

EECS 395

Day 2

Nathan Nichols

Today

- Questionnaire results
- Review
- OOP in thirty seconds
- Terminology
- Demo demo demo

Questionnaire

- Best known language is C++, then Java
- Most people get OOP
- About half get manual memory management
- Favorite book/favorite band results were pretty disappointing
 - For Whom the Bell Tolls by Ernest Hemingway
 - Joanna Newsom, or anything with Spencer Krug
 - Wolf Parade
 - Sunset Rubdown
 - Swan Lake

Questionnaire

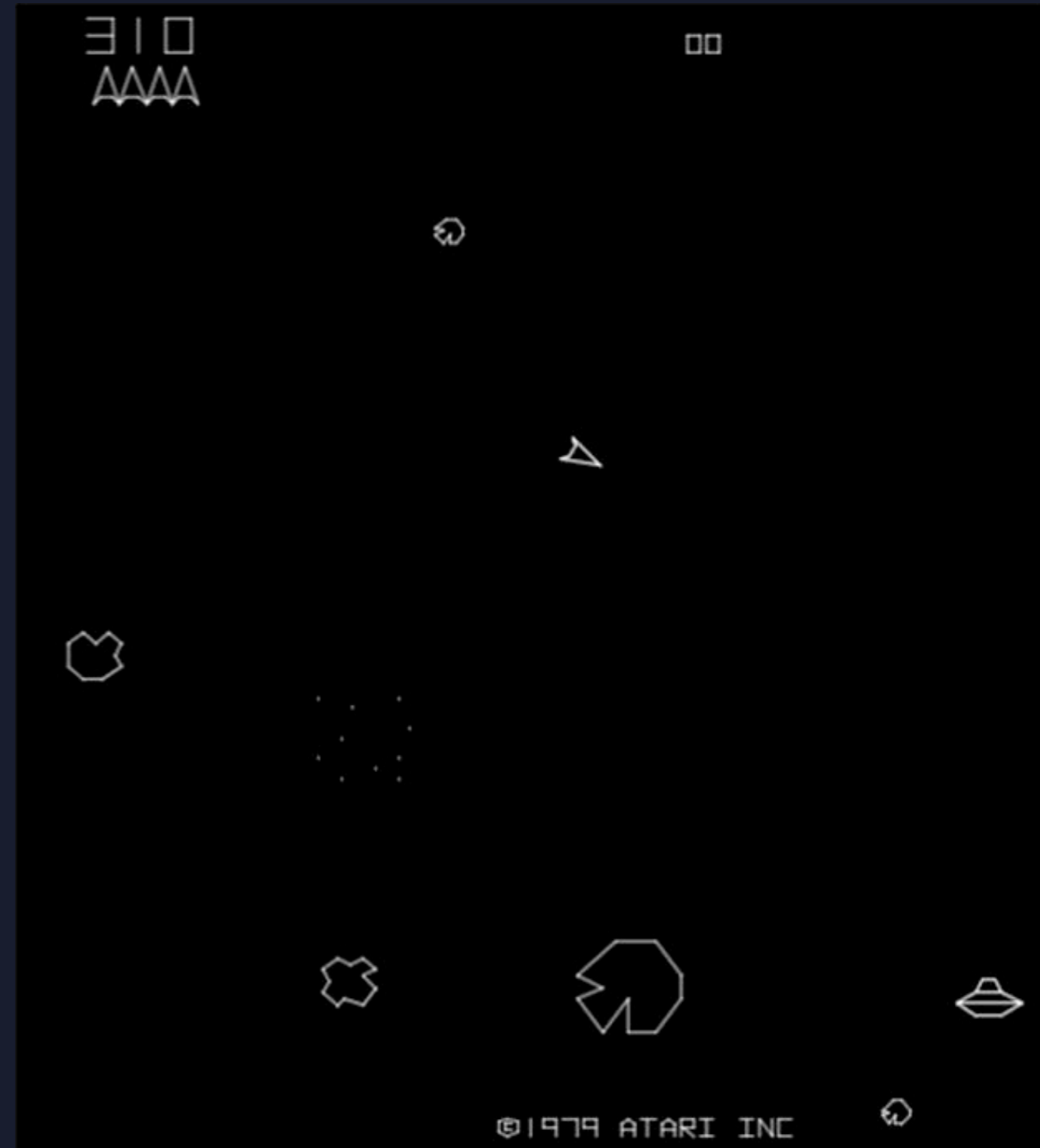
- Closest answers were:
 - Tim Zweibel, with Flatland
 - Andy Mounce, with New Pornographers
- Spirit award:
 - Nik Sethi, with Blonde Redhead

Review

- Talked about the class setup
- History of Objective-C
- Started Objective-C

OOP

- For a lot (most) applications, it is useful to associate data and functions together
- The game “Asteroids” has three different classes: one class for the player’s **ship**, one for the **asteroids**, and one for the **bullets**
- Each individual bullet is an *instance* of the Bullet class



OOP

- Each instance has *private* member variables and *publicly* accessible methods
- The Bullet class may have member variables like position, velocity, timeToLive, etc.
- It would have methods like updatePosition, die, collideWithAsteroid, etc.
- The rest of the program doesn't know or care how Bullet implements its methods
- (We can change it without talking to the guy next door.)

OOP

- Movement of a Bullet is conceptually related to its position, its speed, whether or not it collides with an asteroid, etc.
- We can get clever!
- Bullets really move just like Asteroids
- Let's make a base class, Mover, that handles position, moving, and updating
- Bullet will inherit from Mover, so only needs Bullet-specific stuff.
- Same with Asteroids

Terminology

```
@interface Animal {  
    NSString *name;  
    NSString *sound;  
    int stomachCount;  
}
```

Member Variables

```
-(id)initWithName:(NSString *)aName sound:(NSString *)aSound \  
    stomachCount:(int)aStomachcount;  
@end
```

Method declaration

(a method that happens to be a constructor)

Terminology

```
-(void)makeCow {  
    Animal *cow = [Animal alloc];  
    [cow initWithName:@"Annabelle" sound:@"Moooooo" stomachCount: 4];  
}
```

cow is the *receiver*

The name of the *selector* here is
`initWithName:sound:stomachCount:`

We are passing the *message*

`initWithName:@"Annabelle" sound:@"Mo" stomachCount:4`
to cow.

Terminology

“Powerful” moment

```
//C++  
void foo() {  
    Animal *cow = new Animal("Annabelle", "Mooo", 4);  
    cow->WalkAround();  
}
```

Direct mapping

```
void Animal::WalkAround() {  
    // TODO: Implement WalkAround  
}
```

Terminology

```
//Objective-C
-(void)foo {
    Animal *cow = [Animal alloc];
    [cow initWithName:@"Annabelle" sound:@"Moo" stomachCount:4];
    [cow walkAround];
}
```

MAGIC!



```
@implementation Animal
-(void)walkAround {
    // TODO: Implement walkAround
}
@end
```

“Illusions, Michael!”

- You can do a lot of cool things with selectors and messages that you can't do with C++ method invocation

“Illusions, Michael!”

- Pass selectors around

```
-(void) setupWithAnimal:(Animal *)animal{
    //Not quite right
    UIButton *button = [[UIButton alloc] init];
    [button setTarget:animal];
    [button setAction:@selector(walkAround)];
    //Now when button is pressed, it will call
    //[animal walkAround];
}
```

Could be an NSString *

“Illusions, Michael!”

- Ask objects if they respond to selectors

```
-(void) moveAnimal:(Animal *)animal {
    if ([animal respondsToSelector:@selector(swimAround)]) {
        [animal swimAround];
    }
    else if ([animal respondsToSelector:@selector(walkAround)]) {
        [animal walkAround];
    }
    else {
        [animal stayInTheSamePosition];
    }
}
```

Demo demo demo