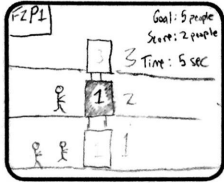


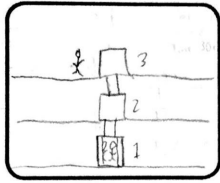
Final Project Ideas

Michael Zachor

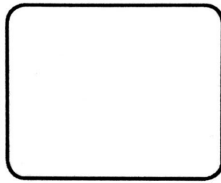
Idea 1 - Elevator Operator (Game Option)



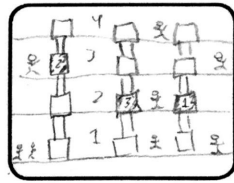
People come in randomly and you have to take them to their floor before time runs out



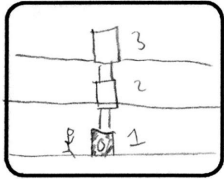
Number on elevator is # of people in elevator. Box with color is where elevator is.



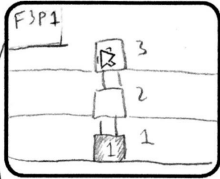
Top-left box says how many people want to go to each floor.



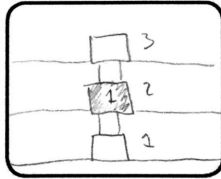
Levels get harder but you can buy stuff with XP like a bigger elevator.



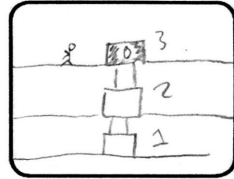
Person gets in elevator, requests floor. You click on floor where elevator should stop.



Floor 3 is desired by one person. So click on floor 3.



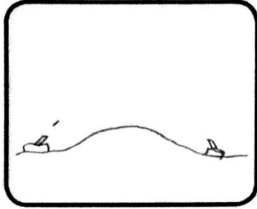
elevator goes up.



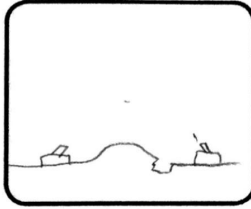
and the person exits at their level.

- UX: mouse clicks
- Time, Score
- Harder levels
- <http://www.crazygames.com/game/i-love-traffic>
- <https://www.coolmathgames.com/0-papas-freezeria>

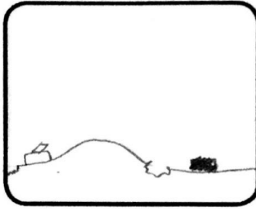
Idea 2 - Tanks (Game Option)



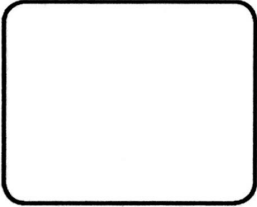
You are a tank. Set your controls (angle, power, location) and shoot the other tank.



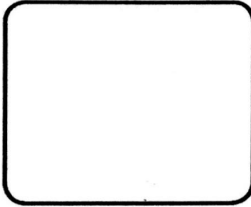
Wind acts as a force pushing against your bullet, so you have to factor that in.



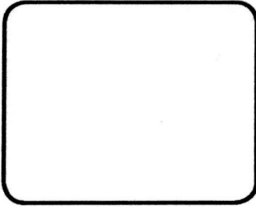
If you shoot the opponent 3 times you win.



Player vs. CPU, arrow keys to move and set shooting angle, space to shoot (and set power?)



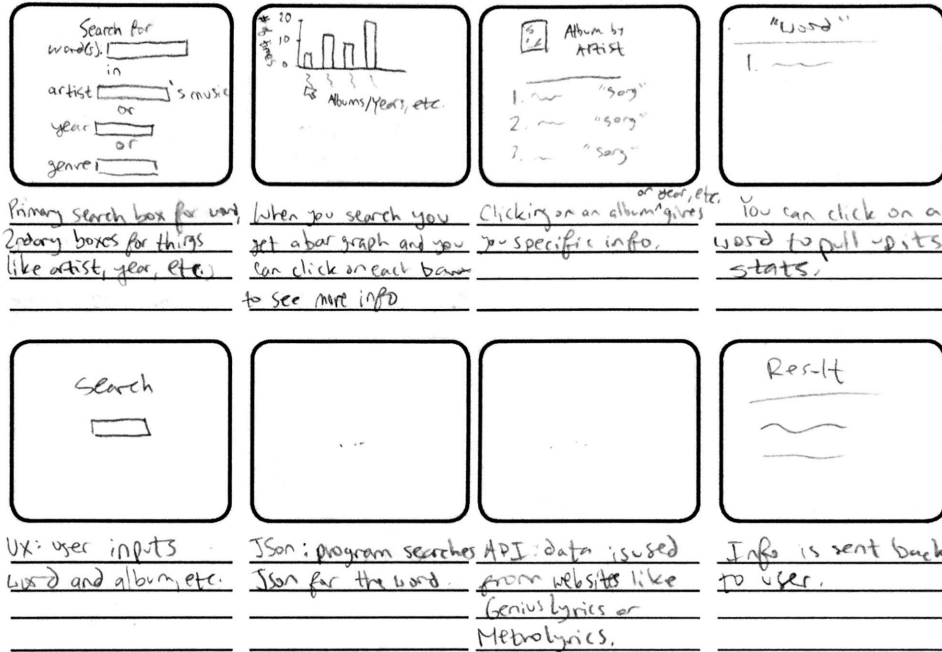
the program takes in what you give it, responds, and sends back the data.



The CPU would pick random but constrained numbers, like pick an angle between 60 and 70°.

- UX: arrow keys, "space"
- CPU responds
- Gravity and Wind
- <http://www.crazygames.com/game/tanks>

Idea 3 - Lyrics Search (Choose Your Own)



- UX: search
- API: Lyrics (MetroLyrics, Genius)
- CPU Response: bar graphs
- More info
- <https://medium.com/svilenk/data-visualization-uncovering-the-hidden-layers-of-hip-hop-lyrics-e6f97be1a932>
- <https://www.promptcloud.com/blog/data-visualization-text-mining-taylor-swift-song-lyrics>

Idea 4 - Music Recommendation (Choose Your Own)

First user answers questions about the music they like, how often they listen to music, etc.

The program takes this info, uses an API like Gracenote to check for similarities with music listeners

Program gives user a list of artists they might like.

User can select artists they already know and the system will replace them w/ others.

For a more detailed response user could also specify albums they like as well as albums + artists they don't like.

- UX: answer questions, eliminate results
- API: music data (Gracenote), match with user
- <http://www.gracenote.com/music/music-discovery/>
- <http://www.gnoosic.com>
- <https://www.music-map.com>
- [SMooACP.jpg](#)