Lesson 01 - Introduction to Interaction Design

1. Introduction to Interaction Design (IxD)
    - Defines structure and behavior of interactive systems
    - Aims to create meaningful relationships between people and products/services
    - Involves designing interactive products to support communication and interaction
    - Good examples: smartphones, iPads, navigation systems
    - Bad examples: SharePoint, self-checkouts, poorly designed lift controls
    - Considers users, activities, context, and optimizes interactions
2. Transdisciplinary Nature of IxD
    - Involves psychology, social sciences, computing, engineering, and design fields
    - Practiced within HCI, ubiquitous computing, cognitive engineering, information systems, etc.
    - Multidisciplinary teams enable more ideas but present communication challenges
3. User Experience (UX)
    - Encompasses product behavior, usage, and users' feelings, pleasure, and satisfaction
    - Cannot be designed directly, only designed for
    - Affected by pragmatic (simplicity) and hedonic (stimulation) qualities
4. Key Principles of IxD
    - Involve users throughout development
    - Identify clear usability and UX goals upfront
    - Use iteration: discovering requirements, designing alternatives, prototyping, evaluating
    - Consider accessibility and inclusivity for widest range of users
5. Usability and UX Goals
    - Usability: effectiveness, efficiency, safety, utility, learnability, memorability
    - UX: desirable aspects (fun, entertainment, helpfulness, motivation) vs. undesirable aspects (boredom, frustration, annoyance, cuteness)
    - Potential tradeoffs between usability and UX goals
6. Design Principles
    - Visibility, feedback, constraints, logical mapping, consistency, affordances
    - Act as guiding heuristics for analyzing and evaluating interactive products
7. User-Centered Design
    - Early focus on users and tasks
    - Empirical measurement
    - Iterative design
    - Four basic activities: discovering requirements, designing alternatives, prototyping, evaluating

Lesson 02 - Data Gathering and Analysis

1. Topics Covered in Lecture 2
    - What data needs to be gathered
    - Planning and running a successful data gathering program
    - Conducting interviews, designing questionnaires, and planning observations
    - Analyzing qualitative and quantitative data, and presenting findings
    - Software packages for data analysis
2. Key Issues in Data Gathering
    - Setting goals and deciding how to analyze data
    - Identifying participants and determining sample size
    - Maintaining clear and professional relationships with participants
    - Obtaining informed consent when appropriate
    - Triangulation: collecting data from multiple perspectives and sources
    - Conducting pilot studies
3. Data Recording Methods
    - Notes, audio, video, and photographs (individually or combined)

- Each method has different challenges and advantages

4. Interviews
    - Unstructured, structured, or semi-structured (recommended)
    - Focus groups: group interviews that generate discussion and debate
    - Avoid long, compound, jargon-filled, leading, or biased questions
    - Follow a logical structure: introduction, warm-up, main body, cool-off, closure
    - Can be conducted using digital conferencing systems (e.g., Skype, Zoom)
5. Questionnaires
    - Closed questions (easier to analyze) vs. open questions
    - Administered to large populations via paper, email, or web
    - Design considerations: question order, clear instructions, balanced layout, question length
    - Response formats: checkboxes, rating scales (Likert and semantic), open-ended responses
    - Encourage good response rates through clear purpose, anonymity, incentives, and follow-up
6. Online Questionnaires
    - Advantages: quick distribution and response, cost-effective, easy data collection and analysis
    - Challenges: unknown population size, preventing multiple responses, question alteration
    - Deployment steps: plan timeline, design offline, program online, test, and recruit participants
7. Observation Methods
    - Direct observation in the field or controlled environments
    - Indirect observation: diaries, interaction logging, video/photo remote collection
    - Structuring frameworks to guide observation (e.g., person, place, thing; Robson's framework)
    - Planning considerations: level of involvement, acceptance, sensitive topics, data collection
8. Data Analysis
    - Quantitative data (numbers) vs. qualitative data (themes, patterns, stories)
    - Basic quantitative analysis: averages (mean, median, mode), percentages, graphical representations
    - Question design affects analysis: open (analyzed separately) vs. closed (analyzed quantitatively)
    - Basic qualitative analysis: critical incidents, identifying themes (inductive), categorizing data (deductive)
    - Interaction analysis: understanding interactions between people and artifacts using empirical observations
    - Tools: spreadsheets, statistical packages (SAS, SPSS), qualitative data analysis tools (NVivo, Dedoose)
9. Interpreting and Presenting Findings
    - Use visualizations (e.g., bubble diagrams, timelines) to communicate insights effectively
    - Employ structured notations and stories to present different viewpoints and summarize findings

Lesson 03 - Interfaces and Interaction Paradigms

1. Evaluation: Why, What, Where, and When
    - Continuous iterative design and evaluation process
    - Why: Check users' requirements, usability, and satisfaction
    - What: Conceptual models, prototypes, and competitive products
    - Where: Natural settings, labs, and in-the-wild
    - When: Throughout the design process and for finished products
2. Types of Evaluation
    - Controlled settings directly involving users (most popular)
    - Natural settings involving users (gaining popularity)
    - Settings not directly involving users (less common)
3. Living Labs
    - Evaluate people's use of technology in everyday lives
    - Examples: Aware Home, EU-funded research projects in cities
4. Evaluation Case Studies
    - Experimental investigation of physiological responses in computer games
    - Ethnographic study of visitors at the Royal Highland show using a mobile app
    - Crowdsourcing opinions and reactions to inform technology evaluation
5. Participants' Rights and Informed Consent

    - Participants must be informed about the evaluation purpose, process, and their rights
    - Informed consent forms act as a contract between participants and researchers
    - Evaluation design, data analysis, and storage methods require ethical approval

6. Interpreting Data

    - Consider reliability, validity, ecological validity, biases, and scope of the results

7. Usability Testing

    - Involves recording performance of typical users doing typical tasks
    - Conducted in controlled settings with 5-10 representative users
    - Quantitative performance measures include success rates, time, errors, and satisfaction

8. Experiments

    - Test hypotheses about relationships between variables (independent and dependent)
    - Different experimental designs: between-subjects, within-subjects, and matched-pairs
    - Must be replicable and validated statistically

9. Field Studies

    - Conducted in natural settings to understand user behavior and technology impact
    - Used to identify opportunities, determine requirements, and evaluate technology in use
    - Example: Painpad study in hospitals to evaluate a pain-monitoring device

10. Inspections (Evaluation without Users)

    - Expert reviews (formal or informal) based on knowledge of users and technology
    - Heuristic evaluation guided by a set of usability heuristics (e.g., Nielsen's heuristics)
    - Cognitive walkthroughs focusing on ease of learning through usage scenarios

11. Web Analytics and A/B Testing

    - Web analytics: Analyzing user activities on websites to inform design improvements
    - A/B testing: Large-scale experiments comparing user performance on two design versions

12. Predictive Models

    - Evaluating products or designs without directly involving users
    - Based on expert error-free behavior (e.g., Fitts' Law for pointing time prediction)

Lesson 04 - Usability Testing and Evaluation

1. Prepare Your Test Plan

    - Include test plan name, scenario name, goals, quantitative and qualitative measurements, scenario, task list, post-scenario interview or questionnaire questions, and test setup details
    - Different roles have different tasks (e.g., admin, users)

2. General Procedure for Usability Test Sessions

    - Prepare test room, greet the guest, conduct pre-test questions, explain interface, present the scenario, administer post-scenario questionnaire or interview, repeat steps for each scenario, conduct post-test questionnaire, thank the participant, and organize files

3. What, Why, and When to Evaluate

    - What: Evaluate various features, some in the lab and others in natural settings
    - Why: Ensure users can use the product and like it
    - When: During design (formative evaluation) or after the design is finished (summative evaluation)

4. Persona, User Story, Scenario, and Storyboard

    - User research is the first step in designing around users
    - Persona: Archetype or character representing a potential user
    - User story: Short statement identifying the user and their need/goal
    - Scenario: Situation capturing how users perform tasks on the site or app
    - Storyboard: Visual representation of how the user would interact with the site or app

5. Example of a Scenario

    - Mary, mother of 16-year-old Amy, wants her daughter to learn about her culture
    - Amy finds cultural heritage sites uninteresting

- Mary finds a mobile app that enables visitors to experience life back in time
- Amy uses smart eyeglasses with the app to immerse herself in the historical atmosphere
- The experience changes Amy's perspective and appreciation for history and her culture

6. Characteristics of a Perfect User Scenario

    - Short story clearly defining the context of product use
    - Answers questions about the user, their goals, how they achieve them, and why they choose the product
    - Keeps scenarios close to reality using typical user expressions and wording

7. Examples of Prototyping and Field Studies

    - Prototyping example: Smart Ambient, a pilot study to contextualize mobile learning in the domain of cultural heritage
    - Field study example: Design challenges for mobile and wearable systems to support learning on the move at outdoor cultural heritage sites

8. Examples of User Testing Using Paper-Based Prototypes

    - Several YouTube video examples demonstrating user testing with paper-based prototypes

Lesson 05 - Design and Prototyping

1. Visibility of System Status

    - Designs should keep users informed about what is going on through appropriate, timely feedback.
    - Example: Interactive mall maps show people their current location to help them navigate.

2. Match Between System and the Real World

    - The design should use words, phrases, and concepts familiar to the user, rather than internal jargon.
    - Example: Stovetop controls are mapped to the corresponding heating elements for easy understanding.

3. User Control and Freedom

    - Users often perform actions by mistake and need a clearly marked "emergency exit" to leave the unwanted action.
    - Example: Digital spaces, like physical spaces, need quick "emergency" exits.

4. Consistency and Standards

    - Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.
    - Example: Check-in counters are usually located at the front of hotels, meeting user expectations.

5. Error Prevention

    - Good error messages are important, but the best designs carefully prevent problems from occurring in the first place.
    - Example: Guard rails on curvy mountain roads prevent drivers from falling off cliffs.

6. Recognition Rather Than Recall

    - Minimize the user's memory load by making elements, actions, and options visible. Avoid making users remember information.
    - Example: People are likely to correctly answer "Is Lisbon the capital of Portugal?" based on recognition rather than recall.

7. Flexibility and Efficiency of Use

    - Shortcuts, hidden from novice users, may speed up the interaction for expert users.
    - Example: Regular routes are listed on maps, but locals with more area knowledge can take shortcuts.

8. Aesthetic and Minimalist Design

    - Interfaces should not contain irrelevant information. Every extra unit of information competes with the relevant units.
    - Example: A minimalist three-legged stool still serves its purpose as a place to sit.

9. Help Users Recognize, Diagnose, and Recover from Errors

    - Error messages should be expressed in plain language (no error codes), precisely indicate the problem, and constructively suggest a solution.
    - Example: Wrong-way signs on the road remind drivers that they are heading in the wrong direction.

10. Help and Documentation

    - It's best if the design doesn't need additional explanation. However, it may be necessary to provide documentation to help users complete their tasks.

- Example: Information kiosks at airports are easily recognizable and solve customers' problems in context and immediately.

Lesson 06 - Interfaces and Interaction

1. 20 Interface Types Covered
    - Command, Graphical, Multimedia, Virtual Reality, Web, Mobile, Appliance, Voice, Pen, Touch, Gesture, Haptic, Multimodal, Shareable, Tangible, Augmented Reality, Wearables, Robots and Drones, Brain-Computer Interaction, Smart
2. Graphical User Interfaces (GUIs)
    - Originated from Xerox Star's WIMP (Windows, Icons, Menus, Pointing Device)
    - Windows: scrollable, stretchable, overlapping, movable screen sections
    - Icons: pictograms representing applications, objects, commands, and tools
    - Menus: lists of options that can be scrolled and selected
    - Pointing Device: mouse controlling the cursor for interacting with GUI elements
3. Window Design
    - Overcome physical constraints of computer displays
    - Enable viewing more information and performing tasks
    - Multiple windows can make finding the desired one difficult
    - Techniques like listing, tabbing, and thumbnails can help
4. Menu Styles
    - Flat list, drop-down, pop-up, contextual, collapsible, mega
    - Each style has its own advantages and use cases
5. Icon Design
    - Assumed to be easier to learn and remember than commands
    - Can be compact and variably positioned on the screen
    - Mapping between representation and referent can be similar, analogical, or arbitrary
    - Most effective icons are similar ones
6. Multimedia
    - Combines graphics, text, video, sound, and animation with interactivity
    - Users click links to navigate, play animations or videos
    - Provides better ways of presenting information when done carefully and consistently
7. Virtual Reality
    - Computer-generated graphical simulations providing an illusion of participation
    - Enables interacting with objects and navigating in 3D space
    - Creates highly engaging user experiences
    - Applications in video games, therapy, travel planning, architecture, design, education
8. Website Design
    - Early websites were text-based with hyperlinks, focusing on information structure
    - Modern websites emphasize distinctive, striking, and aesthetically pleasing designs
    - Need to consider designing for multiple platforms (keyboard vs. touch)
    - Balancing usability and aesthetics is crucial
9. Mobile Interfaces
    - Handheld devices used while on the move (phones, fitness trackers, smartwatches)
    - Larger-sized tablets used in mobile settings by professionals
    - Key concern is the hit area (space for fingers to press accurately)
    - Fitts' law can help design the right spacing
10. Appliances
    - Everyday devices in home, public places, or car (washing machines, remotes, toasters)
    - Personal devices like digital clocks and cameras
    - Used for short periods, need to be usable with minimal learning
    - Design as transient interfaces with short interactions, consider soft vs. hard controls
11. Voice User Interfaces

- Person talking with a spoken language app (timetable, travel planner, phone service)
- Used for inquiring about specific information or performing transactions
- Also used by people with visual impairments
- Design challenges: keeping conversation on track, efficient navigation, error recovery

12. Pen-based Devices
    - Enable writing, drawing, selecting, and moving objects using light pens or styluses
    - Capitalize on well-honed drawing skills developed from childhood
    - Digital ink (e.g., Anoto) combines ordinary ink pen with digital camera for recording
13. Touchscreens
    - Single touchscreens detect presence and location of a person's touch (kiosks, ATMs)
    - Multi-touch surfaces support dynamic finger actions (swiping, flicking, pinching)
    - Used in smartphones, tablets, tabletops, supporting one and two-hand gestures
    - Design concerns: display size, orientation, shape, and their effect on collaboration
14. Gesture-based Systems
    - Involve moving arms and hands to communicate, using camera recognition and sensors
    - Gestures need to be presented sequentially to be understood
15. Haptic Interfaces
    - Provide tactile feedback by applying vibration and forces to a person's body
    - Vibrotactile feedback can simulate the sense of touch between remote people
    - Design considerations: actuator placement, intensity, context, and creative applications
16. Multimodal Interfaces
    - Combine multiple input modes (e.g., facial recognition, gestures, voice recognition)
    - Example: Kinect camera detects multimodal input in real-time to build a person's model
17. Shareable Interfaces
    - Designed for more than one person to use, providing multiple inputs
    - Examples: large wall displays with pens or gestures, interactive tabletops with fingertips
18. Tangible Interfaces
    - Physical objects coupled with digital representations, manipulating objects causes digital effects
    - Example: VoxBox, a tangible system for gathering opinions at events
19. Augmented Reality
    - Virtual representations superimposed on physical devices and objects
    - Examples: Pokémon Go (using smartphone camera and GPS), medical applications
20. Wearables
    - Devices worn on the body (jewelry, glasses, fabrics) to interact with digital information
    - Applications: automatic diaries, tour guides, cycle indicators, fashion clothing
21. Brain-Computer Interfaces
    - Allow direct communication between the brain and an external device
    - Example: a woman who is paralyzed using BCI to select letters on a screen
22. Robots
    - Main types: remote robots (hazardous settings), domestic robots (household chores), pet robots (companions), sociable robots (collaborating with humans)
23. Drones
    - Unmanned aircraft controlled remotely, used in entertainment, agriculture, wildlife protection
    - Can fly low and stream photos to ground stations for mapping and analysis
24. Smart Interfaces
    - Devices connected to the internet and other devices, having some intelligence
    - Context-aware: understand surroundings and execute appropriate actions (e.g., Nest thermostat)
    - Human-building interaction: buildings designed to sense and act on behalf of inhabitants

Lesson 07 - Data Visualization

1. What is a Prototype?
    - One manifestation of a design that allows stakeholders to interact with it
    - In interaction design, prototypes can be screen sketches, storyboards, PowerPoint slides, videos, physical objects, cardboard mock-ups, or limited functionality software
2. Why Prototype?
    - Evaluation and feedback are central to interaction design
    - Stakeholders can see, hold, and interact with a prototype more easily than a document or drawing
    - Team members can communicate effectively, ideas can be tested out, and prototyping encourages reflection
    - Prototypes answer questions and support designers in choosing between alternatives
3. Low-fidelity Prototyping
    - Uses a medium unlike the final medium (e.g., paper, cardboard)
    - Quick, cheap, and easily changed
    - Examples: 'Post-it' notes, storyboards, sketches of screens and task sequences, 'Wizard-of-Oz'
4. Storyboards
    - A series of sketches showing how a user might progress through a task using the product
    - Often used with scenarios, bringing in more detail and a chance to role-play
5. Sketching
    - Low-fidelity prototyping often relies on sketching
    - Don't be inhibited about drawing ability; practice simple symbols
6. 'Wizard-of-Oz' Prototyping
    - The user thinks they are interacting with a computer, but a human is responding to output rather than the system
    - Usually done early in design to understand users' expectations
    - Requires a script, a participant to play the role of the end user, and a human "wizard" simulating the completed product's behavior
7. High-fidelity Prototyping
    - Uses materials expected in the final product; looks more like the final system than a low-fidelity version
    - Can be developed by integrating existing hardware and software components
    - Danger that users think they have a complete system
8. Compromises in Prototyping
    - Prototyping involves compromises (e.g., slow response, sketchy icons, limited functionality)
    - Two common types of compromise: horizontal (wide range of functions, little detail) and vertical (a lot of detail for only a few functions)
    - Compromises in prototypes must not be ignored
9. Conceptual Design
    - A conceptual model outlines what people can do with a product and what concepts are needed to understand and interact with it
    - Understand problem space, current requirements, and empathize with users
    - Use creativity and brainstorming techniques (e.g., mood board)
    - Consider alternatives using scenarios and prototyping
10. Concrete Design
    - Emphasis differs from conceptual design
    - Aspects include color, icons, buttons, interaction devices, user characteristics, context, inclusiveness, accessibility, cross-cultural design, and indigenous knowledge
11. Generating Prototypes
    - Break down scenarios into steps and create a scene for each step to generate a storyboard
    - Sketching out a storyboard prompts designers to think about design issues
    - Generate a card-based prototype from a storyboard or use case by considering each step and drawing a card that captures the interaction element
12. Explore the User's Experience

    - Use card-based prototypes to model the user experience
    - Visual representations called customer or user journey maps, or experience maps
    - Common representations: wheel and timeline

13. Construction: Physical Computing
    - Build and code prototypes using electronics with toolkits like Arduino, LilyPad, Senseboard, BBC micro:bit, and MaKey MaKey
    - Designed for use by a wide range of people
14. Construction: Software Development Kits (SDKs)
    - Programming tools and components to develop for a specific platform (e.g., iOS)
    - Includes IDE, documentation, drivers, sample code, and APIs
    - Makes development much easier
    - Examples: Amazon's Alexa Skills Kit, Apple's ARKit, Microsoft's Kinect SDK

Lesson 08 - Usability Heuristics

1. The Practice of Interaction Design
    - Agile development: short iterative development cycles with early and repeated customer/user feedback
    - Design Patterns: capture design experience as solutions to problems in context
    - Open Source Resources: community-driven components, frameworks, and systems available for free
    - Tools for Interaction Design: support creativity, sketching, simulation, brainstorming, library search, mind mapping, video capture, and prototyping
2. Agile UX
    - Integrates interaction design and agile methods, balancing research and reflection with rapid iterations
    - Requires careful planning of when and how much to use UX techniques
    - Focus on product as deliverable, cross-functional teams
    - Three practical areas: user research, aligning work practices, and documentation
3. User Research in Agile UX
    - Characterizes users, tasks, and context through data gathering and analysis
    - Detailed research may not fit within a limited time box; can be performed in iteration 0 or as an ongoing program
4. Aligning Work Practices in Agile UX
    - Parallel tracks approach: create product vision before development, design work one iteration ahead of development
    - Advantages: no wasted design time, timely feedback, flexibility to handle problems
5. Documentation in Agile UX
    - Agile encourages minimal documentation; use only where needed
    - Consider time spent, users, minimum customer needs, sign-off process efficiency, duplication, and required polish
6. Practical Issues in Interaction Design
    - Identifying users/stakeholders: consider distinct user types and stakeholder groups
    - Determining users' needs: explore problem space, investigate users and activities, try ideas, focus on goals
    - Generating alternatives: research, synthesis, cross-fertilization, user input, product evolution, inspiration, balancing constraints
    - Choosing among alternatives: technical feasibility, user/peer evaluation, A/B testing, quality thresholds
    - Integrating interaction design with other models: careful planning, promising integration with agile development
7. Discovering Requirements
    - Purpose: explore problem space, establish description of what will be developed
    - Capturing requirements: prototypes, operational product, structured or rigorous notations
    - Importance: miscommunication occurs most commonly during requirements activity
8. Different Types of Requirements
    - Functional, data, environment or context of use (physical, social, organizational, technical), user (characteristics, usability goals, user experience goals)
    - Usable security: balancing robust security with user experience to prevent circumvention
9. Data Gathering for Requirements
    - Interviews, observation, questionnaires, studying documentation, researching similar products
    - Combining methods is highly productive (e.g., direct/indirect observation, interviews, diaries, surveys)

    - Probes engage users: cultural, design, technology, provocative
    - Contextual Inquiry: one-on-one field interviews guided by "cool concepts" and contextual design models
    - Brainstorming for innovation: diverse participants, catalysts, capturing ideas, sharpening focus, fun

10. Bringing Requirements to Life

    - Personas: rich descriptions of typical users synthesized from research, relevant to product, aid design decisions
    - Scenarios: informal narrative stories, simple, personal, not generalizable, may be textual, animated, audio, or video
    - Personas and scenarios together augment basic requirements and bring them to life

Lesson 09 - Data Visualization and Interaction Design

1. Introduction to Visualization

    - Visualization is a mental process of transforming information into a visual form
    - Enables users to observe information and perceive hidden features needed for data analysis and exploration
    - Scope of visualization is diverse and intersects with various areas of computer science research and development

2. Applications of Visualization

    - Laser-guided surgery, entertainment, teaching, design & modeling
    - Visualization offers a method for seeing the unseen and fostering profound insights

3. The Birth of Information Visualization (InfoVis)

    - Use of computer-supported interactive visual representation of abstract data to amplify cognition
    - Compact graphical presentation of large datasets enabling users to make discoveries, decisions, and explanations
    - Relates to patterns, groups, and individual items

4. Historical Examples of InfoVis

    - Minard's Map (1812-1813): Depicts Napoleon's Russian campaign, conveying multiple variables in a limited space
    - The Tube Map (1932-present): Evolution of the London Underground map, prioritizing clarity over geographical accuracy
    - Dr. John Snow's Map (1854): Plotted locations of cholera deaths, leading to the identification of the Broad Street Pump as the source

5. Shneiderman's Mantra for InfoVis Design

    - Overview, zoom, filter, details-on-demand
    - Informal design code for creating effective visualizations
    - Additional considerations: relate, history, extract

6. Data Types in InfoVis

    - Qualitative: nominal (no natural order) and ordinal (natural order)
    - Quantitative: discrete (integers) and continuous (floats, singles, doubles)
    - Examples: 1D linear, 2D map, 3D world, multi-dimensional, temporal, tree, network
    - Different data types require different considerations when representing

7. Examples and Resources

    - GapMinder by Hans Rosling: Interactive visualization of global development data
    - Browsing HCIL (Human-Computer Interaction Lab) for various visualization projects
    - Browsing Visual Complexity for a wide range of complex data visualizations
    - Browsing Data to Viz for a comprehensive guide on creating meaningful visualizations

Lesson 10 - OpenGL Introduction

1. Introduction to Computer Graphics

    - Creation and manipulation of graphic images by means of a computer
    - Started as a technique to enhance the display of information generated by a computer
    - Major areas: modeling, rendering, animation, user interaction, virtual reality, visualization, image processing, 3D scanning, computational photography

2. Overview of Graphics Systems

    - Images: two-dimensional shapes with X and Y axes
    - Hardware: input, computation, and output stages
    - Representing objects: vertex method (define objects as a set of points with connectivity information)
    - Interactive computer graphics (ICG): provides two-way communication between the computer and the user

3. Graphics Pipeline
    - Series of interconnected stages through which data and commands related to a scene go through during the rendering process
    - Stages: world coordinate system, projection onto a view plane, clipping planes, window to viewport transformation, logical screen coordinates to physical device coordinates
4. Input Devices
    - Locator devices: mouse, trackball, joystick, tablet, virtual reality trackers, data gloves, digitizers
    - Keyboard: text input, list boxes, GUI, CAD/CAM, modeling
    - Scanners: image scanners (flatbed), laser scanners (Deltasphere), camera-based scanners
    - Other devices: light pens, voice systems, touch panels, camera/vision-based
5. Computation Stage
    - Transformations: converting the model to a form suitable for output
    - Rasterization: converting the transformed model into a framebuffer
    - Framebuffer: a block of memory dedicated to graphics output that holds the contents of what will be displayed
    - Bit depth: number of bits allocated per pixel in a buffer (e.g., 16 bits, 32 bits)
    - Dithering: trading spatial resolution for intensity and color depth to increase the number of apparent colors
6. Output Devices
    - Hardcopy: printers (dot matrix, inkjet, laser) and pen plotters
    - Display: vector and raster scan
    - Cathode Ray Tubes (CRTs): electron gun firing at a phosphor-coated screen
    - Liquid Crystal Displays (LCDs): cells that either allow light to flow through or block it
    - Projection Displays: use bright CRT or LCD screens to generate an image sent through an optical system to focus on a large screen
    - Displays in Virtual Reality: head-mounted displays (HMDs) and head-tracked displays (HTDs)
7. Graphics Software
    - Special-purpose software: Excel, AutoCAD, medical visualization
    - Programming API: provides a way to communicate with the hardware

Lesson 11 - Shaders

1. Introduction to OpenGL
    - OpenGL is a cross-platform graphics API developed by the Khronos Group
    - Set of specifications implemented by graphics card manufacturers
    - Users should keep their graphics drivers updated
2. OpenGL Core vs. Compatibility
    - Old OpenGL (fixed function pipeline): less control, most things hidden from the programmer, inefficient
    - New OpenGL (version 3.3 or higher): split into Core (modern version with more flexibility and control) and Compatibility (to maintain legacy)
3. OpenGL State Machine
    - OpenGL is a large state machine with a collection of variables defining how it should operate
    - Context: the state of OpenGL, changed by state-changing functions
    - Objects: a collection of objects representing a subset of OpenGL's state
    - State-using functions: perform operations based on the current state of OpenGL
4. Drawing Primitives in OpenGL
    - GL_POINTS, GL_LINES, GL_LINE_STRIP, GL_LINE_LOOP
    - GL_TRIANGLES, GL_TRIANGLE_STRIP, GL_TRIANGLE_FAN
5. OpenGL Shading Language (GLSL)
    - GLSL is a mini-program, also known as a shader program
    - Enables the programmer to harness the power of the Graphics Processing Unit (GPU)
    - Fundamental and integral to OpenGL
6. GLSL Common Data Types
    - Built-in Simple Types: float, double, bool, int, uint
    - Vectors: vec2, vec3, vec4 (floats); dvec2, dvec3, dvec4 (doubles); bvec2, bvec3, bvec4 (bools); ivec2, ivec3, ivec4 (ints); uvec2, uvec3, uvec4 (unsigned ints)

  - Square Matrices: mat2, mat3, mat4 (floats); dmat2, dmat3, dmat4 (doubles)
  - Non-Square Matrices: mat(2,3,4) x (2,3,4) (floats and doubles)

7. Simplified Graphics Pipeline
   - Transformation Process: transforming coordinates from 3D to 2D (CPU, Vertex Shader, Primitive Assembly, Geometry Shader, Rasterization, clipping)
   - Coloring Process: applying color and presenting on the screen (Fragment Shader, Tests and Blending, Display)
8. Useful Definitions
   - Vertex: a collection of data per 3D coordinate
   - Vertex Data: a collection of vertices
   - Vertex Attributes: vertex position, vertex color, etc.
   - Uniforms: global variables seen by all shaders, coming directly from the CPU
   - layout (location = 0): specifies the vertex attribute location
   - Vertex Shader: transforms 3D coordinates and allows basic processing on vertex attributes
   - Fragment Shader: calculates the final color of a pixel, where most advanced OpenGL effects occur
9. Useful Links for Learning JOGL (Java OpenGL)
   - https://jogamp.org/wiki/index.php/Jogl_Tutorial
   - https://www.tutorialspoint.com/jogl/index.htm
   - https://www.javatpoint.com/jogl-introduction

Lesson 12 - Two-Dimensional Images

1. Two Dimensional Images
   - Images in this class are two-dimensional shapes with X (horizontal) and Y (vertical) axes
   - The origin (0, 0) is typically located at the top-left or bottom-left corner
2. Hardware Pipeline
   - Input, Computation, and Output stages
   - Describing a rectangle to a computer using a model (object description that a computer understands)
3. Representing Objects
   - Most common method is the vertex method: define the object as a set of points (vertices) with connectivity information
   - Connectivity is important because it defines which vertices are connected via edges
4. Interactive Computer Graphics (ICG)
   - Provides two-way communication between the computer and the user
   - Three major components: Frame Buffer, T.V. Monitor, and Display Controller
5. Graphics Pipeline
   - A series of interconnected stages through which data and commands related to a scene go through during the rendering process
   - Transformation of view plane coordinates to physical device coordinates involves window-to-viewport transformation and logical screen coordinates to physical device coordinates transformation
6. Computation Stage
   - Transformations: converting the model to a form suitable for output
   - Rasterization: converting the transformed model into a framebuffer
7. Framebuffer
   - A block of memory dedicated to graphics output that holds the contents of what will be displayed
   - Consists of pixels (one element of the framebuffer)
   - The size of the framebuffer and the amount of memory allocated depends on the desired resolution and bit depth
8. Bit Depths and Memory
   - More bits per pixel allow for more colors to be stored at each pixel
   - Common bit depths: 16 bits (high color) and 32 bits (true color) per pixel
   - The amount of memory required depends on the resolution and bit depth (e.g., 640 x 480 x 32 bits = 1,228,800 bytes)
9. Dithering
   - Trading spatial resolution for intensity and color depth to increase the number of apparent colors

    - Works by relying on the human eye's tendency to blend areas of high frequency
    - Used when there are not enough bits to represent the desired number of colors

10. Output

    - Hardcopy: printers (dot matrix, inkjet, laser) and pen plotters
    - Display: vector and raster scan
    - The framebuffer contents are converted from digital to analog signals and sent to the monitor by the video card's RAMDAC

11. Raster vs. Vector Images

    - Raster images are compilations of individual colored pixels, with higher resolution allowing for larger resizing without blurriness or jagged edges
    - Vector images are best suited for projects requiring scalable graphics, such as printed projects and marketing images or logos, as they maintain crisp, clear lines when resized

Lesson 13 - 2D Graphics Algorithms

1. 2D Graphics Algorithms

    - Line Drawing Algorithms: DDA Algorithm, Midpoint Algorithm, Bresenham's Algorithm
    - Circle Drawing Algorithms: Midpoint Circle Algorithm
    - Antialiasing
    - Fill-Area Algorithms

2. Line Drawing

    - Fundamental to computer graphics, requiring fast and efficient functions
    - Rasterization Problem: Computing intermediate pixels to closely approximate the ideal line

3. Analytical Method for Line Drawing

    - Uses the equation of a line (y = mx + c) to compute points based on the previous point
    - Incremental algorithms assume the next pixel is on the next column (for small slopes) or row (for large slopes)

4. Digital Differential Analyzer (DDA) Algorithm

    - Incremental algorithm assuming slope is less than 1 (increment along x)
    - Given a point (x_k, y_k) on a line, the next point is given by x_k+1 = x_k + 1 and y_k+1 = y_k + m

5. Midpoint Algorithm

    - Incremental algorithm assuming slope is less than 1
    - Determines the next pixel based on the position of the midpoint between two candidate pixels
    - Decision criteria based on the sign of F(MP) = F(x_k + 1, y_k + ½)

6. Midpoint Circle Drawing Algorithm

    - Determines the closest pixel position to the specified circle path at each step
    - Calculates pixel positions around a circle path centered at the origin, then moves them to the proper screen position
    - Uses the circle function f_circle(x, y) = x^2 + y^2 - r^2 to determine if a point is inside, outside, or on the circle boundary

7. Antialiasing

    - Technique used to diminish jagged edges of an image or line by changing pixels around the edges to intermediate colors or grayscales
    - Can be enabled in OpenGL using the GL_LINE_SMOOTH option

8. Fill-Area Algorithms

    - Used to fill the interior of a polygonal shape by identifying interior points given the polygon boundary
    - Basic Filling Algorithm: Commonly used in interactive graphics packages, requiring a user-specified interior point
    - Types of Basic Filling Algorithms:
        - Boundary Fill Algorithm: For filling a region with a single boundary color
        - Flood Fill Algorithm: For filling a region with multiple boundary colors

Lesson 14 - 2D Transformations

1. 2D Transformations

    - Allow viewing objects from different angles, enlarging or reducing the scale or shape

- Each transformation is a single entity, denoted by a unique name or symbol
- Transformations can be combined through concatenation

2. Types of 2D Transformations
    - Rigid Body Transformations: do not change the object's shape
        - Translate: repositioning an object along a straight-line path
        - Scale: altering the size of an object about a fixed point
        - Rotate: repositioning an object along a circular path, requiring an angle and a pivot point
3. Representing Transformations
    - Vertices represented as (x, y) or as a matrix/vector
    - Transformations can be represented using matrices
    - Translation: P' = P + T, where T is the translation matrix
    - Scaling: P' = S * P, where S is the scaling matrix
    - Rotation: P' = R(θ) * P, where R(θ) is the rotation matrix
4. Combining Transformations
    - Transformations can be combined by multiplying their matrices
    - The order of transformations matters, as matrix multiplication is not commutative
5. Homogeneous Coordinates
    - Increase the dimensionality of the problem to transform the addition component of translation into multiplication
    - By expressing transformations with homogeneous equations and coordinates, all transformations can be expressed as matrix multiplications
6. Coordinate Systems
    - Object Coordinates: local coordinate system for each object
    - World Coordinates: global coordinate system
    - Screen Coordinates: coordinate system of the display device
    - Transformations can be applied hierarchically to convert between coordinate systems
7. 3D Geometry
    - Three-dimensional system with x, y, and z axes
    - Right-handed and left-handed coordinate systems
8. 3D Transformations
    - Translation: movement of an object from one position to another using translation vectors (Tx, Ty, Tz)
    - Scaling: changing the size of an object using scaling factors (Sx, Sy, Sz)
        - Uniform scaling: all scaling factors are equal
        - Differential scaling: scaling factors are different
    - Rotation: moving an object about an angle and an axis (x, y, or z)
        - Rotation matrices for each axis: Rx(θ), Ry(θ), Rz(θ)

Lesson 15 - Image Processing Techniques

1. Simplified Graphics Pipeline
    - Vertex shader takes input variables and outputs variables to the fragment shader
    - Fragment shader takes input variables and outputs variables to the frame buffer
    - Vertex attributes, uniform variables, and interpolation connect the shaders
2. Vectors
    - Magnitude: length of a vector, calculated using the Pythagorean theorem ($\sqrt{x^2 + y^2}$)
    - Normalizing: creating a unit vector with a magnitude of 1, done by dividing the vector by its magnitude
    - Dot product: multiplication of two vectors resulting in a scalar, a · b = |a| · |b| · cos(θ)
    - Cross product: multiplication of two vectors resulting in a third vector perpendicular to the original vectors, a x b = |a| * |b| * sin(θ) * n
3. Shading Models and Components
    - Ambient: represents light that illuminates all surfaces equally and reflects equally in all directions
        - Ia = Ka · La, where Ka is surface reflectivity and La is light source intensity

- Diffuse: models a surface that exhibits purely diffuse reflection (scatters light in all directions equally)
  - L = Kd · Ld · (s · n · cosθ), where Kd is diffuse reflectivity, Ld is light source intensity, and s.n is the dot product between the light direction and normal vector
- Specular: models the shininess of the surface and represents glossy reflection around a preferred direction
  - Is = Ks · Ls · $(r.\, v)$^f, where Ks is specular reflectivity, Ls is light intensity, r is the reflection vector, v is the viewing vector, and f is the power coefficient

4. Phong Reflection Model
   - Combines ambient, diffuse, and specular components
   - I = Ia + Id + Is = Ka . La + Kd . Ld . (s.n) + Ks · Ls · $(r.\, v)$^f
5. Per-vertex vs. Per-fragment Shading
   - Per-vertex (Gouraud) shading: calculations done within the vertex shader, can lead to warped or lost specular highlights
   - Per-fragment shading: calculations applied to every pixel, more realistic but more expensive
6. Attenuation and Directional Light
   - Attenuation: modeling the fall-off of light intensity with distance, done by dividing the light intensity by the inverse square of the distance or using a less steep curve
   - Directional light: has only direction, no position, avoiding the need to calculate light direction in the shader
7. Useful Resources
   - Coordinate systems, transformations, GLSL basics, and linear algebra tutorials are provided as additional learning resources

Lesson 16 - Lighting and Shading

1. Main Types of Lights
   - Directional: similar to sunlight, only light direction matters, position is irrelevant
   - Point: behaves like a light bulb, has a limited range, position is crucial for calculating light direction
   - Spotlight: similar to a real spotlight, position and orientation are important
2. Directional Light
   - Light rays are parallel, intensity stays constant, no fall-off
   - Light direction can be sent directly as a uniform to avoid calculations in the shader
   - Use separate shaders for directional and point lights to take advantage of parallel processing on the GPU
3. Point Light
   - Uses the Phong reflection model: I = Ia + Id + Is
   - Ambient (Ia), Diffuse (Id), and Specular (Is) components are calculated and summed
4. Multiple Point Lights
   - Evaluate the reflection model for each light and sum the results to determine total light intensity
   - Use uniform arrays to store position and intensity of each light
   - Optimize code by using one light intensity for diffuse and specular components
5. Per-Fragment Shading (Phong Shading or Phong Interpolation)
   - Improves the look by doing all calculations in the fragment shader
   - Interpolates position and normal vector, calculates color for each fragment
   - More expensive than per-vertex shading (Gouraud shading) but provides better results, especially for specular highlights
6. Blinn-Phong Reflection Model
   - Calculates a half vector (h) instead of the reflection vector (r)
   - Replaces the dot product of r and v with the dot product of h and n
   - More efficient than the original Phong model
7. Spotlight
   - Light radiates within a cone with the apex located at the light source
   - Light is maximal along the axis of the cone and decreases toward the outside edges
   - Creates similar visual effects to a real spotlight
8. Toon Shading (Cel Shading)

    - Non-photorealistic rendering technique that mimics the style of hand-drawn animation
    - Large areas of constant color with sharp transitions between them
    - Locks the value of the dot product calculated for diffuse to a fixed number of possible values

9. Fog Simulation

    - Achieved by mixing the color of each fragment with a constant fog color
    - Amount of "fog color" applied is determined by the distance from the camera
    - Linear or non-linear interpolation can be used
    - Fog factor (f) is calculated using the formula: f = (dmax - |z|) / (dmax - dmin)

Lesson 17 - Image Processing Techniques

1. Introduction

- Image processing uses fragment shaders and rendering to textures
- Works directly with pixels, typically involves multiple passes
- Framebuffer: portion of RAM containing bitmap driving video display

2. Edge Detection Filter

- Identifies regions with significant brightness changes (edges)
- Uses convolution filters (e.g., Sobel operator with 3x3 kernels)
- Edge magnitude: g = sqrt(sx^2 + sy^2), compared to threshold

3. Gaussian Blur Filter

- Mixes pixel's color with nearby pixels using weighted sum
- Uses 2D Gaussian function for weights
- Decomposed into 1D horizontal and vertical passes for performance

4. HDR Lighting

- High Dynamic Range (HDR) rendering: larger dynamic range
- Tone mapping compresses range using Tone Mapping Operator (TMO)
- Local TMOs: use current and nearby pixels; Global TMOs: entire image
- Log-average luminance used in simple global TMO
- Color space conversions (RGB, CIE XYZ, CIE xyY) using matrices

5. Bloom Effect

- Bright parts bleed into darker areas, simulating airy disc effect
- Implemented in five passes:

    1. Render scene to HDR texture
    2. Extract bright parts (bright-pass filter)
    3. Apply Gaussian blur to bright parts (two passes)
    4. Apply tone mapping
    5. Add tone-mapped result to blurred bright-pass results

6. Useful Links

- Additional resources on HDR, bloom, image processing, colorimetry, gamma correction
-