Benjamin Carson (bkc52), Nicholas Rutledge (njr48), Timothy Eng (te76)

CS 3110

M. Clarkson

MS0: Proposal

Regular status meeting:

Outside of our twice a week discussion, which is where we will bring up any issues that we are having with our TAs, we plan to meet Tuesday around 3:30, and have not yet figured out another day – probably Friday afternoons, but will see as the project progresses. In addition, we have a couple other communication/project management systems set up – besides the obligatory Cornell GitHub repo, we also have a group chat for quick issues that don't need us to be in person to discuss, and an Azure DevOps board, which is an Agile/Scrum project management tool that (te76) has used at work and found helpful for user story tracking, epic/issue/task management, division of labor, sprint tracking, etc.

System proposal:

A proposal for your system. Summarize the system you intend to build. Tell us what will be the most important functionality of your system. This summary should be about one page long.

Provide:

A very short statement of the core vision or key idea for your system.

We plan to build a version of the card game Solitaire. We may allow for the player to play different versions of Solitaire, but will likely focus on the variants of the Klondike form of the game, and specifically the Agnes form (the one the most people are used to playing).

Reference:

https://en.wikipedia.org/wiki/Klondike_(solitaire)

https://en.wikipedia.org/wiki/Agnes_(card_game)

A short (no more than six items) bulleted list of the key features of the system.

- Multiple versions of Solitaire, possibly
- Movable cards
- Win condition
- Check for illegal moves
- Modules with different types (e.g. card with numbers and suits and colors, deck, stack of cards, foundation deck, etc.)

A narrative description of the system you intend to build. Go into enough detail that, if your charter were given to another team, and that team were never allowed to talk to you, they would still understand more or less the functionality that you have in mind.

We will have a representation of cards, containing suit, rank, and if it face-up. We will then implement a deck by creating a stack of cards. The deck will also have a way to shuffle, randomizing the layout of the cards. We will then fill several card stacks on the table, some face down, and have the remainder of the deck be available for drawing. Cards can be moved between stacks depending on the game rules (one rank lower and alternating color for base). There will be four stacks at the top, originally empty, where the player must stack the cards in ascending order by suit. If we implement hints, these will highlight the first card found that can move and the stacks to which it can move, or show no available moves.

A roadmap for your system. There are going to be three “sprints” (implementation phases) after this charter: MS1 (alpha), MS2 (beta), MS3 (release) Each will last about two weeks. Make a list of what features you plan to implement in each sprint. Then, for the features you plan to implement in MS1, sort them into three categories— Satisfactory, Good, and Excellent—like how the programming assignments in this class have been. That gives you a plan for what to do if MS1 is going worse or better than you expected.

Alpha

- Satisfactory: have cards and a deck
- Good: deck contains the correct number and types of cards: 13 of each suit only
- Excellent: moving cards around between stacks

Beta

- Satisfactory: rules for moving cards
- Good: one card vs three card draw
- Excellent: win and lose condition

Release

- Different game variations, restart new game, scoring(#moves)
- If possible: hints, graphics

A preliminary design sketch for your system. Spend some time thinking through what you are going to need to build. Of course, your plans will evolve throughout the project. But it’s good to have talked as a team up front about what seems necessary. In about one page, answer the following questions:

What are the important modules that will be implemented? What is the purpose of each module?

We will implement modules for a card, which will have its suit, rank, and visibility; a deck module to represent all the cards in use; a basic game module, encoding all things similar between different solitaire variations; and modules for each variation included, heavily using the basic game module but adding/changing a few details where needed. We will also have a module for the table, which may change depend on the game variation.

What data will your system maintain? What formats will be used for storage or communication?

Our system will maintain the cards in each stack as well as the deck. It will also keep track of the number of moves in order to calculate the score. We will communicate the current game state to the user using text-based representations of the cards in each stack. If we have lots of extra time, we may use a GUI.

What data structures do you expect to use as part of your implementation?

We will need:

- Multiple stacks to represent the stacks of cards
- Records
- Variants
- Lists
- Queue
- Arrays

What third-party libraries (if any) will you use?

We will most likely not need to use anything as almost everything of what we need is built in.

How will you test your system throughout development? What kinds of unit tests will you write?

We will use some unit tests to check if cards are created properly, decks created, randomly sorted, one of each card, etc. Also check for illegal moves, win condition/impossible to win setup. Large amount of this will involve playtesting like with A3, as it is a game where we can check for bugs more easily by playing. May also try to find a way to use Bisect if possible/if time.

How will you, as a team, commit to following your testing plan and holding each other accountable for writing correct code?

We will use DevOps board to assign tasks and keep track of timeline, bugs, etc. We will also check in with each other in meetings/in discussion and discuss issues with TAs. We will assign tasks to each other as well.