Bejeweled

Tom Choi, Kiya Govek, Kiran Tomlinson, and Ryan Gorey

Project Goals

We want to design and implement a game modeled after the classic 2-D arcade game, Bejeweled. In Bejeweled, players are presented with a grid where each cell contains one of several types of gems. Users must swap adjacent gems to try and create matches of three or more like gems in a row for points. Matched gems disappear, and the new gems shuffle down from the top to fill the gaps. If the user tries to swap two gems that won't create any matches, they are returned to their original positions. Bejeweled is also a timed game - users gain a few extra seconds for each match, but once time reaches zero seconds remaining, the game ends. The ultimate goal for the user is to rack up as high a score as possible before they run out of time.

Instead of designing the visuals around different types of jewels, we may choose to focus our graphics around a different theme. The name of the game would also change.

Planned Features

The features we plan to implement are:

- Players will be able to click two adjacent gems to swap them.
- The gems will swap if there is a match, or not swap if there is not a match.
- The game will check for matches when new gems fall.
- Players will get points and extra time based on the amount of gems in a match they create.
- There will be a timer, which will end the game when it reaches 0.
- Players can increase the amount of time they have left by making matches.
- If there are no possible matches, the board reshuffles all the current gems
- Gems which are easily distinguishable by shape, not just by color.

The features we will implement if we have time are:

- Hint button which shows the player a single possible move if clicked.
- Special jewels that are formed if the player makes an impressive match, which give bonuses.
- A tiered level system, where the game increases in difficulty either by locking particular gems in their positions or adding additional types of gems.

Design Patterns

We plan to use the MVC design pattern. In this design pattern, the view will show the grid board and listen for user clicks. The model will store the state of the board, and the controller will interpret user clicks to play the game. The MVC design pattern itself implements the strategy and observer patterns. The GUI interface will also inherently implement a composite design pattern.

Anticipated Team Roles

For the most part, we will be collaborating on each part of the program. However, we have split the project into a few pieces and assigned leaders to each piece. The leader will ensure that their part of the program is functioning properly, and will be in charge of leading the coding in their area.

- Create interfaces for the view, model, and controller (Kiya)
  - Ensure that classes are properly encapsulated
  - Strive for loose coupling
- Implement the model (Ryan)
  - Store and manipulate the game state
  - Design adjacency algorithms to test for gem matches
  - Communicate with the view
- Implement the view (Kiran)
  - Display the game
  - Get input from the user
- Implement the controller (Tom)
  - Communicate with the view and the model
  - Interpret user actions such as button clicks
- Sprites/graphics (Ryan)
  - Create gem images
  - Make UI

Mock Ups

Bejeweled

MENU

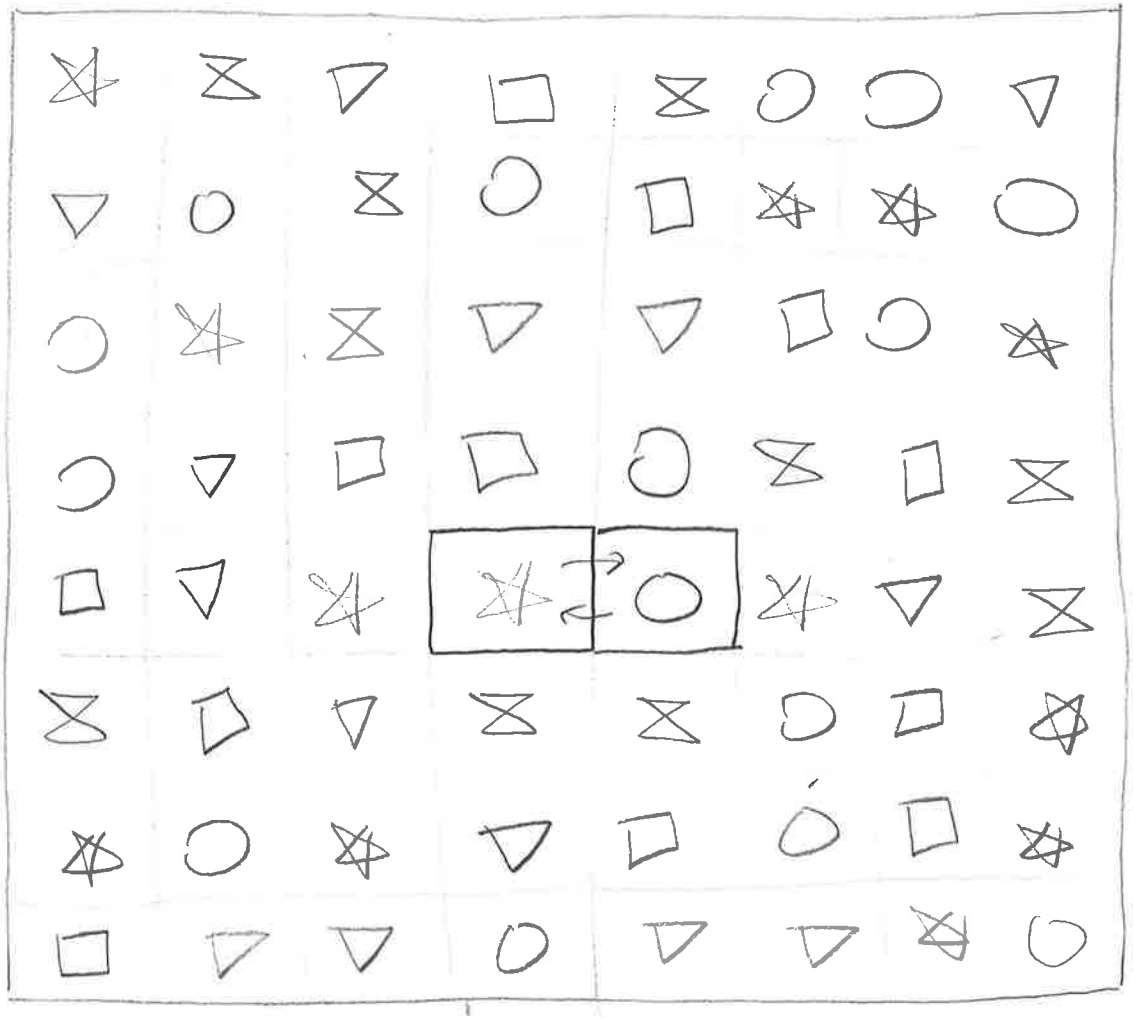
SCORE

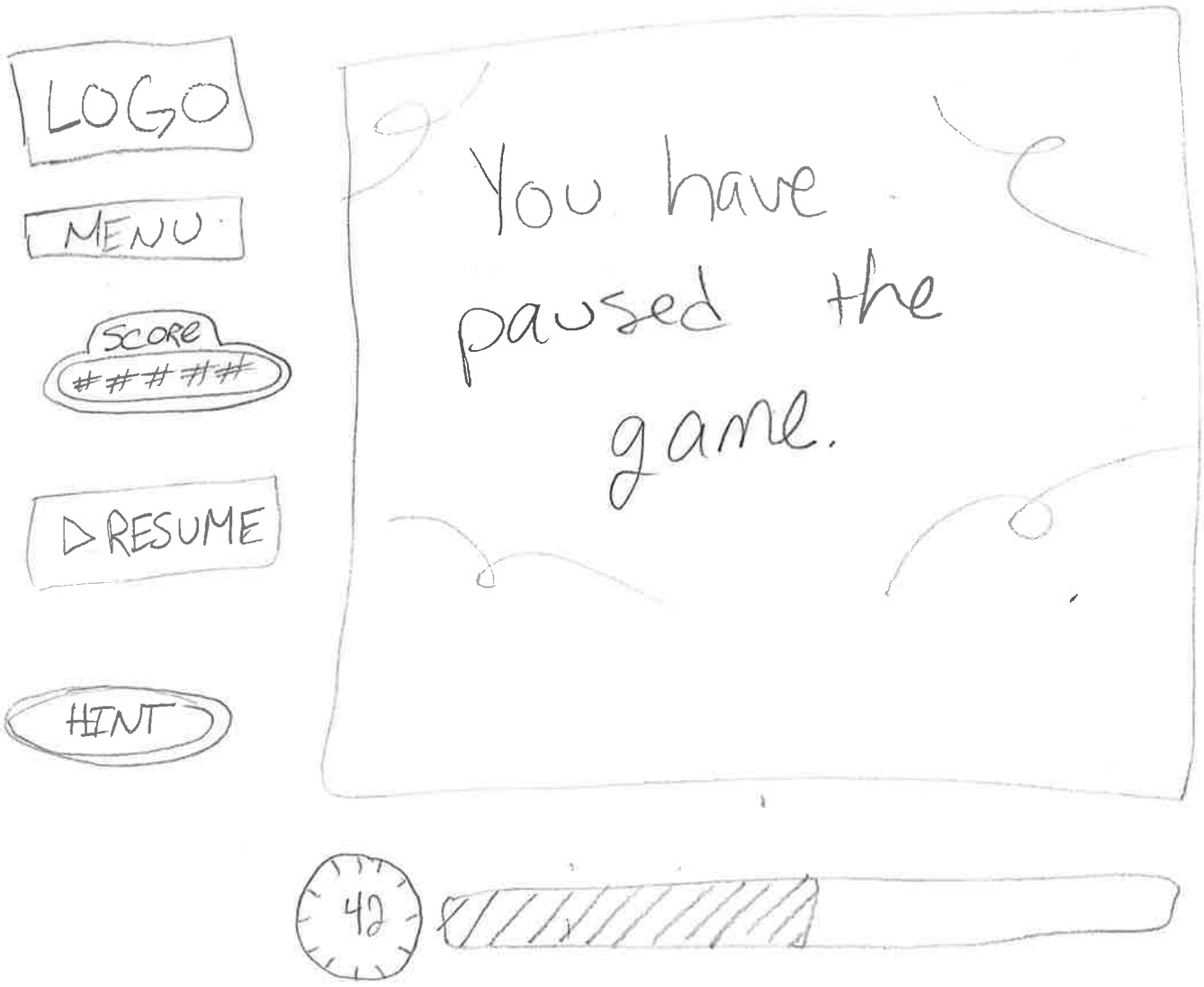
#####

PAUSE

HINT

87




LOGO
MENU
SCORE
#####
▷RESUME
HINT
You have paused the game.
42

Bejeweled

MENU

SCORE

PAUSE

HINT

87

MENU

READ ME

SEE HIGHSCORES

NEW GAME

INSTRUCTIONS