CS3110 Hnefatafl Design Doc

Elizabeth VanDenburgh (eav38) Joseph Dwyer (jmd456) Taeer Bar-Yam (tb442)

November, 12 2015

1 Hnefatafl Game Dynamics

The game is played on an 11×11 checkered board with two uneven sides. 12 white pieces are positioned around a white "king" at the center of the board, with 24 black pieces arranged on the edges surrounding them. See 1. There are three main mechanics involved in the game:

1. **Movement**
   Pieces move like rooks in chess (horizontally or vertically). Only the king has a limited motion of three squares at a time.
2. **Capturing**
   When a piece is flanked by two pieces of the opposing color, that piece is "captured" and is removed from the board. This flanking must be done actively by the offense (i.e. the center piece can move between the others without being captured).
   The king cannot participate in flanking an opponent. In the case of the king, it must be flanked on all sides to be considered "captured."
3. **Winning**
   Black wins when the king is captured, or the king's only escape is to move back to the center square of the board.
   White wins when the king is moved to one of the side squares of the board.

See: `https://en.wikipedia.org/wiki/Tafl_games#Hnefatafl`

2 Interface:

Ultimately we would like to implement the interface as a GUI using openGL, and allow click-and-drag motion commands. Initially, we will implement a text-based interface that uses chess-like notation for movement and output a ASCII grid to show positions of pieces. This interface will likely remain available as an option
This will be playable against another human player, or against an AI.

3 Stretch Goals:

- Hnefatafl is an old game, and there are many versions of it that exist. Time permitting, we would like to implement many of these variants, working off of a general game core.
- Detect if the game has become unwinnable.

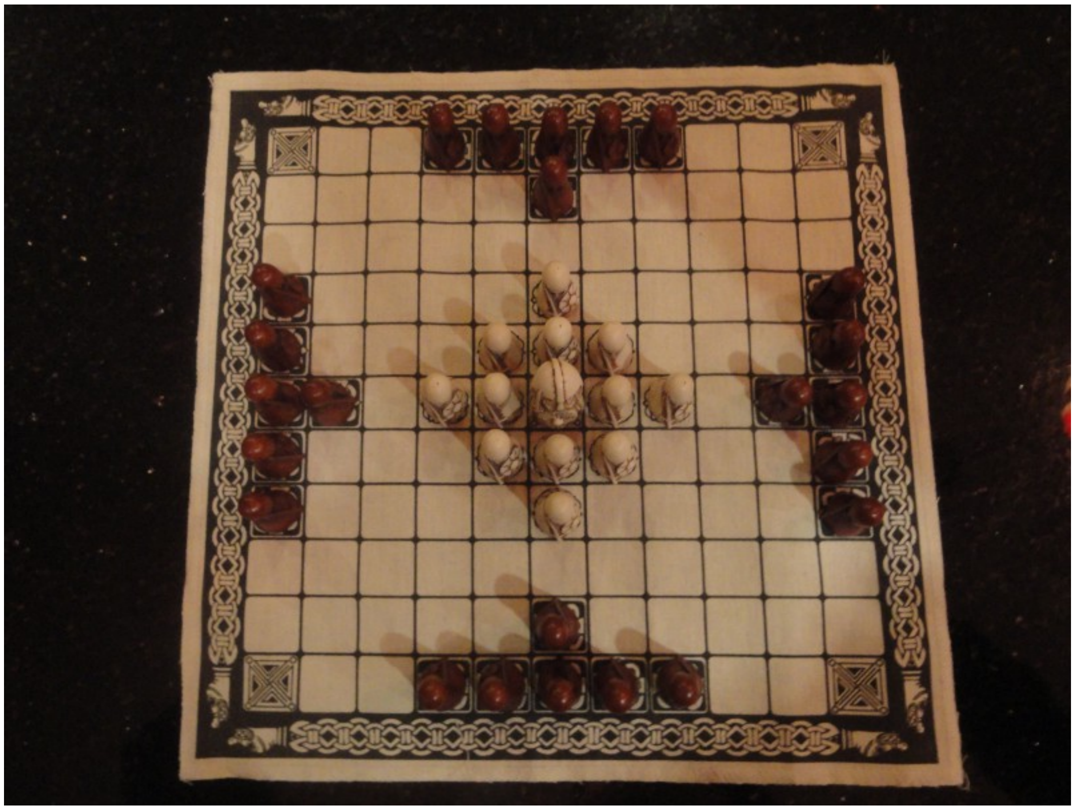
Figure 1: Hnefatafl starting board position. Source: https://medium.com/war-is-boring/you-have-to-play-this-1-600-year-old-viking-war-game-cef088ae4e2d#.10kof827g

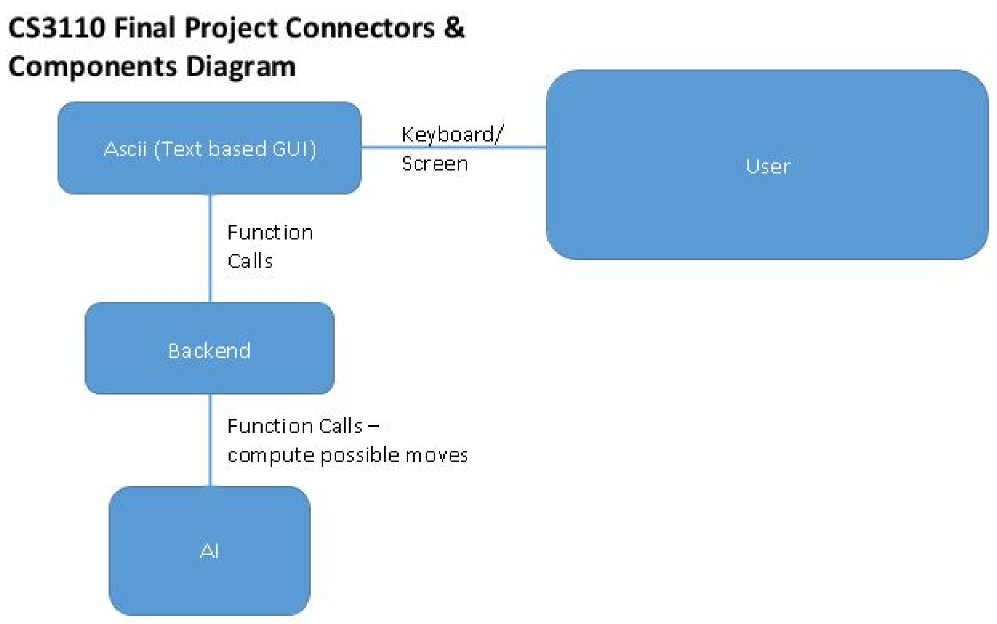
CS3110 Final Project Connectors & Components Diagram
Ascii (Text based GUI)
Keyboard/ Screen
User
Function Calls
Backend
Function Calls – compute possible moves
AI

Figure 2: Connector Component Diagram of the pipe and filter architecture for our project.

4 Architecture

Thinking pipe and filter information goes from user - gui - backend - gui - user

5 System Design

5.1 GUI

5.2 Backend

Depends on GUI. Depends on Menu. Depends on Helpers.

5.3 AI

5.4 Helpers

Makes up for functions that should have been included in OCaml.

5.5 GameType

all types of things, maybe helper functions most things will depend on it.

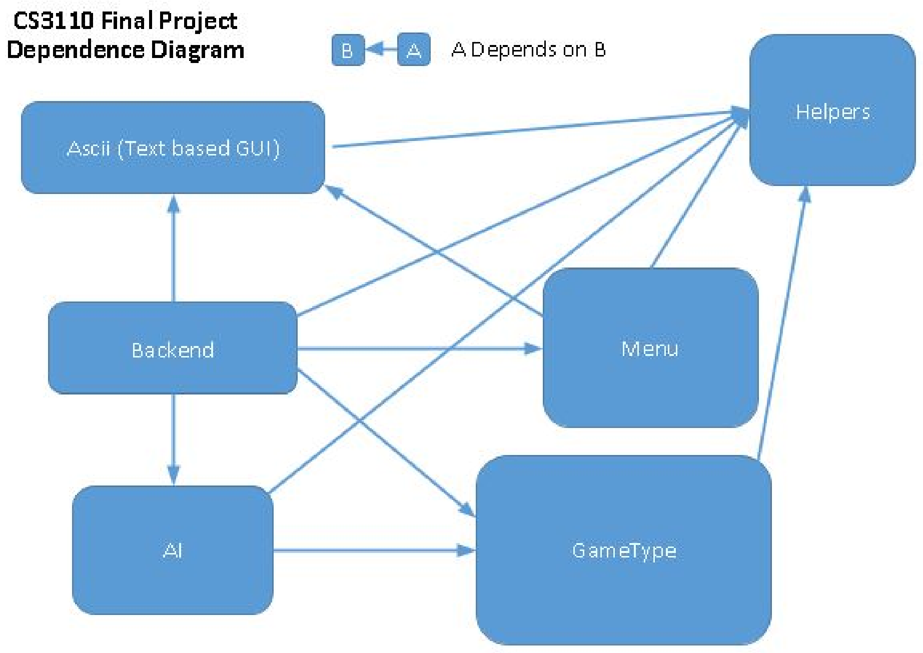

Figure 3: Diagram of dependencies between modules in our project.

6 Module Design

7 Data

- Player turn
- Piece locations to maintain the current board
- Game configurations

Structures include:

- Piece types
- Game configurations
- Board with dimensions and piece locations

8 External Libraries

- Termbox for creating an ascii board
- OpenGL for creating a more graphical gui

9 Testing Plan

- Write unit tests for sate of the board after certain moves
- Watch the AI play against itself
- Get play testers