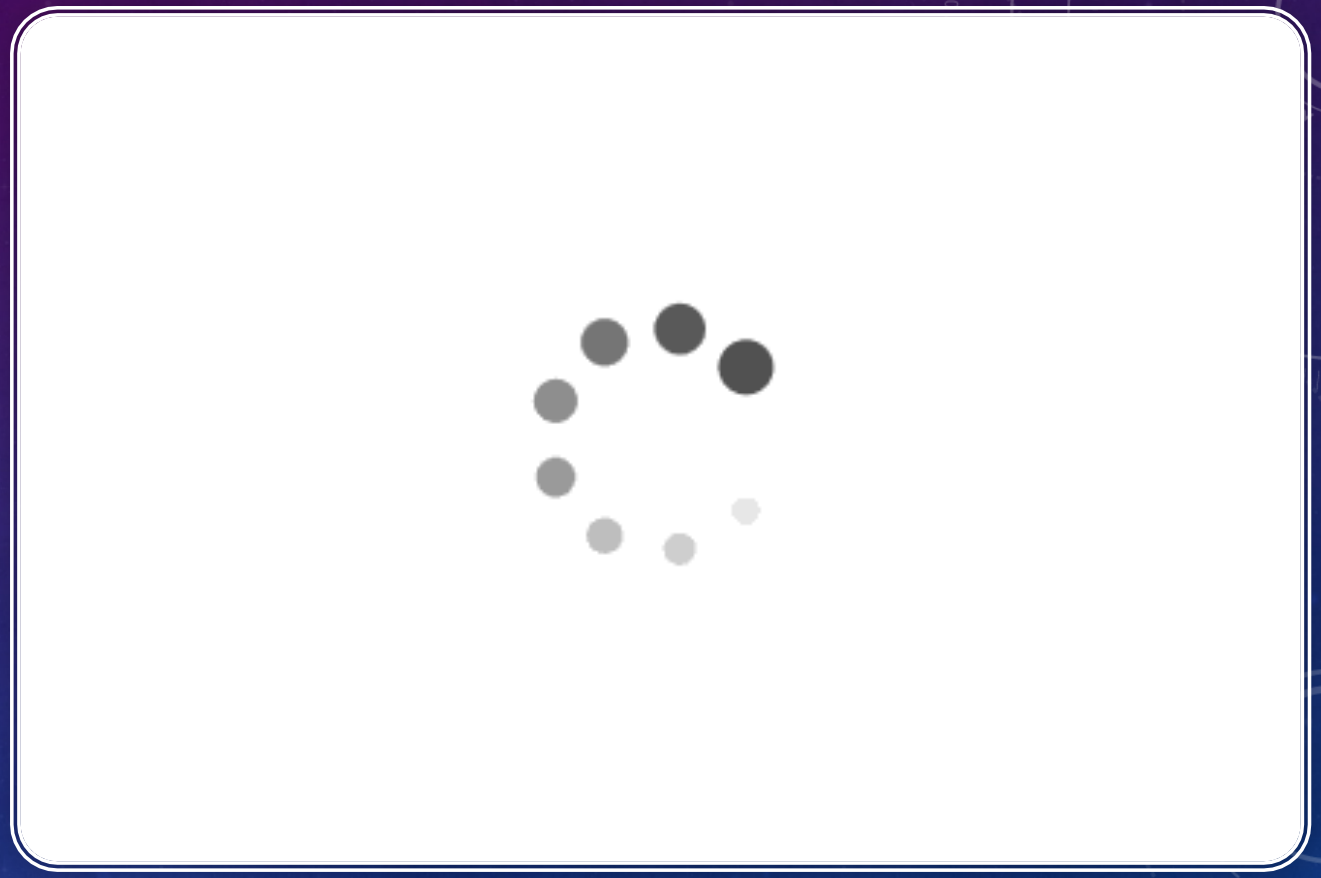
Software Workshop No. 3

UCSB Robotics, Fall 2020 | Alex Mei

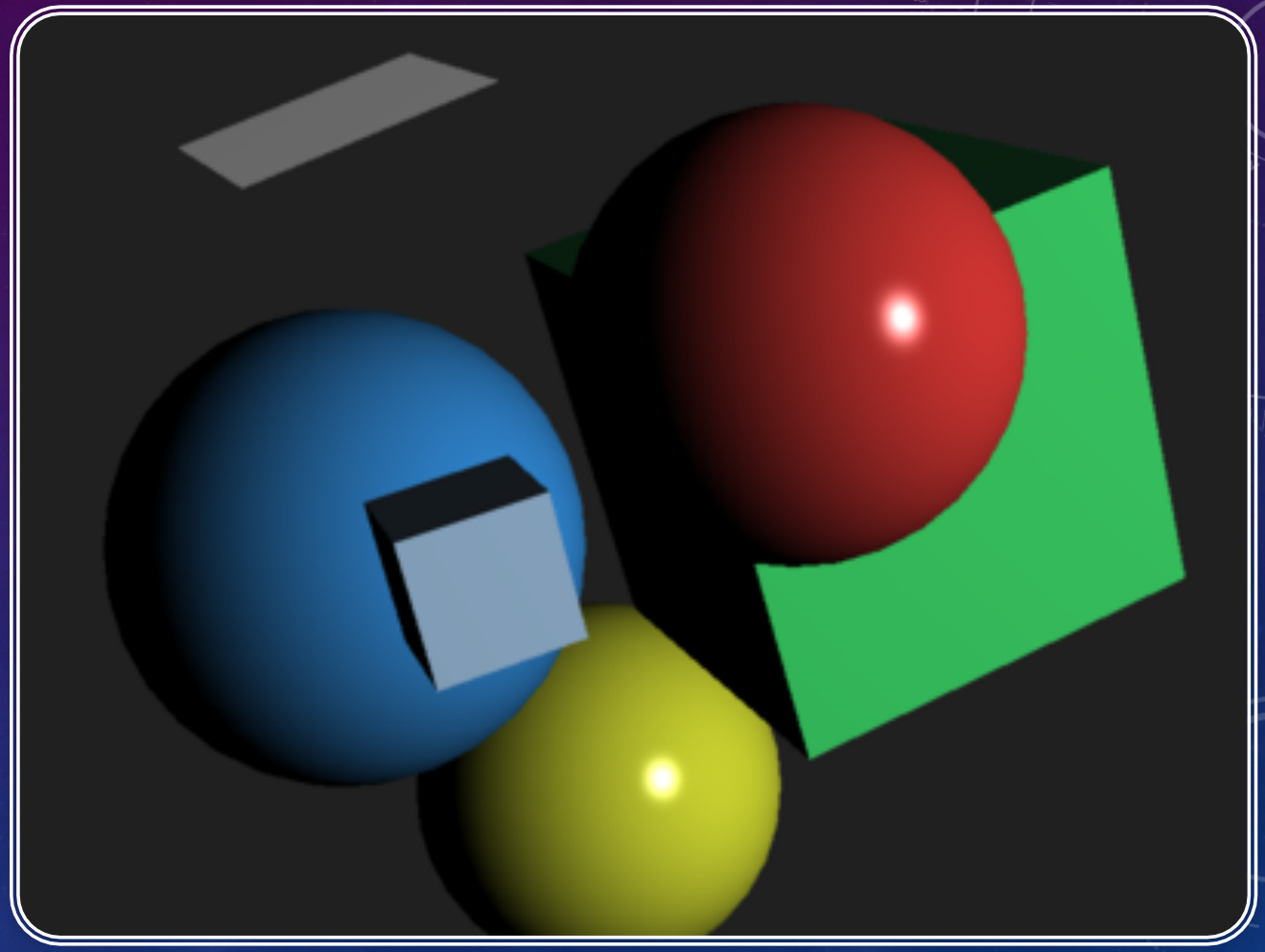
LAST TIME...

- Boolean Operators
- Conditional Statements
- Arrays and Strings
- Loops



TODAY'S AGENDA

- C++ Compilation
- Makefiles
- Structs
- Classes



COMPILATION

- Syntax: {compiler} {filenames} -o {executable name}
- Compiler: program that translates 'high-level code' to 'machine code'
- -o: flag to write the output into an executable file
- Executable: a file that can be run by the computer

COMPILATION

first.cpp

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

//function implementation
void sayHi(){
    cout << "Hello World" << endl;
}
```

second.cpp

```cpp
//function declaration
void sayHi();

int main(){
    sayHi();
    sayHi();
    return 0;
}
```

```
(base) Alexs-MacBook-Pro:Robotics alex$ g++ first.cpp second.cpp -o main
(base) Alexs-MacBook-Pro:Robotics alex$ ./main
Hello World
Hello World
```

MAKEFILES

- Purpose: reduce redundancy in compilation

Makefile

```
1   # General Format:
2   # {target} : {dependencies}
3   # [TAB]     terminal commands
4
5   main : first.cpp second.cpp
6       g++ first.cpp second.cpp -o main
7
8   clean :
9       rm -rf main
```

```
5   main : first.cpp second.cpp
6       g++ $^ -o $@
7       ./main
8       make clean
9
10  clean :
11      rm -rf main
```

YOUR TURN...

- Write a function that takes an array of strings and its length and prints each element on a newline in a file called "printArray.cpp".
- Then, in a file called "main.cpp", write initialize a string array with elements "Rewrite", "The", and "Stars".
- Use this array to call the printArray( ) function. (Remember, the function must be declared as well.)
- Write a Makefile to compile, execute, and clean in a single command.

STRUCTS & CLASSES

- Definition: a complex data type that has several members containing various data types and member functions
- private: only members can access a class's private members
- public: public members can be accessed outside of the class
- Structs: members default to public
- Classes: member default to private

STRUCTS

- Initialization: members variables of a struct type can be initialized similarly to arrays
- Dot Operator (.): members variables of a struct/class type can be accessed using the dot operator

```
struct Point{
    double x;
    double y;
};
```

```
int main(){
    Point one = {5, 10};
    cout << "X-Coordinate: " << one.x << endl;
    cout << "Y-Coordinate: " << one.y << endl;

    return 0;
}
```

```
[Running] cd "/Users/alex/Desktop/Robotics/"
X-Coordinate: 5
Y-Coordinate: 10
```

YOUR TURN...

- Declare a struct to represent a fractional number with a numerator and a denominator.
- Write a function that takes two fraction types and returns the product of the two fractions.
- Using the definition of the Point struct, write a function that takes two points and returns the distance between the two points. (Tip: utilize the sqrt( ) function from the <math> library.)

PROGRAMMING PRINCIPLES

- comments (// or /* */): used to explain code at a high level to others who are reading your code
- .h and .cpp files: separation between the declaration and the implementation of a class
- Pre-Condition: the specification of the input parameters
- Post-Condition: the format of the of the output parameters

CLASSES

- Constructor: a special function that is called when an object is created from this class
- Mutator Functions: functions that change member variables
- Accessor Functions: functions that returns member variables
- const: functions with const cannot modify member variables

motor.h

motor.h > Motor

```
1   class Motor{
2       public:
3           //constructor
4           Motor(double x = 0, double y = 0);
5
6           //accessors
7           double getVelocityX() const;
8           double getVelocityY() const;
9
10          //mutators
11          void setVelocityX(double x);
12          void setVelocityY(double y);
13
14          //functions
15          double calculateVelocity() const;
16          double calculateDirection() const;
17
18      private:
19          //member variables
20          double velocityX;
21          double velocityY;
22  };
```

CLASSES

- Scope Resolution Operator (::): identifies the scope of a function or variables
- Note: the implementation file should only include the definitions of the class's member functions

motor.cpp

motor.cpp > Motor::getVelocityY() const

```cpp
#include "motor.h"

Motor::Motor(double x, double y){
    velocityX = x;
    velocityY = y;
}

double Motor::getVelocityX() const{
    return velocityX;
}

double Motor::getVelocityY() const{
    return velocityY;
}

void Motor::setVelocityX(double x){
    velocityX = x;
}

void Motor::setVelocityY(double y){
    velocityY = y;
}
```

YOUR TURN...

Implement the calculateVelocity( ) and calculateDirection( ) functions for the Motor class.

Recall that | v | = sqrt(x * x + y * y) and direction = arctan(y / x).

Both the sqrt( ) and atan( ) functions are implemented in the <cmath> library.

USING CLASSES

```cpp
#include <iostream>
#include "motor.h"
using namespace std;

int main(){
    Motor m1;
    Motor m2(3, 4);
    cout << "M1 speed: " << m1.calculateVelocity() << endl;
    cout << "M2 speed: " << m2.calculateVelocity() << endl;
    return 0;
}
```

```
(base) Alexs-MacBook-Pro:Robotics alex$ g++ motor.cpp week3.cpp -o week3
(base) Alexs-MacBook-Pro:Robotics alex$ ./week3
M1 speed: 0
M2 speed: 5
```

YOUR TURN...

- Convert the Point struct into a Point class with 3 member variables x, y, z.
- Include accessor and mutator functions for each member variable.
- The constructor should default the point's location to the origin if no values are given.
- Write a calculateDistance( ) function which returns the point's distance from the origin.