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

CS 3110

M. Clarkson

MS0: Proposal

- Cycle of towns, wild encounters, gyms
- Clarkson is the final gym leader has OCamlrupt or something
- Only include certain types
- Level up after winning each battle?
- Evolution excellent scope
- Routes each have a certain number of wild encounters
- Gyms/Routes are Python = Grass = 1110, Java = Ground = 2110, OCaml = Fire = 3110;
- JSON to store moves

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 will implement a small version of a Pokemon game that will be completely text based. Features will include routes, trainers, wilds, gyms, battles, etc. The player will be able to play the game similar to the originals (not Showdown style).

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

- Moving from route to route and into gyms, etc.
- Catching pokemon
- Battling trainers
- Buying items
- Winning (beating the Champion aka Clarkson)

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.

This will be a simplified pokemon game (not showdown or lets go) in which there is a starting town where you pick your starter, and then a route to the next town. This route will give a certain

number of wild pokemon that you can fight and catch. There will also be trainer battles which behave very similarly to that of the wild encounters, (except cannot catch the pokemon, and allows for several pokemon on computer's side). In every other town, the player will have the option to go back to the previous town, go to a PokeMart/PokeCenter, Gym, and next route (only accessible after beating the town gym leader). On each route, after a battle, you have the option to retreat back to town, use an item, or keep going. If all held pokemon have feinted, immediate retreat to PokeCenter, and if on a trainer, lose money.

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: Move around from location to location, basically A3 stuff but for our game.
- Good: ~10 pokemon, 6 types (Normal, Water, Grass, Fire, Electric, Ground) (e.g. JSON files)
- Excellent: working battle state (1 pokemon vs 1 pokemon)

Beta

- Satisfactory: Change states between overworld and battle, random encounters part 1 (pokemon shows up)

- Good: leveling up after battle & exp. and random encounters part 2 (capture, run away)
- Excellent: updating pokemon stats and swap party during battle

Release

- Satisfactory: Change move set/Evolution
- Good: The rest of the types
- Excellent: 2800 gym (Psychic/Ghost/Dark) & other TBD enhancements from the original games (maybe audio playing)

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?

- modules for player, pokemon, trainers, and gyms

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

Data our system will maintain:

- Player's pokemon, items, money(?)
- Pokemon stats - HP, XP, ATK, DEF, SPD
- Trainer/gym leader state (if defeated, what pokemon they have)
- Hash table for moves

Format for storage and communication

- Store main game data in JSON
- Use console for showing info
- Commands include GO, BUY, ITEMS, ROSTER, TGM (toggle God Mode for devs testing)

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

Data structures we will need:

- Lookup table for type effectiveness
- Variants for types
- Json files
- Arrays, lists

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

We will use JSON files to store move sets for different pokemon and will use the Yojson library to do this, similar to A2/A3, and audio libraries if we get there. Other than that everything else will be using built in features.

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 pokemon created properly, characters, moves, etc. Also check for illegal moves, win condition etc. 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.