Medieval Warfare

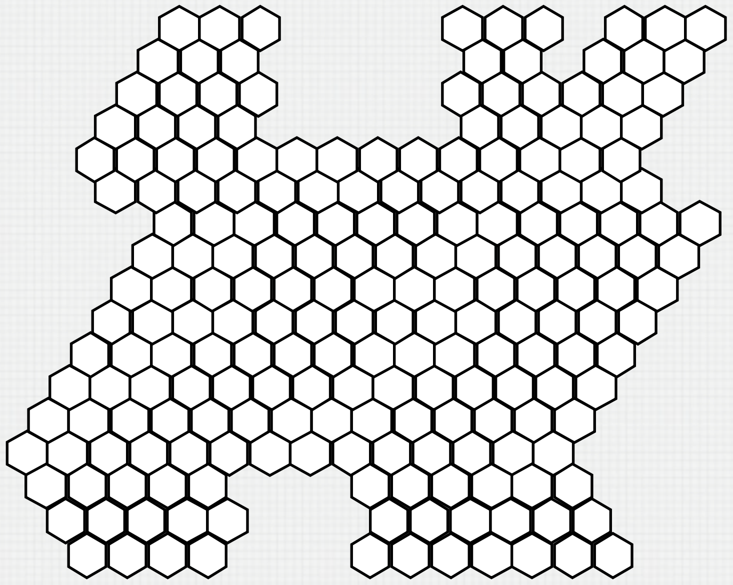
Jörg Kienzle

Game Overview

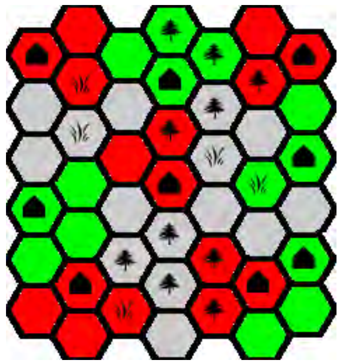
- Turn-based, multiplayer, resource gathering, strategic game
- Players start with a preset amount of land
- Each region controls a village
- Villages can train villagers, that can take over other land tiles or gather wood
- Goal: take over the entire island

Island Map

- At least 300 hexagon tiles
- Any shape, surrounded by sea tiles
- Tiles colored with #players + 1 colors
  - One color represents neutral territory
  - Regions with less than 3 tiles are always neutral
- 20% of tiles contain trees
- 10% of tiles contain meadows



Villages (1)



- Each region of at least 3 tiles always contains a (randomly placed) village
- Each village has a treasury of gold and a wood pile
- At the start of each turn, each land tile generates 1 gold for the village it belongs to
  - Meadow tiles generate 2 gold, tiles with a tree don't generate any gold
- Then, the village has to pay the wage of all its villagers. If a village has insufficient funds to pay his villagers, all the villagers of the region under control of the village perish

Villages (2)

- Villages can be upgraded
- Hovel: initial state (no cost)
  - Can recruit peasants and infantry
- Town: 8 wood
  - Can recruit peasants, infantry and soldiers
  - Can build towers
- Fort: built town and 8 wood
  - Can recruit peasants, infantry, soldiers and knights
  - Can only be invaded by a knight

Villagers

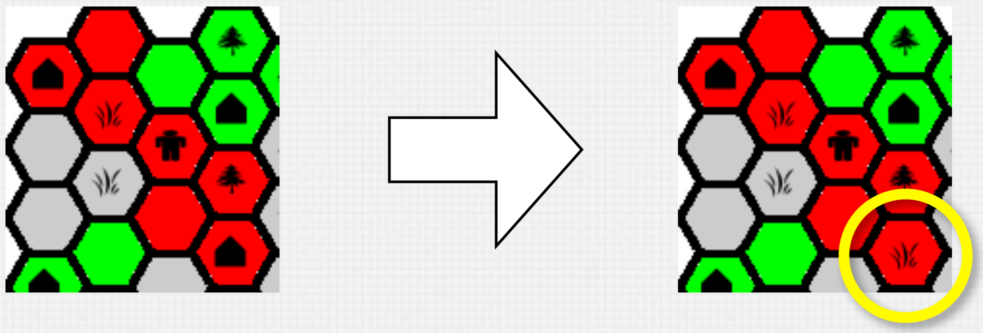
- A village can spend gold to train a villager
  - Peasant: 10 gold, upkeep 2
    - Can't invade enemy territory
    - Can cultivate meadows
  - Infantry: 20 gold, upkeep 6
    - Can't invade villages
  - Soldier: 30 gold, upkeep 18
  - Knight: 40 gold, upkeep 54
    - Can't ride through the forest
    - Tramples meadows
    - Won't do any labor
- Upgrading / combining villagers is possible

Moving Villagers

- A villager commands the tile it is on, as well as the 6 adjacent tiles
- A tile can only hold one villager, structure or tree at a time
- Each turn, a villager can move to any place on his region provided there is a path leading from his current position to the destination position, until he performs one of the following actions:
  - Acquiring New Land
  - Gathering Wood
  - Clearing a Tombstone
  - Cultivating a Meadow
  - Building a Road

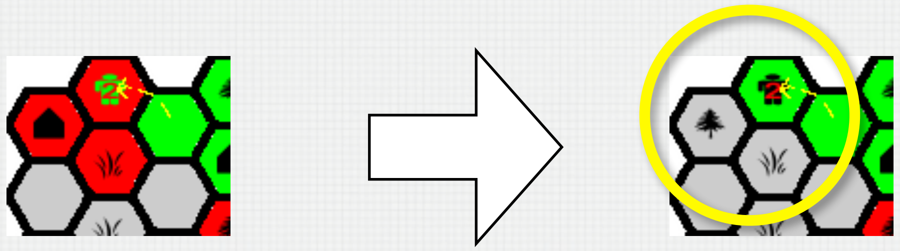
Invading (1)

- Peasants can only acquire neutral land
  - The color of the tile changes
  - If there is a tree on the tile, wood is collected
  - If two regions of the same color are connected, the two villages "join"
    - The most advanced village is kept (or else the one commanding the biggest region), and the resources are joined
    - The other village is removed



Invading (2)

- Villagers of the rank of infantry or higher can invade enemy territory
  - An infantry can only invade enemy territory if it is not protected (1 hex distance) by an infantry of equal rank or higher
  - If a region can no longer support a village after being invaded, the village is turned into a tree and the land converted to neutral territory



Invading Villages

- A village can be invaded by a soldier or knight
  - In this case, all the gold and wood of the village is transferred to the village of the invader
  - The village is destroyed (and a new hovel is recreated somewhere else, if the remaining region is big enough)

Structures

- A town or fort can build a tower
- Towers cost 5 wood
- Once built, towers can not be moved
- Towers act just like an immobile soldier, but with no upkeep

Turn Overview

1. Tree Growth Phase
2. Player Phase: For each player (in predetermined order) iterate through:
    1. Tombstone Phase
    2. Build Phase
    3. Income Phase
    4. Payment Phase
    5. Death Phase
    6. Move & Purchase Phase

Additional Requirements

- UI Assistance for looking at your villages / army
  - Make it playable!
- Loadable or Random Islands
- Distribution
  - Game Server
  - Three players should be able to play against each other on three different machines connected over a network
- Saving
  - You must allow players to save the current game state in order to continue playing at a later time

Project Milestones

- Final grade divided into
  - 3% for the user interface sketch (mid October)
  - 15% for the requirements models (late November)
  - 12% for the design models (early January)
  - 15% for the demo (early March)
  - 20% for the acceptance test (April)
- Groups of maximum 5 students
  - Same grade for all members of a group

User Interface Sketch

- Prepare a sketch (hand drawn or printout) of the main screens of your application
- Should allow the player to trigger all functionality described in the requirements document
  - Interaction with the game server
  - How does a player see the island?
  - How does he control his villages?
  - How does he move his villagers?
  - How does he build structures?
  - How does he build upgrades?
  - How are functions such as saving accessed?

Requirements Models

- Requirement models unambiguously specify the functionality that your game design/ implementation needs to provide
  - Use case model, to specify interactions with the system
  - Concept model, to specify conceptual game state stored within the system
  - Environment model, to specify the interface that the system provides to the environment
  - Operation model, to specify the effects of individual system interactions
  - Protocol model, to specify the supported system interaction scenarios

Design Models

- Design models that provide a detailed blueprint the structure (classes) and behaviour (methods) of your implementation
  - Design class model
  - Interaction model
- Focus is only on the part of the design that deals with the game state / the game rules / moves
  - No graphics-related classes
  - No network / communication-related classes

Development Environment

- Whatever programming language you like
  - Must be object-oriented
  - We will support
    - Java / Minueto
- The demo and acceptance Test will be held either in the Trottier building
  - Bring your laptops / PDAs / game consoles / desktops, if necessary

DEMO

- I will provide you with a list of functionalities that you need to demo
  - You are allowed to demo more
  - You run the show, we sit there and observe
- 80% of the grade is based on correct implementation of the requested functionality (and no crashes / visible bugs)
- 5% for the presentation quality
- 15% for additional functionality

Maintenance Phase

- After the demo week, there will be some slight changes to the game rules.
  - Simulates “real-life” software development
- Write structured, modular, extensible code!

Acceptance Test

- WE are in control! We run the show, you sit and observe.
- We'll use your software, trying to detect bugs / wrong implementations of the game rules
- We will test for ALL the functionality specified in the requirements
- Correct (and playably fast) implementation
  ⇒ A- (80%)
- Additional points for ease-of-use, coolness, innovation, additional features, PLAYABILITY (20%)

Flexibility

- Change properties of units
- Change movement rules
- Add new units / structures
- Add new upgrades
- Add new resources
- Add “races”
- Make the game “real-time”

- Document your changes and discuss them with me
- No “last minute” changes
  (in the Winter semester, no big changes will be accepted)

Some Suggestions

- Start working early.
- First strive for a simple, correct implementation. Later (if there is time enough), add more sophistication.
- Always keep the deadlines in mind. For the demo you must have a functional, convincing application. Do not make big changes on the day that precedes the demo or the acceptance test (or else be sure to have a functional copy on a safe backup...)
- Keep everyone in the group in “a good mood”.
- Come up with an initial architecture of the application, then assign responsibilities to group members. Have regular group meetings to consolidate your work.
- Testing takes time.
- Plan for the unpredictable!
- Start working early!

Questions?