CS354p Lab 8: Technical Design Document

This lab will explore the basics of building a Technical Design Document, and will give you an opportunity to practice your communication and writing skills. Technical Design Docs (TDDs) are created in conjunction with the Game Design Doc (GDD) and Art Bible to guide programmer development and inform development timelines while making a game. We will focus on creating a TDD for a simple game to keep the scope and documentation limited. You will submit a .pdf containing the following information the TDD requires.

Choosing a Board Game

Your first task is to choose a board or card game with a simple set of rules to bring into video game space. I would highly recommend erring on the side of "Candy Land" rather than "Risk" since you'll be describing a complete set of rules and win/conditions (among other tasks) as part of the TDD. Please avoid Eurogames (e.g. "Settlers of Catan"), because those games are way too complex for our purposes.

Game Overview

In this section, please describe at a high level the game you will be creating. This should describe the basic player experience and interactions. For example, you might say this is a networked adaptation of Hungry Hungry Hippos with a top-down view of the arena, touch-based controls for mobile, and MOBA-style powerups and abilities. This overview is intended to give the reader enough information to have a general idea of the game you're trying to make. You should also outline here an overview of the features, listing:

- Playable characters
- Enemies/NPCs
- Interactable items including weapons and powerups
- Overview about the board/arena
- Additional menus and screen functionality
- Any other necessary systems, such as a game round manager, an inventory manager, a network matchmaker, a save system etc

This list should be thorough but the descriptions concise — you'll be going into much more detail in the "Game Features" section for each of these.

Technology Considerations

In technology considerations, you will make several key decisions that will effect how you approach the remaining engineering design decisions. You will decide:

- Choice of engine
- Additional frameworks and plugins you will use rather than build
- Target platform(s) for deployment
- Team size
- Total development time

Obviously most of these choices are not things that we as programmers get to decide, but this will help you fix the scope of your project and therefore the scope of your TDD.

Game Features

This section should go into detail about each of the features listed in "Overview." In addition to all the items listed above, you should also include info on:

- Player controls/inputs
- Win/lose conditions
- Sound

This is your chance to think through all the features that will appear in the game, and it should give you some idea of the work each entails.

Game Logic and State

In this section, you wil explore all necessary logic and state required to make the features described in the previous section. This should include information about:

- Data input (i.e. where is the data coming from and how are you getting it into the game?)
- Data storage (i.e. what sorts of classes and structs are required and who is managing them?)
- Saving data (i.e. what data needs to be saved, if saving is supported at all)
- Rule and state management (i.e. how are rules defined and how do they connect to the associated managers?)

This section can be longer or shorter depending on how much information you put into the previous section, so please don't repeat yourself or try to pad it out if you already described this information.

Custom Classes/Archetypes

This section will include UML diagrams of how you are approaching class/archetype management for the above features/systems. You can certainly describe in words relationships between runtime objects/systems, but a picture truly is worth a thousand words. This section should ideally allow you (or another programmer) to start creating the underlying software architecture without having to think too much or raising too many questions about inter-related functionalities.

Remember: the goal of documentation isn't to be tedious busy work for the person creating it. You are giving future-you (and all your teammates) a precious, precious gift.

Graphics Requirements

This section will describe the graphical components of the game, including anything a developer would need to know about the game's GUI. What is described in here is directly related to the platform you are deploying your game to, and the overview information describing the game's genre. We are not concerned about art style (that's something to include in an Art Bible). In practice we want to know the rough expected polygon count and texture memory, minimum hardware specification, an overview of the shader pipeline, and other performance considerations, but these questions would usually be addressed by a graphics programmer, so just do your best and keep it high level.

Physics Requirements

This section will describe the physics requirements (if any) of the game. Like the graphics requirements, this section is focused on performance and the specific needs of the physics engine, but likely your game will have minimal (or no) physics, so I don't expect anything more than a couple sentences, unless you're making a Half-Life version of Snakes and Ladders or something.

AI Requirements

This section will describe the AI requirements (if any) of the game. Please give an overview of the types of AI you'll require (e.g. an opponent player), expected states/behaviors, and how you might choose to implement this AI (e.g. a Behavior Tree or something simpler).

Network Requirements

This section will describe the networking requirements (if any) of the game. Please give an overview of when and how networking will be used, if you have a more complicated feature such as a matchmaking system, and if your game will support achievements, in-game microtransactions, require a backend server etc. Include an overview of the player experience joining the game and leaving the

game (if not described previously under Features), and include at least high-level considerations about security/cheating concerns (if any).

Submission

Please submit your TDD individually as a .pdf to Canvas.