Challenge Based Code Training Platform Using Unity Games

Abstract

Content

Introduction

What is Unity?

1.Technical Background

1.1 Networking and Protocols

1.1.1 Addresses

1.1.2 Client and Server

1.2.3 TCP Sockets

1.2.4 UDP Sockets

1.2.5

1.2 Unity game development

1.2.1 Why Unity?

1.2.2 Unity Editor Window

1.2.3 Unity Game Objects

1.2.4 Unity Build in Components

1.2.5

1.3

2.Software and Implementation

2.1 Software used

2.2 Application Requirements

2.2.1 Python Client Requirements

2.2.2 Python Server Requirements

2.2.3 Unity Client Requirements

2.3 Program Flow

2.4 Flow Diagram

2.5 Python Client Implementation

2.6 Python Server Implementation

2.7 Unity Client Implementation

3.Results and discussions

Conclusion and future work

Bibliography

Introduction (needs improvement)

Nowadays, the education improved a lot. People are tired of learning something if they do not find it interesting enough or if they do not get sufficient rewards while doing it.

Many students simply rush their studies and homework in order to get rid of them and to have more time for doing what they like. So are the programmers, if they do not like what they are coding, the will never bother coding it.

The purpose of this Bachelor thesis is to combine the utility with the pleasure, learning to code while playing games. This is achieved through a Unity Application that allows the user to play a game by coding the strategy. The code written by the player gets executed and the output of the code is seen on the game.

What is Unity ?



Figure (Unity logo)

Unity3D is a very powerful development framework for building the exciting 2D/3D game. The games developed on this game engine are developed once with a single code but the games can be easily deployed on different platforms.

Unity3D is a game engine that offers revolutionary solutions with features of interactive media installation as well as a stunning architectural visualization that is suitable for cross platform environment. Unity 3D game development engine offers extensive ecosystem for building and deploying.

It ensures rare gaming experience with the enthralling 3D game flow along with a wide array of game authoring tools and intuitive resources. The popularity of the game engine has encompassed for its user-friendly features and other simple and pragmatic reasons.

Unity3D is reaching the new heights in developing games and it has been ruling the world of game development. Unity3D has become the first choice for the independent developers. Game development studios even have highly admired utilization of Unity3D for its versatility, flexibility, and ease-to-access feature.

Technical Background

* 1. Networking and Protocols

A computer network consists of two or more computing devices that are connected in order to share the components of your network (its resources) and the information you store there, as shown in Figure 1

Computer 2

Computer 1

Laptop

Wireless

Figure 1 (Computer network)

The most basic computer network (which consists of just two connected computers) can expand and become more usable when additional computers join and add their resources to those being shared.

The first computer, yours, is commonly referred to as your local computer. It is more likely to be used as a location where you do work, a workstation, than as a storage or controlling location, a server. As more and more

computers are connected to a network and share their resources, the network becomes a more powerful tool, because employees using a network with more information and more capability are able to accomplish more through those added computers or additional resources.

The real power of networking computers becomes apparent if you envision your own network growing and then connecting it with other distinct

networks, enabling communication and resource sharing across both networks. That is, one network can be connected to another network and become a more powerful tool because of the greater resources.

1.1.1 Addresses

A network address is any logical or physical address that uniquely distinguishes a network node or device over a computer or telecommunications network. It is a numeric/symbolic number or address that is assigned to any device that seeks access to or is part of a network.

A network address is a key networking technology component that facilitates identifying a network node/device and reaching a device over a network. It has several forms, including the Internet Protocol (IP) address, media access control (MAC) address and host address.

Computers on a network use a network address to identify, locate and address other computers. Besides individual devices, a network address is typically unique for each interface; for example, a computer's Wi-Fi and local area network (LAN) card has separate network addresses.

A network address is also known as the numerical network part of an IP address. This is used to distinguish a network that has its own hosts and addresses. For example, in the IP address 192.168.1.0, the network address is 192.168.1.

1.1.2 Client and Server

Client–server model is a distributed application structure that partitions tasks or workloads between the providers of a resource or service, called servers, and service requesters, called clients. Often clients and servers communicate over a computer network on separate hardware, but both client and server may reside in the same system. A server host runs one or more server programs, which share their resources with clients. A client does not share any of its resources, but it requests content or service from a server. Clients, therefore, initiate communication sessions with servers, which await incoming requests. Examples of computer applications that use the client-server model are Email, network printing, and the World Wide Web.

The client-server characteristic describes the relationship of cooperating programs in an application. The server component provides a function or service to one or many clients, which initiate requests for such services. Servers are classified by the services they provide. For example, a web server serves web pages and a file server serves computer files. A shared resource may be any of the server computer's software and electronic components, from programs and data to processors and storage devices. The sharing of resources of a server constitutes a service.

Web Server Web Browser Client

Page request

www.google.com/images

Data

Returned page

Figure 2 (Server request in a web browser)

Whether a computer is a client, a server, or both, is determined by the nature of the application that requires the service functions. For example, a single computer can run a web servers and file server software at the same time to serve different data to clients making different kinds of requests. Client software can also communicate with server software within the same computer.

In general, a service is an abstraction of computer resources and a client does not have to be concerned with how the server performs while fulfilling the request and delivering the response. The client only has to understand the response based on the well-known application protocol, i.e. the content and the formatting of the data for the requested service. Clients and servers exchange messages in a request–response messaging pattern. The client sends a request, and the server returns a response.

A server may receive requests from many distinct clients in a short period of time. A computer can only perform a limited number of tasks at any moment, and relies on a scheduling system to prioritize incoming requests from clients to accommodate them.

1.2 Unity Game Development

1.2.1 Why Unity ?

There are really very few options when it comes to Game development. The three main choices if you want to build games are Unreal, Unity or GameMaker.

Unity gives you a very raw product out of the box, but is highly flexible, well-documented, and highly extensible to build pretty much any genre of game you can think of.

There are plenty of highly successful games such as Escape from Tarkov (FPS), Monument Valley (Puzzler), and This War of Mine (Strategy / Survival) all built in Unity.

In reality the engine you build your first game on is probably not critical, so my advice is just to pick one and go with it.

Benefits:

Unity3D is easy to operate.

The structure of animation and rendering is highly flexible that turns several characters in the game lively.

Unity3D renders natural and fluid movement.

It supports cross platforms as well as various devices.

JavaScript and C are the most popular coding languages which can be used with Unity for scripting.

Unity3D is the preferred game engine for developing multiplayer games.

The user-friendly interface of Unity3D makes it easy-to-handle.

Unity3D is compatible with Windows as well as Mac OS X which is one of the reasons why the developers go for Unity3D.

With Unity3D, it is easier to handle and execute different elements by using the technique of “Drag, Drop, and Animate”.

With Unity3D, games can be developed for around twelve different platforms.

1.2.2 Unity Editor Window

The editor window is split up into a couple of sections. We will cover this very briefly as we will refer to it constantly throughout the article. If your familiar with this already just skip past!

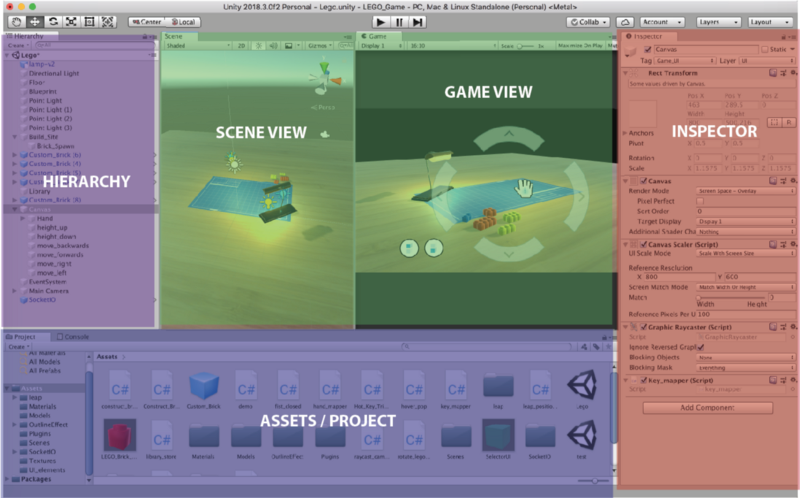


Figure (Unity Editor Window)

Scene View: Allows placement and movement of GameObjects in the Scene

Game View: Previews how the player will see the scene from the camera

Inspector: Provide details on the selected GameObject in the scene.

Assets / Project: All prefabs, textures, models, scripts etc are stored here

Hierarchy: Enables nesting and structuring of GameObjects within the scene

1.2.3 Unity Game Objects

GameObjects are the core building block of everything in the Unity games engine. The name almost gives it away:

Anything you place within a scene in Unity must be wrapped in a ‘game object.’

If you’ve got a web design background, you can think of GameObjects as being a lot like <div> elements! Extremely boring containers, but are highly extensible to create complex functionality or visuals. Below is a list of “primitive” GameObjects that Unity offers.

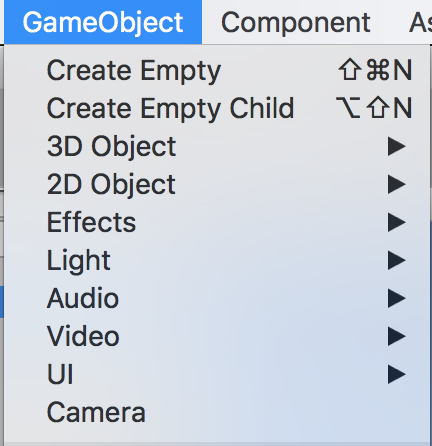


Figure (Unity Standard GameObjects)

2. Software and Implementation

2.1 Software used

This project was developed using Unity Engine for rendering the game, all the scripts being written in C#, Python for the Server communications and Client interface and a Unity Asset for the game models.

2.2 Application Requirements

2.2.1 Python Client Requirements

The Python Client is an interface that connects to the python server for sending and receiving information. The Client requires an IP address and a Port to connect to, followed by a token which is unique and required to connect to the Python Server. As the game runs on rounds, this Client is responsible for receiving the data of the game, processing it and sending back to the server an

action for the player in the due time, otherwise the server will disconnect this client.

2.2.2 Python Server Requirements

The Python Server is the main brain of the game. It creates the sockets available for the players in order to allow them to connect to the server, handles the possible bad scenarios, ensures the normal flow of the data through the sockets, checks if the Unity Clients have all connected and are ready to play, works as a lobby manager, checks if all the sockets are still connected, otherwise safely removes them, logs user friendly messages, generates the game map, combines all the input received from the Python Clients in order to send it to each Unity Client, splits the received from the Unity Clients to send it to the Python Clients in order for the code to decide what to do next.

2.2.3 Unity Client Requirements

The Unity Client works as a visual output for the processed data. It’s role is to ensure that the input received from the Python Server is processed and displayed on the screen. It also contains an UI interface for connectivity to server.

2.3 Program Flow

The user starts by executing the Unity Client. In the Unity Client the user chooses one either to play single player (against a hardcoded AI) or multiplayer (against real players in LAN or Internet). If single player is chosen, Unity Client will create a local unity server which will handle the input from the Python Client. The game on the single player is played in 2, the user and AI. If the multiplayer is chosen, Unity Client will create a new process of Python Server which will manage the clients. Now, the Unity Clients will be able to connect to the server using the IP and PORT of the server. After the server is filled up with Unity Clients or all the Unity Clients press the ready button, each Unity Client will get a token which will be used by the Python Client to connect to the Python Server. The server will then wait for all the Python Clients then start generating a random map on which users will play. The Python Server will then ensure the flow of the data unity client -> python server -> python client and backwards, each round. In case of any problem with the player or the data received, the player will be disconnected from the game and the game will continue.

2.4 Flow Diagram

Rendering

Rendering

Rendering

Rendering

Process Data

Process Data

Process Data

Process Data

Unity Client 1

Unity Client 2

Unity Client 3

Unity Client 4

Handle New Connections

Server

Thread

Handle Disconnected Players

Process Data

Manage Game

Python Client 4

Python Client 3

Python Client 2

Python Client 1

Process Input

Process Input

Process Input

Process Input

Bibliography

**Bibliographic resources:**

**Articles:**

Socket Programming in Python (Guide) - Nathan Jennings, Source : <https://realpython.com/python-sockets/>

The Ultimate Beginners Guide To Game Development In Unity -

<https://www.freecodecamp.org/news/the-ultimate-beginners-guide-to-game-development-in-unity-f9bfe972c2b5/>

<https://medium.com/@vivekshah.P/reasons-why-unity3d-is-so-much-popular-in-the-gaming-industry-705898a2a04>

<https://en.wikipedia.org/wiki/Client–server_model>

https://www.techopedia.com/definition/20969/network-address

**Books:**

C# Game Programming Cookbook for Unity 3D - Jeff W. Murray, 2014

TCP/IP Sockets in C# Practical Guide For Programmers - David Makofske, Michael J. Donahoo, Kenneth L. Calvert