Haunted Ghost Snake

Will Sherrer, Ashton Sobeck, Mohit Singh,
Samuel Kannan

Inspiration

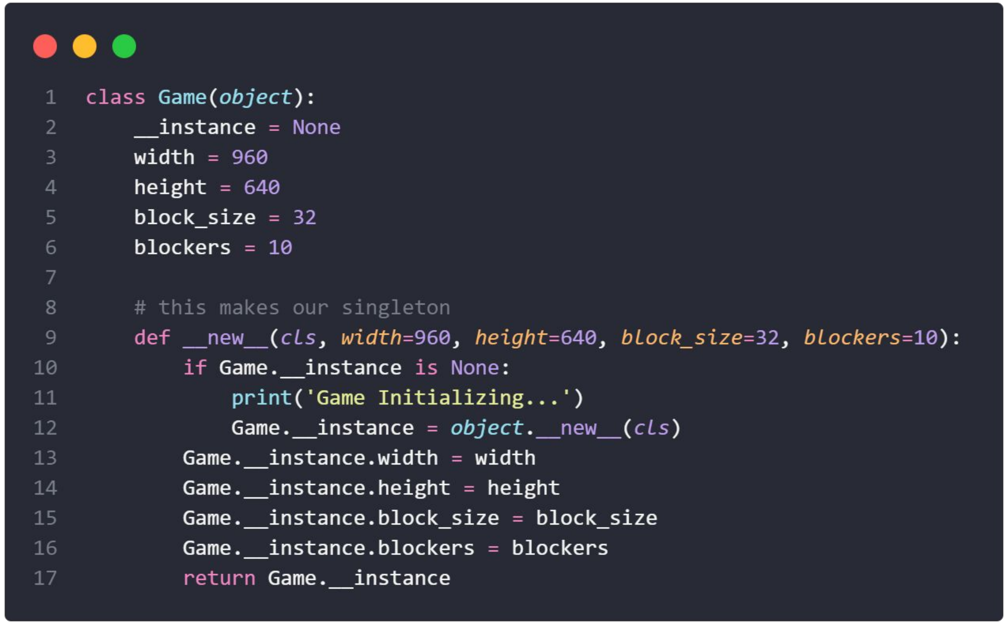
- We wanted to make a game that would test our knowledge of design patterns while also being fun to play.
- Snake is a classic game, and developing it presented many challenges and considerations along the way.
- We also were able to put our own Halloween spin on it with different backgrounds, snake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

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.

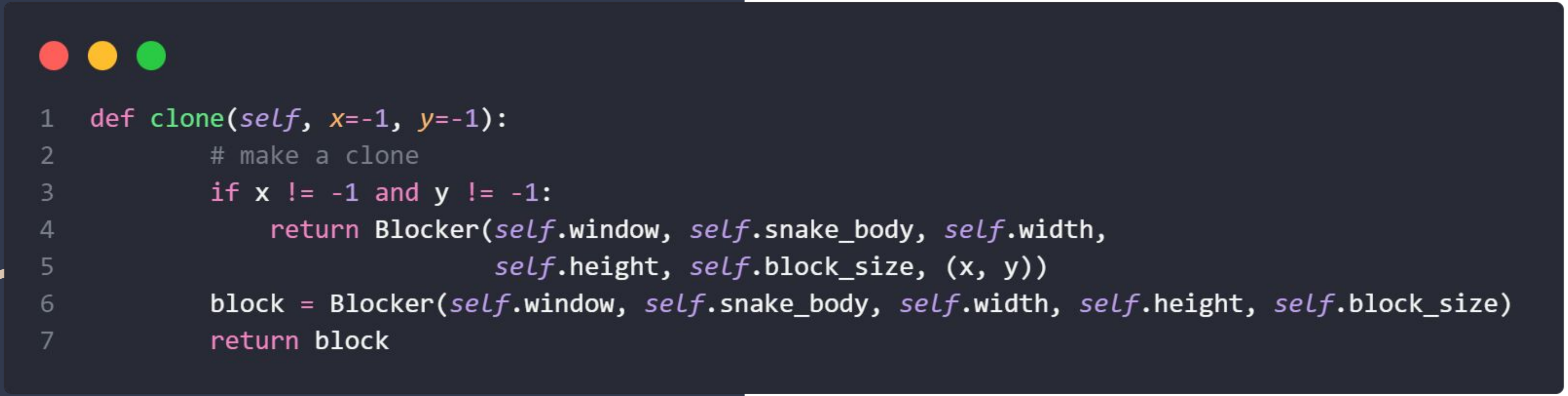


```python
class Game(object):
    __instance = None
    width = 960
    height = 640
    block_size = 32
    blockers = 10

    # this makes our singleton
    def __new__(cls, width=960, height=640, block_size=32, blockers=10):
        if Game.__instance is None:
            print('Game Initializing...')
            Game.__instance = object.__new__(cls)
        Game.__instance.width = width
        Game.__instance.height = height
        Game.__instance.block_size = block_size
        Game.__instance.blockers = blockers
        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.



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

Demo