Web-Based Graph Algorithm Game

Level 4 Honours Project

George Yarr, March 2024

Introduction

Goals

- It should be engaging.
- The better user understands it, the faster they can complete it
- Smooth development process
- Developed with pedagogical ideas in mind

Background and Existing Products

- Can be understood in the context of objectivism vs constructivism

- Independent discovery is key

- Visualisations achieve this to an extent, I want to facilitate a deeper level of independent discovery.

- Concept of DIY

Bill Pelz’s Principles

Online Learning Consortium 2003 Award Winner

- Let the students do (most of) the work!
- Interactivity is at the heart and soul of effective asynchronous learning

- VISUALGO - National University Singapore
- Algmatch - Liam Lau & Callum Ormond
- Visualization of Classical Graph Theory Problems - Fatma Alsayegh

Requirements

MoSCoW

Must Have

- A visualised tutorial for BFS, DFS, DSPA
- Users can “do it themselves” to execute an algorithm - carry out actions such as visiting a vertex
- Users can execute the algorithm with help from the computer
- Graphs must be well-formed and externally consistent
- Pseudo-code with highlighting

MoSCoW

Should Have

- Background general information on the algorithms and concepts
- An intuitive user interface

MoSCoW

Could Have

- Additional algorithms - e.g. Hopcroft-Karp or Tarjan’s Strongly Connected Components

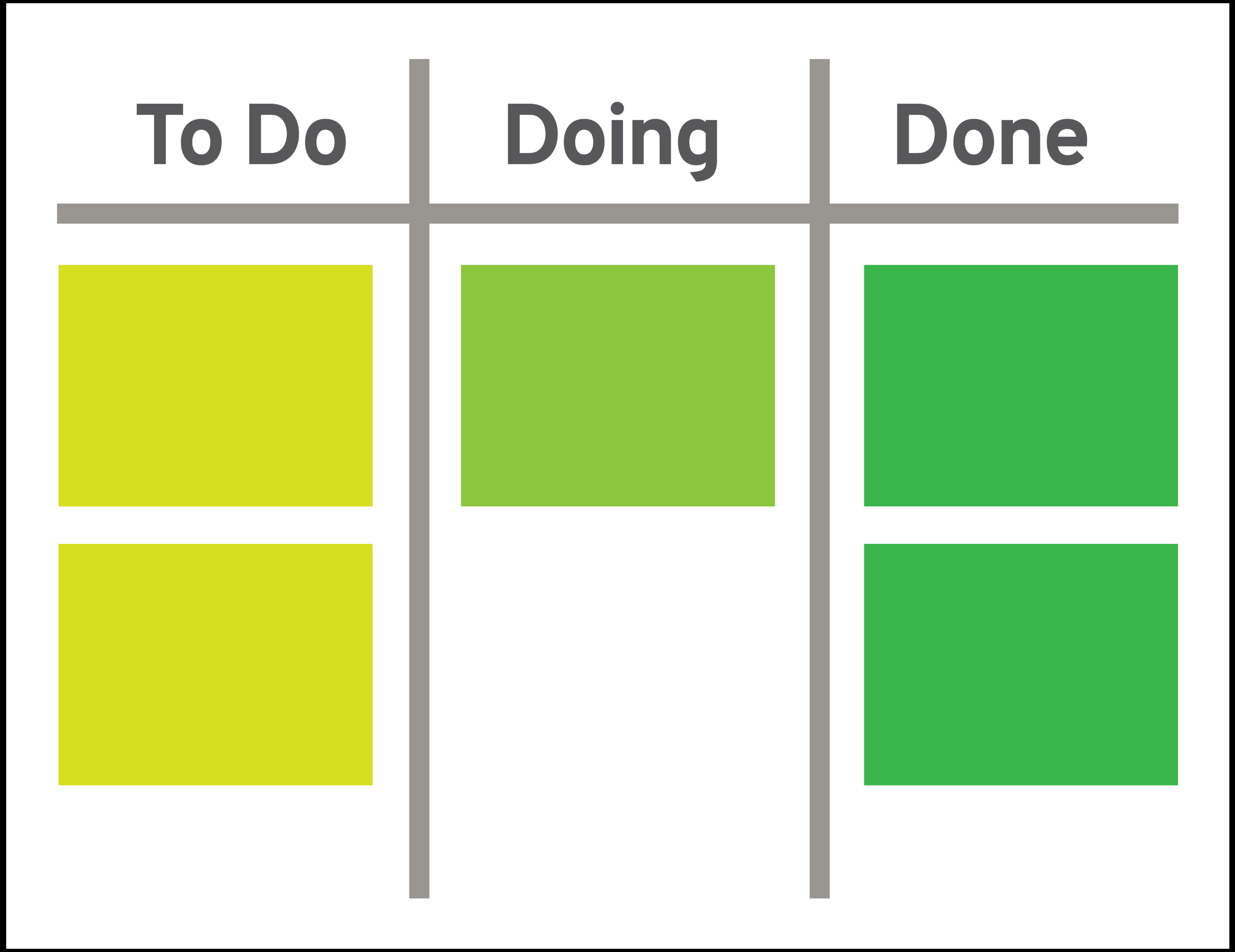
MoSCoW

Won’t Have/Would Be Nice To Have

- Customisation from the user - e.g. user-submitted graphs

Issue Tracking

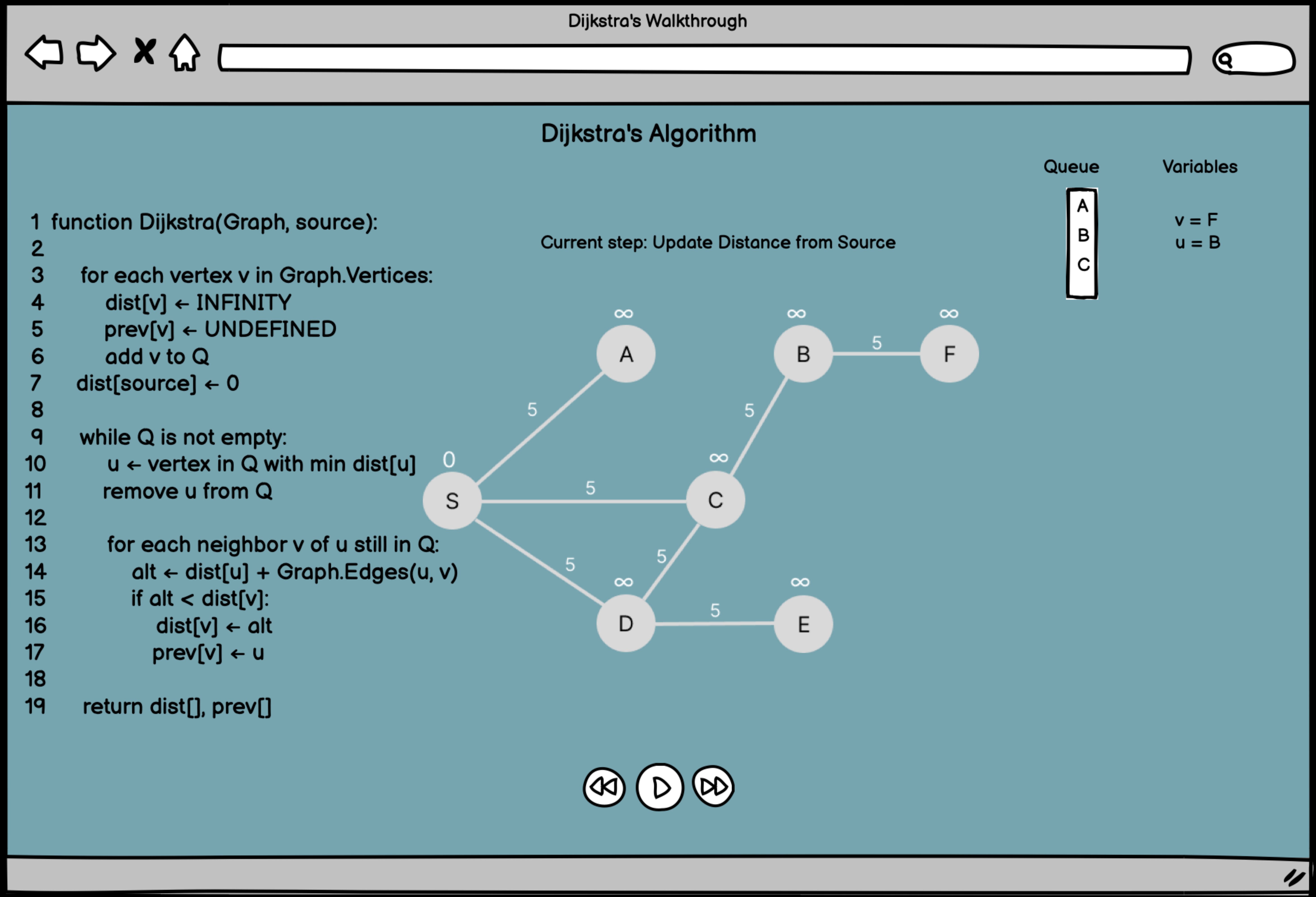
- Issues added to Kanban board



Design

Wireframes

- Three-column structure from algmatch
- Missing some information such as a legend, configurator
- Visualisation uses buttons to step through



Dijkstra's Walkthrough

Dijkstra's Algorithm

```
function Dijkstra(Graph, source):

 for each vertex v in Graph.Vertices:
     dist[v] ← INFINITY
     prev[v] ← UNDEFINED
     add v to Q
 dist[source] ← 0

 while Q is not empty:
     u ← vertex in Q with min dist[u]
     remove u from Q

     for each neighbor v of u still in Q:
         alt ← dist[u] + Graph.Edges(u, v)
         if alt < dist[v]:
             dist[v] ← alt
             prev[v] ← u

 return dist[], prev[]
```

Current step: Update Distance from Source

Queue

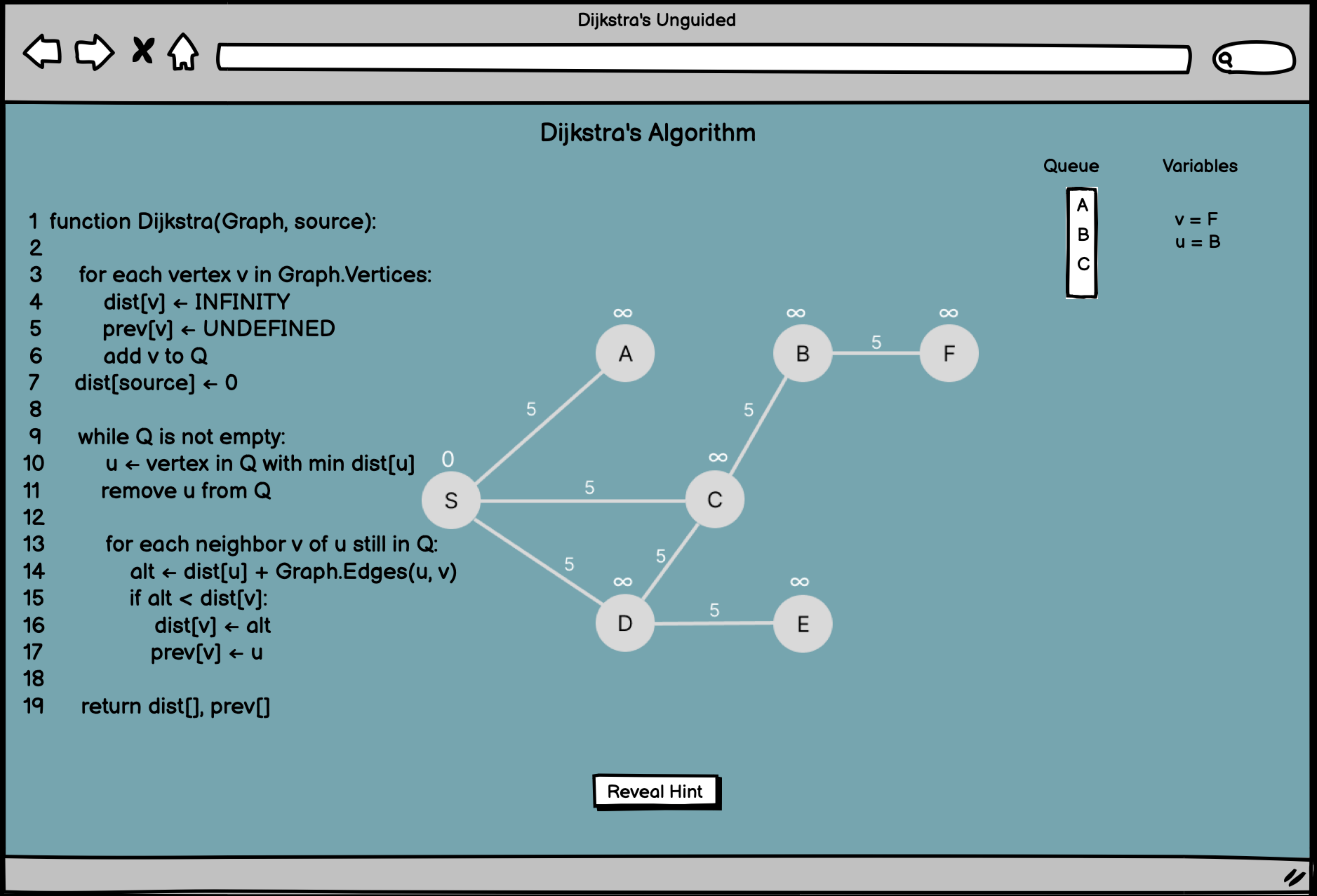
A
B
C

Variables

v = F
u = B

Wireframes

- DIY
- Interaction not decided yet



Dijkstra's Unguided

Dijkstra's Algorithm

```
function Dijkstra(Graph, source):

  for each vertex v in Graph.Vertices:
      dist[v] ← INFINITY
      prev[v] ← UNDEFINED
      add v to Q
  dist[source] ← 0

  while Q is not empty:
      u ← vertex in Q with min dist[u]
      remove u from Q

      for each neighbor v of u still in Q:
          alt ← dist[u] + Graph.Edges(u, v)
          if alt < dist[v]:
              dist[v] ← alt
              prev[v] ← u

  return dist[], prev[]
```

Queue

A
B
C

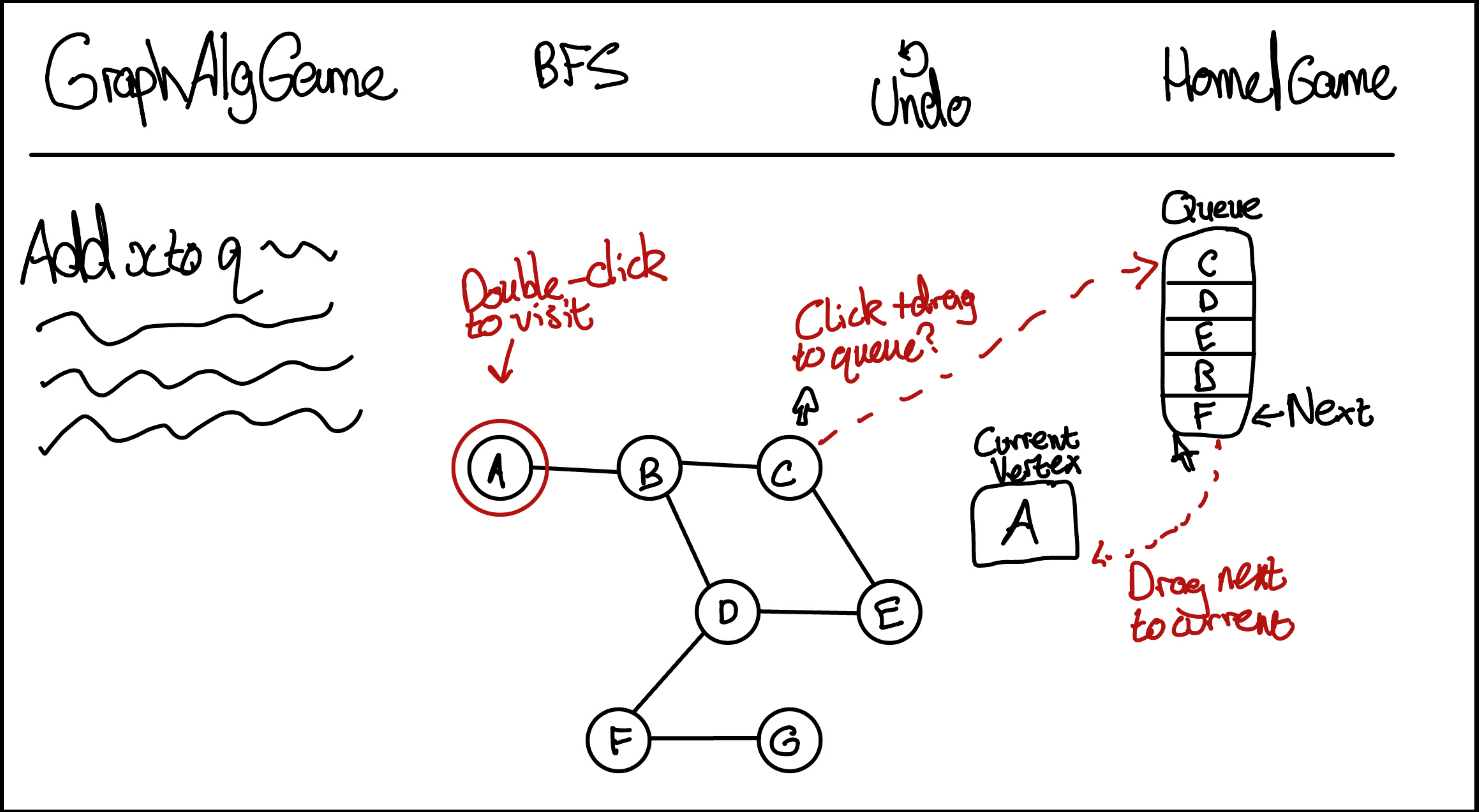
Variables

v = F
u = B

Reveal Hint

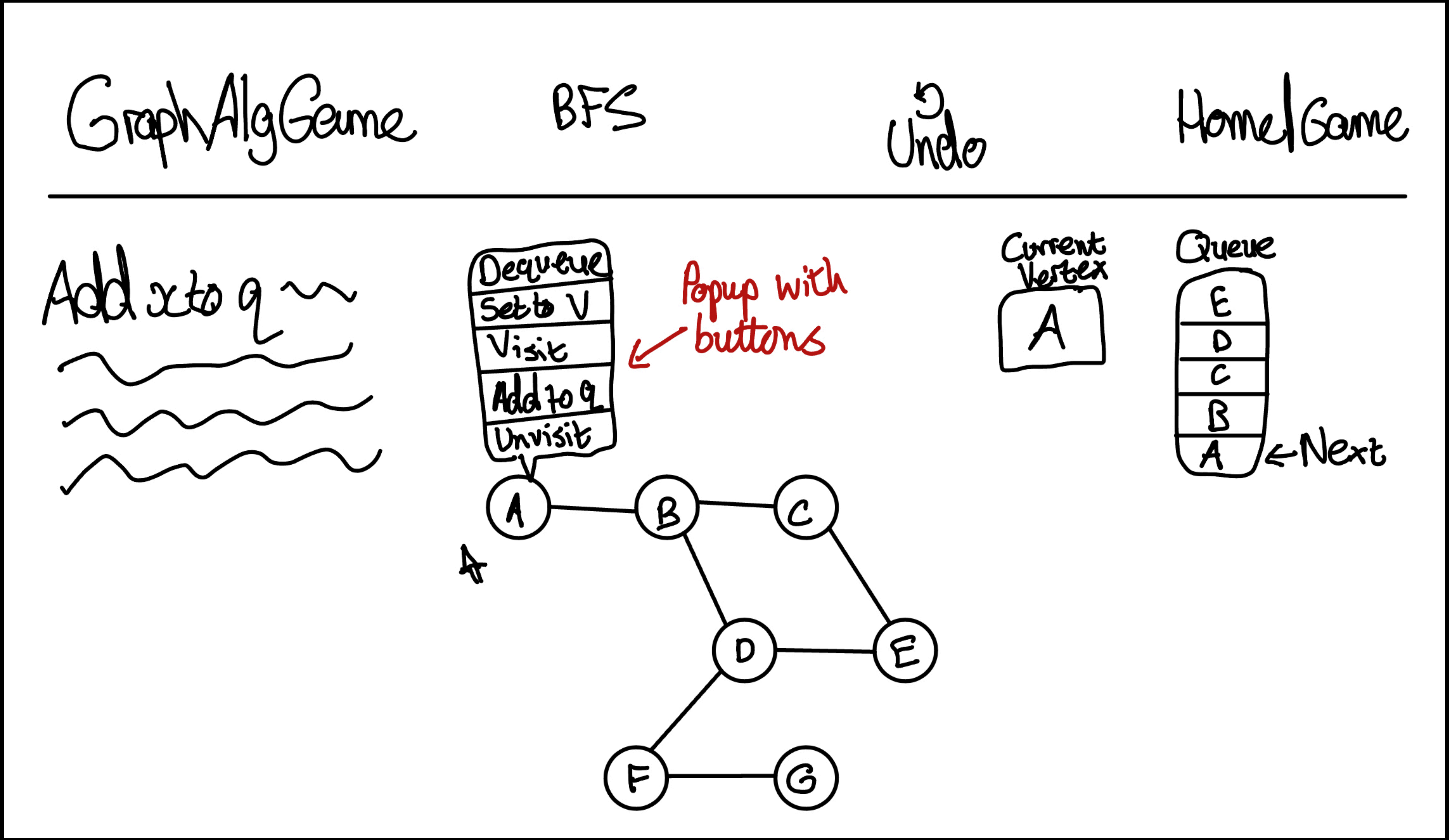
DIY Interaction

- Learned interactions
- Steep learning curve
- How to explain?



DIY Interaction

- Click a vertex
- Reveal options



Implementation

TS
tailwindcss

Demo!

Evaluation

Supervised Evaluations

7 participants

- Workflow
  - Let loose on website
  - Natural pattern was homepage -> BFS vis -> BFS DIY -> DFS vis etc
  - Asked their general thoughts
  - Asked preference of visualisation or DIY

Supervised Evaluations

7 participants

- Observations and Positive feedback
  - All participants engaged with the product
  - All participants learned/refreshed knowledge on the algorithms
  - Liked the UI
  - DIY consolidates learning

Supervised Evaluations

7 respondents

- Negative observations and feedback
  - Unsure how to start with DIY
  - Muscle memory instead of conscious decision-making
  - Not beginner friendly - jargon, difficult concepts

Unsupervised Survey

5 respondents (minimal experience)

- Workflow
  - Background info
  - Try the visualisations then answer questions (baseline)
  - Try the DIY then answer questions
  - General questions and written feedback

Unsupervised Survey

5 respondents (minimal experience)

- DIY improved the baseline of understanding
- All respondents chose the 2nd best option for ease of use of DIY UI
- Improved baseline of confidence in explaining graph algorithms from memory

Unsupervised Survey

5 respondents (minimal experience)

- All respondents found it fun and engaging
- Positive feedback in general
- Negative feedback included - DSPA difficult for beginners, and need more explanation of concepts on page.

Tests

Unit Tests and Manual Testing

- Unit tests with Vitest covering core functionality
- Manual testing for every feature

Conclusion and Future Work

Conclusion

- Requirements met
- Evaluation results
    - Users learn
    - UI improvement necessary

Future Work

- Improve UI/UX
- Implement more algorithms
- Colourblindness accessibility
- Back button

Media Used

- Kanban board - https://torak.com/wp-content/uploads/2016/04/kanban-board-simple3-1-02.png