Outline

1. Review of Capabilities: Users, hardware, software, etc.
2. The Design Process: Task analysis, Development, Evaluation, Trade-offs
3. VE Development: Considerations & Suggestions

from Lecture 1.1:

VR pushes the technical and application boundaries of many disciplines (and is therefore interesting and appropriate to study in these fields):

- **Computer Graphics Technology** – challenges of real-time: efficiency and effectiveness
- **Computer Science** – parallel and distributed systems; networking; end-to-end system performance
- **Human-Computer Interaction** – spatial I/O, multi-sensory interaction; evolving metaphors; need for rigorous evaluation
- **Informatics** – breadth of application areas; effective interface to data/information deluge science, computer graphics, multi-media, and human-computer interaction design.

Challenges of VR

- diversity of hardware
- diversity of input/output methods (vs. desktop systems)
- multi-modal presentation
- diversity of users and applications
- lack of universal standards (hardware, software, interfaces)
- severe nature of trade-offs (quality, speed, accuracy, usability, cost, etc.)

1. Review of Capabilities

1.1 Human (user) capabilities

(in approximate order of importance, practicality, and availability)

- Visual
- Auditory
- Motor
- Haptic (tactile & kinesthetic)
- Olfactory
- Gustation

1.2 Display capabilities

- Visual displays - resolution, brightness, stereo, field of view, refresh rate, etc.
- Audio displays - reproduction, localization, HRTF, etc. (more after spring break - March 30)
- Haptic devices - DOF input, DOF output, output range and frequency, working volume, etc.
1.3 Input capabilities

- Tracking - accuracy, range, latency, degrees of freedom
- Input devices (controllers) - DOF, discrete vs. continuous, efficiency, accuracy, ease of acquisition, etc.
- Speech/Text input

1.4 Computation and Graphics capabilities

- 3D graphics
  - rendering performance: vertices/triangles per second
  - Advanced rendering features - texture techniques, lighting tricks, new shader languages
  - quality issues: anti-aliasing, gamma correction, color balancing, refresh rates, etc.
- Video and image-based techniques
- Simulation computation
  - scientific computation
  - object updates (e.g., physics model)
  - collision detection
- End-to-end latency
- Networking
  - bandwidth
  - latency and jitter
  - unicast/multicast
  - reliable/unreliable

1.5 VR Software capabilities

- Device drivers
- Interface to OS and windowing system
- Device and display abstraction
- automation/hiding of standard or repeated tasks
- low-level “rendering” (visual, audio, haptic)
- high-level rendering - scene graph
- object behaviors
- interaction methods
- collaboration

1.6 Software techniques and algorithms

- Navigation
- Selection
- Manipulation
- Menus
- Collaborative methods - avatars, co-navigation, shared interfaces, etc.

1.7 Developer capabilities!

- programmer - low-level, high-level
- content developer - modeling, animation
- artist/cinematographer - textures, lighting, arrangement, camera planning
- HCI specialist - interface design

2. The Design Process

2.1 Task Analysis

- What problem are you trying to solve?
- Who are you solving it for?
Why use VR? (What are the expected benefits of VR vs. some other method?)

2.1.1 Uses of VR

- Online Performance
- Offline Training and Rehearsal
- Online Comprehension
- Offline Learning and Knowledge Acquisition
- Online Design
- Entertainment
- Communication
- Tool for Researching Human Perceptual-Motor Capabilities

2.1.2 Defining the User

- single vs. multiple users
- remote vs. proximate (local) users
- novices vs. experts
- user aptitude
- individual physiological differences
- users with disabilities
- real user vs. "made up" user or self

2.2 Iterative Design Process

(Stuart: Intro, Chapter 1)

- 1. Task specification
- 2. Prototype implementation
- 3. Small, formative evaluation
- 4. Refine implementation
- 5. Formal, objective evaluation
- (iterative process over steps 3-4, or 1-4, or even 1-5)

2.3 Evaluation - Measuring Effectiveness of the VE

(Stuart: Chapters 9-12)

2.3.1 Measuring System Performance

- Total Latency
- Display Update Rate
- Synchronization of spatial and temporal cues
- Registration
- Robustness and Fault Tolerance
- Working Volume
- Data size / Scene complexity
- Number of users (for collaboration)

2.3.2 Measuring Usability of the VE

- Factors to Measure
  - Task Completion
  - Errors
  - Task Completion Time
  - Learning
  - Workload
  - Novice vs. Expert behavior
Subjective Impressions
Cybersickness
Other psychological after-effects

Methods of Gathering Information
- videotape analysis
- automatic data collection
- think-aloud protocol
- Likert-style questionnaire
- open-ended interview
- physiological monitoring

2.3.3 Measuring Effectiveness of the VE
- Online Performance - successful completion of the task in the real world
- Offline Training and Rehearsal - positive transfer to performance in the real world
- Online Comprehension - insights gained (especially those that could not have been gained otherwise)
- Offline Learning and Knowledge Acquisition - successful acquisition and synthesis of knowledge and understanding
- Online Design - good designs produced
- Entertainment - user enjoys the experience (w/o hurling)
- Communication - VE facilitates effective communication
- Tool for Researching Human Perceptual-Motor Capabilities - isolate perceptual/motor phenomena to be studied

2.4 Design Trade-offs (Stuart: Chapter 8)
- General-purpose vs. Special-purpose VE Systems
- Degree of Encumbrance and Choice of “Flavors”
- Degree of Customizability for Individual Users
- Responsiveness vs. Image Quality vs. World Complexity
- Sharing Resources vs. Bandwidth
- Optimization vs. Cross-platform Compatibility
- Standalone vs. Networked
- Specific Technologies
  - displays
  - tracking
  - input devices
- Costs
  - hardware
  - software
  - development
  - maintenance

3. VE Development - Considerations and Suggestions

3.1 VE Development Considerations
- What is the target display?
  - multi-screen -> VR API
  - single screen (semi-immersive) -> GLUT w/ stereo and tracking may be enough
  - high-resolution
  - multiple display configurations ?
  - haptics
  - *dictates platforms and APIs*
- What are the associated computing requirements?
  - single-threaded or multi-threaded
  - single-processor or multi-processor
- graphics capabilities
  - may impact your choice of operating system, API, and language

How do you intend to deliver the content?
- local system only
- network delivery
- Web
- non-real-time formats: images, movies, stereo imagery
  - impacts the path and tools used for content creation

Is the application stand-alone or collaborative?
- collaboration is best designed into an application from the start (Leigh & Johnson, EVL)

What graphics library to use?
- what is the nature of your application?
  - data-driven -> OpenGL (or visualization library, )
  - visual simulation (hand-developed content) -> scene-graph (VRML, OSG, Performer, etc.)

Does the system need to be scalable?
- develop with small problems and data sets
- be sure to verify occasionally with larger data sets
- is content creation/conversion scalable?

Other practical considerations
- availability of development systems and simulator modes
- support for multiple developers in a team approach

3.2 VE Development Suggestions

- Play to your strengths
  - do what you know: modeling, programming
  - get something working first, try new stuff later
  - if possible, assemble a "Renaissance Team" of subject matter experts, programmers, artists, game developers, musicians, psychologists, and interface designers

- Rapid prototyping is important
  - external users have a short calendar "window of interest" (esp. in an academic environment)
  - it's very helpful to get user's data up and running ASAP
    - there's something "magical" about users seeing their own data/environment
  - standard, simple interfaces are very helpful (e.g., VRML, Perfly, etc.)
    - easy for users (and other developers) to understand
    - promotes further sharing with other colleagues
  - don't plan/design it to death - seeing a result is highly motivating and will promote the iterative design process

- Be aware of limitations of high-level tools and low-level tools
  - high-level, rapid prototyping tools will often "max out"
    - simple things are easy
    - hard things become very hard and awkward
    - some things are nearly impossible
  - general APIs take longer to ramp up on, but usually won't leave you at a dead-end
    - simple things are hard (or at least tedious)
    - hard things are only proportionally harder
  - use a combination of high-level and low-level to your advantage
    - early prototypes in VRML/EON
    - intermediate, immersive designs in VRJuggler/CAVElib with scene-graph API and object import
    - final implementation with VRJuggler/CAVElib with OpenGL and direct data import

- Open Source vs. Commercial/Closed Source
  - <insert usual flaming verbage here>
- open-source
  - removes adoption hurdle - nice given added expense of VR hardware
  - protect investment in learning and development time
- Anticipate and plan for multiple and open-ended development routes
  - Content and data are the hard part.
    - Make sure content is developed in an accessible, but neutral format.
  - Basic display code is easy and "throw-away"
    - don’t become too attached to a design
  - Plan for modularity and reuse