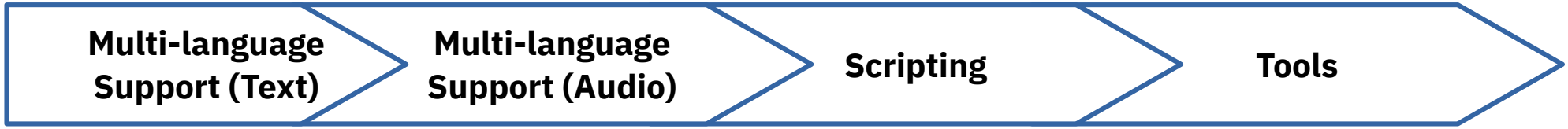
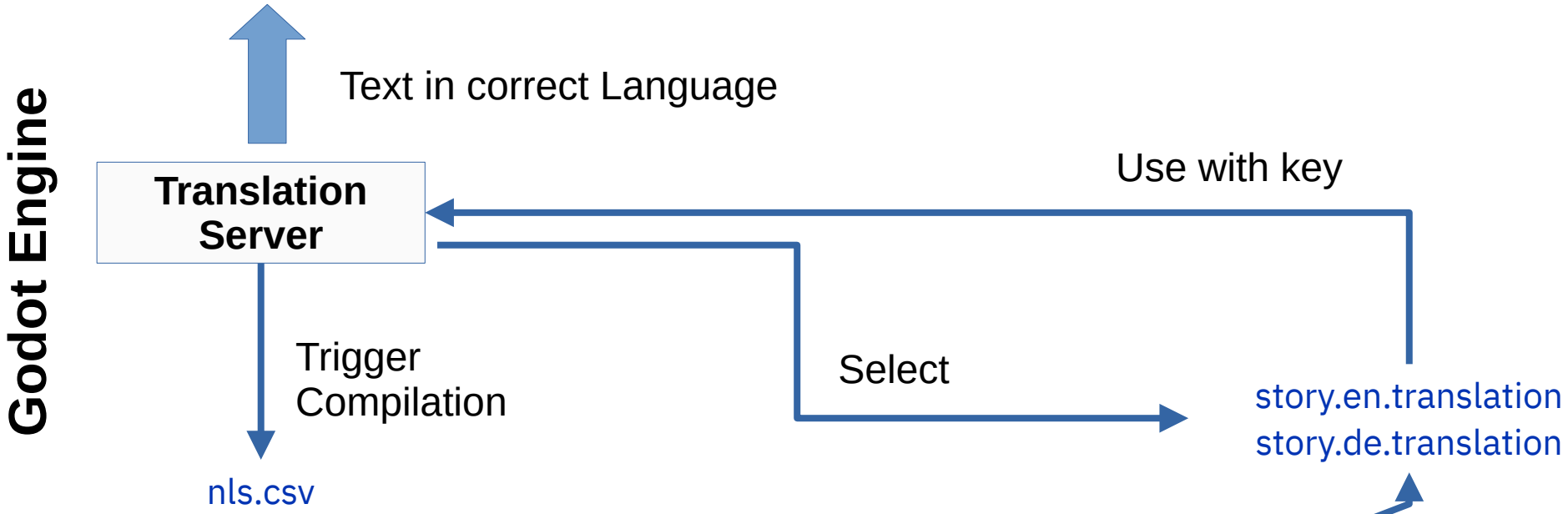
Department of Information Systems
and Operations Management

Creating Games with REXX



Mike Beer (mindcoa.ch) & Till Winkler (IT:U Linz)

Multi-language Support (Text)



key	en	de	...
story_1	Something1 ...	Etwas 1 ...	...
story_2	Something 2 ...	Etwas 2 ...	...

Till Winkler

Translating using DeepL API (1 of 4)

- 1. Get DeepL Key
  - <https://developers.deepl.com/docs/getting-started/managing-api-keys>
- 2. Get APIs.cls
  - <https://gitlab.com/dylwi/oorexx-helpers>
- 3. Create and specify key file
- DeepL API Documentation (2026-04-27)
  - <https://developers.deepl.com/api-reference/translate/request-translation>
    - **Implemented:** Basic text translation, Retrieve usage, Retrieve Languages
    - **Missing:** Translation option (formality, sentence splitting, ...), Document translation, Rephrasing, Glossaries, Detect languages

Multi-language Support (Text)

Translating using DeepL API (2 of 4)

apis.cls

```
::Class Deepl Public
  ::Constant About (.array~of( -
    "Deepl Class, (C) dylwi 2026, https://gitlab.com/dylwi/oorexx-helpers", -
    "", -
    "A class for facilitating the use of Deepl Web API.", -
    "", -
    "Usage:", -
    "  <a deepl>              = <a deepl>~new(<deeplkey-string>)", -
    "  <a string>             = <a deepl>~UNKNOWN (e.g. 2nl, 2de)",-
… cut …
  ::Method Unknown Public
    Expose command
    Use Arg Method, Text
    Lang = Method~substr(2,)
    OriginalText = self~CleanText(Text)
    -- create Json
    Data = .Directory~new
    Data['text']= .array~of(OriginalText)
    Data['target_lang'] = Lang
    Data['source_lang'] = Text[2]
    FinalCommand = command "--header 'Content-Type: application/json' --data '" || .JSON~toJson(data) || "'"
    ReturnDir = self~request(FinalCommand)
  return ReturnDir["text"]
… cut …
  ::Method Request
    Use Strict Arg Command
    --say "#" command
    DeeplArr = .array~new
    ADDRESS System "curl -s -X POST" command with output using (DeeplArr)
    if DeeplArr[1] <> .Nil then do
      json = .json~new
      JsonDir = json~fromJson(DeeplArr[1])
      if DeeplArr[1]~substr(3,12) = "translations" then JsonDir = JsonDir["translations"][1]
      return JsonDir
    end
    else return .directory~new

    ::requires "json.cls"
```

About constant

Unknown method for translation

toJson

FromJson

Json Class

Till Winkler

CC BY SA

Multi-language Support (Text)

Translating using DeepL API (3 of 4)

Usage.rexx

```
#!/usr/bin/rexx

-- Get key
ConfStream = .Stream~new("~/.config/api/deepl-free.conf")
DeeplKey = ConfStream~LineIn(1)
ConfStream~close

-- Init deepl
Deepl = .deepl~new(DeeplKey)

-- Deepl usage
say Deepl~usage



::requires "apis.cls"
```

Output:

```
160 / 500000 [0.0320%]
```

lang.rexx

```
#!/usr/bin/rexx

-- Get key
ConfStream = .Stream~new("~/.config/api/deepl-free.conf")
DeeplKey = ConfStream~LineIn(1)
ConfStream~close

-- Init deepl
Deepl = .deepl~new(DeeplKey)

-- Available languages
Lang = deepL~SupportedDir
say Lang~allitems
say Lang~allindexes


::requires "apis.cls"
```

Output:

```
Galician
Icelandic
Norwegian
Italian
… cut…
GL
IS
NB
IT
```

Translating using DeepL API (4 of 4)

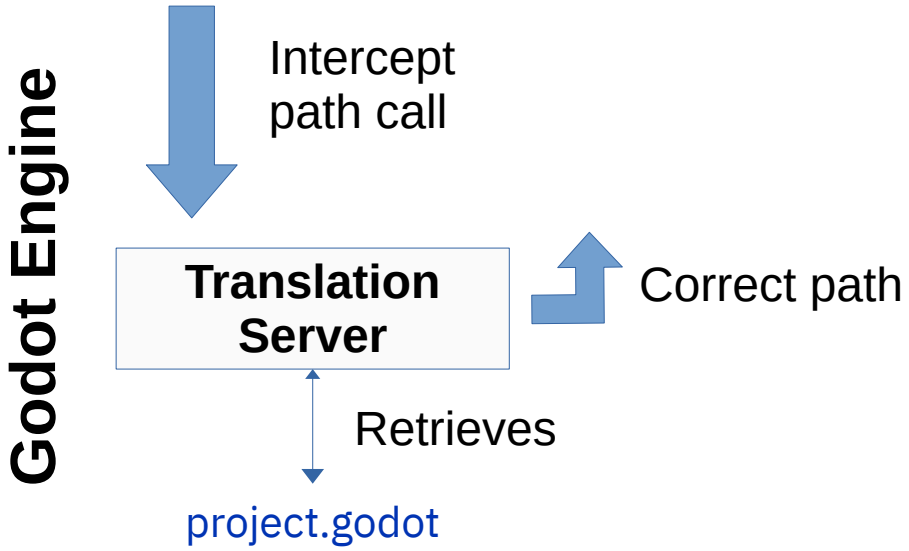
Translate.rexx

```
#!/usr/bin/rexx

-- Get key
ConfStream = .Stream~new("~/.config/api/deepl-free.conf")
DeeplKey = ConfStream~LineIn(1)
ConfStream~close

-- Init deepl
Deepl = .deepl~new(DeeplKey)

-- Deepl usage
say Deepl~usage

-- Deepl translate
say "---"
say Deepl~2es("Hello, Barcelona! How are you all doing?")
say Deepl~2fr("Hello, Barcelona! How are you all doing?")
say Deepl~2nl("Hello, Barcelona! How are you all doing?")
say Deepl~2de("Hello, Barcelona! How are you all doing?")
say "---"

-- Deepl usage
say Deepl~usage


::requires "apis.cls"
```

Output:

```
760 / 500000 [0.1520% used]
---
¡Hola, Barcelona! ¿Cómo estáis?
Bonjour, Barcelone ! Comment allez-vous tous ?
Hallo, Barcelona! Hoe gaat het met jullie?
Hallo, Barcelona! Wie geht es euch allen?
---
920 / 500000 [0.1840% used]
```

Multi-language Support (Audio)




```
…
locale/translation_remaps={
  "res://…/settings_voice_example.mp3": PackedStringArray(
    "res://…/settings_voice_example.mp3:en",
    "res://…/settings_voice_example_ar.mp3:ar",
    ...
  )
}
```


Till Winkler

Audio using ElevenLabs API (1 of 3)

- 1. Get ElevenLabs Key
  - <https://elevenlabs.io/app/developers/api-keys>
- 2. Get APIs.cls
  - <https://gitlab.com/dylwi/oorexx-helpers>
- 3. Create and specify key file
- ElvenLabs API Documentation (2026-04-27)
  - <https://elevenlabs.io/docs/api-reference/introduction>
    - **Implemented:** Text2Speech, Voices, Text2Dialogue
    - **Missing:** Speech2Text, Music, Voice changer & design, Sound Effects, Audio Isolation, Dubbing, ...

Audio using ElevenLabs API (2 of 3)

elevenlabs.rexx

```
#!/usr/bin/rexx

ConfStream = .Stream~new("~/.config/api/elevenlabs.conf")
ElevenLabsKey = ConfStream~LineIn(1)
ConfStream~close

-- Init elvenlabes
ElevenLabs = .ElevenLabs~new(ElevenLabsKey)

-- Available voices
Voices = ElevenLabs~GetVoices
say Voices~allindexes
say "--"
say Voices~allitems

::requires "apis.cls"
```

Output:

```
CALLUM
GIDEON
THEO
LAURA
GEORGE
SARAH
ROGER
BELLA
TESSA
CHARLIE
--
N2lVS1w4EtoT3dr4eOWO
Xq2dbIWNPChFB77imiDe
UmQN7jS1Ee8B1czsUtQh
FGY2WhTYpPnrIDTdsKH5
JBFqnCBsd6RMkjVDRZzb
EXAVITQu4vr4xnSDxMaL
CwhRBWXzGAHq8TQ4Fs17
hpp4J3VqNfWAUOO0d1Us
USEQXnsXRJlw2k9LUzG4
IKne3meq5aSn9XLyUdCD
```

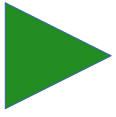
Audio using ElevenLabs API (3 of 3)

elevenlabs.rexx

```
#!/usr/bin/rexx

ConfStream = .Stream~new("~/.config/api/elevenlabs.conf")
ElevenLabsKey = ConfStream~LineIn(1)
ConfStream~close

-- Init elvenlabes
ElevenLabs = .ElevenLabs~new(ElevenLabsKey)

-- Generate files
ElevenLabs~Text2Speech("Programming with Rexx is fun!","RexxIsFun.mp3", "Laura", "en")
ElevenLabs~Text2Speech("Programming with Rexx is fun!",, "Rogger", "en")


::requires "apis.cls"
```

Output: RexxIsFun.mp3

Output: elevenlabs-2026-05-04T12:22:14.mp3



Till Winkler