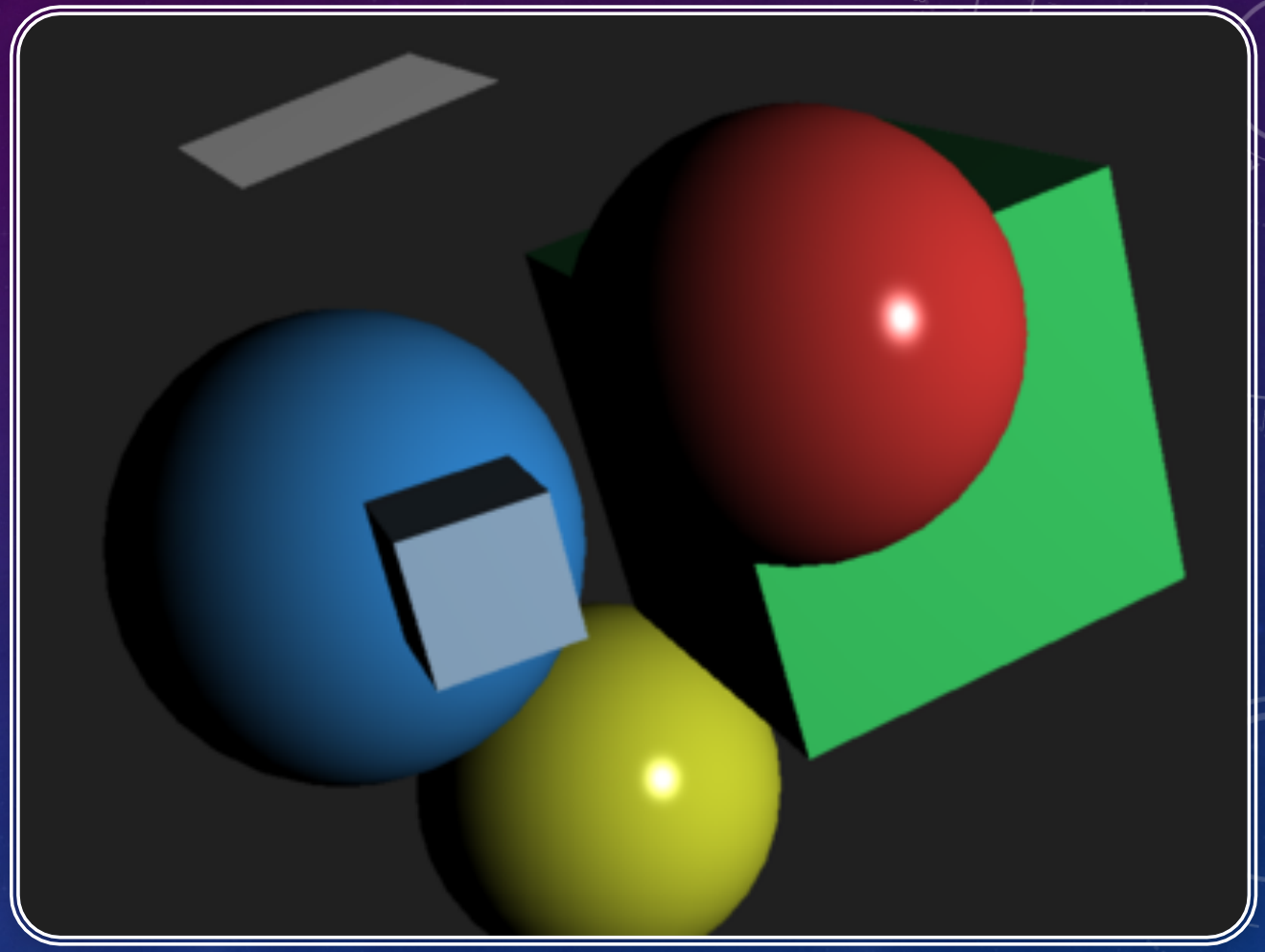
Software Workshop No. 4

UCSB Robotics, Fall 2020 | Alex Mei

LAST TIME...

- C++ Compilation
- Makefiles
- Structs
- Classes



TODAY'S AGENDA

- Stacks
- Queues
- Command Line
- Github

STACKS

- Definition: an object where new elements are placed on the top and only the top element can be removed
- "Last In First Out"
- Syntax: stack<type> varName;
- Note: must include the <stack> library

STACK MEMBER FUNCTIONS

- push(x): adds x to the top of the stack
- top( ): returns the element on the top of the stack
- pop( ): removes the element on the top of the stack
- size( ): returns the number of elements on the stack
- empty( ): returns whether the stack is empty

```cpp
#include <stack>
#include <iostream>
using namespace std;

int main(){
    stack<int> s;
    cout << "Empty? " << s.empty() << endl;
    cout << "Size: " << s.size() << endl;
    for(int i = 0; i < 5; ++i){
        s.push(i);
    }
    cout << "Empty? " << s.empty() << endl;
    cout << "Size: " << s.size() << endl;
    for(int i = 0; i < 5; ++i){
        cout << "Top of Stack: " << s.top() << endl;
        s.pop();
    }
    cout << "Empty? " << s.empty() << endl;
    cout << "Size: " << s.size() << endl;

    return 0;
}
```

```
Empty? 1
Size: 0
Empty? 0
Size: 5
Top of Stack: 4
Top of Stack: 3
Top of Stack: 2
Top of Stack: 1
Top of Stack: 0
Empty? 1
Size: 0
```

YOUR TURN...

- Write a function that takes a string as an arithmetic expression and returns whether that string has a balanced number of parenthesis.

```
cout << balanced("((1 + 2) * (3 + 4))") << endl; //true
cout << balanced("(((sleepy))") << endl; //false
cout << balanced("((snorlax)))") << endl; //false
```

QUEUES

- Definition: an object where the first element to enter a queue will be the first element to leave the queue
- "First In First Out"
- Syntax: queue<type> varName;
- Note: must include the <queue> library

QUEUE MEMBER FUNCTIONS

- push(x): adds x to the back of the queue
- front( ): returns the element at the front of the queue
- pop( ): removes the element at the front of the queue
- size( ): returns the number of elements on the stack
- empty( ): returns whether the stack is empty

```
queue<char> q;
string s = "Orange Chicken";
cout << "Empty? " << q.empty() << endl;
cout << "Size: " << q.size() << endl;
for(int i = 0; i < s.size(); ++i){
    q.push(s[i]);
}
cout << "Empty? " << q.empty() << endl;
cout << "Size: " << q.size() << endl;
for(int i = 0; i < s.size(); ++i){
    cout << q.front();
    q.pop();
}
cout << endl << "Empty? " << q.empty() << endl;
cout << "Size: " << q.size() << endl;
```

```
Empty? 1
Size: 0
Empty? 0
Size: 14
Orange Chicken
Empty? 1
Size: 0
```

STD::VECTORS

- Definition: an array with an unfixed size
- Syntax: vector<type> varName;
- Note: must include the <vector> library

VECTOR MEMBER FUNCTIONS

- push_back(x): adds x to the back of the vector
- front( ) / back( ): returns the element at the front / back of the vector
- Can use [ ] notation to access elements in vector like array
- pop_back( ): removes the element at the back of the vector
- erase(vectorName.begin( ) + index): removes the index element of the vector
- size( ): returns the number of elements in the vector

YOUR TURN...

- Implement an integer Queue class that supports push( ), front( ), pop( ), size( ), and empty( ).
- But first, a demo on how to implement an integer Stack.

COMMAND LINE COMMANDS

- ls: displays list of files in current directory
- cd <path>: changes directory to specified path
- mkdir <path>: creates a new directory at specified path
- rm <path>: removes file at specified path (add the -r flag to remove all files in a folder)
- cat <path>: displays contents of file at specified path

GITHUB

https://github.com/

- Version Control: can easily keep track of changes over time and revert changes as necessary
- Merge Conflicts: sharing code may result in code conflicts; Github makes it efficient to resolve conflicts

GIT

https://git-scm.com/downloads

- Github + Command Line
- **Workflow**: pull changes from Github => make changes locally => push changes to Github

GIT COMMANDS (SET UP)

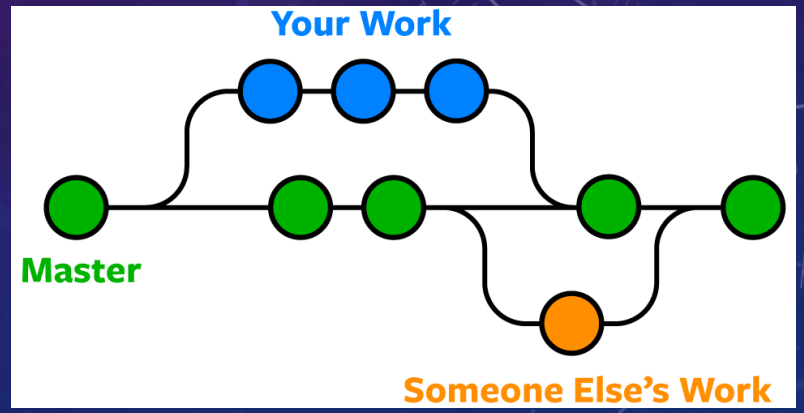
- git init: initializes current directory as a git repository
- git remote add origin <HTTPS>: adds a remote named origin to pull and push your local code to and from
- git pull origin <branchname>: pulls changes from the remote to your local repository
- git clone <HTTPS>: combines the 3 steps above in one command

GIT COMMANDS (COMMIT)

- git status: displays the current status of changed files (unstaged, staged, committed)
- git add <filename>: stages the changes made to the specified file for a commit (use the dot to signify all files)
- git commit -m "message": commits the staged changes with a specified message
- git push origin <branchname>: pushes local changes to the specified branch name of the remote (default: master)

GIT COMMANDS (BRANCHES)

- git fetch <branchname>: pulls a specified branch to your local repository
- git branch: displays all branches on your local repository + active branch
- git switch <branchname>: switches active branch to specified branch
- git checkout <branchname>: creates a new branch locally (add -b flag to switch to the new branch as well)



YOUR TURN...

1. Set up a Github account.
2. Clone the following repository: https://github.com/alexmeigz/RoboticsWorkshopDemo.git
3. Create a new branch named after you.
4. Implement the stack and queue classes from the starter code in this new branch.
5. Create a new repository named "Robotics Workshop Demo" and push your changes to this new repository.
6. Create a pull request on Github and merge those changes into the master branch.

YOUR TURN... (CHOOSE 1)

- Implement a stack class for strings using two queues.
  - The class should have the push( ), top( ), pop( ), size( ), and empty( ) member functions.
- Implement a queue class for strings using two stacks.
  - The class should have the push( ), front( ), pop( ), size( ), and empty( ) member functions.