CS 1530 – SPRINT 3 DELIVERABLE

29 FEB 2017

https://github.com/danieljustice/Hnefatafl

Team Scrumbags

Timothy Kang (TimKang95)

George Mattesi (Mattesigp)

Daniel Justice (YuzuFugu) (Scrum Master)

Zachary Gannon (Gannon93)

Accomplishments

For the third sprint (Sprint 3), the team accomplished all features we sought to complete by the provided delivery date, as well as progress on one feature that was not required by the customer but that we anticipate will be need shortly. The team used Trello to document progress, GitHub to share code, and Slack to instant-message.

During the retrospective for the second sprint, the team defined the following:

- What went well:
  - Scrum meeting went quickly and concisely
  - Planned enough stories to fulfil requirements
  - Division of labor was fair
  - Gradle is running smoothly for everyone now
  - Communicating on Slack went much better
  - Branches were utilized effectively

- What didn't go well:
  - Coordinating everyone's vastly different schedules
  - There was a large push last minute to finish sprint

- What can we improve:
  - More stand ups per week to help spread work out over sprint
  - Better JavaDoc comments
  - Code is peer-reviewed before being merged into master

For the third sprint, the labor was divided as follows:

- Timothy Kang: Complete set of working win conditions, attacking pieces, junit tests
- George Mattesi: Complete legal movements, junit testing
- Daniel Justice: Scrum Master, Asset creation, integration, and unit testing
- Zachary Gannon: Iterable play, Save and Load Function, documentation/write-up, defect testing
- All: Peer review

The division of labor, like the first sprint, was divided judiciously with each member's schedule and capacity in mind during the planning meeting and brought little conflict or disagreement.

User Stories

The team completed all the stories that were scored for the third sprint plus some. The stories completed were related to the core functionality, stability, and maintainability of the game. The total velocity for the second sprint was 36 points. The following are the completed user stories, in order of priority:

- **Functionality: Make a King piece (2 story points)**
  - As a user
  - I want to be able differentiate the king from the other pieces
  - So that I can know either who to keep safe or who to surround

- **Functionality: Win Condition: King reaches a corner (2 story points)**
  - As a user
  - I want to be able to move a king to a desired destination
  - So that I can win the game

- **Functionality: Win Condition: King is directly surrounded (2 story points)**
  - As a user
  - I want to be able to surround the king to where the king cannot move
  - So that I win the game

- **Functionality: Display winner (2 story points)**
  - As a user
  - I want to be able to see who wins the game
  - So that I can be able to follow the rules of the game

- **Functionality: 4 Corners and Throne are visually distinct (4 story points)**
  - As a user
  - I want to be able see the distinct squares on the map
  - So that I can be able to see where I must defend or must move to win

- **Functionality: Win Condition: No more pieces on a team (4 story points)**
  - As a user
  - I want to be able defeat all the pieces
  - So that I can win the game since the other person or I cannot move.

- **Functionality: Legal Moves (8 story points)**
  - As a user
  - I want to be restricted to legal moves that follow the rules
  - So that I can follow the rules and not cheat

- **Functionality: Load and Save Games (8 story points)**
  - As a user
  - I want to be able to load and save games
  - So that I can be able to load my previous games

Velocity for sprint 1: 46
Velocity for sprint 2: 28
Velocity for sprint 2: 36
Average velocity: 33.33

User Stories cont.

The following stories were discussed while planning for the second sprint, but were not critical enough to score and delegate resources to the second sprint, as the stories were not crucial to the customer's requirements for the second sprint.

Backlog for future sprints:

- **Functionality: Only King in Special Squares**
  - As a user
  - I want only the king to be able to enter the special squares
  - So that no player can cheat and gain an advantage by blocking the King's spots
- **Functionality: King cannot return to center**
  - As a user
  - I the king to not be able to return to the center once he has left
  - So that no player can cheat and gain and advantage
- **Functionality: King moves only when it is the Shields' turn**
  - As a user
  - I want the king to move on the correct turn
  - So that no player can cheat and gain and advantage

Decisions

The team planned the third sprint's user stories based mostly upon the code requirements supplied by the customer and the previous sprints' retrospective results.

User stories that were directly requested by the customer for this sprint were given the highest priority. Upon splitting up the user stories in our team we felt confident that we could take on more and set one of our members on a user story we expect to be important in future sprints. This proved to pay off as we successfully finished all of our user stories.

The stories with the lower priority also had lesser story points as they did not require much development time, such as in-line documentation and ensuring code quality. The write up was given low priority but high story points in that it was planned to be one of the last stories to be completed and was expected to take a larger effort than stories of similar priority.

Previous Defects

The following defects were found while manual testing. These defects were actually fixed and merged into master. This had to be done to provide the customer the correct functionality negotiated during reading of the deliverable as well as sprint planning.

1. Player can overwrite own pieces

   Fixed: When attempting to place a piece on top of an already existing piece, the attempt is ignored and the player must try another place/try again

2. Duplicating Pieces

   Description: When placing a piece onto the top 2 rows of the board, the piece would duplicate the piece from the pieces original spot to the desired location

   Fix: This was fixed from an error in the attackPieces() function where it had a mismatched x y location. Picking the desired location was fixed by fixing a 3D array instance

New Defects

1. King can overwrite other game pieces

   Reproduction Steps: Start a new game, click the King game piece, and click the location of an existing piece to “move” it.

   Expected Behavior: No move occurs at all.

   Observed Behavior: The King piece overwrites the other existing piece.

2. King can move at any time

   Reproduction Steps: Start a new game, click the King game piece, and click another location that is in a straight line

   Expected Behavior: King only moves if the clicks are valid moves

   Observed Behavior: The King piece will move, ignoring whose turn it is and any obstacles that may be in the way