Lecture 15

Data Collection
The Rise of Big Data

• Big data is changing game design
  • Can gather data from a huge number of players
  • Can use that data to inform future content

• What can we do with all that data?
  • What types of questions can we answer?
  • How does it affect our business model?

• How do we collect all of this data?
  • What are the technical challenges?
  • What are the legal/ethical challenges?
The Rise of Big Data

• Big data is changing game design
  • Can gather data form a huge number of players
  • Can use that data to inform future content

• What can we do with all that data?
  • What types of questions can we answer?
  • How does it affect our business model?

• How do we collect all of this data?
  • What are the technical challenges?
  • What are the legal/ethical challenges?
Logging Game Data

Query 1

Query 2

Query 3

Data Collection
Issues with Data Collection

- You have to **record** the data
  - Capture data from the game state
  - Do it so it does not hurt performance

- You have to **store** the data somewhere
  - Lots of players means a lot of data
  - Use database or Google-style storage solution

- You have to **move** the data to the datastore
  - Requires an internet connection (while playing?)
Instrumenting Activity

Single Player Game

Online Game

LOG?

LOG?

Server

Client

Data Collection
Instrumenting Activity

Single Player Game

Online Game

LOG?

You Need to Access the Data Log
Instrumenting Activity

Single Player Game

Online Game

Data Collection
Instrumenting Activity

Single Player Game

Battery & Connectivity

Online Game

Server Load

Data Collection
Instrumenting Activity

Single Player Game

Online Game

Data Collection
Even This is Not Enough

- Disks (for logs) are slow
  - SDD: 500 MB/s
  - Standard Disk: 200 MB/s
  - Or about 8 MB/frame
- Memory (for state) is fast
  - RAM: ~15000 MB/s
  - Or about 250 MB/frame
- Cannot write fast enough!
  - Asynchronous logging?
Solution: Limit What you Collect

- Way too much data to collect *everything*
  - Example: 500 objects with 10 integer size fields
  - 20 KB/frame or 1.2 MB/s or 72 MB/minute
  - Console games have more objects, played longer

- What would you do if even if you could?
  - How do you search through all this data?
  - Just saying “use a database” is not enough
Aside: Asynchronous Still Desirable

- I/O has problems with **jitter**
  - Throughput is not constant
  - Might be in use elsewhere
- Update loop also has jitter
  - May finish before 16ms
  - “Sleeps” until next frame
- Remove jitter with a **budget**
  - Use time at end of update
  - Write as much as possible
  - Will keep up in the long run

Data Collection
## The Classic Tradeoff

### Performance
- Only collect what you need
  - Overcoming challenges
  - Significant game events
  - Unusual actions/interactions
- Instrumentation is complex
  - Identify the source of event
  - Find this location in code
  - Add log statement there

### Extensibility
- Who knows what I need?
  - Resource data can get large
  - Player position might matter
  - Eventually have full state
- How do I get more data?
  - Always log it all \((\text{no})\)
  - Ask programmer to add more instrumentation(?)
How Do We Solve this Problem?
How Do We Process All this Data?

• How you **query the data** is more important
  • Example: SQL, Map Reduce, custom set-up
  • This often determines data format for you.

• In general, use a **uniform schema** for the data
  • All data should have the same attributes
  • Avoid key-value **blobs**; make attributes structured
  • This means you **could** use SQL (even if you do not)
Is SQL Suitable for Analytics?

- Game state is (often) naturally relational
- Objects are a long list of attributes
- Store each game object as a row
-Lots of libraries can convert objects to SQL

<table>
<thead>
<tr>
<th>key</th>
<th>player</th>
<th>x</th>
<th>y</th>
<th>health</th>
<th>damage</th>
<th>range</th>
<th>flying</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>12</td>
<td>342</td>
<td>100</td>
<td>50</td>
<td>20</td>
<td>false</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>43</td>
<td>12</td>
<td>100</td>
<td>50</td>
<td>20</td>
<td>false</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>123</td>
<td>90</td>
<td>50</td>
<td>30</td>
<td>50</td>
<td>true</td>
</tr>
</tbody>
</table>
Or Maybe Not

Games are Nested Relational
Can Still Do This Relationally

**Player Table**

<table>
<thead>
<tr>
<th>player</th>
<th>…</th>
<th>inventory</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>…</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>…</td>
<td>11</td>
</tr>
<tr>
<td>3</td>
<td>…</td>
<td>12</td>
</tr>
</tbody>
</table>

**Join Table**

<table>
<thead>
<tr>
<th>player</th>
<th>item</th>
<th>qty</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>21</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>20</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>22</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>21</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>24</td>
<td>1</td>
</tr>
</tbody>
</table>

**Item Table**

<table>
<thead>
<tr>
<th>item</th>
<th>…</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>…</td>
<td>250</td>
</tr>
<tr>
<td>21</td>
<td>…</td>
<td>1000</td>
</tr>
<tr>
<td>22</td>
<td>…</td>
<td>50</td>
</tr>
<tr>
<td>23</td>
<td>…</td>
<td>100</td>
</tr>
<tr>
<td>24</td>
<td>…</td>
<td>5000</td>
</tr>
</tbody>
</table>

But how easy is this to process?
Analytics Involve Time-Sensitive Data

- Each event/data needs to be time-stamped
  - Time it happened in seconds/milliseconds/etc
  - Typically use an int, as it makes grouping easier
- Time attribute is an integral part of queries

<table>
<thead>
<tr>
<th>key</th>
<th>player</th>
<th>x</th>
<th>y</th>
<th>health</th>
<th>damage</th>
<th>range</th>
<th>flying</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>12</td>
<td>342</td>
<td>100</td>
<td>50</td>
<td>20</td>
<td>false</td>
<td>1209823</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>43</td>
<td>12</td>
<td>100</td>
<td>50</td>
<td>20</td>
<td>false</td>
<td>1309875</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>123</td>
<td>90</td>
<td>50</td>
<td>30</td>
<td>50</td>
<td>true</td>
<td>1724304</td>
</tr>
</tbody>
</table>
Example: Aggregating Daily Prices

- Average price of each item, each day
  - Want to group together item transactions *per day*
  - Use SEC_PER_DAY to convert time stamp to days

- Relatively simple *aggregate query*

```sql
SELECT item, avg(price), time/SEC_PER_DAY
FROM gamedata
GROUP BY time/SEC_PER_DAY
```
Hard Example: Quest Order

- Return first quest completed after tutorial level
  - Want a quest that is completed after tutorial
  - Want no other quest completed in between

- This is known as a time-series query

![Diagram showing tutorial, quest 1, and quest 2 sequence]
Time Series in SQL

• Return all pairs of consecutive time events

```sql
SELECT D1.name, D2.name
FROM Data as D1, Data as D2
WHERE D2.time > D1.time
EXCEPT
SELECT D1.name, D2.name
FROM Data as D1, Data as D2, Data as S
WHERE S.time > D1.time and D2.time > S.time
```

This gets REALLY messy
NoSQL is not Much Help

- Time series is a correlation of actions
  - Events X that happen after Y
  - Correlation requires a DB **join**

- NoSQL means **Map Reduce**
  - **Map**: Sort data into key, values
  - **Reduce**: Aggregate values of same keys
  - **Example**: Average price of an item

- Joins are **hard** in Map Reduce
  - Google gets scalability by “cheating”
Alternative: Data Stream Systems

- Given set of **event streams**
  - Database with time stamps
  - Can only read data in order
  - Can only read data once
- **Idea**: Get events as they occur
  - But this is a little unrealistic
  - Delay to create the event
- Also have a **pattern query**
  - Matches event/set of events
  - Outputs match as new event
  - Called a *composite event*

**Data Collection**
Using Data Stream Systems

External Stream System

Internal Stream System

Data Stream System

Game State

Event Stream

Event Stream

Pattern

Data Collection
Using Data Stream Systems

### External Event System
- Game acts as **base stream**
  - Matching done outside
  - Simple but needs bandwidth
- Several **commercial options**
  - Streambase
  - Microsoft StreamInsight
  - Oracle CEP
- Less **open source options**
  - Lot of academic prototypes

### Internal Event System
- Game handles **all matching**
  - Event streams in game state
  - Bandwidth only for output
- **You** must implement system
  - Maybe use open source
  - But always read licenses!
- Why do this? **Complex AI**
  - Just like rule matching!
  - **Bioshock Infinite** does this
Did I Mention A LOT of Data?

- Amount of data might be too much for one log
  - Just one player can create a significant amount

- Data probably in different categories
  - **Examples**: Items sold, quests finished

- Put the data in appropriate databases
  - Each might be on a different server

- Data streams can send data to the right place
Data Streams and Categorization
Putting this All Together

- We need to **record** the data fast
  - Requires a custom logging system in game
  - Have to be choosy about what data is recorded
  - Need to be flexible to respond to designers

- We need to **process** the data fast
  - Depends on the choice of query languages
  - Need support for aggregation and time-series
  - Data stream systems are ideal for this
Brainstorm: Mobile Data Collection