General Game Playing

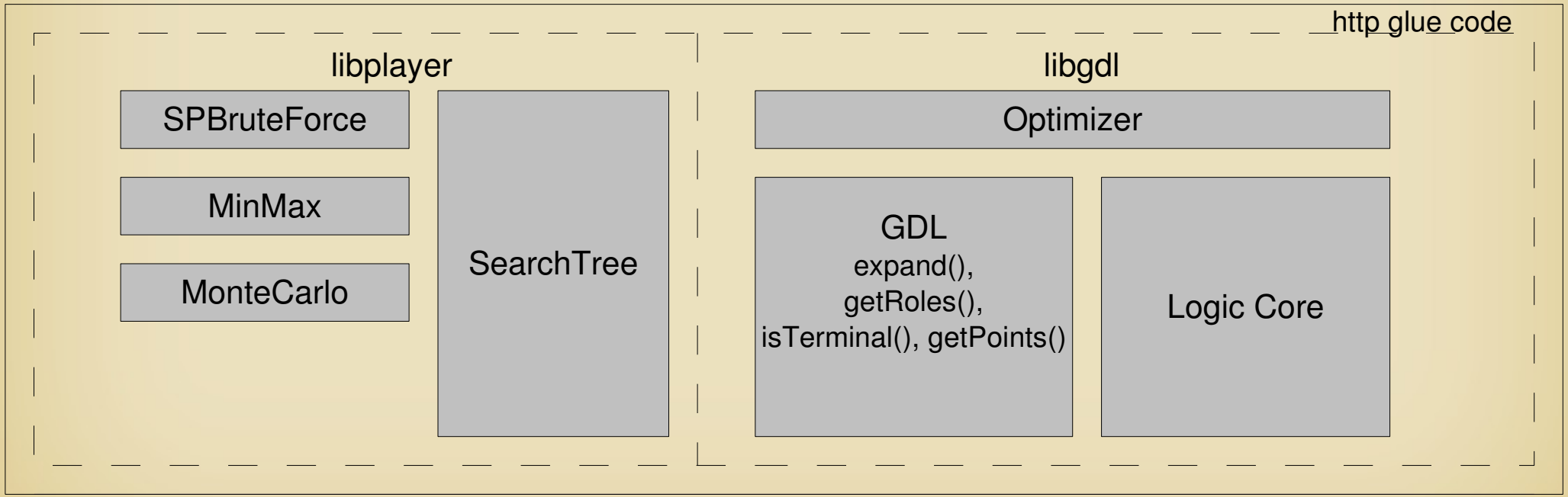
Informafiosi

Group Informafiosi
Norbert Schultz
David Müller
Norbert Manthey

Course WS08/09:
Prof. Thielscher
Dipl.Inf. Stephan Schiffel

Basics

- written in C++
- own logic-interpreter, own HTTP-communication
- all players use the same data structure, SimpleSearchTree
- three players: SinglePlayerBruteForce, 2PlayerMinMax, MultiPlayerMonteCarlo, decision via number of roles



Logic

- based upon unify operation of book “artificial intelligence: a modern approach” (with mgu_check)
- caching of logic questions with no or one result (cuts up to 50% of all questions)
- recognition of linear and comparison relations
- compression of the database into small words : end-to-end decompression. gives 10% performance, but saves memory
- multi threading support for expanding nodes
- faster reimplementation of c++ std::string, std::ostringstream
- hashing on the base of relation signatures (like legal/2)
- support up to basic interface: getRoles(), getMoves(), expand(), isTerminal(), getPoints()

SinglePlayer Brute Force

- own player idea, at first for logic testing
- Iterative deepening, trusting into its own speed ;)
- using monte carlo approach on ~ 30% of it's time, so able to solve more complex games
- optimized via hash table

Two-Player MinMax

- standard MinMax-algorithmen
- transposition table
- iterative deepening for sorting nodes
- only for zerosum games with alternating moves
- heuristics: novelty, mobility, inverse mobility
- weighting is with upon a RandomGamer test at initialization

>2 Players MonteCarlo

- used for players with more than 2 roles
- play random games for each possible move
- make more random games for move, which promises great reward

Strengths & Weaknesses

- Good
  - 1P: speedy at small games, solves most common games
  - 2P: good for zero sum games with alternating moves
  - >3P: good for non-lookthroughable games
  - All: memory efficient (although some players don't clean up)
- Weak
  - 1P: Only one solution (e.g. 8puzzle)
  - 2P: Non MinMax games are solved bad
  - >3P: simple games are not solved that efficient
  - no serial / parallel game optimization

Still Todo

- Faster!
- MaxMaxPlayer
- Clean memory at the end of non 1P-Players
- fuzzy logic approach for recognition of “near the end” states, could use the already recognised linear / comparison relations. some code is already done; but never came ready.
- identification of serial / parallel games

the end

let the games begin

image by goldenknabe / flickr