

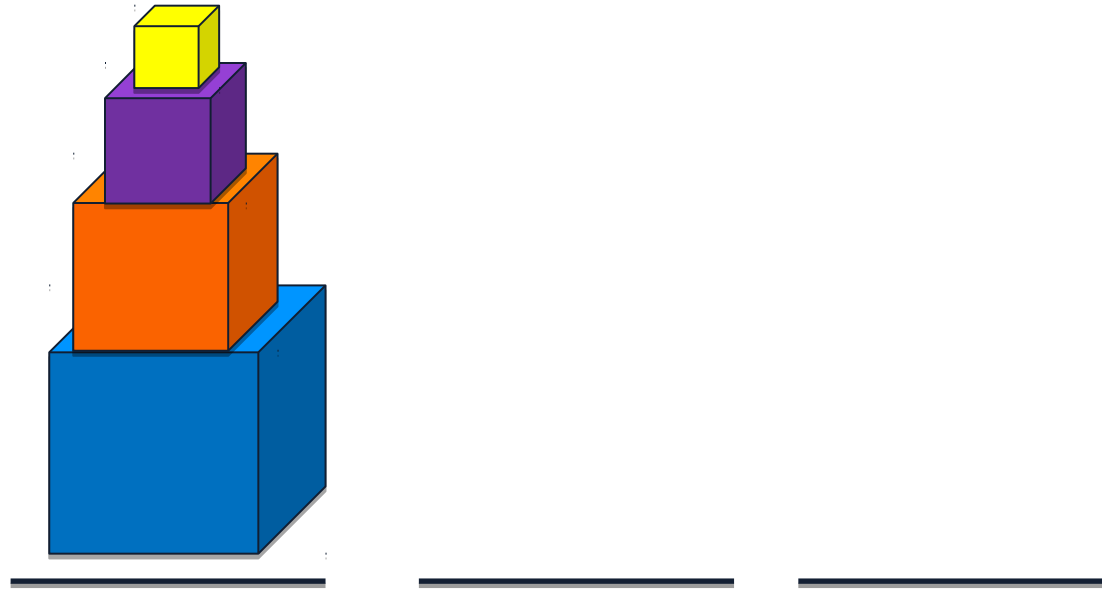
Tower of Hanoi

Prof. Wade Fagen-Ulmschneider

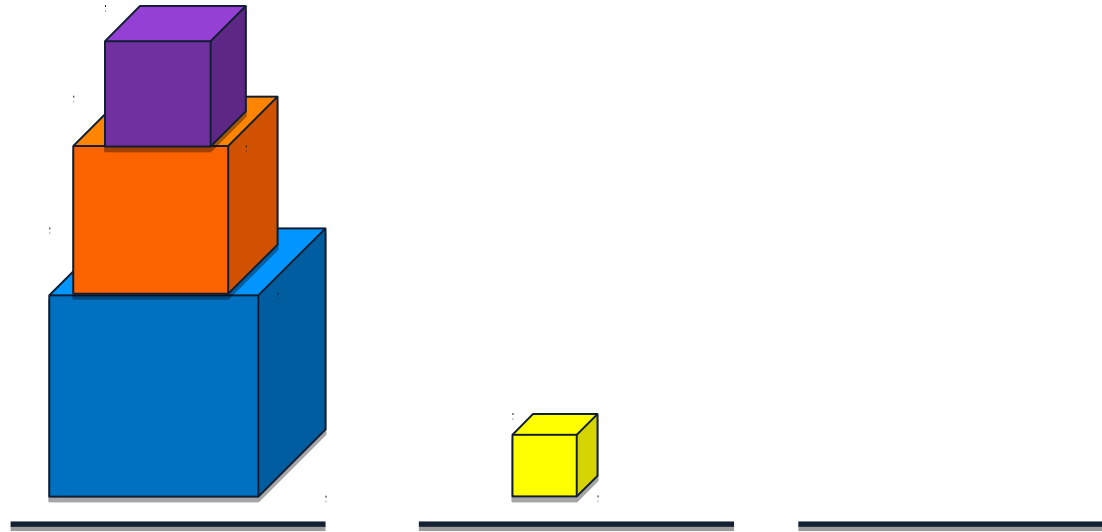
I ILLINOIS



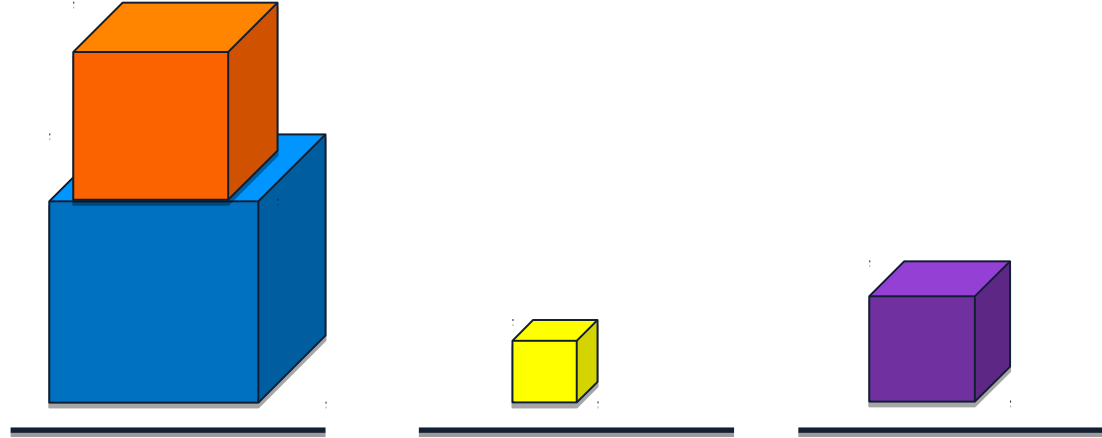
Consider the Tower of Hanoi problem, where multiple cubes must be transferred to a new location in such a way that a larger cube cannot be placed on top of a smaller cube:



Consider the Tower of Hanoi problem, where multiple cubes must be transferred to a new location in such a way that a larger cube cannot be placed on top of a smaller cube:



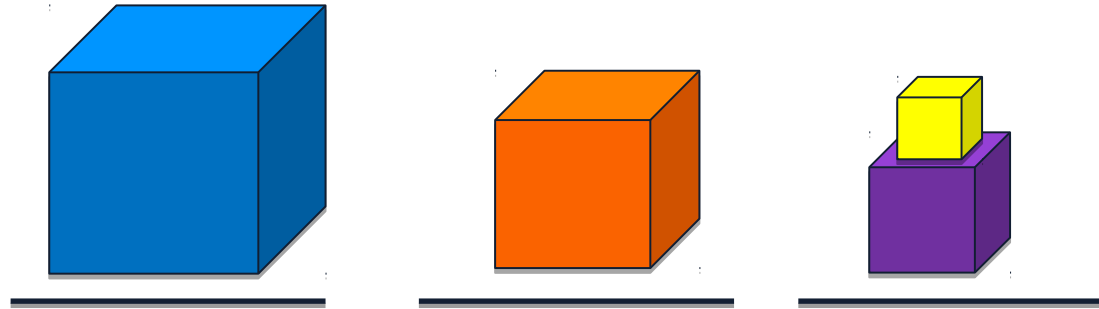
Consider the Tower of Hanoi problem, where multiple cubes must be transferred to a new location in such a way that a larger cube cannot be placed on top of a smaller cube:



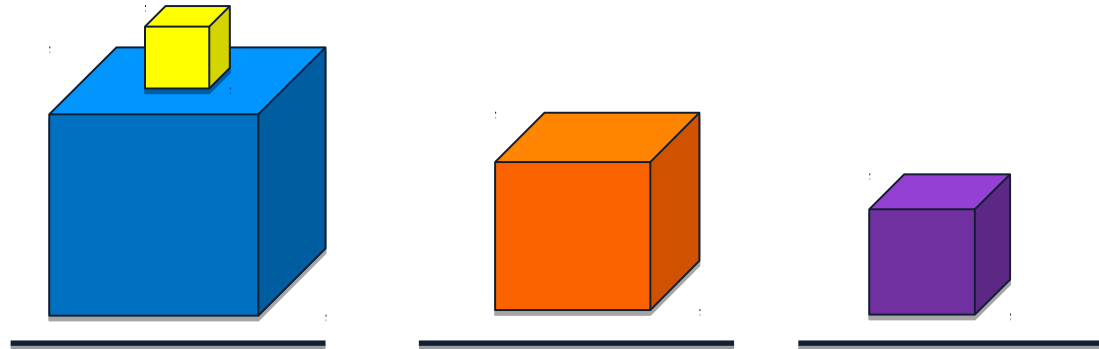
Consider the Tower of Hanoi problem, where multiple cubes must be transferred to a new location in such a way that a larger cube cannot be placed on top of a smaller cube:



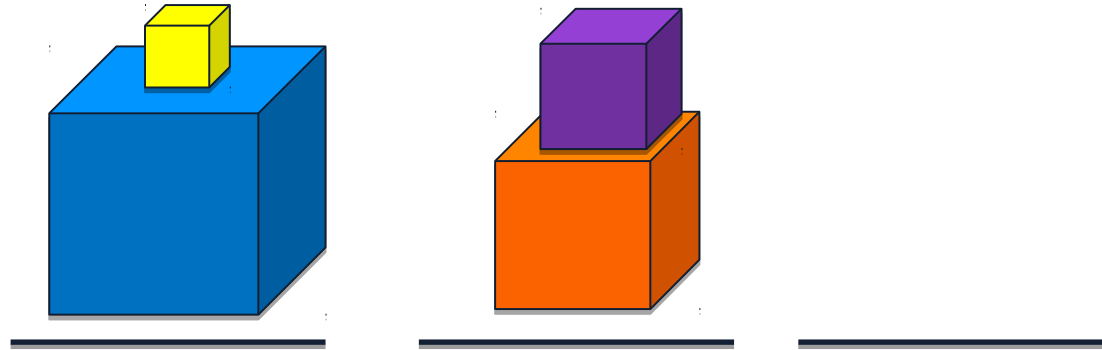
Consider the Tower of Hanoi problem, where multiple cubes must be transferred to a new location in such a way that a larger cube cannot be placed on top of a smaller cube:



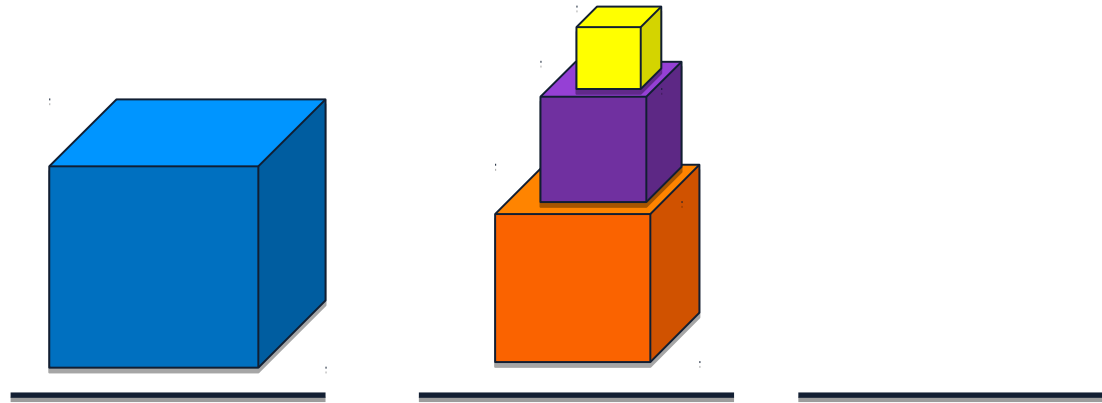
Consider the Tower of Hanoi problem, where multiple cubes must be transferred to a new location in such a way that a larger cube cannot be placed on top of a smaller cube:



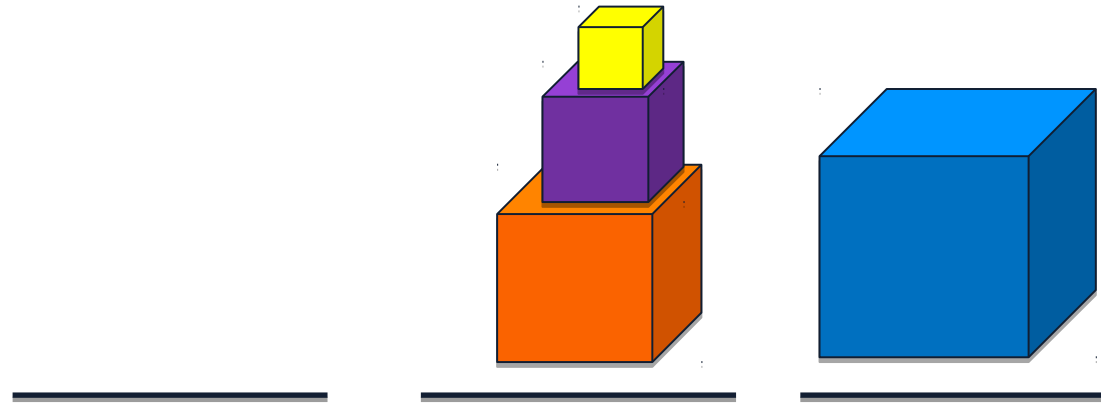
Consider the Tower of Hanoi problem, where multiple cubes must be transferred to a new location in such a way that a larger cube cannot be placed on top of a smaller cube:



Consider the Tower of Hanoi problem, where multiple cubes must be transferred to a new location in such a way that a larger cube cannot be placed on top of a smaller cube:

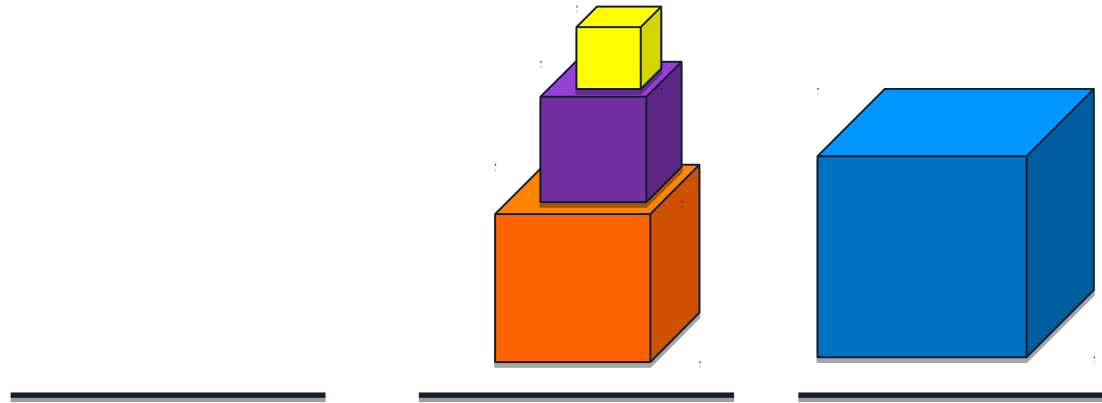


Consider the Tower of Hanoi problem, where multiple cubes must be transferred to a new location in such a way that a larger cube cannot be placed on top of a smaller cube:



Programmatic Structure

The Tower of Hanoi problem involves three distinct entities that must be represented in code:



Extending the Cube

The cube class needs a color:

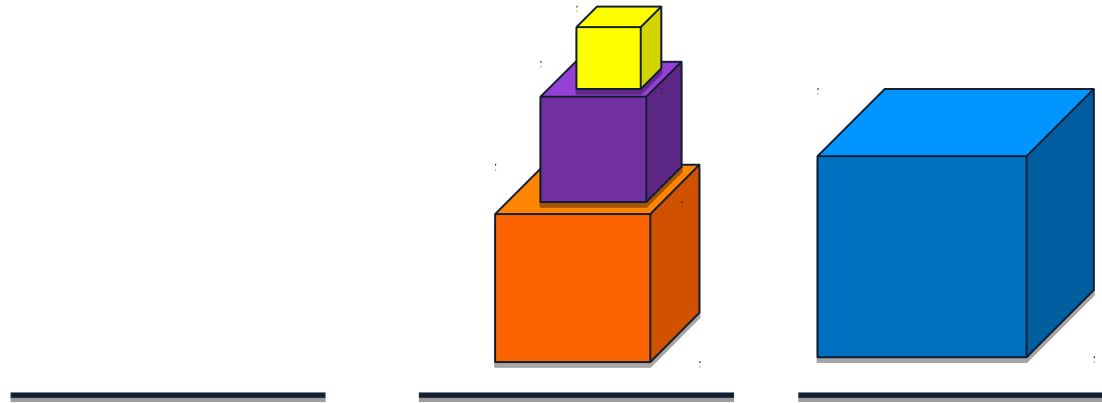
- In last week's assignment, we built `uiuc::HSLAPixel`.

cpp-tower/uiuc/Cube.h

```
12 namespace uiuc {
13     class Cube {
14     public:
15         Cube(double length, HSLAPixel color);
16
17         double getLength() const;
18         void setLength(double length);
19
20         double getVolume() const;
21         double getSurfaceArea() const;
22
23     private:
24         double length_;
25         HSLAPixel color_;
26     };
27 }
```

Building the Tower of Hanoi Game

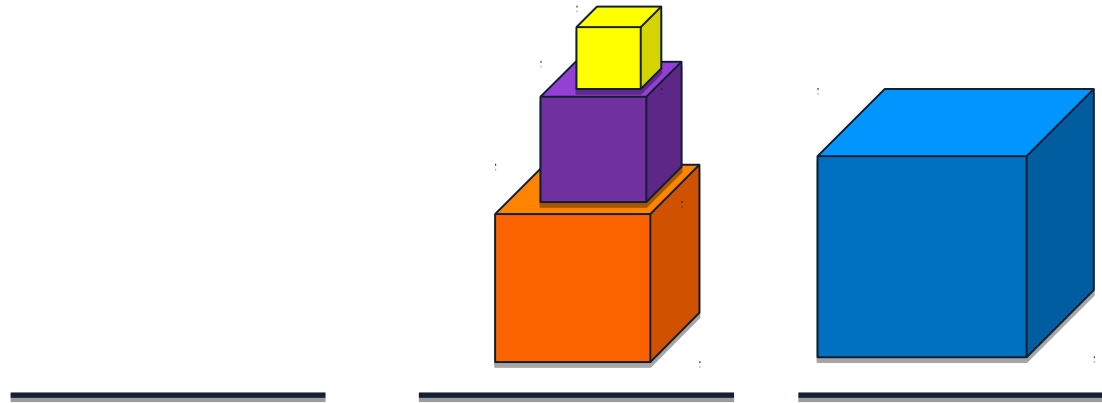
A new class must be created to represent each of the stacks in the Tower of Hanoi game:



Building the Stack Class

A single stack must contain:

- An vector of cubes
- Operations to interact with the top of the stack



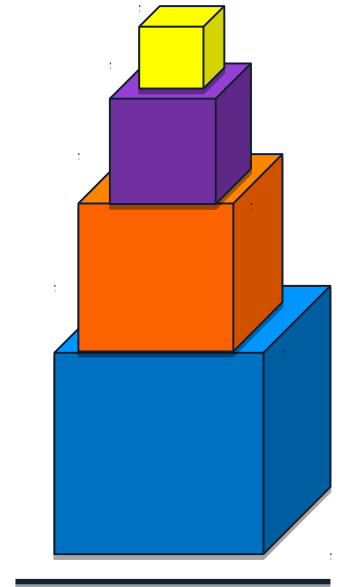
cpp-tower/Stack.h

```
14 class Stack {
15     public:
16         void push_back(const Cube & cube);
17         Cube  removeTop();
18         Cube & peekTop();
19         unsigned size() const;
20
21         // An overloaded operator<<, allowing us to print the stack
22         // via `cout<<`:
23         friend std::ostream& operator<<(std::ostream & os,
24                                         const Stack & stack);
25
26     private:
27         std::vector<Cube> cubes_;
28 };
29
```


Building the Tower of Hanoi Game

Finally, the game is built from the components we have already built:

- An array of three stacks
- The initial state has four cubes in the first stack

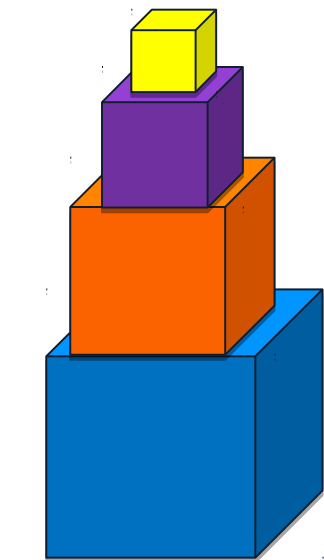


cpp-tower/Game.h

```
8  #pragma once
9
10 #include "Stack.h"
11 #include <vector>
12
13 class Game {
14     public:
15         Game();
16         void solve();
17
18         friend std::ostream& operator<<(std::ostream & os,
19                                         const Game & game);
20
21     private:
22         std::vector<Stack> stacks_;
23 };
```

cpp-tower/Game.cpp

```
27 Game::Game() {
28     // Create the three empty stacks:
29     for (int i = 0; i < 3; i++) {
30         Stack stackOfCubes;
31         stacks_.push_back( stackOfCubes );
32     }
33
34     // Create the four cubes, placing each on the [0]th stack:
35     Cube blue(4, uiuc::HSLAPixel::BLUE);
36     stacks_[0].push_back(blue);
37
38     Cube orange(3, uiuc::HSLAPixel::ORANGE);
39     stacks_[0].push_back(orange);
40
41     Cube purple(2, uiuc::HSLAPixel::PURPLE);
42     stacks_[0].push_back(purple);
43
44     Cube yellow(1, uiuc::HSLAPixel::YELLOW);
45     stacks_[0].push_back(yellow);
46 }
```



cpp-tower/main.cpp

```
8 #include "Game.h"
9 #include <iostream>
10
11 int main() {
12     Game g;
13
14     std::cout << "Initial game state: " << std::endl;
15     std::cout << g << std::endl;
16
17     g.solve();
18
19
20     std::cout << "Final game state: " << std::endl;
21     std::cout << g << std::endl;
22
23     return 0;
24 }
```