Domain → Design

Plans are worthless.
Planning is priceless.

(misquote of Eisenhower
-- but what he meant)

Designs are worthless.
Designing is priceless.

(don't quote me!)

Domain to Design classes

Design class should (almost always)
- Represent domain classes
- Refine domain classes
- Add control, creator, and expert classes

Modeling Tools

1. CRC Cards
2. UML Class Diagrams

Class Diagram

- Ball
  - vector velocity
  - void draw()
  - void setCenter(vector p)
  - void setVelocity(vector)

- Sphere
  - vector center
  - void draw()
  - void setCenter(vector p)
Exercise

Construct a UML class diagram for Triangle World.

(30 minutes)

Modeling Tools

1. CRC Cards
2. UML Class Diagrams
3. UML Sequence Diagrams

UML

Static View:
Class Diagram

Dynamic View:
Sequence Diagram, Activity Diagram, etc.

Exercise

Construct a sequence diagram for Triangle World.

(30 minutes)

Design Heuristics
Example

Which did you have?
- Ball
- Sphere
- Triangle
- Polygon
- Thing
- Shape
- Moving shape
- Moving object
- Velocity
- Walls
- Floor
- World
- Force
- Gravity
- Collision
- Position
- Item

Ball vs. Sphere vs. Shape

What is the relationship?

design heuristic

think like an object

Ball vs. Sphere vs. Shape

What is the relationship?

Ball vs. Sphere vs. Shape

What is the relationship?
Ball vs. Moving
What is the relationship?

Shape vs. Moving
What is the relationship?

Ball
velocity

ball doesn't move?
set velocity to 0

Shape vs. Moving
What is the relationship?

Shape vs. Moving
What is the relationship?

Shape vs. Moving
What is the relationship?
Inheritance vs. Composition

- White-box reuse
- Black-box reuse

Design heuristic

- Favor composition over inheritance

Collisions

- Ball
  - Move(dt,triangles)
- Triangle
  - Collide(dt,ball)

Design heuristic

- Class should have high cohesion and low coupling

Cohesion & coupling

- Cohesion: how closely the operations in a class or method are related
- Coupling: the strength of a connection between two classes or methods
**Design heuristics**

- Think like an object
- Favor composition over inheritance
- Low coupling & high cohesion
- Only talk to your immediate friends (LoD-DPIC)

**LoD-DPIC**

- An object should only invoke methods of:
  - objects that are *declared* within it
  - objects that are *parameters of its methods*
  - *itself*
  - objects that it *creates*

**Design heuristics**

- Think like an object
- Favor composition over inheritance
- Low coupling & high cohesion
- Only talk to your immediate friends (LoD-DPIC)
- Derived class should be able to stand in for base class (LSP)

**More design heuristics**

- Objects as organism: Class objects should be responsible for their own behavior
- Use utility classes freely in place of primitives.
- Keep your classes light.
- Favor pointer members over instance members.
- Avoid forgery: do not keep the same data in more than one place.

**XP Philosophy**

*Build only what you need now!*

**My suggestion**

*Build only what you need now but have an evolution plan in mind!*
Exercise

- evaluate your design relative to each design heuristic
- re-design