From Model to Application: Developing a believable opponent in the game of Set!

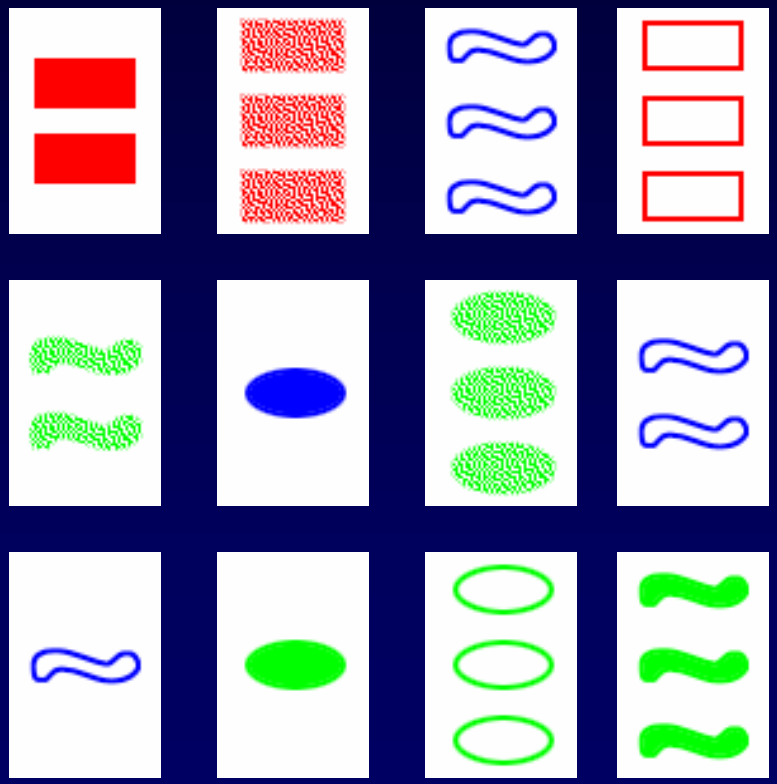
Niels Taatgen, Marcia van Oploo, Jos Braaksma and Jelle Niemantsverdriet

Contents

- The Game
- The Predictions
- The Experiment
- The Model
- The Application
- The Issues

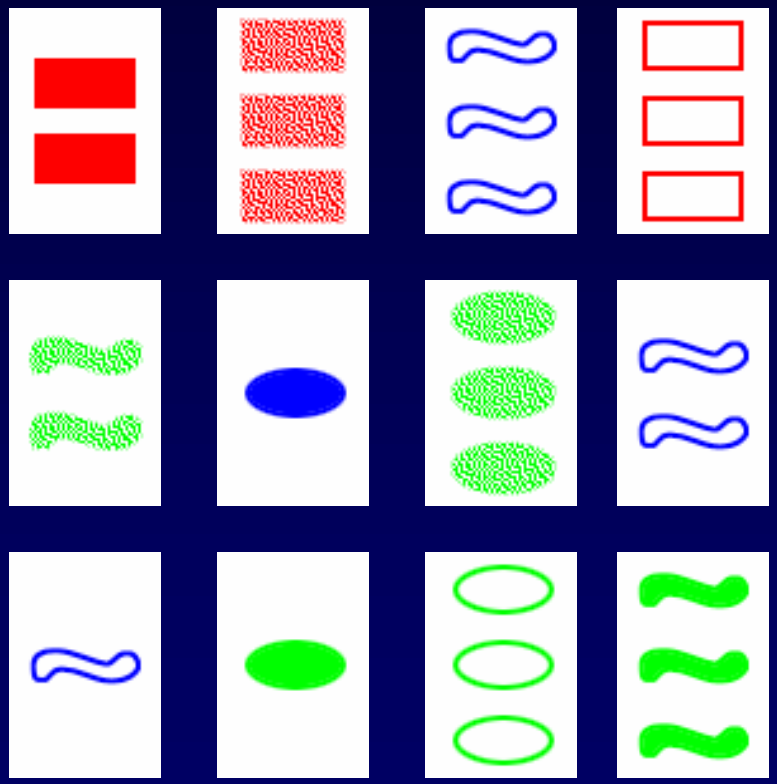
The Game of Set!

- Game consists of 81 cards
- Each card has four attributes: color, shape, filling and number

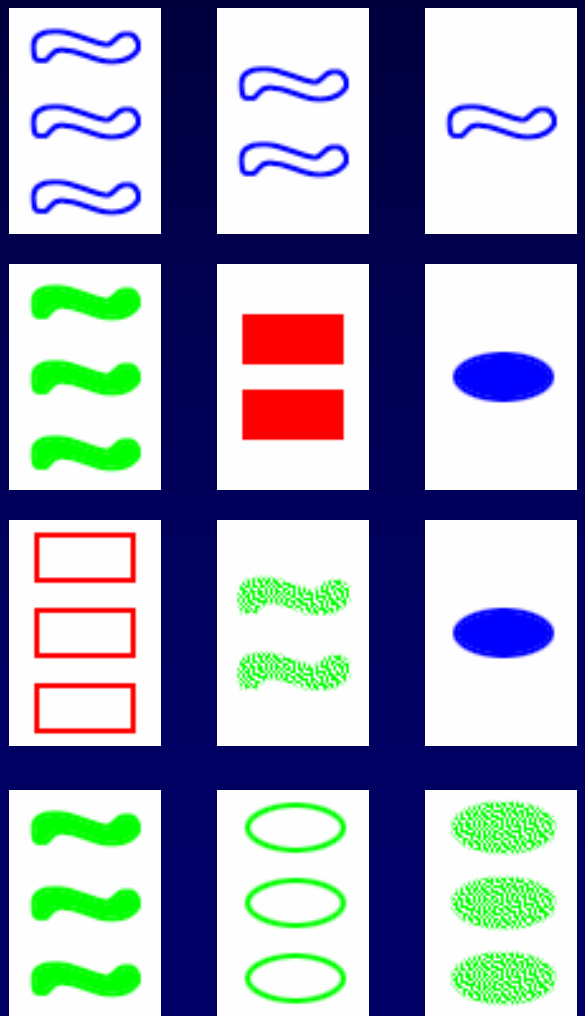


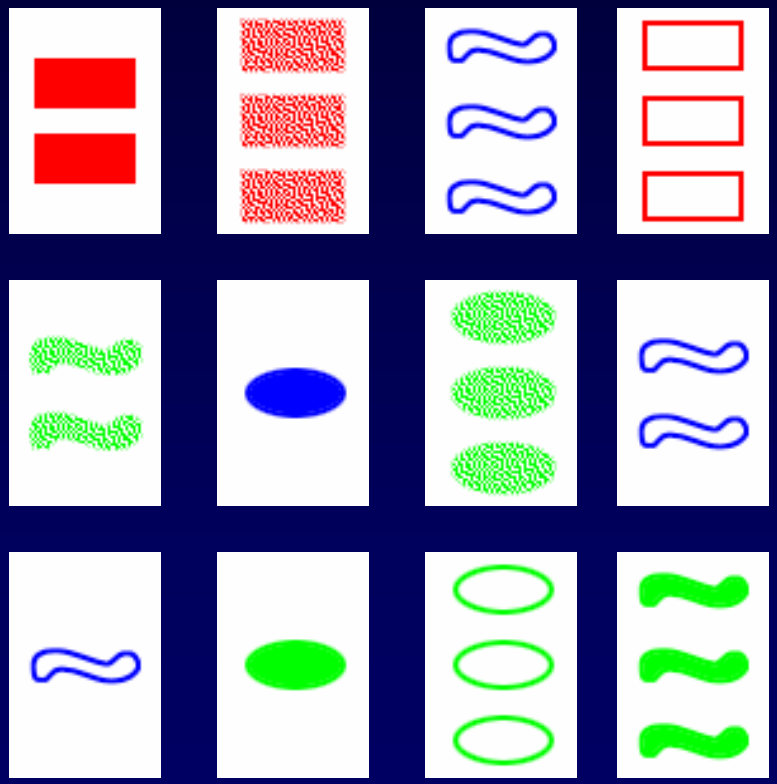
Goal of the Game

- Twelve cards are dealt on the table
- Find a Set: three cards in which for each attribute, the attribute values for each of the cards are all different, or all the same



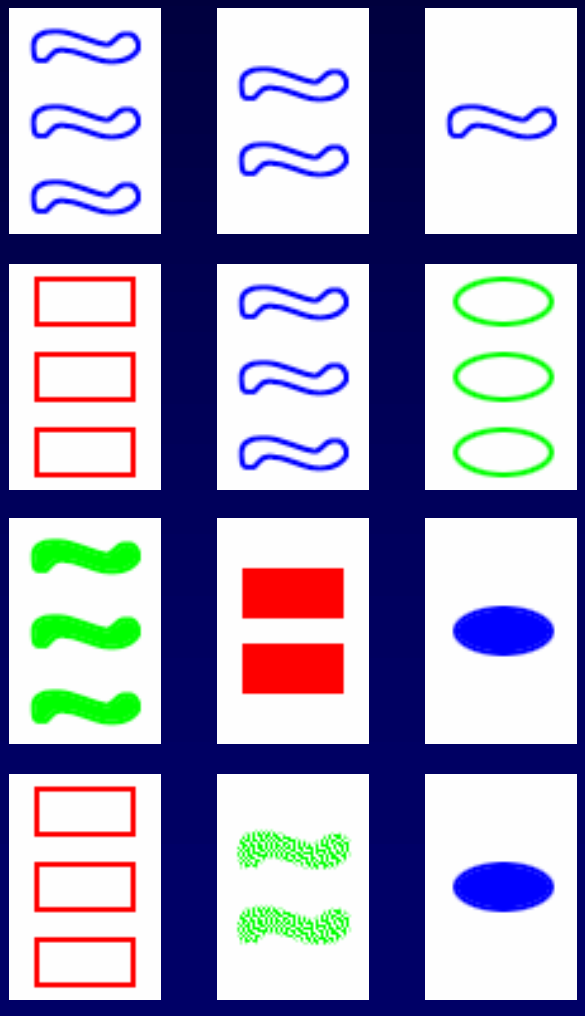
Goal of the Game





Not a Set!

Some Sets are more difficult then others



One attribute different

Two attributes different

Three attributes different

Four attributes different

Set! as a game to play against the computer

- For a computer, the game is trivial (as opposed to chess, etc.)
- The challenge is to program an opponent that acts as a human player
- So the computer opponent has to be fast at sets that people are fast at, and slow at sets that people are slow at

The Predictions

1. The “easy” sets will be found faster than the “hard” sets.
2. Experts on the game will mainly excel in finding the hard sets, and will be approximately equally good as beginners on the easy sets

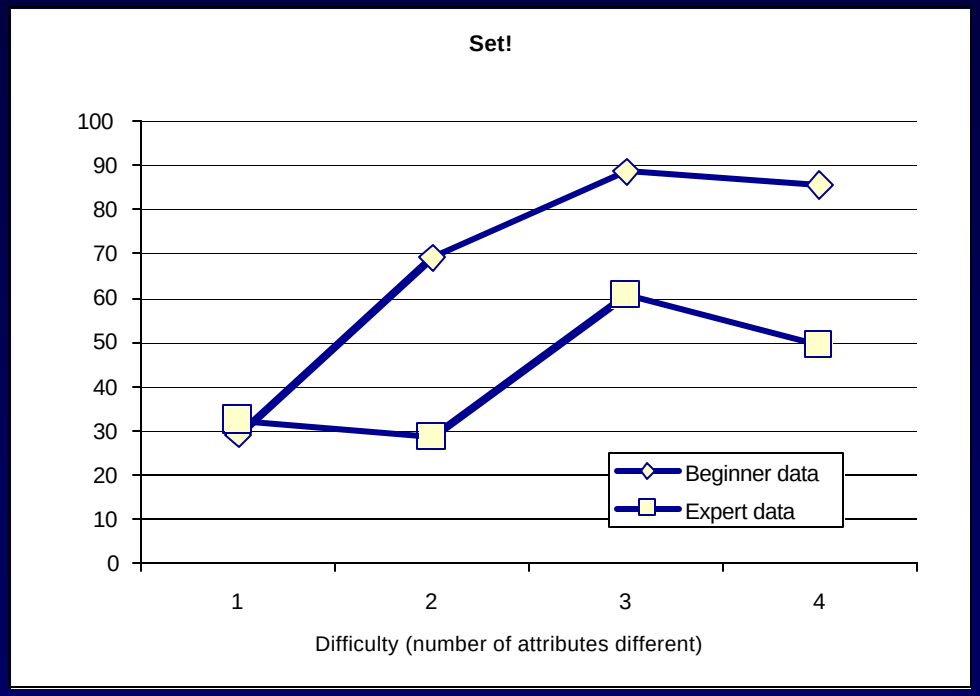
The Experiment

- 8 subjects, 4 (self-proclaimed) beginners, 4 experts
- 20 set-problems (12 cards, find the set as fast as possible)
- 5 problems of each of the four levels of difficulty

Experimental results

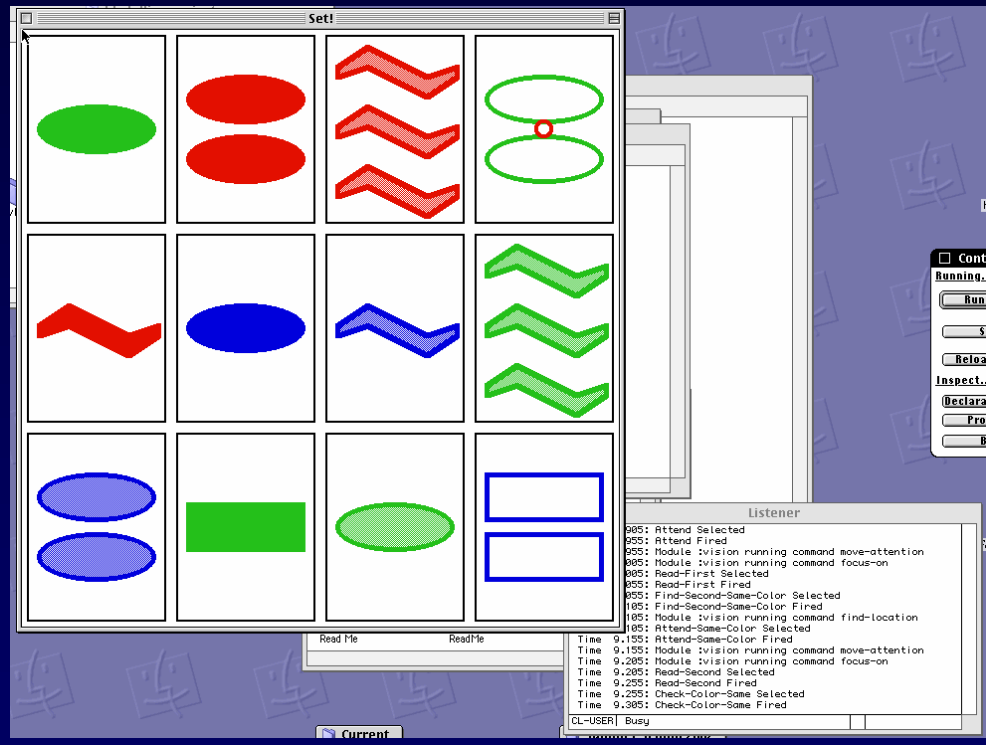
Experiment confirms both hypotheses:

1. Difficult problems take longer
2. Beginners and Experts are equally good at easy problems, but Experts excel on hard problems



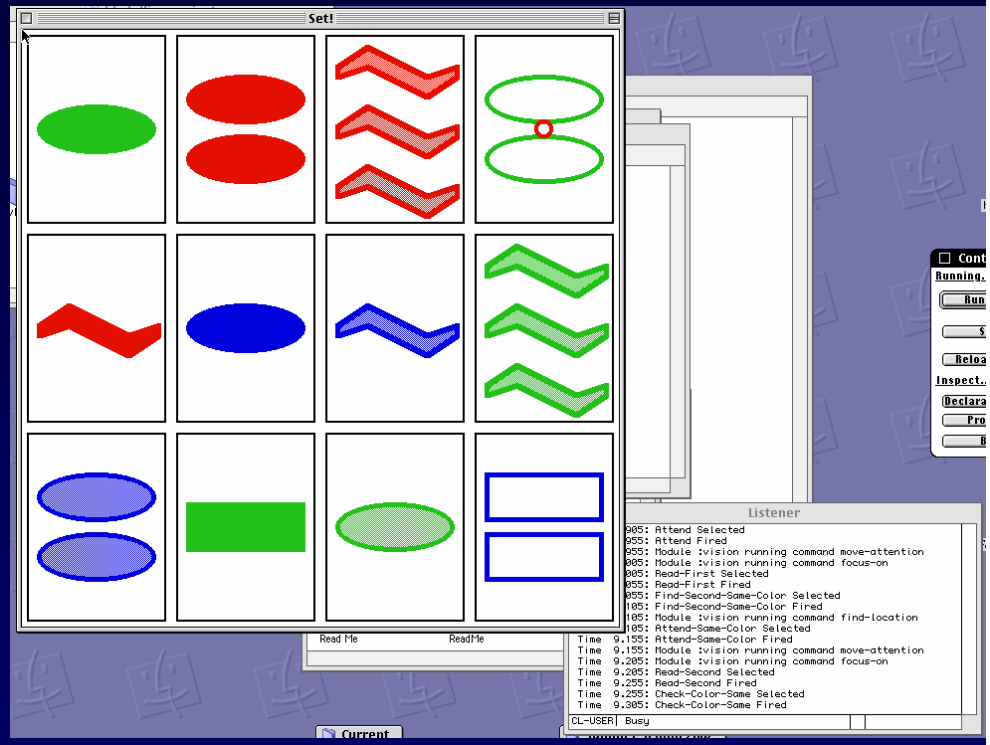
The model

- Why do hard sets take longer?
  - Checking for unequal attributes takes longer
- Why are experts better at hard problems?
  - They are better at multi-tasking, which pays off in hard problems

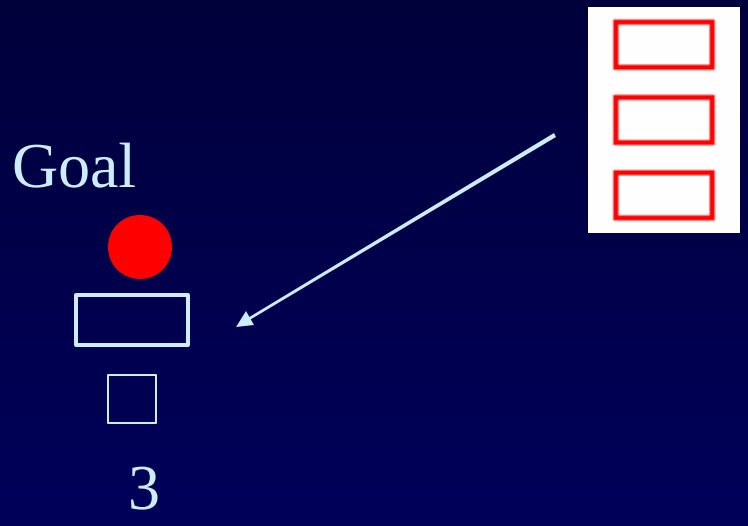


The model

- Implementation in ACT-R 5.0
- Custom visual object: set card with attributes color, shape, filling and number

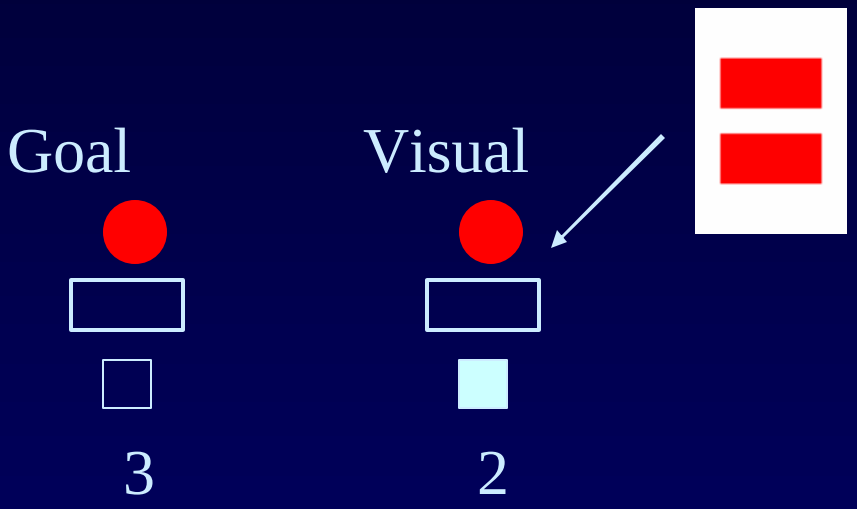


How the model works



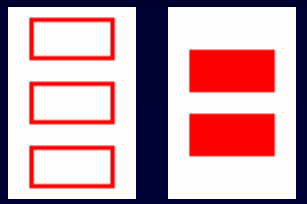
- First, pick a random card and stick it in the goal

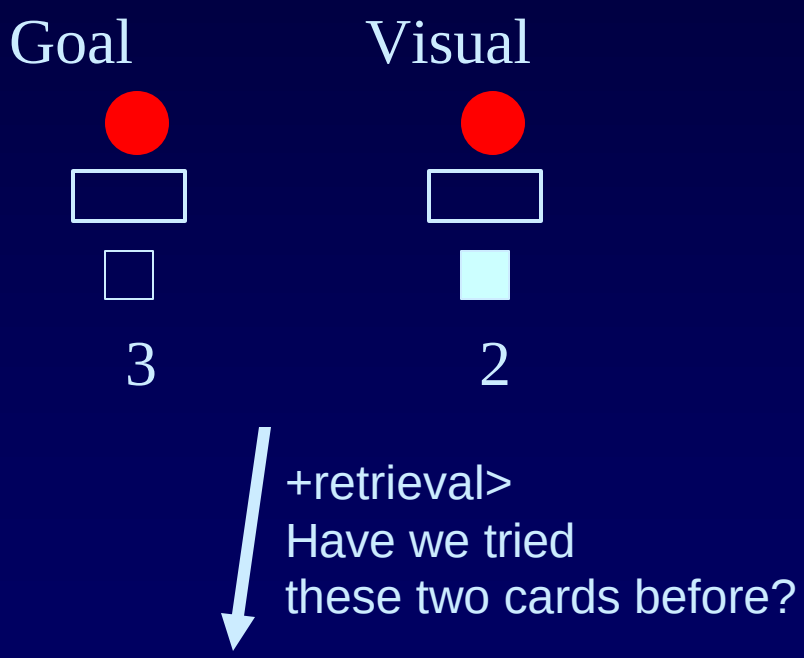
How the model works



- Second, search for a card of the same color. If this fails, search for an arbitrary different card
- We don’t put this card in the goal, but leave it in =visual

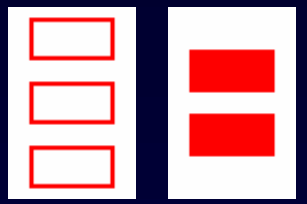
How the model works

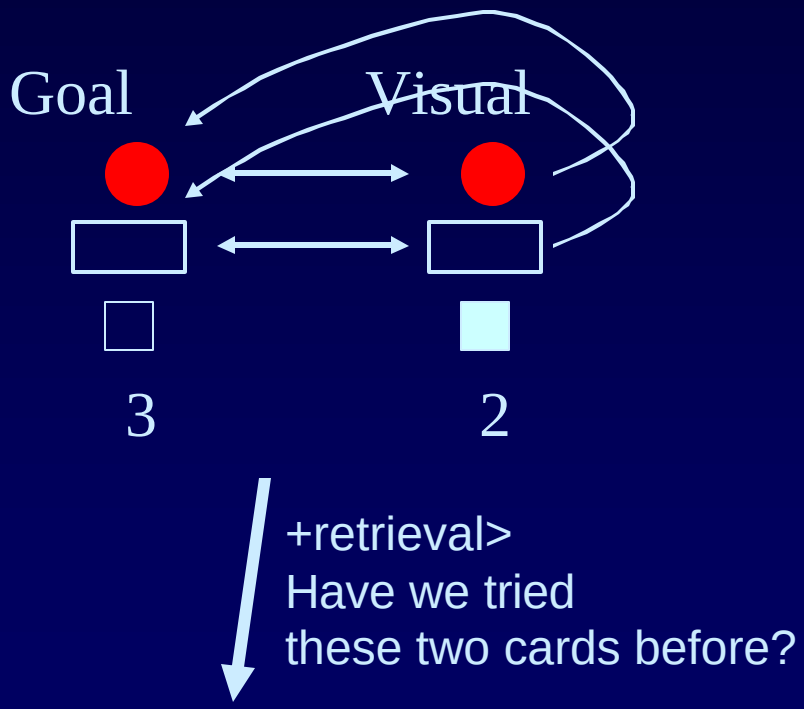




- Now the model is going to do two things in parallel:
  - Check in declarative memory whether or not we tried this combination of two cards before
  - Make a prediction what the third card has to look like

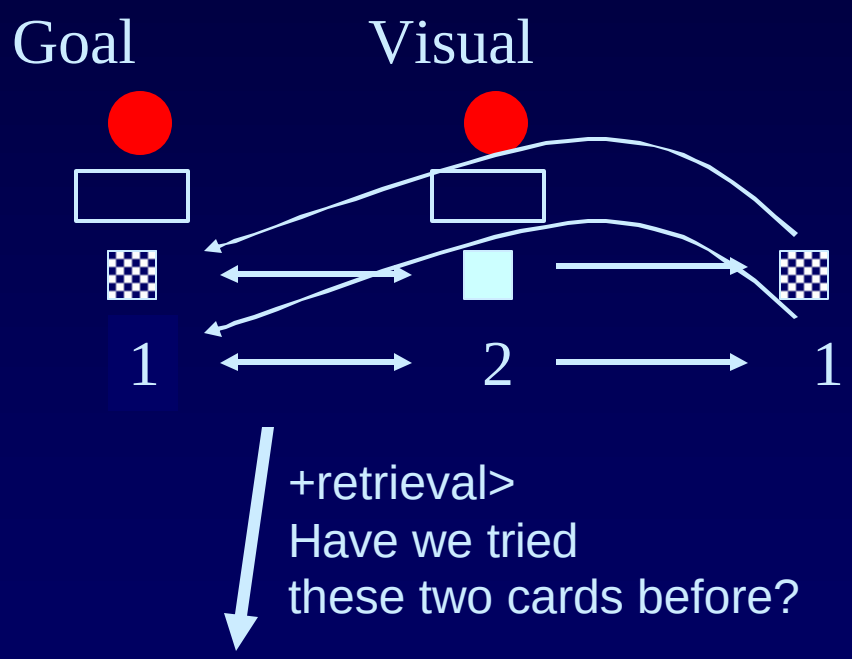
How the model works





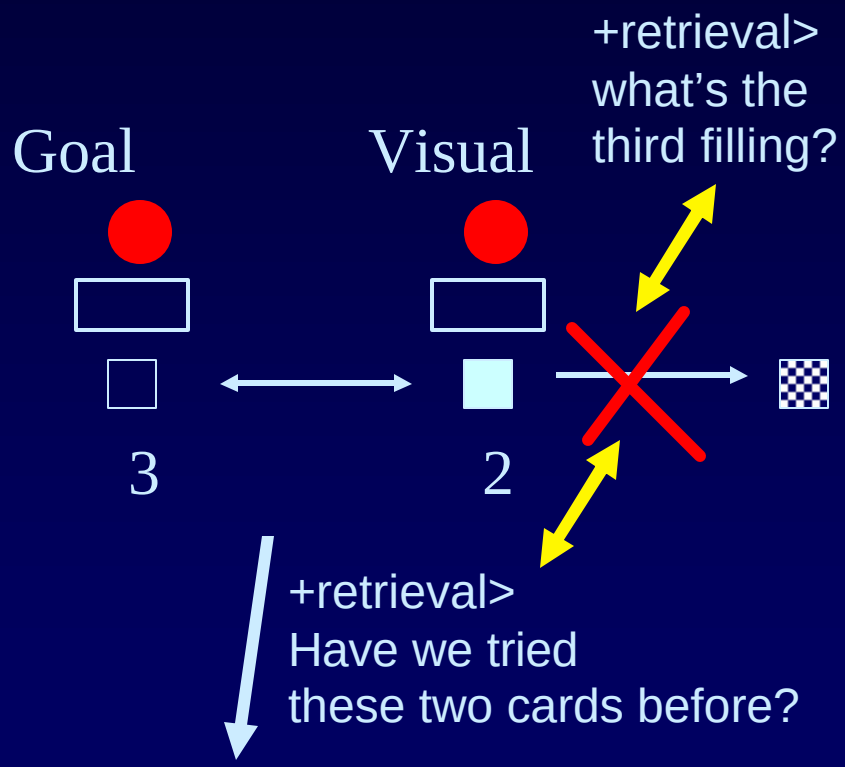
- Predicting the third card
  - For each attribute, we determine what it has to be like in the third card, and put this back into the goal
  - When the attribute for goal and visual are equal, this attribute is also the desired attribute for the new card

How the model works



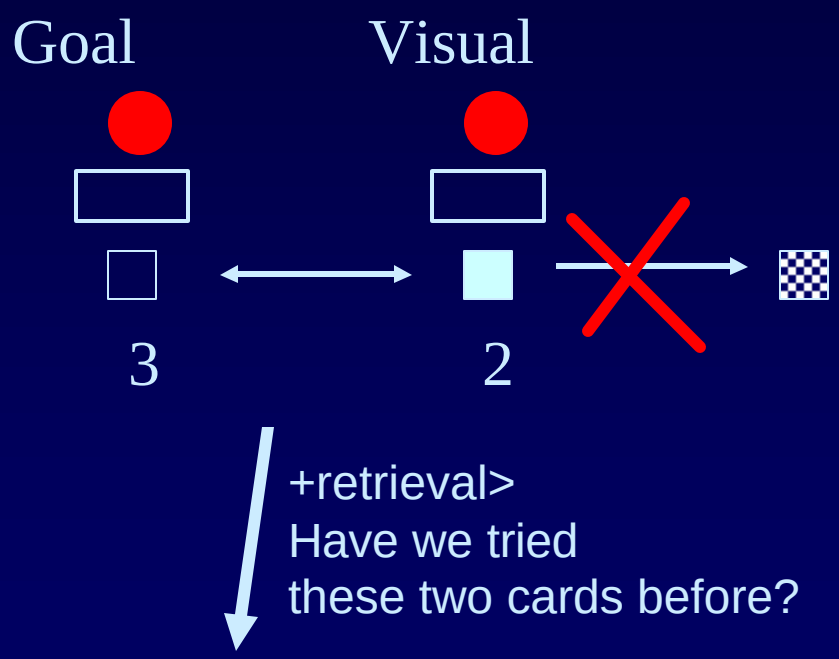
- Predicting the third card
  - For each attribute, we determine what it has to be like in the third card, and put this back into the goal
  - When the attribute for goal and visual are equal, this attribute is also the desired attribute for the new card
  - When the attributes are different, we have to determine the third value

How the model works



- Beginners:
  - At the moment that (in this example) the filling has to be determined, a declarative retrieval is needed
  - This is however impossible, because declarative memory is still engaged in another retrieval!
  - So the beginner has to wait until the first declarative retrieval is done
- Experts
  - Have proceduralized the retrieval for the third attribute value

How the model works

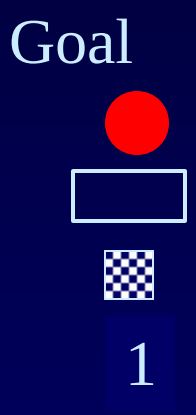


For beginners, the prediction process is blocked as soon as two attribute values are different.

Experts can just proceed

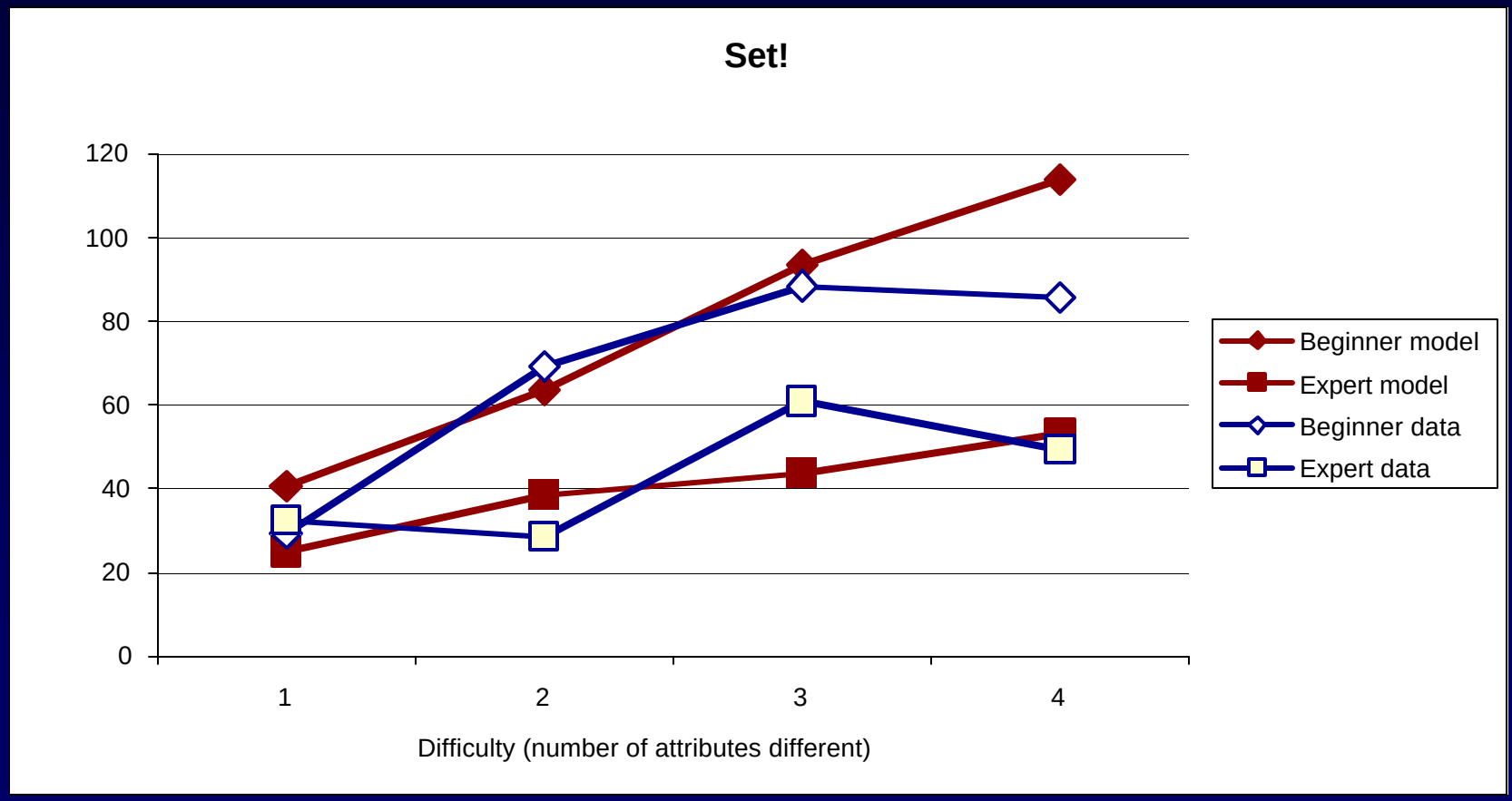
That's why experts are especially good at the hard sets, in which many attributes are different

How the model works

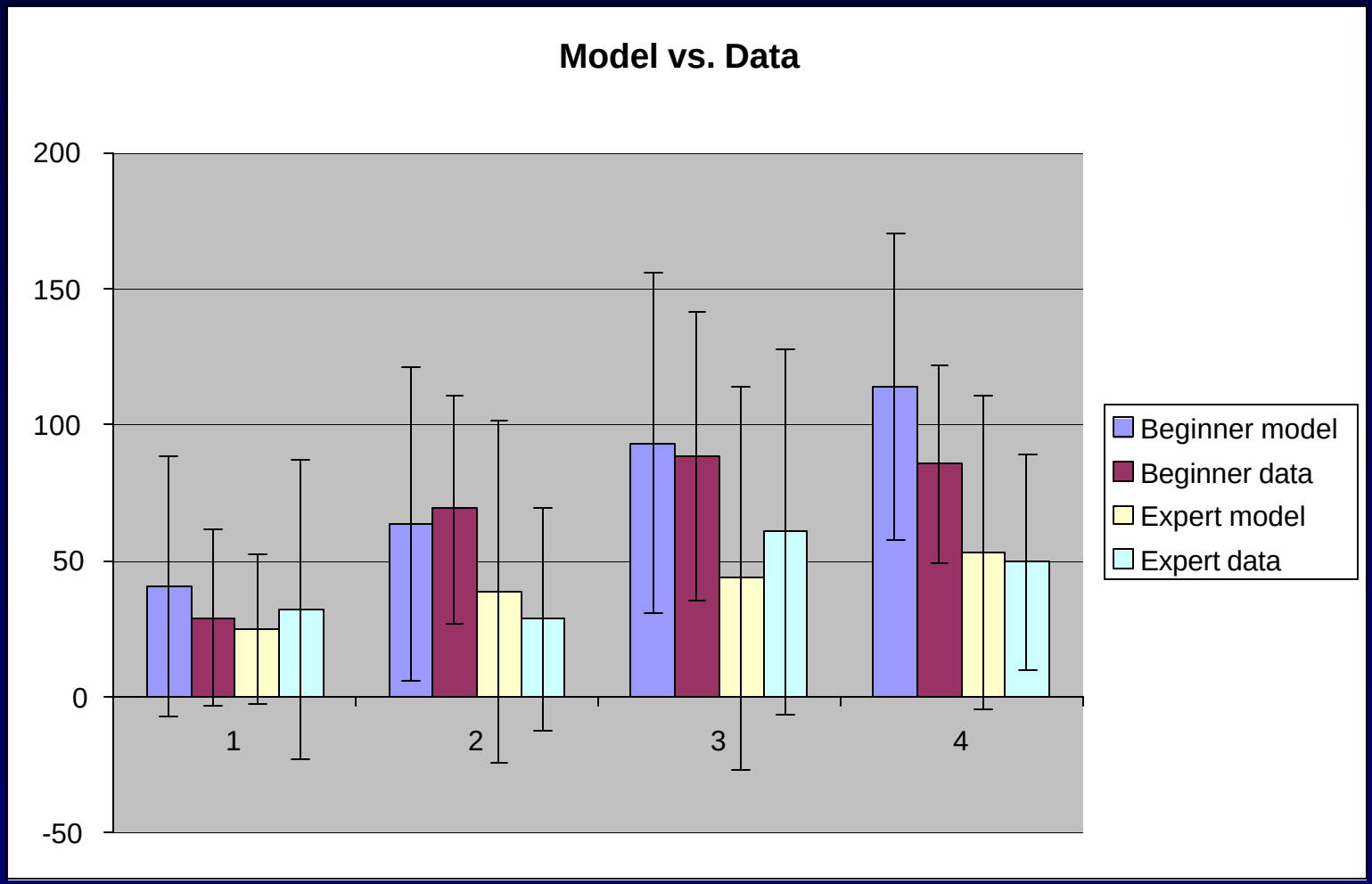


- After predicting the third card
  - Now that the third card has been predicted, the model tries to find it on the screen.
  - If this fails, it starts all over again
  - If this succeeds, it announces it has found a Set!

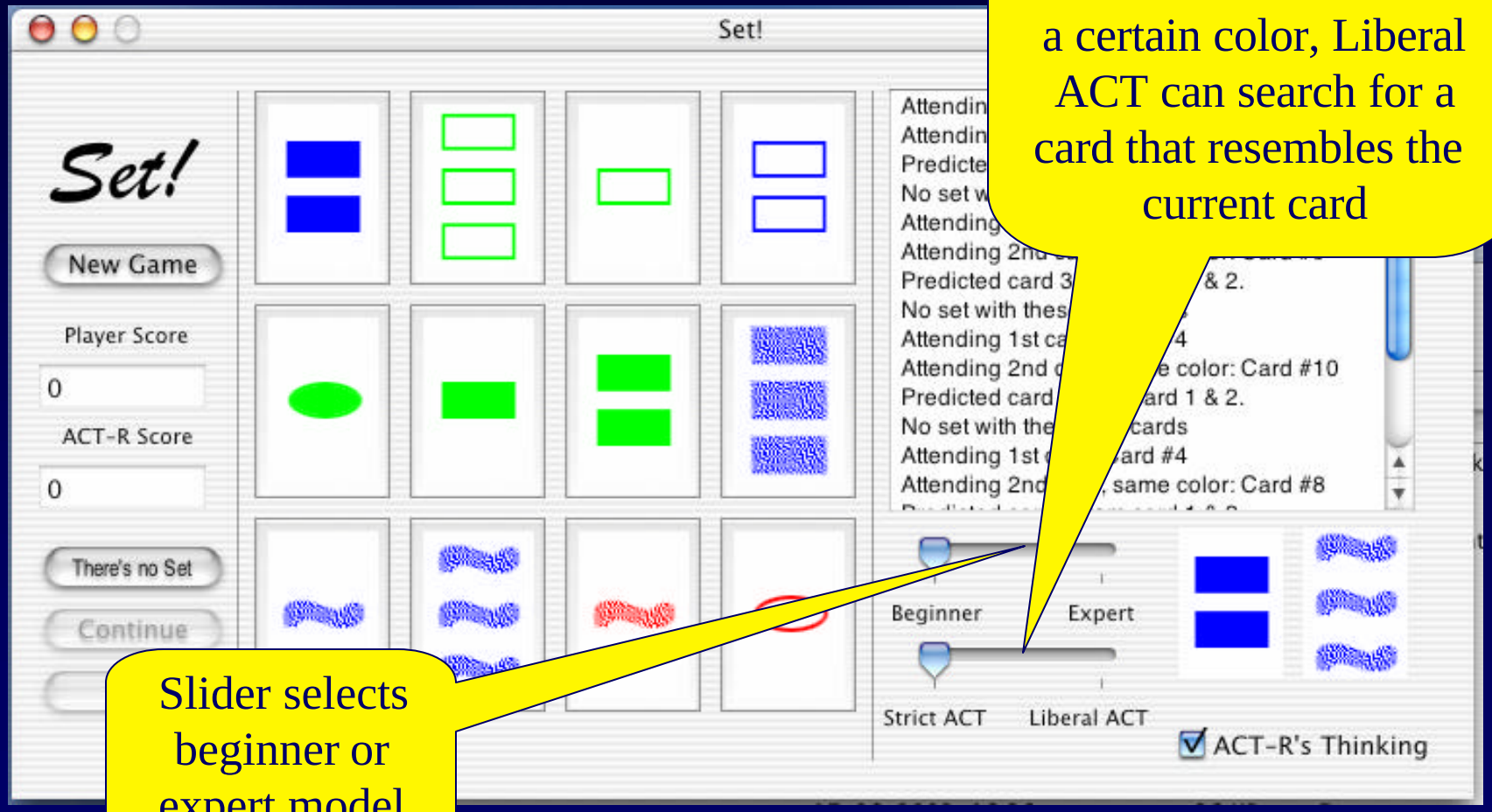
Results of the Model



Results of the Model

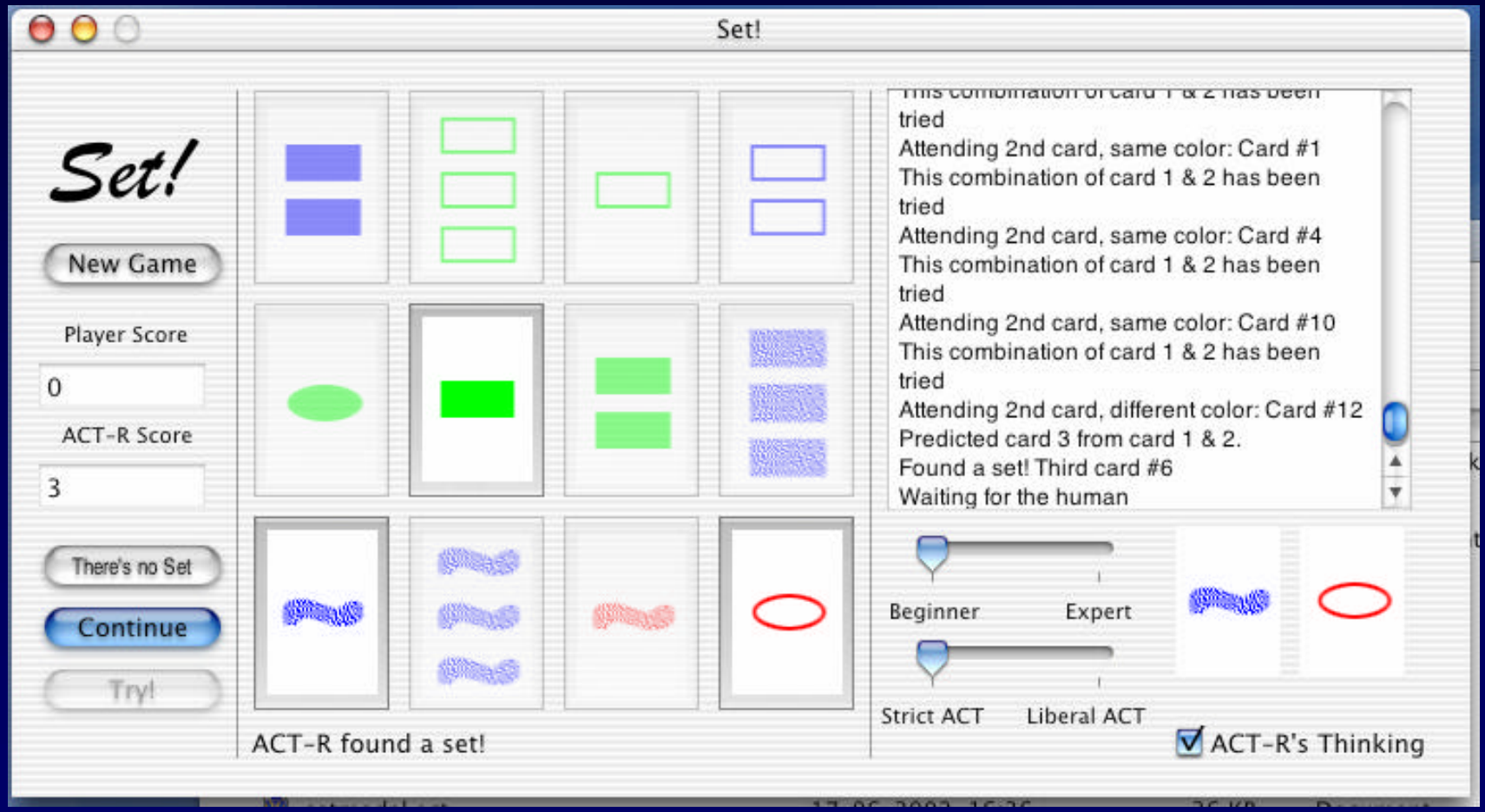


The Application



The Application

Demo!



The Issues

- For multi-tasking, ACT-R has to be able to know whether or not a module is in use at the moment.
- For declarative memory this is currently impossible (except by keeping track of it in the goal)
- For the visual buffer, there is the =visual-state>, but it will signal “free” when there is something waiting in =visual> to be processed

The Issues

- The “liberal ACT” setting produces behavior that is more believable than “strict ACT”
- There’s more to visual perception. (Of course, we already knew this)

Conclusions

- Possible fruitful domain for ACT-R with respect to games: believable opponents (but that's what the military simulations are also about)
- Interesting interaction between production compilation and multi-tasking (see also: Lee & Taatgen at CogSci)
- Download the game from: http://www.ai.rug.nl/~niels/set-app