Potential Exam Topics

• As Univ. of California students, you are expected to be able to assess complex material and make judgments concerning its relative importance.

• That said, it can be helpful to have some input from the Professor to help focus studying activity.

• The following are questions/topics that are likely, but not guaranteed to appear on the exam.

• Anything covered in class or in the assigned readings may appear, even if not explicitly mentioned today.
Potential Exam Questions

- C# language
  - Basic syntax
  - Foreach
  - Properties
  - Delegates
  - Lists
  - Arrays
  - Visibility rules
  - Enumerations
  - Interfaces and Abstract classes
    - Know how to use these, and difference between these
    - Abstract methods
  - Be prepared to read and understand C# code examples, and to write short segments of C# code
Potential Exam Questions

• Object-oriented design
  – Classes
  – Interfaces
  – Design patterns

• Debugging in Visual Studio
  – Syntax errors
  – Logical errors
XNA

• What is a spritebatch?
• What is a clock tick? How long is the default clock tick in XNA?
• What is the difference between Update() and Draw() methods?
• Be able to understand code that receives input from an Xbox controller
• How is different types of content included, loaded, and
  – Textures
  – Sounds
Sprites

- Parameters
  - Look
  - Scale etc.

- Behaviors
  - Movement along fixed path
  - Movement on input

- Be able to understand and write code for various operations on sprites
Menus

• Delegates and Events
• Menu system design
  – Data structures
  – Interaction
Particle Systems

- Engine
  - Emitter
  - Rate
  - Number of particles
  - Recycling
- Particle
  - Lifetime
  - Movement
- Be able to understand code for particle systems and know how it will work
- Be able to write code for particle effects
Design Patterns

• Design patterns
  – Know the design patterns covered in class
    • Strategy, Observer
    • Need to memorize these, and be able to reproduce them
    • Need to understand how they work
    • Given an example of one these patterns, be able to identify the pattern
  – Know what these mean:
    • Favor composition over inheritance
    • Write to an interface, not an implementation
  – Go over slides and code examples
    • Recall discussion in class from your notes
• Collision Detection
  – Bounding volumes
  – 2D collision detection
Subversion

• Operations
  – Checkout
  – Commit
  – Tags
  – Branches

• Benefits
File I/O

• Why?
  – Save game state
  – Player profiles
  – Initialization settings
  – Levels and add-ons

• Two types
  – Text files
  – XML files
Types

• Text files
  – Easy to handle and edit
  – Must follow strict formatting

• XML files
  – Easy if you know how XML processing libraries work
  – Harder to edit (usually edited through interface)
  – Wrapping data in tags makes it easy to verify
  – No need to follow strict formatting or order
C# file handling

• Example
  – Text
  – XML
CMPS 20: Game Design Experience

HUD
Heads-Up Display (HUD)

- Information about the game world
- Aim (Crosshair)
- Status of the player
- Inventory etc.
- Feedback
  - Control
  - Damage
  - ...
- Orientation/Location (mini map, next target pointer)
### Spatial Representation in 3D Game Space

<table>
<thead>
<tr>
<th>Fictional</th>
<th>Spatial</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

### Venn Diagram

- **Geometry**
- **Fiction**
- **Rules**

The diagram illustrates the relationship between spatial representation and fictional elements within a 3D game space.
Design

• Type of information
  • Complex / Minimal
• Direct / Clickable
• Fixed / Zoomable
• Location
  – Periphery
  – Center
Examples
Examples
Examples
Examples
Select this option to access multiplayer.
Examples
Code Structure and Elements

- **Sprites** (text and images)
- **Overlays** (crosshair, maps, health, etc.)
- **Functions**
  - Draw
  - Update
- **Example**: Unreal Tournament