Authors: Daniel Pérez Pérez, Dolimar Roldán Vélez, Raul Ortiz Rivera
CIIC 4030/ICOM 4036
Prof. Wilson Rivera
September 9, 2021

Introduction:

As technology advances so do the programming languages that run them. Even though most of our devices rely on programming, it is estimated that only 0.5% of the world population knows how to code. Currently, Unity is one of the most popular platforms for creating 2D and 3D applications using .NET and the C# programming language. Due to its features, Unity is generally easy to use for those new to game development, but it still requires them to learn C# and scripting. For this reason, we have decided to develop a new programming language that allows users to create complex 3D environments with simple commands. The idea is to simplify the creation of 3D environments in Unity by creating an easy-to-use programming language that adds objects such as cars, people, houses, and more to develop intricate environments without any previous programming experience.

Language Features:

Our programming language will feature Object Oriented Programming:

- Reusability of code through inheritance
- Flexibility through polymorphism
- Superclass CityGenerator
- Subclass addCar, addHouse, addTrashCan and addPeople

Example of Program:

In the codes described below you can see the amount of extra code that has to be written in the C# language vs. CityGenerator language to get an interactable car.

CityGenerator Programming Language:

```
# The execution of this functions will generate a object in the Unity environment.
# When the user adds the desired location using the x, y and z
# coordinates, the object will auto-generate on the map.
# You will also be able to interact with the objects of the environment.
mainCityObjects():
    addCar = (posX, posY, posZ, color, style)
    addHouse = (posX, PosY, PosZ, Color, style)
    addTrashCan = (posX, PosY, PosZ)
    addPeople = (posX, PosY, PosZ, style)
```

C# Interactable Car Example:

```
using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class CarController : MonoBehaviour
{
    private const string HORIZONTAL = "Horizontal";
    private const string VERTICAL = "Vertical";

    private float horizontalInput;
    private float verticalInput;
    private float currentSteerAngle;
    private float currentbreakForce;
    private bool isBreaking;

    [SerializeField] private float motorForce;
    [SerializeField] private float breakForce;
    [SerializeField] private float maxSteerAngle;

    [SerializeField] private WheelCollider frontLeftWheelCollider;
    [SerializeField] private WheelCollider frontRightWheelCollider;
    [SerializeField] private WheelCollider rearLeftWheelCollider;
    [SerializeField] private WheelCollider rearRightWheelCollider;

    [SerializeField] private Transform frontLeftWheelTransform;
    [SerializeField] private Transform frontRightWheeTransform;
    [SerializeField] private Transform rearLeftWheelTransform;
    [SerializeField] private Transform rearRightWheelTransform;

    private void FixedUpdate()
    {
        GetInput();
        HandleMotor();
        HandleSteering();
        UpdateWheels();
    }


    private void GetInput()
    {
        horizontalInput = Input.GetAxis(HORIZONTAL);
        verticalInput = Input.GetAxis(VERTICAL);
        isBreaking = Input.GetKey(KeyCode.Space);
    }

    private void HandleMotor()
    {
        frontLeftWheelCollider.motorTorque = verticalInput * motorForce;
        frontRightWheelCollider.motorTorque = verticalInput * motorForce;
        currentbreakForce = isBreaking ? breakForce : 0f;
        ApplyBreaking();
    }

    private void ApplyBreaking()
    {
        frontRightWheelCollider.brakeTorque = currentbreakForce;
        frontLeftWheelCollider.brakeTorque = currentbreakForce;
        rearLeftWheelCollider.brakeTorque = currentbreakForce;
        rearRightWheelCollider.brakeTorque = currentbreakForce;
    }

    private void HandleSteering()
    {
        currentSteerAngle = maxSteerAngle * horizontalInput;
        frontLeftWheelCollider.steerAngle = currentSteerAngle;
        frontRightWheelCollider.steerAngle = currentSteerAngle;
    }

    private void UpdateWheels()
    {
        UpdateSingleWheel(frontLeftWheelCollider, frontLeftWheelTransform);
        UpdateSingleWheel(frontRightWheelCollider, frontRightWheeTransform);
        UpdateSingleWheel(rearRightWheelCollider, rearRightWheelTransform);
        UpdateSingleWheel(rearLeftWheelCollider, rearLeftWheelTransform);
    }

    private void UpdateSingleWheel(WheelCollider wheelCollider, Transform wheelTransform)
    {
        Vector3 pos;
        Quaternion rot
;       wheelCollider.GetWorldPose(out pos, out rot);
        wheelTransform.rotation = rot;
        wheelTransform.position = pos;
    }
}
```

Implementation Requirement and Tools:

For the development of this programming language several tools will be used: Visual Studio Code (VSC), PyCharm, Python Lex-Yacc (PLY), Unity, and GitHub. VSC and PyCharm are both integrated development environments (IDEs) that will be used to code in Python. Additionally, the PLY tool will be used because it allows for the implementation of parsing tools such as lex and yacc in Python. Unity will also be used because of its popularity for creating 2D and 3D applications. Lastly, GitHub which is an online git repository will be used to collaborate with team members.

Project Timeline:

Contributors:

- Daniel J. Pérez Pérez
- Raúl A. Ortiz Rivera
- Dolimar Roldan Velez

Task No.	Project Timeline
1	Project Proposal (August 23 – September 10)
2	Introduction (August 23 – 27)
3	Languages Features (August 23 – September 10)
4	Example of a program (August 23 – September 10)
5	Implementation requirements and tools (August 30 – September 10)
6	Reference (September 6 – September 10)

7	Project Implementations (September 20 – November 12)
8	Lexical analyzer (scanner) (September 20 – October 15)
9	Syntax analyzer (parser) (September 20 – October 15)
10	Intermediate Code (October 18 – November 12)
11	Github page and repository (November 1 – November 12)
12	Project Demonstration (November 15 – November 30)
13	Introduction and motivation (November 15 – November 19)
14	Description of language features (November 22 – November 26)
15	Link code repository (November 29 – November 30)
16	Short video describing the project (November 22 – November 30)

References:

U. Asset Store, “Arcade: Free racing car: 3d land: Unity asset store,” *3D Land | Unity Asset Store*, 28-Jan-2020. [Online]. Available: https://assetstore.unity.com/packages/3d/vehicles/land/arcade-free-racing-car-161085. [Accessed: 09-Sep-2021].

U. Asset Store, “VILLAGE houses PACK: 3D Characters: Unity asset store,” *3D Characters | Unity Asset Store*, 07-Jun-2016. [Online]. Available:

https://assetstore.unity.com/packages/3d/characters/village-houses-pack-63695. [Accessed: 09-Sep-2021].

U. Store, “Town houses PACK: 3D urban: Unity asset store,” *3D Urban | Unity Asset Store*, 07-Apr-2016. [Online]. Available: https://assetstore.unity.com/packages/3d/environments/urban/town-houses-pack-42717. [Accessed: 09-Sep-2021].

U. Store, “Desert village (Houses) LowPoly: 3D ENVIRONMENTS: Unity asset store,” *3D Environments | Unity Asset Store*, 21-Jul-2021. [Online]. Available: https://assetstore.unity.com/packages/3d/environments/desert-village-houses-lowpoly-200247. [Accessed: 09-Sep-2021].

U. Store, “Vegetation spawner: TERRAIN: Unity asset store,” *Terrain | Unity Asset Store*, 25-Aug-2020. [Online]. Available: https://assetstore.unity.com/packages/tools/terrain/vegetation-spawner-177192. [Accessed: 09-Sep-2021].

U. Store, “Trash can: 3d Exterior: Unity asset store,” *3D Exterior | Unity Asset Store*, 09-Oct-2014. [Online]. Available: https://assetstore.unity.com/packages/3d/props/exterior/trash-can-23183. [Accessed: 09-Sep-2021].

U. Store, “Distant lands Free Characters: 3D Characters: Unity asset store,” *3D Characters | Unity Asset Store*, 03-Sep-2020. [Online]. Available: https://assetstore.unity.com/packages/3d/characters/distant-lands-free-characters-178123. [Accessed: 09-Sep-2021].

InfoQ. 2021. *IDC Study: How Many Software Developers Are Out There?*. [online] Available at: <https://www.infoq.com/news/2014/01/IDC-software-developers/> [Accessed 9 September 2021].

Y. Mulonda, “Object-Oriented programming Concepts ‘In SIMPLE ENGLISH,’” *Medium*, 27-May-2020. [Online]. Available: https://yannmjl.medium.com/object-oriented-programming-concepts-in-simple-english-3db22065d7d0. [Accessed: 09-Sep-2021].