**BugsWorld Project**

- The Game
- The Simulator
- The Language
- The Translator

**The Game: It's a BugsWorld!**
The Simulator

One server, multiple clients and displays

Server keeps track of world, processes client requests, resolves conflicts

Client simulates creature behavior for all creatures of one species

Display shows current state of world plus some statistics about the simulation

Each process can run on a different computer (distributed simulation)
**The Language: BL**

- The behavior of each species is determined by a program in BL
- Primitive instructions: move, turnleft, turnright, infect, skip
- Control structures: IF-THEN, IF-THEN-ELSE, WHILE-DO
- Defining new instructions: INSTRUCTION-IS
- Conditions: test whether next cell is empty, friend, enemy, or wall (plus true and random)

**An Example**

```plaintext
PROGRAM TryToGuess IS
  INSTRUCTION FindObstacle IS
    WHILE next-is-empty DO
      move
    END WHILE
  END FindObstacle

BEGIN    # TryToGuess
  WHILE true DO
    FindObstacle
    IF next-is-enemy THEN
      infect
    ELSE
      IF next-is-wall THEN
        turnleft
      ELSE  # next-is-friend
        skip
      END IF
    END IF
  END WHILE
END TryToGuess
```
Language Continued...

- Precise syntax
- Case sensitive
- Matching ENDS
- Comments
- Identifiers
  - start with 'a'..'Z', 'A'..'Z'
  - followed by any of 'a'..'z', 'A'..'Z', '0'..'9', '-'

The Translator

PROGRAM TryToGuess IS
  INSTRUCTION FindObstacle IS
    WHILE next-is-empty DO
      move
    END WHILE
  END FindObstacle
  BEGIN  # TryToGuess
    WHILE true DO
      FindObstacle
      IF next-is-enemy THEN
        infect
      ELSE
        IF next-is-wall THEN
          turnleft
        ELSE # next-is-friend
          skip
        END IF
      END IF
    END WHILE
  END TryToGuess
**Translator Continued...**

![Diagram](image)

**What You Will Do**

- Build the translator
  - Implement abstract program component (Lab #2, Closed Lab #4)
  - Implement parser extension (Lab #3)
  - Implement code generator extension (Lab #4)
  - Implement tokenizer component (Lab #5)

- Complete the client
  - Implement interpreter (Closed Lab #6)