Self-Assessment – Final Week

# All the C++, all at the same time.

You will be designing an adventure game.

Choose themes and features that you think will be fun to play. You do not have to implement every feature mentioned here. A clean, extensible design is more important than a rich feature set.

An adventure game typically has at least one *player character*, who moves around on a *map* and encounters magic, monsters, friends, treasure, raw materials, tools, etc.

A map might be the interior of a building or tunnel system, or it might be an entire landscape, an archipelago, or a galaxy.

A map typically contains various types of terrain that affect movement. Some characters or monsters might be able to fly, or swim, etc.

A map is typically populated with monsters and/or non-player characters. Some of these stay in or near fixed abodes, and others wander freely. Some are friendly, some are not. Some are talkative, some are taciturn. Some are edible, some can be ridden, some can carry your luggage, some will steal your luggage, some will eat you.

A character often has various fixed attributes that affect gameplay. He might belong to a particular *species* (which might give him quasi-magical abilities like the ability to see in the dark), and he might have a particular *vocation*, e.g. a wizard is good at using magic but bad at using swords. He might be a particularly strong individual, or particularly charming to strangers.

In addition, a character has various attributes that can change over time. His health may decrease if he is struck in battle, or increase if he drinks a healing potion. He may gain proficiency in various tasks as a result of experience.

A character may obtain weapons, armor, magic items, food, etc. that can confer various abilities or penalties.

He may be able to buy or sell items (usually at some fixed location on the map). He may be able to build things (possibly including magic items) by gathering all the needed resources.

A game can be combat-centric (see something alive? kill it, steal its treasure), exploratory (see new things, make new friends), economic (make your town thrive through investment and trade) or story-driven (rescue the princess). If you have a system for keeping score, you can shape the player's behavior by giving points for good outcomes. You can assign *quests* by which players earn points. (e.g., "slay the green dragon that lives over the hills to the East", or "find the Amulet of Dyxcda". )

**DISCUSSION:** The map is going to be one of the challenges you have to design around. A good approach is to start with a small, manageable map. You might divide your map into small "levels" connected by staircases/tunnels/magic-portals.

Communicating the map to the user is another challenge. See *Dwarf Fortress* for an example of how it's possible (but not pretty!) to show a map as ASCII art. You might also consider a purely text-based game, with descriptions but no pictures (see *Colossal Cave Adventure*).

**BONUS FEATURE IDEAS:**

1. **challenge:** Randomize the map, either in small ways, or by having the entire map generated at runtime. Be careful, though: generating a good map is harder than it sounds. It's no fun to play on a map where you're completely surrounded by lava and can't get anyhere.
2. **hard research challenge:** Add a Save Game feature. Store the saved game(s) in a file(s). Add a Resume Saved Game feature. You'll need to learn about file i/o, and you may run into challenges in *serializing* and de-serializing your objects.
3. **Extra-hard research challenge:** If you have lots of monsters or non-player characters moving, and they each have to choose which way to move next or who to attack, try implementing each monster as its own thread.
4. **Super-hard research challenge:** Use a graphics library to make a GUI.

## **Activity 1 : Make a Plan**

You will be responsible for managing your own schedule. Create a document with your schedule on it, with specific goals that you can measure yourself against. (E.g., "research an easy-to-use a graphics library for C++ – choose one by Monday 3PM. Have a basic map display working by Monday 5:30 PM.")

Your schedule should account for:

1. Setting goals / directions / themes
2. design time / research / proof-of-concept prototypes
3. core features / design prototypes
4. adding features / content
5. extra debugging / testing
6. overhead: meetings, interruptions, job interviews, time spent tracking or revising the schedule
   1. daily stand-up meetings (10 minutes each)
   2. Design/code reviews
      1. Tuesday and Wednesday: 1 hr
   3. Final presentation: Friday afternoon: 1hr++
7. Murphy's Law.

## **Activity 2 : Execute Your Plan**

Design and implement your program.

In daily stand-up meetings, you will share how your progress compares to your schedule.

(If you realize your schedule is unrealistic, toss it out and make a new schedule at any time. Likewise, if your design proves unworkable, change your design.)

**FINAL ADVICE**: back up your work early and often!

## **Activity 3 : Show Your Results**

On Friday afternoon you get to share your code with your classmates. The game voted most fun to play will be awarded 1,000,000 Internet bragging points.

You will also turn in your code on GitHub classroom to be awarded thoughtful, helpful comments from your instructor.

**NOTES ON GRADING:** I'm looking for forethought, not features. Your implementation of a Map should make it easy to design, load, and use more maps. Your menagerie of monsters and characters should be organized into useful class hierarchies. If you have a Werewolf monster and a Vampire monster already defined, it should be easy to make a Vampire Werewolf with practically no coding needed – and to place this new monster at practically any spot on the map without changing any of the gameplay logic. User interactions (display and input) should be isolated from the rest of the code, so that if next week you learn how to use a fancy graphics framework, you can plug your existing gameplay logic right in with minimal changes.

Credit for creative themes (Forest of the Pikachu? Garage of the Haunted Volvos?), but spend your time wisely. Design not details.

Comments – even more than usual. Discuss your design decisions; comment on how you applied the Design Principles we learned.