Explicit, Dynamic Memory Management with Temporal and Spatial Guarantees

Christoph Kirsch and Ana Sokolova
Universität Salzburg

Artist Summer School 2009
Tsinghua University, Beijing, China

Memory Management

- Allocation:
  - `malloc`

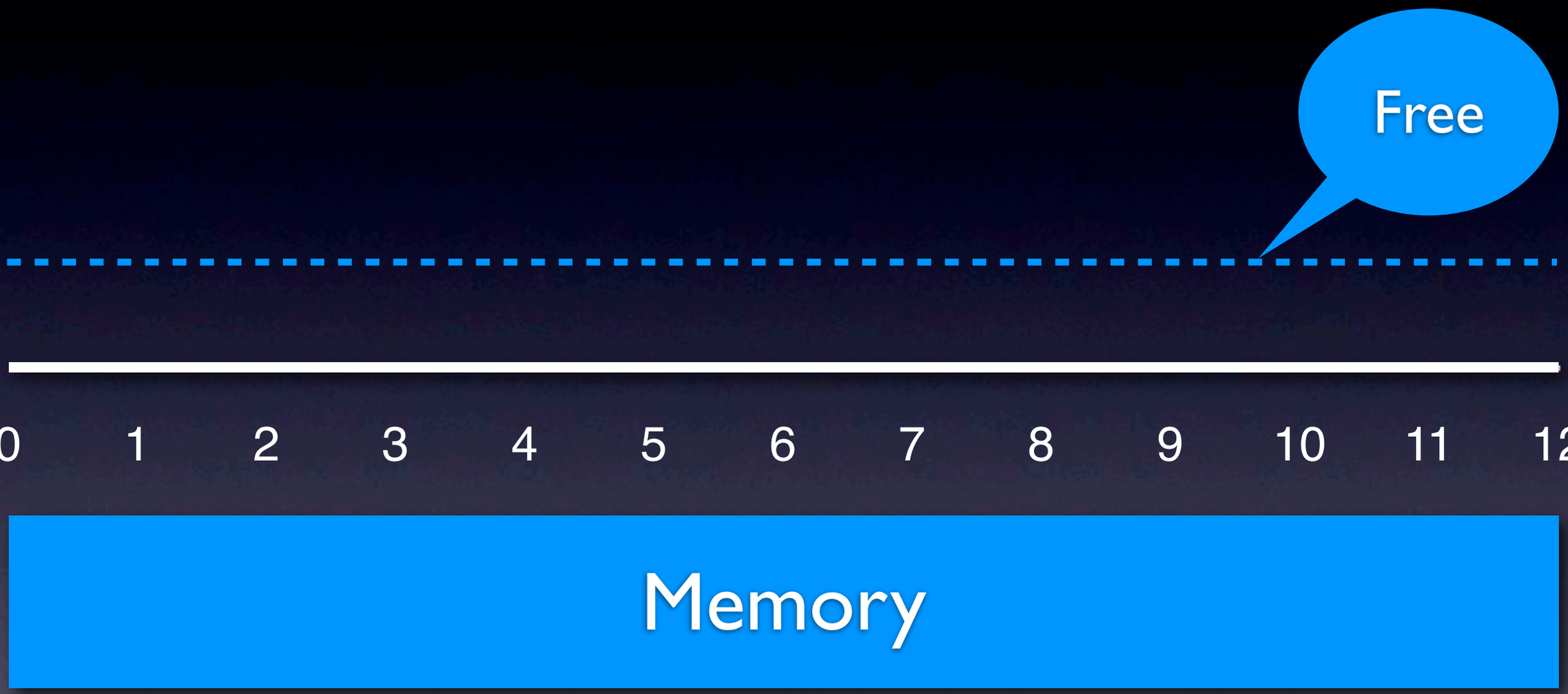
Memory Management

- Allocation:
  - `malloc`
- Deallocation:
  - `free`

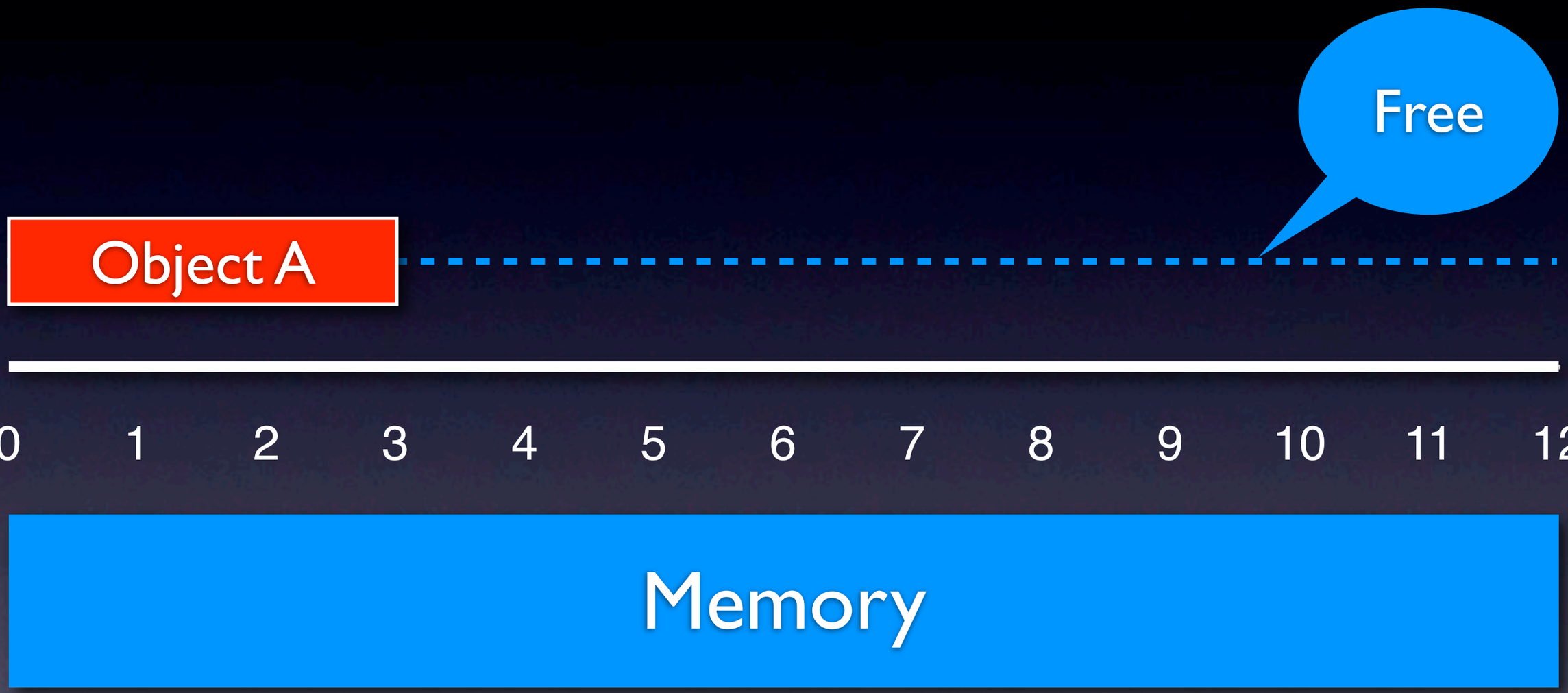
Memory Management

- Allocation:
  - `malloc`
- Deallocation:
  - `free`
- Access:
  - `read` and `write`

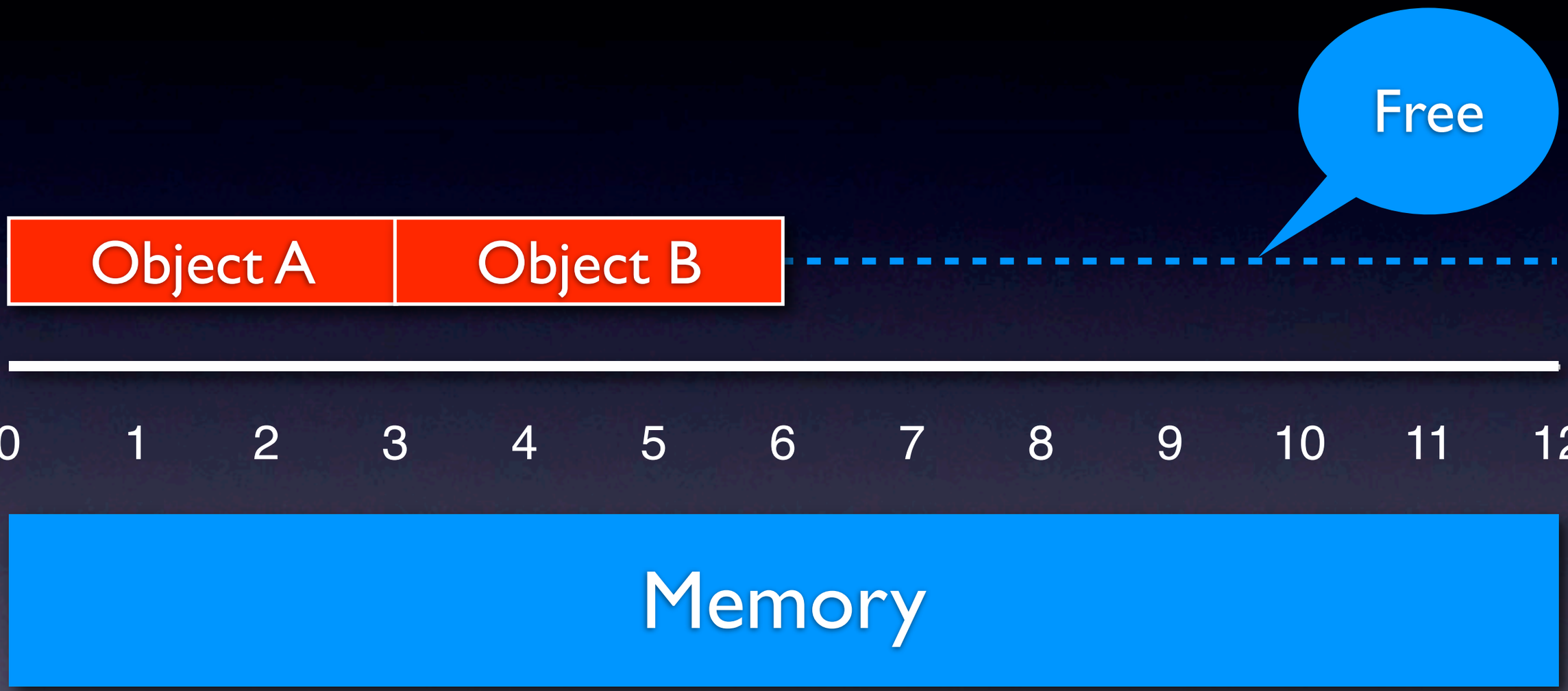
Allocation



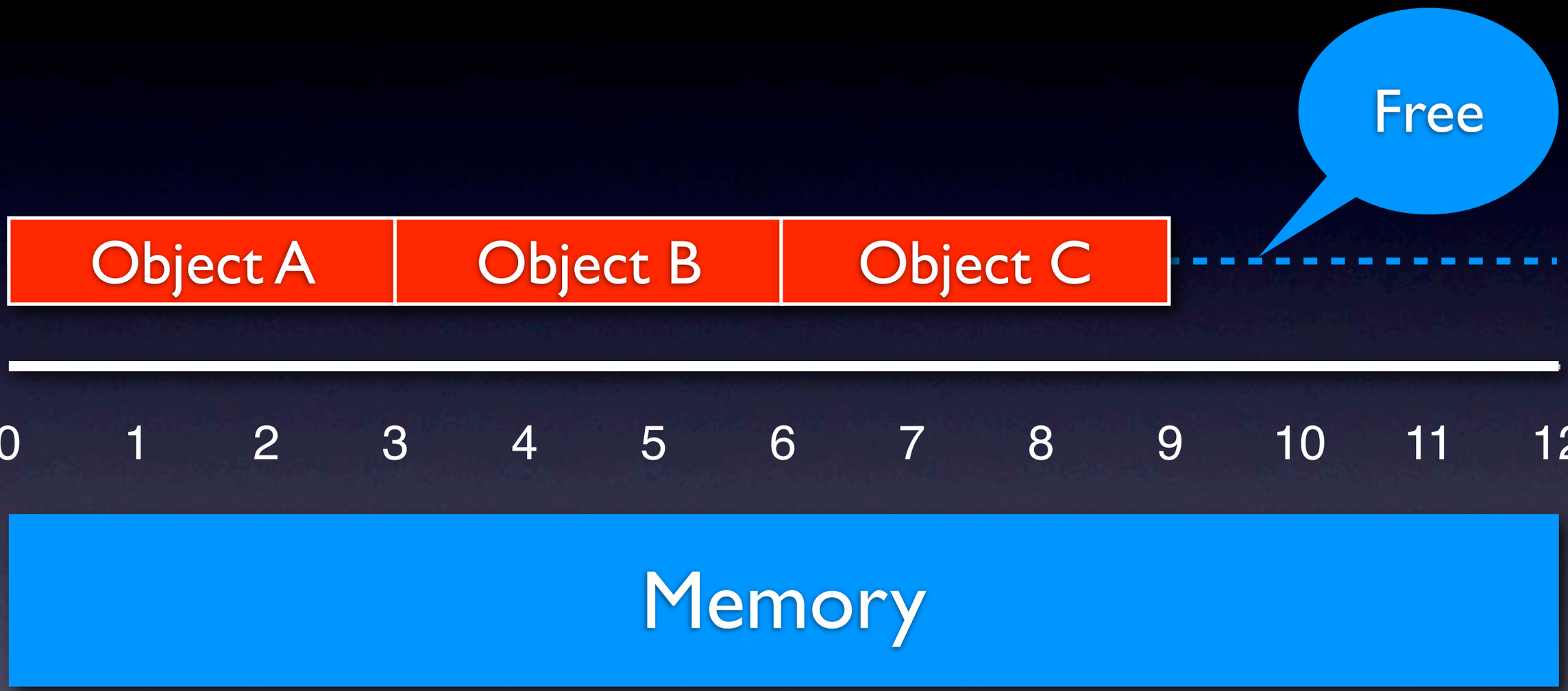
Allocation



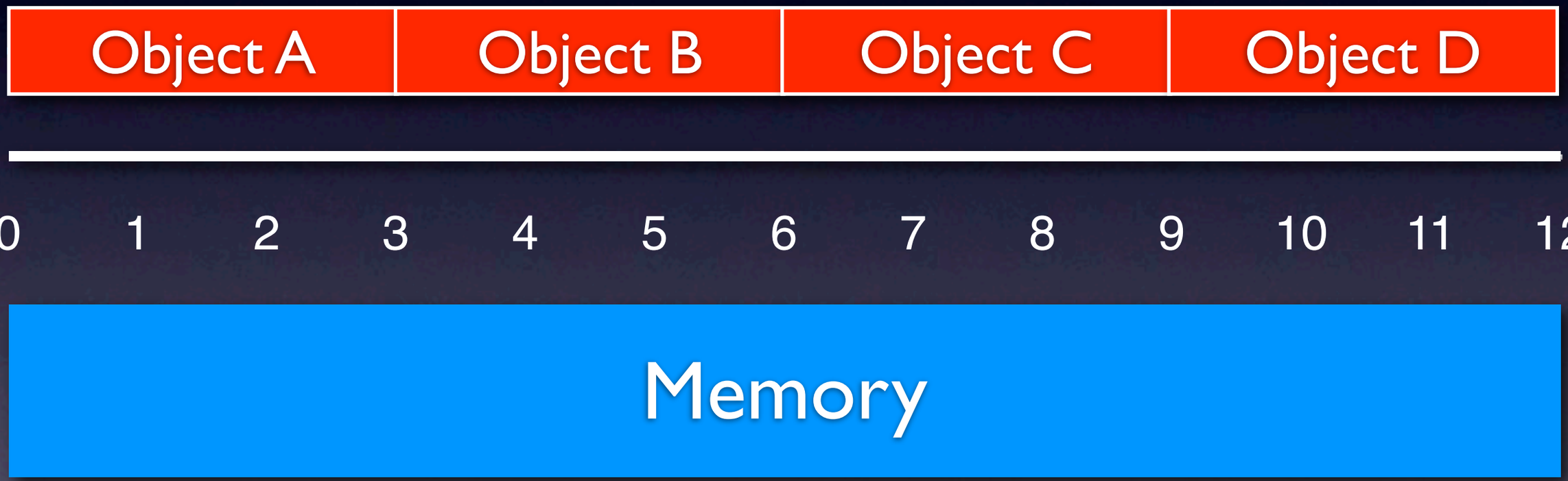
Allocation



Allocation



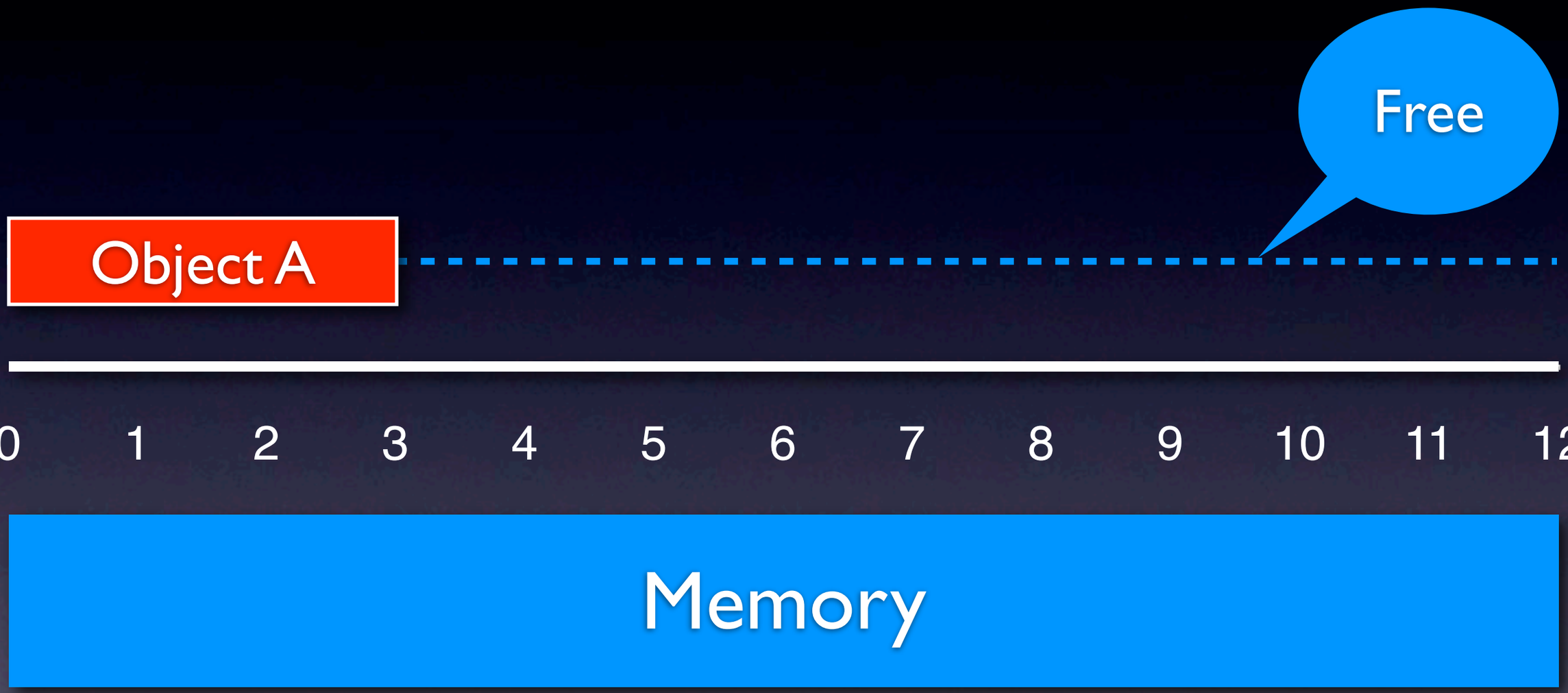
Allocation



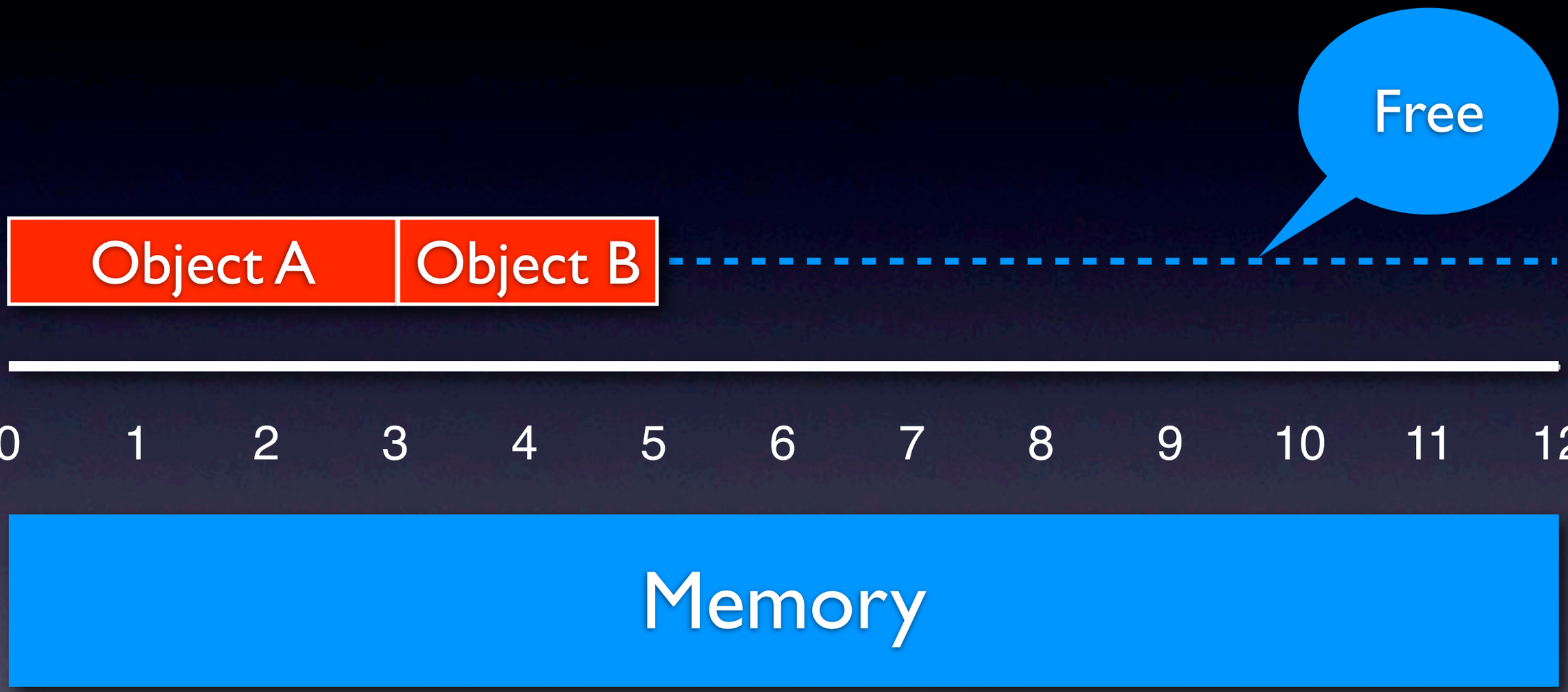
1. Assumption:

Objects may have different sizes

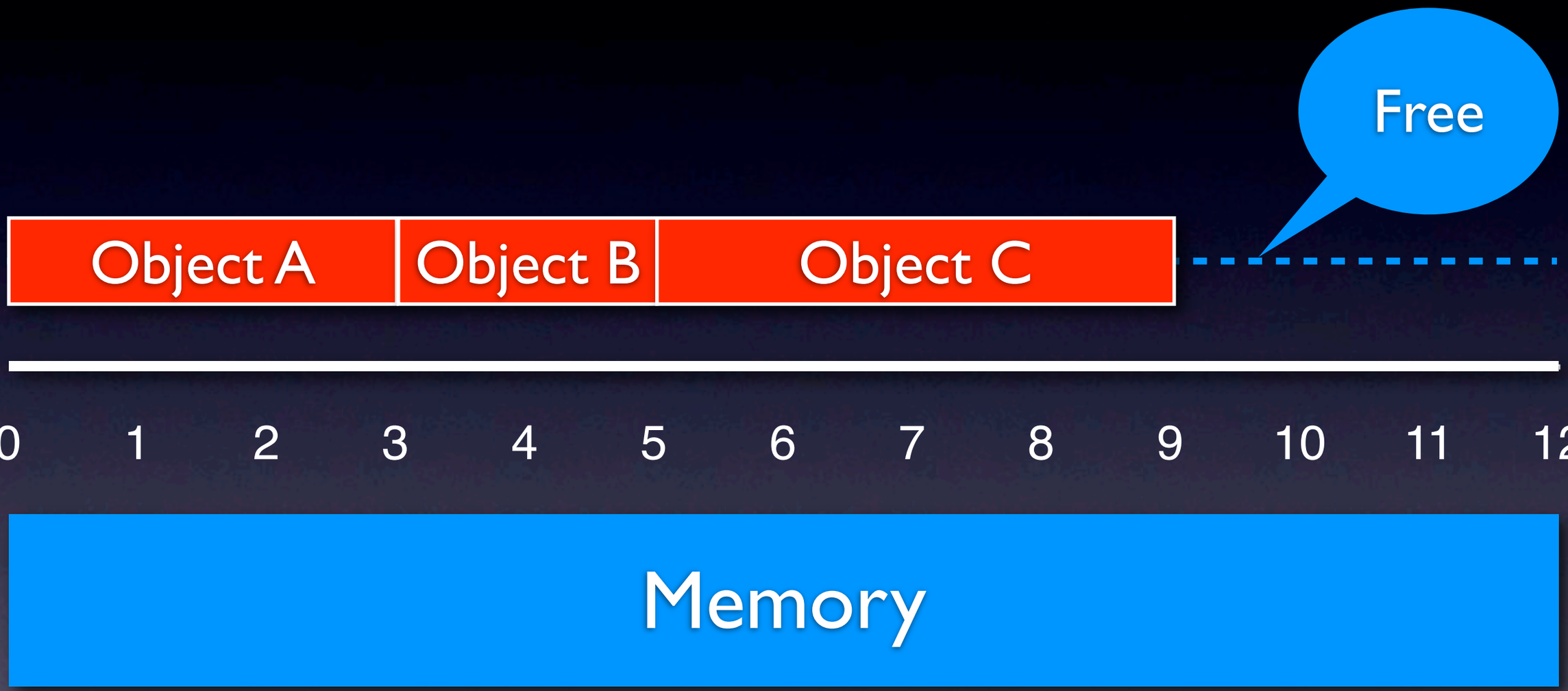
Allocation



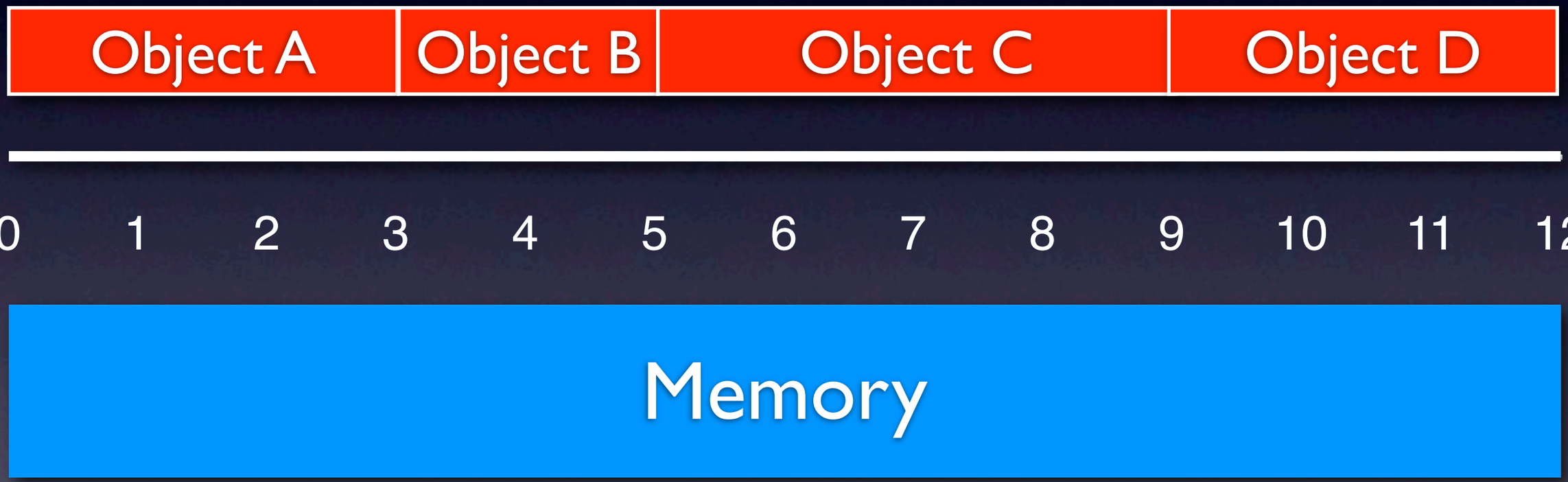
Allocation



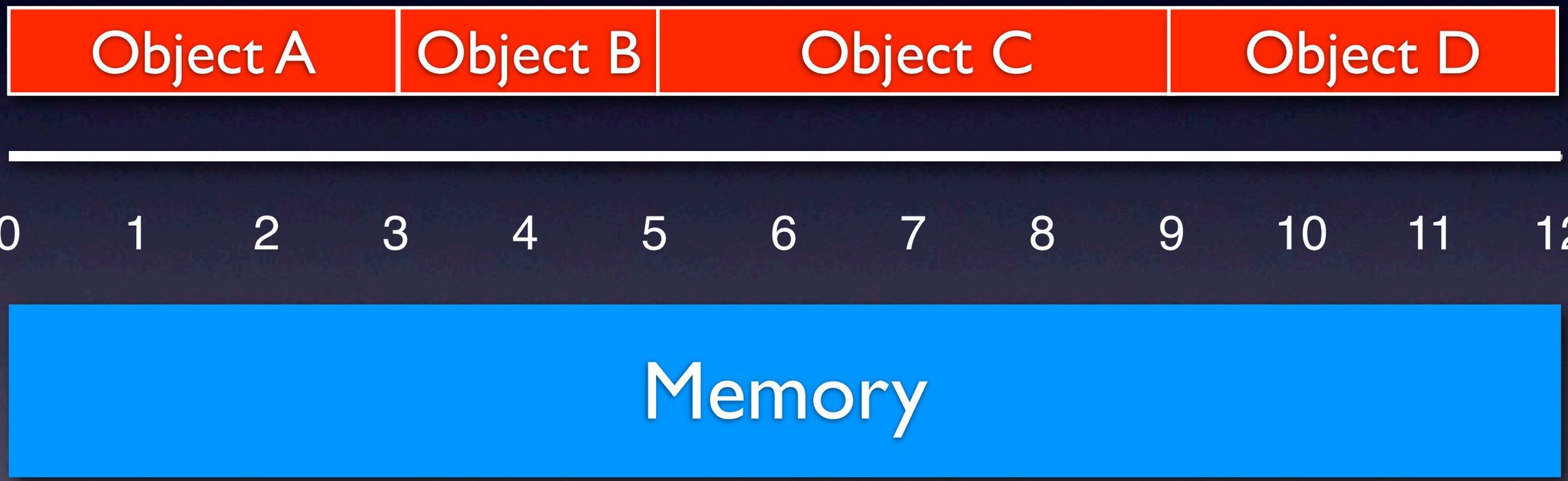
Allocation



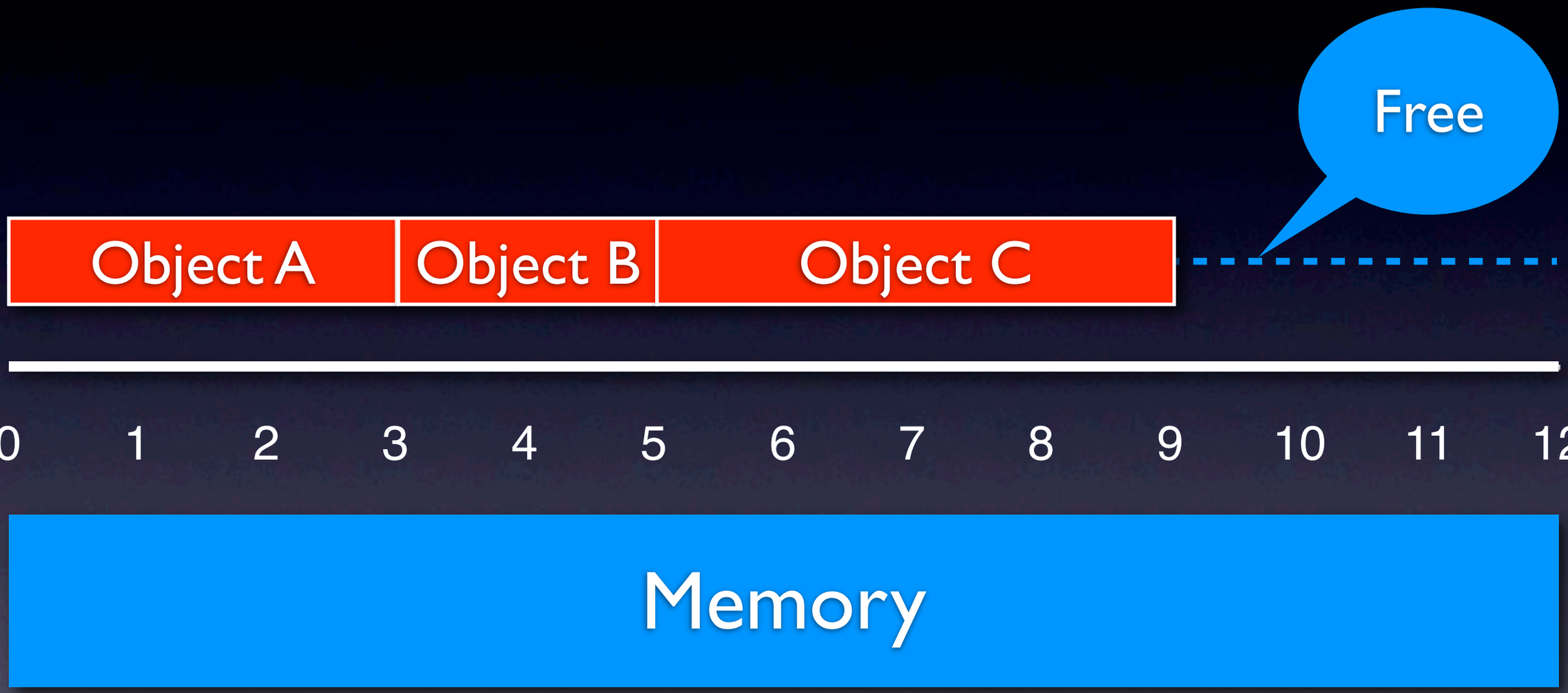
Allocation



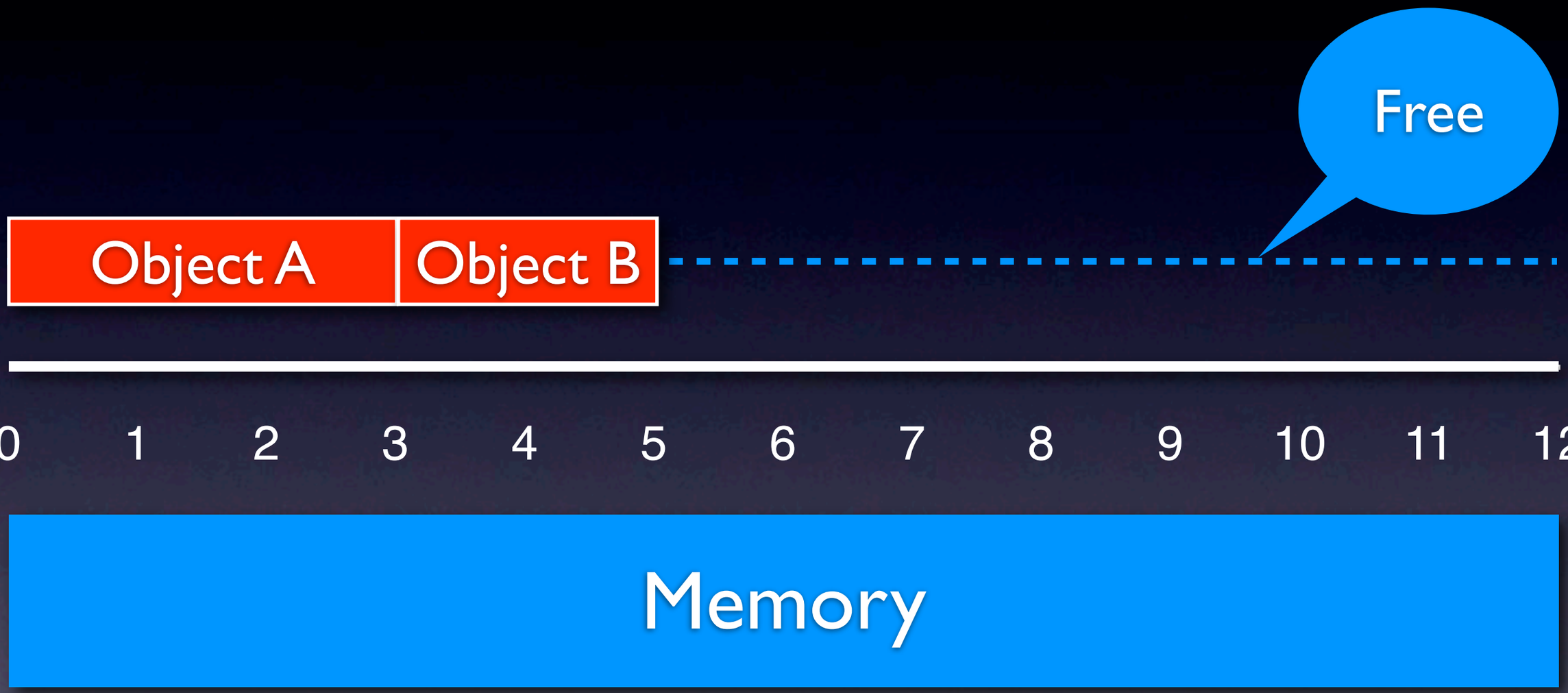
Deallocation



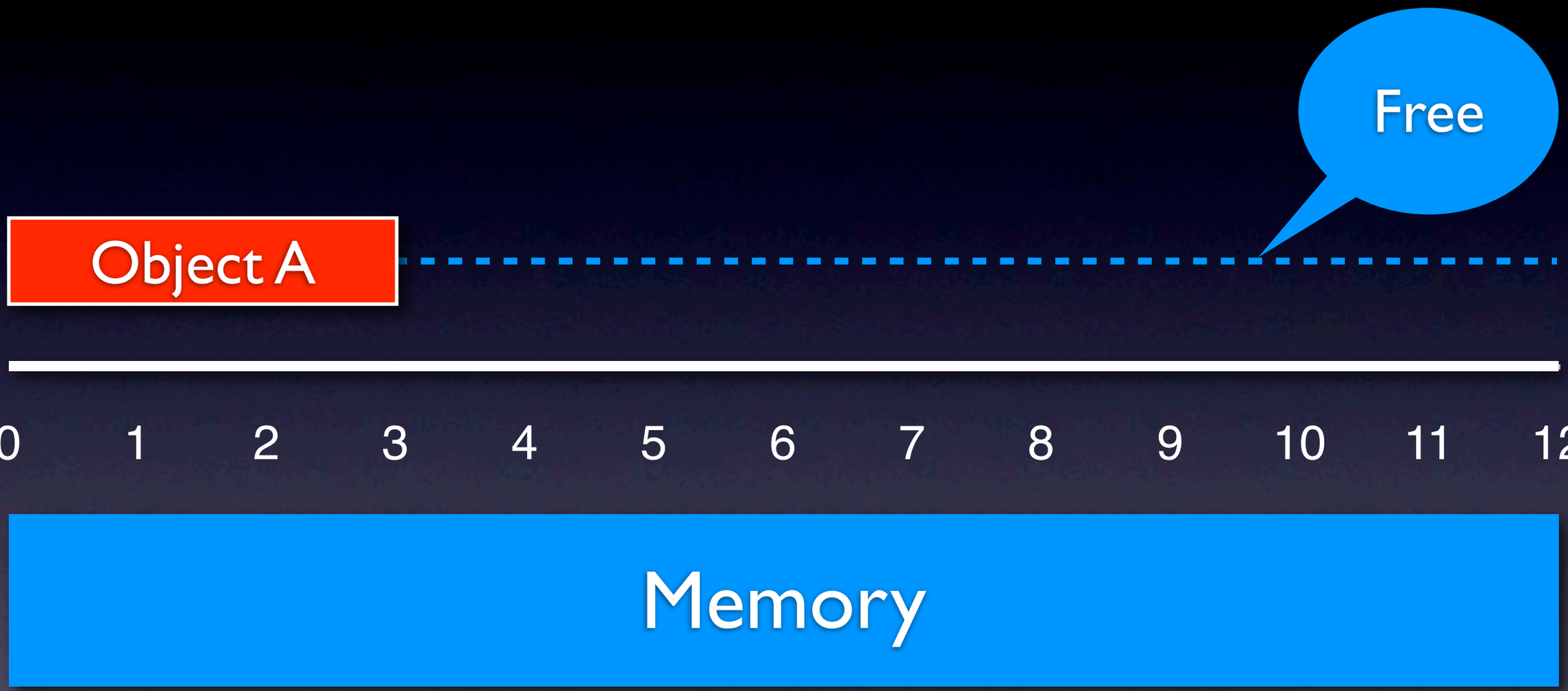
Deallocation



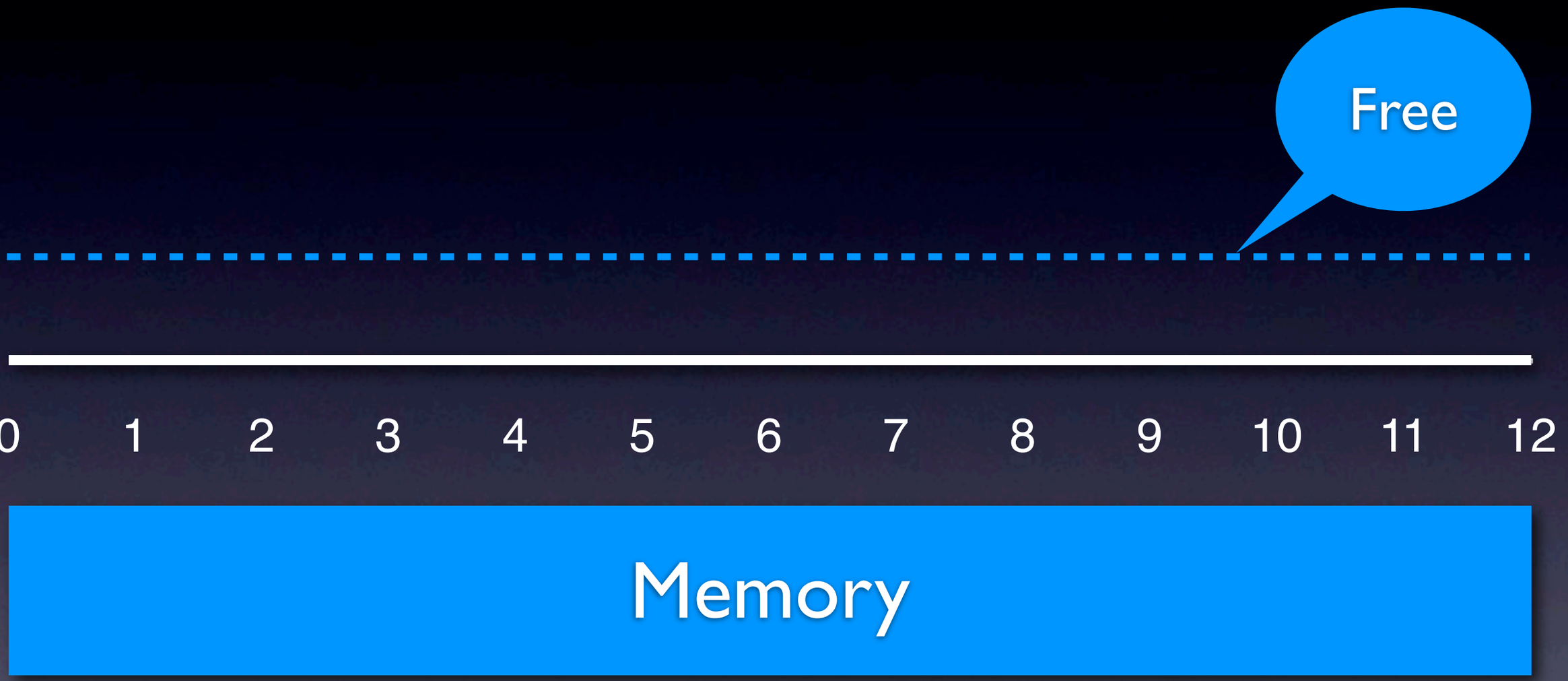
Deallocation



Deallocation



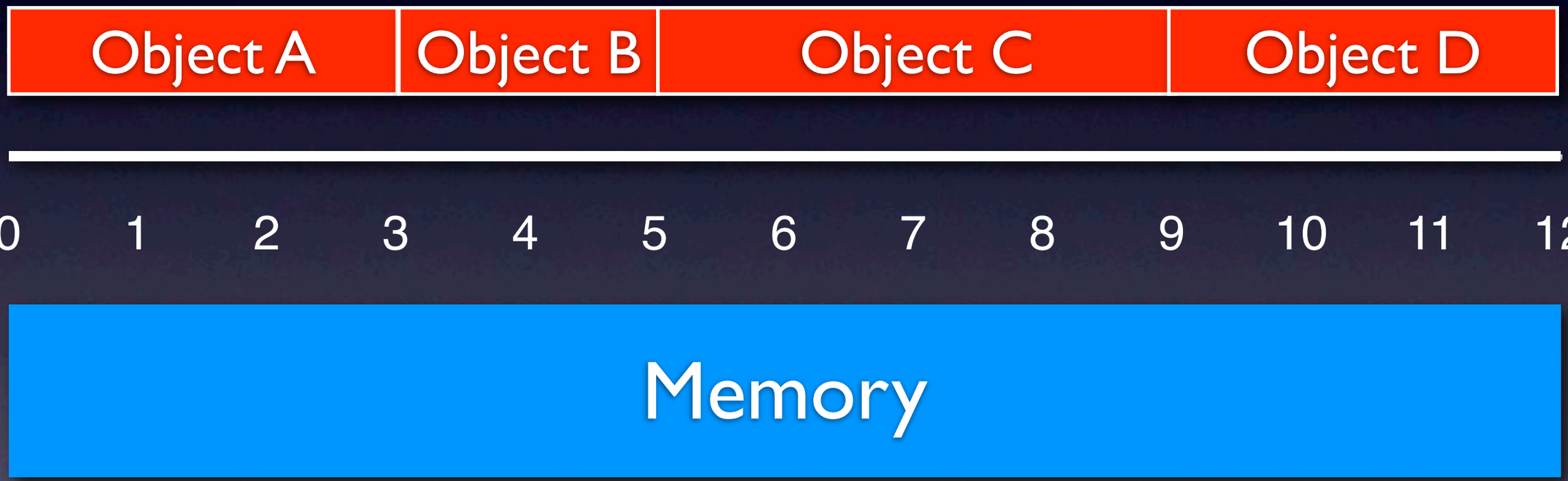
Deallocation



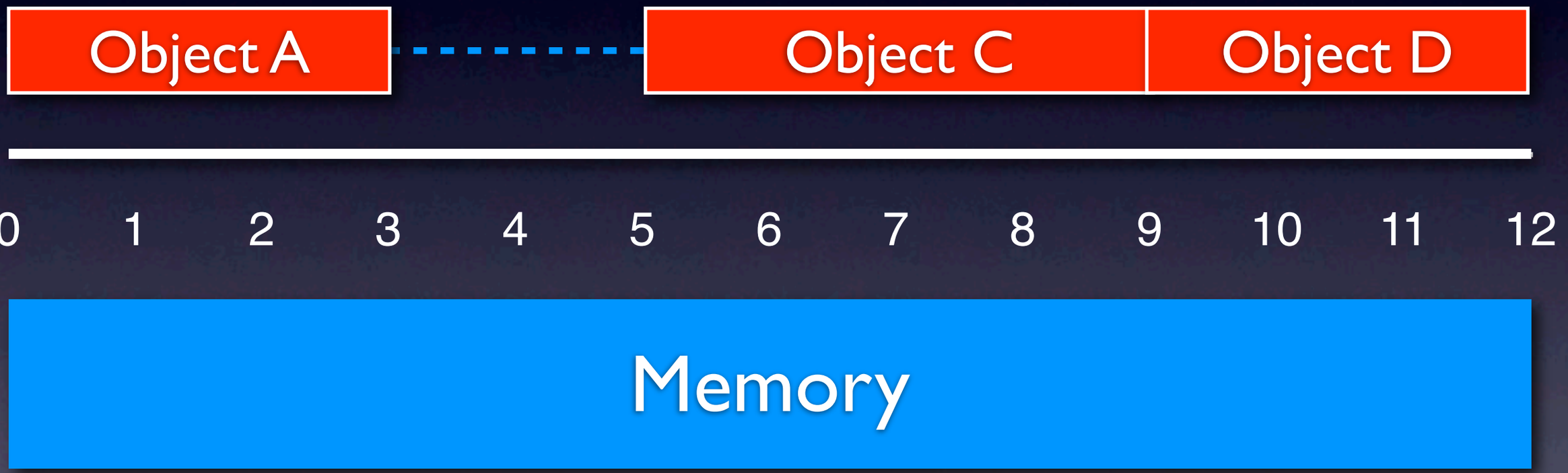
2. Assumption:

Objects may be allocated and deallocated in random order

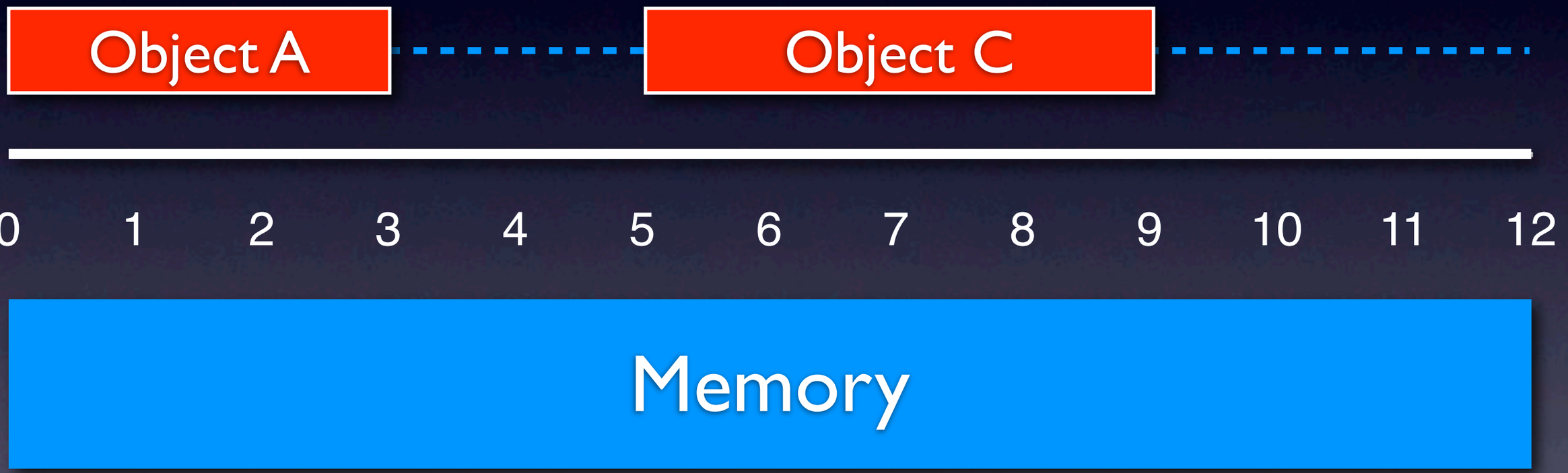
Deallocation



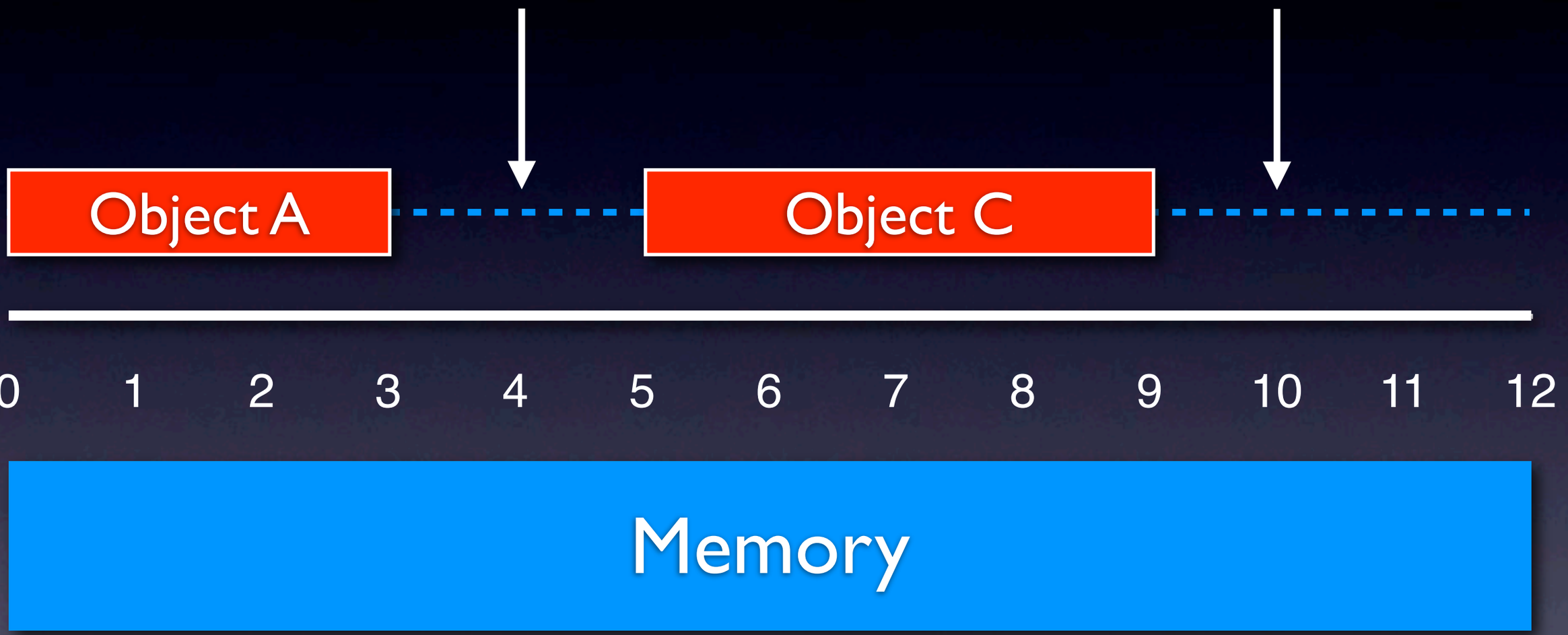
Deallocation



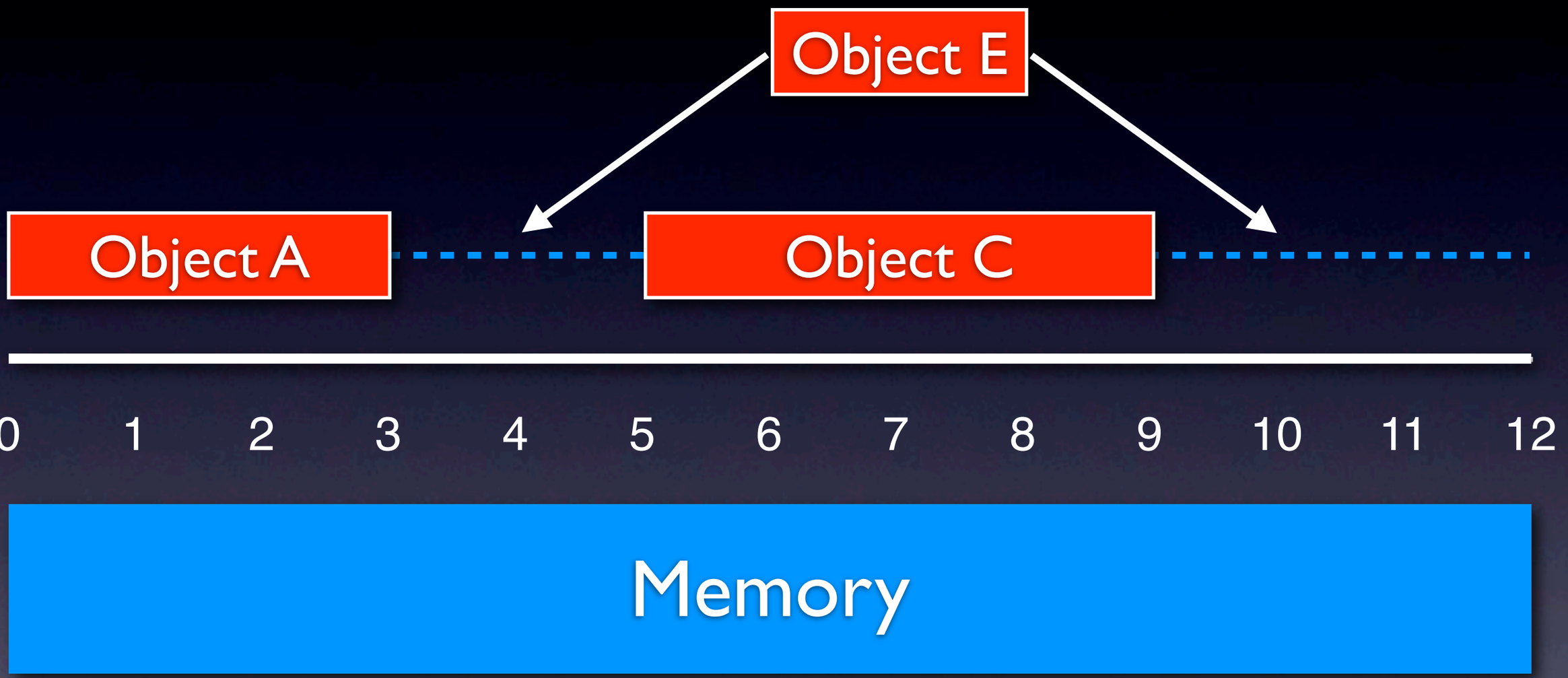
Deallocation



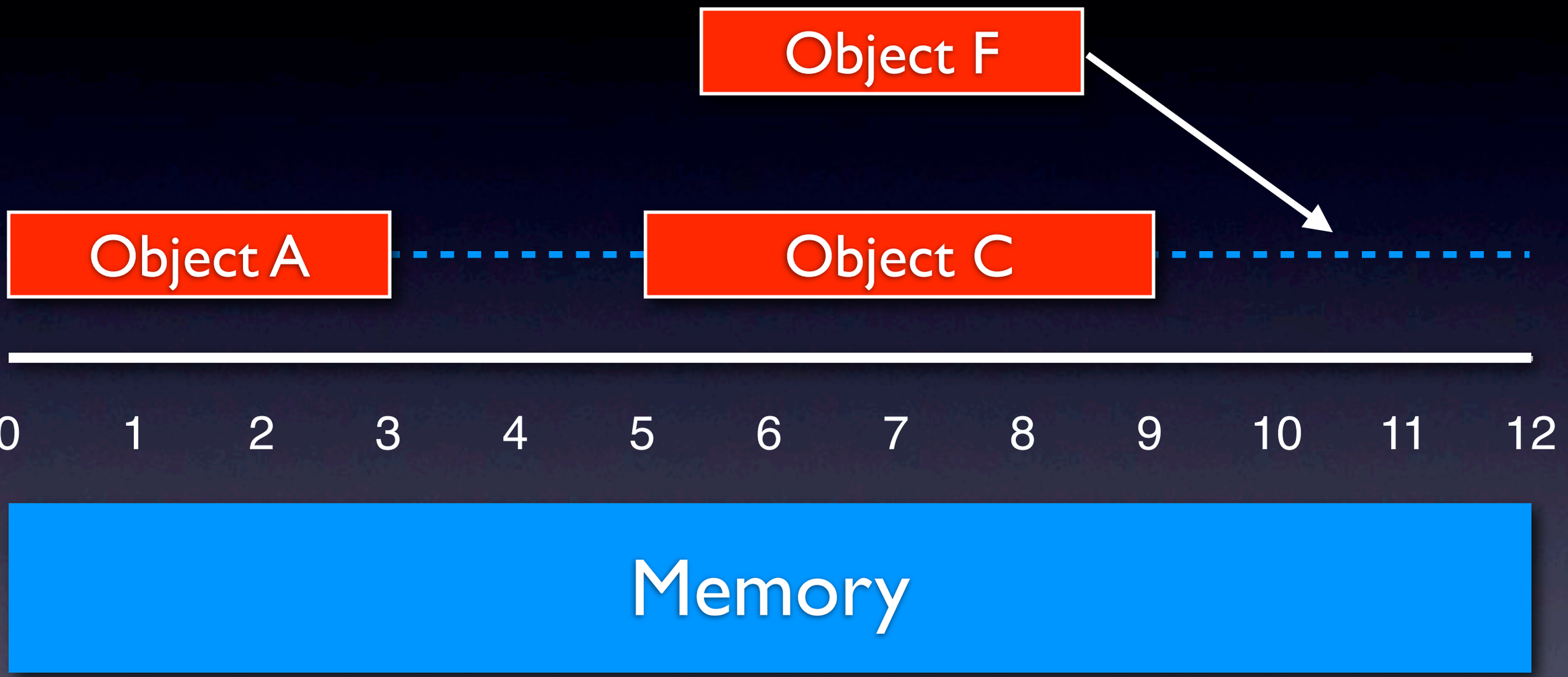
External Fragmentation



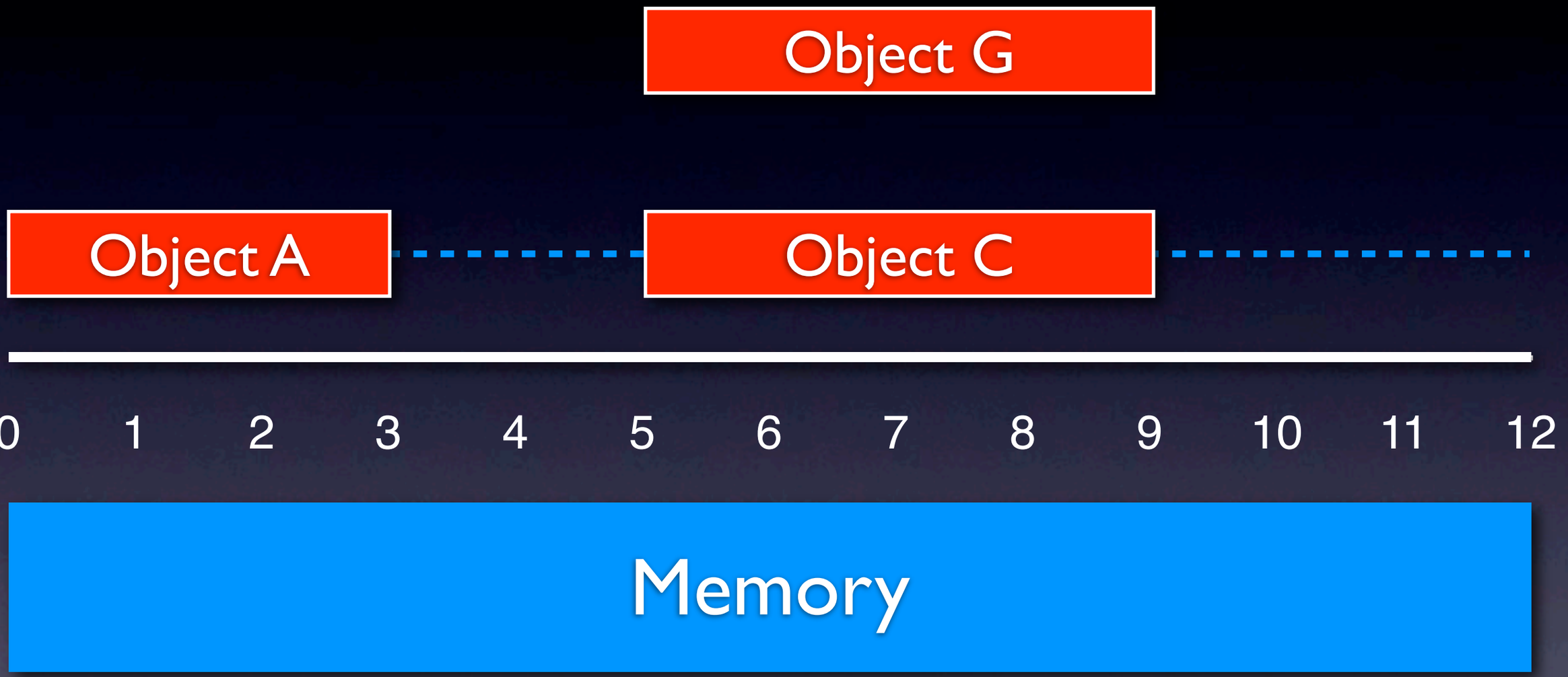
Allocation



Allocation



Allocation



Memory is *fragmented* if the largest, contiguous piece of available space is smaller than the total available space

Fragmentation

- Memory objects may have different sizes
- Memory objects may be allocated and deallocated in random order
  - creates the problem of memory fragmentation!

Explicit, Dynamic Memory Management with Temporal and Spatial Guarantees

Static versus Dynamic

- Static memory management:
  - Preallocate all memory at compile time

Static versus Dynamic

- Static memory management:
  - Preallocate all memory at compile time
- Dynamic memory management:
  - Allocate and deallocate memory at run time

Explicit, Dynamic Memory Management with Temporal and Spatial Guarantees

Implicit versus Explicit

- Implicit, dynamic memory management:
  - Garbage collector (GC) deallocates objects, not programmer (implicit `free` calls by GC)

Implicit versus Explicit

- Implicit, dynamic memory management:
  - Garbage collector (GC) deallocates objects, not programmer (implicit `free` calls by GC)
- Explicit, dynamic memory management:
  - Objects are deallocated by programmer (explicit `free` calls)

Programming Abstraction

Runtime Overhead

Implicit, Dynamic Memory Management

Explicit, Dynamic Memory Management

Static Memory Management

Programming Abstraction

Runtime Overhead

Implicit

Web, Safety

Explicit

Server, Performance

Static

Embedded, Real Time

Programming Abstraction

Runtime Overhead

Implicit

Web, Safety

Explicit

Server, Performance

Static

Embedded, Real Time

Programming Abstraction

Runtime Overhead

Implicit

Web, Safety

Explicit

Server, Performance

Static

Embedded, Real Time

Temporal Performance

- Throughput:
  - 10MB/s allocation rate
  - 10MB/s deallocation rate

Temporal Performance

- Throughput:
  - 10MB/s allocation rate
  - 10MB/s deallocation rate
- Latency/Responsiveness:
  - 1ms execution time (malloc/free)
  - 0.1ms preemption time (malloc/free)

Spatial Performance

- Degree of fragmentation:
  - The number of contiguous pieces of memory of a given size that can still be allocated

Spatial Performance

- Degree of fragmentation:
  - The number of contiguous pieces of memory of a given size that can still be allocated
- Administrative space:
  - meta data structures (used, free lists)

There is a trade-off
between
temporal and spatial
performance

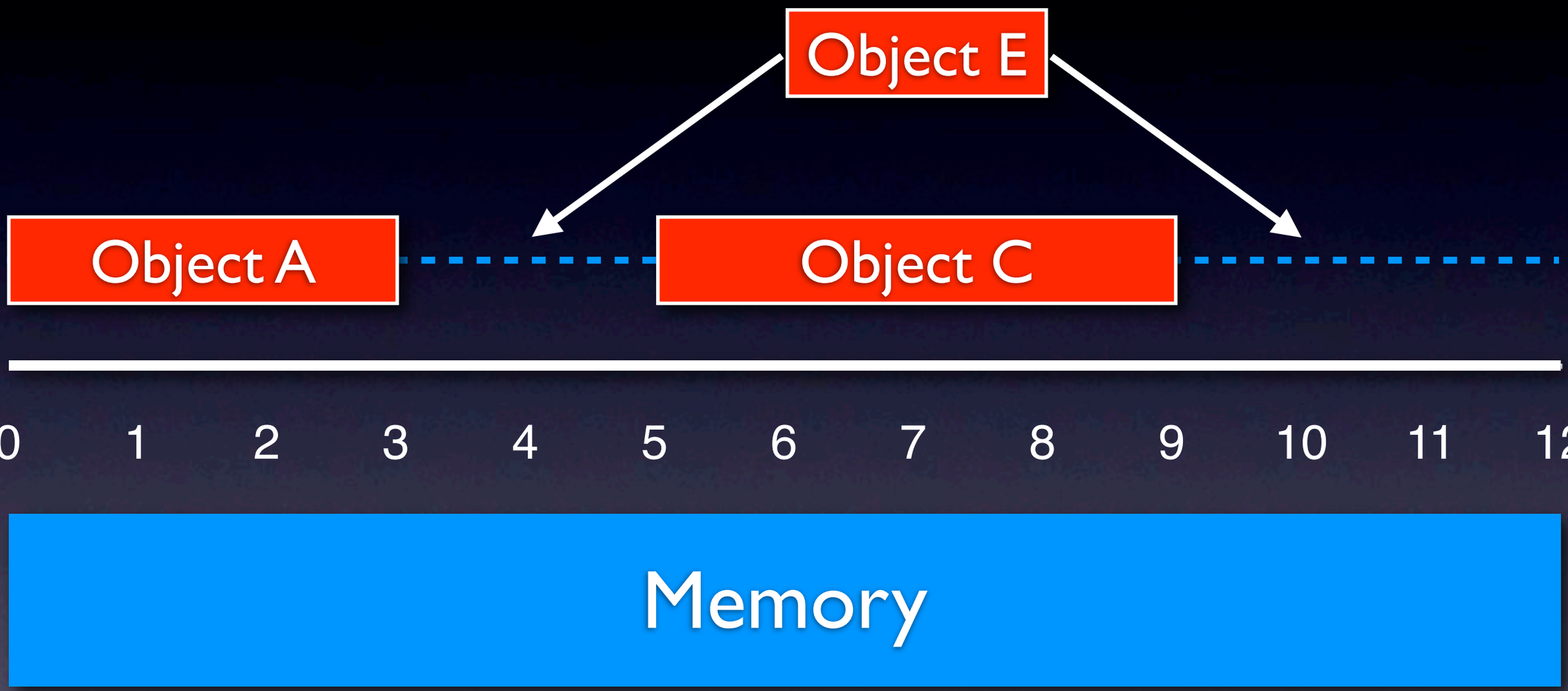
Temporal Predictability

- Unpredictable complexity (in terms of input):
  - allocation/deallocation may take time proportional to the total size of memory

Temporal Predictability

- Unpredictable complexity (in terms of input):
  - allocation/deallocation may take time proportional to the total size of memory
- Predictable complexity (in terms of input):
  - allocation/deallocation takes time at most proportional to the size of involved object
  - access takes time at most proportional to the size of involved object

Allocation Complexity



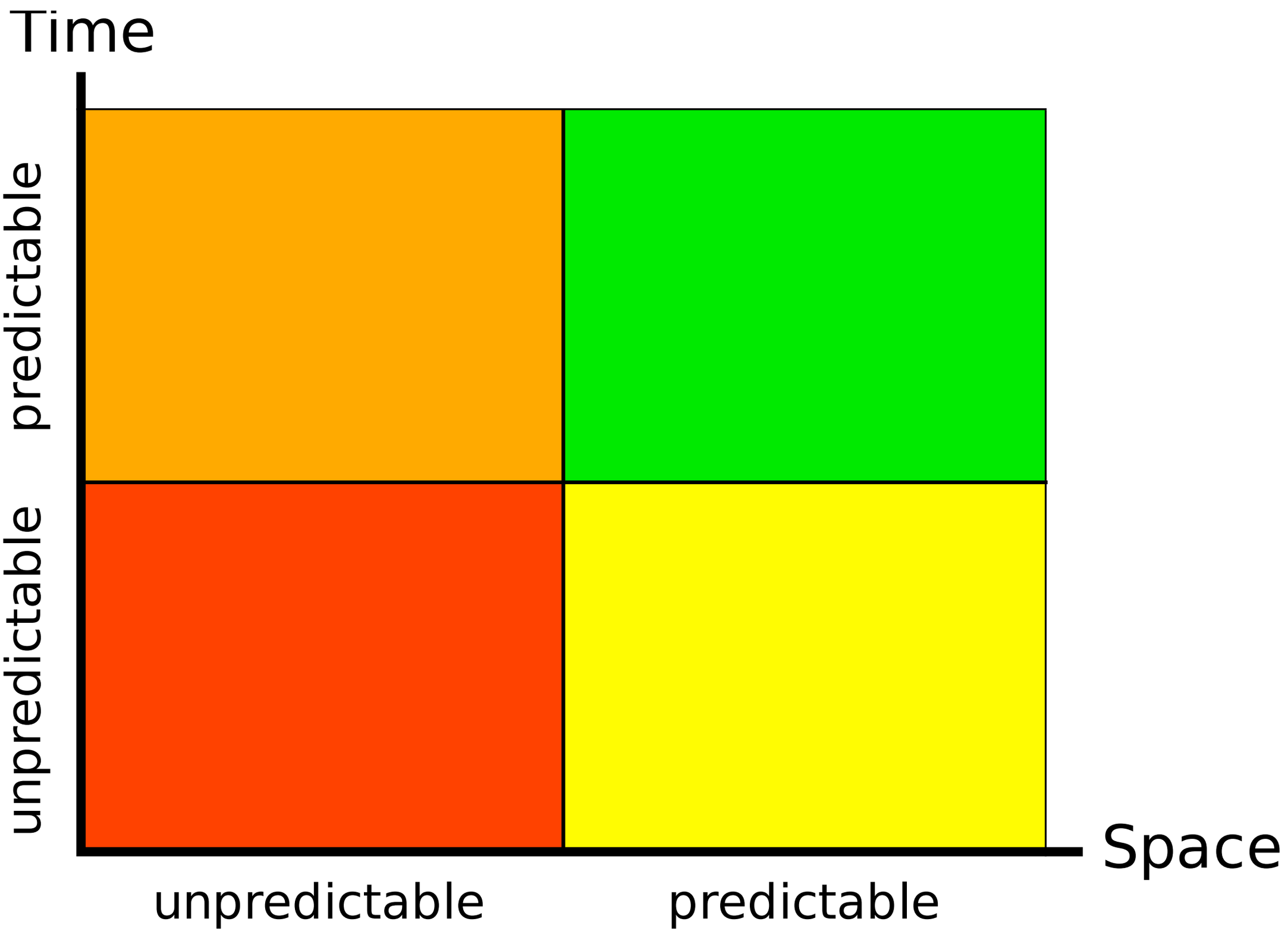
It may be difficult to
improve
average performance
but it may still be possible to
improve
predictability
without loosing too much
performance

Spatial Predictability

- Unpredictable fragmentation:
  - the degree of fragmentation may depend on the full allocation and deallocation history, i.e., the order of invocations

Spatial Predictability

- Unpredictable fragmentation:
  - the degree of fragmentation may depend on the full allocation and deallocation history, i.e., the order of invocations
- Predictable fragmentation:
  - the degree of fragmentation only depends on the number of allocations and deallocations, independently of the order of invocations



Explicit, Dynamic Memory Management with Temporal and Spatial Guarantees

Programming Abstraction

Runtime Overhead

Implicit

Web, Safety

Explicit

Server, Performance

Static

Embedded, Real Time

Programming Abstraction

Runtime Overhead

Implicit

Web, Safety

Explicit

Server, Performance

Static

Embedded, Real Time

Programming Abstraction

Runtime Overhead

Implicit

Web, Safety

Explicit

Server, Performance

Static

Embedded, Real Time

tiptoe.cs.uni-salzburg.at[#]

- Silviu Craciunas[#] (Programming Model)
- Andreas Haas (Memory Management)
- Hannes Payer [#] (Memory Management)
- Harald Röck (VM, Scheduling)
- Ana Sokolova* (Theoretical Foundation)

[#]Supported by a 2007 IBM Faculty Award, the EU ArtistDesign Network of Excellence on Embedded Systems Design, and Austrian Science Fund Project P18913-N15.
*Supported by Austrian Science Fund Project V00125.

Tiptoe

- Tiptoe is a microkernel-based virtual machine and process monitor for embedded systems

Tiptoe

- Tiptoe is a microkernel-based virtual machine and process monitor for embedded systems
- Tiptoe virtualizes the host platform (system VM) and provides infrastructure to run process VMs and processes in real time

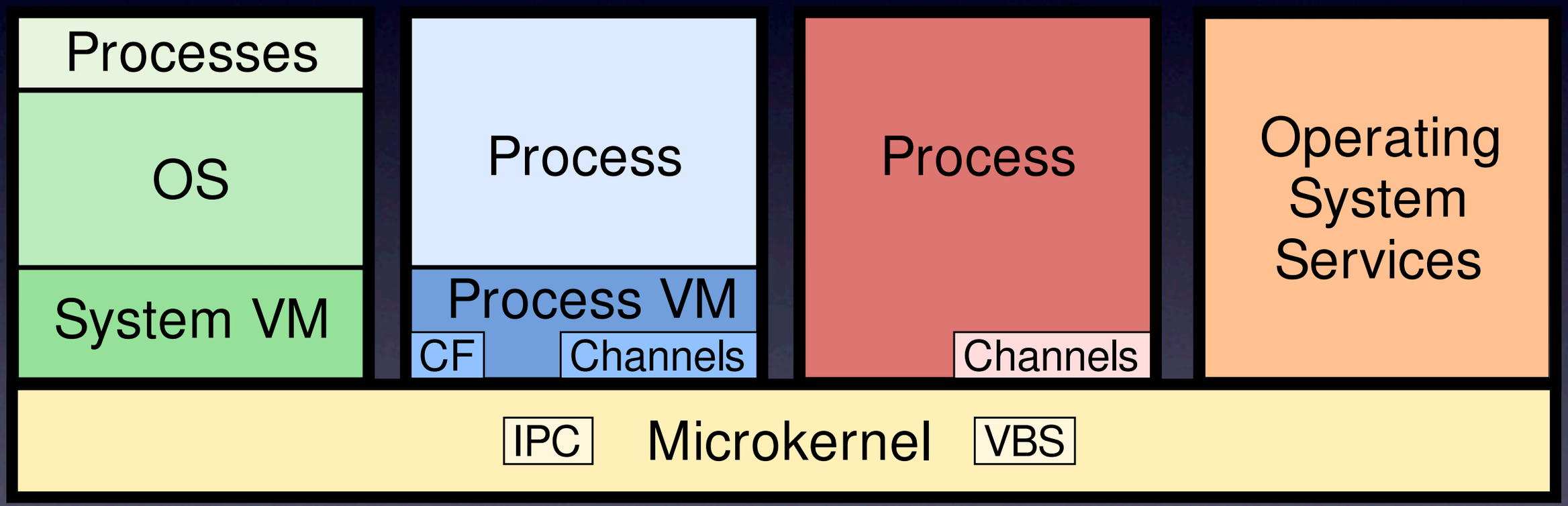
Tiptoe

- Tiptoe is a microkernel-based virtual machine and process monitor for embedded systems
- Tiptoe virtualizes the host platform (system VM) and provides infrastructure to run process VMs and processes in real time
- Tiptoe controls throughput and latency of CPU, memory, and I/O

Tiptoe

- Tiptoe is a microkernel-based virtual machine and process monitor for embedded systems
- Tiptoe virtualizes the host platform (system VM) and provides infrastructure to run process VMs and processes in real time
- Tiptoe controls throughput and latency of CPU, memory, and I/O
- I/O is multiplexed through IPC to a system VM running Linux

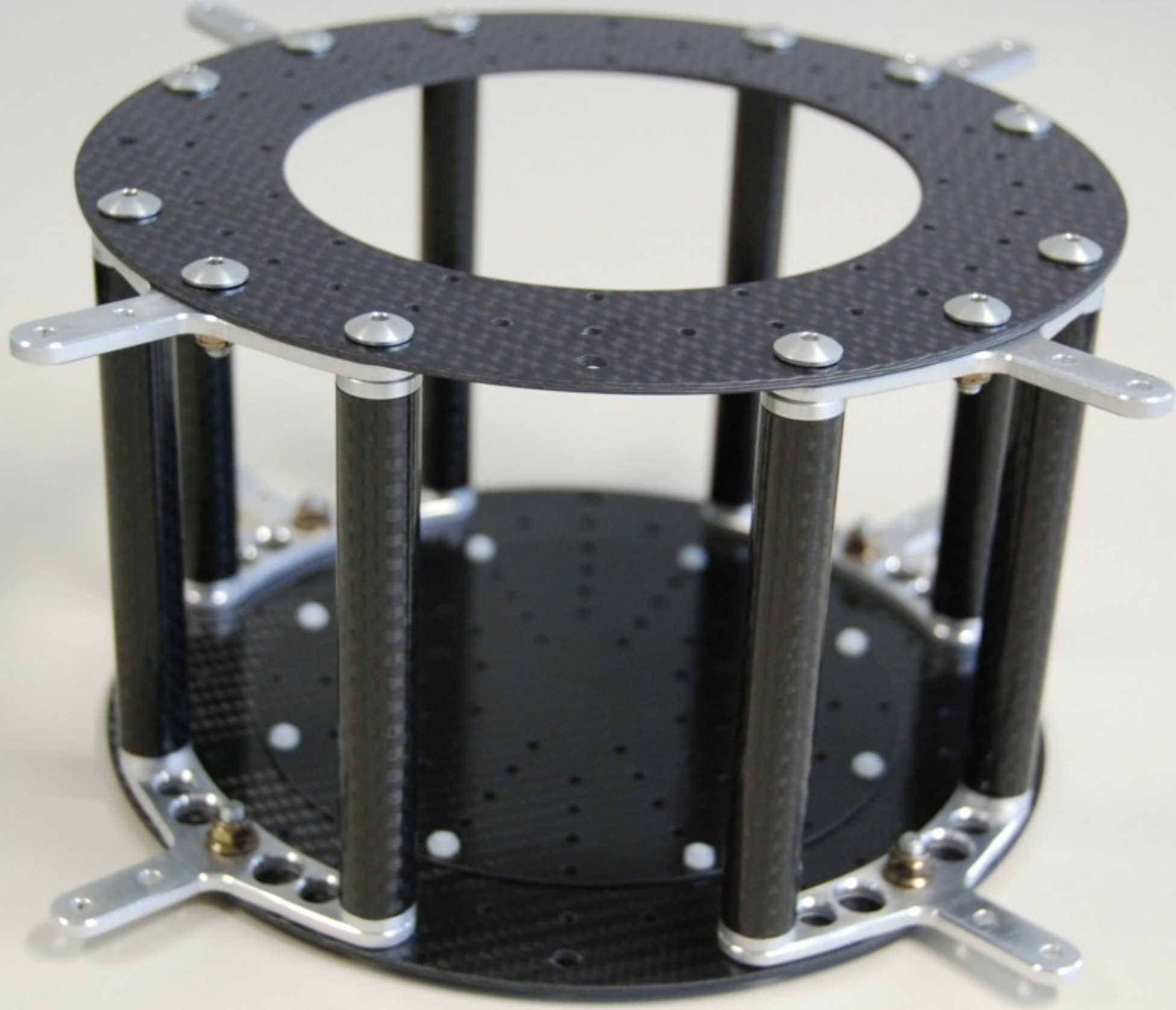
Tiptoe

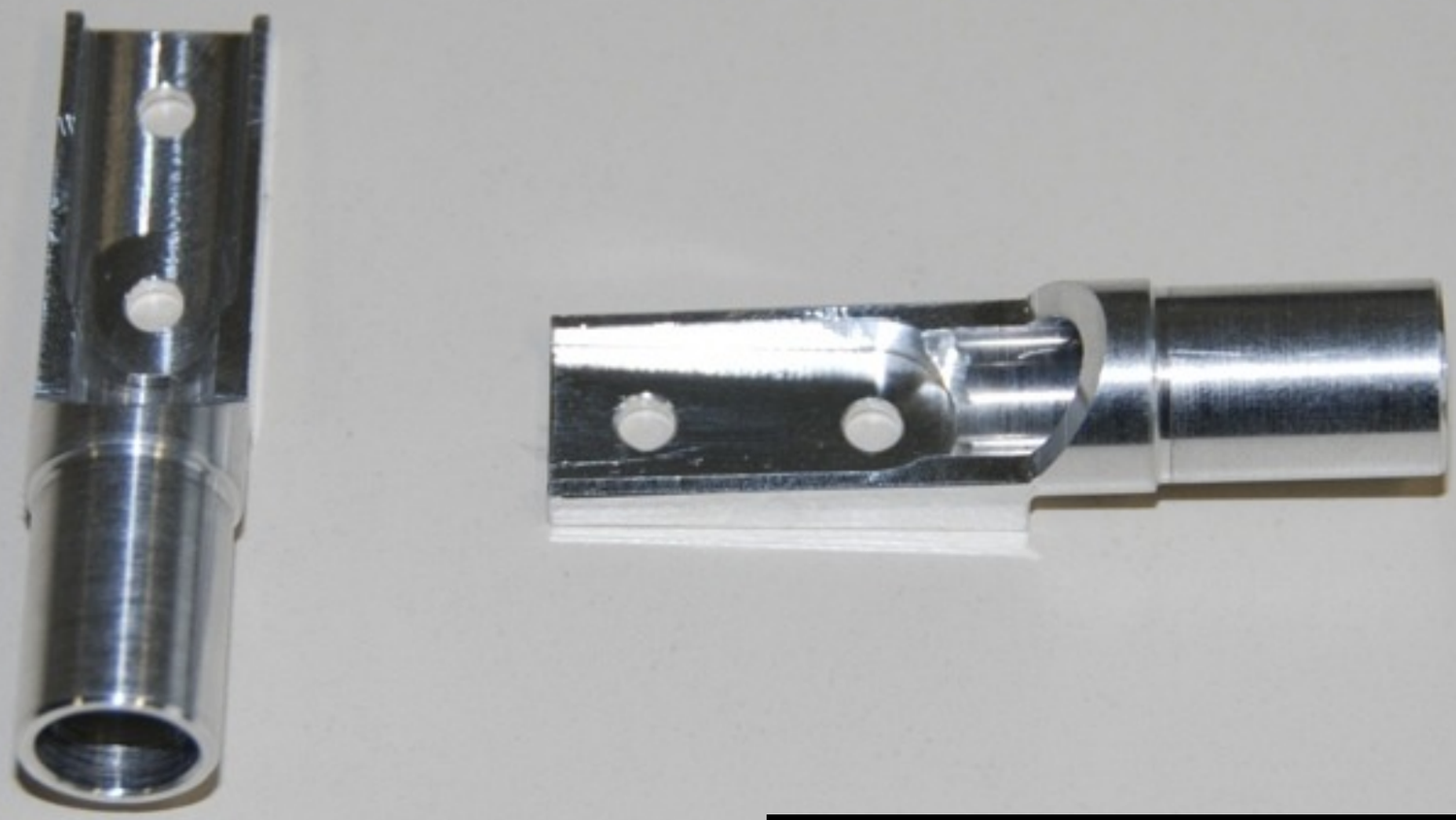


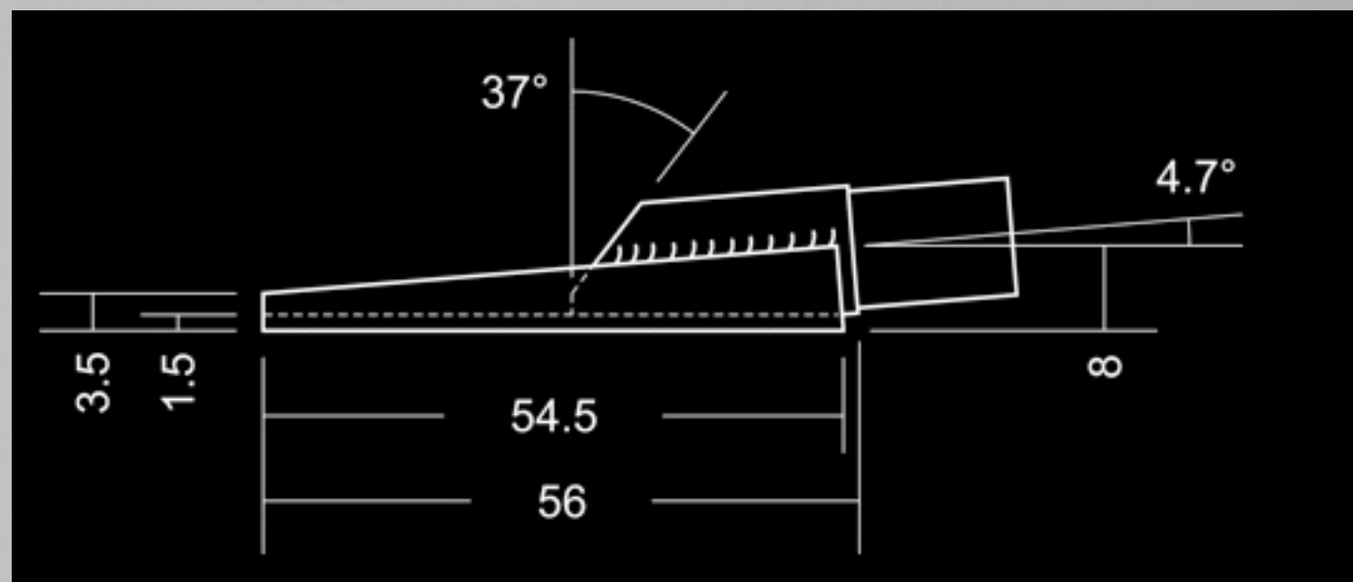
The JAviator

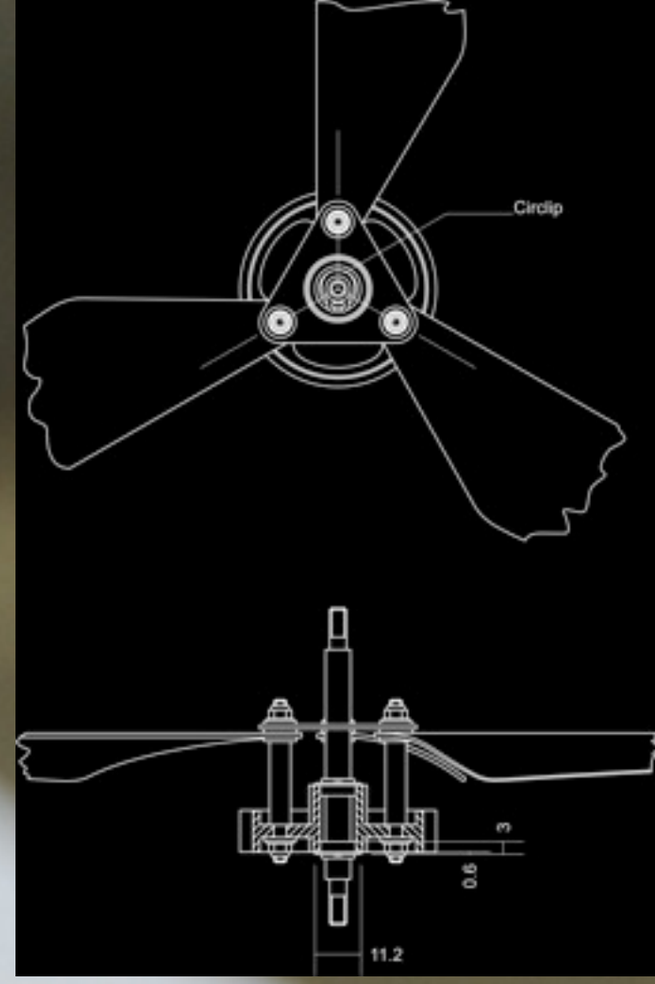
javiator.cs.uni-salzburg.at

Quad-Rotor Helicopter

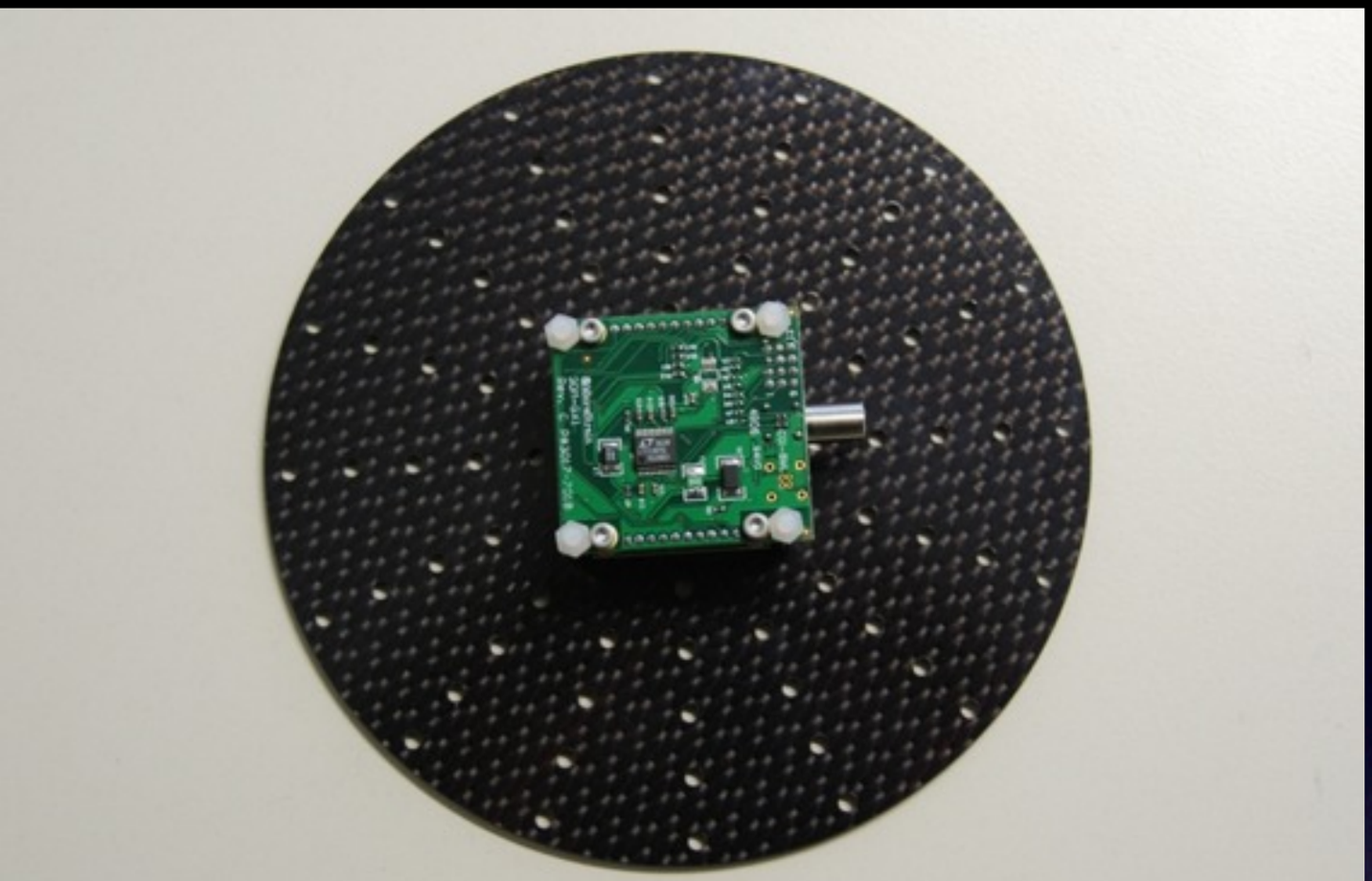





37°
4.7°
3.5
1.5
54.5
56
8

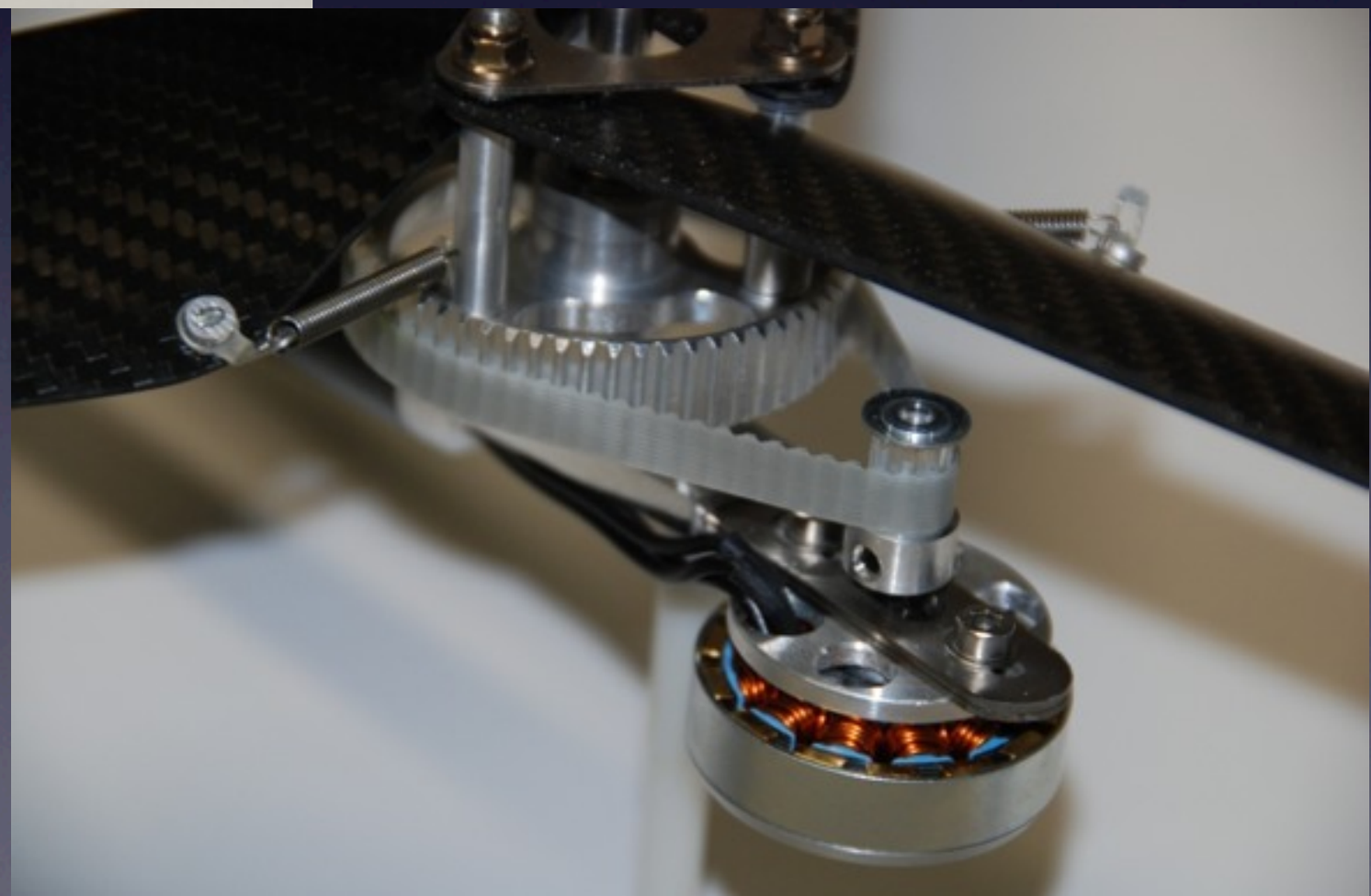

Circlip
3
0.5
11.2


25/12/2005

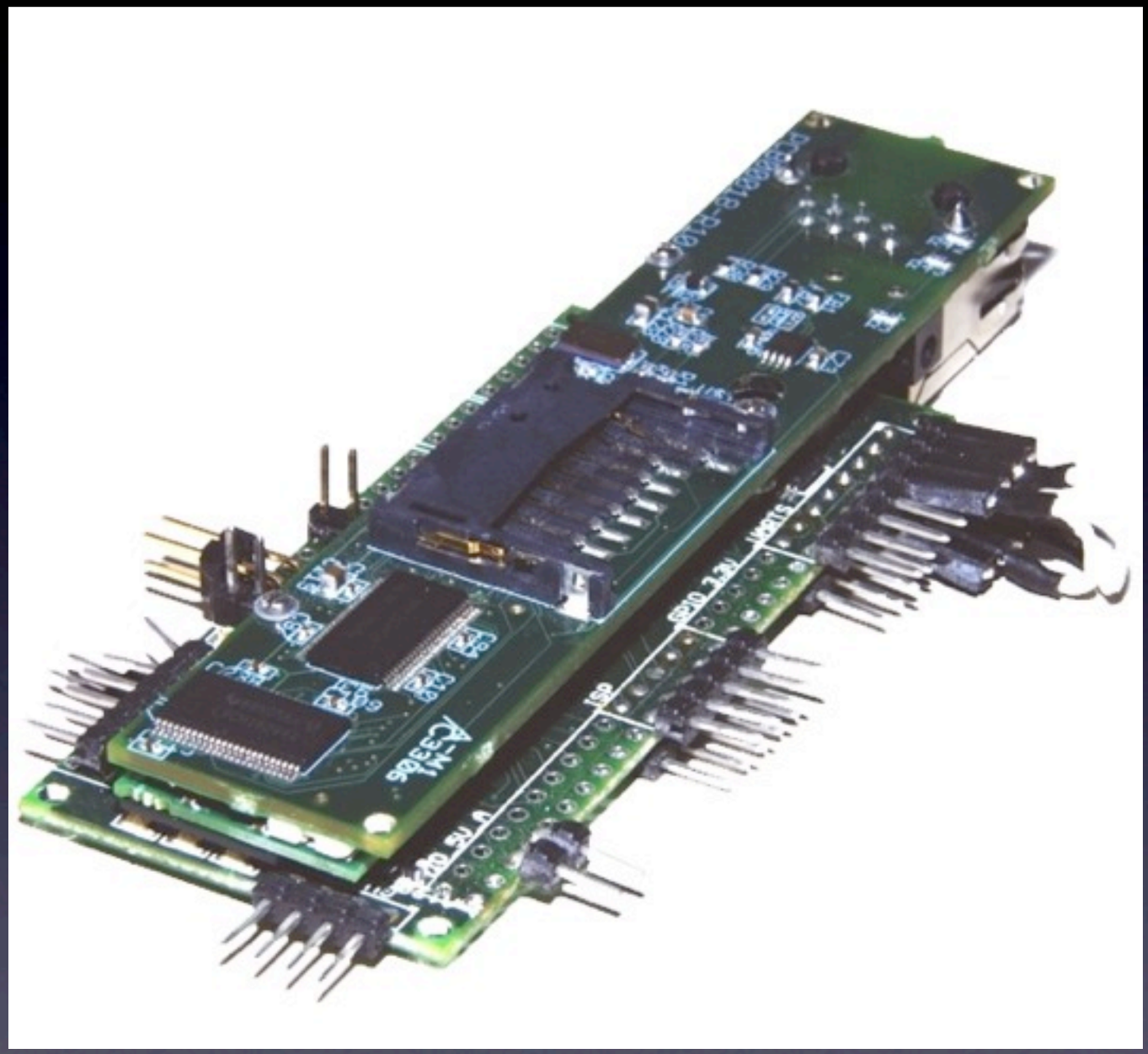


Gyro

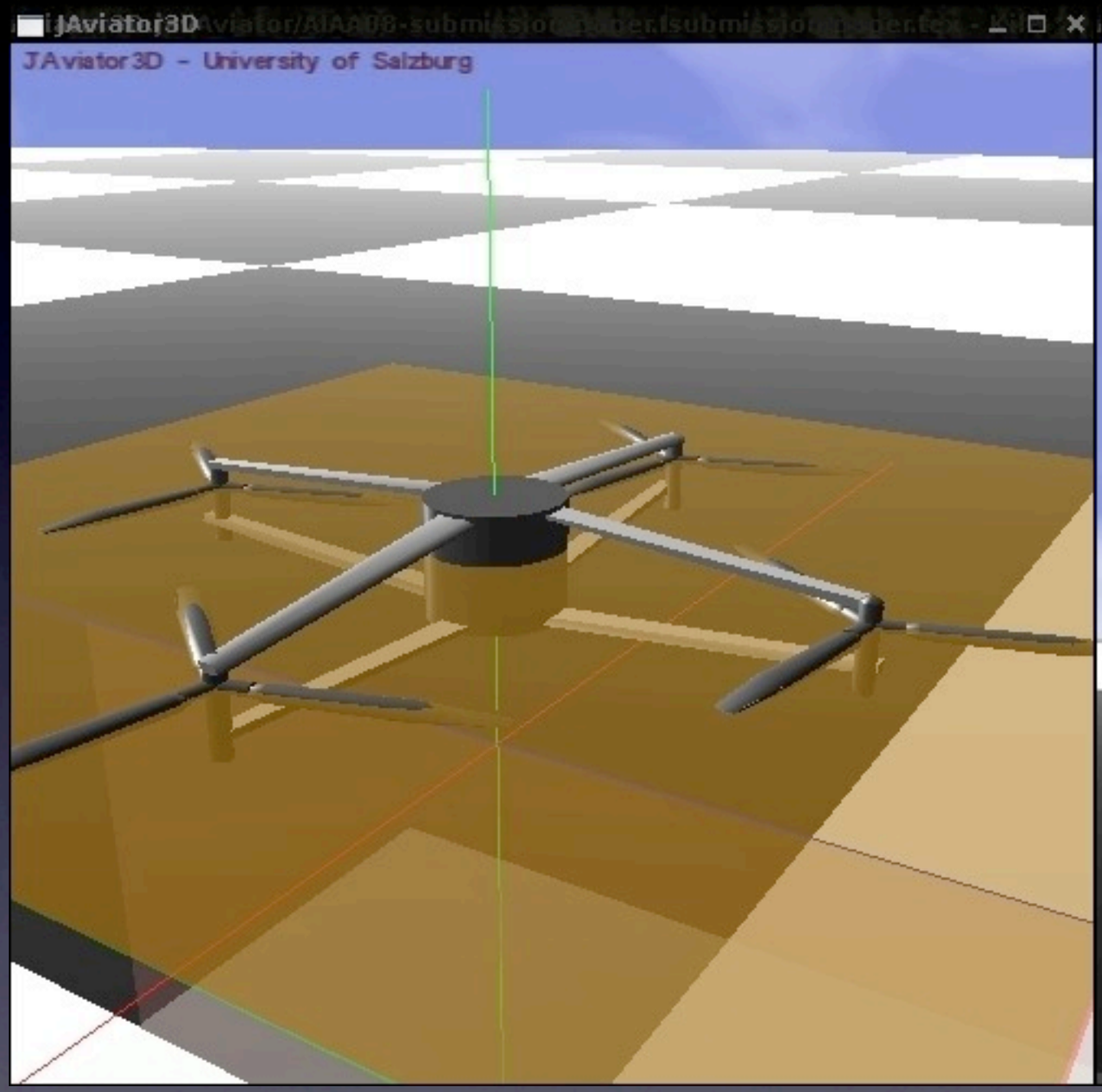
Propulsion

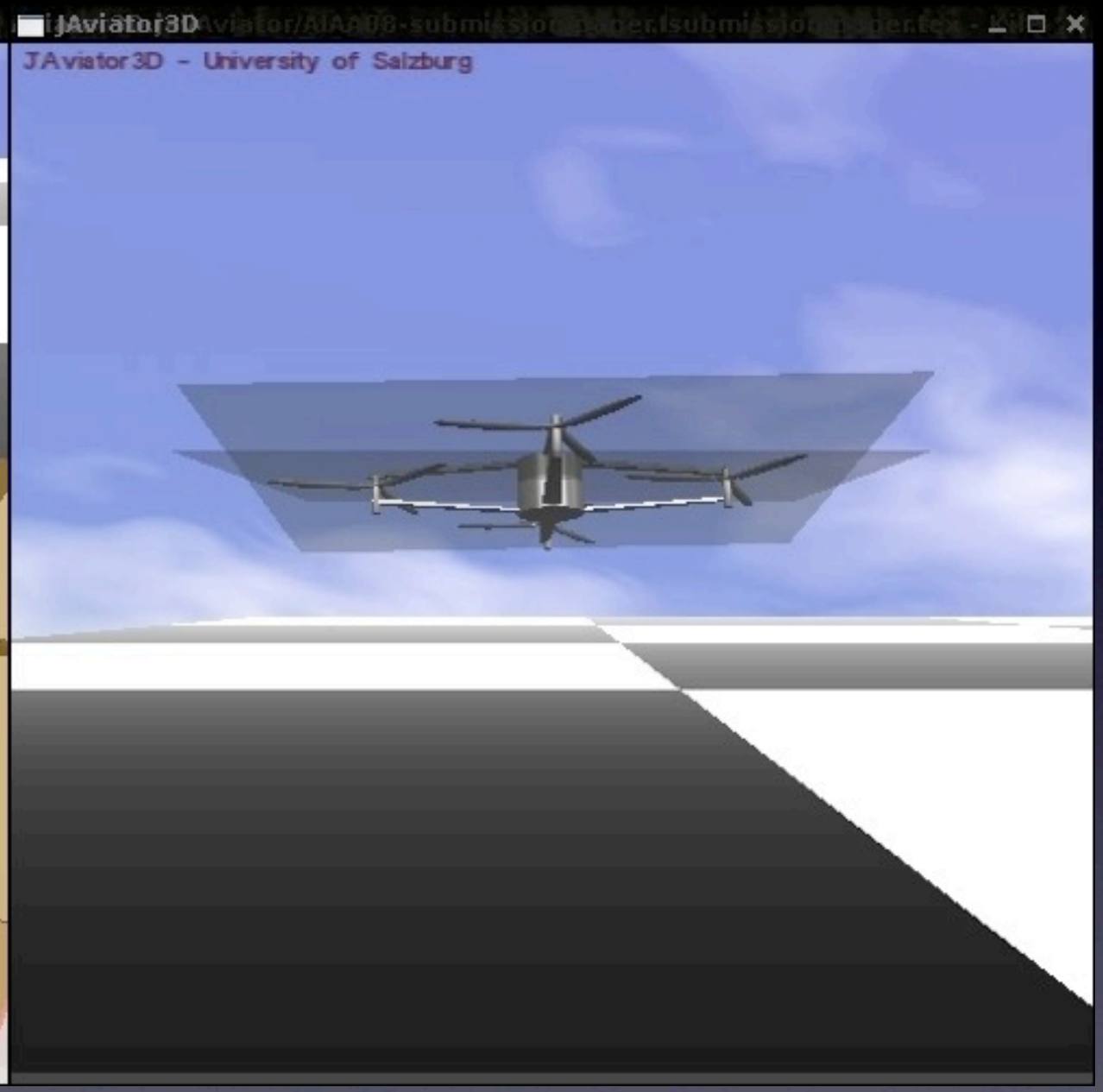


Gumstix



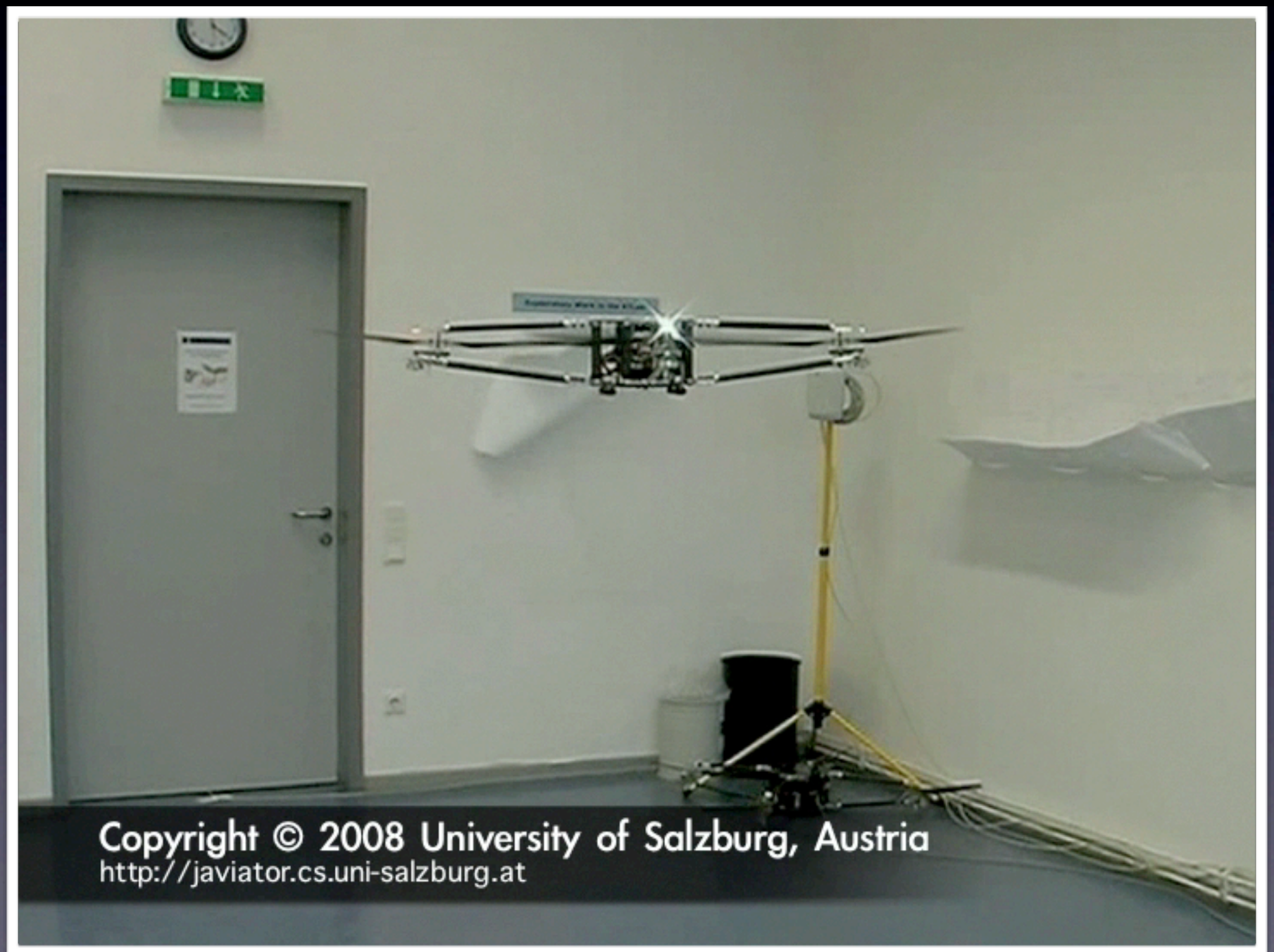
600MHz XScale, 128MB RAM, WLAN, Atmega uController





Indoor Flight
STARMAC Controller

Indoor Flight
STARMAC Controller



Outdoor Flight
STARMAC Controller

Outdoor Flight
STARMAC Controller

Outdoor Flight
Salzburg Controller

Outdoor Flight
Salzburg Controller

What's next?

- Autonomous single-vehicle flights
  - position controller
  - waypoint controller

What's next?

- Autonomous single-vehicle flights
  - position controller
  - waypoint controller
- Autonomous multi-vehicle flights
  - mission controller

javiator.cs.uni-salzburg.at

Salzburg Soft Walls Controller on JJ

Salzburg Soft Walls Controller on JJ