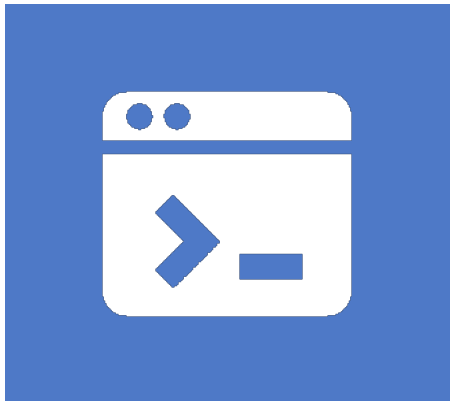
Advanced Game Programming



Week 9

Homework Review

Object Pooling

Web Servers

Refresher from Simple Networking Lecture

- Socket -> IP Address and a Port Number
  - 172.217.10.46:80
- Server: Centralized computer designed to send and receive information from clients
- Packet: Information / data that’s traveling on a network
- Dedicated Server
  - Centralized server all clients connect to
  - More powerful hardware/higher bandwidth connection, clients don’t see each other addresses, no player advantage
  - Expensive, need multiple server locations

Don’t Block Main Thread!!!

```
private string data;
private IEnumerator GetData(string URL, int port)
{
    var www = UnityWebRequest.Get(URL + ":" + port);
    yield return www.SendWebRequest();

    if(www.isNetworkError || www.isHttpError) {
        Debug.Log(www.error);
        data = "";
    }
    else {
        data = www.downloadHandler.text;
    }

}
```

Network Serialization

- Only send what you need
  - Don’t send **useless** information (the location of objects that aren’t dynamic)
  - Don’t send **superfluous** information (send a single bit representing that the player didn’t move, rather than their full position and rotation)
  - Don’t send **extraneous** information (don’t send an enum that has 8 values as a 4-byte integer)
  - Don’t send **uncompressed** information (don’t format everything as a string, encode data in as few bytes as possible)

Javascript

- Javascript is **dynamically typed** and **weakly typed**
  - What is ‘3’ == 3 ? And ‘3’ === 3?
  - What is 3 < 2 < 1 ?
- Javascript is a **scripting language** (just-in-time compilation)
- Over 97% of websites use Javascript client-side for web page behavior
- All major web browsers have a dedicated JavaScript engine
- Supports event-driven, functional, and imperative programming styles
  - **Event-Driven** – flow of the programming is determined by events (user actions, sensor outputs, message passing from other programs).
  - **Functional** – Declarative programming paradigm where instead of functions being imperative statements that change the state of the program, they’re trees of expressions that map values to other values.
  - **Imperative** – Statements that change the program’s state (program as series of commands for the computer to perform).

In comparison to C#

- Can only detect errors in Javascript while executing code
- Not compiled
- Cumbersome to maintain large applications
- Less consistent syntax than C# (because of dynamic typing, have unexpected behavior w/out knowing how to do things the “Javascript” way)
- Originally intended to be used only in web-browsers, but now is used in servers and other applications

Typescript

- Statically compiled
- Make Javascript code
  - Clearer
  - Simpler
  - Statically Typed
  - Generic Types
  - Classes
  - Interfaces
- Ideal for larger project

Node.js

- JavaScript runtime (open source server environment)
- Built on Chrome’s V8 JavaScript Engine
- Allows writing of command line and server-side scripts outside of a browser
- Allows asynchronous programming
  - Can handle multiple requests/inputs while processing previous requests
- Single Threaded
- Non-Blocking

Why Node.JS?

- There’s a module for everything
- Very wide adoption, easy to get help/fixes
- Can create dynamic behavior

WebSockets

- Tool for bidirectional communication between a browser client and a server
  - Allows server to communicate with the client without a client-request
- Essentially creates a pipeline for communication
  - Good for semi-realtime gameplay
  - Good for web based multiplayer games
  - Not lowest-lag solution (likely not appropriate for an FPS).

Heroku

- Cloud platform
  - Build
  - Deliver
  - Monitor
  - Scale
- Provide the infrastructure for globally shipping a server

Databases

What is a Database?

- Data
  - At rest (stored)
  - Structured (some kind of relationship to other data)
  - Organized (allow some kind of recall)
- Controlled by a Database Management System (DBMS)
  - Allow processing, management, modification, control, organization, and access
- The data and the DBMS is often referred to as a “database”
- Often modeled in rows and columns in a series of tables

What Does a Database Give Us?

- Databases provide **persistence**
  - Web applications are stateless
  - Exist only as a sequence of requests and responses
- Databases provide **data management**
  - Can process and compare data at scale
  - Often can be hosted on more powerful architecture than consumer hardware
  - Powerful analytics tool
- Databases provide **space**
  - If you’re dealing with data at bulk, they can fit the job.

Types of Databases - Relational

- What is it?
  - Data stored in multiple, related tables
  - Data stored as rows and columns
  - Relational Database Management System (RDBMS)
- Examples
  - Structured Query Language (SQL)
  - Oracle
  - MySQL, PostgreSQL
- Usages
  - For very structured data – SQL uses a “schema” to process/add/update data

Types of Databases - NoSQL

- What is it?
  - Any database that doesn’t us SQL as the primary data access language
  - Sometimes called “non-relational”
  - Data doesn’t have to conform to a schema
- Examples
  - Apache Cassandra
  - MongoDB
  - CouchDB
- Usages
  - Good for semi-structured or unstructured data
  - Can make changes to the database without affecting applications using the database

Types of Databases – Key-value

- What is it?
  - Simplest kind of NoSQL database
  - Sometimes referred to as a “key-value store”
  - Highly scalable
  - Can handle high volumes of traffic
- Examples
  - Amazon DynamoDB
  - Redis
- Usages
  - Session management for web aplications
  - User sessions for massive multi-player online games
  - Online shopping carts

Types of Databases - Document

- What is it?
  - Use JSON-like documents to model data instead of rows and columns
  - Sometimes referred to as “document-oriented databases”
  - Store and manage semi-structured data
  - Simple and scalable
- Examples
  - MongoDB
  - Amazon DocumentDB
  - Apache CouchDB
- Usages
  - Good for schema-less structured data
  - Useful for mobile apps
  - Useful for fast iteration

Types of Databases – Cloud

- What is it?
  - Any database that’s designed to run in the cloud
- Examples
  - Microsoft Azure SQL Database
  - Amazon Relational Database Service
  - Oracle Autonomous Database
- Usages
  - Very flexible
  - Very scalable
  - High availability
  - Often low maintenance (often use a Software as a Service (SaaS) model)

Types of Databases - Columnar

- What is it?
  - Also referred to as “column data stores”
  - Store data in columns rather than rows
- Examples
  - Google BigQuery
  - Cassandra
  - MariaDB
  - Azure SQL Data Warehouse
- Usages
  - Data warehouses
  - Great at handling analytical queries

Uses for Games

- Logging of player analytics
- Representing turn-based gamestate for online multiplayer
- Storing game data
  - Real-time leaderboard
  - Player notes/hints
- Storing support data
  - User submitted errors
  - Crash logs
- Notifications
- Account names/passwords/statistics
- Anti-cheating
- Combinatorial lookup (endgame/opening analysis)

Knowledge Share Prep