AngryHEX: An Angry Birds-playing Agent based on HEX-Programs

Francesco Calimeri and **Michael Fink** and **Stefano Germano** and **Andreas Humenberger**
Giovambattista Ianni and **Christoph Redl** and **Daria Stepanova** and **Andrea Tucci**

calimeri@mat.unical.it, fink@kr.tuwien.ac.at, stefanogermano0@gmail.com, e1026602@student.tuwien.ac.at
ianni@mat.unical.it, redl@kr.tuwien.ac.at, dasha@kr.tuwien.ac.at, andrea.tucci.cs@gmail.com

Abstract

Our agent is a joint development of two groups at Technische Universität Wien (TUWIEN) and Università della Calabria (UNICAL). A distinctive characteristic of our agent is that it uses a declarative, logic-programming based module for reasoning about the target to shot at next. It is realized as a so-called HEX-program [Eiter *et al.*, 2006], i.e., by means of Answer Set Programming (ASP) with external sources and other extensions.

TUWIEN

Team members from Technische Universität Wien (Vienna, Austria) are affiliated with and/or students of the **Knowledge Based Systems Group (KBS)**[1] at the **Institut für Informationssysteme**. Research at the group focuses on foundations and formal aspects of knowledge-based systems and Artificial Intelligence, with emphasis on (but not restricted to):

- Knowledge Representation and Reasoning,
- Computational Logic and Complexity.
- Declarative Problem Solving,
- Intelligent Agents and Mobile Robots,
- Knowledge-Based Systems in Engineering.

The respective AngryHEX team members are Michael Fink, Andreas Humenberger, Christoph Redl, and Daria Stepanova.

UNICAL

Team members from Università della Calabria (Rende, Italy) are affiliated with the **Computer Science Group** at the **Department of Mathematics and Computer Science**[2]. Research at the group focuses on formal aspects and applications of knowledge-based systems and Artificial Intelligence. A non-comprehensive list of research topics features:

- Knowledge Representation and Reasoning,
- Computational Logics,
- Advanced Data and Knowledge Management,
- Intelligent Agents,
- Intelligent Information Systems,
- Modeling and Simulation of Complex Systems.

[1] http://www.kr.tuwien.ac.at
[2] http://www.mat.unical.it

The respective AngryHEX team members are Francesco Calimeri, Stefano Germano, Giovambattista Ianni, and Andrea Tucci.

The AngryHEX agent

Our agent, called AngryHEX, builds on the Base Framework provided by the organizers and extends it with declarative means for decision making models by means of an Answer Set Programming (ASP).

Declarative logic programming kicks in on two different layers for AngryHEX: the **Tactics** layer, which plans shots, and decides how to complete a level; and the **Strategy** layer, which decides the order of levels to play and possible multiple attempts to solve the same level.

Tactics is declaratively realized by HEX-programs, i.e., an extension of ASP to incorporate external sources of computation via so-called *external atoms*. It is implemented using the DLVHEX solver and computes optimal shots based on information about the current scene and on domain knowledge modeled within the HEX-program. Its *input* comprises scene information encoded as a set of logic program facts (position, size and orientation of pigs, ice, wood and stone blocks, slingshot, etc.); its *output* are *answer sets* that contain a dedicated atom describing the target to hit, and further information about the required shot. Physics simulation results and other information are accessed via external atoms.

The **Strategy** layer decides, at the end of each level, which level to play next. This layer is also realized declaratively as an (ordinary) ASP program encoding our strategy on three priority levels: (1) each available level is played once; (2) levels where the agent score differs most from the current best score are selected; (3) levels where AngryHEX achieved a score higher than the current best scores and that have the minimum difference from the best score, are selected. For each level, the **Strategy** layer keeps tracks of previously achieved scores and previously selected initial target objects.

References

[Eiter *et al.*, 2006] Thomas Eiter, Giovambattista Ianni, Roman Schindlauer, and Hans Tompits. Effective Integration of Declarative Rules with External Evaluations for Semantic-Web Reasoning. In *ESWC*, volume 4011 of *LNCS*, pages 273–287. Springer, 2006.

AngryHex: an Angry Birds-playing Agent based on HEX-programs

Francesco Calimeri [1] Michael Fink [2] Stefano Germano [1]
Andreas Humenberger [2] Giovambattista Ianni [1] Christoph Redl [2]
Daria Stepanova [2] Andrea Tucci

[1] Università della Calabria, Dipartimento di Matematica e Informatica,

[2] Technische Universität Wien, Institut für Informationssysteme

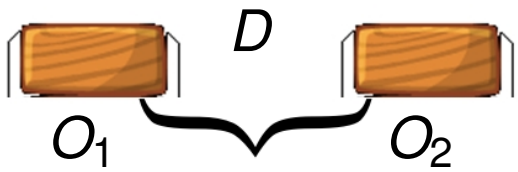
ECAI, Angry Birds Competition–August 19, 2014

Motivation

- **Approach:** design an agent based on declarative logic programming
- **Challenge:** plan optimal shots under consideration of some physics
- **Our means:** HEX-programs, i.e. Answer Set Programs (ASP) with external sources

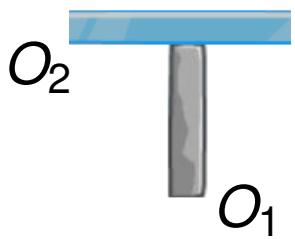
HEX-programs

- HEX-program Π is a set of ASP rules, where external atoms are allowed in rule bodies:
  - $\&distance[O_1, O_2](D)$ is true iff distance between O_1 and O_2 is D
  - $\&canpush[ngobject](O_1, O_2)$ is true iff O_1 can push O_2 given additional info in the extention of *ngobject*



- $Rule_1$ estimates the likelihood that object O_2 falls when O_1 is hit

$$Rule_1: \quad pushDamage(O_2, P_1, P) \leftarrow pushDamage(O_1, \_, P_1), P_1 > 0$$
$$\&canpush[ngobject](O_1, O_2),$$
$$pushability(O_2, P_2), P = P_1 * P_2/100.$$



Architecture of Angryhex

- We use the provided framework (browser plugin, vision module, . . .)
- Agent builds on tactics and strategy, both are realized declaratively

- **Tactics:** reasoning about the next shot is done in a HEX-program Π
  - **Input**: scene info from the vision module (facts of Π)
  - **Output**: desired target (models of Π)

- **Strategy:** next level to played is computed in an ASP program Π'
  - **Input:** info about the number of times levels were played, best scores achieved, scores of our agent (facts of Π')
  - **Output:** next optimal level to be played (models of Π')

HEX Encoding for Tactics

- **Physics simulation results** are accessed via external atoms:
  - decide which O' intersect with trajectory of a bird after hitting O
  - decide whether O_1 falls whenever O_2 falls ...

- **Tactics in details:**
  - Consider each shootable target
  - Compute the estimated damage on each non-target object
  - Rank the targets (=answer sets) using weak constraints
  - Consider history: never play a level in the same way again!

ASP Encoding for Strategy

- **Decides which level to play next** based on info about:
  - number of times each level was played
  - best scores
  - our agent's scores . . .

- **Strategy in details:**
  - First play each level once
  - Second play levels in which our score maximally differs from the best one
  - Third play levels in which we played best and the difference to the second best score is minimal

Conclusion and Future Work

- **Wrap-up:**
  - Agent is realized using declarative programming means
  - Vision module provided by the organizers is integrated
  - Declarative strategy is realized (used to be in java)
  - Fixes and improvements in comparison to previous version

- **Possible improvements:**
  - Combine objects which behave like a single one
  - Plan over multiple shots
  - Improve object recognition and general precision of shots