VizDoom Pathfinding AI

NATHAN BALCARCEL

WASHINGTON STATE UNIVERSITY

LAWRENCE HOLDER

Problem

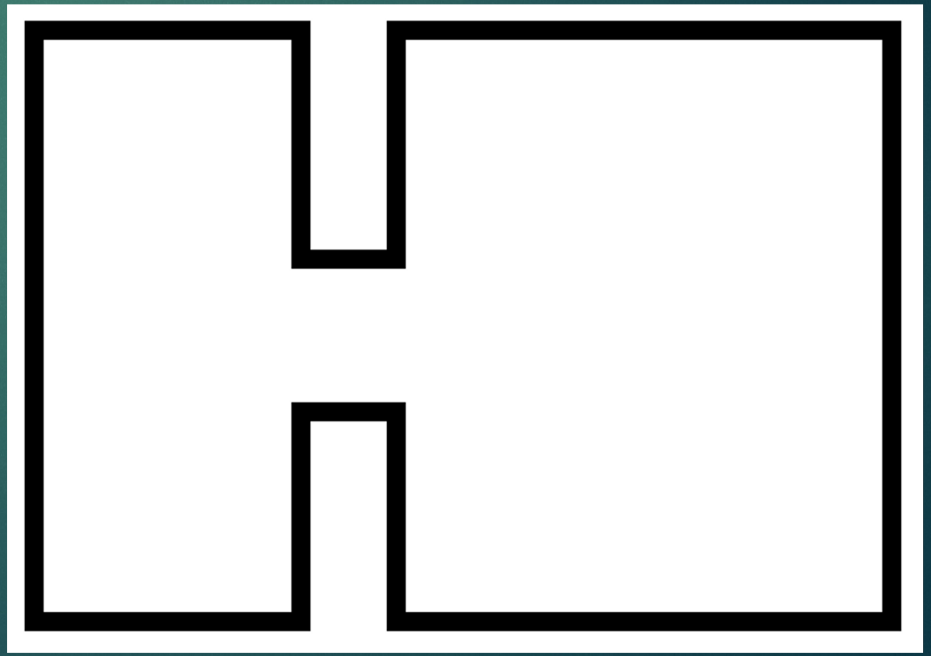
The goal was to explore different pathfinding and navigation techniques within the game of VizDoom. The new approach must replace the older q-learning agent that exists now.

- Make pathfinding efficient and scalable
- Allow pathfinding to be interfaced from a higher module

Approach

Agent has access to the coordinates of the walls.

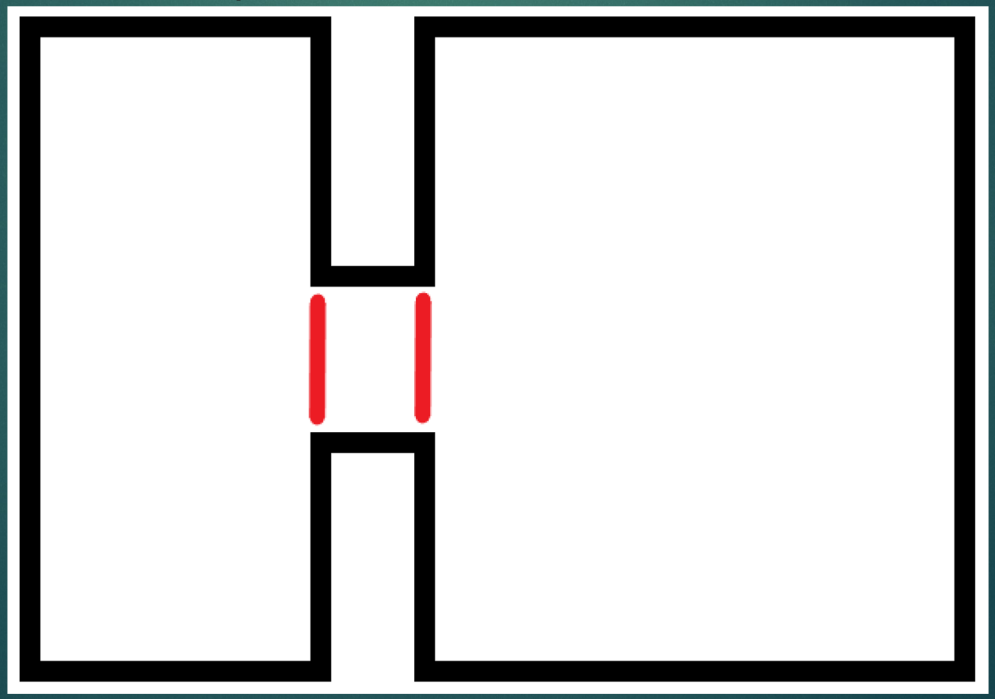
- Ended up defining rooms and doorways, then pathfinding from doorway to doorway
- Requires no training, very flexible with new rooms and maps



Approach

Step 1: Define doorways

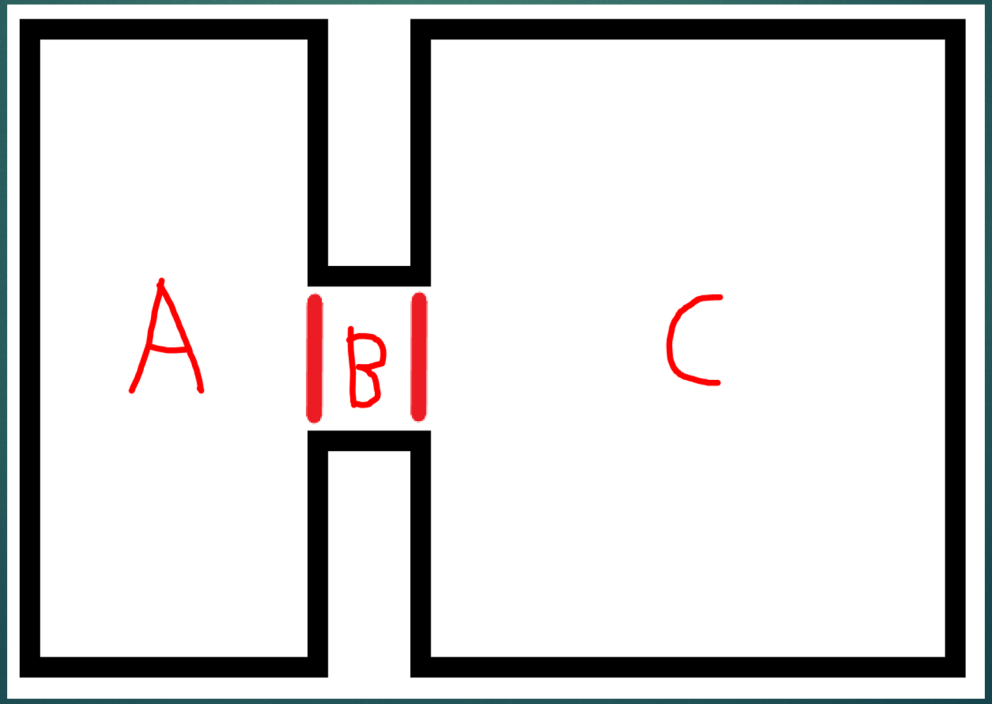
- Find parallel lines that point to each other and draw a doorway between them



Approach

Step 2: Define rooms

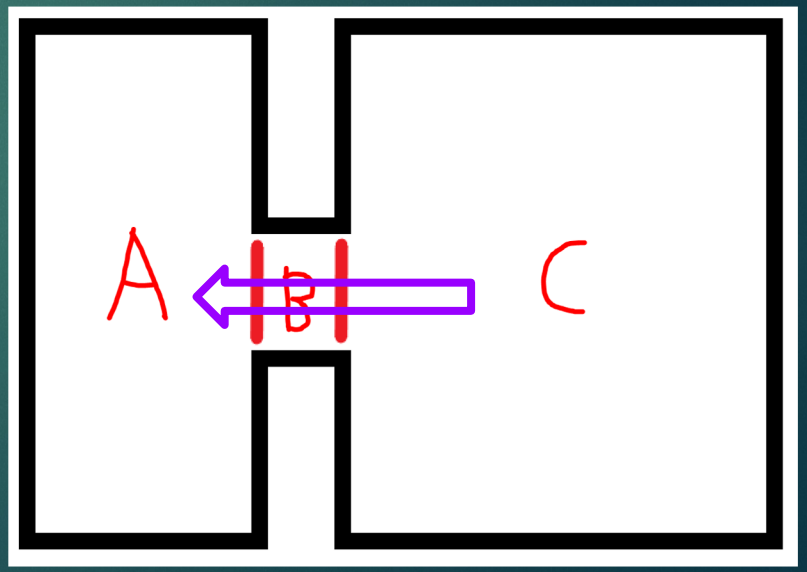
- Find the areas bounded by walls and doorways and define them as a room



Approach

Step 3: Find a path of doorways we must navigate through

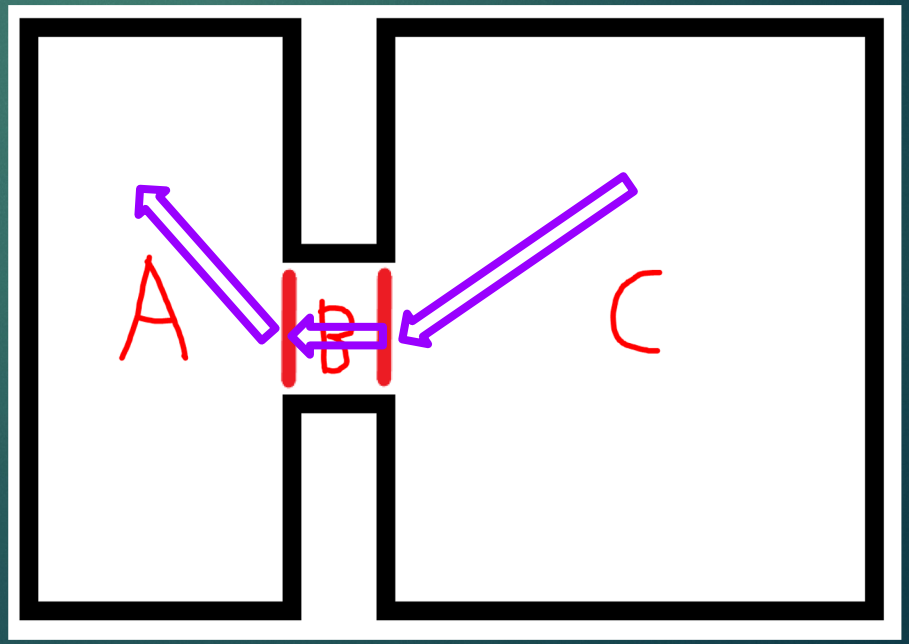
- Find a path of rooms from the starting room to the destination room (i.e. {C, B, A})
- Convert this into a path of doorways we must navigate through (i.e. {(C,B), (B,A)})



Approach

Step 4: Navigate to the next doorway

- Find a path from the current location to the midpoint of the next doorway.
- Once we're past the last doorway, pathfind to the destination.



Evaluation

New technique was evaluated based on two criteria: interfacing ability and performance

- Navigation skill is controlled by a higher module, which will decide where to move and when to divert attention to another skill, makes difficult decisions
- Pathfinding must be efficient, both in move count and real processing time
  - Consider performance in new maps it has never seen

Results

Approach uses breadth-first search, which should find an optimal path. Additionally, every next move is retrieved manually

- Easily able to control when to start, stop, continue, and cancel the navigation
- Performance is similar no matter how many times the agent has seen a map, non-optimal algorithms in areas of the code

Conclusions

- Impacts
  - Smarter bots in video games, with less dependence on the current maps and faster pathfinding times.
  - Possible applications to real life robotics i.e. search and rescue, delivery
- Next steps
  - Implement diagonal smoothing, improve hitbox detection, keep squashing bugs
  - Grey out areas of map, improve complexity of algorithms

Thank You

- Thank you for attending our presentations
- Questions?