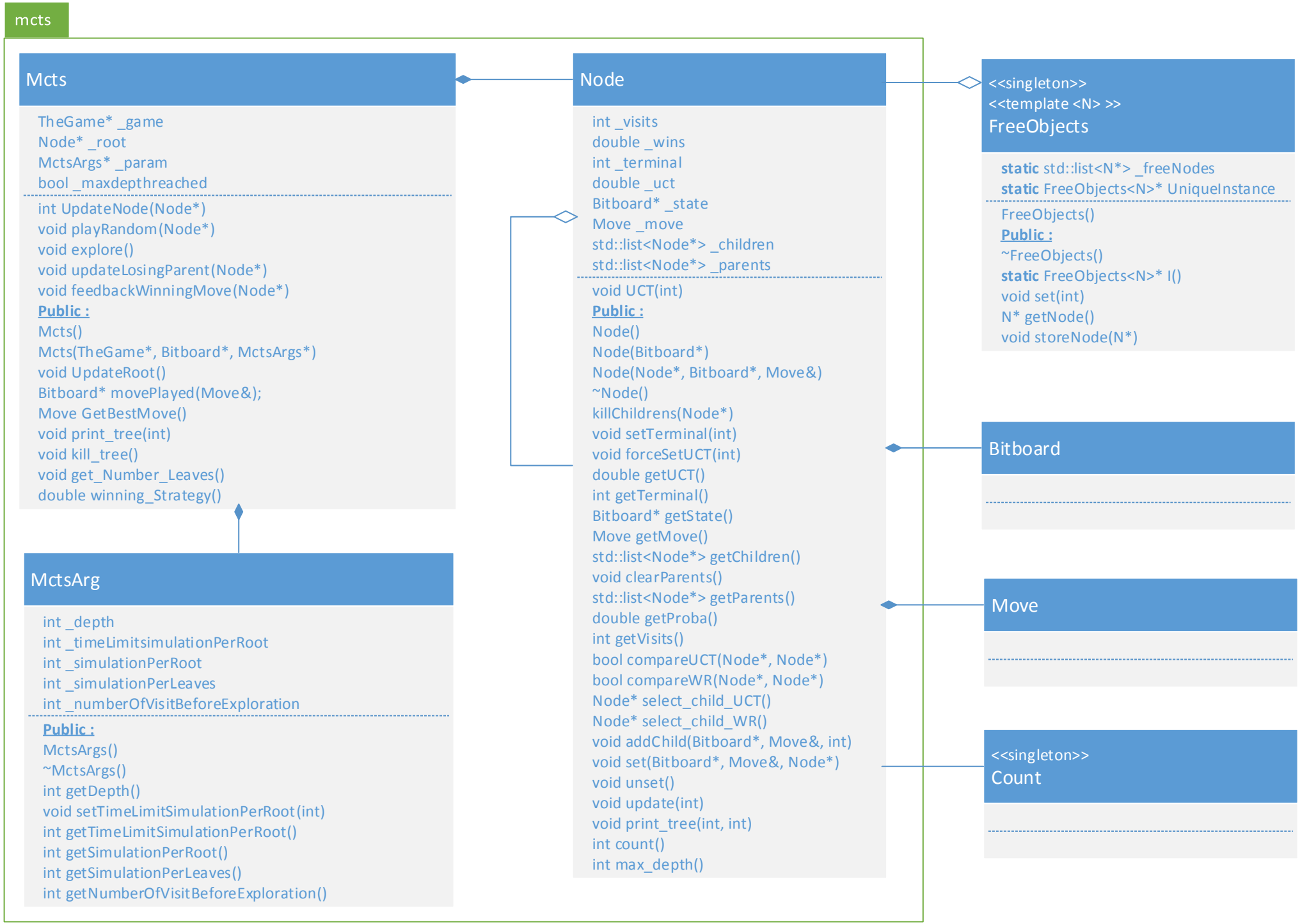
mcts
Mcts
TheGame* _game
Node* _root
MctsArgs* _param
bool _maxdepthreached
int UpdateNode(Node*)
void playRandom(Node*)
void explore()
void updateLosingParent(Node*)
void feedbackWinningMove(Node*)
Public :
Mcts()
Mcts(TheGame*, Bitboard*, MctsArgs*)
void UpdateRoot()
Bitboard* movePlayed(Move&);
Move GetBestMove()
void print_tree(int)
void kill_tree()
void get_Number_Leaves()
double winning_Strategy()
MctsArg
int _depth
int _timeLimitsimulationPerRoot
int _simulationPerRoot
int _simulationPerLeaves
int _numberOfVisitBeforeExploration
Public :
MctsArgs()
~MctsArgs()
int getDepth()
void setTimeLimitSimulationPerRoot(int)
int getTimeLimitSimulationPerRoot()
int getSimulationPerRoot()
int getSimulationPerLeaves()
int getNumberOfVisitBeforeExploration()
Node
int _visits
double _wins
int _terminal
double _uct
Bitboard* _state
Move _move
std::list<Node*> _children
std::list<Node*> _parents
void UCT(int)
Public :
Node()
Node(Bitboard*)
Node(Node*, Bitboard*, Move&)
~Node()
killChildrens(Node*)
void setTerminal(int)
void forceSetUCT(int)
double getUCT()
int getTerminal()
Bitboard* getState()
Move getMove()
std::list<Node*> getChildren()
void clearParents()
std::list<Node*> getParents()
double getProba()
int getVisits()
bool compareUCT(Node*, Node*)
bool compareWR(Node*, Node*)
Node* select_child_UCT()
Node* select_child_WR()
void addChild(Bitboard*, Move&, int)
void set(Bitboard*, Move&, Node*)
void unset()
void update(int)
void print_tree(int, int)
int count()
int max_depth()
<<singleton>>
<<template <N> >>
FreeObjects
static std::list<N*> _freeNodes
static FreeObjects<N>* UniqueInstance
FreeObjects()
Public :
~FreeObjects()
static FreeObjects<N>* I()
void set(int)
N* getNode()
void storeNode(N*)
Bitboard
Move
<<singleton>>
Count