Data Oriented Design

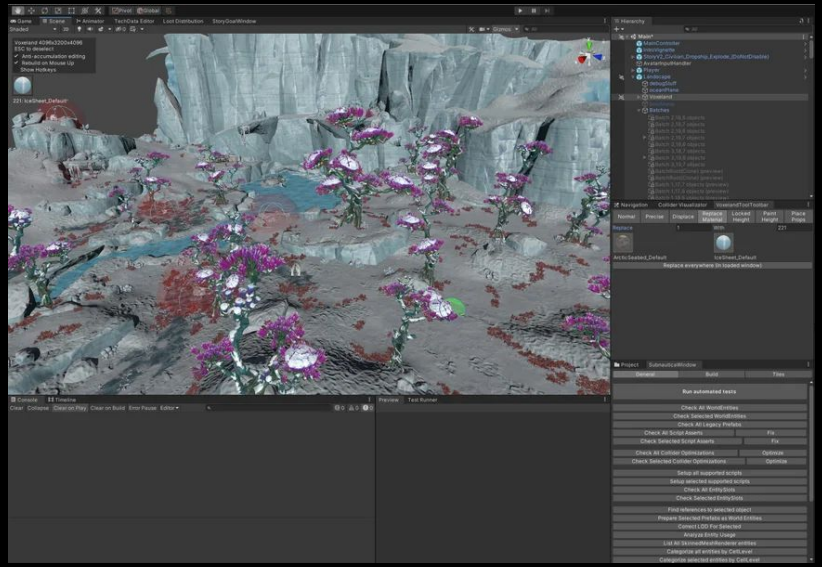
Fabrice Lété - Unity Technologies

Intro


Video Games Development
Video Games Middleware
Video Games Education

18 years

Unity®



Deliver incredible possibilities

Create once, deploy across 25+ leading platforms and technologies to reach the largest possible audience.

iOS

android

WebGL

PS4

PS5

XBOX ONE

SERIES XS

androidtv

tvOS

ARCore

Microsoft HoloLens

magic leap

Our impact by the numbers

In 2021

5B

downloads per month of apps built with Unity

72%

of the top 1,000 mobile games were made with Unity

50%+

of games across mobile, PC, and console were made with Unity

3.9B

monthly active users who consumed content created or operated with Unity solutions

20+

different platforms run Unity creations

Our impact by the numbers

In 2021

5B

downloads per month of apps built with Unity

72%

of the top 1,000 mobile games were made with Unity

50%+

of games across mobile, PC, and console were made with Unity

3.9B

monthly active users who consumed content created or operated with Unity solutions

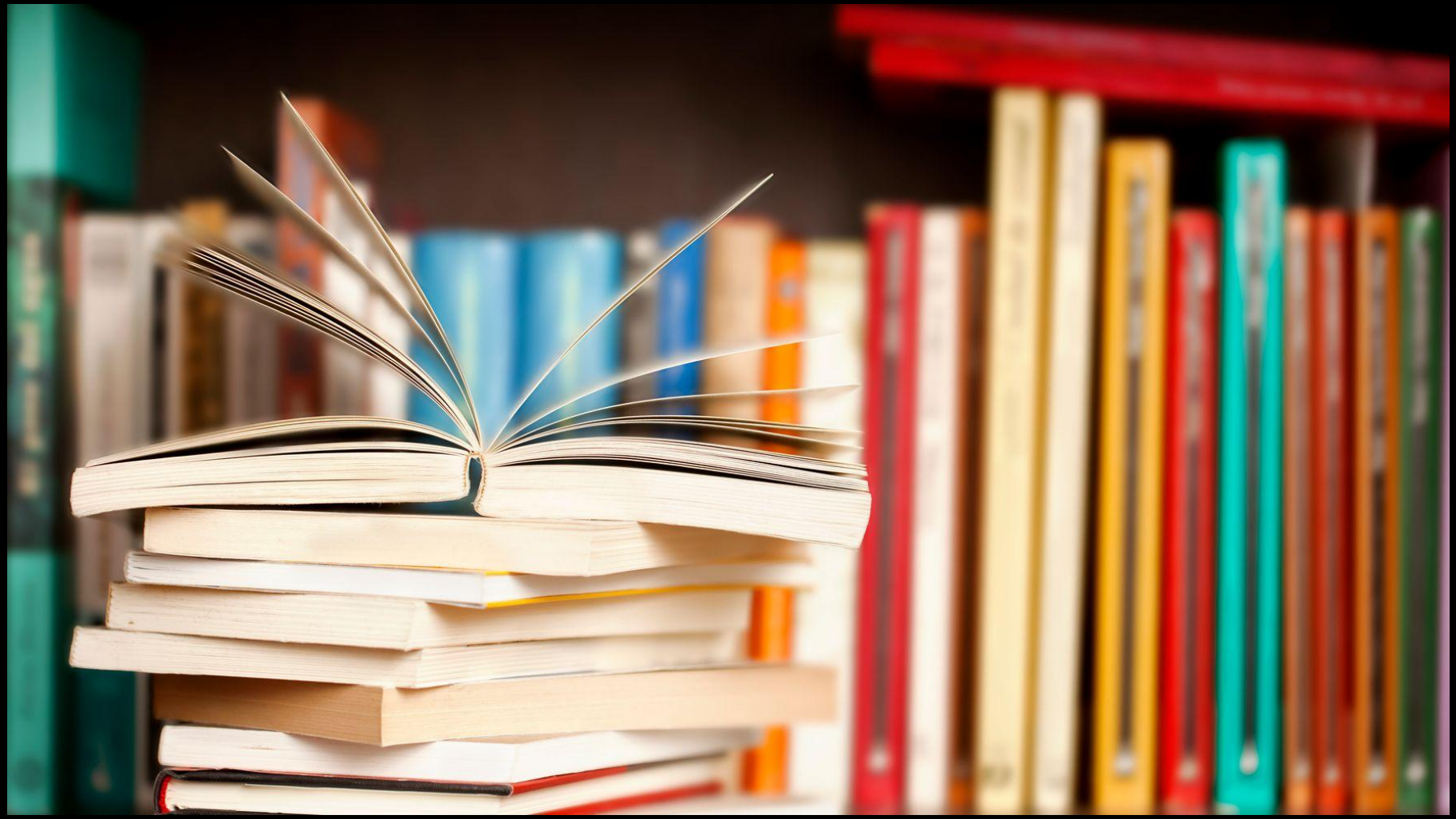
20+

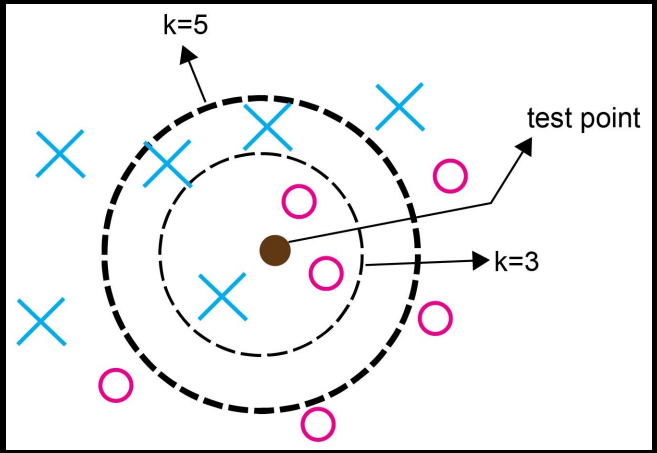
different platforms run Unity creations

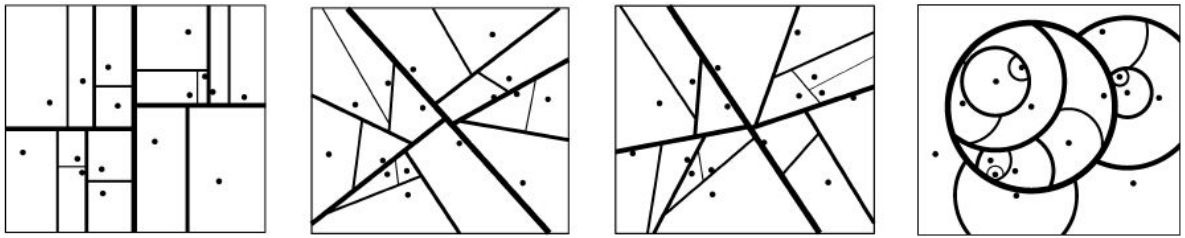
6,874

Revenue: 1.11 billion USD (Fiscal Year Ended 31 December 2021)

Theory and practice







(a) kd-Tree (b) PCA Tree (c) Ball Tree (d) vp-Tree

Fig. 2: The partitioning of a 2D point set using different types of nearest neighbor trees, all with a maximum leaf size of 1 and a branching factor of 2. Line thickness denotes partition order (thicker lines were partitioned first). Note the very different structures created by the methods, which result in very different search speeds.

- kd-tree
- Sort
- “iterative improvement” aka. my dumb idea
- Something else ???

stackoverflow About Products For Teams Search...

Home

PUBLIC

Finding the first n largest elements in an array

Asked 10 years, 6 months ago Active 3 years, 6 months ago Viewed 44k times

stackoverflow About Products For Teams Search...

Home

PUBLIC

Finding the first n largest elements in an array

Asked 10 years, 6 months ago Active 3 years, 6 months ago Viewed 44k times

35

Find the kth biggest element, using selection algorithm.
Next, iterate the array and find all elements which are larger/equal it.

complexity: O(n) for selection and O(n) for iterating, so the total is also O(n)

Share Follow

edited Sep 1, 2011 at 15:46

answered Sep 1, 2011 at 15:35

amit
171k • 25 • 222 • 321

stackoverflow About Products For Teams Search...

Home

PUBLIC

Finding the first n largest elements in an array

Asked 10 years, 6 months ago Active 3 years, 6 months ago Viewed 44k times

35

Find the kth biggest element, using selection algorithm.
Next, iterate the array and find all elements which are larger/equal it.

complexity: O(n) for selection and O(n) for iterating, so the total is also O(n)

Share Follow edited Sep 1, 2011 at 15:46 answered Sep 1, 2011 at 15:35

amit
171k • 25 • 222 • 321

WIKIPEDIA
The Free Encyclopedia

Article Talk

Quickselect

From Wikipedia, the free encyclopedia

stackoverflow About Products For Teams Search...

Home

PUBLIC

Finding the first n largest elements in an array

Asked 10 years, 6 months ago Active 3 years, 6 months ago Viewed 44k times

35

Find the kth biggest element, using selection algorithm.
Next, iterate the array and find all elements which are larger/equal it.

complexity: O(n) for selection and O(n) for iterating, so the total is also O(n)

Share Follow edited Sep 1, 2011 at 15:46 answered Sep 1, 2011 at 15:35

amit
171k • 25 • 222 • 321

WIKIPEDIA
The Free Encyclopedia

Article Talk

Quickselect

From Wikipedia, the free encyclopedia

Rosetta Code

Website

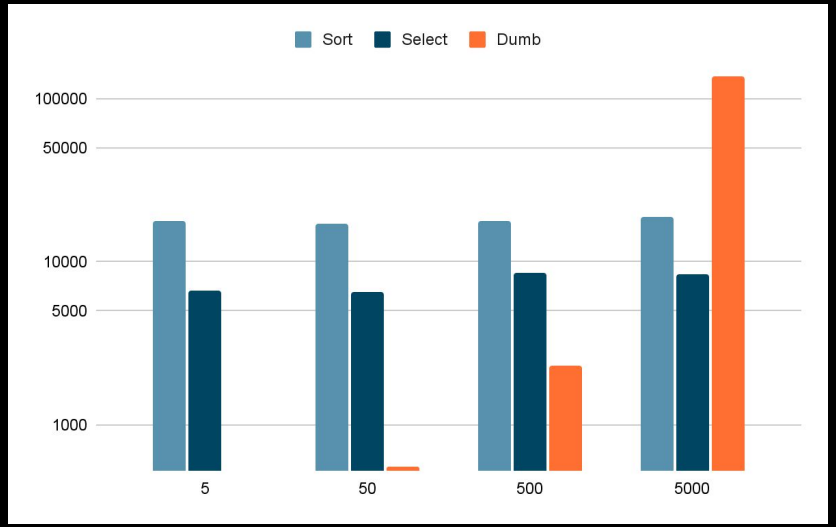
ROSETTACODE.ORG

Method	Count	Take	Mean
Sort	1000000	5	17,984.3 us
Sort	1000000	50	17,258.8 us
Sort	1000000	500	18,003.1 us
Sort	1000000	5000	19,007.7 us

Method	Count	Take	Mean
Sort	1000000	5	17,984.3 us
Select	1000000	5	6,639.3 us
Sort	1000000	50	17,258.8 us
Select	1000000	50	6,569.4 us
Sort	1000000	500	18,003.1 us
Select	1000000	500	8,532.6 us
Sort	1000000	5000	19,007.7 us
Select	1000000	5000	8,470.6 us

Method	Count	Take	Mean
Sort	1000000	5	17,984.3 us
Select	1000000	5	6,639.3 us
Dumb	1000000	5	522.0 us
Sort	1000000	50	17,258.8 us
Select	1000000	50	6,569.4 us
Dumb	1000000	50	553.8 us
Sort	1000000	500	18,003.1 us
Select	1000000	500	8,532.6 us
Dumb	1000000	500	2,313.0 us
Sort	1000000	5000	19,007.7 us
Select	1000000	5000	8,470.6 us
Dumb	1000000	5000	138,750.9 us

Method	Count	Take	Mean
Sort	1000000	5	17,984.3 us
Select	1000000	5	6,639.3 us
Dumb	1000000	5	522.0 us
Sort	1000000	50	17,258.8 us
Select	1000000	50	6,569.4 us
Dumb	1000000	50	553.8 us
Sort	1000000	500	18,003.1 us
Select	1000000	500	8,532.6 us
Dumb	1000000	500	2,313.0 us
Sort	1000000	5000	19,007.7 us
Select	1000000	5000	8,470.6 us
Dumb	1000000	5000	138,750.9 us



ParticleToParticleSearch (20,000 positions, 20,000 queries)

	Burst Safety OFF			Burst Safety ON		
	Building	Querying	Total	Building	Querying	Total
KDTree	0.62ms	9.26ms	**9.88ms**	0.708ms	10.64ms	**11.348ms**
KNN	1.4ms	29.05ms	**30.45ms**	8.21ms	58.85ms	**67.06ms**
On^2	0ms	68.624ms	**68.624ms**	0ms	384.0ms	**384.0ms**

Random Access Memory

Sequential Access

Maurice Wilkes inspecting the mercury delay line of the EDSAC in construction


Tombow ABT 837

TI-1200 (1975)

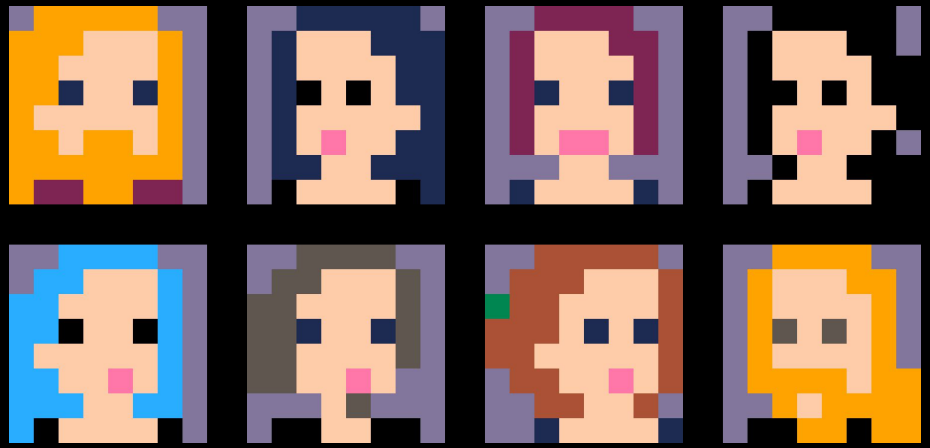
ROM 3520 bits
RAM 182 bits

Register A	
Register B	
Register C	
Flags 0	Flags 1

Atari 2600 (1982) - 128 bytes of RAM

Pitfall! by Activision (1982) - 4k ROM

Source:
https://benfry.com/distellamap/



8 x 8 pixels = 64 pixels

64 x 4 bits = 32 bytes

https://metakiki.net/pixelart-portraits/

```
livesPat            .byte   ;           number of lives, stored as displayed pattern ($a0 = 3, $80 = 2, $00 = 1)
random              .byte   ;           all scenes are generated randomly with this
random2             .byte   ;           used for random object animation
joystick            .byte   ;           stores joystick directions
fireButton          .byte   ;           stores fire button state
hitLiana            .byte   ;           Harry collided with liana? (bit 6 = 1 -> yes)
cxHarry             .byte   ;           Harry's collisions (stored but _never_ read!)
colorLst            ds 9    ;           some (mostly constant!?) colors
lianaBottom         .byte   ;           bottom row of liana
objectType          .byte   ;           type of the objects on the ground (hazards & treasures)
sceneType           .byte   ;           type of the scene (0..7
HMFineLst           ds 3    ;           fine positioning value for: Harry, ground-object, underground-object
HMCoarseLst         ds 3    ;           coars positioning value for: Harry, ground-object, underground-object
posLeftBranch       .byte   ;           values for positioning left branch graphics
posRightBranch      .byte   ;           values for positioning right branch graphics
ladderFlag          .byte   ;           0 = no ladder, $ff = with ladder
noGameScroll        .byte   ;           0 = game is running
PF2QuickSand        .byte   ;           PF2 data for top quicksand row
PF2Lst              ds 7    ;           copied pit pattern data
objColLst           ds 7    ;           copied object colors
objPatLst           ds 7    ;           copied object patterns
harryPatPtr         .word   ; = $b5     pointer to Pitfall Harry patterns
objPatPtr           .word   ;           pointer to object (hazards, treasure) patterns
harryColPtr         .word   ;           pointer to Pitfall Harry colors
objColPtr           .word   ;           pointer to object (hazards, treasure) colors
wallPatPtr          .word   ;           pointer to wall patterns
wallColPtr          .word   ;           pointer to wall colors
undrPatPtr          .word   ;           pointer to underground object (wall, scorpion) patterns
undrColPtr          .word   ;           pointer to underground object (wall, scorpion) colors
digitPtr            ds.w 6  ;           pointers for score display
```


```
ShowScore:

00 load digit0, '0'
01 load digit1, '0'
02 load digit2, '1'
03 load digit3, '8'
04 load digit4, '9'
05 load digit5, '9'
06 jump ShowDigits1

ShowTimer:

07 load digit0, '0'
08 load digit1, '1'
09 load digit2, '9'
10 load digit3, ':'
11 load digit4, '3'
12 load digit5, '8'
13 jump ShowDigits2

RestOfTheGame:

...
```

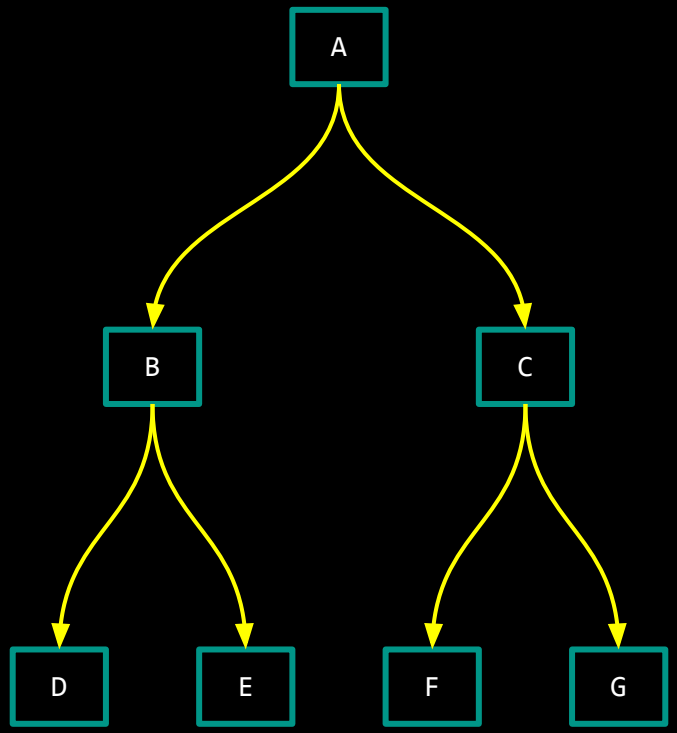
```
ShowDigits1:
...
jump ShowTimer
```

```
ShowDigits2:
...
jump RestOfTheGame
```


```
ShowScore:

00 load digit0, '0'
01 load digit1, '0'
02 load digit2, '1'
03 load digit3, '8'
04 load digit4, '9'
05 load digit5, '9'
06 jump ShowDigits1

ShowTimer:

07 load digit0, '0'
08 load digit1, '1'
09 load digit2, '9'
10 load digit3, ':'
11 load digit4, '3'
12 load digit5, '8'
13 jump ShowDigits2

RestOfTheGame:

...
```

```
ReturnAddress

ShowDigits:
...
jump ReturnAddress
```

```
Program Flow        Stack

exec A
call B push C       C
exec B              C
call D push E       CE
exec D              CE
exec E              C
exec C
call F push G       G
exec F              G
exec G
```



```
jumpIndex           .byte   ;           index of jump-table (0..32)
oldJoystick         .byte   ;           saved old joystick direction
yPosHarry           .byte   ;           y-position of Pitfall Harry
atLiana             .byte   ;           Harry at liana? (0 = no, -1 = yes)
treePat             .byte   ;           id of the leaves pattern (0..3)
climbPos            .byte   ;           position of Harry at ladder (0/11..22)
treasureBits        ds 4    ;           remember which treasures haven't been found
treasureCnt         .byte   ; = $f1     number of remaining treasures-1
patOfsHarry         .byte   ;           pattern offset (5 while kneeing, 0 else)
soundDelay          .byte   ;           play a new note every 4th frame
xPosQuickSand       .byte   ;           border of quicksand
jumpMode            .byte   ; = $f5     similar to jumpIndex (JTZ: superfluous?)
temp1               .byte
temp2               .byte
temp3               .byte
 .. .. .. .. .. .. .. .. .. ; = $f9
 .. .. .. .. .. .. .. .. .. ; = $fa
 .. .. .. .. .. .. .. .. .. ; = $fb
 .. .. .. .. .. .. .. .. .. ; = $fc
 .. .. .. .. .. .. .. .. .. ; = $fd
 .. .. .. .. .. .. .. .. .. ; = $fe
 .. .. .. .. .. .. .. .. .. ; = $ff
```

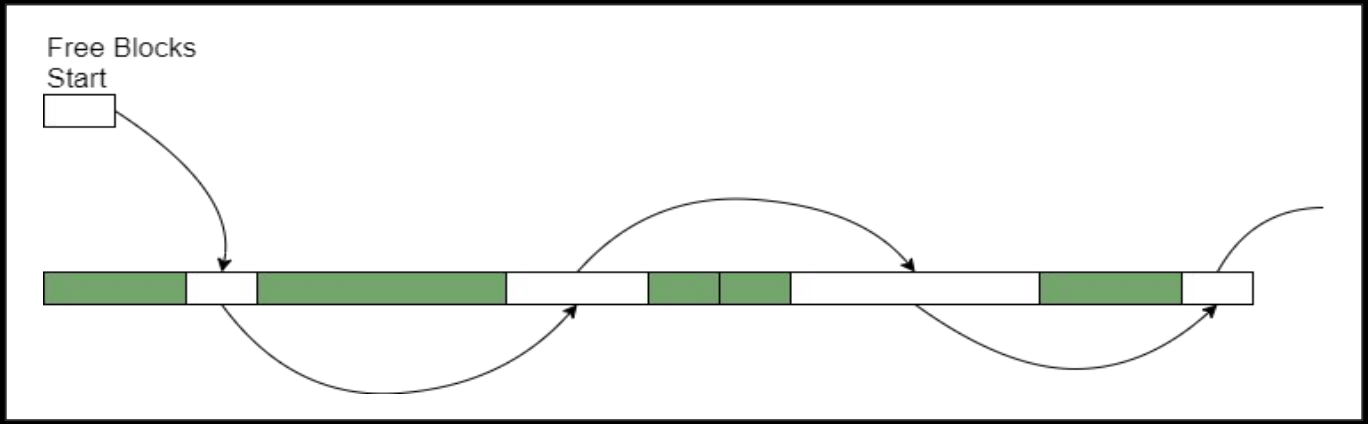
```
int[] array;
int result;
int i;

void sum()
{
    for(i = 0; i < array.Length; i++)
    {
        result += array[i];
    }
}
```

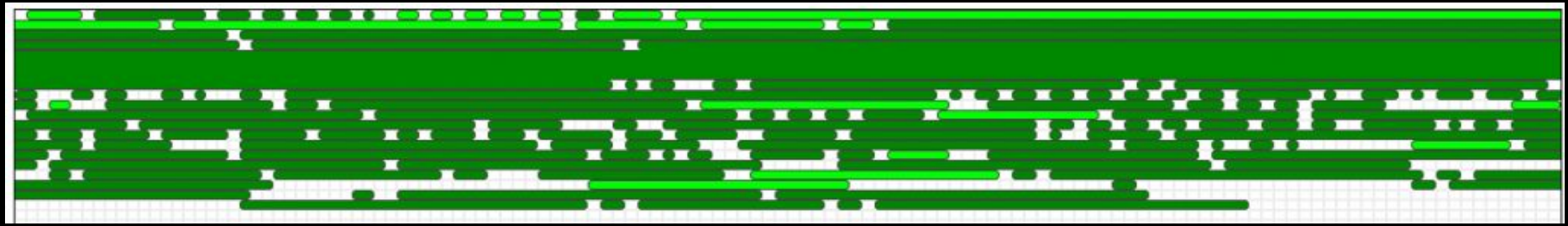
```
int sum(int[] array)                                    return address, array address
{
    int result = 0;                                     return address, array address, result
    for(int i = 0; i < array.Length; i++)               return address, array address, result, i
    {
        result += array[i];
    }
    return result;                                      return value
}
```


Clock
Reversi
Game Skill
MS-DOS Executive
File View Special
A C D
C: \WINDOWS
ABC.T
BUILD
CALC.
CALEN
CARDF
CGA.D
CGA.G
CGA.L
CITOH
CLIPB
CLOCK
COMM.
CONTROL.EXE
COURA.FON
COURB.FON
COURC.FON
EGAMONO.GRB
EGAMONO.LGO
EMM.AT
EMM.PC
HPLA
IBMG
JOYN
KERN
Microsoft Windows
MS-DOS Executive
Version 1.01
Copyright © 1985, Microsoft Corp.
Ok
Disk Space Free: 30024K
Memory Free: 303K
Write - README.DOC
File Edit Search
Character Paragraph
Document
Page 1

```
public class Game
{
    string m_ScoreText;

    public void SetScore(int score)
    {
        m_ScoreText = score.ToString();
    }
}
```

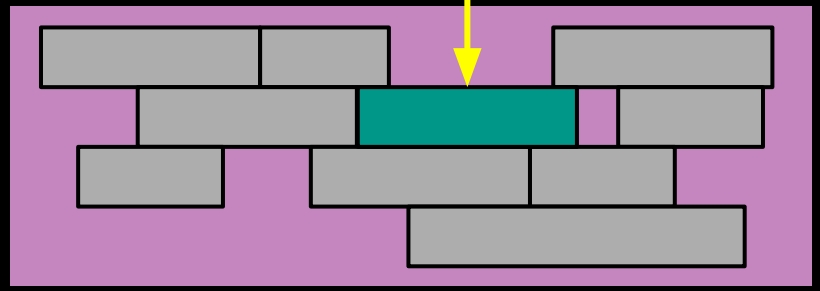
Dynamic Allocation


Free Blocks
Start



Memory Issues

- Fragmentation

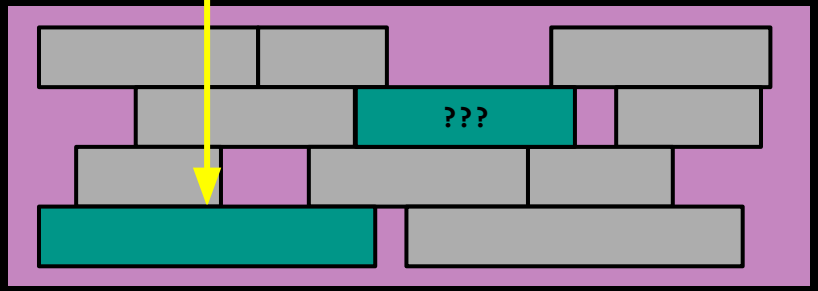


```
int* x = new int[20];
```

Memory Issues

- Fragmentation
- Leaks

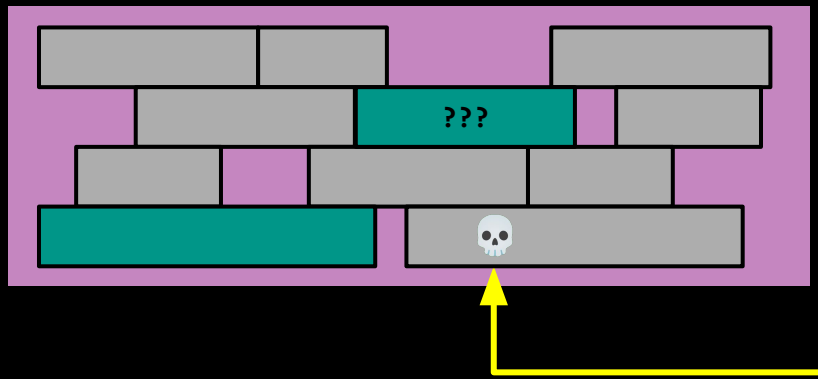
```
int* x = new int[20];

x = new int[30];
```

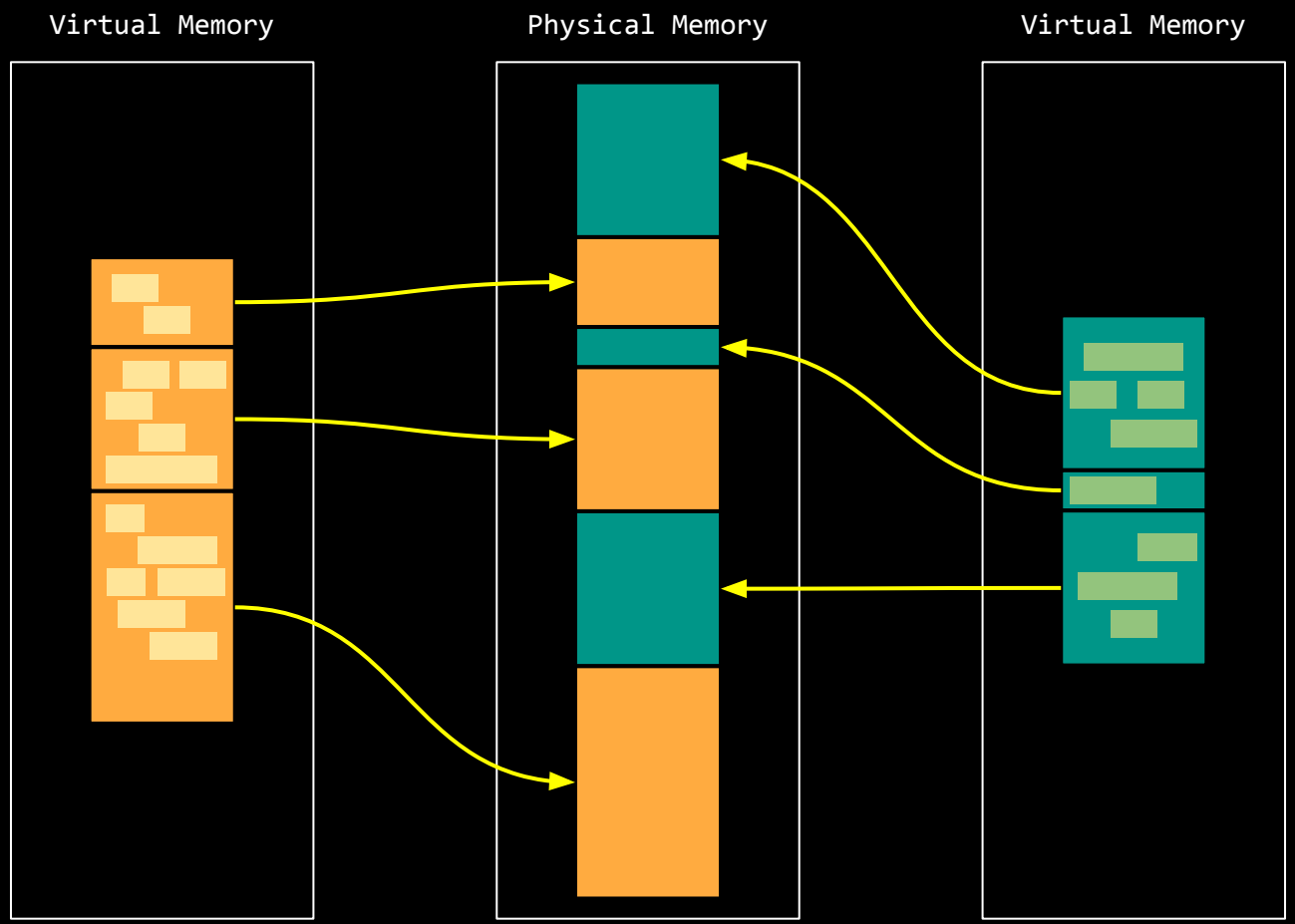


Memory Issues

- Fragmentation
- Leaks
- Invalid Access

```
int* x = new int[20];

x = new int[30];

x[40] = 123;
```




Virtual Memory
Physical Memory
Virtual Memory

Invalid Access

Leaks

Fragmentation

Invalid Access

NullReferenceException
ArgumentNullException
IndexOutOfRangeException
ArgumentOutOfRangeException

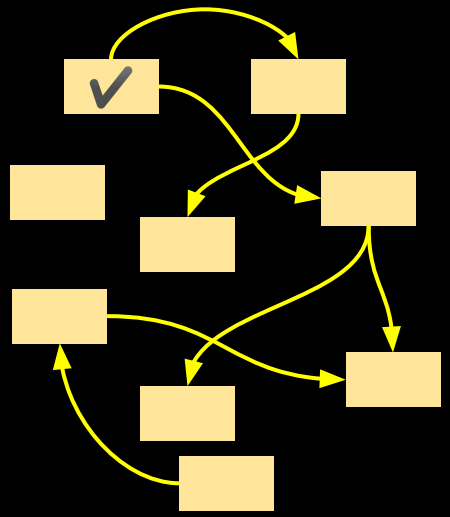
Leaks

Fragmentation

Invalid Access

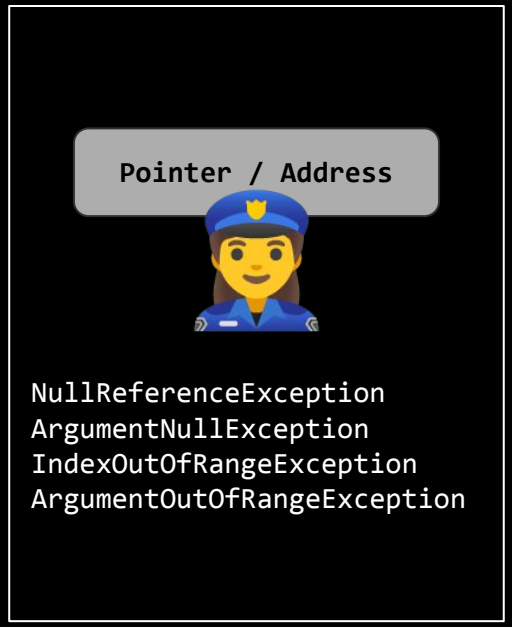
NullReferenceException
ArgumentNullException
IndexOutOfRangeException
ArgumentOutOfRangeException

Leaks

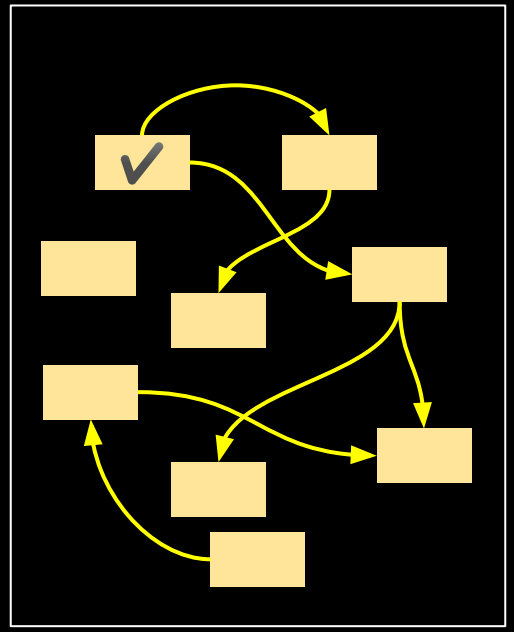


Fragmentation

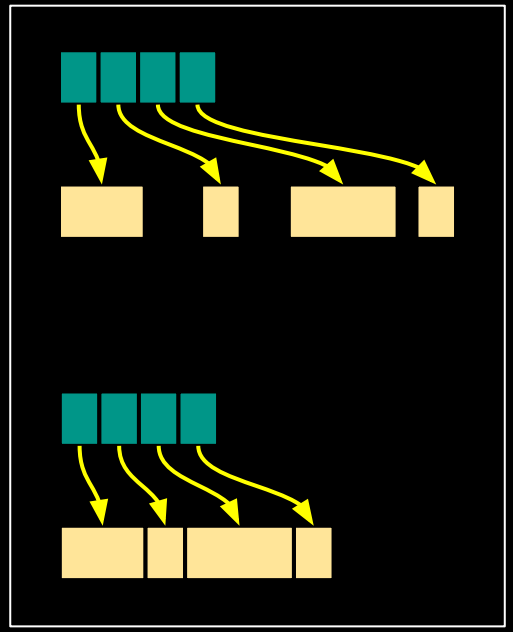
Invalid Access



Leaks



Fragmentation

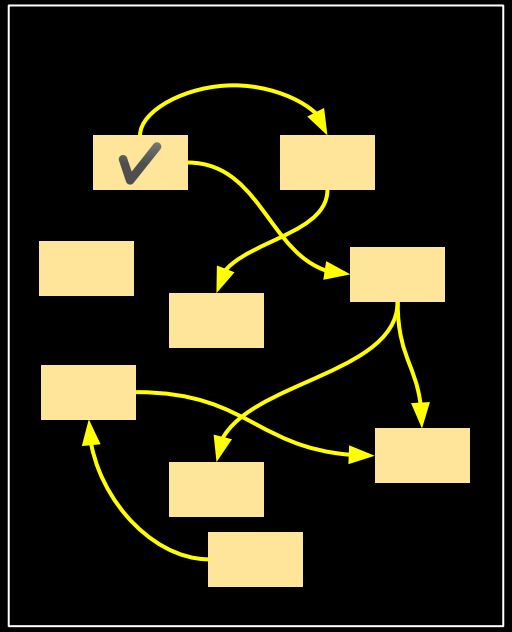


Invalid Access

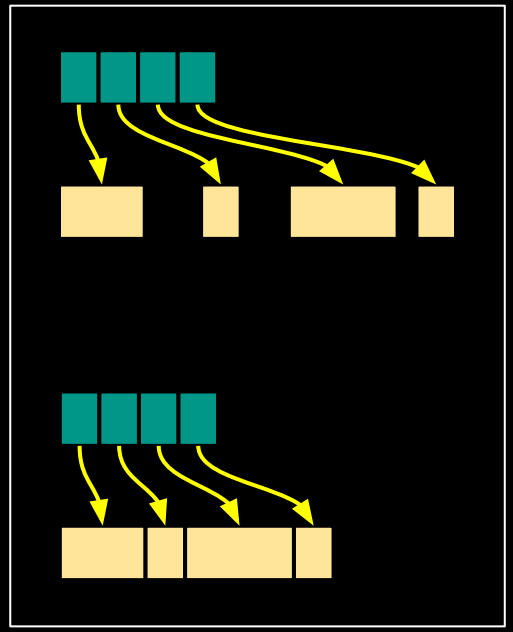
NullReferenceException
ArgumentNullException
IndexOutOfRangeException
ArgumentOutOfRangeException

Language

Leaks



Fragmentation



Compacting Garbage Collector

Optimization

```
for (int x = 0; x < Src.Width; ++x)
{
    for (int y = 0; y < Src.Height; ++y)
    {
        var pxl = Src.GetPixel(x, y);
        Dst.SetPixel(x, y, pxl);
        Dst.SetPixel(x + Src.Width, y, pxl);
        Dst.SetPixel(x, y + Src.Height, pxl);
        Dst.SetPixel(x + Src.Width, y + Src.Height, pxl);
    }
}
```

Method	Mean
ConvertA	3,519.644 ms

```
var srcPixels = new byte[Src.Width * Src.Height * 3]; var dstPixels = new byte[Dst.Width * Dst.Height * 3];

Marshal.Copy(srcLock.Scan0, srcPixels, 0, srcPixels.Length);
Src.UnlockBits(srcLock);

for (int x = 0; x < Src.Width; ++x)
{
    for (int y = 0; y < Src.Height; ++y)
    {
        var srcIdx = (x + y * Src.Width) * 3;
        var pxl_r = srcPixels[srcIdx + 0];
        var pxl_g = srcPixels[srcIdx + 1];
        var pxl_b = srcPixels[srcIdx + 2];

        {
            var dstIdx = (x + y * Dst.Width) * 3;
            dstPixels[dstIdx + 0] = pxl_r;
            dstPixels[dstIdx + 1] = pxl_g;
            dstPixels[dstIdx + 2] = pxl_b;
        }
        {
            var dstIdx = (x + Src.Width + y * Dst.Width) * 3;
```

1. Remove abstractions

Method	Mean
ConvertA	3,519.644 ms
ConvertB	1,234.643 ms

```
var srcRect = new Rectangle(0, 0, Src.Width, Src.Height);
var dstRect = new Rectangle(0, 0, Dst.Width, Dst.Height);

var srcLock = Src.LockBits(srcRect, ImageLockMode.ReadOnly, Src.PixelFormat);
var dstLock = Dst.LockBits(dstRect, ImageLockMode.WriteOnly, Dst.PixelFormat);

var srcPixels = new byte[Src.Width * Src.Height * 3]; var dstPixels = new byte[Dst.Width * Dst.Height * 3];

Marshal.Copy(srcLock.Scan0, srcPixels, 0, srcPixels.Length);
Src.UnlockBits(srcLock);

for (int y = 0; y < Src.Height; ++y)
{
    var srcIdx = y * Src.Width * 3;
    Array.Copy(srcPixels, srcIdx, dstPixels, y * Dst.Width * 3, Src.Width * 3);
    Array.Copy(srcPixels, srcIdx, dstPixels, (Src.Width + y * Dst.Width) * 3, Src.Width * 3);
    Array.Copy(srcPixels, srcIdx, dstPixels, (y + Src.Height) * Dst.Width * 3, Src.Width * 3);
    Array.Copy(srcPixels, srcIdx, dstPixels, (Src.Width + (y + Src.Height) * Dst.Width) * 3, Src.Width * 3);
}

Marshal.Copy(dstPixels, 0, dstLock.Scan0, dstPixels.Length);
Dst.UnlockBits(dstLock);
```

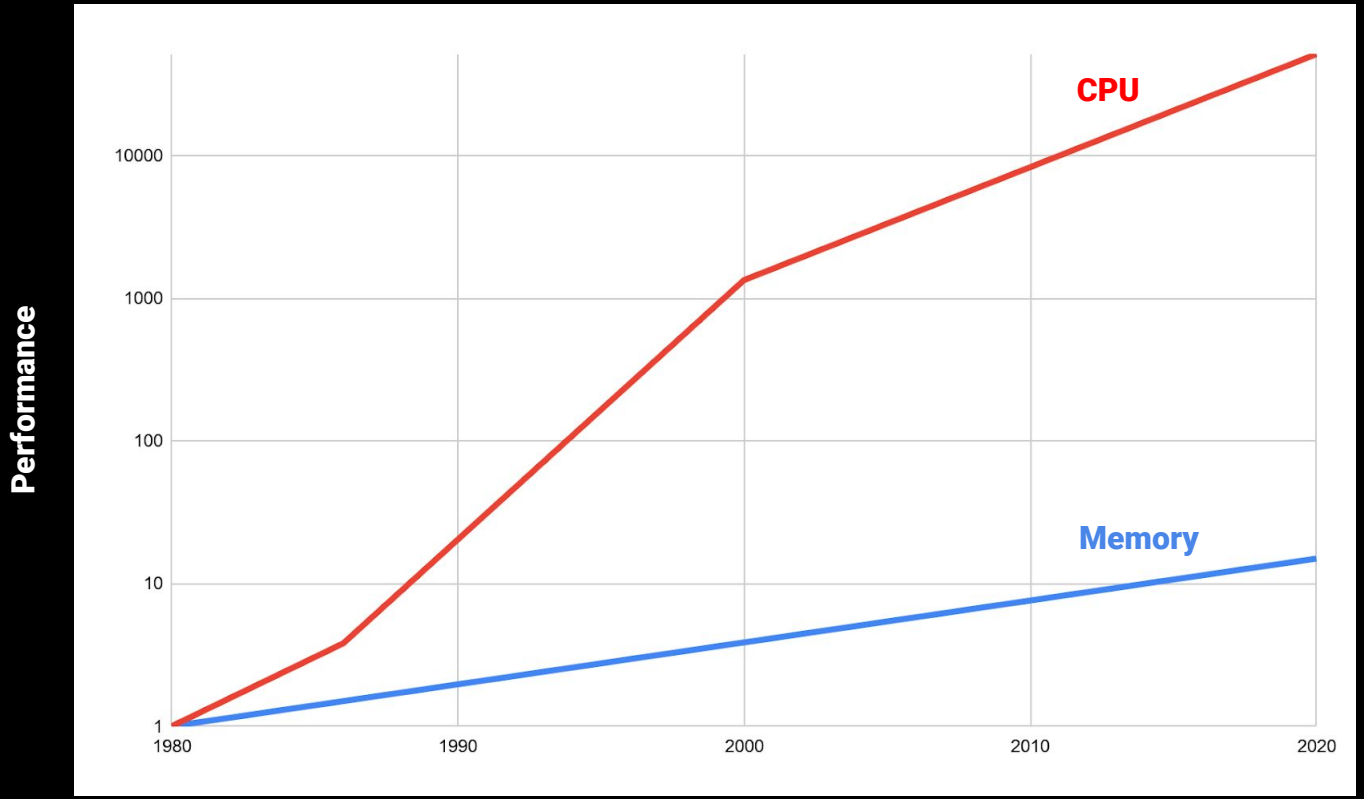
1. Remove abstractions
2. Do things in batches

Method	Mean
ConvertA	3,519.644 ms
ConvertB	1,234.643 ms
ConvertC	27.034 ms

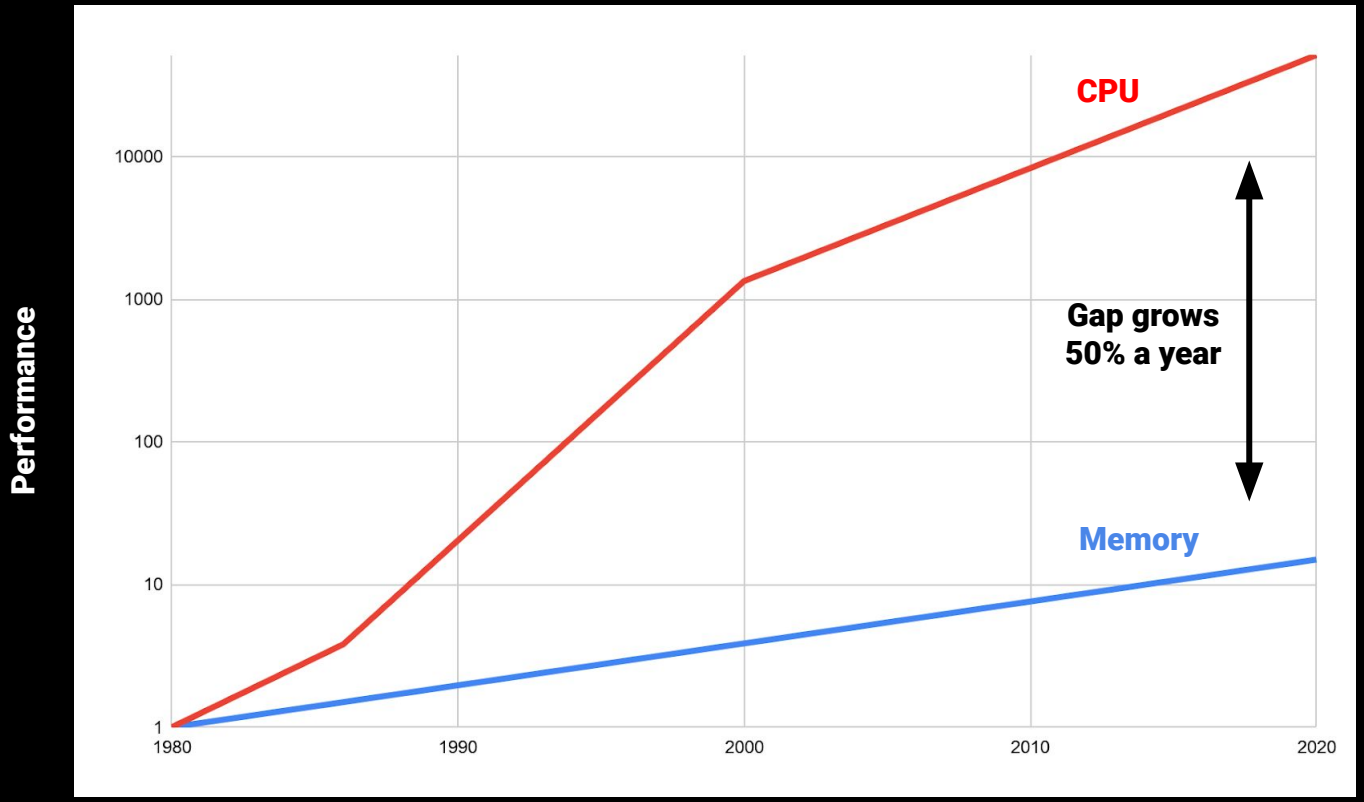
```
var srcRect = new Rectangle(0, 0, Src.Width, Src.Height);
var dstRect = new Rectangle(0, 0, Dst.Width, Dst.Height);

var srcLock = Src.LockBits(srcRect, ImageLockMode.ReadOnly, Src.PixelFormat);
var dstLock = Dst.LockBits(dstRect, ImageLockMode.WriteOnly, Dst.PixelFormat);

var row = new byte[Src.Width * 3];

for (int y = 0; y < Src.Height; ++y)
{
    var srcIdx = y * Src.Width * 3;
    Marshal.Copy(srcLock.Scan0 + srcIdx, row, 0, Src.Width * 3);
    Marshal.Copy(row, 0, dstLock.Scan0 + y * Dst.Width * 3, Src.Width * 3);
    Marshal.Copy(row, 0, dstLock.Scan0 + (Src.Width + y * Dst.Width) * 3, Src.Width * 3);
    Marshal.Copy(row, 0, dstLock.Scan0 + (y + Src.Height) * Dst.Width * 3, Src.Width * 3);
    Marshal.Copy(row, 0, dstLock.Scan0 + (Src.Width + (y + Src.Height) * Dst.Width) * 3, Src.Width * 3);
}

Src.UnlockBits(srcLock);
Dst.UnlockBits(dstLock);
```

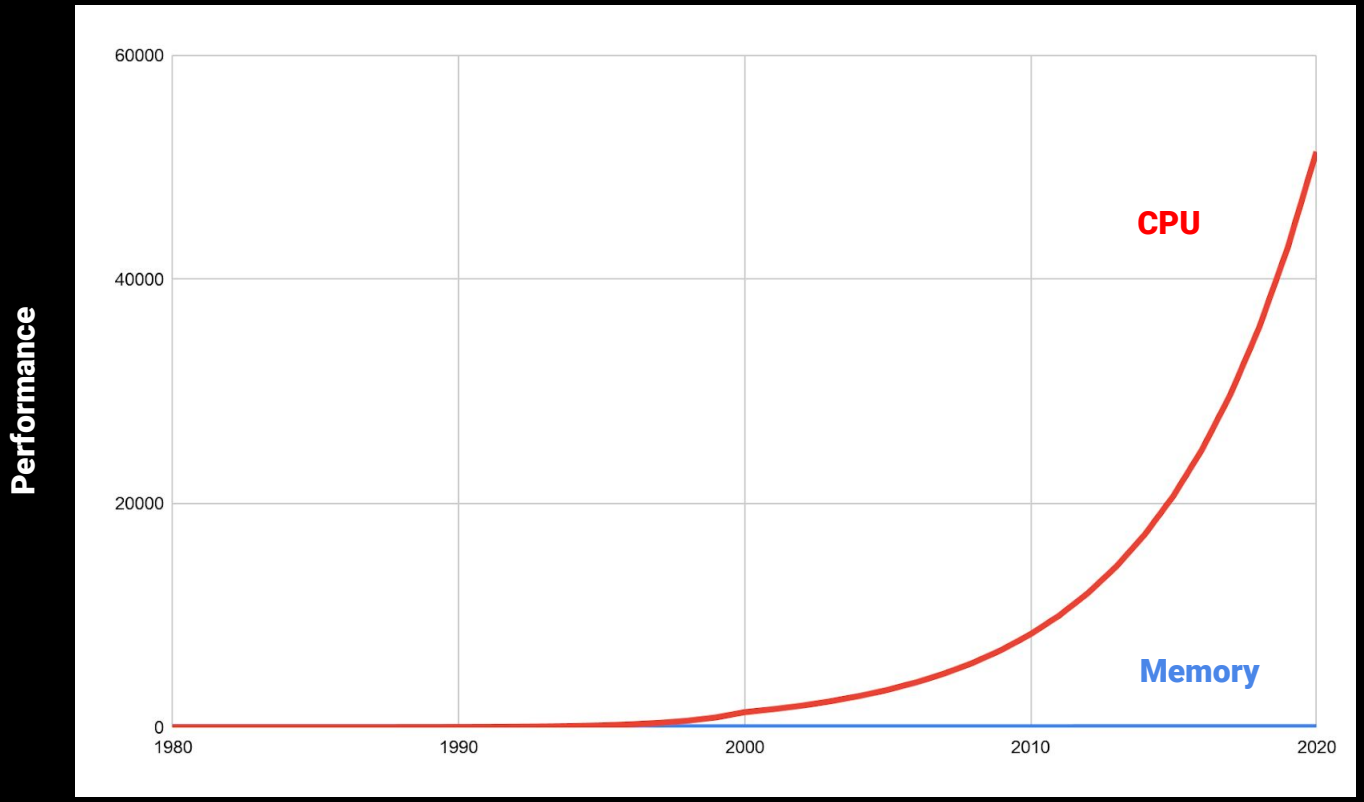
1. Remove abstractions
2. Do things in batches
3. Reduce movement

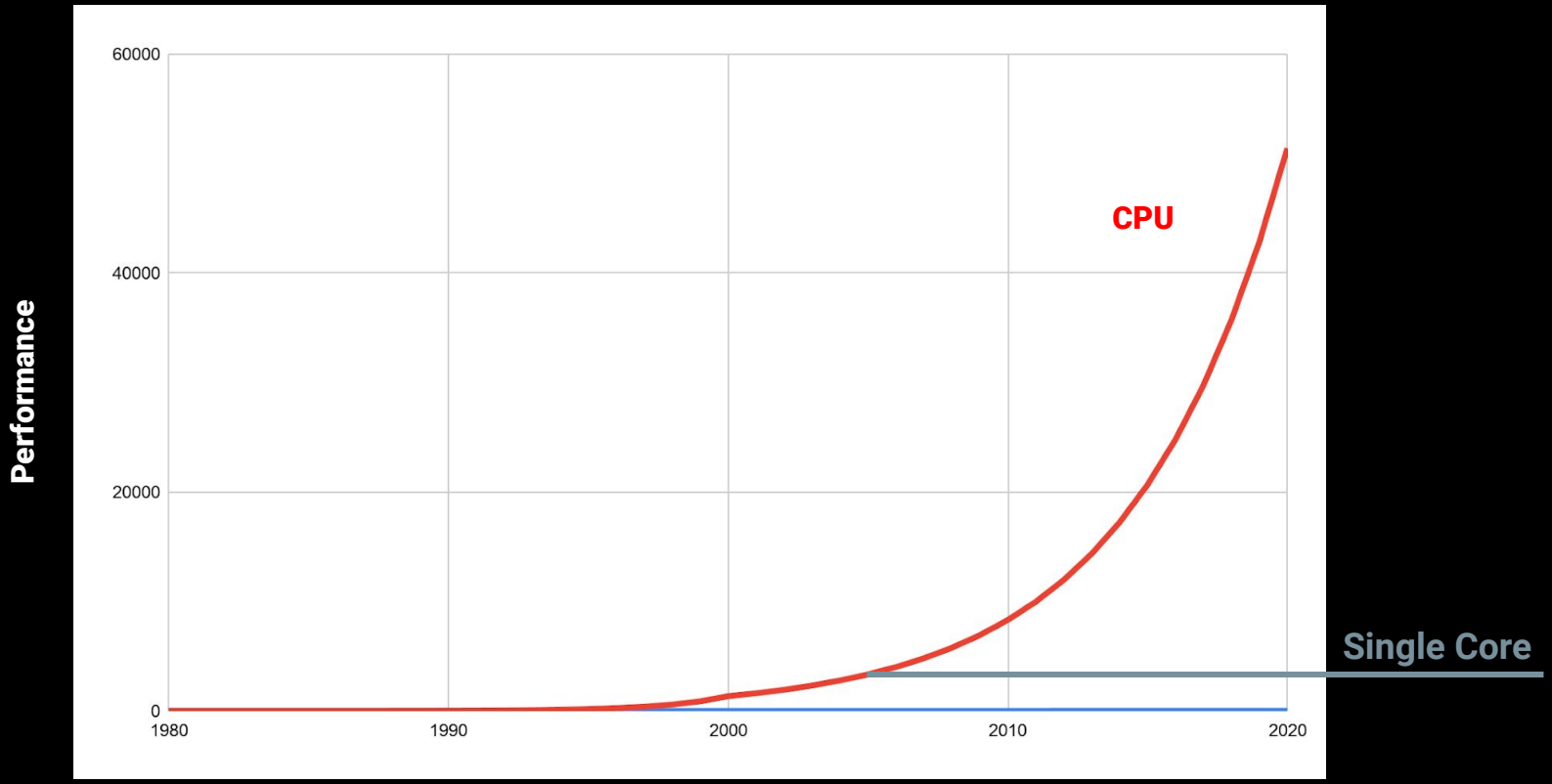
Method	Mean
ConvertA	3,519.644 ms
ConvertB	1,234.643 ms
ConvertC	27.034 ms
ConvertD	3.142 ms



Based on data from: Hennessy, Patterson: Computer Architecture: A Quantitative Approach, 5th Edition.


Performance
10000
1000
100
10
1
1980
1990
2000
2010
2020
CPU
Memory
Gap grows
50% a year


Performance
60000
40000
20000
0
1980
1990
2000
2010
2020
CPU
Memory

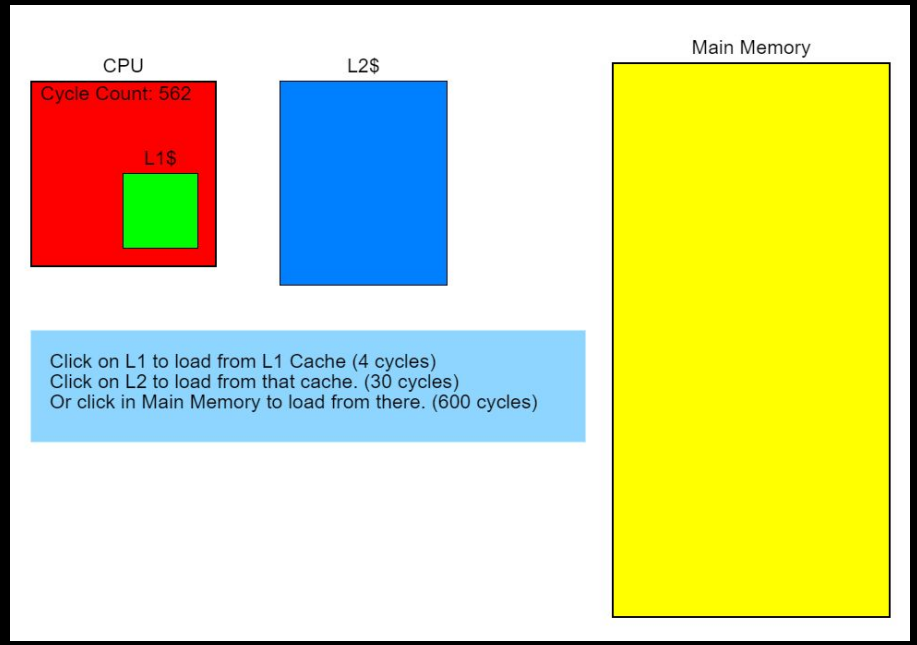

Performance
60000
40000
20000
0
1980
1990
2000
2010
2020
CPU
Single Core

2,000 * 1,666 = 3,332,000 * 5 = 16,660,000 * 3 = 49,980,000

max throughput = 50 GB/s

Method	Mean
ConvertA	3,519.644 ms
ConvertB	1,234.643 ms
ConvertC	27.034 ms
ConvertD	3.142 ms

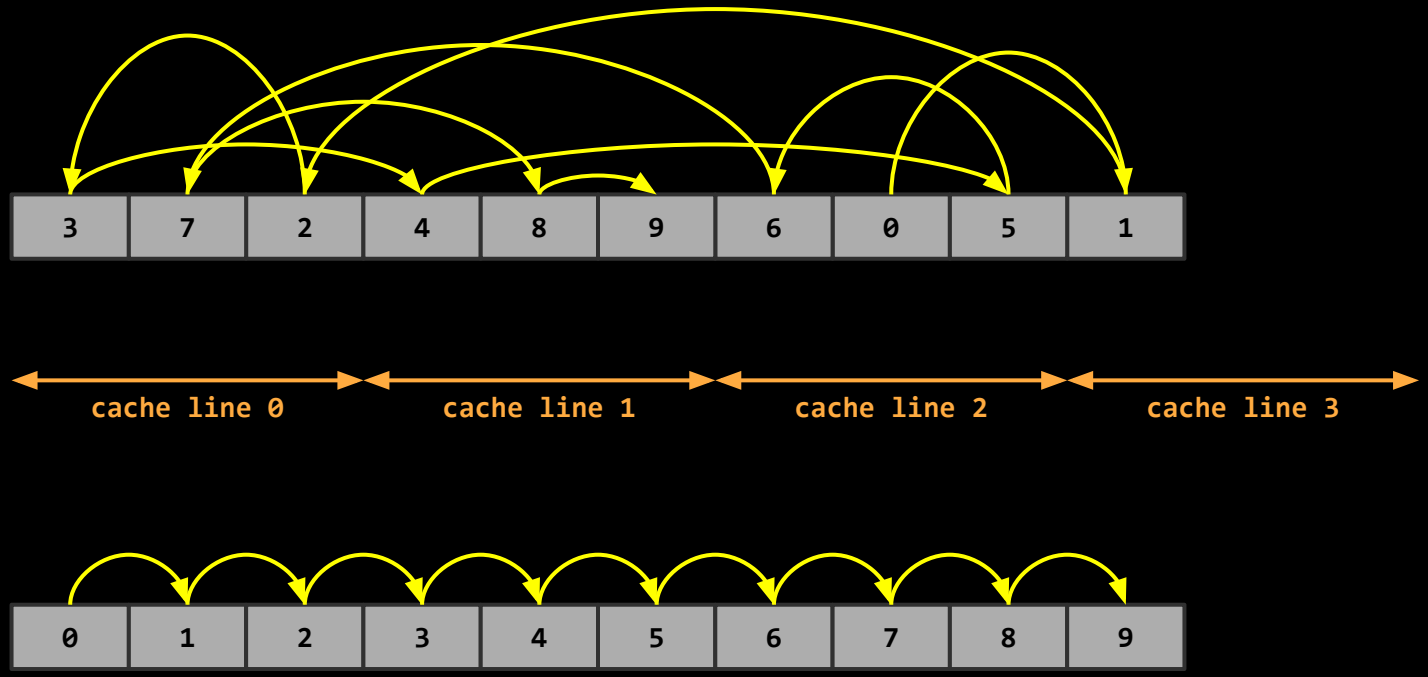
Book currently being read (~1)
Books on the table (~5)
Books on the shelves (~1,000)
Books in the building (~20,000)



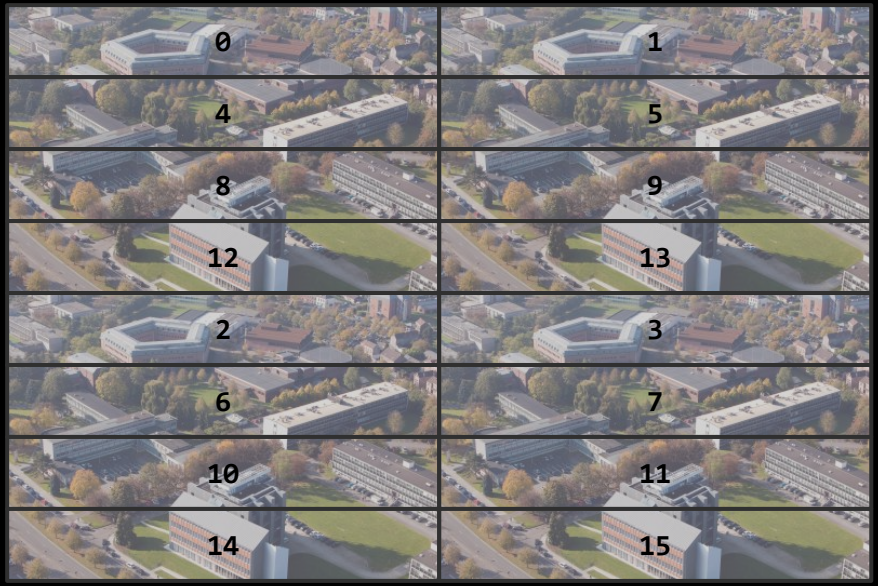
http://www.overbyte.com.au/misc/Lesson3/CacheFun.html

Cache Line Size

Cache Prefetching


3
7
2
4
8
9
6
0
5
1
cache line 0
cache line 1
cache line 2
cache line 3
0
1
2
3
4
5
6
7
8
9

Total Cache Size



Strip Mining

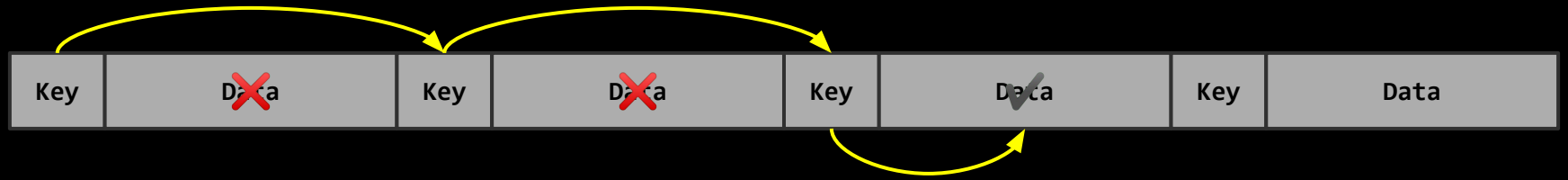
Hot / Cold

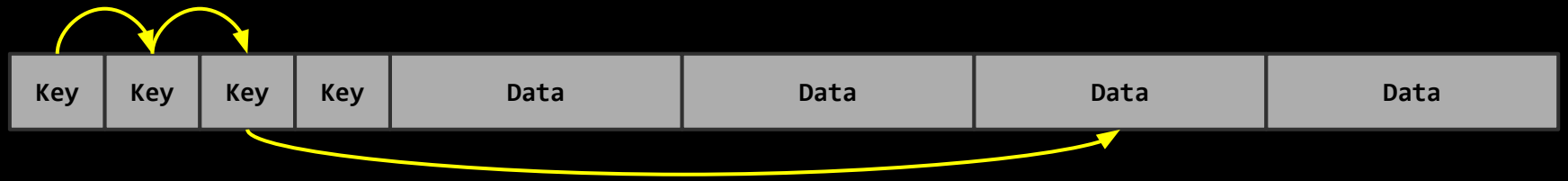
```
struct KeyValue
{
    public int Key;
    public Matrix4x4 Value;
}

KeyValue[] Data;
```

```
struct Data
{
    public int[] Keys;
    public Matrix4x4[] Values;
}
```

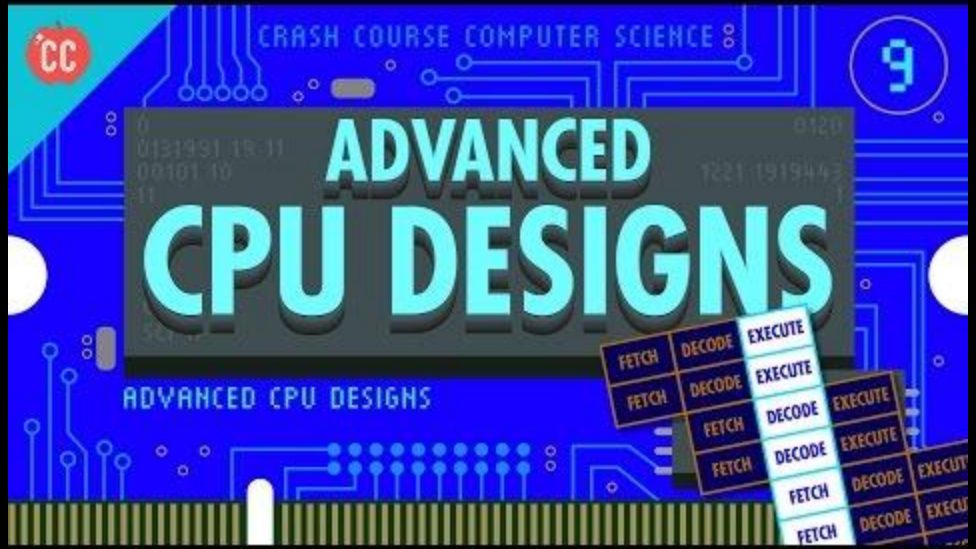
```
struct KeyValue
{
    public int Key;
    public Matrix4x4 Value;
}

KeyValue[] Data;
```

```
struct Data
{
    public int[] Keys;
    public Matrix4x4[] Values;
}
```





References



Advanced CPU Designs: Crash Course Computer Science #9

CppCon 2016: Nicholas Ormrod “The strange details of std::string at Facebook"

Handmade Seattle 2021 - Andreas Fredriksson - Context is Everything
https://vimeo.com/644068002

[EPILEPSY WARNING] How fast should an unoptimized terminal run?
Refterm v2 - Resource usage, binary splat, glyph sizing, and more
Refterm Lecture Part 1/2/3/4/5 - Philosophies of Optimization

- Build things
  - Stay motivated, stay focused
  - Reinvent the wheel
  - Raytracer / Emulator
  - Optimize

- Profile / Measure
  - Early and often
  - In context

- Learn tools
  - Languages
  - Debugger
  - Profiler
  - Version Control

- Teach
  - Research and experiment
  - Train communications skills
  - ~~Get rich~~

That’s all folks!

fabricel@unity3d.com