CPSC 427
Video Game Programming

Human Computer Interaction
and User Experience

Helge Rhodin
# ECS: Memory Locality

## Array of Structs (AoS)

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>vx</th>
<th>vy</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>y</td>
<td>vx</td>
<td>vy</td>
</tr>
</tbody>
</table>

- Default in object-oriented programming
- Also possible with ECS:
  ```
  struct Motion {
    float x, y;
    float vx, vy;
  }
  ```

## Structs of Array (SoA)

<table>
<thead>
<tr>
<th>x</th>
<th>x</th>
<th>x</th>
<th>y</th>
<th>y</th>
<th>y</th>
<th>y</th>
<th>vx</th>
<th>vx</th>
</tr>
</thead>
<tbody>
<tr>
<td>vx</td>
<td>vx</td>
<td>vy</td>
<td>vy</td>
<td>vy</td>
<td>vy</td>
<td>vy</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Default in ECS
- Efficient vector operations (SIMD)

```
struct PosX {float x;}
struct PosY {float y;}
struct PosVx {float vx;}
struct PosVy {float vy;}
```
Peer grading

• If unsure, leave a comment on why you graded what
• Leave constructive feedback

• No double counting *(be fair: imagine it is your submission)*
  • It's difficult and requires time/care
  • What if Task#1 is incorrect and Task#2 uses Task#1?
    • Grade Task#2 as if Task#1 would be correct
  • What if an incorrect Task#1 implementation eases Task#2?
    • Give partial points for both tasks
Teamwork!

- **We have 18 teams (104 students!)**
- **We have three TAs -> 6 teams for each TA**

- **TODO#1:** Register for a weekly 15-minute meeting slot
  - *choose a time that works for all teammates!*
  - *Put team ID here:*
    - [https://docs.google.com/spreadsheets/d/1K_8Vi9cZxowBcBevMG24ZgU1d3mfDvc7EAbhjwnMMus/edit?usp=sharing](https://docs.google.com/spreadsheets/d/1K_8Vi9cZxowBcBevMG24ZgU1d3mfDvc7EAbhjwnMMus/edit?usp=sharing)

- **TODO#2:** Create a github repository on the course team:
  - *Name it Team##GameName (note zero padded)*
  - [https://github.students.cs.ubc.ca/CPSC427-2023W-T1](https://github.students.cs.ubc.ca/CPSC427-2023W-T1)
Teamwork: Oral and written pitch!

• **TODO#3: Oral pitch (1 minute)**
  • On Wednesday
  • Fill your slide here: [https://docs.google.com/presentation/d/1h9wt4b-rBJ27OtjOcObe102B3uc59O6lhWNWGSBSibc/edit?usp=sharing](https://docs.google.com/presentation/d/1h9wt4b-rBJ27OtjOcObe102B3uc59O6lhWNWGSBSibc/edit?usp=sharing)

• **TODO#4: Submit your written pitch**
  • On Wednesday
  • Submit on your github repo, commit & push before the deadline
Designing for People (DFP)

- [https://dfp.ubc.ca/](https://dfp.ubc.ca/)
What are HCI & UX?

• Human Computer Interaction (HCI)
  • Research in designing & understanding the way humans and technology interact

• User Experience (UX)
  • Perception of a particular product, system or service

• Part of user-centered design
Even Big Companies Get UX Wrong

- Easy & expensive to get UX wrong

Google Glass failed in the market because it wasn’t clear why people should need it

and the privacy issue…
Connection to Game Design

• Impact of design on ease of use & engagement

In Wind Waker, the direction Link looked indicated to the player something of interest was there.

• Design applications & philosophies are interconnected
How do HCI and UX Connect to Game Design?

• Poor UX design can prevent players from experiencing games as intended

For example, having to follow in-game characters with different walk speeds than your characters
Game Design Philosophy

• **User-centered game design = Put players needs first**

• **Make play easy (& fun)**

• **Good design is often invisible**
  - *How to play is subtly implied*
Design Concepts

• **Design concepts:** Basic ideas that help us understand & design *what’s happening* in a user interface

• **Norman’s Design Concepts:**
  - Affordances
  - Mapping
  - Feedback
  - Constraints
  - Visibility
  - Consistency
Affordances

• **Affordance** is a physical characteristic that suggests **function**
  – *i.e. inviting interaction/use*

• **Chairs afford sitting**, but so do tables, boxes, and floors
Example of Affordances in Games

What does the pipe afford?
Example of Affordances in Games

• What does the slingshot afford here?

• What do the blocks afford?

• What does the (pause) button afford?
Mapping

- Some controls are direct (slingshot), some indirect (button)
- **Mapping** is the relationship between look/feel of indirect controls & their implied actions

<table>
<thead>
<tr>
<th>Control</th>
<th>Implied action</th>
</tr>
</thead>
<tbody>
<tr>
<td>push button</td>
<td>start/stop function</td>
</tr>
<tr>
<td>twist knob</td>
<td>increase/decrease value</td>
</tr>
<tr>
<td>turn wheel</td>
<td>rotate left/right</td>
</tr>
</tbody>
</table>
Mapping Example

- Which is better?
Mapping Example

- Natural mapping minimizes the need for labeling relationships
Mapping Example in Games

Clear mapping between up, down, right & left controls and game in Zelda.
Feedback

- **Feedback**: response to action
- The color **changes** to inform us a connection has been made
- The **sound** of a ‘click’ tells us if it connected to the port
Feedback in Games

- Feedback in games is **continuous**

  ![Game Screenshot]

- Visual
  - *interaction between sprites*

- Sound
  - *music on defeat*

- Touch
  - *controller vibrating*
Design Principles Example in Games

• Affordances?
• Mappings?
• Feedback?
Design Principles

• Affordances
• Mapping
• Feedback
Users

- Who are the players?
  - Age: Children, adults, university students
  - Culture

- Where will they be playing?
  - Commuting, at home, remotely

- What do they need or want?
  - Fulfilling plot, relaxing play
Examples

• Who is this game designed for?
  (A) children  
  (B) adults  
  (C) elderly  
  (D) all ages

Why does it matter?  
.... Design choices...
Examples

- Who is this game designed for?
Examples

• Who is this game designed for?
  (A) children
  (B) adults
  (C) elderly
  (D) all ages

Why does it matter?
Examples

- What do the players of this game want?
  (A) fast-paced action
  (B) relaxing play
  (C) rich environments
  (D) other
What Motivates Users?

- Work has been done to identify player types
- Users can be classified by preference for interacting/acting with/on others/the world
- The four classifications tell us what motivates each player type
Think:

• Who is your game designed for (demographics/type)?

• What do the players of your game want?

• (How is your game going to stand out?)
The Design Process

Brainstorming

Sketch → Wireframe → Mockup → Prototype → Release

Low fidelity prototyping

High fidelity prototyping
Low Fidelity Prototyping

- Used for **early** stages of design
  - *Quick & cheap to deploy*
  - *Easy to test*

- Iterate on **story** and core gameplay mechanics

- **Sketches** are a great way to start designing
Testing Low Fidelity Prototypes

• Don’t commit to one approach, design a few prototypes & compare

• Invite someone to try them out

• Try to drill down on feedback
  – *If they just say it’s “fun”, ask why?*
Fail Early, Fail Often, and Iterate on Feedback

• Designing something that people will use is both an art & a science
  – *Iteration is how you make it better*

• Early feedback ensures design meets users’ needs

• Throwing around ideas is *quick*
  – *Fixing a bad design is expensive*

• No idea is perfect the first time around
Medium Fidelity Prototyping

• Use medium fidelity prototyping for the early to middle stages of design
  – Identify questions before coding
  – Be selective with what gets built
  – Get it right in black and white first

• Iterate on tone & feel of game
  – Supplementary game mechanics
  – Rough visuals & audio
  – Feedback
Greyboxing

- Greyboxing blocks out all elements as shapes to test gameplay
High Fidelity Prototyping

• High fidelity prototyping happens during the late stages of design
  – Alpha & beta releases
  – Polish artwork
  – Perform playtesting
  – Fix bugs
  – Release

• Fine tuning before release
Technical Designs
The Light Gun

Classic: NES Zapper

http://www.arcadecab.com/News.htm
The Light Gun (first glance)

- a laser?!  

https://makingstudio.blog/2017/09/20/teardown-nintendo-zapper/
Principle I: Black&white target

The Light Gun

- the sensor (single-pixel-camera) is in the gun,
- receive light from the on-screen targets,
- flash the screen, and ???

Light sensor! laser
Principle II: Timing on Cathode Ray Tube (CRT) displays
Video Signal From NES

LINE 11  PIXEL 16

LIGHTGUN WITH CRT TV
Read the zoom chat?

https://github.com/tesseract-ocr/tesseract

• does optical character recognition
• works with c++
• works on windows and linux (not sure about mac)
• might be too slow?!

How to apply tesseract on a screen capture (zoom)?

• https://stackoverflow.com/questions/22924209/how-to-make-tesseract-ocr-read-from-coordinates-on-a-screen
Can we exploit the Zoom window?

• **Multi player?**
Read the zoom chat (hacks)

• **Capture the screen**
  - [https://github.com/smasherprog/screen_capture_lite](https://github.com/smasherprog/screen_capture_lite)

• **Search for the zoom window**

• **Check for colored symbol**
  - red, green, gray, blue?
    - *only need to read a few pixels*
    - *it's fast!*

• **Recognize numbers?**
  - *only 10 different ones, brute force?*
Mouse gestures

Regression
• **least squares fit**
  • linear, polynomial, and other parametric functions

Search
• brute force?
• binary search?

Detection
• key events
• pattern matching
Mouse gesture detection

1. Determine start and end time, i.e. store all mouse curser positions in a vector.
2. Resample your vector to have a fixed number of elements (e.g., N=20). This is done to gain invariance to different drawing/sampling speeds.
3. Subtract the start point (or the mean of the curve) as reference point. Yields translation invariance, it should not matter where on the screen you draw.
4. If you want scale invariance (detect small and big circles), divide all points by the maximum or mean position of all points (you need to try what is better)
5. Compare this normalized curve to a reference curve (you drawing the pattern once for reference and saving the points) that was processed with 1-4. The comparison metric could simply be the distance between the N points in the reference and new curve (after all the normalizations).

Debugging: Plot the curve after every processing step, e.g., save as .csv and plot in excel (to save you from coding a graph plotter)

http://depts.washington.edu/acelab/proj/dollar/index.html