

Object-oriented History Garbage collection

- › Garbage collection
- › Object-oriented programming
 - › Louden Chapters 10

1

CS330 Spring 2003
Copyright George Tzanetakis, University of Victoria

Mark and sweep

- › Lazy: Allocator runs out of space
- › First pass: Follow all pointers recursively and mark everything reachable (extra bit)
- › Second pass: Move all unreferenced cells back to free list
- › Problem: processing delays

2

CS330 Spring 2003
Copyright George Tzanetakis, University of Victoria

Generational garbage collection

- › Spread cost more evenly
- › Allocated objects that survive long enough are simply copied to permanent space and never get reallocated
- › Only newer storage allocations need to be searched

3

CS330 Spring 2003
Copyright George Tzanetakis, University of Victoria

Object-oriented programming

- › Three major characteristics
 - › Data Hiding
 - › Inheritance
 - › Dynamic Binding

4

CS330 Spring 2003
Copyright George Tzanetakis, University of Victoria

History

- › Simula 1967
- › Simulation of the real world – objects
- › Software reuse
 - › Extension
 - › Restriction
 - › Redefinition
 - › Abstraction
 - › Polymorphization

5

CS330 Spring 2003
Copyright George Tzanetakis, University of Victoria

Data Hiding

- › Global variables can potentially be accessed and updated by every part of the program
- › 1970 David Parnas – Information Hiding
 - › module
- › ML module system, Ada packages, Modula-2 modules, C++ namespaces, Java packages are examples of information hiding

6

CS330 Spring 2003
Copyright George Tzanetakis, University of Victoria

Inheritance

- › Object class is a set of objects that share some operations
- › Objects are first-class values (user-defined types)
- › Inheritance = ability to organize object classes into an hierarchy of bases classes and derived classes (super and sub)

7

CS330 Spring 2003
Copyright George Tzanetakis, University of Victoria

Dynamic Binding

- › Way to distinguish between the general properties of a “base” class and the properties of a “specific kind of” derived class
- › The classic example
 - › Shape
 - › center, color, draw
 - › Circle, Triangle, Square

8

CS330 Spring 2003
Copyright George Tzanetakis, University of Victoria

The classic solution

```
enum Kind {circle, triangle, square};

class Shape{
    Kind k; // type field
    Point center;
    Color color;

public:
    void draw();
    void rotate(int);
}

void Shape::draw()
{
    switch(k) {
    case circle:
        // draw circle
        break;
    case triangle:
        // draw triangle
        break;
    ....
    }

    // PROBLEM: must change
    for every new shape
}
```

9

CS330 Spring 2003
Copyright George Tzanetakis, University of Victoria

The object-oriented solution(C++)

```
class Shape {
    Point center;
    Color col;

public:
    Point where() {return center;}
    virtual void draw() = 0;
    virtual void rotate(int angle) = 0;
}

void rotate_all(vector<Shape *>& v, int angle)
{ for (int i=0; i < v.size(); ++i)
  v[i]->rotate(angle);
}
```

Now the code works for
any kind of shape. The
appropriate rotate functions
is called at run-time (dynamic
binding)

10

CS330 Spring 2003
Copyright George Tzanetakis, University of Victoria

Smalltalk

- Pure-object oriented language
 - everything is an object
- Xerox Palo Alto Research Center 1970s
 - other contributions: mouse, windows, ide
- Self-contained interactive programming system (still a revolutionary idea)
- Alan Key – The best way to predict the future is to invent it – Lot's of little computers (objects)

11

CS330 Spring 2003
Copyright George Tzanetakis, University of Victoria

Primitive Objects

- Integer, Float, Boolean are objects
- expression $m+n$ requests m to return the sum of its own value and that of integer object n ; the result is another Integer object
- Dynamic Typing
 - Picture can contain arbitrary objects
 - Flexibility but insecurity

12

CS330 Spring 2003
Copyright George Tzanetakis, University of Victoria

Smalltalk notation

Eo is evaluated to determine the receiver object:

“Eo I” perform operation I, with no arguments. Example: n squared

“Eo * E1” requests the receiver object to performed the operator named * with the object yielded by E1 as argument. Example: n * 4

“Eo I1:E1....In:En” requests the receiver object to perform the operation named 'I1:....In', with the objects yielded by E1..En as arguments.

Example: 'm between: 1 and: n-1' where the operation name is 'between: and:'

For example an array:

monthsize at: 2 put: 29

Assignment expression:

V <- E

n <- n - m

13

CS330 Spring 2003
Copyright George Tzanetakis, University of Victoria

A smalltalk example

placeat: xnew and: ynew

x <- xnew . y <- ynew

(. is the sequencing operator)

xcoord

^ x

ycoord

^ y

moveby: xshift and: yshift

x <- x + xshift. y <- y + yshift

distance: other

^ ((x - other xcoord) squared

+ (y - other ycoord) squared) sqrt

14

CS330 Spring 2003
Copyright George Tzanetakis, University of Victoria

History C++

- > Started as “C with classes” by B.Stroustrup
 - > AT&T Bell Labs 1980s
- > Influenced by Simula67
- > Based on C (easy portability, interoperation with traditional C environments (Unix))
- > One of the most widely used languages
- > VERY COMPLEX and BIG but can be learned in smaller chunks

15

CS330 Spring 2003
Copyright George Tzanetakis, University of Victoria

Motivation

- > Systems programming
 - > No runtime overhead
 - > Explicit memory management
- > Support
 - > Modularity – Information Hiding
 - > Inheritance
 - > Dynamic Binding (controlled)
 - > Generic programming (templates)

16

CS330 Spring 2003
Copyright George Tzanetakis, University of Victoria