Sorry!

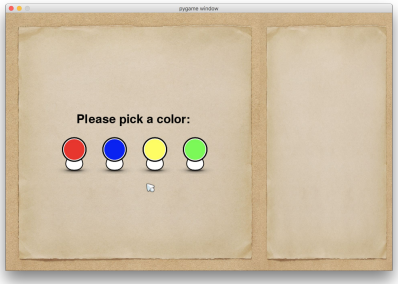
Chia-Chun Chao, Joe Embrey,
Gavin Gunkle, Peter Macksey

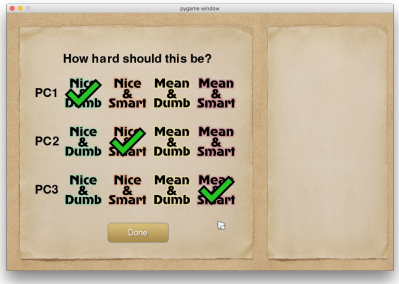
Outline

- Summary
- Description of Components
  - Board GUI, game logic, pawn movement
  - Instruction, computer AI, testing
  - Main menu GUI, statistics, possible movement, images
  - Database, deck, save/resume
- What worked
- What did not work
- What we would do differently
- Demo

Summary

- Provide detailed instructions of this game
- Play the game through an interactive interface
- Provide fancy GUI
- Enter player's name and select player's color
- Decide the number of computers, how nice and how smart they are



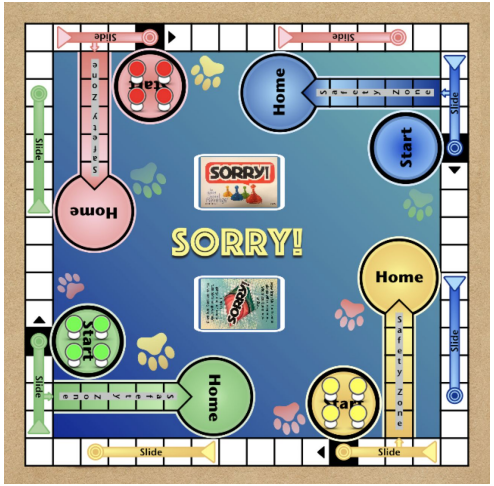


Summary

- Relaxed start button for easy start
- Game logic:
  - Draw a card
  - Select an option
  - Click the first pawn / Skip
  - (Click the second pawn)
  - Next turn
- Computer draws a card and selects an option
  - Select a pawn based on how smart and how nice they are
- Save the current game and resume later
- Game summary after someone wins

Description of Components (1/4)

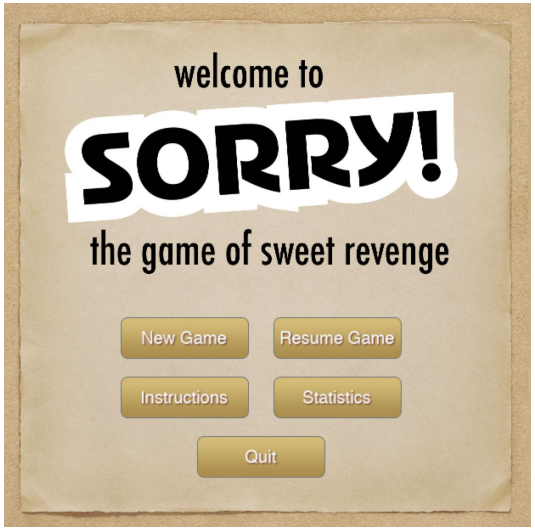
- Board GUI
  - Imitate the real board and draw our board
  - Mapping pawn's position to the real position in the screen
- Pawn movement
  - Make pawns move one step at a time
- Game logic
  - Change from clicking pawns to clicking draw card button
  - Currently
    - Draw a card
    - Find possible moves for the card
    - Select an option
    - Click pawns
    - Process movement for the card



Description of Components (2/4)

- Instructions
  - How to play our computer version of SORRY!
- Computer AI
  - Scoring system used to calculate computers move
    - System loops through all possible moves and gives a score
  - Four different computer options to simulate
    - Nice & Dumb computers receive a lower score with moves that bump opponents but does not calculate moves strategically
    - Nice & Smart computers receive a lower score for bumping a pawn but calculates the best move by moving from start, into the safety zone or into home
    - Mean & Dumb computers receive a higher score for bumping a pawn but does not calculate moves strategically
    - Mean & Smart computers receive a higher score for bumping a pawn, moving into the safety zone, moving into home or getting a pawn out of start
- Testing
  - Attempts to play through all parts of the game to find any bugs and polish before presentation

Description of Components (3/4)

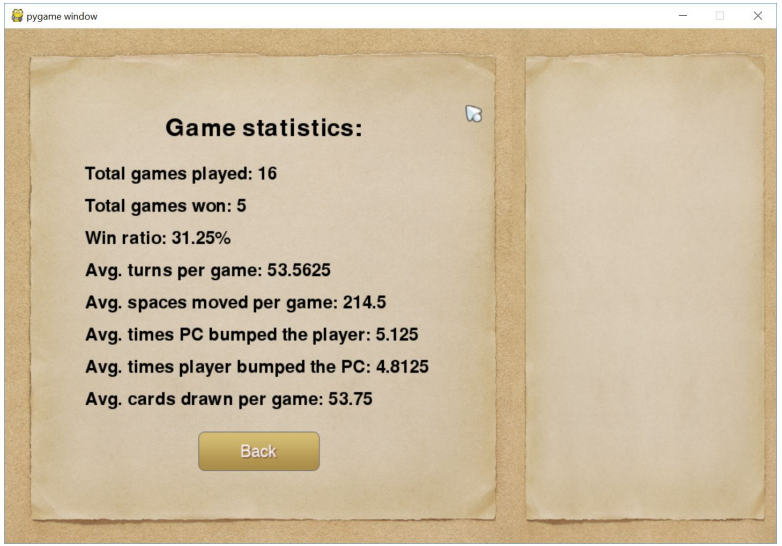


- Main menu
  - Start and configure a new game
    - Enter your name, select a color, the number of opponents and their difficulty
  - Resume the previous game
  - Read the instructions
  - View overall game statistics
- Statistics
  - The end game summary shows statistics from your game
  - The main menu statistics page hows cumulative information from all games
- Movement
  - Pawns that could be moved are highlighted to help guide the user
    - This helps keep you from having to count your moves
- Images
  - Created custom graphics to fit our needs such as the board

Description of Components (4/4)

- Database
  - Used MySQL-connector
  - Used to save game stats and data
  - Writes to database at end of game with that data (Name, Date, AI types, # of moves, etc)
  - Reads from database when “Statistics” button is clicked to return some game statistics
- Deck
  - Complete deck of sorry cards
  - Shuffles upon starting game and reshuffles when all cards have been drawn
- Save/Resume
  - Auto saves game every time a card is drawn and also with save button
  - Stores all game data in pickle module
  - Loads the data back in when clicking resume and sets values from that
  - Saves all player and pawn info, deck order, current card, game stats, etc

Description of Components 4/4



What worked

- Meeting together every week
  - Met for at least 2 hours a week in a library study room, more towards end of project
  - First talked about what we worked on in the past week
  - Then went over problems and worked them out together
  - At the end we discussed what we would work on and aim to complete by next meeting
- Lots of communication via Facebook
  - Problems or bugs in code worked out together via Facebook messages
  - Kept things moving instead of discussing problems only in meetings
- Dividing work at start of process
  - Each person took on different parts of the project after we designed our components
  - Allowed us to continue work without having to wait for one another to finish one part
  - Continuous integration via Github
  - Tied our individual parts together as we went along

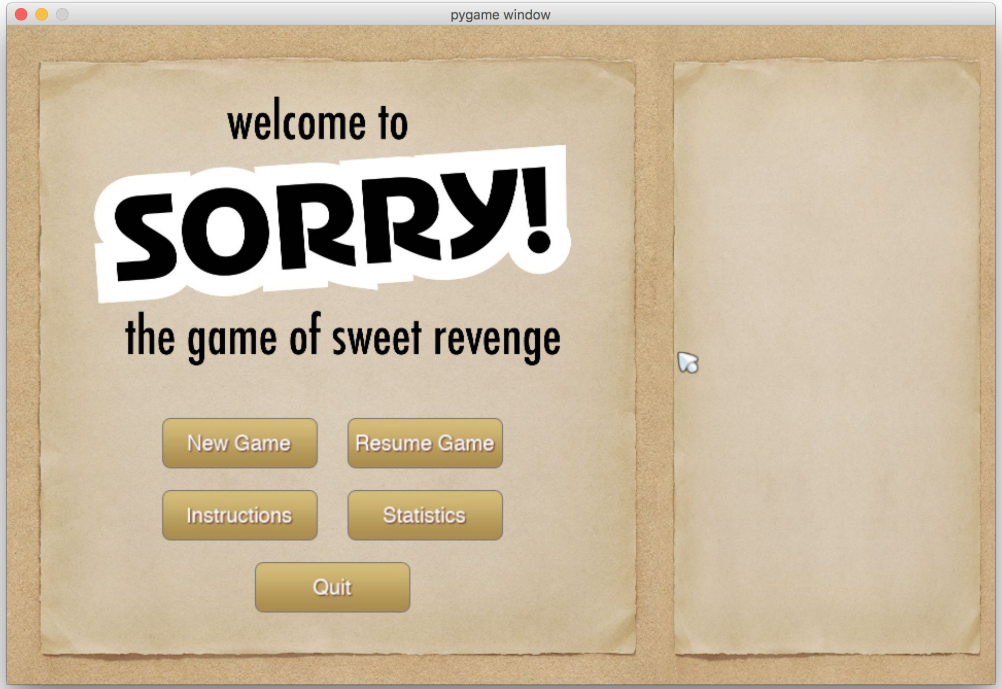
What did not work

- Github problems
  - Merge conflicts
  - Overwritten code
- Database issues
  - Issues with everyone being able to access the database (early on)
- Pygame
  - Old and somewhat out of date, making it difficult to construct certain components of the game
  - Issues with clicking and typing in window
- Creating an executable
  - Difficult to do with pygame

What we would do differently

- Better coordination with Github
  - Avoid merge conflicts
  - Avoid overwriting code
- Not use Pygame
  - Not the most useful software to use for creating this game
  - Took longer to figure out than expected, and had to plan to do the game differently after
  - Slow and difficult graphics
- Use different database
  - Not use UVM database
- Planning
  - Spend a little more time at the beginning of the project planning some of the bigger components
  - Plan in time to learn pygame

Demo


pygame window
welcome to
SORRY!
the game of sweet revenge
New Game
Resume Game
Instructions
Statistics
Quit