Object Oriented Programming in Action
Object Oriented Analysis and Design

- Find and define the objects
- Organize the objects
- Describe how the objects interact with one another
- Define the external behavior of the objects
- Define the internal behavior of the objects
Robots Game

- Robots Game
  - Turn-based game used to teach the navigation keys of text editor vi
  - Invented for Unix
  - Still available as a Linux console game
Game Description

- The player controls an avatar in a two-dimensional playing field
  - The player can move left and right, upwards and downwards and also move diagonally
- A number of robots try to kill the player by reaching the same space on which the avatar is
  - Robots only move up and down, left and right, but not diagonally
  - If robots collide with each other, they die and leave behind a heap
  - If the avatar or a robot moves into a heap, they also die
  - The player can use a teleportation device that places the player in a random location in the field, possibly next to a robot or a heap, but not in a heap.
- If all robots are dead, then the player has won the level and advances to a higher level with more robots
Game Design

• Designing something complex is very difficult
  • Can use design patterns
  • We are going to use the Model-View-Controller pattern
    • Model: The data and its business logic
    • View: The window on the screen
    • Controller: The glue between the two
Game Design

- Model is independent of view and controller
  - You can work simply in the model to implement the business logic
- Regardless of visual presentation and user interface
Game Design

- MVC is popular in web development
- Used in many frameworks such as Django, web2py, Pyramid, ...
Game Design

- Model - The logic of the application
  - Model has a state and methods for changing the state such as a player move
  - We should be able to change the controller and the view without changing the logic
Game Design

- **View** - the display of the model
  - View receives data from the model through the controller
  - Responsible for visualization
  - Does not contain complex logic
    - This belongs in the controller and the model
Game Design

- Controller - Glue between Model and View
  - Controller receives data from user requests and sends it to other parts of the system
  - Controller also receives data from the model and passes them on to the View
Game Design

• Design recommendations:
  • Smart Models
  • Thin controllers
  • Dumb viewers
Game Design

- Modeling the Model
  - Look through the description
  - Identify substantives and verbs
- We have actors:
  - Avatar
  - Robots
  - Heap
  - We have the playground
Game Design

- Create cards with the name of the object
- Add notes on what they need to interact with the other parts.

### Playground
- Grid contains robots, heaps, avatars

### Avatar
- Location
- Collision
- Move

### Heap
- Location
- Collision

### Robot
- Location
- Collision
- Move
Game Design

- Because we use location so often, we pull it into its own class.

- We also need to figure out how the View and the Control are interacting with the model.
  - View needs to get coordinates for all entities in order to display them
  - Control needs to steer the avatar.
    - This is done with a direction
      - which we make into another class.
Avatar
  has a
  moves
  collides

Heap
  has a
  collides

Robot
  has a
  moves
  collides

Location
  __equal__
  change

Direction
Game Design

• Design of View
  • View depends on the underlying architecture
  • We are going to rebuild the game using Tkinter graphics soon
  • Currently, the view is based on an Idle shell ASCII art
  • Need to move geometry between Controller and View
  • Need to get things to display from controller
  • Need to move decisions from View to Controller
Game Design

• Design of Control
  • Needs to start the game: Play
  • Needs to transmit the model data to view
  • Needs to obtain User input
Game Design

Command
- run_game()
- clear_screen()
- draw_screen()

View
- clear_screen()
- draw_screen()

Model
- avatar
- robots
- heaps
- update()

Location
- x
- y
- ==
- generate_random_locality()
- update_location()
- get()

Direction
- move_towards()

Geometry
- width
- height
Game Design

• Write all the substantives on little cards
  • You get a design like this one
  • Now you go and expand all of the cards with methods
  • The design will change as you go through it
Game Design
Implementation

• Now it is time to start implementing
  • During implementation, design issues can emerge and force a redesign
  •
Software Engineering Errors

- All implementations and designs will contain errors
  - Design errors are easiest to fix during design
  - Implementation errors are easiest to fix during implementation
- Need a thorough testing phase
Testing

- Need something better than just playing and fix arising issues
- Systematic
  - For unit and for the whole game