THE MIGHTY CHALLENGE OF MODELLING GEOPOLITICAL BEHAVIOUR IN TOTAL WAR:

HOW AI CAN DELIVER ENTERTAINMENT.

Guy Davidson
@hatcat01

Duygu Cakmak
@aphrael
Overview

- Introducing the Total War campaign
- An overview of AI systems and the world state
- A consideration of diplomacy
- Tasks and resources
- Profiling and timing
Campaign mechanism

- Factions
- Territory
- Resources
Campaign mechanism

- Factions
- Territory
- Resources
- Opportunity cost
Turn sequence

• Inspect resources
• Cities – taxation, construction
• Armies
Turn sequence

- Inspect resources
- Cities – taxation, construction
- Armies
- Agents
Turn sequence

- Now it’s the AI’s turn(s)...
- Attack, besiege, ally, trade, espionage...
- Until...
Turn sequence

• Now it’s the AI’s turn(s)...
• Attack, besiege, ally, trade, espionage...
• Until...
• Win conditions
Win condition

- Control N territories
- Control particular territories
- Eliminate a faction
Win condition

- Control N territories
- Control particular territories
- Eliminate a faction
- Alliances with all factions
We provide...

- Model of the world
- AI opponent for each faction
- Position and scalar data
We provide...

- Model of the world
- AI opponent for each faction
- Position and scalar data
- Model resolved over time
Equipping the AI

- Perfect information
- Privileged information
Equipping the AI

- Perfect information
- Privileged information
- Fun opponent
- Credible opponent
Equipping the AI

- View API
- Control API
Equipping the AI

- View API
- Control API
- Motivation
Equipping the AI

- Personality
- Informed by circumstance
- Traders
Equipping the AI

- Personality
- Informed by circumstance
- Traders
- Fighters
Overview

- Introducing the Total War campaign
- An overview of AI systems and the world state
- A consideration of diplomacy
- Tasks and resources
- Profiling and timing
Team effort

- Dr Tim Gosling
- Piotr Andruszkiewicz
**Decision domains**

- Economy
- Construction
- Diplomacy
- Army composition
- Army deployment / movement
- Technology
- Characters & Skills

**AI Subsystems in Campaign**

- Financial System
- Construction System
- Diplomacy System
- Task Management System
- Technology Management
- Character Management
Decision domains

- Economy
- Construction
- Diplomacy
- Army composition
- Army deployment / movement
- Technology
- Characters & Skills

AI Subsystems in Campaign

- Financial System
- Construction System
- Diplomacy System
- Task Management System
- Technology Management
- Character Management
Decision domains

Economy
Construction
Diplomacy
Army composition
Army deployment / movement
Technology
Characters & Skills

AI Subsystems in Campaign

- Financial System
- Construction System
- Diplomacy System
- Task Management System
- Technology Management
- Character Management

World State (Analysers)
AI Subsystems in Campaign
- Financial System
- Construction System
- Diplomacy System
- Task Management System
- Technology Management
- Character Management

World State (Analysers)
- CAI_MILITARY_STRENGTH_ANALYSER
- CAI_ATTITUDE_ANALYSER
- CAI_TASK_GEOSPATIAL_ANALYSER
- CAI_UNIT_AVAILABILITY_ANALYSER
- CAI_IMPORTANT_FACTION_ANALYSER
- CAI_ANALYSER
- CAI_RESPECT_ANALYSER
- CAI_FINANCIAL_ANALYSER
- CAI_PRODUCED_RESOURCE_ANALYSER
What are we building and what we need the most?
(CAI_ACTIVE_CONSTRUCTION_ANALYSER)

Which factions are important to me?
(CAIIMPORTANT_FACITION_ANALYSER)

Attitudes of me
Attitudes of others
(CAI_ATTITUDE_ANALYSER)

Current income & treasury
(CAI_FINANCIAL_ANALYSER)

Available units
(CAI_UNIT_AVAILABILITY_ANALYSER)

My strength
Opponent’s strength
(CAI_MILITARY_STRENGTH_ANALYSER
CAI_FACTION_ALLIANCE_STRENGTH_ANALYSER
CAI_FACTION_STRENGTH_ANALYSER)
What are we building and what we need the most?
(CAI_ACTIVE_CONSTRUCTION_ANALYSER)

Which factions are important to me?
(CAI_IMPORTANT_FACTION_ANALYSER)

Attitudes of me
Attitudes of others
(CAI_ATTITUDE_ANALYSER)

Current income & treasury
(CAI_FINANCIAL_ANALYSER)

My strength
Opponent’s strength
(CAI_MILITARY_STRENGTH_ANALYSER
CAI_FACTION_ALLIANCE_STRENGTH_ANALYSER
CAI_FACTION_STRENGTH_ANALYSER)

Available units
(CAI_UNIT_AVAILABILITY_ANALYSER)
What are we building and what we need the most?
(CAI_ACTIVE_CONSTRUCTION_ANALYSER)

Which factions are important to me?
(CAI_IMPORTANT_FACTION_ANALYSER)

Attitudes of me
Attitudes of others
(CAI_ATTITUDE_ANALYSER)

Current income & treasury
(CAI_FINANCIAL_ANALYSER)

My strength
Opponent’s strength
(CAI_MILITARY_STRENGTH_ANALYSER
CAI_FACTION_ALLIANCE_STRENGTH_ANALYSER
CAI_FACTION_STRENGTH_ANALYSER)

Available units
(CAI_UNIT_AVAILABILITY_ANALYSER)
Lazy Evaluation

Lazy evaluation (or call-by-name) is an evaluation strategy which delays the evaluation of an expression until its value is needed.

I know what to do. Wake me up when you really need it.
Analysers

CAI_FINANCIAL_ANALYSIS

CAI_IMPORTEDRESOURCE_ANALYSER

CAI_FACTION_ALLIANCE_STRENGTH_ANALYSIS

CAI_OWNED_REGIONS_ANALYSIS

CAI_DIPLOMATIC_ANALYSIS

CAI_TRESPASSING_ANALYSIS

CAI_FACTION_STRENGTH_ANALYSIS

FACTION

CAI_RESOURCE_MOBILE_MILITARY_STRENGTH_ANALYSIS
Analysers

CAI_ACTIVE_CONSTRUCTION_ANALYSER
CAI_ACTIVE_RECRUITMENT_ANALYSER
CAI_ATTITUDE_ANALYSER
CAI_ATTRITION_ANALYSER
CAI_BASIC_REGION_GROUP_ANALYSER
CAI_BUILDING_AVAILABILITY_ANALYSER
CAI_CHARACTER_ROLE_ANALYSER
CAI_ANALYSER
CAI_DIRECT_ATTITUDE_ANALYSER
CAI_FACTION_ALLIANCE_STRENGTH_ANALYSER
CAI_FACTION_RESEARCH_TECHNOLOGY_ANALYSER
CAI_FACTION_STRENGTH_ANALYSER
CAI_FACTION_TAXATION_ANALYSER
CAI_FACTIONWIDE_UNIT_AVAILABILITY_ANALYSER
CAI_FERTILITY_ANALYSER
CAI_FINANCIAL_ANALYSER
CAI_FOOD_ANALYSER
CAI_IMPORTANT_FACTION_ANALYSER
CAI_IMPORTED_RESOURCE_ANALYSER
CAI_MILITARY_ACCESS_ANALYSER
CAI_MILITARY_STRENGTH_ANALYSER
CAI_NEIGHBOURING_FACTIONS_ANALYSER
CAI_IMPORTED_REGIONS_ANALYSER
CAI_PLAYER_PROXIMITY_ANALYSER
CAI_PRODUCED_RESOURCE_ANALYSER
CAI_RESPECT_ANALYSER
CAI_STRATEGIC_CONTEXT_ANALYSER
CAI_TASK_GEOGRAPHICAL_ANALYSER
CAI_TASK_RECRUITMENT_PREFERENCE_ANALYSER
CAI_TRESPASSING_ANALYSER
CAI_UNIT_AVAILABILITY_ANALYSER
CAI_VICTORY_REGION_ANALYSER
CAI_ANALYSER_BASE

```
template<class ANALYSED_CLASS, class ANALYSIS_CLASS, BDI_CLASSES_IDENTIFICATION analyser_id>
class CAI_ANALYSER_BASE : public CAI_ANALYSER
{
    public:

    CAI_ANALYSER_BASE
    virtual bool post_load_fix_up
    virtual bool validate
    virtual void save

    ANALYSIS_CLASS & determine_analysis_for (ANALYSED_CLASS & target);
    const ANALYSIS_CLASS * determine_analysis_for_const(const ANALYSED_CLASS & target) const;

};
```
template<class ANALYSED_CLASS, class ANALYSIS_CLASS, BDI_CLASSES_IDENTIFICATION analyser_id>
ANALYSIS_CLASS &CAI_ANALYSER_BASE<ANALYSED_CLASS, ANALYSIS_CLASS, analyser_id>::
determine_analysis_for(ANALYSED_CLASS & target)
{
    CAI_ANALYSIS *general_analysis = get_analysis(target.get_bdi_index());
    ANALYSIS_CLASS *specific_analysis = nullptr;
    if( general_analysis )
    {
        specific_analysis = static_cast<ANALYSIS_CLASS *>(general_analysis);
    }
    else
    {
        specific_analysis = new ANALYSIS_CLASS(target);
        add_analysis(*specific_analysis, target.get_bdi_index());
    }
    if( specific_analysis->is_invalidated() )
    {
        specific_analysis->do_validation(get_bdi_pool());
    }
    return *specific_analysis;
}
template<class ANALYSED_CLASS, class ANALYSIS_CLASS, BDI_CLASSES_IDENTIFICATION analyser_id>
ANALYSIS_CLASS &CAI_ANALYSER_BASE<ANALYSED_CLASS, ANALYSIS_CLASS, analyser_id>
::determine_analysis_for(ANALYSED_CLASS & target)
{
    CAI_ANALYSIS *general_analysis = get_analysis(target.get_bdi_index());
    ANALYSIS_CLASS *specific_analysis = nullptr;
    if( general_analysis )
    {
        specific_analysis = static_cast<ANALYSIS_CLASS *>(general_analysis);
    }
    else
    {
        specific_analysis = new ANALYSIS_CLASS(target);
        add_analysis(*specific_analysis, target.get_bdi_index());
    }
    if( specific_analysis->is_invalidated() )
    {
        specific_analysis->do_validation(get_bdi_pool());
    }
    return *specific_analysis;
}
template<class ANALYSED_CLASS, class ANALYSIS_CLASS, BDI_CLASSES_IDENTIFICATION analyser_id>
ANALYSIS_CLASS &CAI_ANALYSER_BASE<ANALYSED_CLASS, ANALYSIS_CLASS, analyser_id>::determine_analysis_for(ANALYSED_CLASS & target)
{
    CAI_ANALYSIS *general_analysis = get_analysis(target.get_bdi_index());
    ANALYSIS_CLASS *specific_analysis = nullptr;
    if( general_analysis )
    {
        specific_analysis = static_cast<ANALYSIS_CLASS *>(general_analysis);
    }
    else
    {
        specific_analysis = new ANALYSIS_CLASS(target);
        add_analysis(*specific_analysis, target.get_bdi_index());
    }
    if( specific_analysis->is_invalidated() )
    {
        specific_analysis->do_validation(get_bdi_pool());
    }
    return *specific_analysis;
}
template<class ANALYSED_CLASS, class ANALYSIS_CLASS, BDI_CLASSES_IDENTIFICATION analyser_id>
ANALYSIS_CLASS &CAI_ANALYSER_BASE<ANALYSED_CLASS, ANALYSIS_CLASS, analyser_id>::determine_analysis_for(ANALYSED_CLASS & target)
{
    CAI_ANALYSIS *general_analysis = get_analysis(target.get_bdi_index());
    ANALYSIS_CLASS *specific_analysis = nullptr;
    if( general_analysis )
    {
        specific_analysis = static_cast<ANALYSIS_CLASS *>(general_analysis);
    }
    else
    {
        specific_analysis = new ANALYSIS_CLASS(target);
        add_analysis(*specific_analysis, target.get_bdi_index());
    }
    if( specific_analysis->is_invalidated() )
    {
        specific_analysis->do_validation(get_bdi_pool());
    }
    return *specific_analysis;
}
Financial system
class CAI_FINANCIAL_ANALYSIS : public CAI_ANALYSIS
{
public:
    CAI_FINANCIAL_ANALYSIS (CAI_FACTION & faction);
    virtual void on_pool_add (CAI_BDI_POOL & bdi_pool);

    // Information
    float32 currently_acceptable_balance (); const;
    float32 absolute_acceptable_turn_on_outgoings () const;
    void calculate_spending_value (CAI_BDI_POOL & bdi_pool, float32 & relative_acceptable_balance,
                                   float32 & absolute_turn_on_turn_outgoings,
                                   CAI_FACTION * optional_additional_faction_to_consider_at_war);

private:
    CONST_SAFE_PTR<CAI_FACTION> m_faction;
    float32 m_currently_acceptable_balance;
    float32 m_absolute_acceptable_turn_on_turn_outgoings;

    virtual bool validate (CAI_BDI_POOL & bdi_pool); // Affect: True if things have
    changed as a result
};
bool CAI_FINANCIAL_ANALYSIS::validate(CAI_BDI_POOL &bdi_pool)
{
    // Force update of analysis
    bdi_pool.get_central_bdi_pool().owned_regions_analyser().determine_analysis_for(*m_faction);

    // Generate new information
    float32 temp_selected_acceptable_balance = 0.0f;
    float32 temp_absolute_turn_on_turn_outgoings = 0.0f;

    calculate_spending_value(bdi_pool, temp_selected_acceptable_balance,
                             temp_absolute_turn_on_turn_outgoings, null);

    // Update if needed
    if( temp_selected_acceptable_balance != m_currently_acceptable_balance ||
        temp_absolute_turn_on_turn_outgoings != m_absolute_acceptable_turn_on_turn_outgoings )
    {
        m_currently_acceptable_balance = temp_selected_acceptable_balance;
        m_absolute_acceptable_turn_on_turn_outgoings = temp_absolute_turn_on_turn_outgoings;
        return true;
    }
    return false;
}
Decision domains
- Economy
- Construction
- Diplomacy
- Army composition
- Army deployment / movement
- Technology
- Characters & Skills

AI Subsystems in Campaign
- Financial System
- Construction System
- Diplomacy System
- Task Management System
- Technology Management
- Character Management

World State (Analysers)
Overview

• Introducing the Total War campaign
• An overview of AI systems and the world state
• A consideration of diplomacy
• Tasks and resources
• Profiling and timing
Decision domains

- Economy
- Construction
- Diplomacy
- Army composition
- Army deployment / movement
- Technology
- Characters & Skills

AI Subsystems in Campaign

- Financial System
- Construction System
- Diplomacy System
- Task Management System
- Technology Management
- Character Management

World State (Analysers)
Diplomacy system
Diplomacy system
Diplomacy system
Diplomacy system

Goal generation

Goal 1
Goal 2
Goal 3
Goal n

Personality based priorities

Deal evaluation

Deal to offer
Goal Generation

Goal generation

Goal 1
Goal 2
Goal 3
Goal n

Personality based priorities

Deal evaluation

Deal to offer

@aphrael
Goal generation

- DECLARE_WAR_ON_NEARBY_FACTIONS
- TREATIES_WITH_ENEMIES_OF_ENEMIES
- DEMAND_GIFTS_FROM_NEARBY_FACTIONS
- DECLARE_WAR_ON_FACTIONS_I_DISLIKE
Goal generation

Goal Generators

- DECLARE_WAR_ON_NEARBY_FACTIONS
- TREATIES_WITH_ENEMIES_OF_ENEMIES
- DEMAND_GIFTS_FROM_NEARBY_FACTIONS
- DECLARE_WAR_ON_FACTIONS_I_DISLIKE

List of goals

- War to my bitter enemy Faction1
- War to unfriendly Faction2
- Non aggression pact with Faction3
- Non aggression pact and demand money from Faction 1
- Trade agreement with Faction3
- Trade agreement with Faction1
- Demand 1000 payment from neighbor Faction1
- Demand 200 regular payment from Faction2
- Declare war on Faction 1
- Declare war on Faction 2
Goal generation

// Peace
CAI_GOAL_GENERATORS::ASK_FOR_PEACE (balance_analyser, *this).generate_goals(goals, faction);

// War
CAI_GOAL_GENERATORS::DECLARE_WAR_ON_FACTIONS_I_DISLIKE(balance_analyser, *this).generate_goals(goals, faction);
CAI_GOAL_GENERATORS::OFFER_TO_JOIN_ALLYS_WAR (balance_analyser, *this).generate_goals(goals, faction);
CAI_GOAL_GENERATORS::REQUEST_ALLY_TO_JOIN_WAR (balance_analyser, *this).generate_goals(goals, faction);
CAI_GOAL_GENERATORS::DEMAND_VASSALAGE_OR_CLIENT_STATE (balance_analyser, *this).generate_goals(goals, faction);
CAI_GOAL_GENERATORS::OFFER_VASSALAGE_TO_NEIGHBOURS (balance_analyser, *this).generate_goals(goals, faction);

// Trade
CAI_GOAL_GENERATORS::OBTAIN_TRADE_AGREEMENT (balance_analyser, *this).generate_goals(goals, faction);

// Gifts
CAI_GOAL_GENERATORS::DEMAND_GIFTS_FROM_WEAKER_FACTIONS (balance_analyser, *this).generate_goals(goals, faction);
CAI_GOAL_GENERATORS::OFFER_GIFTS_TO_STRONGER_FACTIONS (balance_analyser, *this).generate_goals(goals, faction);

// Treaties
CAI_GOAL_GENERATORS::TREATIES_WITH_NEIGHBOURS (balance_analyser, *this).generate_goals(goals, faction);
CAI_GOAL_GENERATORS::TREATIES_WITH_ALLIES_OF_ALLIES (balance_analyser, *this).generate_goals(goals, faction);
CAI_GOAL_GENERATORS::UPGRADE_TREATIES (balance_analyser, *this).generate_goals(goals, faction);
CAI_GOAL_GENERATORS::MARRIAGE (balance_analyser, *this).generate_goals(goals, faction);
Goal generation

// Peace
CAI_GOAL_GENERATORS::ASK_FOR_PEACE
(balance_analyser, *this).generate_goals(goals, faction);

// War
CAI_GOAL_GENERATORS::DECLARE_WAR_ON_FACTIONS_I_DISLIKE
(balance_analyser, *this).generate_goals(goals, faction);
CAI_GOAL_GENERATORS::OFFER_TO_JOIN_ALLYS_WAR
(balance_analyser, *this).generate_goals(goals, faction);
CAI_GOAL_GENERATORS::REQUEST_ALLY_TO_JOIN_WAR
(balance_analyser, *this).generate_goals(goals, faction);
CAI_GOAL_GENERATORS::DEMAND_VASSALAGE_OR_CLIENT_STATE
(balance_analyser, *this).generate_goals(goals, faction);
CAI_GOAL_GENERATORS::OFFER_VASSALAGE_TO_NEIGHBOURS
(balance_analyser, *this).generate_goals(goals, faction);

// Trade
CAI_GOAL_GENERATORS::OBTAIN_TRADE_AGREEMENT
(balance_analyser, *this).generate_goals(goals, faction);

// Gifts
CAI_GOAL_GENERATORS::DEMAND_GIFTS_FROM_WEAKER_FACTIONS
(balance_analyser, *this).generate_goals(goals, faction);
CAI_GOAL_GENERATORS::OFFER_GIFTS_TO_STRONGER_FACTIONS
(balance_analyser, *this).generate_goals(goals, faction);

// Treaties
CAI_GOAL_GENERATORS::TREATIES_WITH_NEIGHBOURS
(balance_analyser, *this).generate_goals(goals, faction);
CAI_GOAL_GENERATORS::TREATIES_WITH_ALLIES_OF_ALLIES
(balance_analyser, *this).generate_goals(goals, faction);
CAI_GOAL_GENERATORS::UPGRADE_TREATIES
(balance_analyser, *this).generate_goals(goals, faction);
CAI_GOAL_GENERATORS::MARRIAGE
(balance_analyser, *this).generate_goals(goals, faction);
Goal generation

```cpp
for (const CAI_PERSONALITY_DEAL_GENERATION_GENERATOR_RECORD* generator : deal_generators())
{
    CAI_DIPLOMATIC_GOAL_GENERATOR(*generator).generate_goals(goals, nullptr);
}
```
Goal generation

```c++
for (const CAI_PERSONALITY_DEAL_GENERATION_GENERATOR_RECORD* generator : deal_generators()) {
    CAI_DIPLOMATIC_GOAL_GENERATOR(*generator).generate_goals(goals, nullptr);
}
```
## Goal generators

<table>
<thead>
<tr>
<th>Generator Key</th>
<th>Goal key</th>
<th>Target group</th>
<th>Condition set</th>
</tr>
</thead>
<tbody>
<tr>
<td>BREAK_TREATIES</td>
<td>goal_break_military_access</td>
<td>cai_target_group_known_factions</td>
<td>BREAK_MILITARY_ACCESS_CONDITION_SET</td>
</tr>
<tr>
<td>BREAK_TREATIES</td>
<td>goal_break_non_aggression</td>
<td>cai_target_group_known_factions</td>
<td>BREAK_NON_AGGRESSION_CONDITION_SET</td>
</tr>
<tr>
<td>BREAK_TREATIES</td>
<td>goal_break_trade</td>
<td>cai_target_group_known_factions</td>
<td>BREAK_TRADE_CONDITION_SET</td>
</tr>
<tr>
<td>BREAK_TREATIES</td>
<td>goal_break_offer_regular_payments</td>
<td>cai_target_group_known_factions</td>
<td>BREAK_OFFER_REGULAR_PAYMENTS_CONDITION_SET</td>
</tr>
<tr>
<td>TREATIES_WITH_NEARBY_NONHOSTILES</td>
<td>goal_non_aggression_pact</td>
<td>cai_target_group_neighbours</td>
<td>NON_AGRESSION_PACT_CONDITION_SET</td>
</tr>
<tr>
<td>KICK_DISLIKED_ALLIANCE_MEMBER</td>
<td>goal_kick_coalition_member</td>
<td>cai_target_group_allies</td>
<td>KICK_COALITION_MEMBER_CONDITION_SET</td>
</tr>
<tr>
<td>DEMAND_ANCILLARY_FROM_FRIEND</td>
<td>goal_demand_ancillary</td>
<td>cai_target_group_known_factions</td>
<td>ANCILLARY_CONDITION_SET</td>
</tr>
</tbody>
</table>
## Goal generation: Treaties

<table>
<thead>
<tr>
<th>Goal key</th>
<th>Mandatory treaty key</th>
</tr>
</thead>
<tbody>
<tr>
<td>goal_break_military_access</td>
<td>treaty_components_break_military_access</td>
</tr>
<tr>
<td>goal_break_non_aggression</td>
<td>treaty_components_break_non_aggression</td>
</tr>
<tr>
<td>goal_break_offer_food</td>
<td>treaty_components_break_food_supply_offer</td>
</tr>
<tr>
<td>goal_break_offer_regular_payments</td>
<td>treaty_components_break_payment_regular_offer</td>
</tr>
<tr>
<td>goal_break_trade</td>
<td>treaty_components_break_trade</td>
</tr>
<tr>
<td>goal_demand_ancillary</td>
<td>treaty_components_ancillary_demand</td>
</tr>
<tr>
<td>goal_demand_food</td>
<td>treaty_components_food_supply_demand</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Goal key</th>
<th>Optional treaty key</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>goal_break_military_access</td>
<td>treaty_components_ancillary_demand</td>
<td>2</td>
</tr>
<tr>
<td>goal_break_military_access</td>
<td>treaty_components_payment_offer</td>
<td>1</td>
</tr>
</tbody>
</table>
## Goal generation: Conditions

<table>
<thead>
<tr>
<th>Condition_set_key</th>
<th>Condition_key</th>
<th>Evaluates</th>
<th>Param_faction</th>
<th>owner</th>
<th>status</th>
<th>treaty</th>
<th>stance</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>BREAK_MILITARY_ACCESS_CONDITION_SET</td>
<td>deal_generation_condition_strategic_stance_better_than</td>
<td>false</td>
<td>respondent_recipient</td>
<td>respondent_proposer</td>
<td></td>
<td></td>
<td>strategic_stance_friendly</td>
<td>0</td>
</tr>
<tr>
<td>OFFER_VASSALAGE_CONDITION_SET</td>
<td>deal_generation_condition_target_faction_is_stronger_than_me_by_at_least</td>
<td>true</td>
<td>respondent_recipient</td>
<td>respondent_proposer</td>
<td></td>
<td></td>
<td></td>
<td>0.2</td>
</tr>
<tr>
<td>OFFER_VASSALAGE_CONDITION_SET</td>
<td>deal_generation_condition_has_treaty</td>
<td>false</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Goal generation: Conditions

class CAI_DIPLOMATIC_GOAL_GENERATION_CONDITION_FUNCTIONS :
    public CAI_DEAL_COMPONENT_EVALUATION_FUNCTIONS<CAI_CONDITION_RECORD, GEN_COND_FUNC>
{
    public:

    CAI_DIPLOMATIC_GOAL_GENERATION_CONDITION_FUNCTIONS();

    private:

    void init_map () override;

    static CONDITION_RESPONSE has_treaty (const CAI_FACTION& owner_faction, const PARAMETERS& parameters, bool evaluates_to);
    static CONDITION_RESPONSE has_treaty_with_anyone (const CAI_FACTION& owner_faction, const PARAMETERS& parameters, bool evaluates_to);
    static CONDITION_RESPONSE in_alliance (const CAI_FACTION& owner_faction, const PARAMETERS& parameters, bool evaluates_to);
    static CONDITION_RESPONSE knows (const CAI_FACTION& owner_faction, const PARAMETERS& parameters, bool evaluates_to);
    static CONDITION_RESPONSE neutral_with (const CAI_FACTION& owner_faction, const PARAMETERS& parameters, bool evaluates_to);
    static CONDITION_RESPONSE strategic_stance_better_than (const CAI_FACTION& owner_faction, const PARAMETERS& parameters, bool evaluates_to);
    static CONDITION_RESPONSE target_faction_is_stronger_than_me_by_at_least (const CAI_FACTION& owner_faction, const PARAMETERS& parameters, bool evaluates_to);
};

@aphrael
CONDITION_RESPONSE
CAI_DIPLOMATIC_GOAL_GENERATION_CONDITION_FUNCTIONS::target_faction_is_stronger_than_me_by_at_least
(const CAI_FACTION& owner_faction, const PARAMETERS& parameters, bool evaluates_to)
{
  if (parameters.param_faction == nullptr)
  
    return CONDITION_RESPONSE_IRRELEVANT;

  const CAI_FACTION_ALLIANCE_STRENGTH_ANALYSIS& strength_analysis_us =
    faction_alliance_strength_analyser().determine_analysis_for(deconst(owner_faction));
  const CAI_FACTION_ALLIANCE_STRENGTH_ANALYSIS& strength_analysis_them =
    faction_alliance_strength_analyser().determine_analysis_for(deconst(*param_faction));

  if (strength_analysis_us.strength() != 0 && (strength_analysis_them.strength() -
    strength_analysis_us.strength()) / strength_analysis_us.strength() > param_value) == evaluates_to)
  
    return CONDITION_RESPONSE_HOLDS;

  return CONDITION_RESPONSE_DOES_NOT_HOLD;
}
Goal generation: Target groups

class CAI_DIPLOMATIC_GOAL_GENERATION_TARGET_GROUP_FUNCTIONS :
    public CAI_DEAL_COMPONENT_EVALUATION_FUNCTIONS<CAI_TARGET_GROUP_RECORD, TARGET_FUNC>
{
    public:
        CAI_DIPLOMATIC_GOAL_GENERATION_TARGET_GROUP_FUNCTIONS();
    private:
        void init_map() override;
        static void allies(CAI_BDI_POOL& bdi_pool, CAI_FACTION& faction, CA_STD::VECTOR<const CAI_FACTION*>&& factions);
        static void allies_of_allies(CAI_BDI_POOL& bdi_pool, CAI_FACTION& faction, CA_STD::VECTOR<const CAI_FACTION*>&& factions);
        static void disliked_factions(CAI_BDI_POOL& bdi_pool, CAI_FACTION& faction, CA_STD::VECTOR<const CAI_FACTION*>&& factions);
        static void enemies(CAI_BDI_POOL& bdi_pool, CAI_FACTION& faction, CA_STD::VECTOR<const CAI_FACTION*>&& factions);
        static void factions_with_treaties(CAI_BDI_POOL& bdi_pool, CAI_FACTION& faction, CA_STD::VECTOR<const CAI_FACTION*>&& factions);
        static void known_factions(CAI_BDI_POOL& bdi_pool, CAI_FACTION& faction, CA_STD::VECTOR<const CAI_FACTION*>&& factions);
        static void neighbours(CAI_BDI_POOL& bdi_pool, CAI_FACTION& faction, CA_STD::VECTOR<const CAI_FACTION*>&& factions);
        static void horde_neighbours(CAI_BDI_POOL& bdi_pool, CAI_FACTION& faction, CA_STD::VECTOR<const CAI_FACTION*>&& factions);
};
Goal generation: Target groups

```cpp
void CAI_DIPLOMATIC_GOAL_GENERATION_TARGET_GROUP_FUNCTIONS::neighbours(
    CAI_BDI_POOL& bdi_pool, CAI_FACTION& faction, CA_STD::VECTOR<const CAI_FACTION*>&& factions)
{
    CAI_NEIGHBOURING_FACTIONS_ANALYSIS& neighbour_analysis =
        bdi_pool.get_central_bdi_pool().neighbouring_factions_analyser().determine_analysis_for(faction);

    factions.insert(factions.end(), neighbour_analysis.neighbours_begin(), neighbour_analysis.neighbours_end());
}
```
void CAI_DIPLOMATIC_GOAL_GENERATOR::generate_goals(CA_STD::VECTOR<CAI_DIPLOMATIC_GOAL>& goals)
{
    const CAI_DIPLOMATIC_GOAL_GENERATION_GOAL_TEMPLATES& deal_generation_goals =
        cai_diplomatic_goal_bases().deal_generation_goals(*m_generator_record);
    bool conditions_satisfied = true;
    if ((*goal_template).m_condition_data_vector != nullptr)
    {
        for (const CAI_FACTION* target : target_factions)
        {
            for (CAI_DIPLOMATIC_GOAL_GENERATION_CONDITION_DATA condition : (*goal_template).m_condition_data_vector)
            {
                PARAMETERS params(&(*cond_itr).m_param_status->m_faction_status, (*cond_itr).m_param_treaty,
                                    (*cond_itr).m_param_stance,
                                    (*cond_itr).m_param_value);
                CONDITION_RESPONSE response = (*cond_itr).m_func(*owner_faction, nullptr, params, (*cond_itr).m_evaluates_to);
                if (response == CONDITION_RESPONSE_DOES_NOT_HOLD)
                {
                    conditions_satisfied = false;
                    break;
                }
            }
        }
    }
    if (conditions_satisfied)
    {
        goals.emplace_back(*(*goal_template).m_goal, m_faction, *target, values, m_failure_timeout, m_priority_base);
    }
}
Goal generation

Goal 1

Goal 2

Goal 3

Goal n

Deal evaluation

Deal to offer

Personality based priorities

Goal Prioritisation

@aphrael
Goal prioritisation

List of goals

- War to my bitter enemy Faction1
- War to unfriendly Faction2
- Non aggression pact with Faction3
- Non aggression pact & demand money from Faction 1
- Trade agreement with Faction3
- Trade agreement with Faction1
- Demand 1000 payment from neighbor Faction1
- Demand 200 regular payment from Faction2
- Make peace, and become my vassal to Faction1
- Make peace, and become my vassal to Faction2

Prioritisation

- (20) War to my bitter enemy Faction1
- (10) War to unfriendly Faction2
- (5) Non aggression pact with Faction3
- (2) Non aggression pact & demand money from Faction 1
- (10) Trade agreement with Faction3
- (0) Trade agreement with Faction1
- (8) Demand 1000 payment from neighbor Faction1
- (6) Demand 200 regular payment from Faction2
- (1) Make peace, and become my vassal to Faction1
- (3) Make peace, and become my vassal to Faction2
Goal generation

Goal 1
Goal 2
Goal 3
Goal n

Personality based priorities

Deal evaluation

Deal to offer
Deal evaluation

Goal 1

- Mandatory treaty 1
- Mandatory treaty 2
- Optional treaty 1
- Optional treaty 2

Stance Factor: 6
Diplomatic Factor: 2
Respect Factor: -4
Economic Factor: 0
Strategic Factor: 1

@aphrael
Evaluation factors

- Stance Factor
- Respect Factor
- Strategic Factor
- Economic Factor
- Diplomatic Factor
Evaluation factors

Stance Factor
Respect Factor
Economic Factor
Diplomatic Factor
Strategic Factor
Evaluation factors

- Stance Factor
- Respect Factor
- Strategic Factor
- Economic Factor
- Diplomatic Factor
Evaluation factors

- Stance Factor
- Respect Factor
- Strategic Factor
- Economic Factor
- Diplomatic Factor
Evaluation factors

- Stance Factor
- Respect Factor
- Strategic Factor
- Economic Factor
- Diplomatic Factor
Evaluation factors: Criteria

Stance Factor
STANCE_ALLIANCE_MEMBER_TO_PROPOSER
STATEGIC_RECENT_WAR_PENALTY_TO_PEACE

Economic Factor
ECONOMIC_TRADE

Strategic Factor

Respect Factor
RESPECT_FACTION_TO_ALLIANCE_MEMBERS
RESPECT_DEFAULT

Diplomatic Factor
DIPLOMACY_CONSEQUENCES_OF_BREAK_ALL_TREATY_EVENTS
DIPLOMACY_CONSEQUENCES_OF_JOIN_FACTIONS_WARS
## Evaluation factors

<table>
<thead>
<tr>
<th>Factor</th>
<th>Criterion type</th>
</tr>
</thead>
<tbody>
<tr>
<td>diplomatic_factor</td>
<td>DIPLOMACY_CONSEQUENCES_OF_EVENT</td>
</tr>
<tr>
<td>strategic_factor</td>
<td>STRATEGIC_STRENGTH_OUR_AND_VASSALS_TO_TOTAL_IF_POSITIVE_BALANCE</td>
</tr>
<tr>
<td>strategic_factor</td>
<td>STRATEGIC_ALLIANCE_SIZE_PENALTY</td>
</tr>
<tr>
<td>respect_factor</td>
<td>RESPECT_DEFAULT</td>
</tr>
<tr>
<td>economic_factor</td>
<td>ECONOMIC_ZERO</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Treaty component</th>
<th>Criterion type</th>
<th>negate</th>
</tr>
</thead>
<tbody>
<tr>
<td>break_non_aggression</td>
<td>STANCE_DEFAULT</td>
<td>false</td>
</tr>
<tr>
<td>break_non_aggression</td>
<td>STRATEGIC_STRENGTH_OUR_AND_VASSALS_TO_TOTAL_IF_POSITIVE_BALANCE</td>
<td>true</td>
</tr>
<tr>
<td>break_non_aggression</td>
<td>DIPLOMACY_CONSEQUENCES_OF_EVENT</td>
<td>false</td>
</tr>
<tr>
<td><strong>break_non_aggression</strong></td>
<td>ECONOMIC_ZERO</td>
<td>false</td>
</tr>
<tr>
<td>break_non_aggression</td>
<td>RESPECT_DEFAULT</td>
<td>false</td>
</tr>
</tbody>
</table>
class CAI_DEAL_COMPONENT_EVALUATION_CRITERIA_FUNCTIONS :
    public CAI_DEAL_COMPONENT_EVALUATION_FUNCTIONS<CAI_DIPLOMACY_DEAL_EVALUATION_CRITERION_TYPE_RECORD, EVAL_FUNC>
{

public:
    CAI_DEAL_COMPONENT_EVALUATION_CRITERIA_FUNCTIONS();

private:
    void init_map() override;

static float32 character_default_demand (TREATY_COMPONENT_RECORD& record, COMPONENT_DATA& component_data, PARAMS params);
static float32 consequences_of_join_alliances_wars (TREATY_COMPONENT_RECORD& record, COMPONENT_DATA& component_data, PARAMS params);
static float32 economic_regular_demand (TREATY_COMPONENT_RECORD& record, COMPONENT_DATA& component_data, PARAMS params);
static float32 economic_regular_offer (TREATY_COMPONENT_RECORD& record, COMPONENT_DATA& component_data, PARAMS params);
static float32 consequences_of_break_all_treaties (TREATY_COMPONENT_RECORD& record, COMPONENT_DATA& component_data, PARAMS params);
static float32 respect_default (TREATY_COMPONENT_RECORD& record, COMPONENT_DATA& component_data, PARAMS params);
static float32 respect_faction_to_alliance_members (TREATY_COMPONENT_RECORD& record, COMPONENT_DATA& component_data, PARAMS params);
static float32 stance_faction_to_alliance_members (TREATY_COMPONENT_RECORD& record, COMPONENT_DATA& component_data, PARAMS params);
static float32 strategic_alliance_size_penalty (TREATY_COMPONENT_RECORD& record, COMPONENT_DATA& component_data, PARAMS params);
static float32 strategic_recent_peace_penalty (TREATY_COMPONENT_RECORD& record, COMPONENT_DATA& component_data, PARAMS params);
static float32 strategic_victory_region (TREATY_COMPONENT_RECORD& record, COMPONENT_DATA& component_data, PARAMS params);
};
Evaluation factors: Criteria functions

```cpp
float32 CAI DEAL COMPONENT EVALUATION CRITERIA FUNCTIONS::
consequences_of_break_all_treaties(TREATY COMPONENT RECORD& record, COMPONENT DATA& component_data, PARAMS params)
{
    const FACTION& evaluator_faction = component_data.m_evaluator().get_campaign_faction();
    const FACTION& opponent_faction = component_data.m_opponent()->get_campaign_faction();
    const DIPLOMACY::DEAL::DEALS & deals = evaluator_faction.campaign_model().diplomacy().active_deals();
    CA_STD::VECTOR<CAI LOG MANAGER::SPECULATIVE COMPONENT> components;
    for (const DIPLOMACY::DEAL& deal : deals)
    {
        if (DIPLOMACY::faction_involved_in_deal(evaluator_faction, deal) &&
            DIPLOMACY::faction_involved_in_deal(opponent_faction, deal))
        {
            for (const DIPLOMACY::DEAL_COMPONENT& deal_component : deal.components())
            {
                components.push_back(CAI LOG MANAGER::SPECULATIVE COMPONENT(evaluator_faction, opponent_faction,
                                     &deal_component.component(), DEAL COMPONENT CHANGE TYPE::BROKEN));
            }
        }
    }
    return diplomatic_value(bdi_pool, components, component_data.m_evaluator(), proposer);
}
```
Deal evaluation - value of a treaty component

```cpp
float32 CAI_DEAL_EVALUATION::component_value(CAI_BDI_POOL& bdi_pool,
    const DIPLOMACY::NEGOTIATION_STATE& negotiation_state, CAI_FACTION& evaluator, CAI_FACTION& to,
    CAI_FACTION &proposer, const CAMPAIGN_DIPLOMACY_TREATY_COMPONENT_RECORD& record, int32 value)
{
    CAI_DEAL_COMPONENT deal_component(bdi_pool.get_central_bdi_pool(), negotiation_state,
        evaluator, to, proposer, record, value);

    float32 total_component_value = 0.0f;
    const CAI_DIPLOMACY_DEAL_EVALUATION_FACTOR_TYPES_TABLE& factors = deal_evaluation_factor_types_table();
    for (const CAI_DIPLOMACY_DEAL_EVALUATION_FACTOR_TYPE_RECORD *factor : factors)
    {
        total_component_value += deal_component.deal_evaluation_factor(*factor);
    }

    return total_component_value;
}
```
float32 CAI_DEAL_COMPONENT::deal_evaluation_factor(const DEAL_EVALUATION_FACTOR_TYPE_RECORD& factor) {
    DEAL_CRITERIA_MAP::const_iterator criteria_map_itr = m_deal_evaluation_criterias_by_factor->find(&factor);
    float32 sum = 0.0f;
    for (const CRITERION_BASE& criteria_base : criteria_map_itr->second) {
        COND_FUNC condition = cai_deal_component_evaluation_conditions.func(criteria_base.m_condition);
        if (condition == nullptr ||
            condition(m_bdi_pool, m_component_data.m_evaluator, m_recipient(), &m_proposer(), m_params)) {
            EVAL_FUNC eval = cai_deal_component_evaluation_criteria.func(criteria_base.m_criterion_type);
            sum += eval(m_bdi_pool, *m_record(), m_component_data, m_proposer(), m_params) *
                (criteria_base.m_should_negate ? -1.0f : 1.0f);
        }
    }
    return sum;
}

Diplomacy system

Goal generation

Goal 1
Goal 2
Goal 3
Goal n

Personality based priorities

Deal evaluation

Deal to offer
Overview

- Introducing the Total War campaign
- An overview of AI systems and the world state
- A consideration of diplomacy
- Tasks and resources
- Profiling and timing
Decision domains

- Economy
- Construction
- Diplomacy
- Army composition
- Army deployment / movement
- Technology
- Characters & Skills

AI Subsystems

- Financial System
- Construction System
- Diplomacy System
- Task Management System
- Technology Management
- Character Management

World State (Analysers)
Task management system
Task management system

- Task Generation
- Resource Allocation
- Resource Coordination
Task management system

- Task Generation
- Resource Allocation
- Resource Coordination
Task generation

- Attack region
- Attack force
- Recruit new army
- Raid region
- Defend region
- Embed agent
- Recruit in region
Task generation: attack region

```cpp
// Task for performing simple region attack
class CAI_TASK_ATTACK_REGION : public CAI_TASK
{
public:
    CAI_TASK_ATTACK_REGION(CAI_REGION &region, CAI_TASK_RESOURCE_REALLOCATION_BASE_POLICY base_resource_realloc_policy);

    virtual void on_pool_add(CAI_BDI_POOL &bdi_pool);

    // Resource Allocation
    virtual CAI_TASK_REQUIREMENTS * generate_requirements_information(CAI_BDI_POOL &bdi_pool, const CAI_STANCE_INFORMATION &stance_info);
    virtual CAI_TASK_TARGET_OBJECTIVE generate_requirements_objective() const;
    virtual void populate_stance_information(CAI_BDI_POOL &bdi_pool, CAI_STANCE_INFORMATION_VECTOR &stance_info);

    virtual CAI_FACTION * primary_task_target_faction();
    virtual const CAI_FACTION * primary_task_target_faction() const;

private:
    // Operation
    virtual void execute(CAI_BDI_POOL &bdi_pool);

    CONST_SAFE_PTR<CAI_REGION> m_region;
};
```
CAI_TASK_REQUIREMENTS* CAI_TASK_ATTACK_REGION::generate_requirements_information

(CAI_BDI_POOL &bdi_pool, const CAISTANCE_INFORMATION &/*stace_info*/)

{

CAI_TASK_REQUIREMENT_TARGET target_info = generate_requirements_target();
CAI_TASK_REQUIREMENT_BOUNDS army_strength_bounds(0, 0, 0);
CAI_TASK_REQUIREMENT_BOUNDS time_bounds(0, 0, 0);

CAI_FRACTION &our_faction = bdi_pool.get_faction_bdi_pool() -> get_faction();
CAI_RESOURCE_MOBILE_MILITARY_STRENGTH required_army_strength;
CAI_MILITARY_STRENGTH_ANALYSER &military_strength_analyser = bdi_pool.military_strength_analyser();

for (auto mobile : m_region->mobiles_in_region())
{
    if (mobile->is_an_army())
    {
        required_army_strength += military_strength_analyser.determine_analysis_for(*mobile).absolute_strength();
    }
}

if (m_region->has_settlement())
{
    const CAI_GARRISONABLE_MILITARY_STRENGTH_ANALYSIS& region_strength_analysis =
    military_strength_analyser.determine_analysis_for(*m_region->get_settlement());
    required_army_strength += region_strength_analysis.citizenry_strength_land();
}
card32 recommended_minimum = ca_round_to_card(static_cast<float32>(recommended_minimum) * 
osfaction.faction_personality().strategic_component().enemy_strength_modifier());

army_strength_bounds.m_minimum = recommended_minimum;
army_strength_bounds.m_recommended = recommended_minimum * 2;
army_strength_bounds.m_maximum = recommended_minimum * 8;

// Attritional Effects
if( m_region->has_settlement() )
{
    if(!our_faction.is_immune_to_attrition(*attrition_record) )
    {
        float32 attrition_multiplier = m_region->get_settlement()->is_fortified() ? 2.5f : 1.5f;
army_strength_bounds.m_minimum *= attrition_multiplier;
army_strength_bounds.m_recommended *= attrition_multiplier;
army_strength_bounds.m_maximum *= attrition_multiplier;
    }
}
return new CAI_TASK_REQUIREMENTS(*this, target_info, army_strength_bounds, time_bounds,
    CAI_ALLOCATION_ACCEPTANCE_PREFERENCE_AT_LEAST_MINIMUMS, CAI_ARMY_PREFERENCE_ALWAYS_REQUIRED,
    CAI_AGENT_PREFERENCE_AT_LEAST_ONE_IF_POSSIBLE, CAI_TIMING_PREFERENCE_ENFORCE,
    CAI_FORCE_RECRUITMENT_REQUIREMENT::UNSPECIFIED,
    CAI_FORCE_RECRUITMENT_REQUIREMENT::UNSPECIFIED );
void generate_attack_neighbouring_war_regions_tasks(const CAI_ATTACK_REGION_TASK_SPECIFICATION::TASK_TYPE_MAP_NCC &, CAI_ATTACK_REGION_TASK_SPECIFICATION::TASK_INSTANTIATION_INFO_PRIORITY_DATA_VECTOR &new_tasks_with_priority, CAI_TASK_GENERATOR_CONTROL_DATA &task_generator_control_data, BDI_COMPONENT_ARRAY &)
{
    CAI_DIPLOMATIC_ANALYSIS &da_us = task_generator_control_data.diplomatic_analyser().determine_analysis_for(us);
    CA_STD::UNORDERED_MAP<card32, CAI_REGION *> regions_to_attack;
    std::for_each(task_generator_control_data.get_faction().regions_begin(),
        task_generator_control_data.get_faction().regions_end(),
        [&us, &da_us, &regions_to_attack](CAI_REGION *region)
        {
            std::for_each(region->neighbours_begin(), region->neighbours_end(), [&](CAI_REGION_BOUNDARY *boundary)
            {
                CAI_REGION &other = boundary->borders(*region);
                if( !other.is_abandoned() && other.owner() != nullptr && da_us.has_war_with(other))
                {
                    regions_to_attack.insert(CA_STD::make_pair(other.get_bdi_index(), &other));
                }
            });
        });
    for(CA_STD::UNORDERED_MAP<card32, CAI_REGION *>::ITERATOR itr_region = regions_to_attack.begin(), itr_region_end = regions_to_attack.end(); itr_region != itr_region_end; ++itr_region)
    {
        new_tasks_with_priority.push_back(CAI_ATTACK_REGION_TASK_SPECIFICATION::TASK_INSTANTIATION_INFO_PRIORITY_DATA (((*itr_region).second, 1.0f));
    }
}
Task management system
Task management system
Resource allocation

Tasks Mobiles

- Attack region
- Recruit new army
- Defend region
- Attack force
- Raid region
- Embed agent
- Recruit in region

Resource Mobiles

- Army 1
- Army 3
- Agent 1
- Navy 1
- Army 2
Task management system
Resource allocation

**Tasks Mobiles**
- Attack region
- Recruit new army
- Defend region
- Recruit in region
- Attack force
- Raid region
- Embed agent

**Resource Mobiles**
- Army 1
- Army 3
- Agent 1
- Army 2
- Navy 1

@aphrael
Task management system
Monte Carlo Tree Search
With upper confidence bound for trees, UCT
Monte Carlo Tree Search
With upper confidence bound for trees, UCT

Selection
Expansion
Simulation (Playout)
Backpropagation

@aphrael
Task management system

- Task Generation
- Resource Allocation
- Resource Coordination
Task management system
Resource allocation - MCTS
Task management system
Resource allocation - MCTS
Task management system

- Task Generation
- Resource Allocation
- Resource Coordination
Task management system
Resource coordination

Tasks
- Attack region
- Attack force
- Recruit new army
- Raid region
- Embed agent
- Recruit in region

Resource Mobiles
- Army 1
- Army 2
- Army 3
- Navy 1
- Agent 1

Tasks
- Embed agent 1 with Army 1
- Attack Region 1 with Army 1
- Recruit new Army 2 in Region 3
- Defend Region 2 with Army 3 and 4
Attack Region 1 with Army 1
Defend Region 2 with Army 3 and 4
Recruit new Army 2 in Region 3
Embed agent 1 with Army 1
Attack region
Defend region
Recruit new army
Raid region
Embed agent
Recruit in region
Resource Mobiles
Army 1
Army 3
Agent 1
Army 2
Navy 1
Defend Region 2 with Army 3 and 4
Task management system
Resource coordination

Node Types

Target nodes
- Own Army
- Their Army
- Own Settlement
- Their Settlement
- Their Agents

Action nodes
- Agent Bolster Actions
- Agent Hindering Actions
- Attack Actions
- Siege/ Blockade/ Assault
- Garrison
- Move
Task management system

Resource coordination

Target nodes
- Own Army
- Their Army
- Own Settlement
- Their Settlement
- Their Agents

Node Types
- Target : Enemy Army 1
- Action nodes:
  - Agent Bolster Actions
  - Agent Hindering Actions
  - Attack Actions
  - Siege/ Blockade/ Assault
  - Garrison
  - Move

@aphrael
Root node

Target: enemy army E2
Target: own army O1
Target: enemy settlement S1
Target: enemy settlement S1
Target: enemy army E1
Target: own army O2
Target: enemy settlement S1
Target: enemy army E2
Agent 1 action
Agent 2 action
Attack
Siege

Embed Agent 1
Embed Agent 2
Embed Agent 2
Embed Agent 2

Resource coordination MCTS

Task management system

@aphrael
Task management system
Resource coordination MCTS

Root node

- Target: own army O1
  - Embed Agent 1
    - Target: own army O2
      - Embed Agent 2
      - Target: enemy settlement S1
    - Embed Agent 2
- Target: enemy army E1
  - Agent 2 action
- Attack
  - Agent 1 action
- Siege
  - Target: enemy army E2

@aphrael
Embed Agent 1
Target: own army O1

Embed Agent 2
Target: enemy settlement S1

Agent 2 action

Attack

Target: enemy army E1

Agent 1 action

Siege

Target: enemy army E2

Embed Agent 2
Target: own army O2

Embed Agent 2
Target: enemy settlement S1
<table>
<thead>
<tr>
<th>Resource coordination</th>
<th>Optimisations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pruning</td>
<td></td>
</tr>
<tr>
<td>Sub-phases</td>
<td></td>
</tr>
<tr>
<td>Ordering</td>
<td></td>
</tr>
<tr>
<td>Path caching</td>
<td></td>
</tr>
<tr>
<td>Spatial partitioning</td>
<td></td>
</tr>
</tbody>
</table>
Resource coordination

Optimisations

- Targets: unreachable
- Targets: attacks unsuccessful
- Targets: unsuccessful agent actions.
Resource coordination
Optimisations

• Targets: unreachable
• Targets: attacks unsuccessful
• Targets: unsuccessful agent actions.
Resource coordination
Optimisations

Sub-phases

- Sabotage Target 1
  - Sabotage Target 2
    - Attack Target 1
      - Garrison Target 3
    - Garrison Target 3
  - Attack Target 1
    - Garrison Target 3
  - Sabotage Target 2
    - Garrison Target 3
  - Garrison Target 3
  - Sabotage Target 2
    - Garrison Target 3
  - Garrison Target 2
Resource coordination
Optimisations

Sub-phases

Embed agents

Army targets

Agent targets

Garrison

Move
Resource coordination
Optimisations

Node Ordering

- Enemy Target
  - Agent Action 1
  - Agent Action 2
  - Attack
  - Attack
- Agent Action 2
- Attack
- Agent Action 1
- Attack
Resource coordination
Optimisations

Spatial Partitioning

Army 1
Army 2
Army 3
Army 4

Enemy Army
Our Army
Target

@aphrael
Task Management System

Summary

- Task Allocation and Coordination using MCTS
  - Customisable
  - Good exploitation vs exploration tradeoff
  - Anytime
- Optimisations: Pruning, Lazy evaluation, Spatial partitioning
Overview

- Introducing the Total War campaign
- An overview of AI systems and the world state
- A consideration of diplomacy
- Tasks and resources
- Profiling and timing
Time is the master

- 269 Regions
- 139 Factions
- 1-3 Settlements
- 1-5 Armies
- 0-3 Agents
Time is the master

- 130 Factions
- 1-3 Settlements
- 1-5 Armies
- 0-3 Agents
Using the cycles

- Caching
- Planners
Using the cycles

- Caching
- Long-term planners
- Quadratic => Cubic
- Profiling
Getting performance data

- Sleepy
- Telemetry collection
Getting performance data

- Sleepy
- Telemetry collection
- Chrome tracing
- You can’t control what you can’t measure
Generating performance data

- “Oh, that’s weird…”
- Data series over time
Generating performance data

- “Oh, that’s weird…”
- Data series over time
- Autotesting
- Step-changes
Budgets

- Varying durations
- Hardware specific
Budgets

- Varying durations
- Hardware specific
- Caps
- Continuous interference
Dividing the time

- Factions versus components
- Component versus component
Dividing the time

- Factions versus components
- Component versus component
- Diplomacy
- Scalability not always availability
Doing less

• Do the minimum
• Build and repair
Doing less

- Do the minimum
- Build and repair
- Do what is observable
- Avoid what is silly
Evading responsibility

- Let the user decide
- Not everything scales (yet)
- More time for tasks
Overview

- Introducing the Total War campaign
- An overview of AI systems and the world state
- A consideration of diplomacy
Overview

• Introducing the Total War campaign
• An overview of AI systems and the world state
• A consideration of diplomacy
• Tasks and resources
• Profiling and timing
• One more thing...
ONE OF EUROPE’S LONGEST-STANDING AND LARGEST STUDIOS
STATE-OF-THE-ART FACILITIES IN THE UK AND BULGARIA
MULTI-AWARD WINNING HERITAGE OF AAA TITLES

JOIN US @CAGAMES
CREATIVE-ASSEMBLY.COM
Join us?

• Tools programmer – NEW IP, console
• Engine programmer – NEW IP, console
• Backend developer – Total War Arena
Join us?

- Tools programmer – NEW IP, console
- Engine programmer – NEW IP, console
- Backend developer – Total War Arena
- Engine programmer – Total War Arena
- Tools programmer – Total War
- Engine programmer – Total War
- Campaign AI programmer – Total War
Thank you!
Any questions?