Angel Carrillo
Luis Cintrón
Fernando Guzman
Jean Carlos Rodríguez
CIIC 4030 / ICOM 4036
Professor: Wilson Rivera
December 4, 2018

PuppetScript
Final Report

Introduction

Unity is a cross-platform video game engine developed by Unity Technologies. As of 2018, the engine has been extended to support 27 platforms. The engine can be used to create both three-dimensional and two-dimensional games as well as simulations for its many platforms. Unity offers a primary scripting API in C#, for both the Unity editor in the form of plugins, and games themselves, as well as drag and drop functionality. The popularity of this engine and its features make it generally easy to use for those new to game development, but the first hurdle that some starting developers must overcome while using Unity is understanding C# and how scripting works with Unity. Our goal with *PuppetScript* is to make player movement scripts, probably the most important aspect to a game, for playable characters much easier.

Language tutorial

How to use:

1. Open the script file
2. Enter the type of movement script for Unity
3. Enter the inputs for directions in X, Y and Z
4. Add a type of action (Jump, Sprint, Walk, Jetpack) with its in-game input
5. Run Script.py to generate the C# file
6. In Unity create the game object that you wish to use and add either a rigidbody or character controller component (the one that matches your script)
7. Then drag and drop that generated script into the object inspector

Language reference manual

From Github repository:
Script.py - for executing the code
Premade script (script.txt)

Requirements

1. *PuppetScript* - https://github.com/jeanrodriguez27/PuppetScript
2. Unity
3. Python 3
4. Ply

Code

Note that the following is further detailed in our Reference Manual:
https://github.com/jeanrodriguez27/PuppetScript/wiki/Reference-Manual

Scripts (type of scripts that are supported):

- RIGIDBODY
- CHARACTERCONTROLLER
- SIMPLE

Speed: float variable. The movement speed of the player in the game.

Gravity: float variable. The gravity applied to the player's body in the game.

Movements (The Inputs for the player movement):

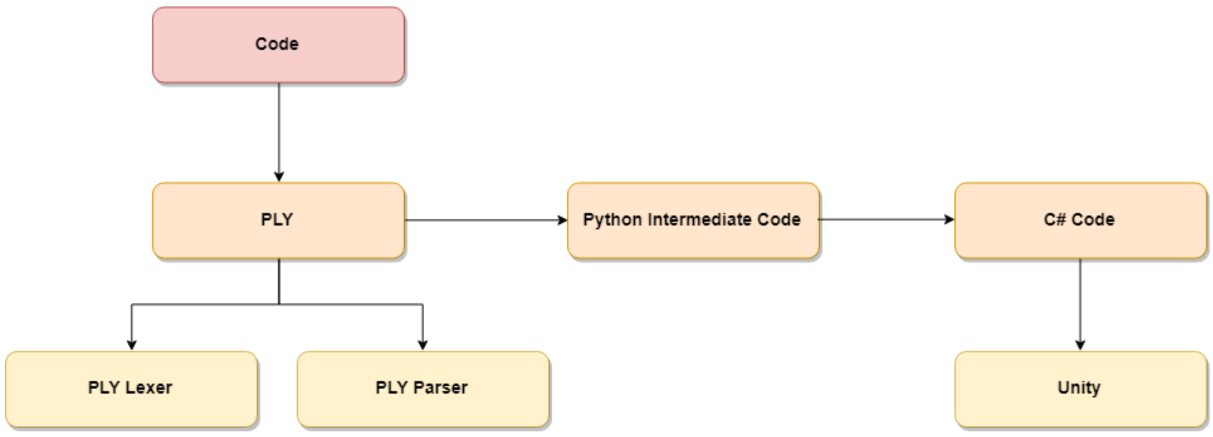
- Horizontal
- Vertical
- None

Actions (Actions that are supported):

- Jump
- Sprint
- Walk
- Jetpack

Language development

Translator architecture



Interfaces between modules

Our language has 3 main modules: *CodeLexer.py*, *CodeParser.py* and *CodeGenerator.py*. The user writes a script and runs the *Script.py* to translate it, this is then passed through the parser and lexer where code is parsed and checked for errors. Finally, it passes to the *CodeGenerator.py* which translates the script into C# and then generates a C# script which can be used in Unity.

Software Development Environment

- *PyCharm* - an IDE used for programming in Python.
- *PLY* - Parser implementation in Python. Implementation of lex and yacc parsing tools for python.
- *Unity* - A cross-platform video game engine, works on all major platforms from game consoles, computers and mobile devices. Supports 2D and 3D games and uses C# as its primary scripting API.
- *Github* - Git repository hosting service. Used to collaborate with team members

Test methodology

The first stage of testing *PuppetScript* was testing the parser and lexer. Once that was done the next phase was testing our code translation and the generating of the output file. To test that all of our scripts were working we used one of the sample projects in Unity.

Scripts used for testing

```
CHARACTERCONTROLLER
Speed = 10.0
Gravity = 0.1
movex = Horizontal
movey = NONE
movez = Vertical
JUMP = Key_Space
```

```
SIMPLE
Speed = 10.0
movex = Horizontal
movey = NONE
movez = Vertical
```

Conclusions

While working on *PuppetScript* we learned a lot from understanding more from how the scripting API works in Unity to how much effort is involved in implementing one's own programming language. One of our key observations from this project is that building a programming language involves a lot of trial and error. Another key observation is that in order to build the best possible programming language intended to greatly simplify an existing process, one must be extremely informed regarding the current methods of performing said process and how people who use the language. Due to the fact that three-dimensional player movement can be customized in such an array of ways, we are sure that our language can benefit from being expanded in the future. In the end we can say that *PuppetScript* makes implementing basic character movement scripts in Unity much easier and that knowledge of either Unity's scripting API or C# is only optional rather than necessary when using it.

Main GitHub link:
https://github.com/jeanrodriguez27/PuppetScript

Website link:
https://jeanrodriguez27.github.io/PuppetScript/