Disney Streaming Services - Connected Device Native Client Engineering

Interview Candidate Assignment

Exercise #1: Take-Home Coding Assignment

Welcome to Disney Streaming Services,

We'd like to have you code an implementation of a common use case in the Disney Streaming Services Connected Device Engineering group. Please use one of the following technologies to complete the exercise.

1. C *or* C++ Codebase
   - You may use any version of C *or* C++; please do specify the language generation used (e.g. C99 or C++14). Be prepared to talk to the differences/features in more modern releases if you are familiar with them.
   - You may use 3rd party libraries or frameworks. You will need to include these libraries in source or object code form so we can compile your exercise locally on our machines. 3rd party libraries should be used as a *compliment* to the code that you write, and to speed work like drawing to screen or rendering UI.
   - For UI frameworks, please use an OpenGL style library for rendering (Examples, including but not limited to: SDL, GLFW, or GLUT).
     1. Do NOT use Qt or a similar UI library to render the UI.
     2. Do NOT use an "off the shelf" game or rendering engine like Unreal, Unity, Cocos2D, etc.
     3. You MAY use your own personal project if you have one

     Our team is building a custom rendering engine, so showcasing your ability to contribute to that effort will be important.

2. Rust Code base
   - IF you have talked to the recruiter about a position on our team developing in Rust, you may use Rust instead of C/C++ for your take home assignment.
   - *You MUST confirm with your recruiter before proceeding down the Rust path.*
   - The same limitations on 3rd party libraries and frameworks apply as mentioned above in the C/C++ Codebase section.

Note: Due to limitations in screen sharing in Zoom, we ask that you render to a window instead of full screen (this is especially important for candidates who are invited for a final interview).

Although the assignment is fairly limited in scope, treat it as a feature in a larger, real-world application, where architecture, design patterns and optimized code are important. Quality is more important than quantity, so good architectural and coding decisions are more important than completing any of the extra credit tasks. Once the assignment is complete, we'll review your work to assess the technical decisions made, and the completeness and quality of the work.

Our Native Client engineering teams focus on writing code on devices ranging from game consoles to very low powered set-top boxes. Be prepared to show and talk to techniques for writing optimized code in very CPU and memory constrained environments. Think 3000 DMIPS processors with very slow GPU's and limited memory bandwidth.

Here's the assignment:

1. Review the requirements below and ask any questions you have.
2. Then, provide an estimate of time to completion. Please estimate both total hours, and the projected completion date
    - Our intent is not to take too much of your time, so try to scope a solution that can be completed with a reasonable amount of effort
    - In the interview, we can talk about anything you might do differently if given more time
3. Code the example. Feel free to ask any additional questions during this phase.
4. Provide the code for review, either on a code repository (GitHub) or by email, DropBox, etc.
5. Provide instructions for running the app and include:
    - Instructions for building and compiling the source on Mac OS X or Windows
    - A compiled binary executable runnable on Mac OS X or Windows

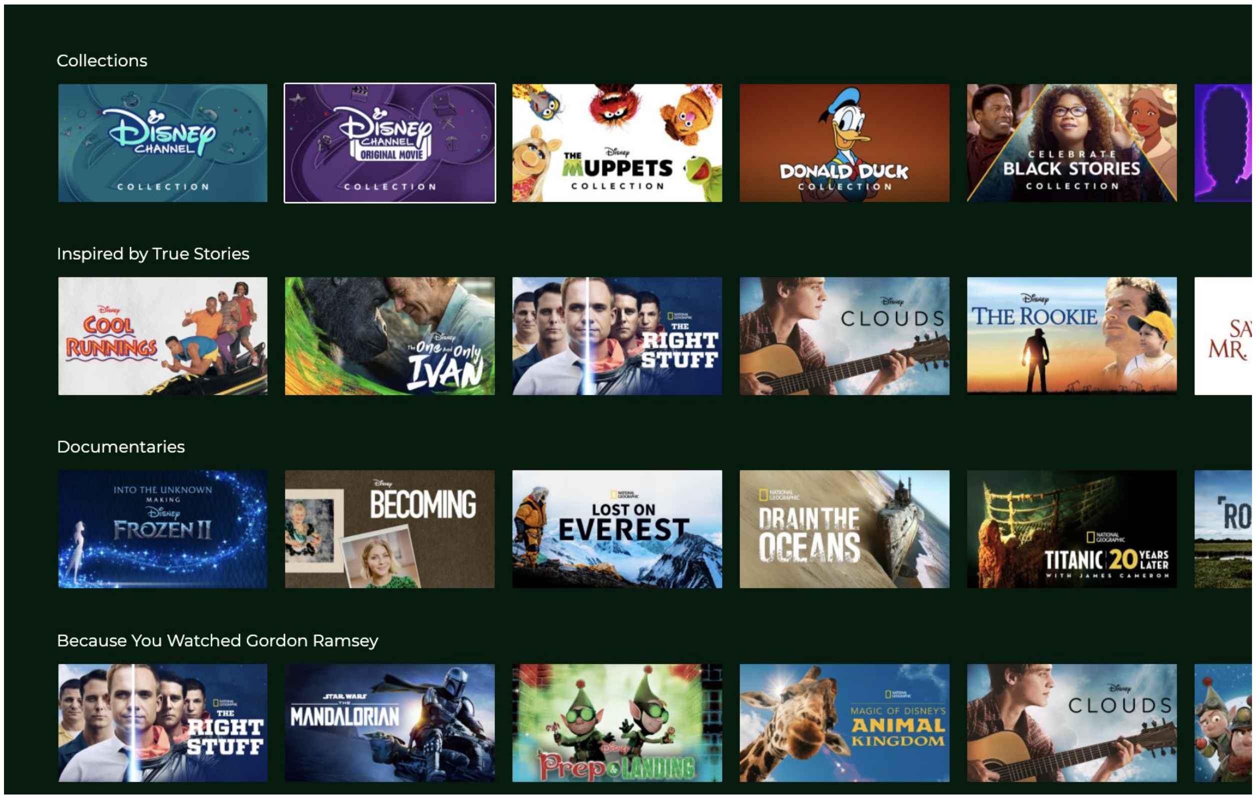
Data API

- https://cd-static.bamgrid.com/dp-117731241344/home.json will provide data to populate a "Home" page similar to the current Disney+ experience.
- https://cd-static.bamgrid.com/dp-117731241344/sets/<ref id>.json will provide data for dynamic "ref" sets. The "ref id" will be provided in the "home.json".

Requirements

- Create a screen that consumes the home page API and renders the pre-populated data onto the screen.
- The focused tile must be scaled up.
- The app should support navigation similar to a remote control, e.g. up/down/left/right/enter/back/etc. Avoid mouse input.
- Minimum layout should be multiple rows of data, but please feel free to add in your own design ideas as well!

Example Layout Idea



Extra Credit

1. Dynamically populate the "ref" sets as they come into view.
2. Allow interaction or selection of a tile. For example, show a modal with data on selection.
3. Incorporate transitions and/or visual aesthetics.
4. Add some Disney magic.