the gamedesigninitiative at cornell university

Lecture 12

Memory Management

Gaming Memory (Current Generation)

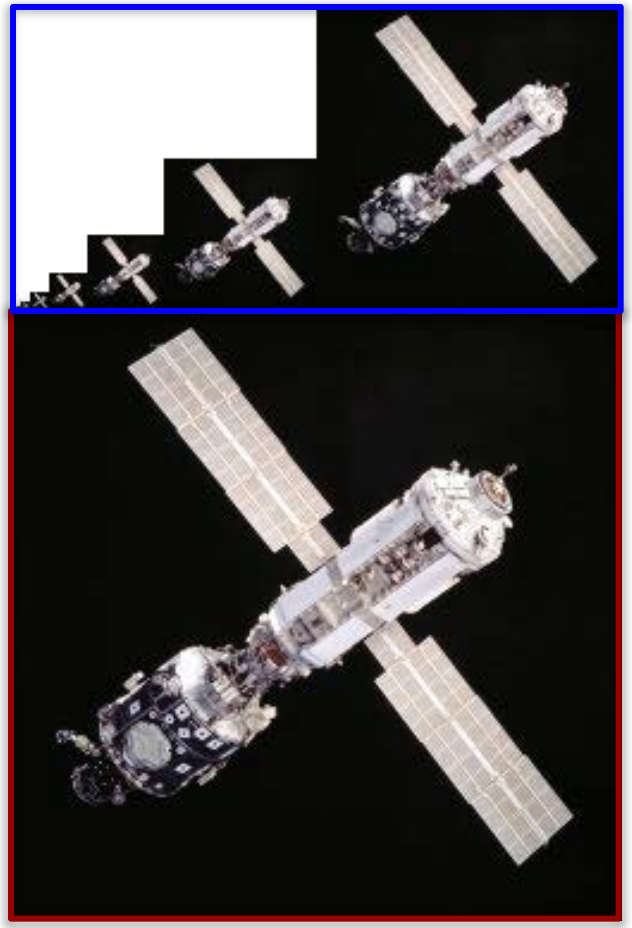
- **Playstation 4**
  - 8 GB RAM (unified)
- **X-Box One (X)**
  - 12 GB RAM (unified)
  - 9 GB for games
- **Nintendo Switch**
  - 3 GB RAM (unified)
  - 1 GB only for OS
- **iPhone/iPad**
  - 2 GB RAM (unified)
  - Better than an XBox 360

Memory Usage: Images

- Pixel color is 4 bytes
  - 1 byte each for r, b, g, alpha
  - More if using HDR color
- Image a **2D array** of pixels
  - 1280x1024 monitor size
  - 5,242,880 bytes ~ 5 MB
- More if using **mipmaps**
  - Graphic card texture feature
  - Smaller versions of image
  - Cached for performance
  - But can double memory use

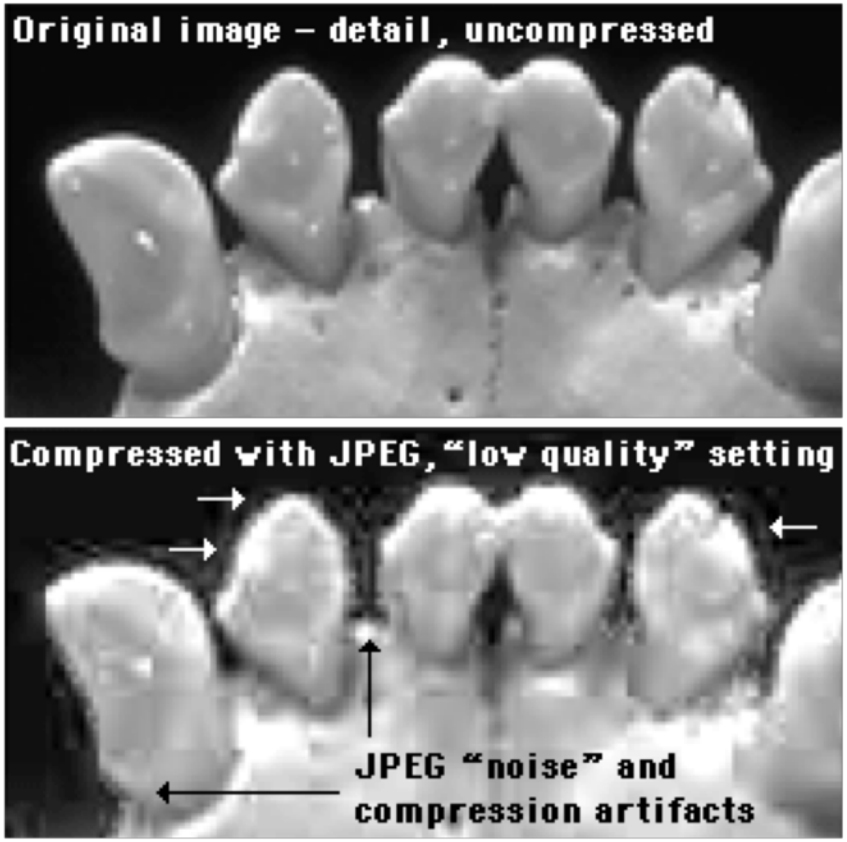
MipMaps

Original Image



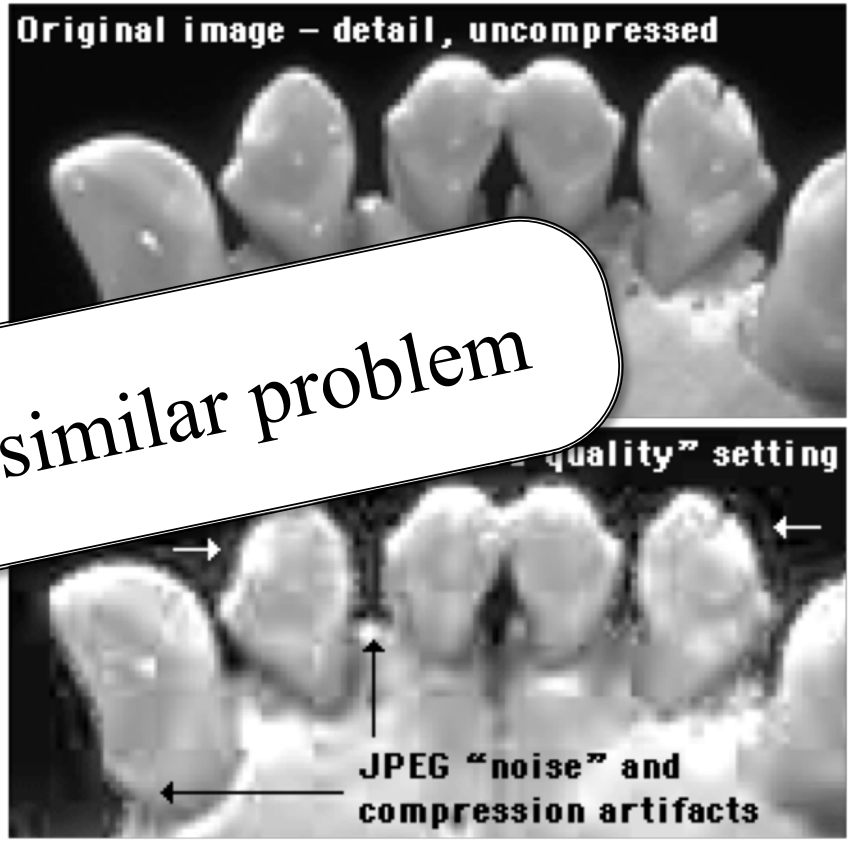
But My JPEG is only 8 KB!

- Formats often **compressed**
  - JPEG, PNG, GIF
  - But not always TIFF
- Must **uncompress** to show
  - Need space to uncompress
  - In RAM or graphics card
- Only load when needed
  - Loading is primary I/O operation in AAA games
  - Causes “texture popping”



But My JPEG is only 8 KB!

- Formats often **compressed**
  - JPEG, PNG, GIF
  - But not always TIFF
- Must **uncompress** to show
  - Need space to uncom
  - In RA
- Only lo
  - Loading is primary I/O operation in AAA games
  - Causes "texture popping"

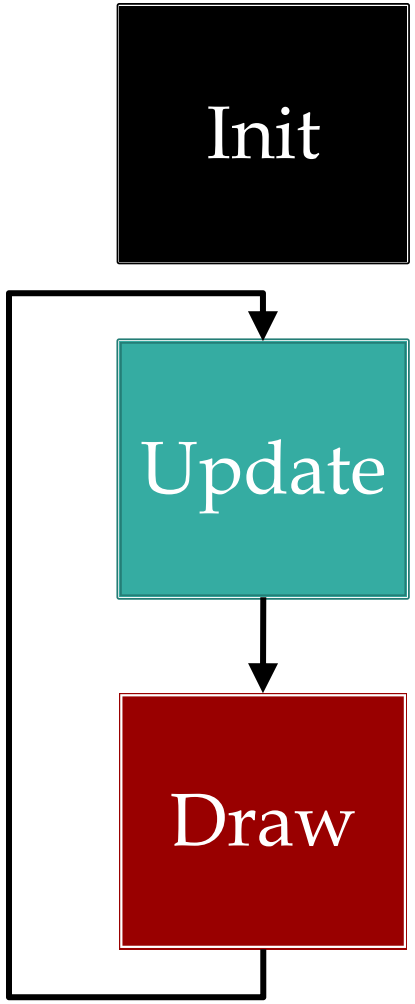


Sounds have a similar problem

Loading Screens

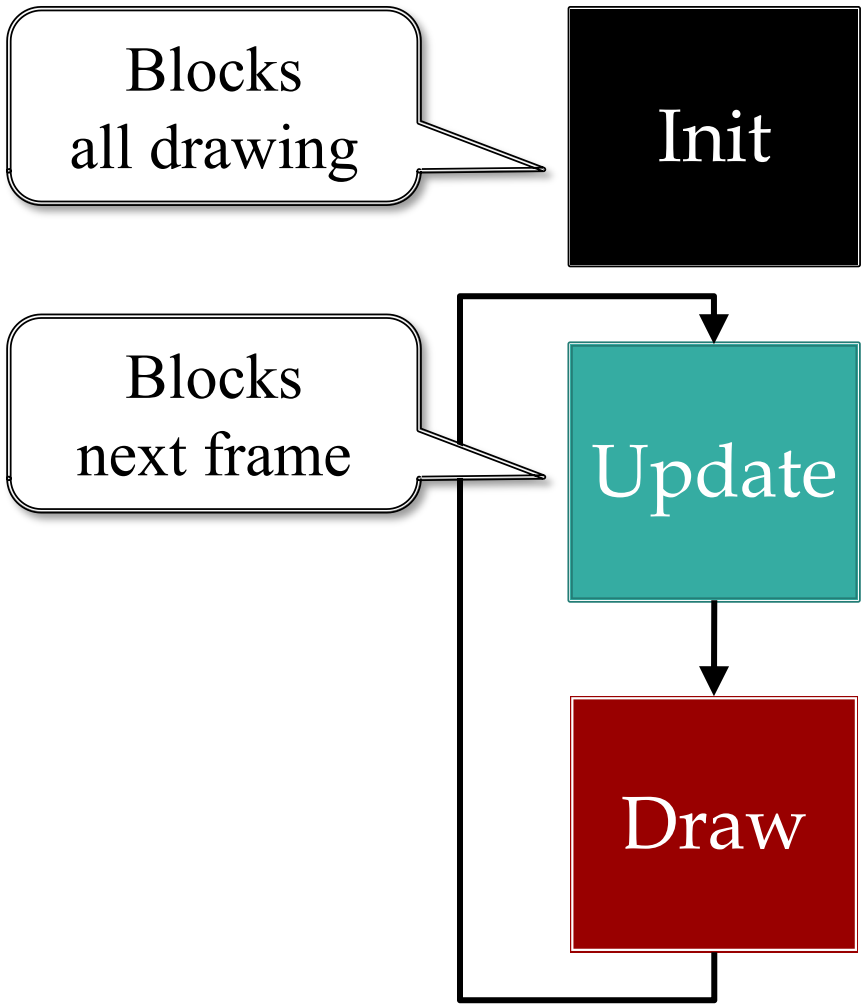
Problems with Asset Loading

- How to load assets?
  - May have a lot of assets
  - May have large assets
- Loading is **blocking**
  - Game stops until done
  - Cannot draw or animate
- May need to **unload**
  - Running out of memory
  - Free something first



Problems with Asset Loading

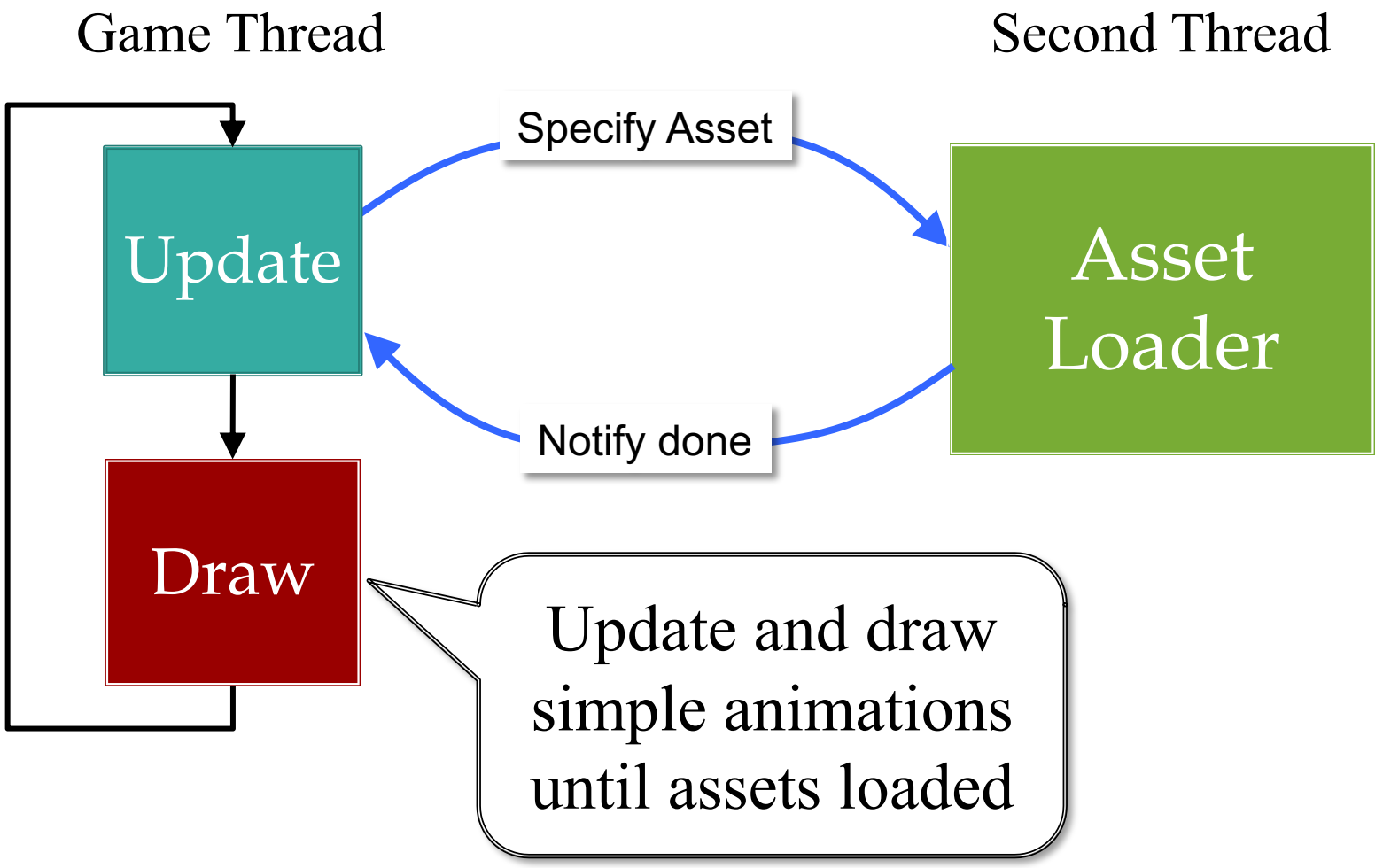
- How to load assets?
  - May have a lot of assets
  - May have large assets
- Loading is **blocking**
  - Game stops until done
  - Cannot draw or animate
- May need to **unload**
  - Running out of memory
  - Free something first



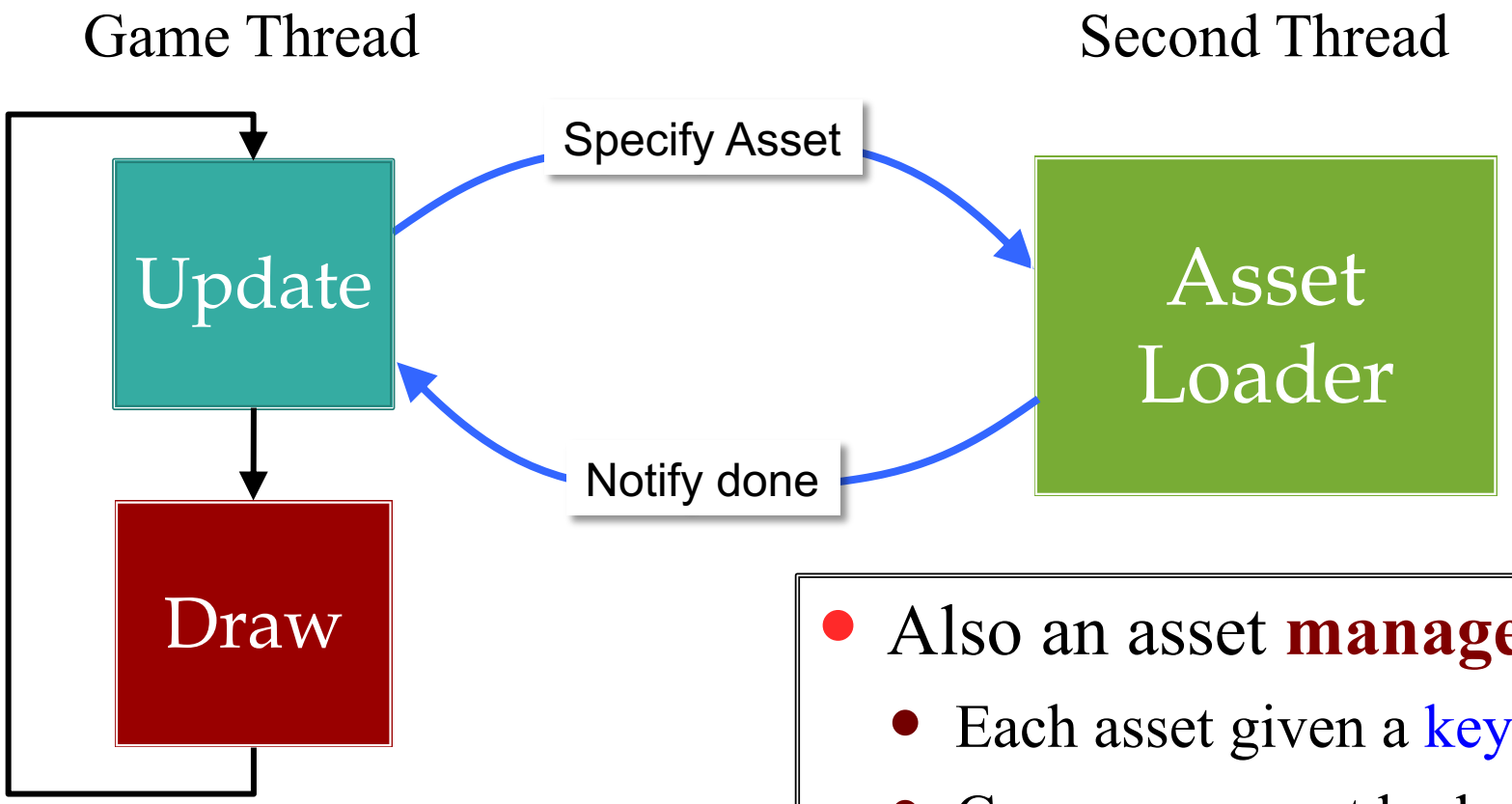
Loading Screens

Minimal animation/feedback while loading assets

Solution: Asynchronous Loader

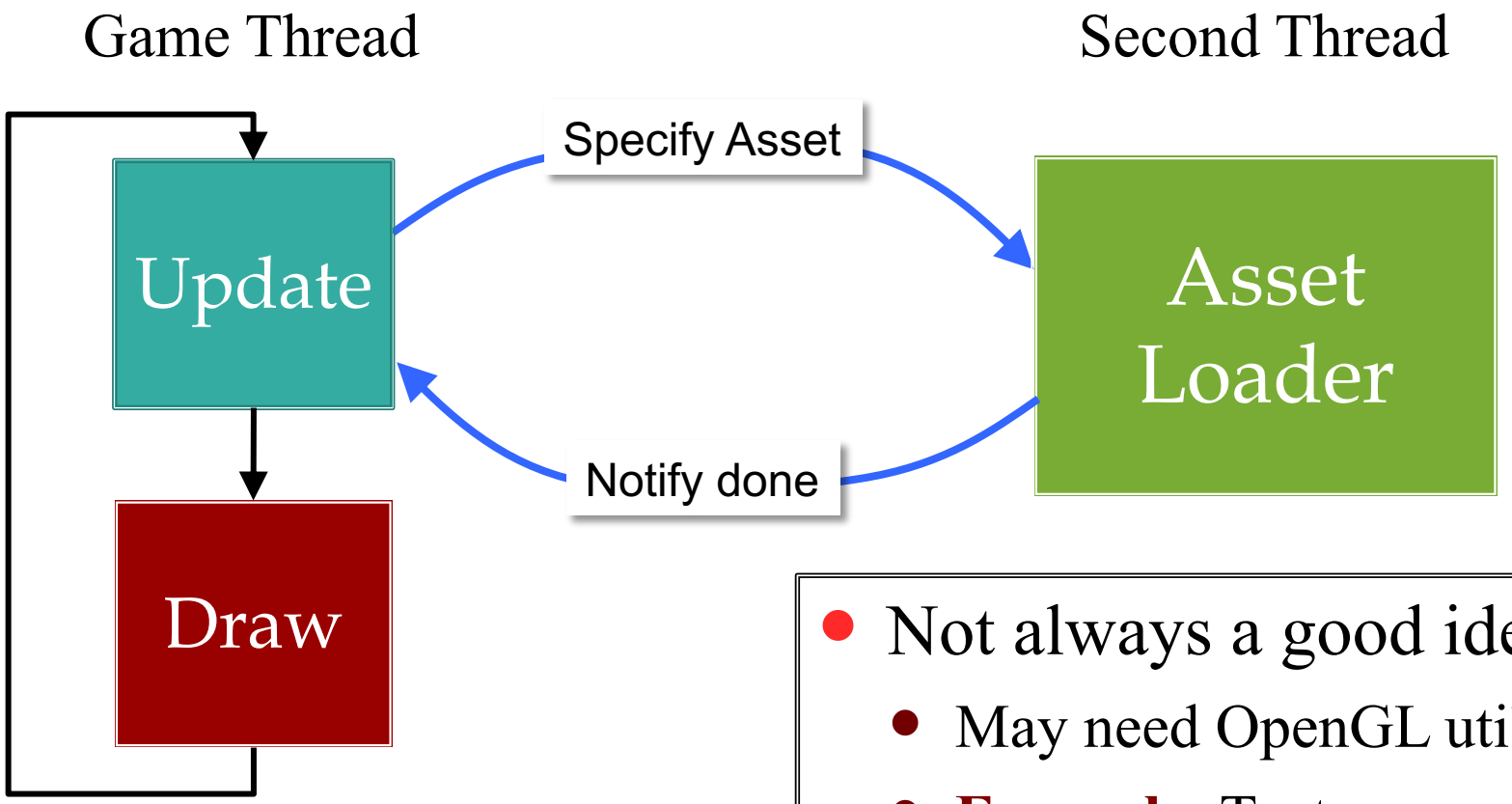


Solution: Asynchronous Loader



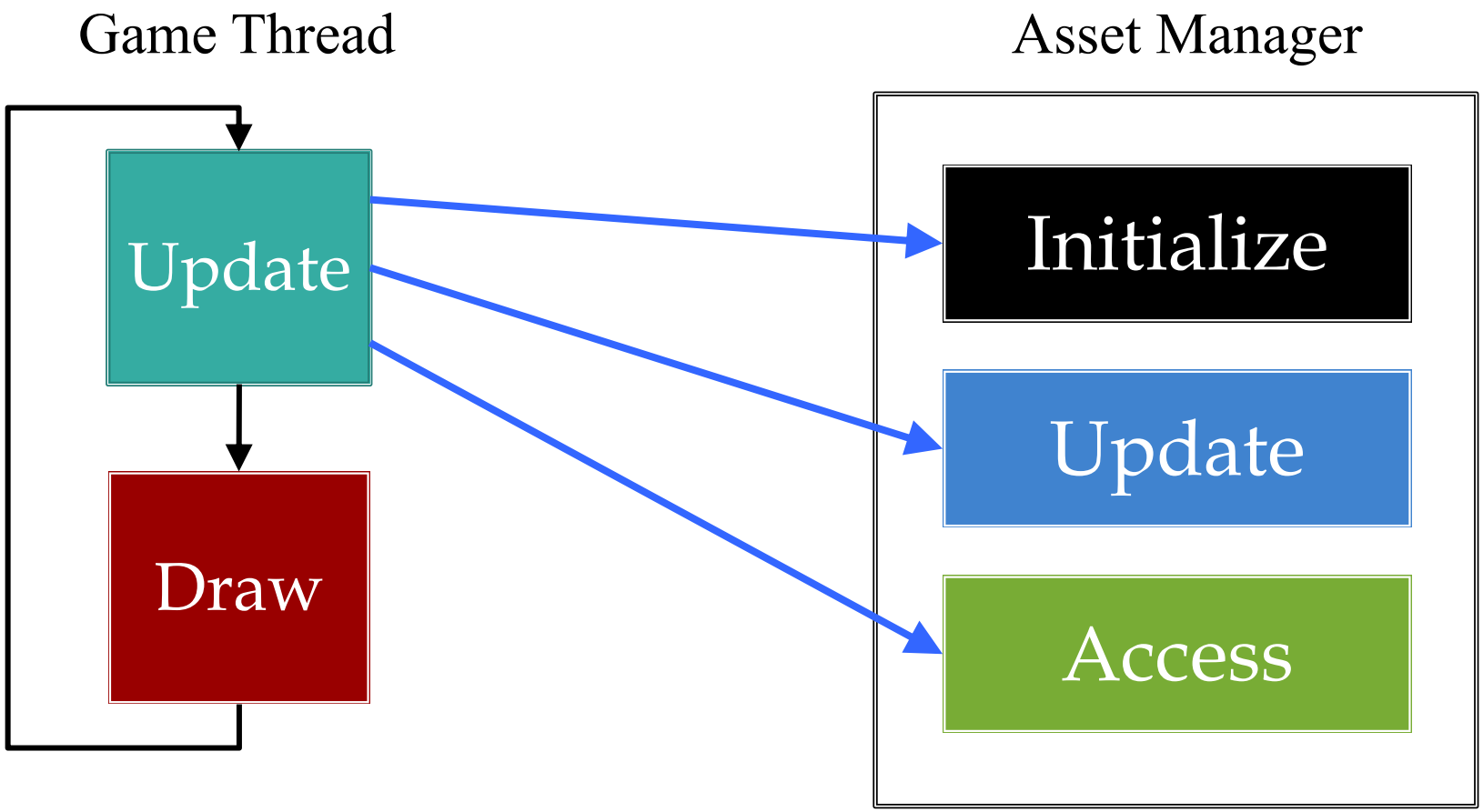
- Also an asset **manager**
  - Each asset given a key
  - Can access asset by key
  - Works like hash table

Solution: Asynchronous Loader



- Not always a good idea
  - May need OpenGL utils
  - **Example**: Textures
  - Limited to main thread

Alternative: Iterative Loader

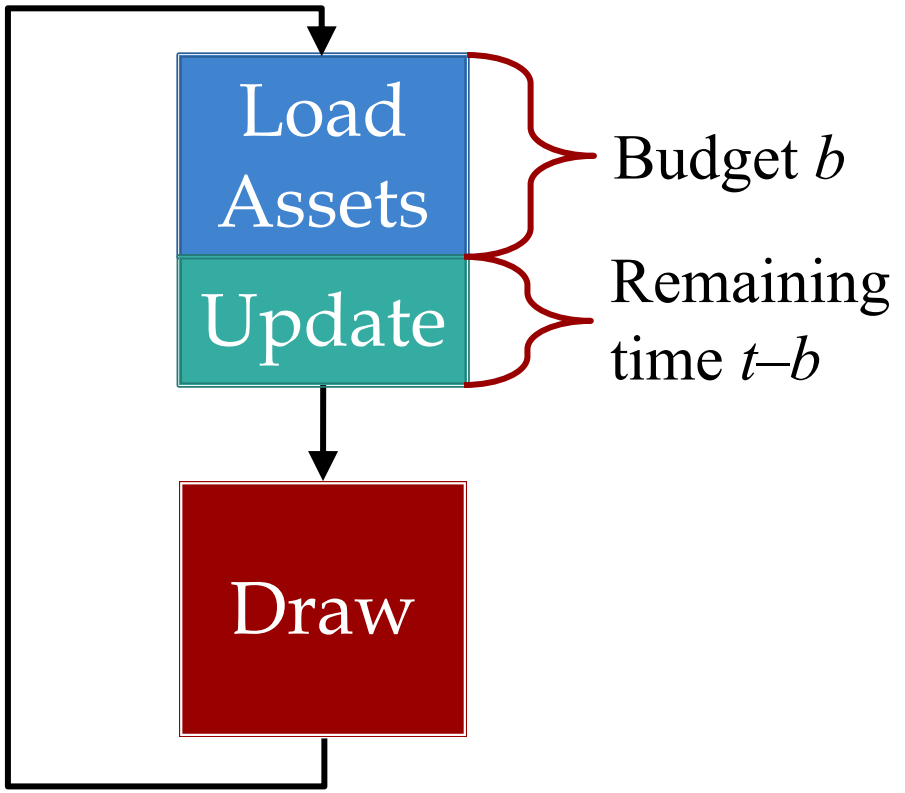


Alternative: Iterative Loader

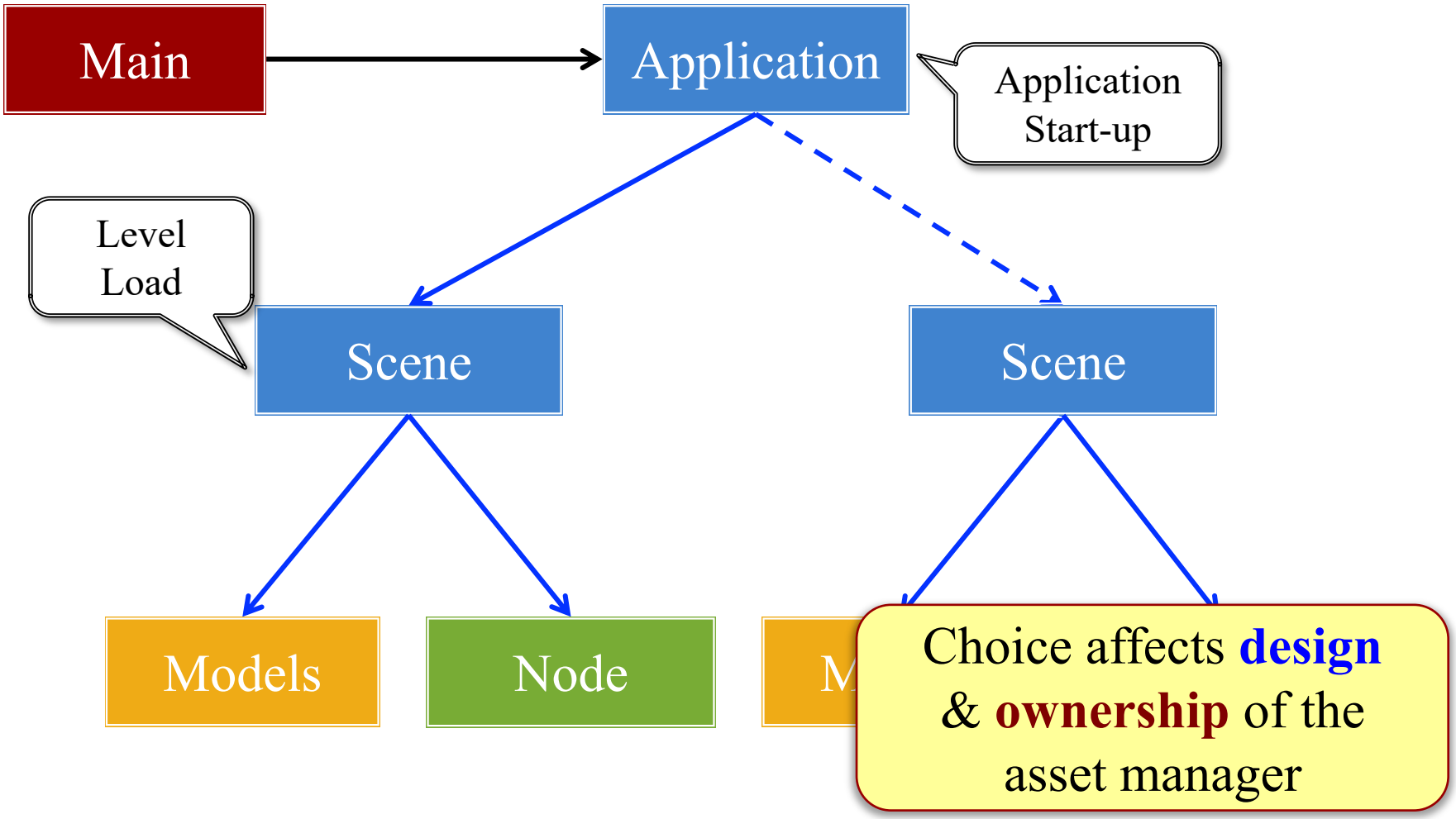
- Uses a time budget
  - Give set amount of time
  - Do as much as possible
  - Stop until next update
- Better for OpenGL
  - Give time to manager
  - Animate with remainder
  - No resource contention
- LibGDX approach
  - CUGL is asynchronous

Alternative: Iterative Loader

- Uses a time budget
  - Give set amount of time
  - Do as much as possible
  - Stop until next update
- Better for OpenGL
  - Give time to manager
  - Animate with remainder
  - No resource contention
- LibGDX approach
  - CUGL is asynchronous



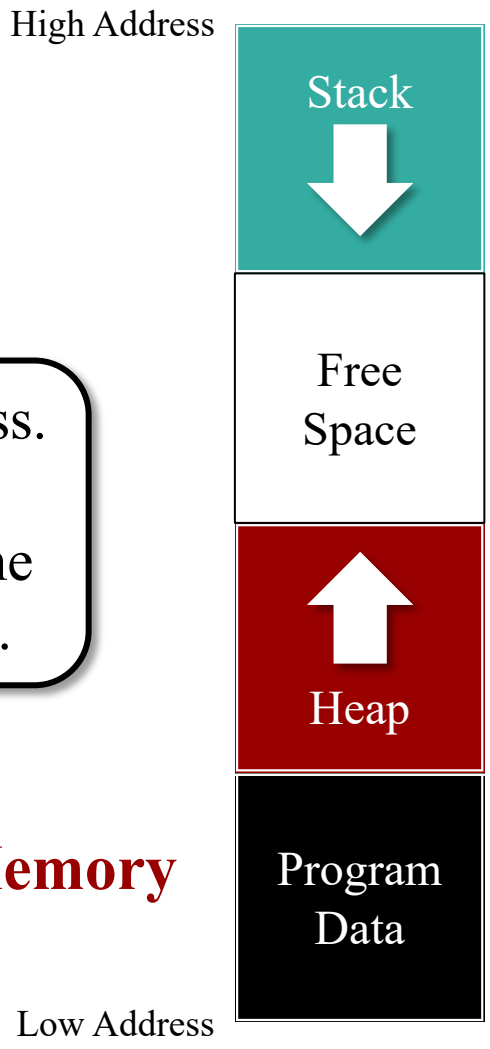
Aside: When Do We Load Assets?



Assets Beyond Images

- AAA games have a lot of 3D geometry
  - Vertices for model polygons
  - Physics bodies **per polygon**
  - Scene graphs for organizing this data
- When are all these objects created?
  - At load time (filling up memory)?
  - Or only when they are needed?
- We need to understand memory better

Traditional Memory Organization



Function parameters
Local variables
Return values

Objects created via `new`
Allocations with `malloc`

Program Code
Static Variables

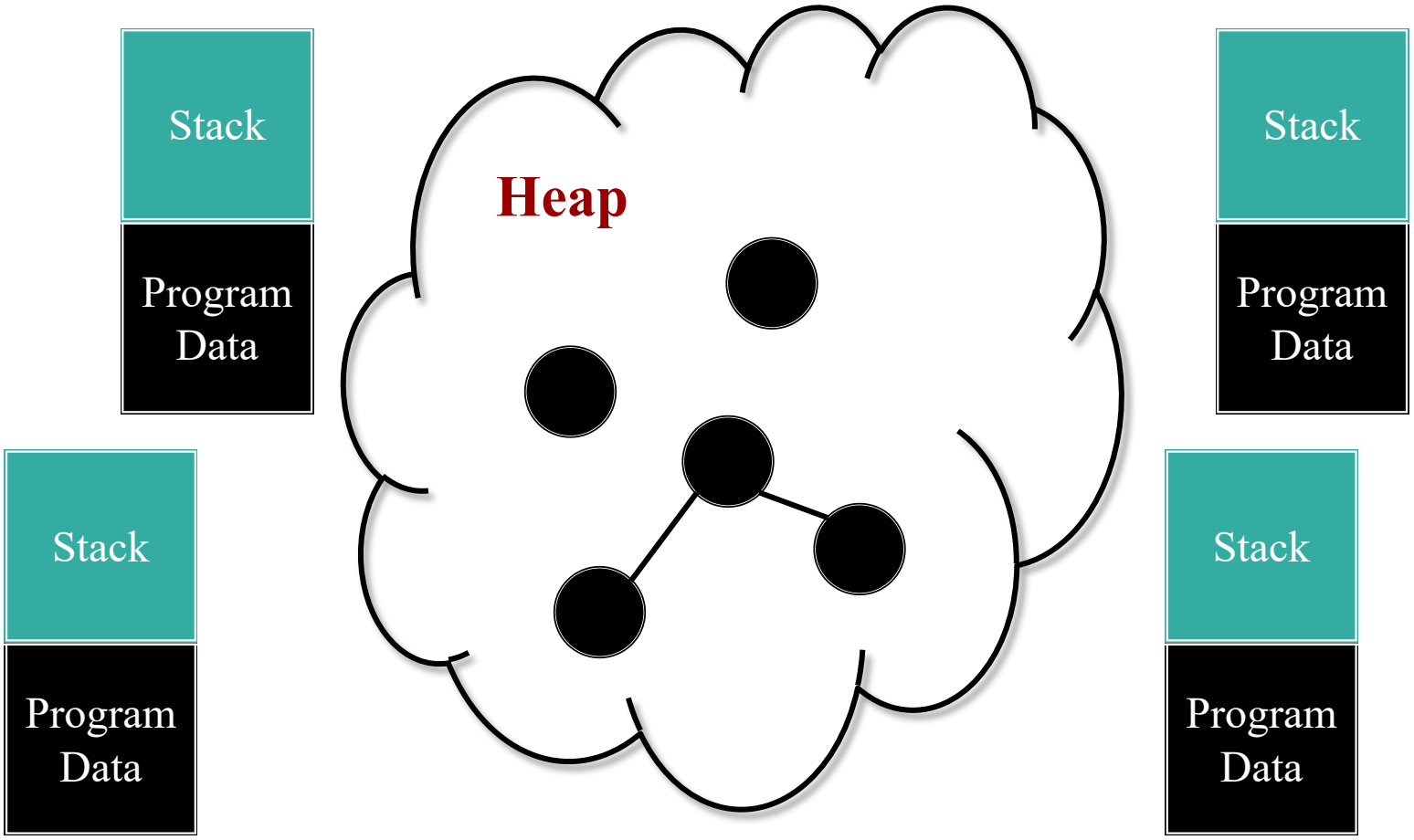
Dedicated to process.

Consists of machine addressable space.

Leverages Virtual Memory

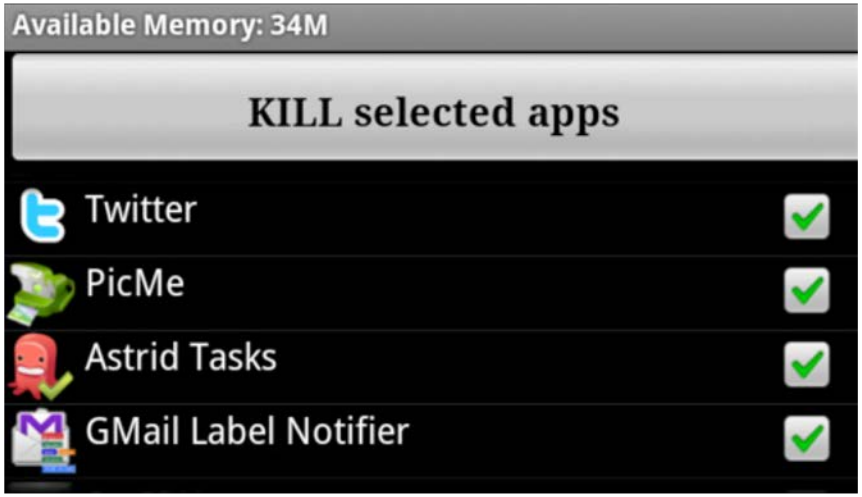
Mobile Memory Organization

Device Memory



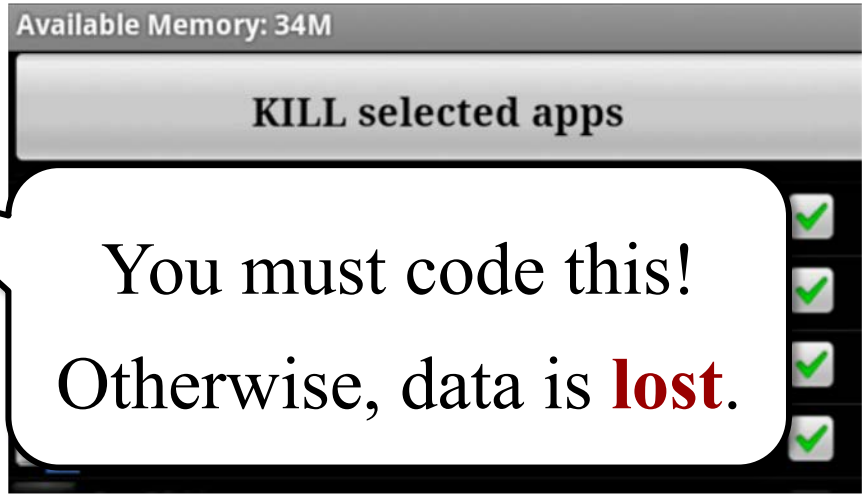
How Do Apps Compete for Memory?

- Active app takes what it can
  - Cannot steal from OS
  - OS may *suspend* apps
- **App Suspension**
  - App quits; memory freed
  - Done only as needed
- Suspend apps can *recover*
  - OS allows limited paging
  - Page out on suspension
  - Page back in on restart

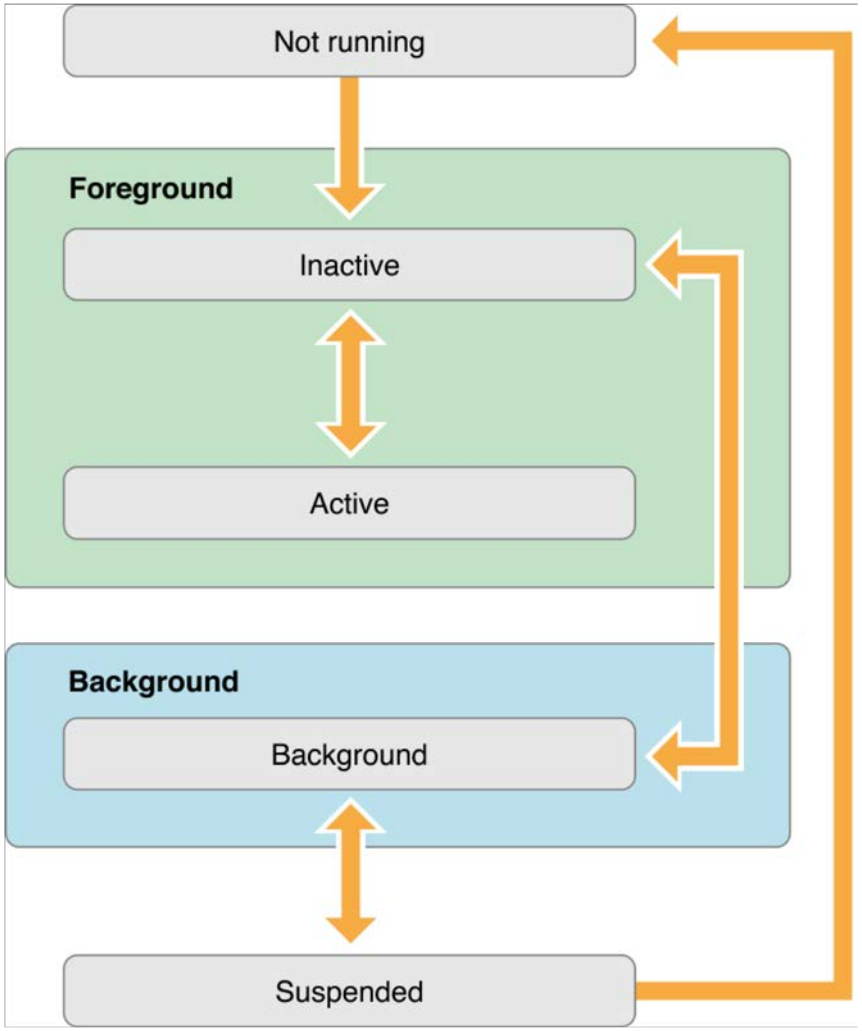


How Do Apps Compete for Memory?

- Active app takes what it can
  - Cannot steal from OS
  - OS may *suspend* apps
- **App Suspension**
  - App quits; memory freed
  - Done only as needed
- Suspend apps can *recover*
  - OS allows limited paging
  - Page out on suspension
  - Page back in on restart

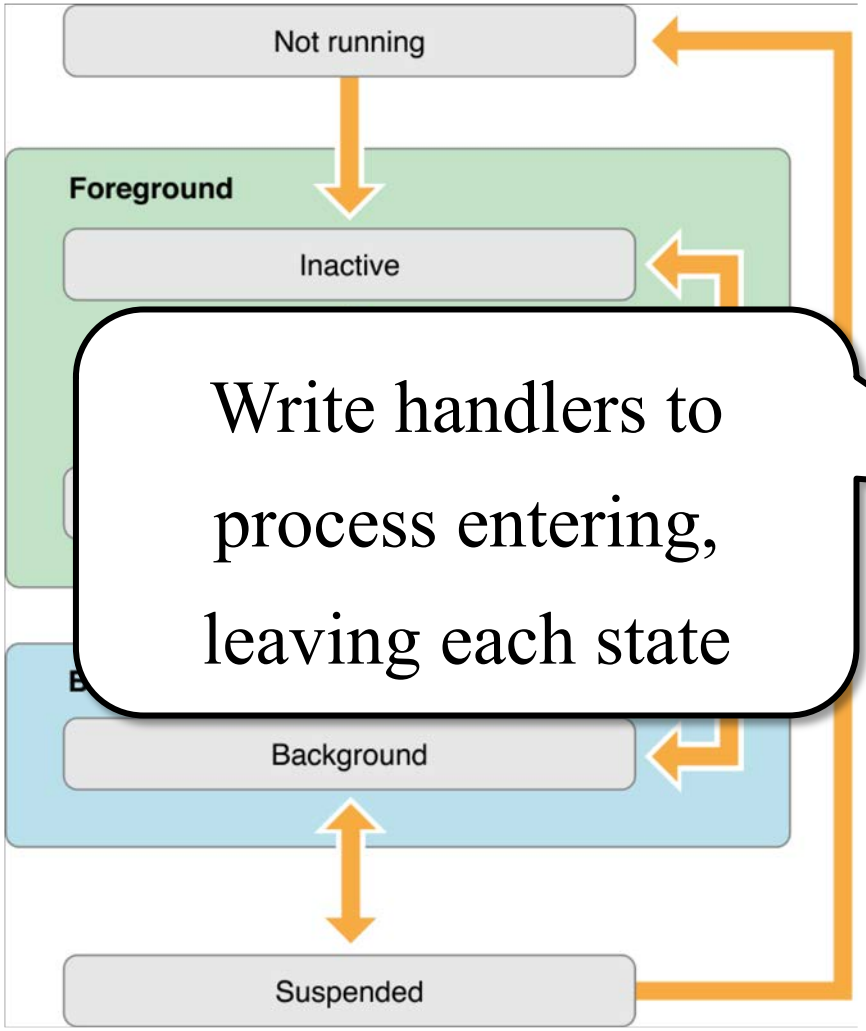


State Management in iOS 7+



- **Active**
  - Running & getting input
- **Inactive**
  - Running, but no input
  - Transition to suspended
- **Background**
  - Same as inactive
  - But apps can stay here
  - **Example**: Music
- **Suspended**
  - Stopped & Memory freed

State Management in iOS 7+

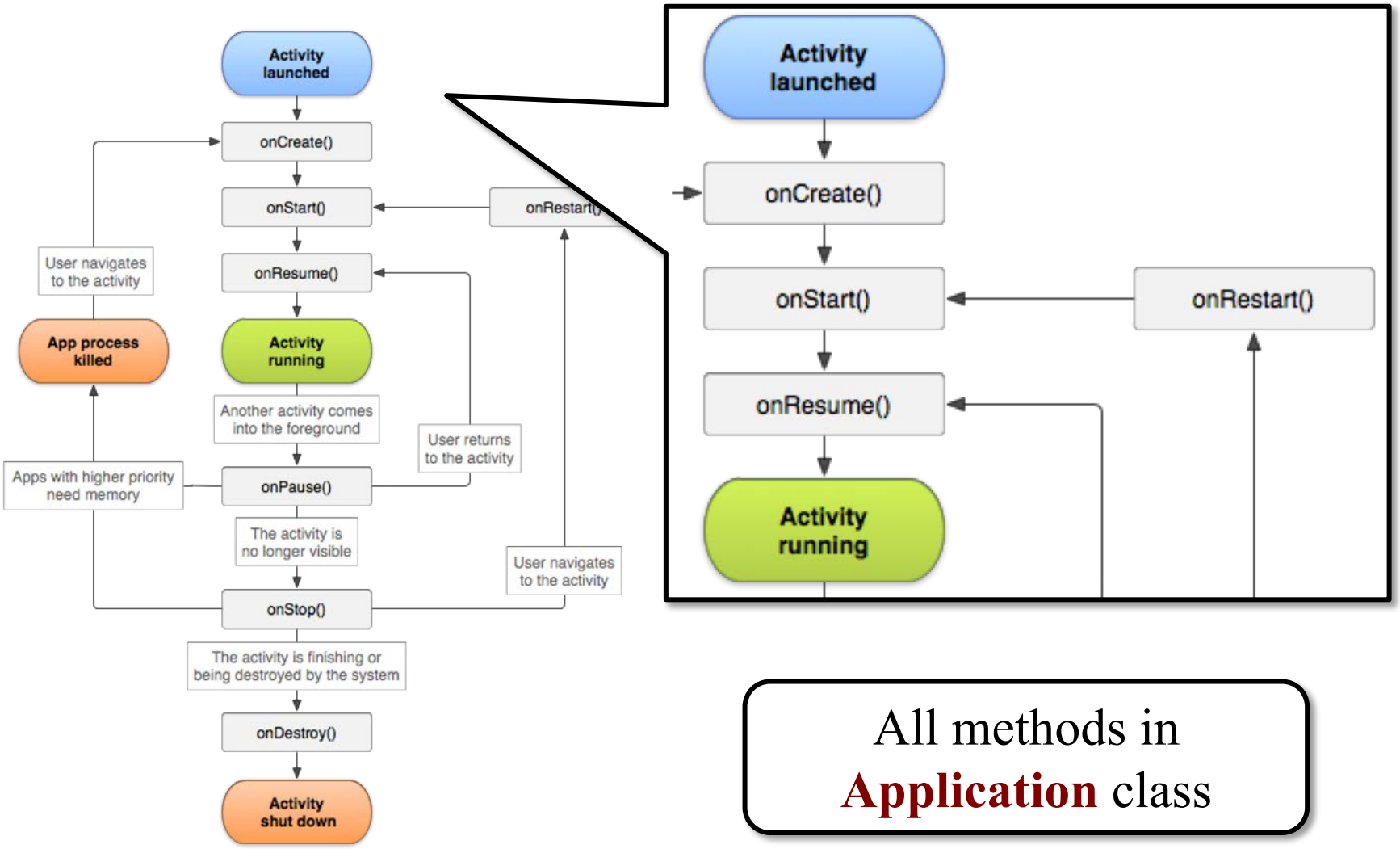


- **Active**
  - Running & getting input
- **Inactive**
  - Running, but no input
  - Transition to suspended
- **Background**
  - Same as inactive
  - But apps can stay here
  - **Example**: Music
- **Suspended**
  - Stopped & Memory freed

iOS State Handling

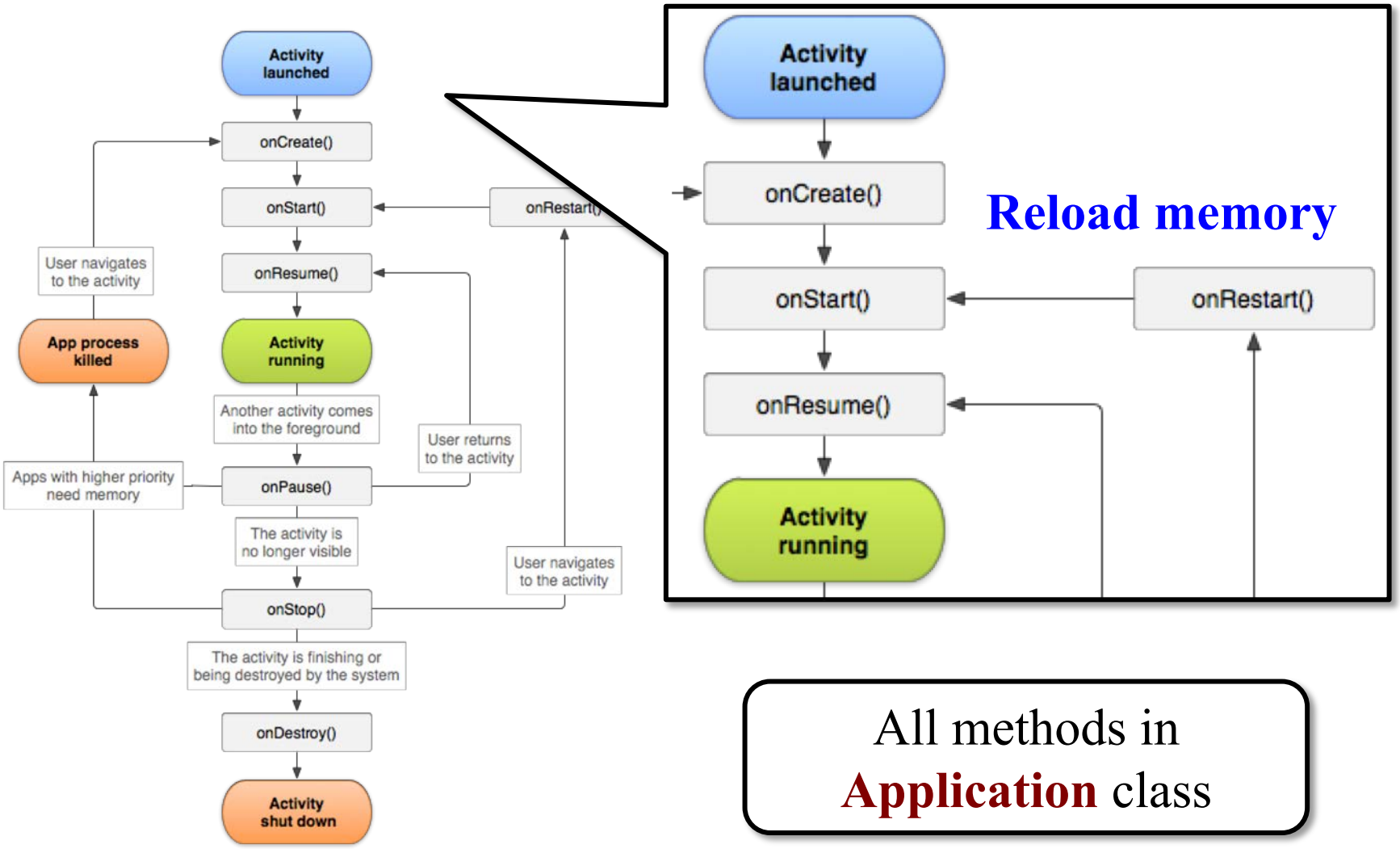
- `applicationDidBecomeActive:`
  - Your app became (resumed as) the foreground app.
  - Use this to recover memory state.
- `applicationWillResignActive:`
  - Your app will switch to inactive or background.
  - Stop the game loop and page out memory.
- `applicationDidEnterBackground:`
  - Your app is in the background and may be suspended.
- `applicationWillEnterForeground:`
  - Your app is leaving the background, but is not yet active.

Android State Handling



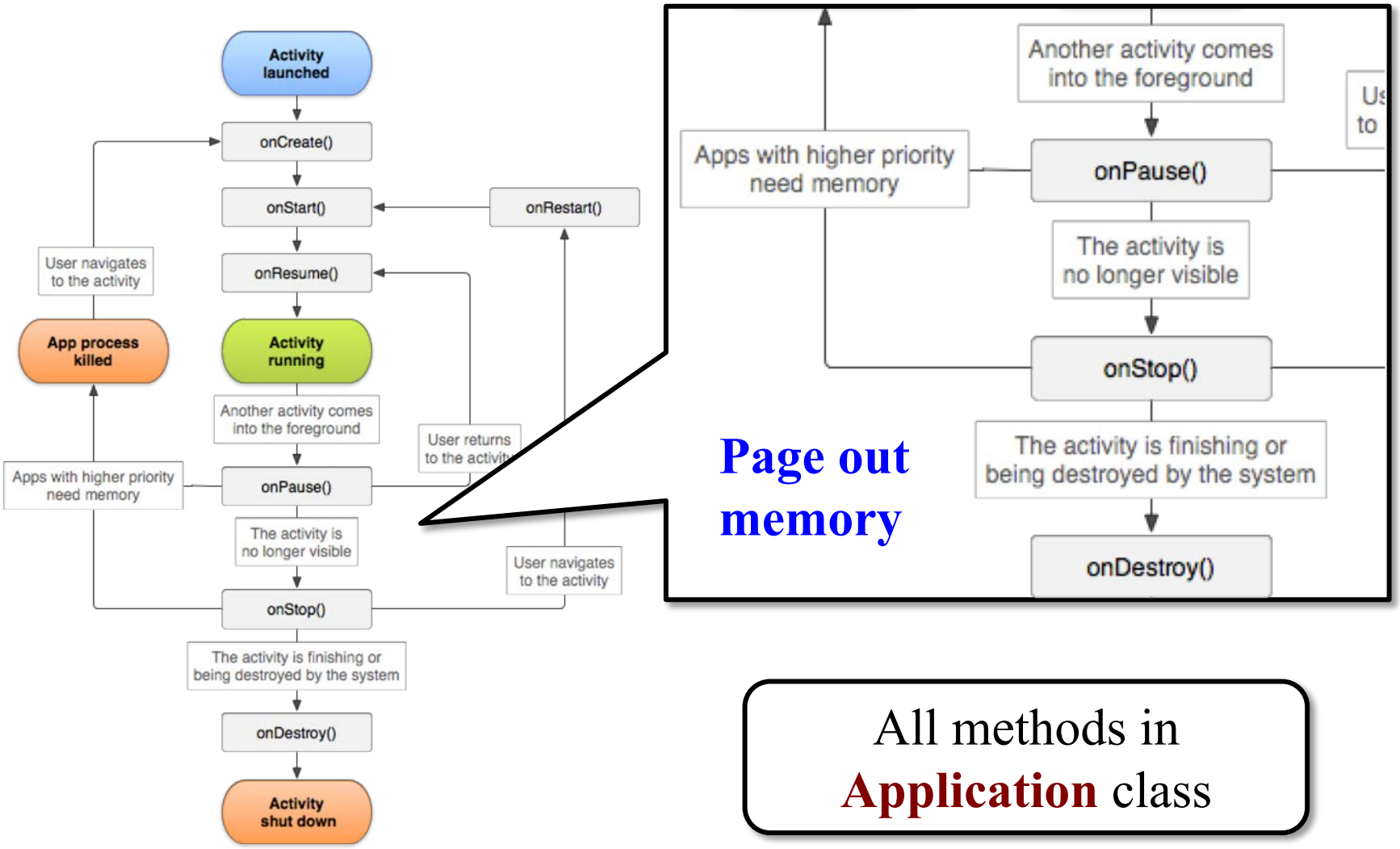
All methods in **Application** class

Android State Handling



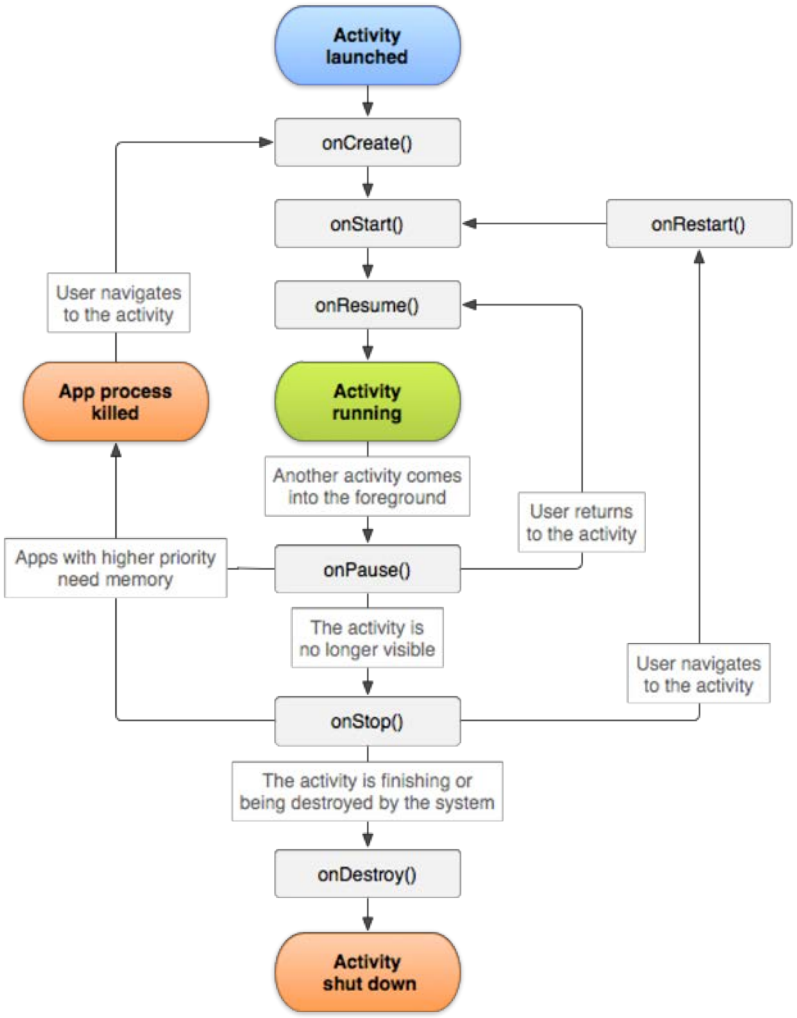
All methods in **Application** class

Android State Handling



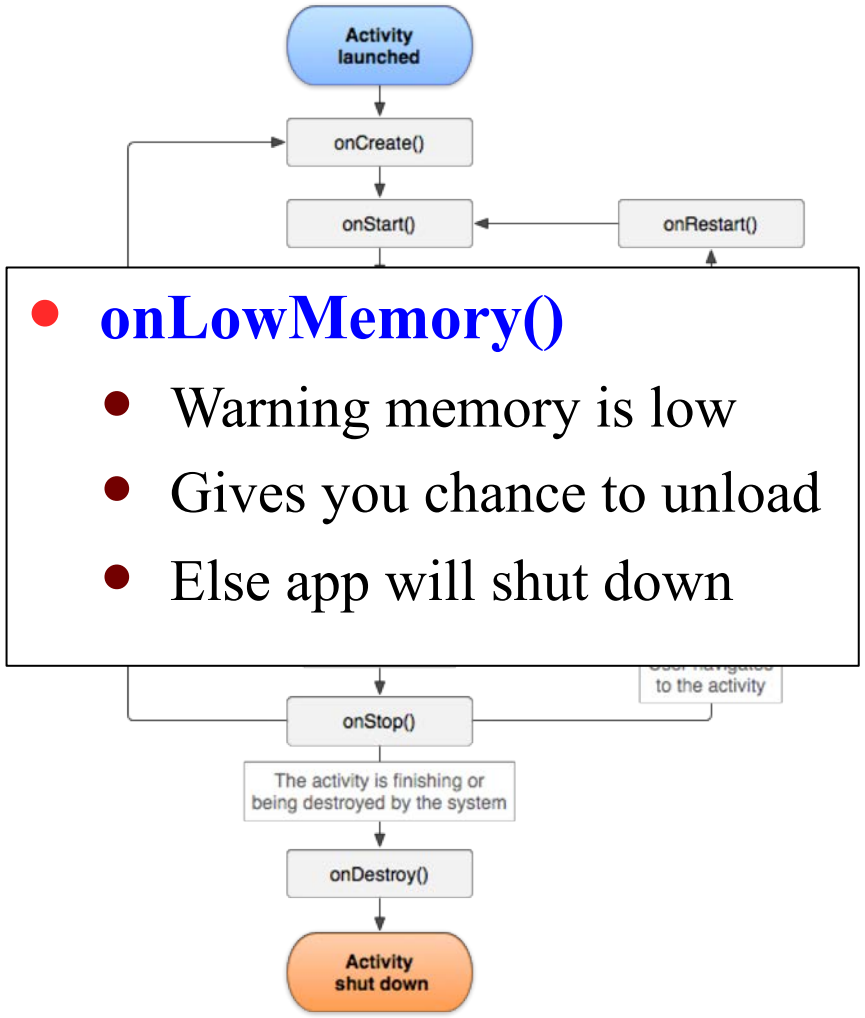
All methods in **Application** class

CUGL is Simplified Android Model



- **onStartup()**
  - Initialized and now active
- **onSuspend()**
  - Sent to background
  - Gives you chance to save
  - Also time to pause music
- **onResume()**
  - Returns to app to active
  - Allows you to restore state
- **onShutdown()**
  - Stopped & memory freed

CUGL is Simplified Android Model



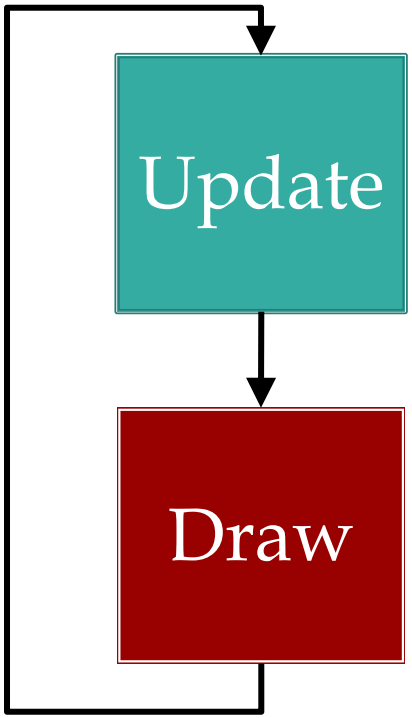
- **onLowMemory()**
  - Warning memory is low
  - Gives you chance to unload
  - Else app will shut down

- **onStartup()**
  - Initialized and now active
- **onSuspend()**
  - Sent to background
  - Gives you chance to save
  - Also time to pause music
- **onResume()**
  - Returns to app to active
  - Allows you to restore state
- **onShutdown()**
  - Stopped & memory freed

Memory Organization and Games

Inter-Frame Memory

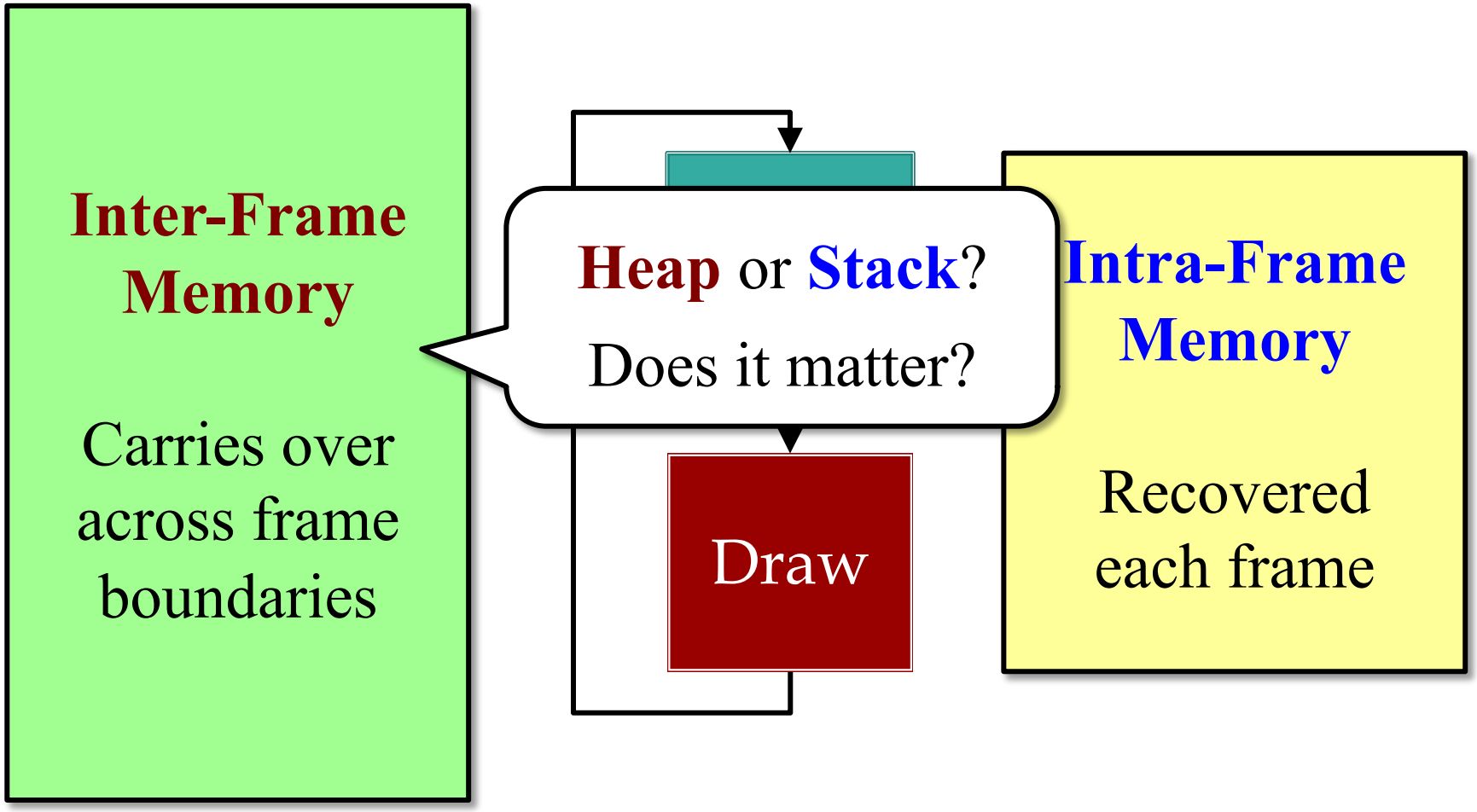
Carries over across frame boundaries



Intra-Frame Memory

Recovered each frame

Memory Organization and Games



Distinguishing Data Types

Intra-Frame

- **Local computation**
  - Local variables (managed by compiler)
  - Temporary objects (not necessarily managed)
- **Transient data structures**
  - Built at the start of update
  - Used to process update
  - Can be deleted at end

Inter-Frame

- **Game state**
  - Model instances
  - Controller state
  - View state and caches
- **Long-term data structures**
  - Built at start/during frame
  - Lasts for multiple frames
  - May adjust to data changes

Distinguishing Data Types

Intra-Frame

- **Local computation**
  - Local variables
  - (not necessarily managed)
- **Transient data structures**
  - Built at the start of update
  - Used to process update
  - Can be deleted at end

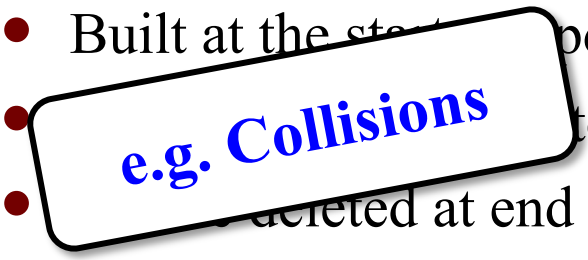
Inter-Frame

- **Game state**
  - Model instances
  - caches
- **Long-term data structures**
  - Built at start/during frame
  - Lasts for multiple frames
  - May adjust to data changes

Distinguishing Data Types

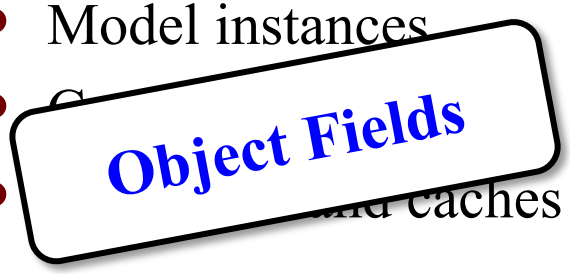
Intra-Frame

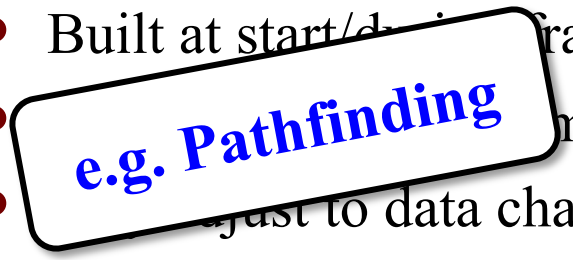
- **Local computation**
  - Local variables
  - (not necessarily managed)
- **Transient data structures**
  - Built at the start of update
  - deleted at end



Inter-Frame

- **Game state**
  - Model instances
  - caches
- **Long-term data structures**
  - Built at start/during frame
  - to data changes





Handling Game Memory

Intra-Frame

- Does not need to be paged
  - Drop the latest frame
  - Restart on frame boundary
- Want size reas
  - Loca
  - Limit allocations
  - Limit `new` inside loops
- Often use **custom allocator**
  - GC at frame boundaries

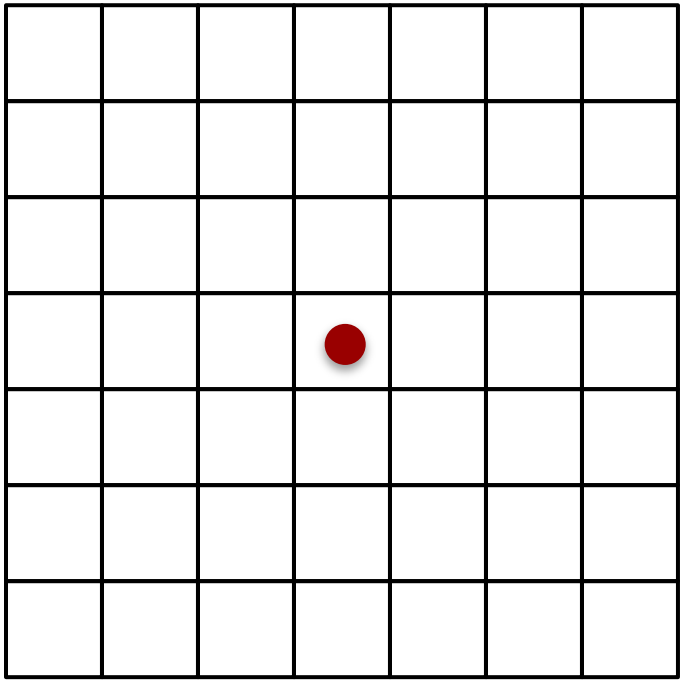
Inter-Frame

- Potential to be paged
  - Defines current game state
  - l start
  - **lexible**
  - No. of objects is variable
  - Subsystems may turn on/off
  - User settings may affect
- **OS allocator** okay, but…
  - Recycle with **free lists**

Topic of Next Lecture

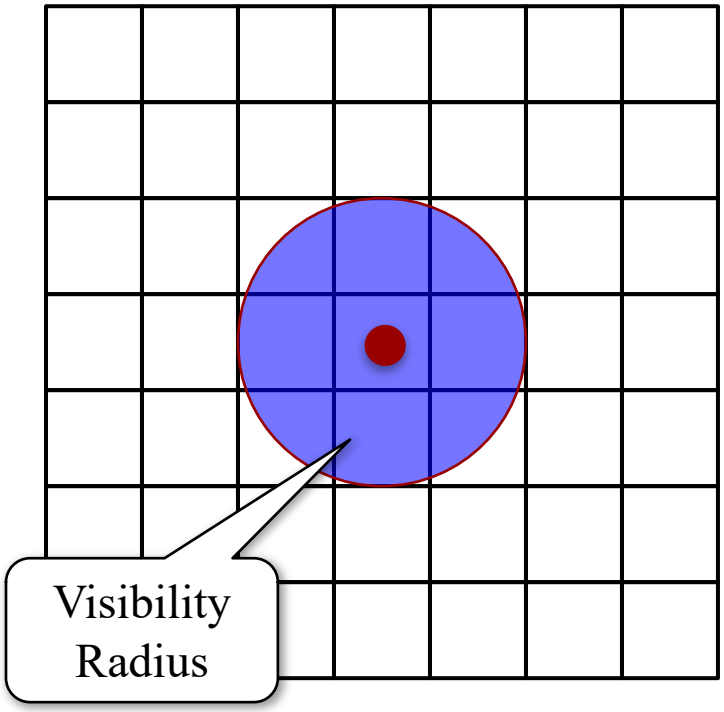
Advanced: Spatial Loading

- Most game data is ***spatial***
  - Only load if player nearby
  - Unload as player moves away
  - Minimizes memory used
- Arrange memory in ***cells***
  - Different from a memory pool
  - Track player visibility radius
  - Load/unload via outer radius
- **Alternative**: loading zones
  - Elevators in *Mass Effect*



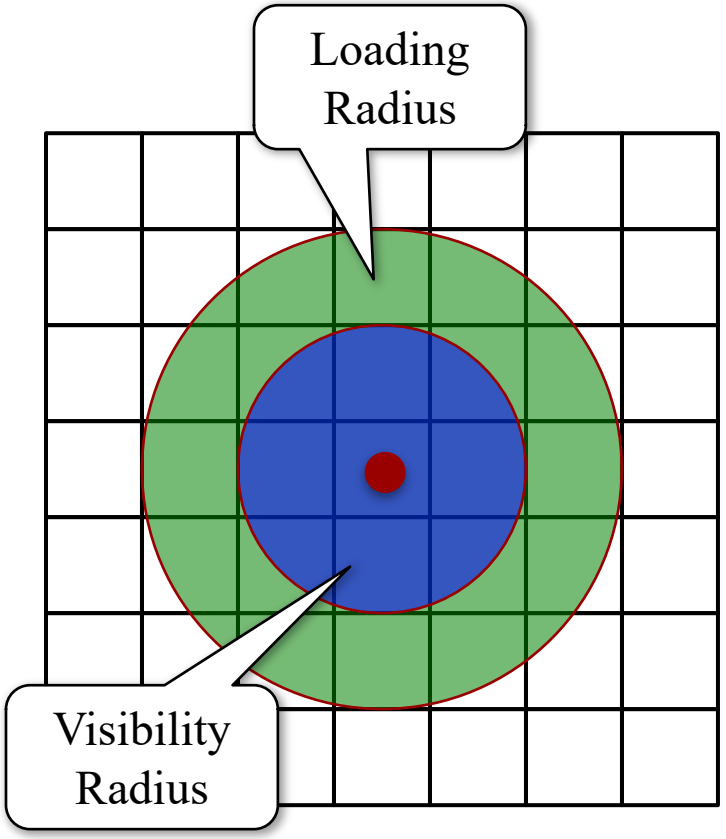
Advanced: Spatial Loading

- Most game data is ***spatial***
  - Only load if player nearby
  - Unload as player moves away
  - Minimizes memory used
- Arrange memory in ***cells***
  - Different from a memory pool
  - Track player visibility radius
  - Load/unload via outer radius
- **Alternative**: loading zones
  - Elevators in *Mass Effect*



Advanced: Spatial Loading

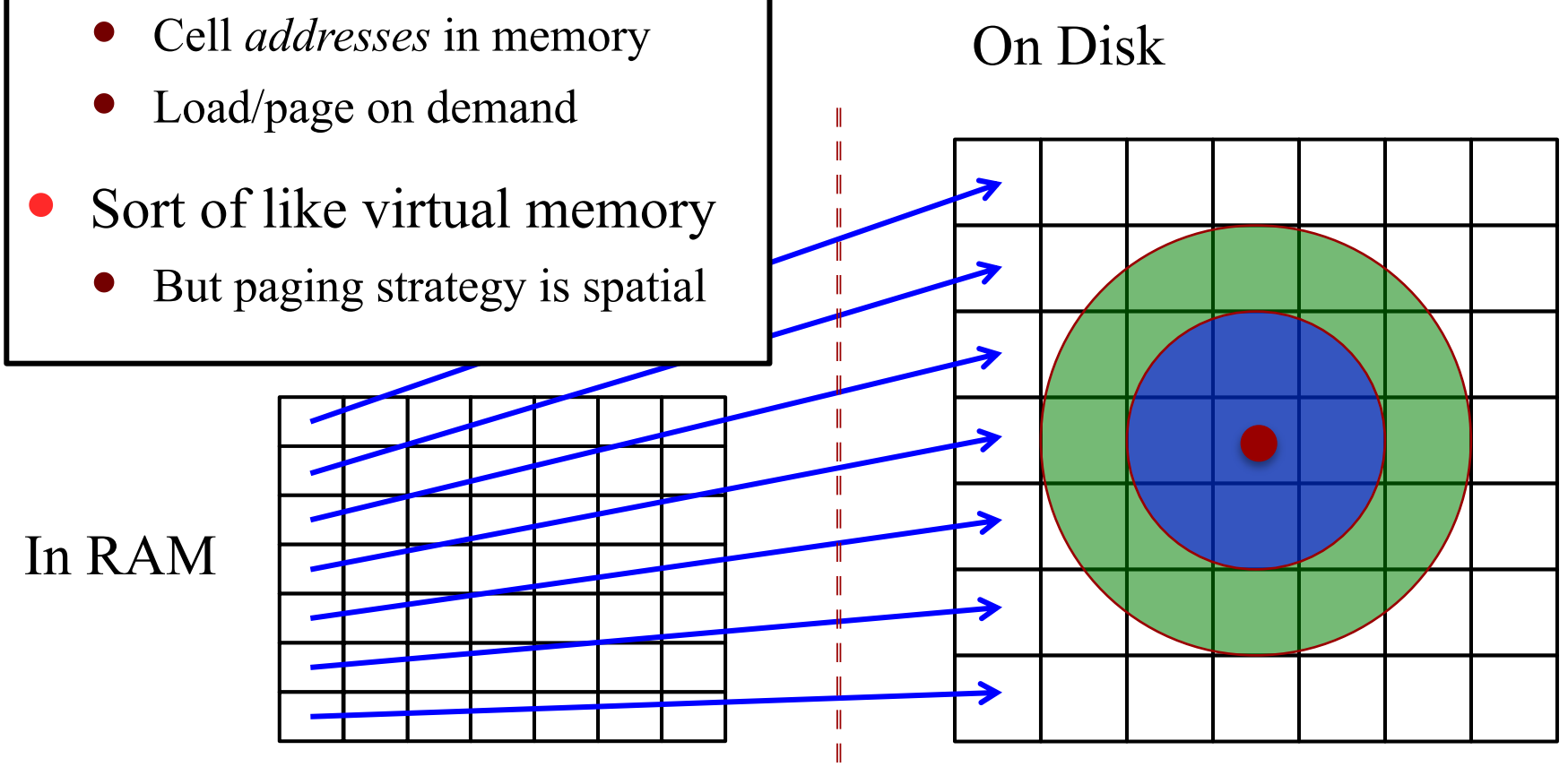
- Most game data is ***spatial***
  - Only load if player nearby
  - Unload as player moves away
  - Minimizes memory used
- Arrange memory in ***cells***
  - Different from a memory pool
  - Track player visibility radius
  - Load/unload via outer radius
- **Alternative**: loading zones
  - Elevators in *Mass Effect*



Spatial Loading in *Assassin's Creed*

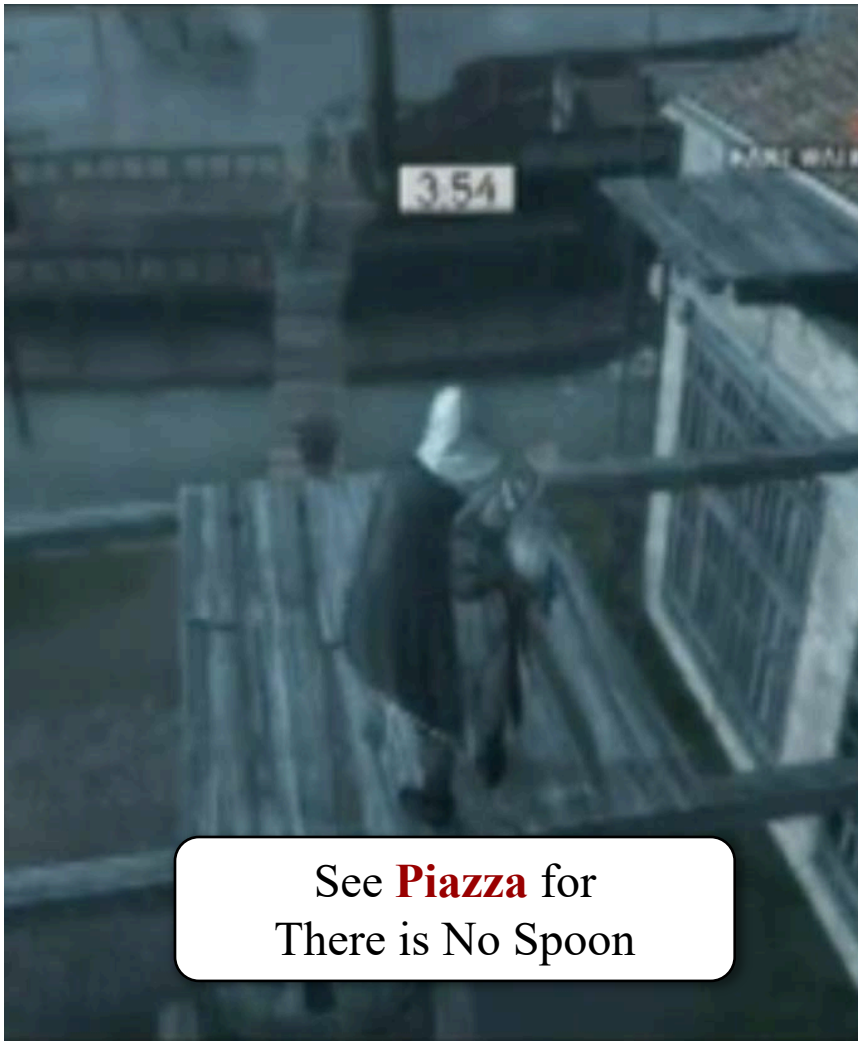
Implementing Spatial Loading

- Part of serialization model
  - Level/save file has the cells
  - Cell *addresses* in memory
  - Load/page on demand
- Sort of like virtual memory
  - But paging strategy is spatial



Spatial Loading Challenges

- **Not same** as virtual memory
  - Objects unloaded do not exist
  - Do not save state when unload
  - Objects loaded are new created
- Can lead to *unexpected states*
  - "Forgetful" NPCs
  - Creative *Assassin's Creed* kills
- **Workaround**: Global State
  - Track major game conditions
  - **Example**: Guards Alerted
  - Use to load objects in standard, but appropriate, configurations



Next Time: Low-Level Details