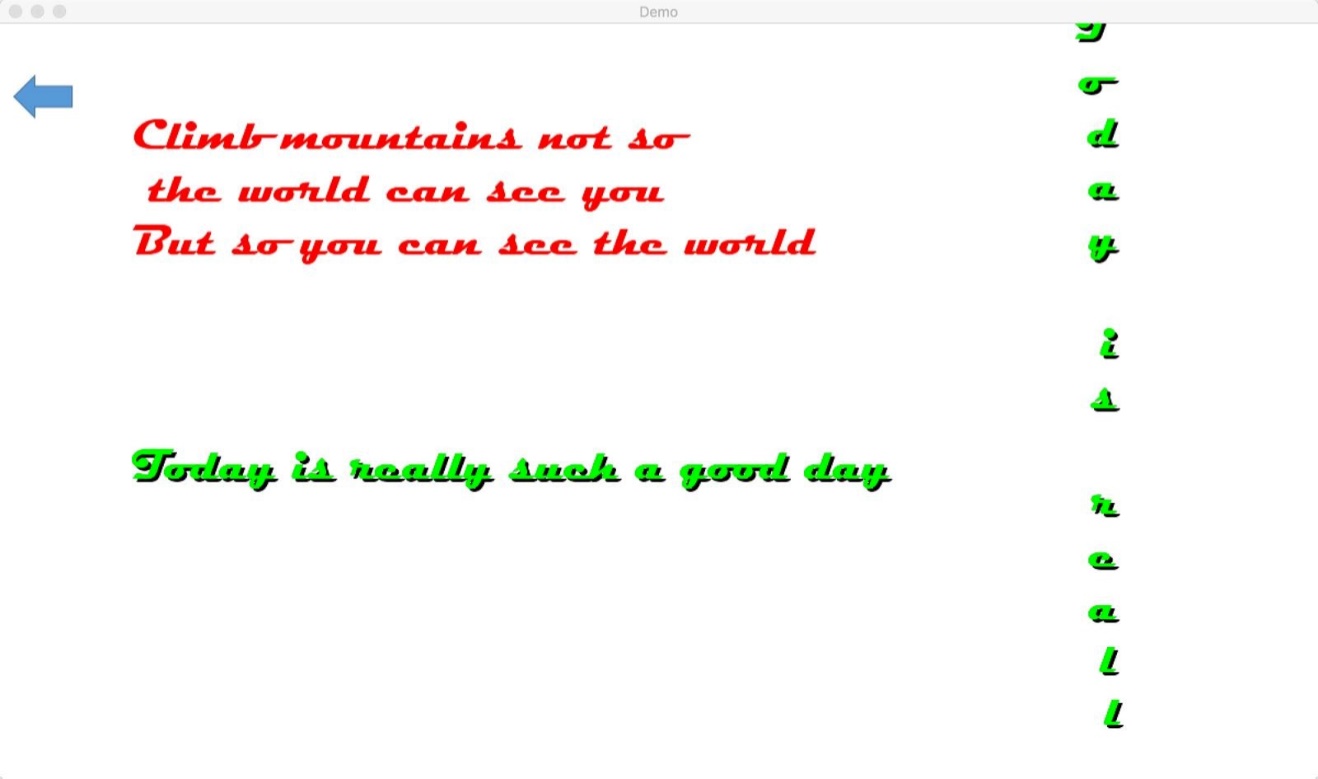
*Image: Start screen*



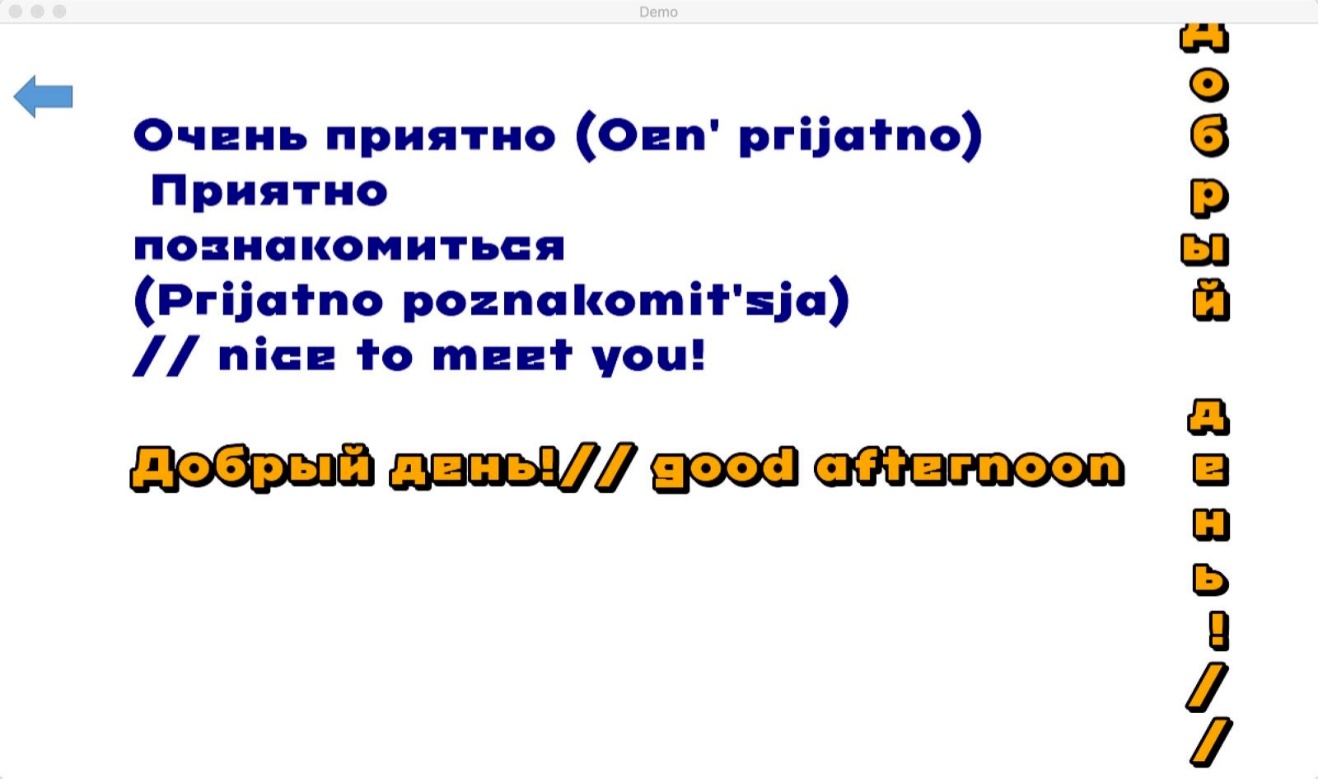
*Image: Main menu screen*



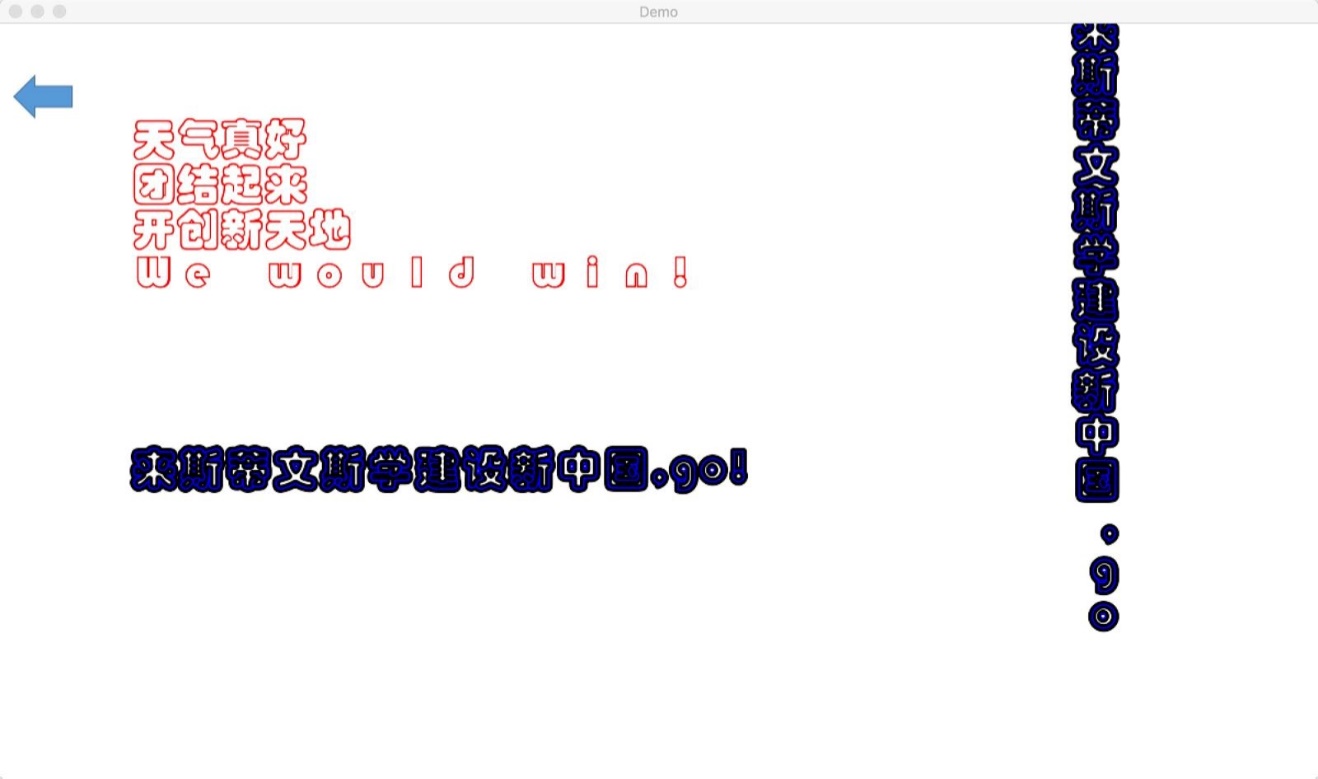
*Image: Spanish language screen*



*Image: English Language Screen*



*Image: Russian language screen*

**

*Image: Chinese language screen*

Future work

* Explore a way that let the user input the text and change it by user input on the screen.
* Combine it into our big project so that everyone in this project can implement this method.
* Find out more different parameters and implement different .ttf files so that it can be more specific and more commercial.

Project link

<https://github.com/StevensECE/CPE810J_2016F_XDF/tree/Text/X_TEXT_Final_DEMO>