Introduction to Algorithmic Problem Solving

Game Development

Lesson 4 - Managing Objects

Data Structures

A **data structure** is a way for the computer to **organize information** in a way that's **easy to use.** We won't use all of them in this class, but we will use **Lists:**

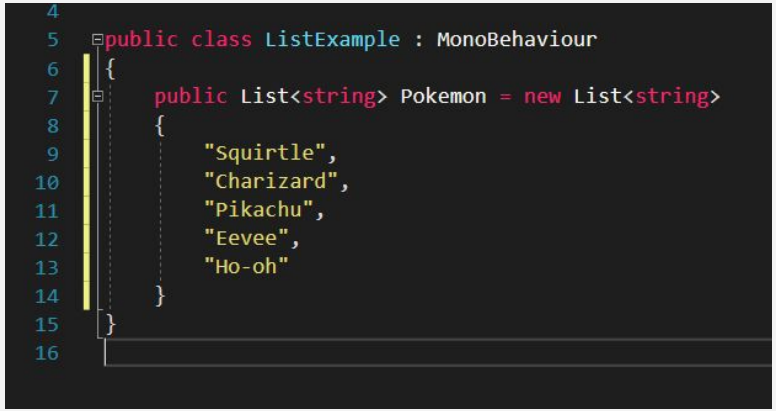
Some examples:

- **Lists**
- Trees
- Dictionaries
- Stacks
- Queues

Lists

A **list** is a **collection of values** ordered by **index.**

If this sounds familiar, it's because it's just like an **array**!



```
4
5   public class ListExample : MonoBehaviour
6   {
7       public List<string> Pokemon = new List<string>
8       {
9           "Squirtle",
10          "Charizard",
11          "Pikachu",
12          "Eevee",
13          "Ho-oh"
14      }
15  }
16
```

A **C# List** has some extra features that an **array** doesn't:

- **Adding elements** to the end of the list - no need to know its length.
- **Removing elements** from the list and having it automatically reorder.
- **Querying** the list to see if it contains a specific value.

Events

Unity's **Event System** is very powerful and extremely useful for **keeping your code clean.**

A desperate cry echoes in the night - an event! But will it be answered?

- An **event** can be invoked from **anywhere** in your code.
- **Any other** piece of code can **listen** to the event, and run its own code when the event is fired.
- The class that **fired** the event **doesn't have to know** about any of the code that listens to it.
- This is **very good** for avoiding spaghetti code.

Data Structures and Events

By the end of today, you will be comfortable using C# Lists to track the state of your game, and updating that state based on events.

Today's Plan:

15 min - Demonstration

15 min - Collaborative problem solving

75 min - Writing code