Software Requirements Specification

Hanoi†

† Development code name, not final.

Contents

Summary

Hanoi is a turn-based board game that simulates territory control and the **indirect** combat of opposing armies. The game can be played on a variety of board layouts ranging from square grids of various sizes through *Risk*® or *Pandemic*™ style disjoint maps.

Central to gameplay are the concepts of *critical mass*[1] and *chain reactions*[2]. Mastery of the game is only possible after grasping the recursive nature of chain reactions and being able to intuitively identify piece placements that both capture the largest territory possible while leaving one in a strong, defensible position immune to retaliation. Initial gameplay and the sole victory condition are deceptively simple: place pieces one at a time and be the last player remaining on the board.

The game draws several comparisons to the ancient game of *Go*. Gameplay is trivially simple at first glance yet rapidly evolves into a personal preference for aggressiveness, evaluation of the correctness of moves, and emergent patterns similar to *Conway's Game of Life* or the practice problems of *Go*. I believe, though this will require additional research, that *Hanoi* represents an **NP–hard** problem. As the permutations increase over time—pieces are never removed from the board vs. chess, where the possibilities dwindle down to a stalemate—it will be extremely difficult to develop a competent AI to challenge adept human players.

Chance plays a role only for reckless or unstudied players as the result of individual placements is purely deterministic.

Project Scope

The development of generalized game logic capable of easy rule customization and the integration of this core game logic on a minimum of two platforms: server–side Python for ranked matches and match recording, and JavaScript for interactivity, training, offline play, and AI. Also, integration as a Facebook application with the possibility for future expansion to device-native applications such as iOS.

Unranked online multiplayer use does not require application server intervention, greatly reducing complexity and infrastructure requirements during initial development.

---

[1] The number of placed units needed for a given board position to overflow into the surrounding positions.

[2] The situation where a board position overflows into another causing the new position to reach critical mass and so forth.

SWOT Analysis

SWOT is an acronym identifying four key areas of examination: Strengths, Weaknesses, Opportunities, and Threats.

Strengths

- Deceptively simple mechanics.
- Addictive gameplay.
- Short matches on normal modes.
- High replay value esp. with added variants.
- Rapid development cycle.
- Easily scaled with minimal server requirements.

Weaknesses

- Difficult to monetize as a Facebook app. (Easier as a native app.)
- Without animation, chain reactions are potentially confusing, seemingly random, and possibly frustrating to new players.
- Increased map size grows match length cubically.
- Difficult to conceive of in–game purchases, rewards, achievements, and power–ups.
- Standard recursive–exploration solvers unsuited for AI play.
- No Angry Birds–style cute plushies.

Opportunities

- Unique concept with broad appeal.
- Community development through organized competitive play, tournaments.

Threats

- Easily duplicated concept.
- No legal protection of game mechanics.

Details

Product Perspective

Hanoi is, as far as can be determined, a new, self-contained game concept not currently available in any of the popular application or game markets in North America.

Broad and inexpensive distribution is available via Facebook and native App Stores. Additionally, physical play is possible, if tedious on larger board sizes.

Features

Described herein are multiple primary game variants, board layouts, scoring systems, and timing mechanics. Being turn–based, Hanoi allows for the integration of ranked and unranked online multiplayer, disjoint play, offline "hot seat" multiplayer, and solo play. AI players can be utilized on unranked matches (both online and offline) to fill positions not occupied by human actors.

Online play will require central server infrastructure to facilitate matchmaking and message passing—moves and text–based interactive chat—between physically diverse players. Peer–to–peer communication over a variety of channels is possible when developing native clients.

User Classes and Characteristics

This game should have broad appeal and could garner a passionate user–base through competitive play. The barrier to entry is low, and mastery an entertaining and addictive challenge. *Hanoi* encourages logical and analytical thinking making it a surprisingly useful tool for early education and experiments with machine problem solving alike. With board shape, size and rule variants the length of gameplay ranges from a few minutes to, potentially, several hours, allowing new users to begin play, rapidly experiment with different strategies, and gain familiarity.

The board game market in the USA alone exceeded $800 million in 2006—the latest data I was able to easily find—and over the last 8 years or so multiple indie game developers have successfully penetrated this market.

Operating Environment

The game engine of *Hanoi* would require implementation in both JavaScript and Python, as mentioned, with the possibility of expansion to Java (Android) and Objective–C (iOS). Underlying the online version described in this document would be: a message passing service,

an application service to handle server–side recording of matches and anti–cheat mechanisms, and a database system to store the recorded matches and rankings.

Turn–based games are well served by the fast, atomic operations of the MongoDB database server. Message passing would be accommodated by the Nginx Push Module. The WebCore Python Web Application Framework is extremely suited towards fast, light–weight operation and the design of pure web APIs, and the current version of Python 2.7 will be utilized during development and in production.

All three of these primary components—messaging, application, and database—can be independently scaled as necessary. Additionally, this infrastructure can be shared amongst all ports of the game engine allowing cross–platform online play.

Estimated requirements are described in detail later in this document.

Design and Implementation Constraints

Ranked online play requires server moderation to prevent cheating and allow records to be kept for the purpose of replay, scoring, and abuse response. Unranked play (online or offline) can be safely policed by the client–side game engine itself.

User Documentation

The interface for the game includes several self–documenting features which should introduce new players quickly. In–game prosaic documentation, solo matches vs. AI players of increasing difficulty, pop–up quick introductions with animated graphics, ticker tips, and integrated puzzles of a variety of types will be available.

Specific Requirements

User Interfaces

1. **Primary Navigation**
   Navigation between areas of the game: offline play, online play, puzzles, and help.

2. **Game Creation**
   A user interface to select amongst game settings and variants. The majority of this user interface is shared between online and offline play.

3. **Game Discovery / Matchmaking**
   Online play requires some method whereby users can discover other players.

   3.1. **Channels**
   Both public and private matches are organized into channels. Example channels include "Beginners", "Masters", and "Ranked Matches". Channels include both match discovery and text–based interactive chat.

   3.2. **Friend Invitation / Management**
   Private matches can be requested between users already known to each–other. This includes friends paired on Facebook as well as users "bookmarked" during open play.

4. **Puzzle Selection**
   An interface to organize and discover pre–set game puzzles or challenges. Needed is a way to identify which puzzles have been solved, how efficiently they have been solved (e.g. optimum vs. sub–optimal solutions), etc.

5. **Game Board**
   The game board interface requires the availability of one or more skins, animation behaviours, and interactive match–private chat for online play.

6. **Ancillary Interfaces**
   Various pop–up dialogs for win/loss, abuse reporting, etc.

Hardware and Software Interfaces

Basic gameplay utilizes the client browser. Browsers are rapidly moving targets and as such only the current revision of each of the popular browsers can be officially supported unless there is high demand for specific prior versions. The calculations utilized by the game engine are relatively trivial, requiring little computational or graphical power.

Server–side things are different. Each additional player in a match multiplies the physical bandwidth and storage requirements for a match. Given the following set of messages:

```
{p:19292868552,a:1,x:1,y:1}
{p:19292868553,a:1,x:7,y:7}
{p:19292868552,a:2,m:“Off to a great start, I see.”}
```

Also given a standard 2–player, unranked, 7×7 grid game with normal rules averages 80 rounds like the above, we can estimate the bandwidth and data storage requirements for one million active games: (ignoring text chat for the moment)

Two unit placements:	70 bytes (59 bytes + 11 bytes overhead)
Complete game:	5,600 bytes (70 bytes × 80 rounds)
Incoming bandwidth:	6,720 bytes (5,600 bytes + 1,120 bytes overhead)
Outgoing bandwidth:	13,440 bytes (6,720 bytes × 2 players)
Memory requirements:	840 bytes (70 bytes × 10 rounds + 140 bytes overhead)
1M games bandwidth in:	6.25 GiB
1M games bandwidth out:	12.52 GiB
1M games bandwidth total:	18.78 GiB
1M games memory:	801 MiB
Estimated infrastructure cost:	$90/mo. (1×2 GiB RAM server, 15 GiB outbound bandwidth)

These numbers are optimistic, however, they also clearly demonstrate that even in substantially worse cases (e.g. *quadrupling* the memory requirements) a *single* messaging server will be able to accommodate at *least* one million simultaneous unranked games.

Communications Interfaces

The messaging protocol, as illustrated above, utilizes JSON–encoded data with overhead minimized by utilizing single–character keys. Messages are “pushed” via long–poll AJAX GET requests with messages sent using simple HTTP POST requests to the messaging server. Ranked matches HTTP POST to the application server instead of the messaging server, with the application server passing these messages on to the messaging server for final delivery to the players.

System Features

Primary Game Engine

The primary game engine manages whose turn it is, where they are allowed to place, how to organize the underlying data structures for board arrangement, identifying victory conditions, optional timing restrictions, AI actors, etc.

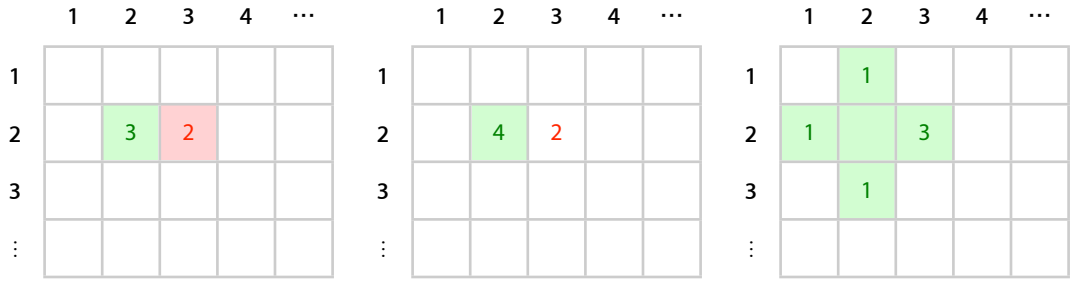
1. **Victory Conditions**

    1.1. **Standard**
    Eliminate all opposing players by capturing their territory.

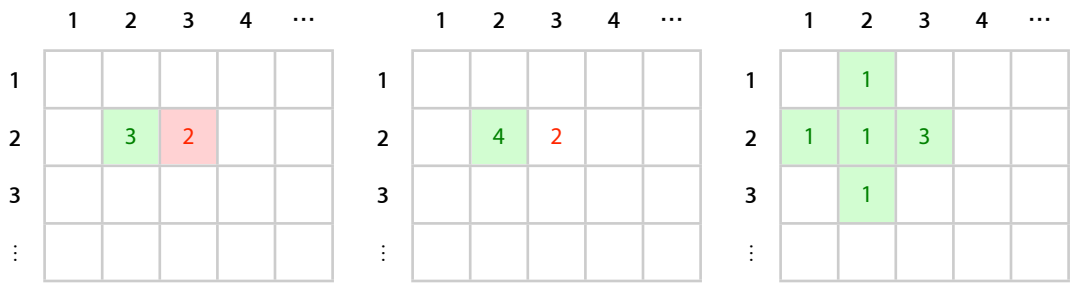
    1.2. **Suicide**
    Ensure your own territory gets captured by forcing chain reactions.

2. **Placement Strategies**

    2.1. **Standard**
    Place one unit per turn. If a territory reaches critical mass spread units placed there amongst connected territories, capturing enemy territory.



    2.2. **Flashback**
    Place one unit per turn. If a territory reaches critical mass, spread units placed there amongst connected territories and place an additional unit on that territory. *Greatly increases the likelihood of successful retaliation.*



3. **Scoring Methods**

3.1. **Standard**
= 10 × controlled territories + captured units - placed units

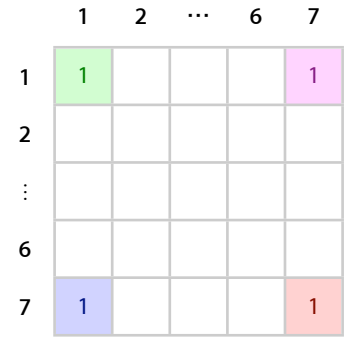
3.2. **Accelerated**
= 10 × controlled territories + active units + captured units

3.3. **Simple**
= active units

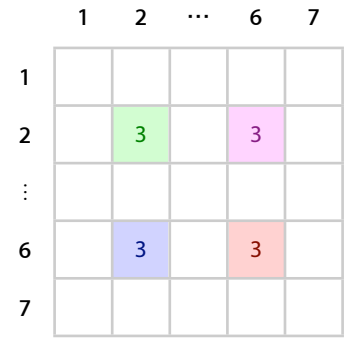
3.4. **Land Grab**
= controlled territory

4. **Starting Arrangements**

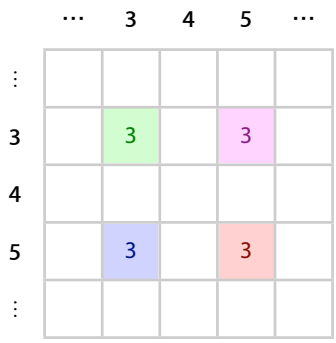
4.1. **Standard**
One unit per player arranged as evenly as possible on the field in corners.

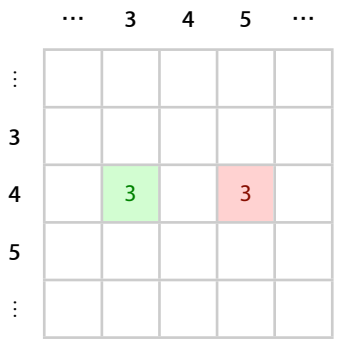


4.2. **Accelerated**
One unit less than critical mass one in from the corner.



4.3. **Face Off**
One unit less than critical mass placed side-by-side in the centre of the field, separated by a single empty territory. (Examples are for four player and two player layouts.)





5. **Field Layouts**

    5.1. **Square Grid**
    A square grid of odd dimensions. Standard sizes include 5×5, 7×7, 13×13, and 19×19. Custom sizes are possible.

    5.2. **Triangular Grid**
    Corners chopped off (they only have one adjacent territory), allows for even play of three players.

    5.3. **Hexagonal Grid**
    Both 4–player arrangement, similar to a square grid, and Chinese Chess–style for many simultaneous players.

    5.4. **Mapped**
    Risk®–style. Potentially allows both pre–set maps as well as custom player–designed maps.

6. **Timing Systems**

    6.1. **Sudden Death**
    When the player's clock runs down, their pieces are removed from the board and they lose. The starting amount of time is configurable.

    6.2. **Score Deduction / Penalty**
    Count time upwards and subtract a point value based on the amount of time elapsed, either after a threshold value is exceeded or continuously.

    6.3. **Hourglass**
    As one player's timer runs down, the other runs up, loss when out of time as per Sudden Death.

6.4. **Overtime**

6.4.1. **Japanese byo-yomi**
N periods of overtime, each M seconds long. If an overtime period is not fully depleted before the turn is over, replenish it.

6.4.2. **Canadian Overtime**
Unlimited periods of overtime, each M seconds long. A player immediately loses if they do not perform P placements within that time period. If the player does, the overtime clock is reset. There exists a variant where P increases overtime round after overtime round.

7. **Handicap**
Players can utilize handicap to even out ranked play and normalize skill. This also adjusts scoring, with weak players who forgo handicap gaining more points against strong players.

8. **Player Statistics**

9. **Computer Player Strategies**

9.1. **Dumb**
Place units randomly on controlled territory. A slightly less dumb variant would prefer territories closest to reaching critical mass.

9.2. **Defensive**
Chain react, if possible, as far away from other players as possible. E.g. corner, edge, or centre of mass.

9.3. **Offensive**
Chain react, if possible, as close to a vulnerable enemy territory as possible while avoiding retaliation conditions.

9.4. **Insane**
Ensure minimal unit distribution to prevent enemy chain reactions through controlled territory. Chain react at the maximal gain territory, or build up edges to efficiently spread units towards the front line.

Puzzles

Puzzle problems provide entertainment by themselves, but are also a useful tool for learning both basic and advanced game mechanics.

1. Beginner Problems
Problems with obvious solutions and simple goals.

2. Intermediate Problems
   Indirect attacks, planning ahead, anticipating opponent moves.

3. Advanced Problems
   Go–style problems, e.g. "life or death".

Multiplayer Matchmaking

1. **Match Creation**

   1.1. **Configuration**

   1.2. **Match Registration**

2. **Match Discovery / Search**

   2.1. **Filters**

   2.2. **Peer–to–Peer Matches**

3. **Channel Organization**

Communication

1. **Move Messaging**

2. **Textual Chat**

   2.1. **Lobby / Channel Chat**

   2.2. **Peer–to–Peer Match Chat**

   2.3. **Observer Group Chat**

Record Keeping

1. **Current Game State**
   Facilitates cheat prevention and disjoint play.

2. **Move Recording**
   Including timing information; allows for VCR–style replay and match verification.

3. **Chat Recording**
   Abuse protection, also for VCR replay.

Appendix A: Glossary

territory — a position on the game board, e.g. square or "city"

combat — the act of capturing territory and pieces; see also critical mass, chain reaction, and retaliation

army, armies — a conventional way to visualize the units/pips/coins/etc. utilized to mark occupied positions and occupation strength

board — the field of play where users place units and coordinate their armies; conventional play utilizes a square grid

critical mass — the number of placed units needed for a given board position to overflow into the surrounding positions

chain reaction — the situation where a board position overflows into another causing the new position to reach critical mass and so forth

retaliation — the situation where after a chain reaction the attacked player reclaims all or more territory than was previously conquered

strong position — a territory neighbouring an enemy territory where an attack by your enemy will not chain react, surrounding an enemy territory on two or more sides, or having initiative in a placement race

weak position — the opposite of a strong position

placement race — when two players, turn after turn, build up units next to each–other with the intention of causing a chain reaction

initiative — in a chain of move/counter–move, a player is considered to have initiative if they are 'leading' the play or will win a placement race