welcome to

GAMELET DESIGN

for education

Objectives

- V& L Awards
- Sokoban reflections
- Hands on: Space Invaders

V& L Awards

- Main criteria
  - Fun/engagement - time spent on playing it
  - No cheating: “press cursor up at the beginning of game. If frog moves all the way up without having any accident the game is bad”

reflections

Cursor control & rule order

- If you want to have responsive controls make sure to have time check (once every) **follow** key check

- If you have key check in long list of rules make sure they precede collision check rule to avoid user cheating
- Ideal: factor out collision and keyboard check into separate methods, e.g., “collide” “keyboard” with collision preceding keyboard

Person, Box, Space interaction

- How did you do it?

Agent-based programming

- ZEN: Be the agent
- ZEN: Delegate if you can
  - Sokoban person: If you cannot determine if the space next to the box that you are trying to push is free have the box figure it out. If the box can move it can tell you to move as well
- Avoid:
  - Undo approaches, e.g, move assuming you may be able to, if you end up on box undo, move back where you came from
  - Independent key checking: person checks for key, box checks for key, both move if they are next to each other: do not rely on dispatch order! Could end up pilling up or separating

Move right, next to box, no obstacle

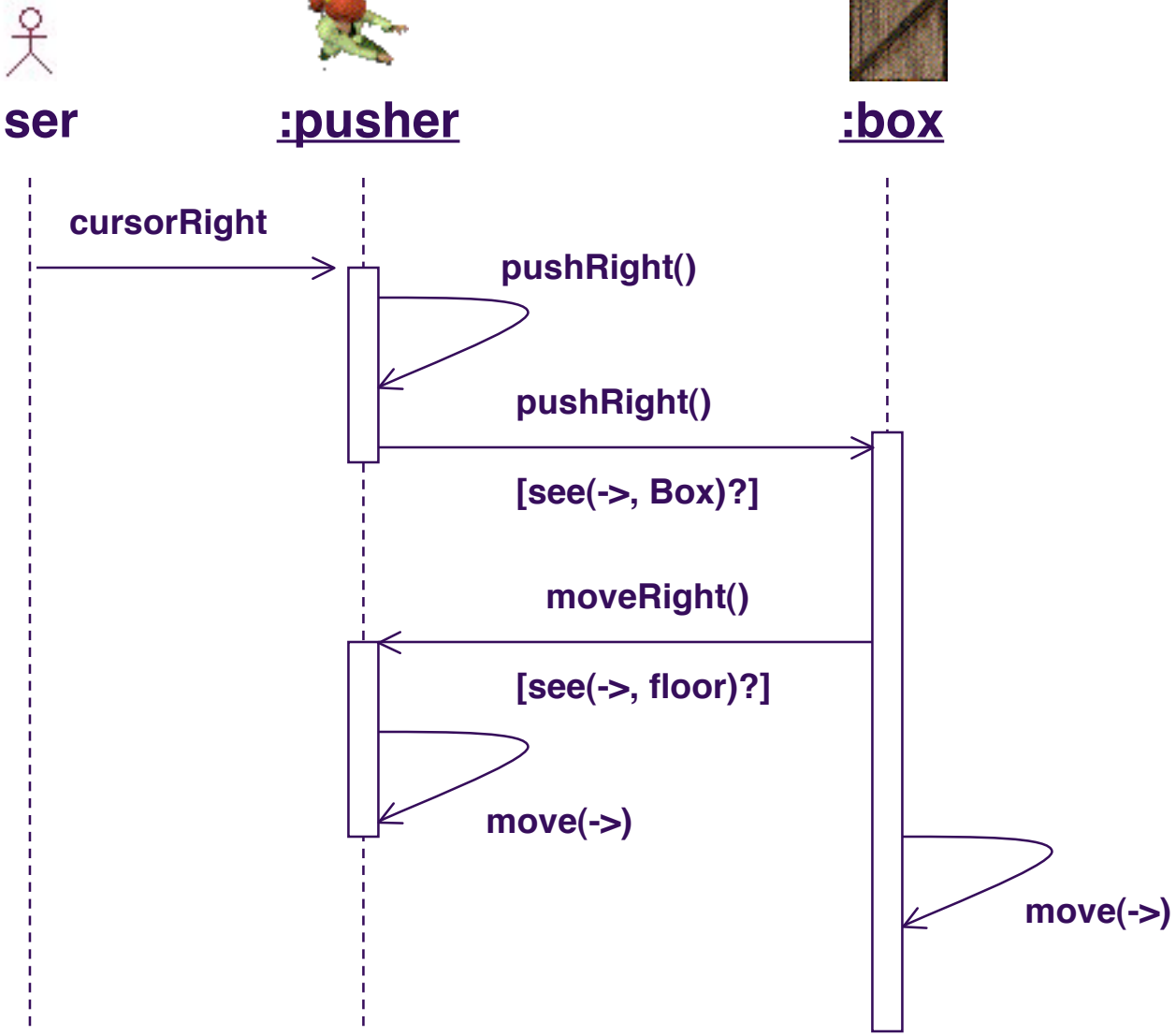
User presses cursor right key

There is a box to the right

There is space to push the box

Move pusher

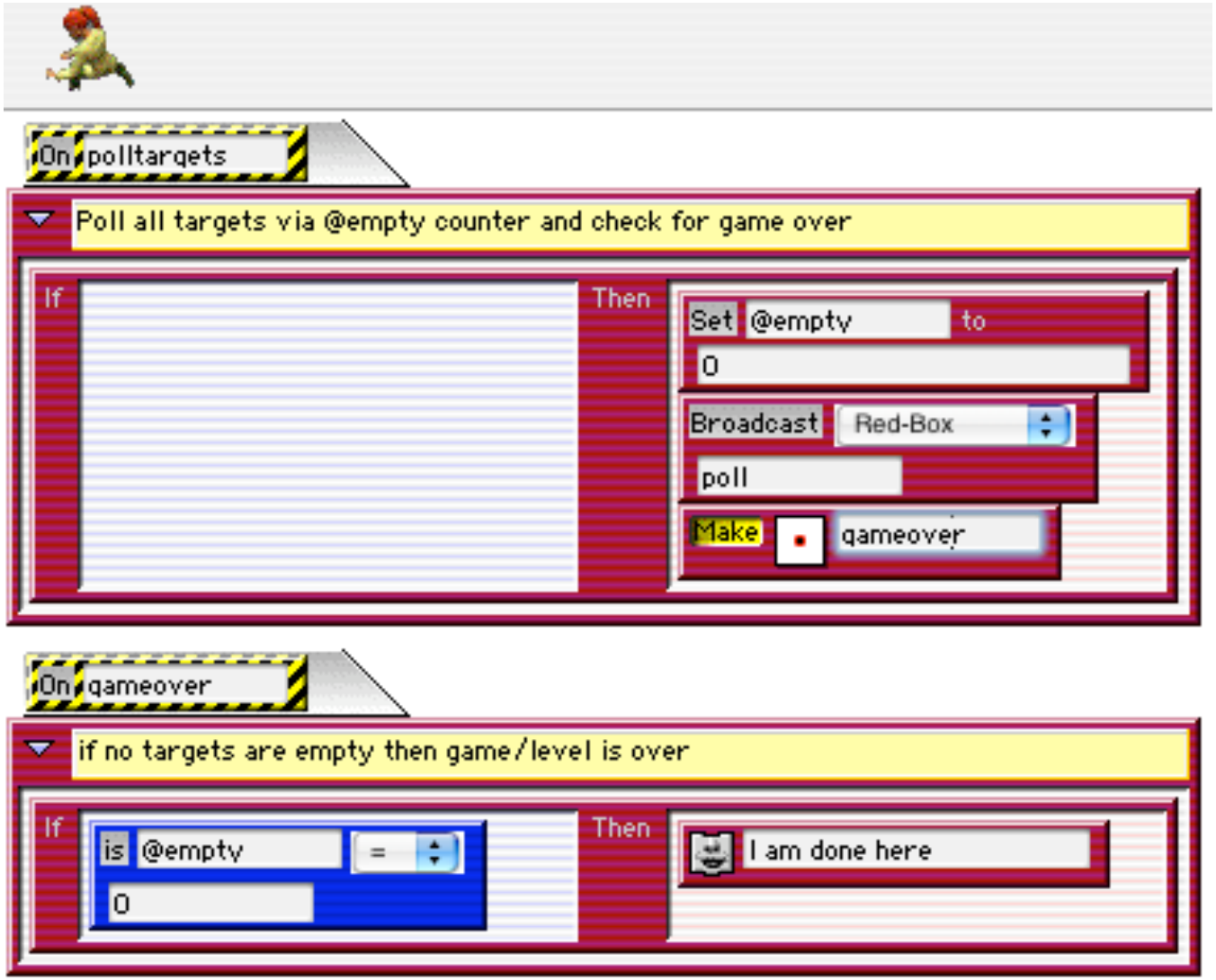
Move box



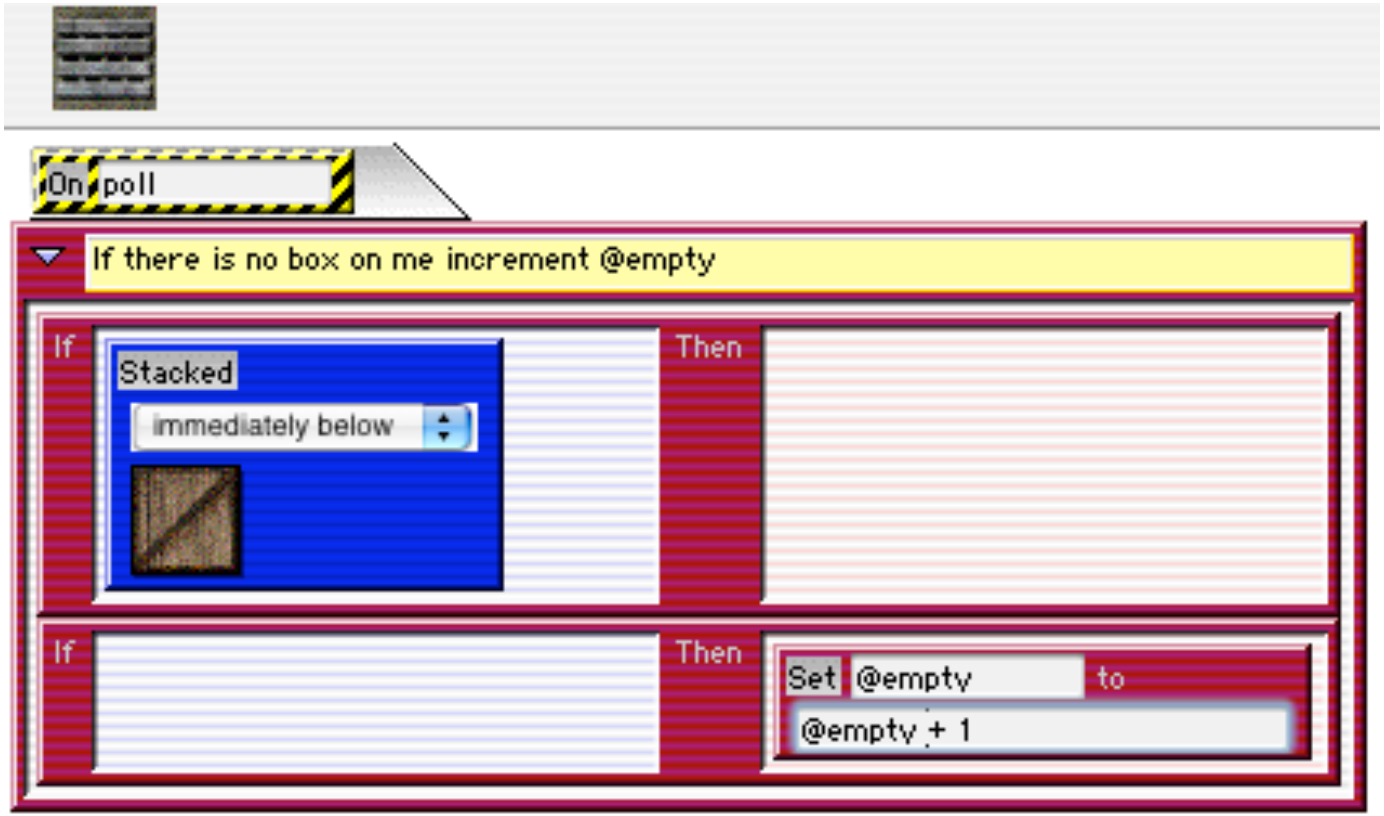
End of level detector

- Need to be able to sense that all targets are covered by boxes
- How did you do it?

Person


On polltargets
Poll all targets via @empty counter and check for game over
If
Then
Set @empty to
0
Broadcast Red-Box
poll
Make
gameover
On gameover
if no targets are empty then game/level is over
If
is @empty =
0
Then
I am done here

Target (red-box)



UML: useful or painful?

Did you make diagram before or after making game?

patterns of agent interactions

- Collide: 1 agent : 1 agent
  - Frogger: truck vs frog
- Transport: 1 agent : 1 agent
  - Transporter move, make transported move
  - Frogger: frog on log
- Push: 1 agent : 1 agent
  - Pusher send message indicating intention to move to pushed, pushed checks if it can move, pushed makes pusher move, pushed moves
  - Sokoban: person pushing box
- Poll: 1 agent : N agents
  - Poller: init var, broadcast poll message to pollees, pollees update var, poller checks var
  - Sokoban: level finished

patterns of agent interactions cont

- Launch: 1 agent : 1 agent
  - Agent creates new agent on top or next to it
  - Frogger: tunnel launches cars and trucks
  - Launch: missiles, lasers, electrons, bullets, …
- Absorb: 1 agent : 1 agent
  - Agent erases agents on top or next to it
  - Frogger: “right” tunnel absorbs cars

NEW pattern of agent interactions

- Script: 1 agent : N agents
  - One agent scripts behavior (e.g, movement) of many other agents
  - Game of live: one agent broadcasts “perceive” and “act” messages to many cell agents to implement parallel update
  - Space Invaders: mother ship scripts left - right, and down movement of fleet spaceships

Homework # 3

Space Invaders

Space Invaders

Space Invaders was an arcade video game designed and programmed by Toshihiro Nishikado and originally manufactured by Taito; it was licensed for production in the U.S. by the Midway Manufacturing division of Bally. Released in 1978, it ranks as one of the most influential video games ever created. Though simplistic by today's standards, it (along with other contemporary games such as *Pac-Man*) was one of the forerunners of modern video gaming.

Flash version:
http://www.thepcmanwebsite.com/media/flash_space_invaders/

http://en.wikipedia.org/wiki/Space_Invaders

Homework

- Due: Feb 6: 11:59pm in GORP
- 100 points
  - Cursor controlled defender
  - Left & right sweeping, gradually descending fleet of space ships
  - Game over when all ships are gone (won) or fleet landed (lost)
  - Defender shoots up, space ships down
  - Lasers, missiles will crumble bunker incrementally
- 20 extra