

DREXEL GAME DEVS



Meet A.D.A.M.!

TEAM MEMBERS AND ROLES

Joseph Lipinski - Project Lead, Level Designer

Sarah Kushner - Game Artist

Zachary Lopez - Technical Lead

Matthew Freihofer - Player Programmer/Physics Programmer

Patrick Nwanah - A.I. Programmer

Cory Cellucci - Programmer

PROBLEM TO BE ADDRESSED

Bored? Do you have nothing to do? Especially not reading assignments or homework assignments during finals week?

Instead of being bored, why not play our game!?



BROADER IMPACT

As addressed in our Developer Documentation, this game is easily extensible! To anyone who wants to create a game in Python (specifically Pygame), this is a great way to start!



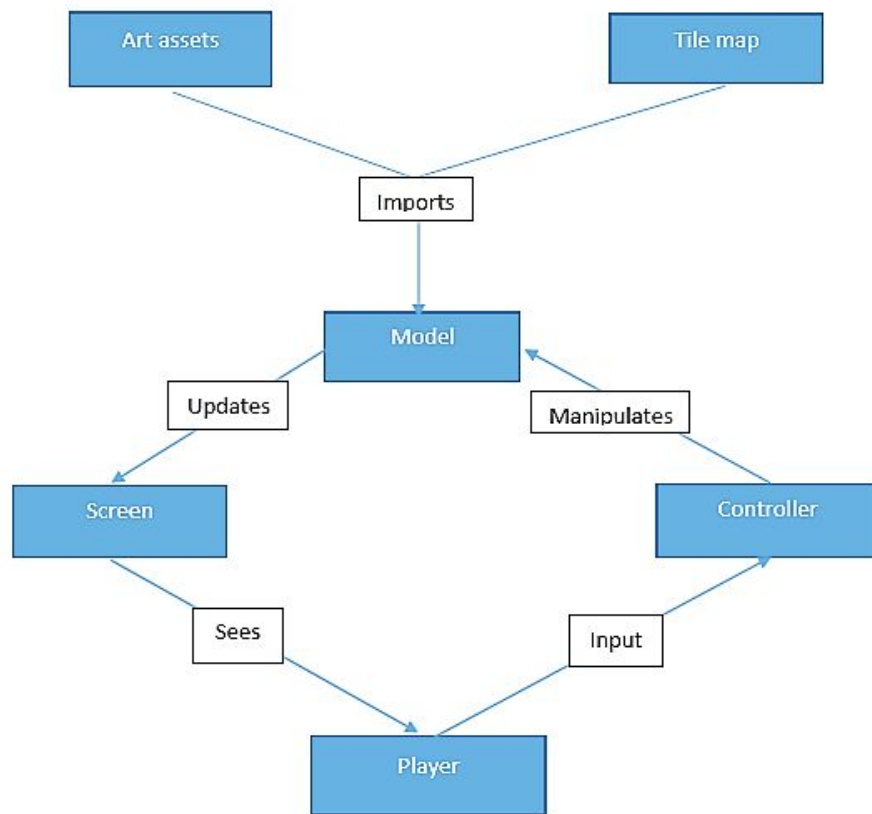
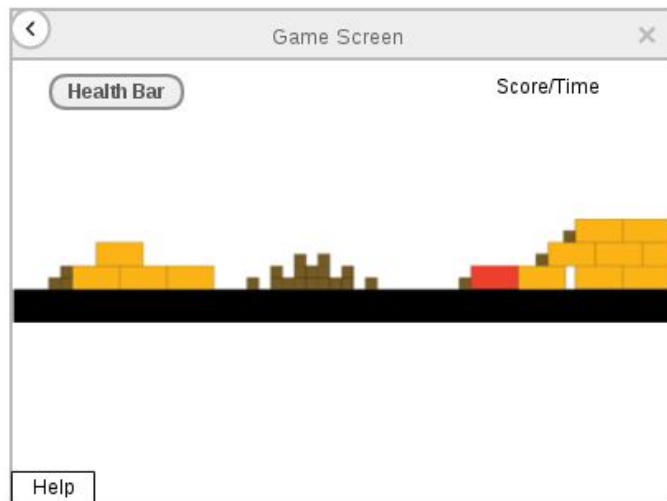
Each member of our group also had the chance to try completing a task they had never done before this class! Very exciting!

WHAT WE DID

- Procedural level generation
- A.I. behavior responsive to player's position
- Physics affects enemies and player alike
- 2D platforming and side-scrolling shooting
- 8-bit art style
- Scoring system

HOW WE DID IT

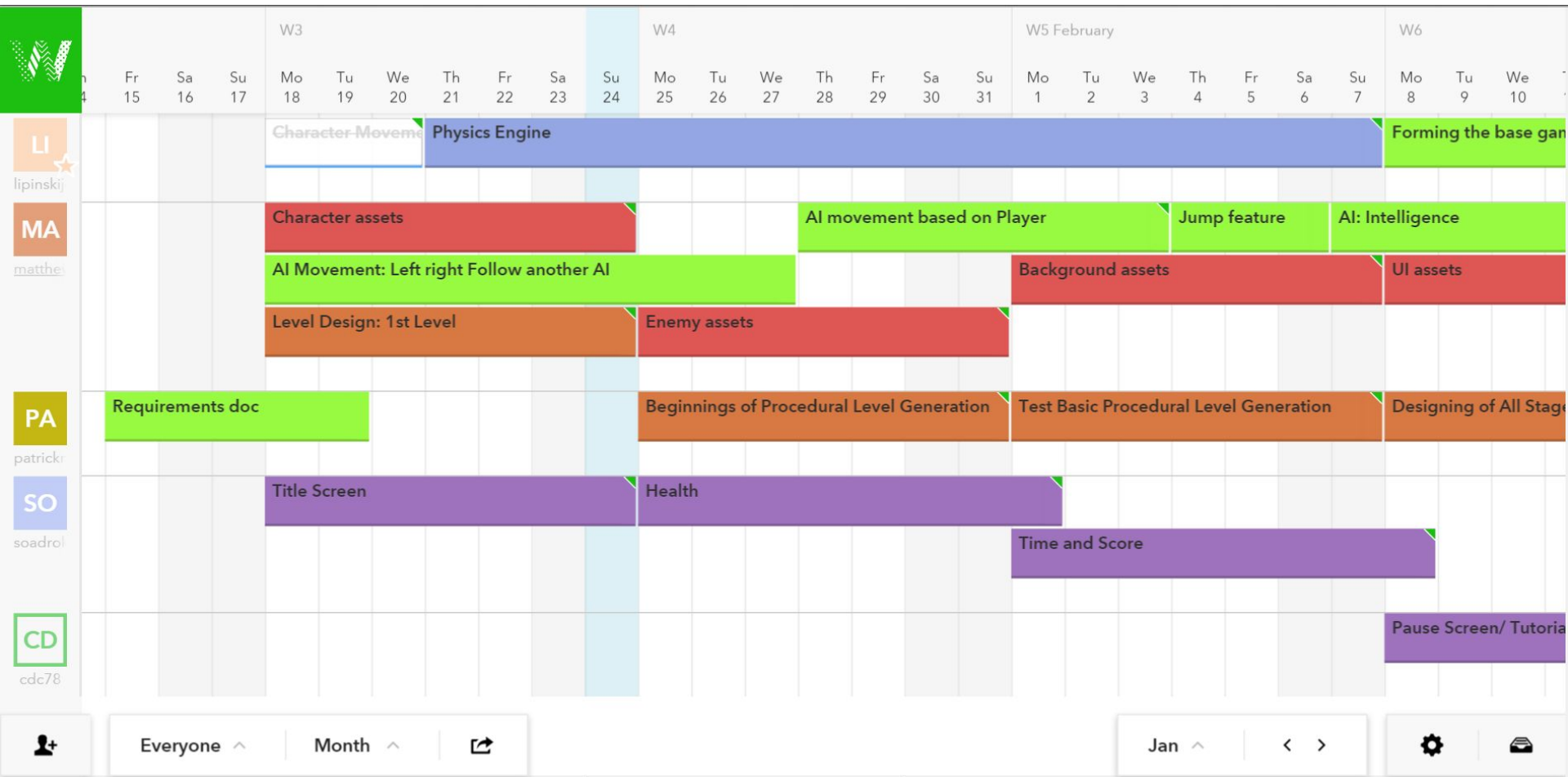
Design

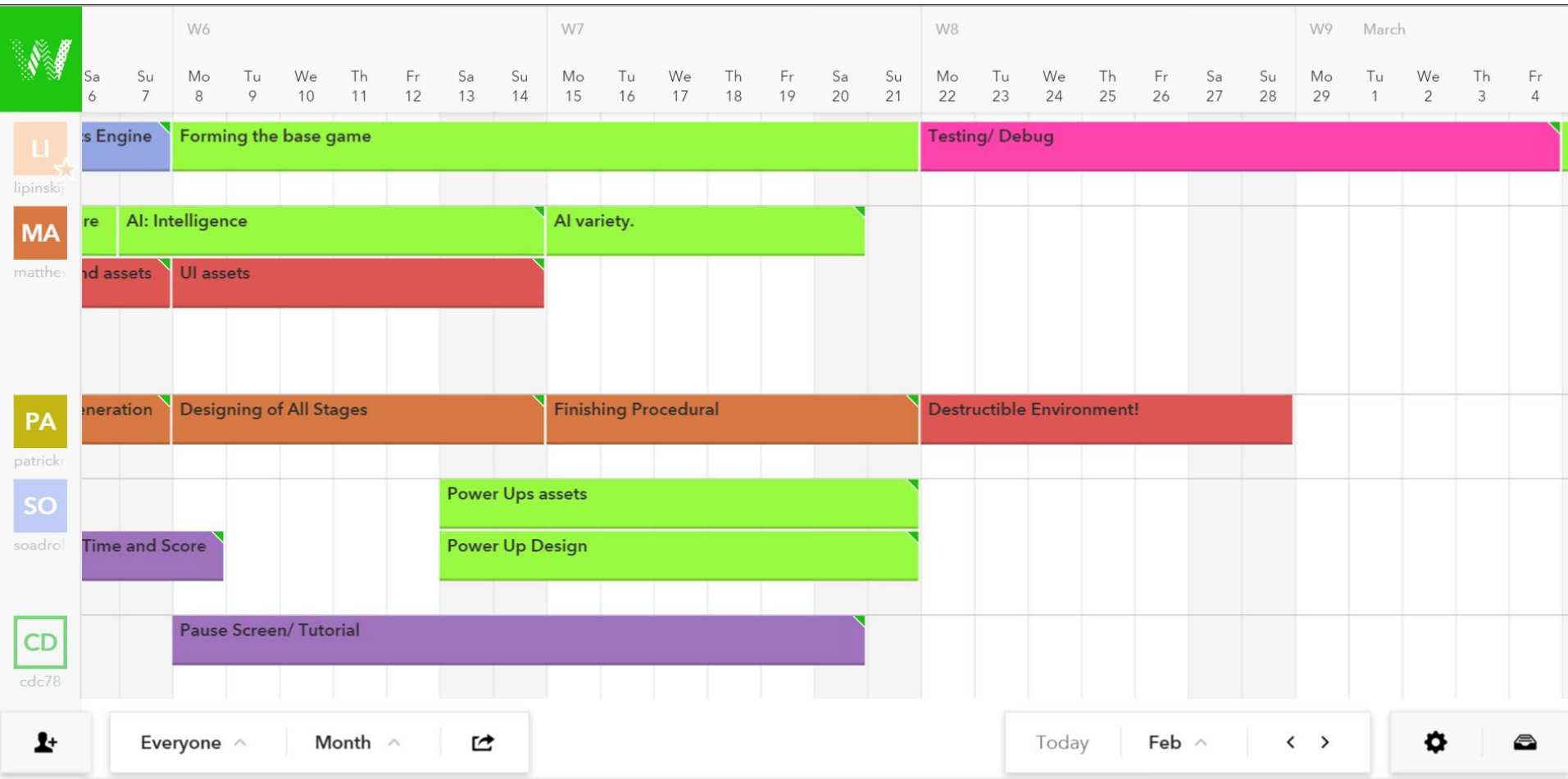


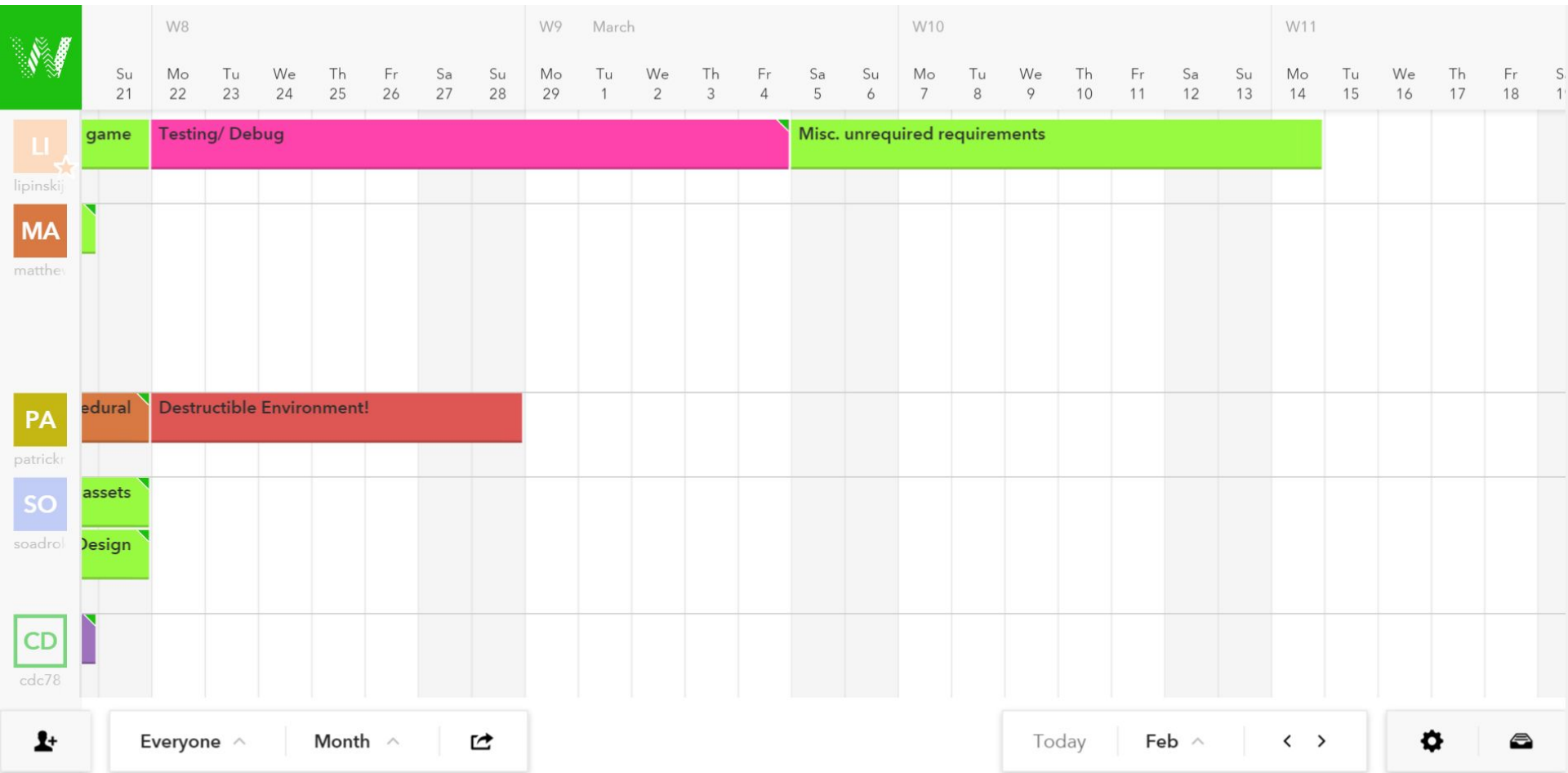
PLAN

- Our design document and team roles made for convenient independent work
 - Easier to collaborate by combining parts in the end
- Gantt Chart to show responsibilities, timeline, and dependencies
- Incorporate feedback from playtesting









HOW IT ACTUALLY WENT

Some setbacks:

- General time management
- Pygame/Tiled/PyTMX version issues
 - Led to a plan change
 - Needed another feasible way to procedurally generate levels
- Slower progress than expected, but we still did it!



SETBACKS ELABORATED

Original Plan was to use Tiled/PyTMX for streamlined Map Creation. Issues with this are listed below:

- PyTMX requires specific versions of Pygame (2.7/3.2)
- PyTMX requires specific Python libraries (libraries not inherently apart of 2.7 or 3.2).
- PyTMX requires specific Pygame libraries (libraries that were discontinued after Pygame 2.7).
- Pygame 2.7 binaries do not mesh with 3.2 binaries
- The list goes on... You get the picture.

Spent a week sifting through issues. Abandoned the idea. Why?

- Too much dev time wasted. We cut our losses. Looked for another solution.

SETBACK SOLUTION

- Much of our original code was still useful, thanks to modular design.
- We looked for solutions online that seemed to use our individual modules.
- Found a plethora of tutorials. We stitched our code together following their format, and was able to produce the final product.

LESSONS LEARNED

1. We all developed a better understanding of Python, and Game Development in general.
2. We learned a hard lesson regarding reinventing the wheel.
 - a. We set out to individually develop our own AI system; our own Physics engine; our own Procedural generation.
 - i. While this, without a doubt, gave us a better understanding of each topic, we may have learned more if we had used other tools instead.
3. All of us now know the value of documenting and planning (Thank you Professor Mongan :D !).

DEMO!