GamesCrafters Spring 2020 Proposal

Jordan K, Hayden, Sophie

This semester we plan to work as a team to finish implementing the GUI for the 5x7 Connect 4. Last semester we were able to make the clickable game board and the red and blue tokens but we still need to work on the correct placement of the tokens and implementing ‘hints’. We hope to do it in a way that can be easily translated to fit the GUI for all other Connect 4 game dimensions.

Once we complete this, together we are leading a Design Team tasked with updating and creating logos, designing a coherent theme to span across all of the front end applications, and adding more creative and informative images to represent each game. This team will either be subgroup or work closely with the frontend web development team with Shein. It is important we work closely with the frontend team to make sure our designs look good on the website.

Game: Connect 4

Group Members: Sophie Yoo, Hayden Koerner, Jordan Knox

Our group began the implementation of the GUI for Connect 4. We began by writing out pseudo code for the game logic of Connect 4. After we were able to clearly visualize the game and the pieces we needed, we first began creating the 5x7 game board though formatting a grid. We were also able to create the game tokens as red and blue circles that fit inside the grid of the gameboard. The last thing we were able to complete was implementing the clickable areas on the board. As of now, each of the columns on the gameboard are highlighted so if a column is selected a token will appear in the column. Next time we wish to correctly format the placement of the tokens so that when a column is selected they are automatically placed on the appropriate square. For Connect 4, the appropriate square would have to be the lowest square in the column that is not currently occupied by a token. We also need to implement the hinting system and how/where we would like hints to appear on or around the board. Our goal is to implement the 5x7 Connect 4 GUI in a way that it can be easily translated to the other available board dimensions.

```
<template>
 <div id="app-game-board-connect4-regular">
   <svg viewBox="0 0 108 108" xmlns="http://www.w3.org/2000/svg">
     <defs>
       <g id="board">
         <path id="board-bar" d="M15.5,26.5 L92.5,26.5" />
         <use xlink:href="#board-bar" transform="translate(0 11)" />
         <use xlink:href="#board-bar" transform="translate(0 22)" />
         <use xlink:href="#board-bar" transform="translate(0 33)" />
         <use xlink:href="#board-bar" transform="translate(0 44)" />
         <use xlink:href="#board-bar" transform="translate(0 55)" />
         <path id="board-bar-j" d="M15.5,26.5 L15.5,81.5" />
         <use xlink:href="#board-bar-j" transform="translate(11, 0)" />
         <use xlink:href="#board-bar-j" transform="translate(22, 0)" />
         <use xlink:href="#board-bar-j" transform="translate(33, 0)" />
         <use xlink:href="#board-bar-j" transform="translate(44, 0)" />
         <use xlink:href="#board-bar-j" transform="translate(55, 0)" />
         <use xlink:href="#board-bar-j" transform="translate(66, 0)" />
         <use xlink:href="#board-bar-j" transform="translate(77, 0)" />
       </g>
       <circle id="turn-0-token" cx="21" cy="32" r="5" />
       <circle id="turn-1-token" cx="21" cy="32" r="5" />
       <circle id="hint" cx="21" cy="22" r="1" />
       <rect id="move" x="23.5" y="22" width="9.5" height="54" />
     </defs>
     <use xlink:href="#board"/>
     <!-- <use xlink:href="#turn-0-token"/>
     <use xlink:href="#turn-1-token" transform="translate(11,0)"/> -->
     <g v-for="cell in cellCount" :key="cell">
       <use
         v-if="boardData[cell].token === 'X'"
         xlink:href="#turn-0-token"
         :x="((cell - 1) % 7) * 11"
         :y="Math.floor((cell - 1) / 7) * 11"
       />
       <use
         v-else-if="boardData[cell].token === 'O'"
         xlink:href="#turn-1-token"
         :x="((cell - 1) % 7) * 11"
         :y="Math.floor((cell - 1) / 7) * 11"
       />
       <g v-else>
         <use
            v-if="nextMovesVisibility && boardData[cell].hint"
           :class="getHintClass(boardData[cell].hint)"
           xlink:href="#hint"
           :x="((cell - 1) % 7) * 11"
           :y="Math.floor((cell - 1) / 7) * 11"
         />
         <use
           :class="remoteness && 'move-pointer'"
           @click="remoteness && runMove(cell.toString())"
           xlink:href="#move"
           :x="((cell - 1) % 7) * 11 - 7"
           :y="Math.floor((cell - 1) / 7) * 11 + 5"
         />
       </g>
     </g>
   </svg>
 </div>
</template>

<script lang="ts">
import { Component, Vue, Watch } from "vue-property-decorator";

@Component
export default class GConnect47x5x4 extends Vue {
 cellCount: number = 7;
 boardData: {
   [cell: number]: { token: string; hint: string };
 } = this.initBoardData();

 get loadingStatus() {
   return this.$store.getters.loadingStatus;
 }

 get roundNumber() {
   return this.$store.getters.roundNumber;
```

```
  }

  get position() {
    return this.$store.getters.position;
  }

  get remoteness() {
    return this.$store.getters.remoteness;
  }

  get nextMoveDataArray() {
    return this.$store.getters.nextMoveDataArray;
  }

  initBoardData(): { [cell: number]: { token: string; hint: string } } {
    let boardData: { [cell: number]: { token: string; hint: string } } = {};
    for (let cell: number = 1; cell <= this.cellCount; cell++) {
      boardData[cell] = { token: "", hint: "" };
    }
    return boardData;
  }

  updateBoardData(): void {
    if (!this.loadingStatus) {
      this.boardData = this.initBoardData();
      for (let cell: number = 1; cell <= this.cellCount; cell++) {
        this.boardData[cell].token = this.position[cell - 1];
      }
      for (let nextMoveData of this.nextMoveDataArray) {
        this.boardData[+nextMoveData.move].hint = nextMoveData.moveValue;
      }
    }
  }

  runMove(move: string): void {
    let notbottom = true;
    let current = this.boardData;
    let pos;
    let column;
    let change = "                                   ";
    for (pos = 0; pos < length(current); pos = pos + 1) {
      if move[pos] == current[pos] {
        change[pos] = "-";
      } else {
        change[pos] = move[pos];
        column = pos % 7;
      }
    }
    while (notbottom) {
      if this.boardData[column-7] == "-" {
        //change move to be one column lower
        //
      } else {
        notbottom = false;
      }
    }
    this.$store.dispatch("runMove", move);
  }

  created() {
    this.boardData = this.initBoardData();
  }

  mounted() {
    this.updateBoardData();
  }

  @Watch("loadingStatus")
  onAsyncRoundChange(): void {
    !this.loadingStatus && this.updateBoardData();
  }

  @Watch("roundNumber")
  onSyncRoundChange(): void {
    this.updateBoardData();
  }
}
</script>
```

```
<style lang="scss" scoped>
svg {
 height: 15em;
 width: 15em;
 margin: auto;
 vertical-align: middle;
 > * {
   fill: none;
   stroke: var(--neutralColor);
   stroke-width: 1;
 }
}

#turn- {
 &0-token {
   fill: var(--turn0Color);
 }
 &1-token {
   fill: var(--turn1Color);
 }
}

.hint- {
 &win {
   stroke: var(--winColor);
 }
 &draw {
   stroke: var(--drawColor);
 }
 &tie {
   stroke: var(--tieColor);
 }
 &lose {
   stroke: var(--loseColor);
 }
}

#move {
 fill: var(--backgroundColor);
 fill-opacity: 0;
 stroke-opacity: 0;
}

.move-pointer {
 cursor: pointer;
}
</style>
```