Security Cameras: Visually Vigilant

1.
WHAT'S THE PROBLEM

Big Picture

- Develop an application that allows a user to graphically illustrate the positioning of security cameras to view all areas of an enclosed location with obstructions.

Smaller Details

- Allow the user to manipulate the space
  - Be able to retain it
- Allow user to gain visual feedback of coverage from placement of a camera
- Allow movement of a camera, and update visuals accordingly

Smaller Details

- Calculate and Display % of total coverage of room
- Consider placement constraints of camera
- Allow user to be able to tell if a camera can see a point
- Compute minimum number of cameras necessary to cover room

What I did

- C# Program
  - Using Windows Presentation Forms
- Classes:
  - Room
    - Camera
    - Walls
  - BitmapPixelMaker

What I did

- Allow user to draw room via wall building
- File options for room
  - Start New Room
  - Save/Load Room
  - Print as png
  - Clear current Room
  - Exit

What I did

- Cameras can be placed, and moved within the walls
- Calculated Area of Room
- Computed minimum number of cameras necessary
- Allow user to “mute” camera

Exceptions

Optimal Camera Placement

Crafting an algorithm to calculate the minimum number of cameras is relatively straightforward based on the number of walls in the room. Calculating the locations of these optimal points is not.

Calculate Location and Cameras of Object

Not 100% sure if I got this or not. I showed the visuals of each camera over each space it can see. However, if the user were to hover their mouse over a pixel, it does not show which camera numbers can see that space.

Disclaimer: These Exceptions would make for some great possibilities for the next ~~poor soul~~ willing student to improve upon this project in future years. I may even continue to tinker with some of these on my own after graduation.

Exceptions

Update movement and placement of cameras in real time

I almost tanked out my work laptop several times throughout this, and blew out the stack many more. The heft of the algorithms and the number of them to build the rooms and adjust cameras requires serious horsepower, as it is all but in name ray tracing. I opted for the update in “real time” option.

Anytime multiple actions are stacked on top of each other

Loading a new document after clicking the wall option allows you to draw walls over an existing room, which isn’t the end of the world, just not what I was aiming for. However, this causes cameras to color outside the room. Also backing up on a wall and then opening a new room allows you to draw two walls simultaneously.

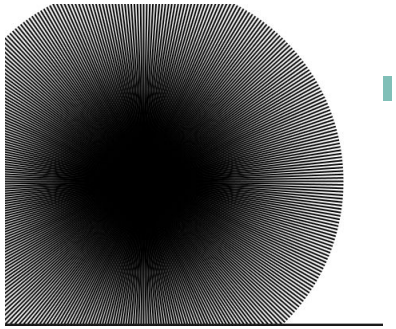
Disclaimer: These Exceptions would make for some great possibilities for the next ~~poor soul~~ willing student to improve upon this project in future years. I may even continue to tinker with some of these on my own after graduation.

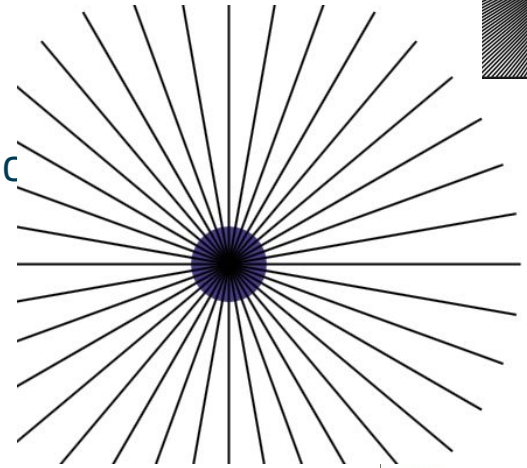
Methods

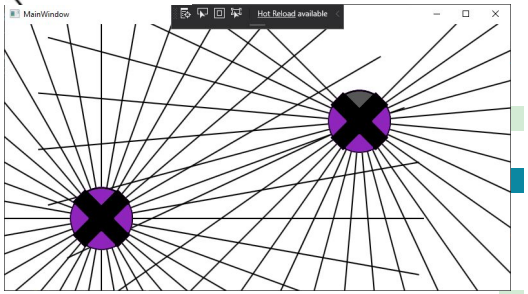
Began playing with how to colc from a camera

This included how to create a line, and how to draw from a point.

This was eventually done by using supercover_line, some angles, and a WriteableBitmap which was modified pixel by pixel.



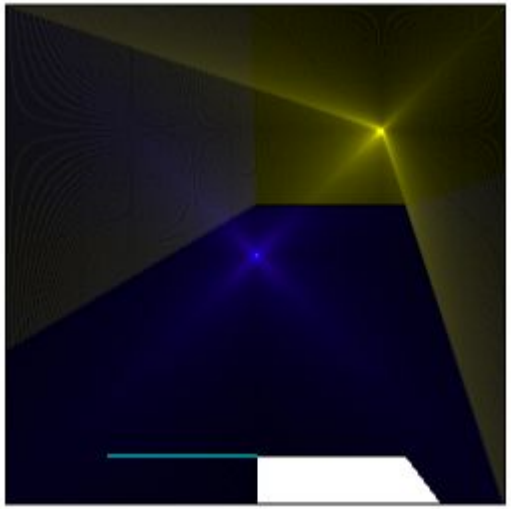


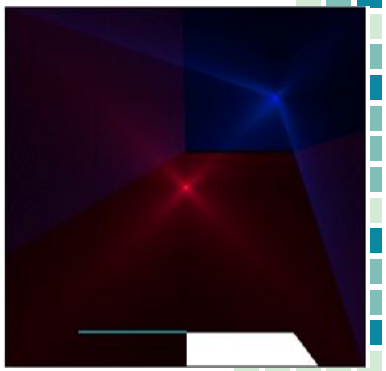


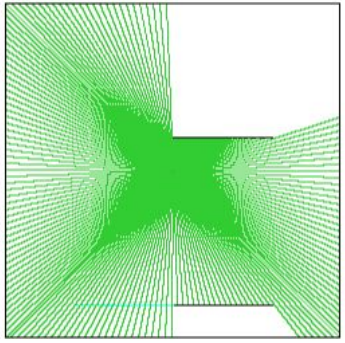
Methods

Eventually used the WriteableBitmap functions to deal with color collision

This structure carried me through most of the project, and hasn't needed to be changed much since this was completed.



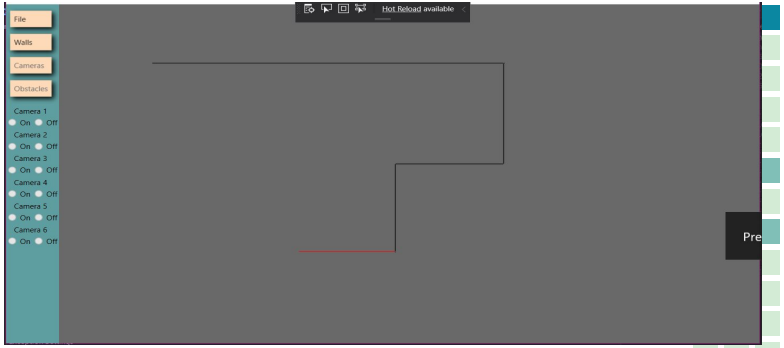


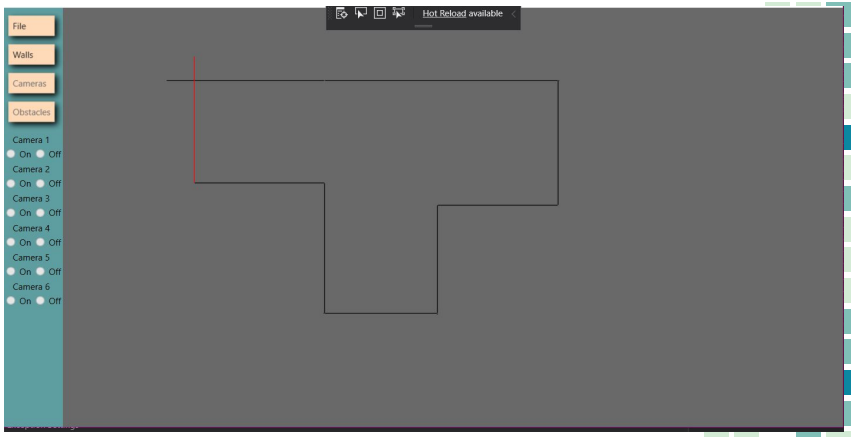


Methods

Used MouseEvents to create Walls and place Cameras

Uses color of the pixel in order to determine feasibility of placement as well as show field of view.

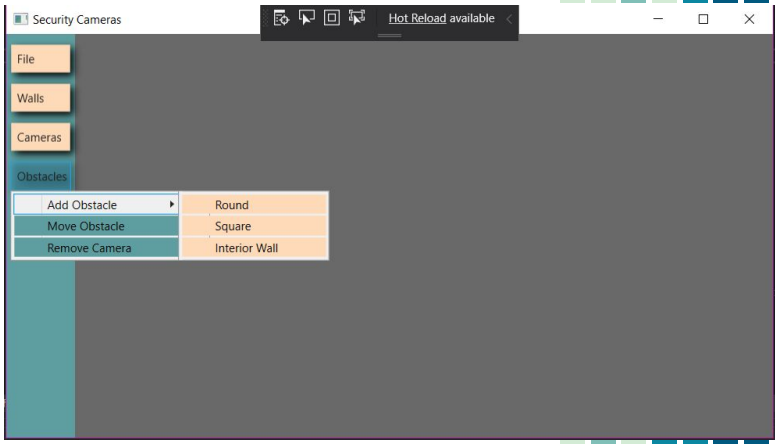


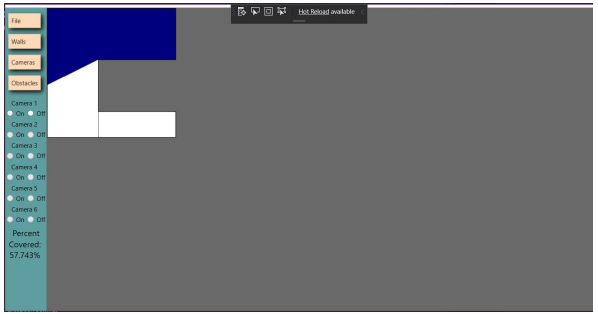


Methods

Created Menu

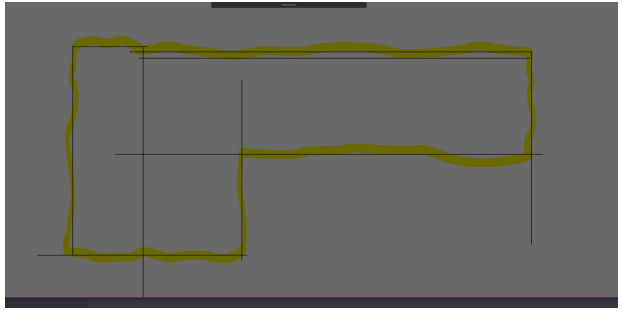
This menu drives the program, containing buttons which all lead to the functions which build this program.

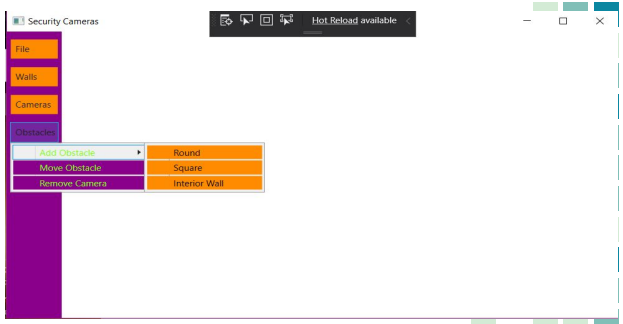


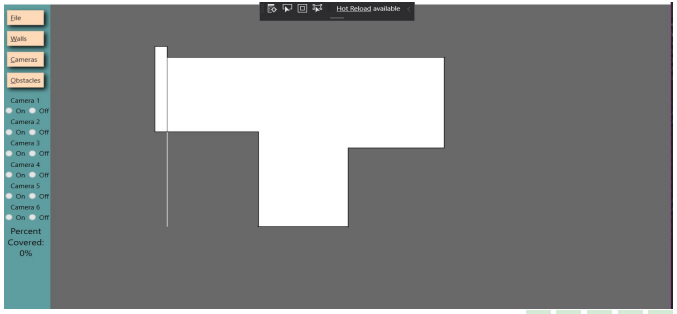


Issues

Aside from almost bricking my computer, I ran into a lot of issues of Out of Bounds on calculations. Stack overflows were common as well.







2. DEMONSTRATIONS

File

Walls

Cameras

Obstacles

Camera 1
On Off

Camera 2
On Off

Camera 3
On Off

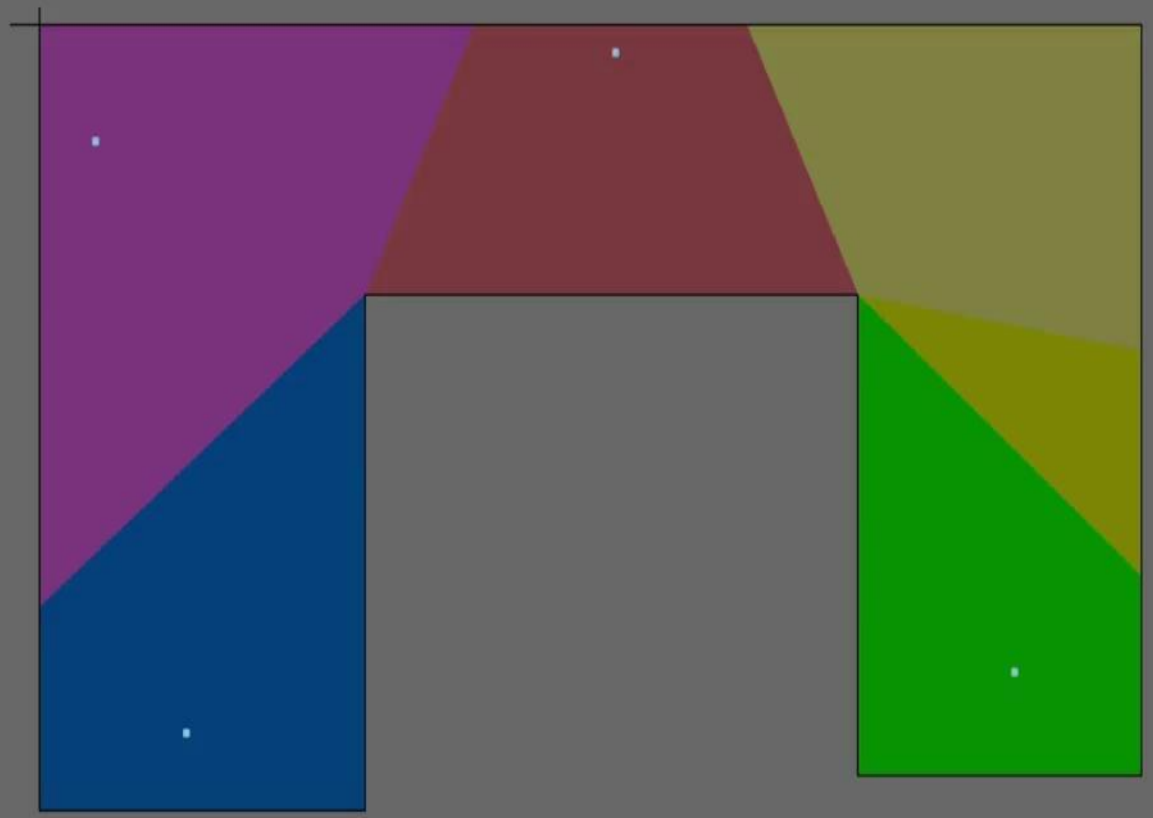
Camera 4
On Off

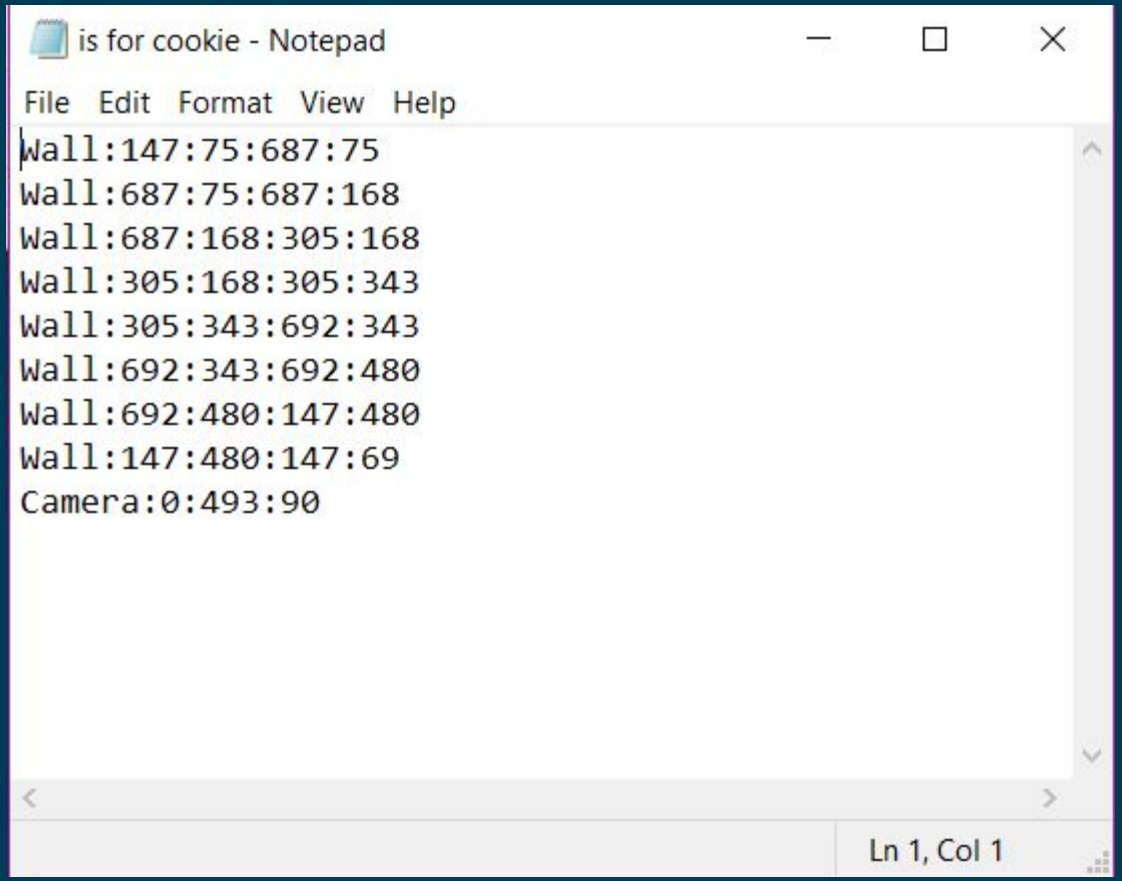
Camera 5
On Off

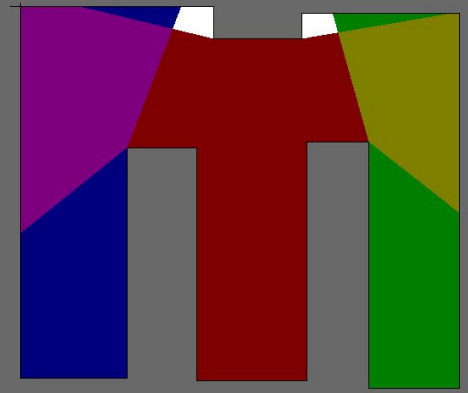
Camera 6
On Off

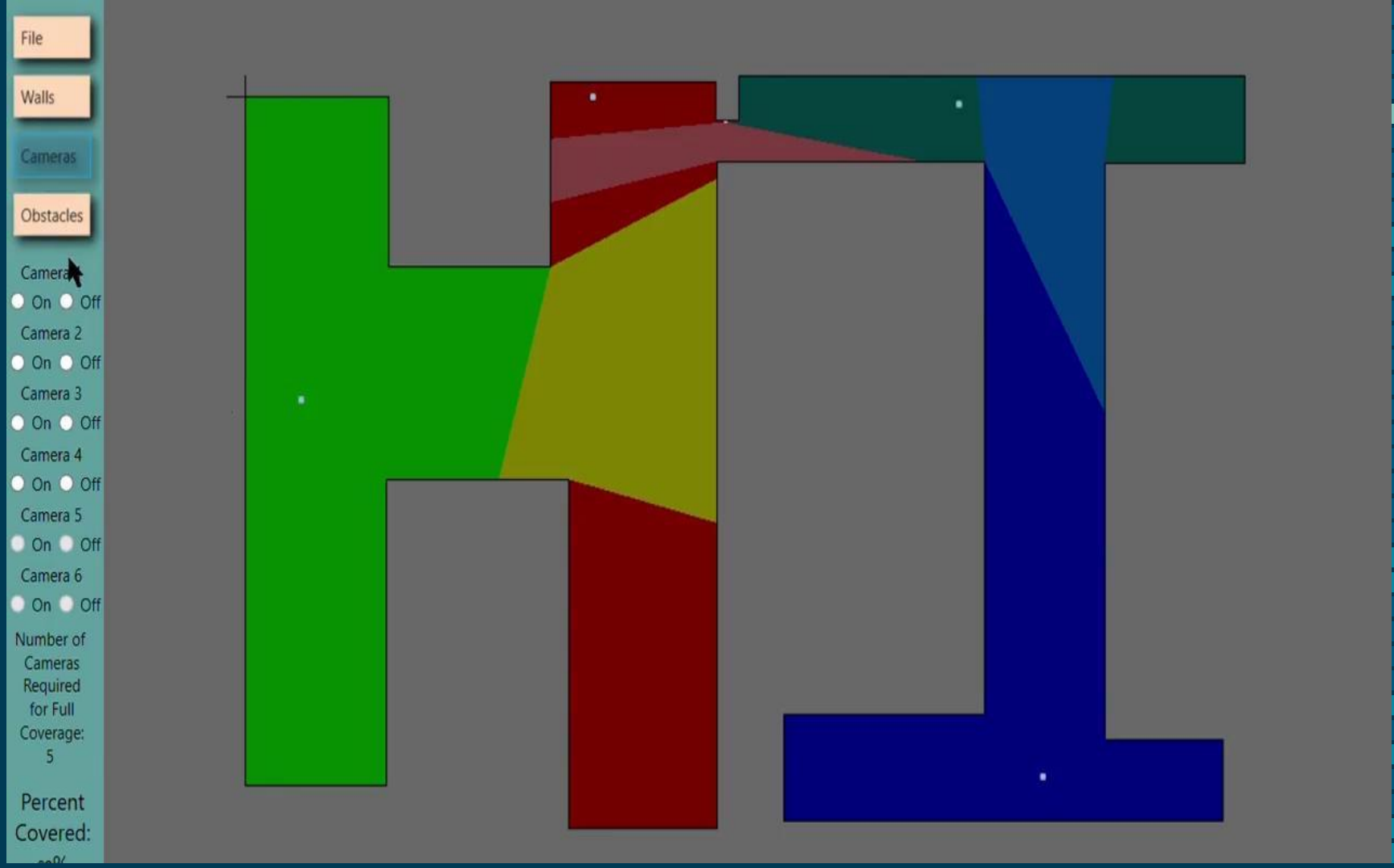
Number of Cameras Required for Full Coverage: 1

Percent Covered:




is for cookie - Notepad
File Edit Format View Help
Wall:147:75:687:75
Wall:687:75:687:168
Wall:687:168:305:168
Wall:305:168:305:343
Wall:305:343:692:343
Wall:692:343:692:480
Wall:692:480:147:480
Wall:147:480:147:69
Camera:0:493:90
Ln 1, Col 1




File
Walls
Cameras
Obstacles
Camera
On Off
Camera 2
On Off
Camera 3
On Off
Camera 4
On Off
Camera 5
On Off
Camera 6
On Off
Number of Cameras Required for Full Coverage: 5
Percent Covered:

CREDITS

- Microsoft Documentation
  - Understanding built-in C# classes and hierarchy
- Stack Overflow
  - Little, technical problems, that someone else had before me

CREDITS

- C# Helper
  - Understanding the power of the language, created the bulk of BitmapPixelMaker
- RedblobGames
  - Documented and implemented supercover_line
- Dr. Pankratz and Dr. McVey

CREDITS

Special thanks to all the people who made and released these awesome resources for free:

- Presentation template by SlidesCarnival
- Photographs by Unsplash

THANKS!

Any questions?

You can find me at:

mark.nichols@snc.edu