Slide 1

Administrivia

- Quiz solution available on Web.
- Info about tools coming soon.
- If your prox card doesn't let you into the labs, talk to the people in the Tiger Card Office. Supposedly we have told them who should have access!

Slide 2

Game Elements

- Playing field — "screens" (two-dimensional grids of "blocks" — can have many kinds).
- "Player" character controlled by user input.
- Other "game entities" controlled by program.
- Menu bar; "game status panels" showing other info (e.g., score).
- Player and other entities each have "location" in terms of screen and coordinates within screen; coordinates based on "graphics convention".
- Screen maintains a list of entities on that screen.
- Global "priority queue" of all entities.

CSCI 1321 September 23, 2003

Slide 3

Game Framework

- What the game framework does at each "game tick":
  - Draws screen where player is, using methods of screen, block, and game entity classes (including player).
  - Updates each game entity that wants to be updated, including player. Entities can move (change location) or do other things based on surroundings — types of blocks, proximity to other entities, etc. Player can also respond to keyboard or mouse input.

Slide 4

Overview of Homeworks

- Homework 2 — set up "playing field" (screen and block classes, game setup). Replace `BasicScreen` and `BasicBlock` with your classes.
- Homework 3 — start defining player, how it responds to user input and interacts with blocks. Replace `BasicPlayer` with your class.
- Homework 4 — start defining other entities, how they interact with player.
- Homework 5 — continue defining other entities, how they move and interact with blocks.
- Homework 6 — define something using GUI classes (game status panel(s) or menu items).
- Homework 7 — compare different implementations of key data structure.
- Homework 8 — finish game.

Slide 5

Homework 2

- Modify `GameSetup` — should be relatively straightforward. Can take one of two approaches:
  - Create configuration (screens, blocks, entities) yourself. Good to do for initial debugging.
  - Use "screen editor" to create configuration and save in binary file, read in file. Program uses your screen class.
- Write classes implementing `Block` interface — use `BasicBlock` as a model, but *do not subclass it.* Okay to define your own "master block class" and subclasses.
- Write class implementing `Screen` interface — replacement for `BasicScreen`, *not subclass*.

Slide 6

Defi ning a Class

- What methods do I need? If implementing an interface, you at least need those. May want additional methods.
- What variables do I need to implement the needed methods? e.g., to implement `Screen` interface you probably need a two-dimensional grid (array) of blocks and a list of entities.
- "Good style" tips:
  - Make methods public if needed by code that uses your class, private otherwise.
  - Make variables private unless there's a good reason not to — prevents unwanted/inconsistent access.
  - Use named constants (static final variables) rather than hard-coded values.

    E.g., `private static final screenWidth = 20;`

Slide 7

Arrays in Java

- Arrays are objects — unlike in C/C++, where they're basically pointers.
- Declaring (references to) arrays — denote by putting brackets after type (or use syntax shown in book).
- Creating arrays — use `new`, e.g.,

  ```
  new int[10]
  new String[n]
  ```

  (Remember that the second one only creates *references*.)
- All arrays have `length` variable.
- Otherwise, syntax is same as C/C++; indices start at 0.
- Java runtime does automatic bounds-checking — unlike in C/C++, get ArrayBoundsException rather than random problems.

Slide 8

Multidimensional Arrays

- "Arrays of arrays", e.g.,

  ```
  int[][] x = new int[10][100];
  ```
- For 2D arrays, first index is row, second is column. (Note that this is not the "graphics convention" used in the game.)

Minute Essay

- Write code to define an array of four `String`s and fill it with data of your choice.
- Write code to define a two-by-three array of `int` and set each element to the sum of its row and column.