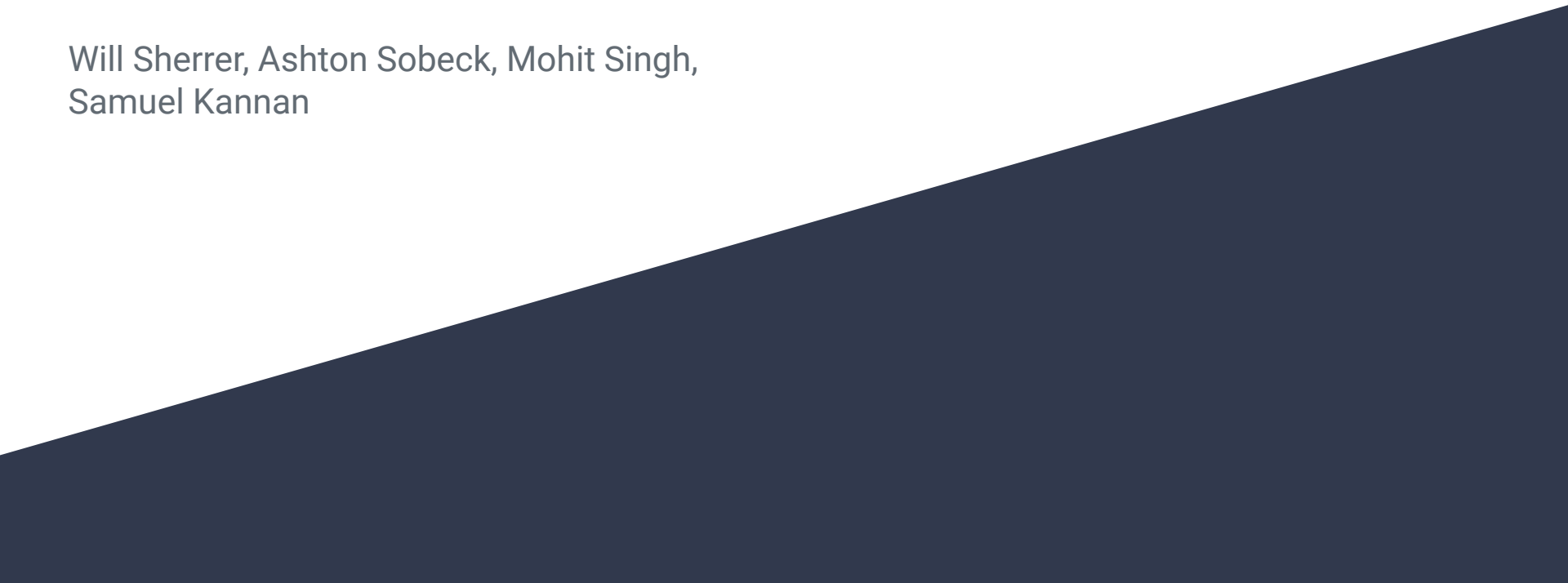


# Haunted Ghost Snake

Will Sherrer, Ashton Sobeck, Mohit Singh,  
Samuel Kannan

A dark blue diagonal gradient bar that starts from the bottom left and extends towards the top right, covering the lower half of the slide.

# Inspiration

- We wanted to make a game that would test our knowledge of design patterns while also being fun to play.
- Snake is a classic game, and developing it presented many challenges and considerations along the way.
- We also were able to put our own Halloween spin on it with different backgrounds, snake colors and the pumpkin that the snake eats.

# Technologies Used

- Python 3.10
  - Allows us to declare types on arguments to keep track of data
- Pygame
  - A python library that allows for easy game development and GUI manipulation
- NumPy
  - We used this for converting image data to an array in order to change the color of our snake

# Design Patterns Used

- Singleton

- Our game used this pattern. We only need one instance of our game being used at a time.
- Makes use of the `__new__` function within python instead of `__init__`.
  - Allows for checking of how many instantiations of the class there has been within the program.
  - This allows us to make only one “instance” of the game even if there are multiple variables of the Game type.

```
1 class Game(object):
2     __instance = None
3     width = 960
4     height = 640
5     block_size = 32
6     blockers = 10
7
8     # this makes our singleton
9     def __new__(cls, width=960, height=640, block_size=32, blockers=10):
10         if Game.__instance is None:
11             print('Game Initializing...')
12             Game.__instance = object.__new__(cls)
13             Game.__instance.width = width
14             Game.__instance.height = height
15             Game.__instance.block_size = block_size
16             Game.__instance.blockers = blockers
17         return Game.__instance
```

# Design Patterns Used

- Clone
  - We used the clone pattern to quickly create similar objects of the same type.
  - Mainly used on the pumpkins that the snake eats and the blockers on the screen.
  - Shallow copy most information, but update position on screen once the object was cloned.
  - This pattern allowed us to have more concise and clean development instead of calling constructors many times during methods.



```
1  def clone(self, x=-1, y=-1):
2      # make a clone
3      if x != -1 and y != -1:
4          return Blocker(self.window, self.snake_body, self.width,
5                          self.height, self.block_size, (x, y))
6      block = Blocker(self.window, self.snake_body, self.width, self.height, self.block_size)
7      return block
```

Demo