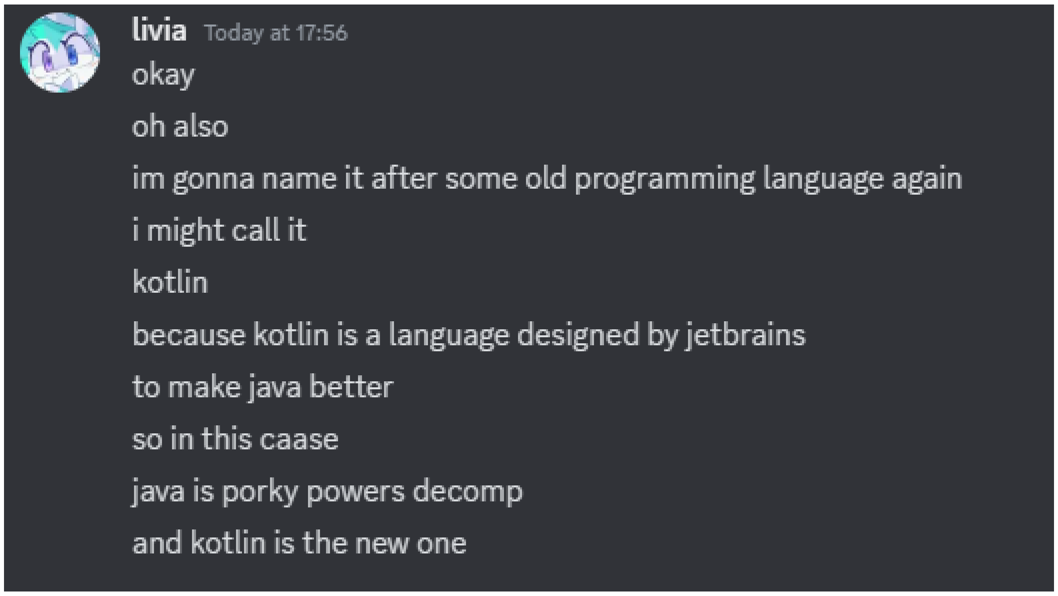
KOTLIN DECOMP

last updated 1.0.0

TABLE OF CONTENTS

A Brief History of Kotlin

Kotlin is PorkyPowers decomp on steroids!... and a decomp aimed at simplifying the Baldi modding process, by packaging your average tutorials into one big decomp, with handy customising features. The decomp was started on the 12th of July 2024 at around 17:50 (BST), and named after the programming language Kotlin, let past me explain:


livia Today at 17:56

okay

oh also

im gonna name it after some old programming language again

i might call it

kotlin

because kotlin is a language designed by jetbrains

to make java better

so in this caase

java is porky powers decomp

and kotlin is the new one

truly groundbreaking stuff

What Does Kotlin Package

Kotlin packages with 2 main assets

- Surge
- NaughtyAttributes
- Newtonsoft.JSON
- PrimeTween

These are all located in *Assets/_KOTLIN/Internal/Dependancies*

Surge is generally used internally for *Singletons*, and
NaughtyAttributes is used for *improving the Inspector*
Newtonsoft.JSON is used for *translation deserialization*
PrimeTween is used for *Plus elevator gates*

What Does Kotlin Change

Code

- Token comments are removed
- Tiny cleanups
- Organises scripts
- Items are structs, see KOTLIN.Items
- Interactions are inheritable, see KOTLIN.Interactions

Optimisation

- The map is converted to Quads
- Interactions are checked once a click, not a billion times
- Optimised Billboard & Pickup Animation scripts, see Contributors

Simplication

- Image text elements are replaced with text (for translations)

Kotlins API

Kotlin has an API to make your life easier. Everything you need is documented below:

KOTLIN.Interactions

This is a class to *inherit* from, the GameController checks for an *Interactable* component on click and fires the *Interact* method, which you should override, for example:

```
public class InteractTest : KOTLIN.Interactions.Interactable
{
    public override void Interact()
    {
        UnityEngine.Debug.Log("wow ive been inteteracted");
    }
}
```

KOTLIN.Subtitles

This namespace handles *Subtitles*, all you need to know in this section is how to create a subtitle.

KOTLIN.Subtitles.SubtitleManager is a singleton, so you'll need SubtitleManager.Instance, then just call the CreateSubtitle method:

Arguments

SubtitleType type - 2D or 3D (or 4D but sets to 3D)

string text - what subtitle say

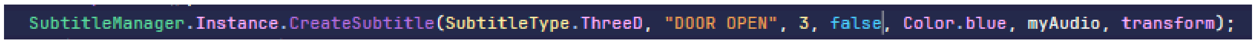
float time - how long on screen for

bool forever - should the subtitle stay on screen forever

Color colour - colour of subtitle text

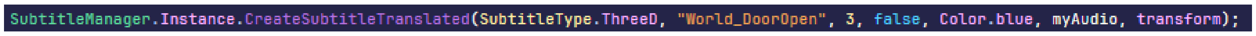
AudioSource audSource - what audio source created subtitle

Transform creator - what gameobject created subtitle



```
SubtitleManager.Instance.CreateSubtitle(SubtitleType.ThreeD, "DOOR OPEN", 3, false, Color.blue, myAudio, transform);
```

To translate a subtitle, use SubtitleManager.Instance.CreateSubtitleTranslated.



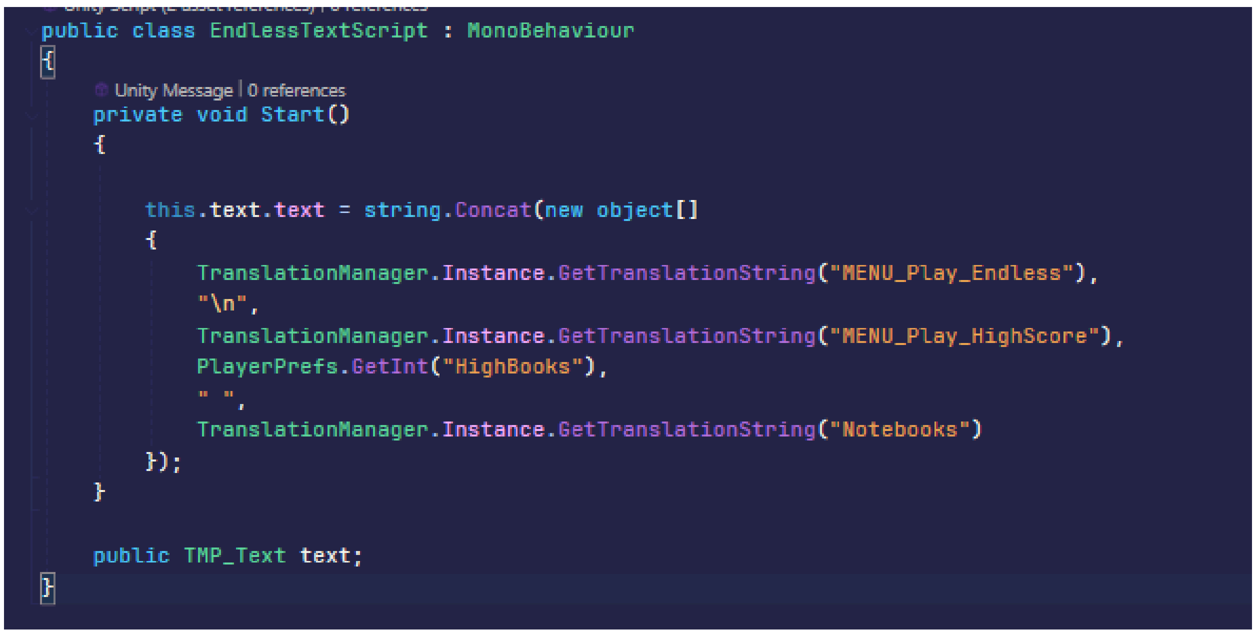
```
SubtitleManager.Instance.CreateSubtitleTranslated(SubtitleType.ThreeD, "World_DoorOpen", 3, false, Color.blue, myAudio, transform);
```

All arguments are the same, except text (argument 2) which should be the translation key, see the [Translation segment](#) for more information on translations

KOTLIN.Translation

This namespace handles *translation*, translations are pretty simple.

To get a translation from a key, simply do TranslationManager.Instance.GetTranslationString("Key")
For example:

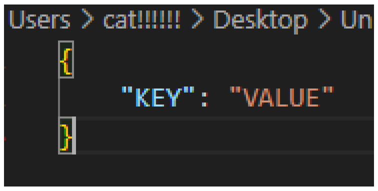


```
public class EndlessTextScript : MonoBehaviour
{
    Unity Message | 0 references
    private void Start()
    {

        this.text.text = string.Concat(new object[]
        {
            TranslationManager.Instance.GetTranslationString("MENU_Play_Endless"),
            "\n",
            TranslationManager.Instance.GetTranslationString("MENU_Play_HighScore"),
            PlayerPrefs.GetInt("HighBooks"),
            " ",
            TranslationManager.Instance.GetTranslationString("Notebooks")
        });
    }

    public TMP_Text text;
}
```

To create a translation, you need to create a JSON file in the StreamingAssets folder, and call it *Subtitles_{ANYTHING}*.json (file extension). Now add all your keys and stuff!1 It's kind of like a dictionary



```
{
    "KEY": "VALUE"
}
```

Value will show up in game

Translating Text

Add a *TranslationObject* to the gameobject and select what type of text yours is

TMPText is TMP text in the *World*

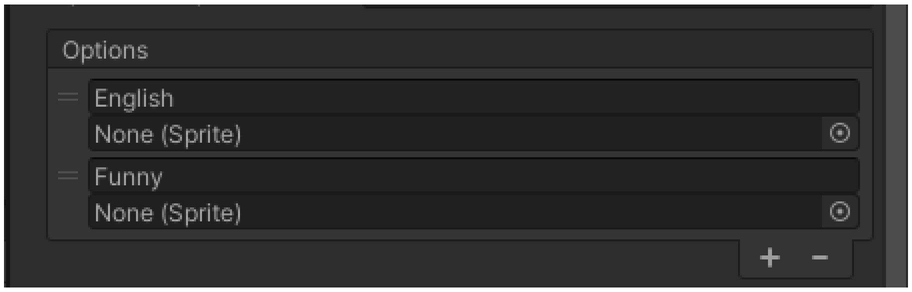
TMPText_UI is TMP text in *UI*

Text is unity's default text component

Then type in the key of the translation.

Adding translations to the options menu

In the *OptionsMenu* of the *MainMenu* scene, go to the *LanguageSelection* GameObject and scroll down to the *Dropdown* component. Find where it says *Options* and add a new entry, call it what you want the player to see.



Options

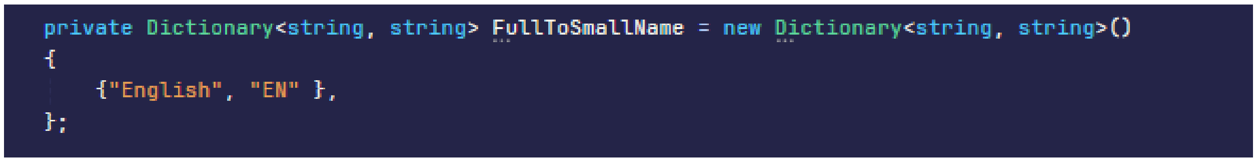
English

None (Sprite)

Funny

None (Sprite)

Now open the *LanguageSelector* script and find the *FullToSmallName* dictionary.



```
private Dictionary<string, string> FullToSmallName = new Dictionary<string, string>()
{
    {"English", "EN" },
};
```

Add a new entry to the dictionary, with the key (first string) being what you inputted in the dropdown, then the value (second string) being what you called the subtitle file identifier (so for Subtitles_EN.json you would put EN)

should work now idk

KOTLIN.Items

Hi :D
Hewwo!!!! :3
This is a twutowial dwoc on hwow to wuse kwotlin uwu
Lets get started OwO

Contributors

- BlueVapor1234 - , Plane 2 Quad Convertor (Edited), Subtitle Position & Scale Calculation
- YuraSuper2048 - Optimized Billboards, Optimized Pickup Animation
- Benefond - Polish Translations